RL78 Development Environment Migration Guide

Migration between RL78 family (Compiler ed; Coding) (CA78K0R to CC-RL)

December 28, 2016
R20UT3416EJ0102

Software Business Division
Renesas System Design Co., Ltd.

Introduction

- This document describes the source code differences when migrating projects or source codes created for the CA78K0R C compiler for the RL78 family of MCUs to the CC-RL C compiler for the RL78 family of MCUs.
- This document describes the CA78K0R C compiler for the RL78 family of MCUs and the CC-RL C compiler for the RL78 family of MCUs.
  The applicable versions are as follows.
  - CA78K0R V1.20 and later
  - CC-RL V1.03.00

Agenda

Compiler Language Specifications

Differences in the language specifications(1/2)

Item	CA78K0R	CC-RL	Remarks
Language	C language	C language	
Language standard	C89	C90 and some functions of C99 are supported.	
Endian	little	little	
Usable multibyte characters	EUC, SJIS	EUC, SJIS, UTF-8, big5, gbk	
Range of support for multibyte characters	Japanese can be written in comments.	Japanese and Chinese can be written in comments and strings.	
Handling of char type not specified as signed or unsigned	Signed integer Unsigned integer when the -qu option is specified.	Unsigned integer Signed integer when the -signed_char option is specified.	
double type	Conforms to IEEE754-1985. 32-bit data	Conforms to IEEE754-1985. • When -dbl_size=4 is used 32-bit data • When -dbl_size=8 is used 64-bit data	-dbl_size=8 is available only for the RL78-S3 core.

Differences in the language specifications(2/2)

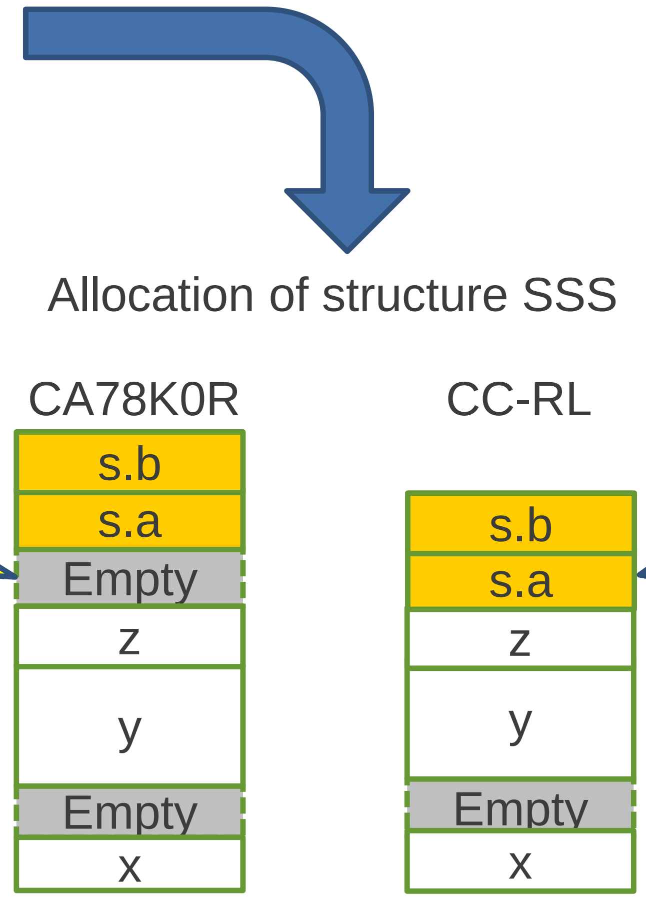
Item	CA78K0R	CC-RL	Remarks
int-type bit field in a structure or union specifier	Handled as unsigned.	Handled as unsigned. Becomes signed int type when -signed_bitfield is used.	
Allocation order of the bit field in a structure or union specifier	Allocated from lower to higher bits. Allocated from higher to lower bits when the -rb option is specified.	Allocated from lower to higher bits.	
Boundary for each member in a structure or union specifier	• 1-byte boundaries char/signed char/unsigned char • Others: 2-byte boundaries	• 1-byte boundaries char/signed char/unsigned char/_Bool • Others: 2-byte boundaries	
Enumeration specifier	The enumeration type becomes one of the following depending on the range of the enumeration constants. signed char/unsigned char/ signed int	The enumeration type becomes one of the following depending on the range of the enumeration constants. char/signed char/unsigned char/ signed short	

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the boundary for each member in a structure or union specifier

Definition of structure SSS

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



There is one empty byte because a structure is allocated to align with the word boundary.

A structure is allocated to satisfy the alignment condition of the largest member. Since alignment of the char type is 1-byte boundaries, there are no empty bytes.

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the enumeration specifier

The type of internal representation varies according to the range of the enumerator.

- For CA78K0R (priority)

  Range: -128 to 127 → signed char
  Range: 0 to 255 → unsigned char
  Range: -32768 to 32767 → signed int¥

- For CC-RL (priority)

  • With -signed_char
  Range: -128 to 127 (including the case of 0 to 127) → char
  Range: 0 to 255 → unsigned char
  Range: Other than above → signed short

  • Without -signed_char
  Range: -128 to 127 → signed char
  Range: 0 to 255 (including the case of 0 to 127) → char
  Range: Other than above → signed short

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in inclusion of header files

Item	CA78K0R	CC-RL	Remarks
Search order of the include <string> format	(1) Folder specified by the -i option (2) Folder specified by the environment variable INC78K0R (3) Folder containing the standard include files	(1) Folder specified by the -I option (2) Folder containing the standard include files	
Search order of the #include ”string“ format	(1) Folder containing the source files (2) Folder specified by the -i option (3) Folder specified by the environment variable INC78K0R (4) Folder containing the standard include files	(1) Folder containing the source files (2) Folder specified by the -I option (3) Folder containing the standard include files	

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the translation limits(1/2)

Item	CA78K0R	CC-RL
Nesting levels of files in an iteration statement, compound statement, and selection statement	45	Depends on memory
Nesting levels of conditional inclusion	255	Depends on memory
Number of pointers, arrays, and function declarators (any combination) that qualify one arithmetic type, structure type, union type, or incomplete type in one declarator	12	128
Nesting levels of declarators that are enclosed within parentheses in one full declarator	—	Depends on memory
Nesting levels of expressions that are enclosed within parentheses in one full expression	1024	Depends on memory
Number of effective starting characters in a macro name	256	Depends on memory
Number of effective starting characters in an internal identifier	249	Depends on memory
Number of effective starting characters in an external identifier	249	Depends on memory
Number of external identifiers in one translation unit	1024	Depends on memory
Number of identifiers that have a block scope for one block	255	Depends on memory
Number of macro identifiers that can be defined simultaneously in one translation unit	60000	Depends on memory
Number of formal parameters in one function definition	39	Depends on memory
Number of arguments in one function call	39	Depends on memory
Number of formal parameters in one macro definition	31	Depends on memory
Number of arguments in one macro call	31	Depends on memory

Differences in the translation limits(2/2)

Item	CA78K0R	CC-RL
Number of characters in one logical source line	32767	Depends on memory
Number of characters in a character string literal or wide string literal (after concatenation)	509	
Number of bytes of one object (in the host environment)	65535	32767 (65535 when the -large_variable option is specified)
Nesting levels of files that are included by #include	50	Depends on memory
Number of case labels in one switch statement (nested switch statements are excluded)	1024	
Number of members in one structure or union	1024	
Number of enumeration constants in one enumeration	255	
Nesting levels of structure or union definitions in one string of member declarations	15	

* The column of CA78K0R indicate values for V1.50 and later.

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the numerical limits(1/2)

Item	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)

* For CHAR_MIN and CHAR_MAX, the values enclosed within parentheses are effective when -signed_char is specified.
For other items, the values enclosed within parentheses are effective when the -dbl_size=8 option is specified (RL78-S3 core only).

Differences in the numerical limits(2/2)

Item	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)

* The values enclosed within parentheses are effective when the -dbl_size=8 option is specified (RL78-S3 core only).

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the #pragma directive

Item	CA78K0R	CC-RL	Actions
Enabling of data insert functions __OPC( )	#pragma opc	—	Delete the #pragma directive and write the data insert processing using #pragma inline_asm and assembly-language instructions.
Function call from boot area to flash memory area	#pragma ext_func	—	There is no relevant directive. Delete the #pragma directive. Specify an absolute address and call the function.
Specification of inline expansion of standard library functions memcpy( ) and memset( )	#pragma inline	—	Delete the #pragma directive. In CC-RL, this means inline expansion of a user-defined function.

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the macros

Macro Name in CA78K0R	Relevant Macro Name in CC-RL	Value
__K0R_LARGE__	None	—
CPU macro	None	—

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the keywords(1/2)

Function	Keyword	Relevant Keyword in CC-RL	Actions
near/far attribute	__near/__far	__near/__far	The specified position is different.
Declaration of bit variables for saddr area	__boolean boolean bit	—	Define and declare the bit fields of a structure and change the bit access processing.
asm statement	__asm #asm to #endasm	#pragma inline_asm	An assembly-language instruction cannot be directly written to a C source program using an asm statement. Define the assembly-language instruction part with an assembly-language function and use #pragma inline_asm.

Differences in the keywords(2/2)

Function	Keyword	Relevant Keyword in CC-RL	Actions
78K0-compatible	__callf callf	—	78K0-compatible keywords are not supported. Delete the relevant keywords.
	noauto	—	
	__leaf norec	—	
	__pascal	—	
	__temp	—	
	__mxcall	—	

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in declaration of bit variables for the saddr area

- CA78K0R

```
Format: bit (or boolean or __boolean ) [variable name]
```

- CC-RL
  - Since there are no bit variables, bit fields are defined in a structure.

```
Format: __saddr struct [tag name] {
                [type name] [field name]: [bit width];
                [type name] [field name]: [bit width];
                …
                [type name] [field name]: [bit width];
                };
```

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

char-type 1-bit numerical value

  - The following types can be used for the types of bit field members.
    char, signed char, unsigned char, signed short, unsigned short, signed int, unsigned int, signed long, unsigned long, signed long long, unsigned long long

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the assembly-language instruction descriptions

- CA78K0R

```
Format: #asm
           –assembly-language description–
        #endasm
```

- CC-RL

```
Format: #pragma inline_asm [(] function name [,...][)]
```

```
#pragma inline_asm func

static int func() {
  /* assembly-language description */
}
```

The func function for assembly-language description is declared.
The assembly-language source program is written in the func function

Refer to the user's manual for the compiler for more information and make changes as required.

Assembly Language Specifications

Differences in the macro operators and the operators

- Differences in the macro operators

Operation Type	CA78K0R	CC-RL	Remarks
Concatenate symbol	&	?	

- Differences in the operators

Operation Type	CA78K0R	CC-RL	Remarks
Arithmetic operation	+ sign, - sign, +, -, *, /, MOD	+ sign, - sign, +, -, *, /, %	
Bit logic operation	NOT, AND, OR, XOR	~, &, |, ^	
Shift operation	SHR (logic), SHL	<<, >>	
Section operation	—	STARTOF, SIZEOF	
Separation operation	HIGH, LOW, HIGHW, LOWW, MIRHW, MIRLW	HIGH, LOW, HIGHW, LOWW, MIRHW, MIRLW,SMRLW	
Comparison operation	=(EQ), <>(NE), >(GT), >=(GE), <(LT), <=(LE) When the result is true, the value is 0FFH.	==, !=, >, >=, <, <= When the result is true, the value is 1.	
Logical operation	—	&&, ||	

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in the directives(1/2)

Instruction Type	CA78K0R	CC-RL	Remarks
Segment definition directives	BSEG	—	
	—	.SECTION	
Memory initialization or area allocation directives	—	.DB8	
	—	.ALIGN	
Macro directives	MACRO	.MACRO	
	LOCAL	.LOCAL	
	REPT	.REPT	
	IRP	.IRP	
	EXITM	.EXITM	
	—	.EXITMA	
	ENDM	.ENDM	
Include directive	—	$BINCLUDE	

Differences in the directives(2/2)

Instruction Type	CA78K0R	CC-RL	Remarks
Conditional assembler directives	—	$IFNDEF	
	—	$IFN	
	—	$ELSEIFN	

Refer to the user's manual for the compiler for more information and make changes as required.

Function Call Interface Specifications

Differences in the normal function call interface

Operation Type	CA78K0R	CC-RL	Remarks
Registers for storing return values	CY BC DE	 A AX BC DE	Since the order or combination for assigning registers is different, refer to the user's manual for the compiler for more information.
Registers for storing arguments	 AX BC	A, X, C, B, E, D AX BC DE	Same as above
Locations for storing auto variables	Stack saddr area (when the -qr option is specified）	Stack	Same as above

Refer to the user's manual for the compiler for more information and make changes as required.

Porting Support Functions

Porting support functions of CC-RL

CC-RL provides the porting support functions.

Porting support functions become valid by specifying the following options.

- compiler porting support functions : -convert_cc option
- assembler porting support functions : -convert_asm option

(Example) -convert_cc=ca78k0r

-convert_asm

It isn't necessary to include “iodefine.h” using #include sentence every source file by specifying the following option.

“iodefine.h” have the definition of SFR name and an interrupt request name.

- Preprocessor control option of Compiler : -preinclude option

(Example) -preinclude=iodefine.h

Applicable #pragma directive(1/4)

When specifying the -convert_cc=ca78k0r option, those descriptions are replaced according to the CC-RL specifications.

Item	CA78K0R	CC-RL	Actions (when porting support function is not used)
C-source level coding of SFR name	#pragma sfr	It isn't replaced to the function of CC-RL. Please use the following way. #include “iodefine.h” or -preinclude=iodefine.h	Delete the #pragma directive. Use the definition of iodefine.h (generated by IDE) for SFR access.
Declaration of interrupt functions	#pragma vect #pragma interrupt	— #pragma interrupt	The format is different. Refer to the manual and rewrite the declaration.

Enabling of interrupt functions
DI( )
EI( )
	#pragma DI
	#pragma EI
		__DI
		__EI
			Delete the #pragma directive and replace the function name as follows:
			__DI( );
			__EI( );

Applicable #pragma directive(2/4)

Item	CA78K0R	CC-RL	Actions (when porting support function is not used)

Enabling of CPU control instructions
HALT( )
STOP( )
BRK( )
NOP( )
	#pragma HALT
	#pragma STOP
	#pragma BRK
	#pragma NOP
		__halt
		__stop
		__brk
		__nop
			Delete the #pragma directive and replace the function name as follows:
			__halt( );
			__stop( );
			__brk( );
			__nop( );

Changing of section name	#pragma section	#pragma section	The format is different. Refer to the manual and change the format.
Changing of module name	#pragma name	—	Delete the #pragma directive. Specify the -rename option of the linker.

Applicable #pragma directive(3/4)

Item	CA78K0R	CC-RL	Actions (when porting support function is not used)

Enabling of rotate functions
rorb( )
rolb( )
rorw( )
rolw( )
	#pragma rot	__rorb
		__rolb
		__rorw
		__rolw
			Delete the #pragma directive and replace the function name as follows:
			__rorb( );
			__rolb( );
			__rorw( );
			__rolw( );

Enabling of multiply functions
mulu( )
muluw( )
mulsw( )
	#pragma mul	__mulu
		__mului
		__mulsi
			Delete the #pragma directive and replace the function name as follows:
			__mulu( );
			__mului( );
			__mulsi( );

Enabling of divide functions
divuw( )
moduw( )
	#pragma div	__divui
		__remui
			Delete the #pragma directive and replace the function name as follows:
			__divui( );
			__remui( );

Applicable #pragma directive(4/4)

Item	CA78K0R	CC-RL	Actions (when porting support function is not used)

Enabling of multiply-and-accumulate functions
macuw( )
macsw( )
	#pragma mac	__macui
		__macsi
			Delete the #pragma directive and replace the function name as follows:
			__macui( );
			__macsi( );

Declaration of RTOS function	#pragma rtos_interrupt	#pragma rtos_interrupt	The format is different. Refer to the manual and rewrite the declaration.

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in declaration of the interrupt functions

- CA78K0R

Format: #pragma vect (or interrupt) [interrupt request name] [function name] [stack switching setting]

```
#pragma interrupt INTP0 inter rb1

void inter ( void ) {
/* interrupt processing for INTP0 pin input */
}
```

- CC-RL

Format: #pragma interrupt [(] interrupt handler request name [(interrupt specification [,...])][)]

```
#include “iodefine.h”
#pragma interrupt inter (vect=INTP0, bank=RB1)
__near void inter ( void ) {
/* interrupt processing for INTP0 pin input */
}
```

It's possible by -preinclude=iodefine.h option, if it isn't written in a source file.

The interrupt request name can be written when iodefine.h is included.

Refer to the user's manual for the compiler for more information and make changes as required.

Differences in changing the section name

- CA78K0R

```
Format: #pragma section [compiler output section name] [new section name] [AT start address]
```

The compiler output section name is changed.

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

The section name @@DATA is changed to DD1 and 2400H is specified as the start address.

- CC-RL

```
Format: #pragma section [section type] [new section name]
```

The section name corresponding with the section type of text, const, data, or bss is changed.

- For the near section: new section name + "_n"
- For the far section: new section name + "_f"
- For the saddr section: new section name + "_s"

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

The section name bss is changed to DD1_f.

Note that when specifying the start address of a section, it should be specified with the -start option of the linker.

Refer to the user's manual for the compiler for more information and make changes as required.

Applicable macros

When specifying the -convert_cc=ca78k0r option, those descriptions are replaced according to the CC-RL specifications.

Macro Name in CA78K0R	Relevant Macro Name in CC-RL	Value
__K0R__	__RL78__	decimal constant 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	

Refer to the user's manual for the compiler for more information and make changes as required.

Applicable keywords(1/5)

When specifying the -convert_cc=ca78k0r option, those descriptions are replaced according to the CC-RL specifications.

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Allocation of variables to saddr area	__sreg sreg	__saddr #pragma saddr	Change __sreg to __saddr.
Absolute address setting	__directmap	#pragma address	An absolute address cannot be specified by __directmap. Use #pragma address. Addresses of variables cannot overlap with each other.
Declaration of hardware interrupt function	__interrupt	#pragma interrupt	Change the declaration using #pragma interrupt.
Declaration of software interrupt function	__interrupt_brk	#pragma interrupt_brk	Change the declaration using #pragma interrupt_brk.

Applicable keywords(2/5)

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Declaration of RTOS function	__rtos_interrupt	#pragma rtos_interrupt	__rtos_interrupt has become unnecessary. Delete "__rtos_interrupt" from the declaration of the interrupt handler function for RTOS.
Declaration of callt function	__callt callt	__callt #pragma callt	Change callt to __callt.
Declaration of bit variables for saddr area*	__boolean boolean bit	—	Change __boolean to char(when the –ansi option is specified). Change __Boolean, Boolean, bit to _Bool(when the –ansi option is not specified).

* Since there are no bit variables, bit variables is treated as 1-byte data when specifying -convert_cc=ca78k0r option.

Applicable keywords(3/5)

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Segment definition directives	CSEG	.CSEG	Change CSEG to .CSEG. The description format of the relocation attribute is different. When the relocation attribute is UNITP, change CSEG to ".CSEG TEXTF" and ".ALIGN 2".
	DSEG	.DSEG	Change DSEG to .DSEG. The description format of the relocation attribute is different.
	BSEG	.BSEG	Change BSEG to .BSEG.
	ORG	.ORG	Change ORG to .ORG.
	EQU	.EQU	Change EQU to .EQU.
	SET	.SET	Change SET to .SET.

Applicable keywords(4/5)

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Branch instruction automatic selection directives	BR	BR !!addr20	Change BR to BR !!addr20.
	CALL	CALL !!addr20	Change CALL to CALL !!addr20.
Memory initialization or area allocation directives	DB	.DB	Change DB to .DB. The interpretation of the “(size)” operand is different.
	DW	.DB2	Change DW to .DB2. The interpretation of the “(size)” operand is different.
	DG	DB4	Change DG to .DB4. The interpretation of the “(size)” operand is different.
	DS	.DS	Change DS to .DS.
	DBIT	.DBIT	Change DBIT to .DBIT.
Linkage directives	PUBLIC	.PUBLIC	Change PUBLIC to .PUBLIC.
	EXTRN	.EXTERN	Change EXTERN to .EXTERN.
	EXTBIT	.EXTBIT	Change EXTBIT to .EXTBIT.

Applicable keywords(5/5)

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Object module name declaration directive	NAME	treated as a comment	Delete the NAME directive. Specify the -rename option of the linker.
Assemble end directive	END	treated as a comment	Delete the END directive.

Refer to the user's manual for the compiler for more information and make changes as required.

Symbol definition directive EQU (when porting support function is not used)

A relocatable label cannot be written as an operand of the symbol definition directive EQU.

Replace the reference points of the name on the left side of EQU with relocatable labels and disable EQU itself.

- CA78K0R

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

Delete this.

Delete this.

Replace this.

Replace this.

- CC-RL

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

Refer to the user's manual for the compiler for more information and make changes as required.

Memory initialization or area allocation directive (when porting support function is not used)

Only one operand can be written in a memory initialization or area allocation directive.

If more than one operand is written, divide the directive into multiple directives.

- CA78K0R

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

Correct this.

- CC-RL

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

Refer to the user's manual for the compiler for more information and make changes as required.

Size in a memory initialization or area allocation directive (when porting support function is not used)

Only one operand can be written in a memory initialization or area allocation directive.

If moreAthan one operand is written, divide the directive into multiple directives.

- CA78K0R

Format: DW (size)

Specify the size in words.

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

Correct this. 3 words x 2 = 6 bytes

- CC-RL

Format: .DS (size)

Specify the size in bytes.

```
.CSEG
.DS 6
```

Refer to the user's manual for the compiler for more information and make changes as required.

Applicable directives(1/3)

When specifying the -convert_asm option, those descriptions are replaced according to the CC-RL specifications.

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Assemble target type specification directive	$PROCESSOR ($PC)	treated as a comment	Specify the -dev option.
Include directive	$INCLUDE ($IC)	$INCLUDE	Change to $INCLUDE.
RAM area allocation specification directive	$RAM_ALLOCATE	treated as a comment	Allocate the target segment using the .CSEG directive.
Conditional assembler directives	$IF	$IF	Use the -define option or .SET.
	$_IF	$IF	Change to $IF.
	$ELSEIF	$ELSEIF	Use the -define option or .SET.
	$_ELSEIF	$ELSEIF	Change to $ELSEIF.
	$ELSE	$ELSE	

Applicable directives(2/3)

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Conditional assembler directives	$ENDIF	$ENDIF	
	$SET, $RESET	treated as a comment	Delete the $SET, $RESET directive.
Debugging information output directive	$DEBUG ($DG), $NODEBUG ($NODG)	treated as a comment	Specify the -debug option.
	$DEBUGA, $NODEBUGA	treated as a comment	
Cross reference list output specification directives	$XREF ($XR), $NOXREF ($NOXR)	treated as a comment	Delete the $XREF ($XR), $NOXREF ($NOXR) directive.
	$SYMLIST, $NOSYMLIST	treated as a comment	Delete the $SYMLIST, $NOSYMLIST directive.
Assemble list directives	$EJECT ($EJ)	treated as a comment	Delete the $EJECT ($EJ) directive.
	$LIST ($LI), $NOLIST ($NOLI)	treated as a comment	Delete the $LIST ($LI), $NOLIST ($NOLI) directive.
	$GEN, $NOGEN	treated as a comment	Delete the $GEN, $NOGEN directive.

Applicable directives(3/3)

Function	Keyword	Relevant Keyword in CC-RL	Actions (when porting support function is not used)
Assemble list directives	$COND, $NOCOND	treated as a comment	Delete the $COND, $NOCOND directive.
	$TITLE ($TI)	treated as a comment	Delete the $TITLE ($TI) directive.
	$SUBTITLE ($ST)	treated as a comment	Delete the $SUBTITLE ($ST) directive.
	$FORMFEED, $NOFORMFEED	treated as a comment	Delete the $FORMFEED, $NOFORMFEED directive.
	$WIDTH	treated as a comment	Delete the $WIDTH directive.
	$LENGTH	treated as a comment	Delete the $LENGTH directive.
	$TAB	treated as a comment	Delete the $TAB directive.
Kanji code directive	$KANJICODE	treated as a comment	Specify the -character_set option.
Other directives	$TOL_INF, $DGS, $DGL	treated as a comment	Delete the $TOL_INF, $DGS, $DGL directive.

Refer to the user's manual for the compiler for more information and make changes as required.

FAQ

FAQ

- After this page, FAQ about Compiler and Linker Error massages at converting from CA78K0R to CC-RL is described.
- You can show the FAQ in Renesas web, so please refer to a web for the latest information.
  - ● http://www.renesas.com/rl78_c
    -> FAQ

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.

FAQ

FAQ No.	Q	A
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
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.

FAQ

FAQ No.	Q	A
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.

FAQ

FAQ No.	Q	A
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.

Revision History

Revision	Description	Page
Rev.1.00	First revision	-
Rev.1.01	Addition of FAQ title	P3
	Addition of method including “iodefine.h”	P27
	Modification of explain “#pragma sfr”	P28
	Addition of method including “iodefine.h” by -preinclude=iodefine.h option	P32
	Addition of FAQ	P47
Rev.1.02	Revised the destination to CC-RL V1.03.00	-

Renesas System Design Co., Ltd.