RL78開発環境移行ガイド

RL78ファミリ間の移行
(コンパイラ編：コーディング)
(CA78K0R→CC-RL)

2016/12/28
R20UT3416JJ0102

ソフトウェア事業部ソフトウエア技術部
ルネサス システムデザイン株式会社

はじめに

- 本資料は、RL78ファミリ用CコンパイラCA78K0R用のプロジェクトやソースコードを、RL78ファミリ用CコンパイラCC-RLへ移行する際のソースコードの差異について、記述しています。

- 本資料では、RL78ファミリ用CコンパイラCA78K0R、およびRL78ファミリ用CコンパイラCC-RLを対象に説明しています。
  対象バージョンは以下の通りです。
  - CA78K0R V1.20以降
  - CC-RL V1.03.00

アジェンダ

- はじめに　ページ 2
- コンパイラ言語仕様　ページ 4
- アセンブラ言語仕様　ページ 21
- 関数呼び出しインタフェース仕様　ページ 25
- 移行支援機能　ページ 27
- FAQ　ページ 47

コンパイラ言語仕様

言語仕様の相違点(1/2)

項目	CA78K0R	CC-RL	備考
言語	C言語	C言語	
言語規格	C89	C90, C99の一部機能をサポート	
エンディアン	little	little	
利用可能多バイト文字	EUC, SJIS	EUC, SJIS, UTF-8, big5, gbk	
多バイト文字サポート範囲	コメントにおいて日本語記述をサポート	コメントと文字列で日本語/中国語記述をサポート	
signed/unsignedの付かないchar型の扱い	符号付き整数 -quオプション指定時は、符号なし整数	符号なし整数 -signed_charオプション指定時は、符号付き整数	
double型	IEEE754-1985に準拠 32 ビット・データ	IEEE754-1985に準拠 ・-dbl_size=4 使用時 32 ビット・データ ・-dbl_size=8 使用時 64 ビット・データ	-dbl_size=8は RL78-S3コアのみ

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

言語仕様の相違点(2/2)

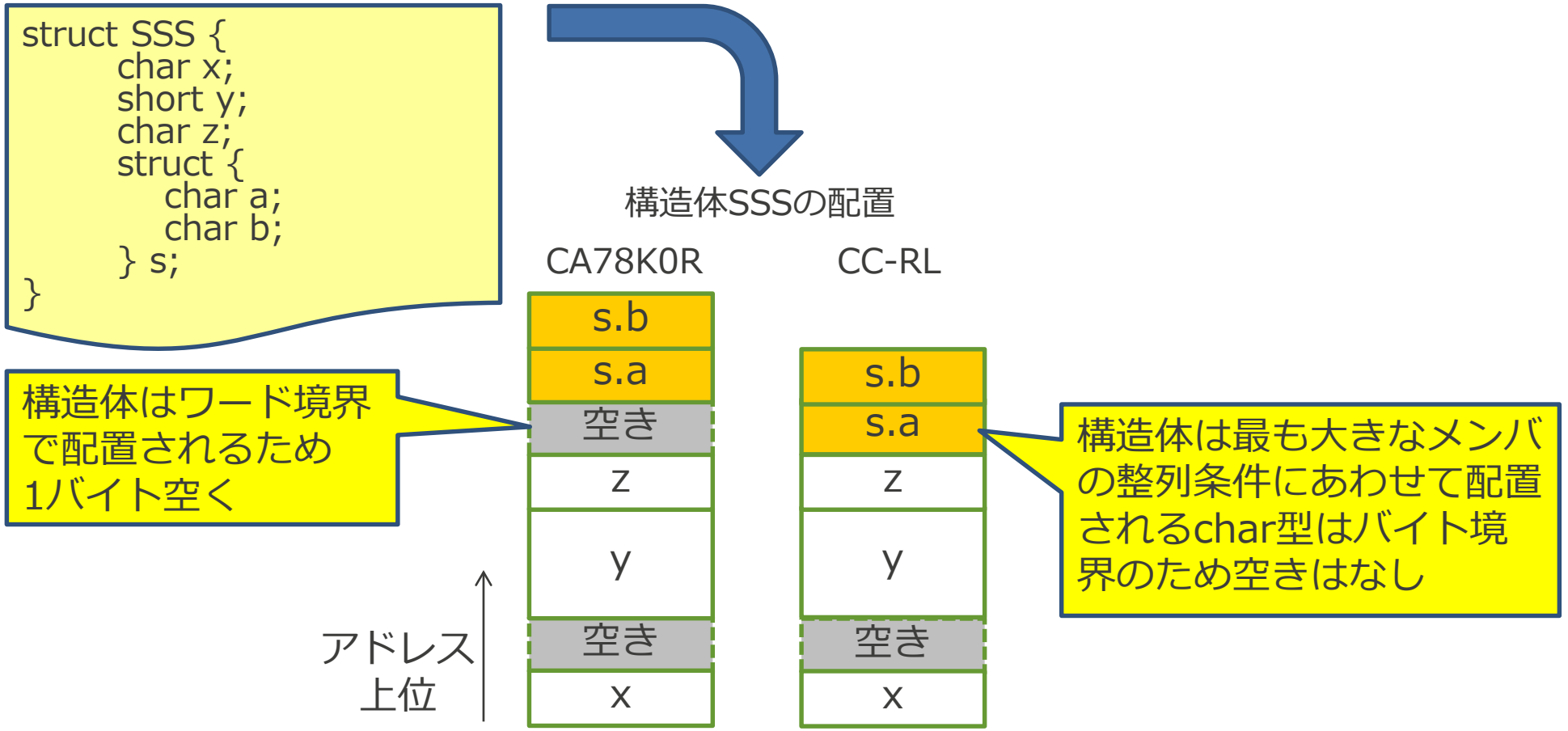
項目	CA78K0R	CC-RL	備考
構造体と共用体指定子内のint型ビット・フィールド	符号なしとして扱う	符号なしとして扱う -signed_bitfield 使用時はsigned int 型となる	
構造体と共用体指定子内のビット・フィールドの割り付け順序	下位から上位 -rb オプション指定時は上位から下位	下位から上位	
構造体と共用体指定子内の各メンバの境界	・1バイト境界 char/signed char/unsigned char ・その他：2バイト境界	・1バイト境界 char/signed char/unsigned char/_Bool ・その他：2バイト境界	
列挙型指定子	列挙定数の範囲により列挙型は以下のいずれかになる signed char/unsigned char/signed int	列挙定数の範囲により列挙型は以下のいずれかになる char/signed charunsigned char/signed short	

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

構造体と共用体指定子内の各メンバの境界の相違点

構造体SSSの定義

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



詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

列挙型指定子の相違点

列挙子の範囲によって内部表現の型が変わります。

- CA78K0Rの場合（優先度順）
  範囲：-128～127　→　signed char
  範囲：0～255　→　unsigned char
  範囲：-32768～32767　→　signed int
- CC-RLの場合（優先度順）
  ・-signed_charあり
  範囲：-128～127(0～127の場合も含む)　→　char
  範囲：0～255　→　unsigned char
  範囲：上記以外　→　signed short
  ・-signed_charなし
  範囲：-128～127　→　signed char
  範囲：0～255(0～127の場合も含む)　→　char
  範囲：上記以外　→　signed short

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

ヘッダ・ファイル取り込みの相違点

項目	CA78K0R	CC-RL	備考
include <文字列>形式の検索順序	(1) -i オプションで指定したフォルダ (2) 環境変数INC78K0R で指定されているフォルダ (3) 標準インクルード・ファイル・フォルダ	(1) -Iオプションで指定したフォルダ (2) 標準インクルード・ファイル・フォルダ	
#include ”文字列“形式の検索順序	(1) ソース・ファイルがあるフォルダ (2) -i オプションで指定したフォルダ (3) 環境変数INC78K0R で指定されているフォルダ (4) 標準インクルード・ファイル・フォルダ	(1) ソース・ファイルがあるフォルダ (2) -Iオプションで指定したフォルダ (3) 標準インクルード・ファイル・フォルダ	

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

翻訳限界値の相違点(1/2)

項目	CA78K0R	CC-RL
複合文、繰返し制御構造及び選択制御構造に対する入れ子のレベル数	45	メモリ依存
条件付取込みにおける入れ子のレベル数	255	
一つの宣言中の一つの算術型、構造体型、共用体型又は不完全型を修飾するポインタ、配列及び関数宣言子（の任意の組み合わせ）の個数	12	128
一つの完全宣言子における括弧で囲まれた宣言子の入れ子のレベル数	–	メモリ依存
一つの完全式における括弧で囲まれた式の入れ子のレベル数	1024	
マクロ名において意味がある先頭の文字数	256	
内部識別子において意味がある先頭の文字数	249	
外部識別子において意味がある先頭の文字数	249	
一つの翻訳単位中における外部識別子数	1024	
一つのブロック中におけるブロック有効範囲をもつ識別子数	255	
一つの翻訳単位中で同時に定義されうるマクロ識別子数	60000	
一つの関数定義における仮引数の個数	39	

※CA78K0Rの列は、V.1.50以降の値です。

翻訳限界値の相違点(2/2)

項目	CA78K0R	CC-RL
一つの関数呼出しにおける実引数の個数	39	メモリ依存
一つのマクロ定義における仮引数の個数	31	
一つのマクロ呼出しにおける実引数の個数	31	
一つの論理ソース行における文字数	32767	
（連結後の）単純文字列リテラル又はワイド文字列リテラル中のおける文字数	509	
（ホスト環境の場合）一つのオブジェクトのバイト数	65535	32767 (-large_variableオプション指定時、65535)
#includeで取り込まれるファイルの入れ子レベル数	50	メモリ依存
一つのswitch文（入れ子になったswitch文を除く）中におけるcase名札の個数	1024	
一つの構造体又は共用体のメンバ数	1024	
一つの列挙体における列挙定数の個数	255	
一つのメンバ宣言並びにおける構造体又は共用体定義の入れ子のレベル数	15	

※CA78K0Rの列は、V.1.50以降の値です。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

数値限界値の相違点(1/3)

項目	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)

※ CHAR_MIN, CHAR_MAXの()内は、-signed_char指定時
それ以外の（）内は、-dbl_size = 8オプション指定時（RL78-S3コアのみ）

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

数値限界値の相違点(2/3)

項目	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)

※ （）内は、-dbl_size = 8オプション指定時（RL78-S3コアのみ）

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

数値限界値の相違点(3/3)

項目	CA78K0R	CC-RL
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)

※ （）内は、-dbl_size = 8オプション指定時（RL78-S3コアのみ）

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

#pragma指定の相違点

項目	CA78K0R	CC-RL	対応方法
データ挿入関数 __OPC( ) の有効化	#pragma opc	–	#pragma指令を削除して、 #pragma inline_asmとアセンブラ命令を使用して、データ挿入処理を記述してください。
ブート領域からフラッシュ領域への関数呼び出し	#pragma ext_func	–	該当する指令はありません。 #pragma指令を削除してください。 絶対アドレスを指定し、関数を呼び出してください。
標準ライブラリ関数 memcpy( )および memset( )のインライン展開指示	#pragma inline	–	#pragma指令を削除してください。 CC-RLでは、ユーザ関数のインライン展開の意味に変わります。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

キーワードマクロの相違点

CA78K0Rのマクロ名	CC-RLの該当マクロ名	値
__K0R_LARGE__	なし	–
CPU マクロ	なし	–

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

キーワードの相違点(1/2)

機能	キーワード	CC-RLの該当キーワード	対応方法
near/far属性	__near/__far	__near/__far	指定位置が異なります。
saddr領域へのビット変数の宣言	__boolean boolean bit	–	構造体のビットフィールドを定義、宣言してビットアクセス処理を変更してください。
asm文	__asm #asm～#endasm	#pragma inline_asm	asm文を使って、Cソース内部に直接アセンブラ命令を記述することはできません。アセンブラ命令部分をアセンブリ記述関数で定義して、#pragma inline_asmを使用してください。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

キーワードの相違点(2/2)

機能	キーワード	CC-RLの該当キーワード	対応方法
78K0互換用	__callf callf	–	78K0互換用はサポートしていません。 該当キーワードを削除してください。
	noauto	–	
	__leaf norec	–	
	__pascal	–	
	__temp	–	
	__mxcall	–	

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

移行支援機能
SADDR領域へのビット変数宣言の相違点

●CA78K0R

書式： bit (またはbooleanまたは__boolean)　変数名

●CC-RL

ビット変数はないため、構造体にビットフィールドを定義します

```
書式： __saddr struct タグ名 {
                    型名 フィールド名: ビット幅 ;
                    型名 フィールド名: ビット幅 ;
                    …
                    型名 フィールド名: ビット幅 ;
                  } ;
```

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

char型1ビットの数値

・ビットフィールドのメンバの型は、次の型を使用できます
char, signed char, unsigned char, signed short, unsigned short, signed int,
unsigned int, signed long, unsigned long, signed long long, unsigned long long

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

アセンブラ命令記述の相違点

●CA78K0R

```
書式：#asm
          ～アセンブラ記述～
        #endasm
```

●CC-RL

```
書式：#pragma inline_asm [(] 関数名 [,...][)]
```

```
#pragma inline_asm func

static int func() {
  /* アセンブラ記述 */
}
```

アセンブリ記述用関数funcを宣言
func関数内にアセンブラ・ソースを記述

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

アセンブラ言語仕様

マクロ・オペレータ/演算子の相違点

- マクロ・オペレータの相違点

演算種別	CA78K0R	CC-RL	備考
コンカティネート記号	&	?	

- 演算子の相違点

演算種別	CA78K0R	CC-RL	備考
算術演算	+符号, -符号, +, -, *, /,MOD	+符号, -符号, +, -, *,/,%	
ビット論理演算	NOT, AND, OR, XOR	~, &, |, ^	
シフト演算	SHR(論理), SHL	<<, >>	
セクション演算	–	STARTOF, SIZEOF	
分離演算	HIGH, LOW, HIGHW, LOWW, MIRHW, MIRLW	HIGH, LOW, HIGHW, LOWW, MIRHW, MIRLW,SMRLW	
比較演算	=(EQ), <>(NE), >(GT), >=(GE), <(LT), <=(LE) 結果が真の場合の値は0FFH	==, !=, >, >=, <, <= 結果が真の場合の値は1	
論理演算	–	&&, ||	

擬似命令の相違点

命令種別	CA78K0R	CC-RL	備考
セグメント定義 疑似命令	BSEG －	－ .SECTION	
メモリ初期化・ 領域確保疑似命令	－	.DB8	
	－	.ALIGN	
マクロ疑似命令	MACRO	.MACRO	
	LOCAL	.LOCAL	
	REPT	.REPT	
	IRP	.IRP	
	EXITM	.EXITM	
	－	.EXITMA	
	ENDM	.ENDM	

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

制御命令の相違点

命令種別	CA78K0R	CC-RL	備考
インクルード制御命令	–	$BINCLUDE	
条件付きアセンブル制御命令	–	$IFNDEF	
	–	$IFN	
	–	$ELSEIFN	

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

関数呼び出しインタフェース仕様

通常関数呼び出しインタフェースの相違点

演算種別	CA78K0R	CC-RL	備考
戻り値を格納するレジスタ	CY BC DE	 A AX BC DE	レジスタの割り当て順序、組み合わせは異なりますので、詳細はコンパイラユーザーズマニュアルを参照してください。
引数を格納するレジスタ	 AX BC	A、X、C、B、E、D、 AX BC DE	同上
自動変数の格納場所	スタック saddr領域（-qrオプション指定時）	スタック	同上

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

移行支援機能

CC-RLの移行支援機能

CC-RLでは移行支援機能を準備しています。

次のオプションを指定することにより、移行支援機能が有効になります。

・コンパイラの移行支援機能：-convert_ccオプション

・アセンブラの移行支援機能：-convert_asmオプション

（例）-convert_cc=ca78k0r

-convert_asm

また、次のオプションを指定することにより、SFR、割り込み要求名が定義されたファイル iodefine.h（IDEで生成）を、ソースファイルごとに#include文でインクルードする必要がなくなります。

・コンパイラのプリプロセッサ制御機能：-preincludeオプション

（例）-preinclude=iodefine.h

コンパイラ移行支援機能対象の#pragma指令(1/4)

-convert_cc=ca78k0rを指定時、CC-RLの仕様に置換します。

項目	CA78K0R	CC-RL	移行支援機能未使用時の対応方法
SFR名をCソース・レベルで記述	#pragma sfr	CC-RLの機能に置換されません。 次の方法を使用してください。 #include "iodefine.h“ または -preinclude=iodefine.h	#pragma指令を削除してください。 SFRのアクセスは、 iodefine.h(IDEが生成)の定義を使用してください。
割り込み関数の宣言	#pragma vect #pragma interrupt	– #pragma interrupt	書式が異なります。 マニュアルを参照して変更してください。

割り込み機能
DI( )
EI( )
の有効化
	#pragma DI
	#pragma EI
		__DI
		__EI
			#pragma指令を削除して、関数名を変更してください。
			__DI( );
			__EI( );

コンパイラ移行支援機能対象の#pragma指令(2/4)

項目	CA78K0R	CC-RL	移行支援機能未使用時の対応方法

CPU 制御命令
HALT( )
STOP( )
BRK( )
NOP( )
の有効化
	#pragma HALT
	#pragma STOP
	#pragma BRK
	#pragma NOP
		__halt
		__stop
		__brk
		__nop
			#pragma指令を削除して、関数名を変更してください。
			__halt( );
			__stop( );
			__brk( );
			__nop( );

セクション名の変更	#pragma section	#pragma section	書式が異なります。 マニュアルを参照して変更してください。
モジュール名の変更	#pragma name	–	#pragma指令を削除してください。リンカの-renameオプションを指定してください。

コンパイラ移行支援機能対象の#pragma指令(3/4)

項目	CA78K0R	CC-RL	移行支援機能未使用時の対応方法

ローテート関数
rorb( )
rolb( )
rorw( )
rolw( )
の有効化
	#pragma　rot	__rorb
		__rolb
		__rorw
		__rolw
			#pragma指令を削除して、関数名を変更してください。
			__rorb( );
			__rolb( );
			__rorw( );
			__rolw( );

乗算関数
mulu( )
muluw( )
mulsw( )
の有効化
	#pragma　mul	__mulu
		__mului
		__mulsi
			#pragma指令を削除して、関数名を変更してください。
			__mulu( );
			__mului( );
			__mulsi( );

除算関数
divuw( )
moduw( )
の有効化
	#pragma　div	__divui
		__remui
			#pragma指令を削除して、関数名を変更してください。
			__divui( );
			__remui( );

コンパイラ移行支援機能対象の#pragma指令(4/4)

項目	CA78K0R	CC-RL	移行支援機能未使用時の対応方法

積和演算関数
macuw( )
macsw( )
の有効化
	#pragma　mac	__macui
		__macsi
			#pragma指令を削除して、関数名を変更してください。
			__macui( );
			__macsi( );

RTOS関数の宣言	#pragma rtos_interrupt	#pragma　rtos_interrupt	書式が異なります。 マニュアルを参照して変更してください。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

割り込み関数宣言の相違点

●CA78K0R

書式： #pragma vect(またはinterrupt) 割り込み要求名 関数名　[スタック切り替え指定]

```
#pragma interrupt INTP0 inter rb1

void inter ( void ) {
/* INTP0端子入力に対する割り込み処理*/
}
```

●CC-RL

書式： #pragma interrupt [(] 割り込みハンドラ要求名[(割り込み仕様 [,...])][)]

```
#include “iodefine.h”

#pragma interrupt inter (vect=INTP0, bank=RB1)
__near void inter ( void ) {
/* INTP0端子入力に対する割り込み処理*/
}
```

ソースファイルに記述せず、
-preinclude=iodefine.h
オプションでも可能

iodefine.hのインクルード時、
割り込み要求名を記述可能

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

セクション名変更の相違点

●CA78K0R

書式： #pragma section コンパイラ出力セクション名 変更セクション名 [AT 開始アドレス]

コンパイラ出力セクション名を変更します

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

セクション名@@DATAをDD1に変更し，開始アドレスを2400H に指定

●CC-RL

書式： #pragma section [セクション種別][変更セクション名]

セクション種別text|const|data|bssに対応したセクション名を変更します

・nearセクションの場合　→　変更セクション名+”_n”

・farセクションの場合　→　変更セクション名+”_f”

・saddrセクションの場合　→　変更セクション名+”_s”

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

bssのセクション名をDD1_fに変更

なお、セクションの開始アドレスを指定する場合、リンカの-startオプションで指定してください。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

コンパイラ移行支援機能対象のキーワードマクロ

-convert_cc=ca78k0rを指定時、CC-RLの仕様に置換します。

CA78K0Rのマクロ名	CC-RLの該当マクロ名	値
__K0R__	__RL78__	10進定数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__	–	

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

コンパイラ移行支援機能対象のキーワード(1/2)

-convert_cc=ca78k0rを指定時、CC-RLの仕様に置換します。

機能	キーワード	CC-RLの該当キーワード	移行支援機能未使用時の対応方法
変数をsaddr領域に配置	__sreg sreg	__saddr #pragma saddr	__sregを__saddrに変更してください。
絶対番地指定	__directmap	#pragma address	__directmapによる絶対番地指定はできません。#pragma addressを使用してください。また変数のアドレスを重ねて配置することはできません。
ハードウェア割り込み関数の宣言	__interrupt	#pragma interrupt	#pragma interruptを使用して変更してください。
ソフトウェア割り込み関数の宣言	__interrupt_brk	#pragma interrupt_brk	#pragma interrupt_brkを使用して変更してください。
RTOS関数の宣言	__rtos_interrupt	#pragma rtos_interrupt	__rtos_interruptは不要になりました。ＲＴＯＳ用割り込みハンドラ関数の宣言から、「__rtos_interrupt」を削除してください。

コンパイラ移行支援機能対象のキーワード(2/2)

機能	キーワード	CC-RLの該当キーワード	移行支援機能未使用時の対応方法
callt関数の宣言	__callt callt	__callt #pragma callt	callltを__callltに置き換えてください。
saddr領域へのビット変数の宣言※	__boolean boolean bit	–	__boolean をchar に置き換えてください（-ansi オプション指定時）。 __boolean，boolean，bitを_Bool に置き換えてください（-ansi オプション非指定時）。

※CC-RLにはビット型はないため、移行支援機能指定時、ビット変数は１バイトのデータとして扱われます。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

アセンブラ移行支援機能対象の疑似命令(1/3)

-convert_asmを指定時、CC-RLの仕様に置換します。

命令種別	CA78K0R	CC-RL	移行支援機能未使用時の対応方法
セグメント定義疑似命令	CSEG	.CSEG	.CSEGに変更してください。 再配置属性の記述形式は異なります。 再配置属性がUNITPの場合には.CSEG TEXTFと.ALIGN 2 に変更してください。
	DSEG	.DSEG	.DSEGに変更してください。 再配置属性の記述形式は異なります。
	BSEG	.BSEG	.BSEGに変更してください。
	ORG	.ORG	.ORGに変更してください。
シンボル定義疑似命令	EQU	.EQU	.EQUに変更してください。
	SET	.SET	.SETに変更してください。
分岐命令自動選択疑似命令	BR	BR !!addr20	BR !!addr20に変更してください。
	CALL	CALL !!addr20	CALL !!addr20に変更してください。

アセンブラ移行支援機能対象の疑似命令(2/3)

命令種別	CA78K0R	CC-RL	移行支援機能未使用時の対応方法
メモリ初期化・領域確保疑似命令	DB	.DB	.DBに変更してください。 「(サイズ)」オペランドの解釈が異なります。
	DW	.DB2	.DB2に変更してください。 「(サイズ)」オペランドの解釈が異なります。
	DG	.DB4	.DB4に変更してください。 「(サイズ)」オペランドの解釈が異なります。
	DS	.DS	.DSに変更してください。
	DBIT	.DBIT	.DBITに変更してください。
リンケージ疑似命令	PUBLIC	.PUBLIC	.PUBLICに変更してください。
	EXTRN	.EXTERN	.EXTERNに変更してください。
	EXTBIT	.EXTBIT	.EXTBITに変更してください。

アセンブラ移行支援機能対象の疑似命令(3/3)

命令種別	CA78K0R	CC-RL	移行支援機能未使用時の対応方法
オブジェクト モジュール名宣言疑似命令	NAME	コメント扱い	NAMEを削除してください。リンカの- renameオプションを指定してください。
アセンブル終了 疑似命令	END	コメント扱い	ENDを削除してください。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

シンボル定義疑似命令EQU（移行支援機能未使用時の対応）

シンボル定義疑似命令EQUのオペランドには、リロケータブルなラベルを記述できません。

EQUの左辺にあるネームの参照個所をリロケータブルなラベルに置き換えて、EQU自体は無効化してください。

●CA78K0R

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

削除

削除

置き換え

置き換え

●CC-RL

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

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

メモリ初期化、領域確保疑似命令（移行支援機能未使用時の対応）

メモリ初期化、領域確保疑似命令のオペランドは1つのみ記述可能です。
複数のオペランドを記述した場合、複数の疑似命令に分割してください。

●CA78K0R

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

修正

●CC-RL

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

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

メモリ初期化や領域確保疑似命令におけるサイズ（移行支援機能未使用時の対応）

メモリ初期化や領域確保疑似命令における「(サイズ)」オペランドの解釈が異なります。

- CA78K0R

書式：DW (サイズ)　サイズはワード単位で指定

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

修正　3ワード×2=6バイト

- CC-RL

書式：.DS (サイズ)　サイズはバイト単位で指定

```
CSEG
DS 6
```

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

アセンブラ移行支援機能対象の制御命令(1/3)

-convert_asmを指定時、CC-RLの仕様に置換します。

命令種別	CA78K0R	CC-RL	移行支援機能未使用時の対応方法
アセンブル対象品種指定制御命令	$PROCESSOR ($PC)	コメント扱い	-devオプションを指定してください。
インクルード制御命令	$INCLUDE ($IC)	$INCLUDE	$INCLUDEに変更してください。
RAM領域配置指定制御命令	$RAM_ALLOCATE	コメント扱い	.CSEG疑似命令で対象セグメントを配置してください。
条件付きアセンブル制御命令	$IF	$IF	-defineオプションまたは.SETを使用してください。
	$_IF	$IF	$IFに変更してください。
	$ELSEIF	$ELSEIF	-defineオプションまたは.SETを使用してください。
	$_ELSEIF	$ELSEIF	$ELSEIFに変更してください。
	$ELSE	$ELSE	
	$ENDIF	$ENDIF	
	$SET, $RESET	コメント扱い	$SET, $RESETを削除してください。

アセンブラ移行支援機能対象の制御命令(2/3)

命令種別	CA78K0R	CC-RL	移行支援機能未使用時の対応方法
デバッグ情報出力制御命令	$DEBUG ($DG), $NODEBUG ($NODG)	コメント扱い	-debugオプションを指定してください。
	$DEBUGA, $NODEBUGA	コメント扱い	
クロスリファレンスリスト出力指定制御命令	$XREF ($XR), $NOXREF ($NOXR)	コメント扱い	$XREF ($XR), $NOXREF ($NOXR)を削除してください。
	$SYMLIST, $NOSYMLIST	コメント扱い	$SYMLIST, $NOSYMLISTを削除してください。
アセンブル・リスト制御命令	$EJECT ($EJ)	コメント扱い	$EJECT ($EJ)を削除してください。
	$LIST ($LI), $NOLIST ($NOLI)	コメント扱い	$LIST ($LI), $NOLIST ($NOLI)を削除してください。
	$GEN, $NOGEN	コメント扱い	$GEN, $NOGENを削除してください。
	$COND, $NOCOND	コメント扱い	$COND, $NOCONDを削除してください。
	$TITLE ($TI)	コメント扱い	$TITLE ($TI)を削除してください。
	$SUBTITLE ($ST)	コメント扱い	$SUBTITLE ($ST)を削除してください。

アセンブラ移行支援機能対象の制御命令(3/3)

命令種別	CA78K0R	CC-RL	移行支援機能未使用時の対応方法
アセンブル・リスト制御命令	$FORMFEED, $NOFORMFEED	コメント扱い	$FORMFEED, $NOFORMFEEDを削除してください。
	$WIDTH	コメント扱い	$WIDTHを削除してください。
	$LENGTH	コメント扱い	$LENGTHを削除してください。
	$TAB	コメント扱い	$TABを削除してください。
漢字コード制御命令	$KANJICODE	コメント扱い	-character_setオプションを指定してください。
その他の制御命令	$TOL_INF, $DGS, $DGL	コメント扱い	$TOL_INF, $DGS, $DGLを削除してください。

詳細は、コンパイラユーザーズマニュアルを参照し変更してください。

FAQ

FAQ

- 本ページ以降では、CA78K0RからCC-RLに移行する際に、コンパイル、リンク時に出力されるエラーに関するFAQを記載します。
- FAQについては、弊社Webでも公開していますので、最新の情報はWebをご参照ください。

  ●http://www.renesas.com/rl78_c
  → よくあるお問い合わせ(FAQ)

FAQ

FAQ No.	Q	A

1011599	SFRをアクセスすると次のエラーになるのですが、対処方法を教えてください。
	
	E0520020:識別子 "xxxx" は定義されていません。
		統合開発環境でプロジェクトを作成時にiodefine.hファイルが生成されます。SFRの予約語を使用する際には、このファイルをインクルードしてください。バイトとワードのSFRとビットのSFR（ハードウェアマニュアルのビット番号が四角で囲まれたもの）は、その名前でアクセス可能です。
		（例） #include"iodefine.h"
		ADM2 = 0x12; /* バイト予約語 */
		ADTYP = 1; /* ビット予約語 */
		なお、ファイルのインクルードは、コンパイラの-preincludeオプションでも指定可能です。
		（例） -preinclude=iodefine.h

FAQ

FAQ No.	Q	A

1011600	SFRのビットをアクセスすると次のエラーになるのですが、対処方法を教えてください。
	
	E0520020:識別子 "xxxx" は定義されていません。
	E0520065:";" がありません。
		統合開発環境でプロジェクトを作成時にiodefine.hファイルが生成されます。SFRの予約語を使用する際には、このファイルをインクルードしてください。バイトとワードのSFRとビットのSFR（ハードウェアマニュアルのビット番号が四角で囲まれたもの）は、その名前でアクセス可能です。ハードウェアマニュアルのビット番号が四角で囲まれていないものは、iodefine.hで定義されている_bitの名前の付いたバイトとワードの予約語を利用してアクセスしてください。
		（例） #include"iodefine.h"
		P0_bit.no2 = 1; /* ビットに予約語がない */
		また、CC-RLでは、移行支援機能のオプションを用意していますので、コンパイラの-convert_ccオプションを使用すると、_bitの名前の付いたバイトとワードの予約語を利用せずに、CA78K0Rで記述していたように記載もできます。
		（例） #include"iodefine.h"
		P0.2 = 1; /* ビットに予約語がない */
		なお、ファイルのインクルードは、コンパイラの-preincludeオプションでも指定可能です。
		（例） -preinclude=iodefine.h

FAQ

FAQ No.	Q	A
1011601	割り込み関数を#pragma定義すると次のエラーになるのですが、対処方法を教えてください。 E0523005:#pragma の構文が不正です。	割り込み関数は、 #pragma interrupt [(]割り込みハンドラ名[(割り込み仕様 [,...])][)] の形式で記述してください。 統合開発環境でプロジェクトを作成時にiodefine.hファイルが生成されます。この中で、割り込み要求名が定義されていますので、割り込み要求名を使用するCソースファイルでは、iodefine.hをインクルードしてください。 また、CC-RLでは、移行支援機能のオプションを用意していますので、コンパイラの-convert_ccオプションを使用すると、一部の記述については、CA78K0Rの形式で記述することが可能です。
1011602	割り込み関数の定義すると次のエラーになるのですが、対処方法を教えてください。 E0520065:";" がありません。	割り込み関数は、#pragma interruptで指定してください。 割り込み修飾子__interruptはありません。 また、CC-RLでは、移行支援機能のオプションを用意していますので、コンパイラの-convert_ccオプションを使用すると、一部の記述については、CA78K0Rの形式で記述することが可能です。

FAQ

FAQ No.	Q	A
1011603	割り込み関数を定義すると次のエラーになるのですが、対処方法を教えてください。 E0520014:前処理指令の後に不正な文字があります。	統合開発環境でプロジェクトを作成時にiodefine.hファイルが生成されます。この中で、割り込み要求名が定義されていますので、#pragma interrupt の前に、iodefine.hをインクルードしてください。なお、ファイルのインクルードは、コンパイラの-preincludeオプションでも指定可能です。 （例）-preinclude=iodefine.h
1011604	ライブラリファイルを指定すると次のエラーになるのですが、対処方法を教えてください。 E0562201:Illegal library file："xxxx.lib"	CA78K0Rのライブラリファイルを指定していないか確認してください。CA78K0RとCC-RLはオブジェクトファイルの形式が異なるため、CA78K0Rで作成したライブラリをCC-RLでリンクすることはできません。CC-RL用にライブラリファイルを作成しなおして、リンクしてください。
1011605	RL78コンパイラCC-RLを使用しています。入力ファイルにオブジェクトファイルを指定すると次のエラーになるのですが、対処方法を教えてください。 E0562200:Illegal object file："xxxx.rel"	CA78K0Rのオブジェクトファイルを指定していないか確認してください。CA78K0RとCC-RLはオブジェクトファイルの形式が異なるため、CA78K0Rで作成したオブジェクトファイルをCC-RLでリンクすることはできません。CC-RL用にオブジェクトファイルを作成しなおして、リンクしてください。

FAQ

FAQ No.	Q	A
1011606	コンパイルすると次のワーニングになるのですが、なぜでしょうか？ W0511179:この評価版は残りxx日間有効です。	CC-RLのライセンスキーが登録されていないためです。 最初にビルドした時から60日間の試用期間があり、その間は、無償評価版として使用制限されません。その残りの日数を表示しているメッセージです。 試用期間終了後は、リンクサイズが64Kバイト以内に制限され、MISRA-Cチェック機能は使用できません。
1011607	iodefine.hをインクルードしているのですが、PSWをアクセスすると次のエラーになるのですが、対処方法を教えてください。 E0520020:識別子 "PSW" は定義されていません。	PSWを直接アクセスすることはできませんので、iodefine.hファイルに、PSWの定義はありません。 PSWを操作する次の組み込み関数を使用してください。 ・__get_psw PSWの内容を返します。 ・__set_psw PSWに値を設定します。

改定履歴

版数	内容	箇所
Rev. 1.00	初版	-
Rev. 1.01	FAQの項目追加	P3
	iodefine.hのインクルード方法の説明追加	P27
	#pragma　sfr の CC-RLの説明見直し	P28
	-preinclude=iodefine.h オプションの説明追加	P32
	FAQの追加	P48
Rev. 1.02	移行先CC-RLのバージョンをV1.01.00→ V1.03.00	-

ルネサス システムデザイン株式会社