User's Manual

CcnvNC30

Cソースコンバータ

ユーザーズマニュアル

対象デバイス

RL78 ファミリ

対象バージョン

V1.00.00 以降

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2016.04

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

(2012. 4)

このマニュアルの使い方

このマニュアルは，RL78ファミリ用アプリケーション・システムを開発する際のCソースコンバータ（CcnvNC30）について説明します。

対象者　　このマニュアルは，RL78 ファミリ用コンパイラ CC-RL を使用してアプリケーション・システムを開発するユーザを対象としています。

目的　　このマニュアルは，M16C シリーズ，R8C ファミリ用 C コンパイラ NC30 の開発環境を，RL78 ファミリ用 C コンパイラ CC-RL 用に移行するための資料として役立つことを目的としています。

構成　　このマニュアルは，大きく分けて次の内容で構成しています。

1. 概要
2. コマンド・リファレンス
3. コンバータ変換仕様
4. メッセージ
5. 注意事項

読み方　　このマニュアルを読むにあたっては，電気，論理回路，マイクロコンピュータに関する一般知識が必要となります。

凡例

データ表記の重み		: 左が上位桁，右が下位桁
注		: 本文中についた注の説明
注意		: 気を付けて読んでいただきたい内容
備考		: 本文中の補足説明
数の表記		: 10 進数 ... XXXX
		16 進数 ... 0xXXXX

NC30，CC-RL については下記のマニュアルを参照してください。最新版はルネサス エレクトロニクスのホームページに掲載されています。

コンパイラ	資料名	資料番号
NC30	M3T-NC30WA V.6.00 C/C++コンパイラ ユーザーズマニュアル (M16C シリーズ, R8C ファミリ用 C/C++コンパイラパッケージ)	RJJ10J2777-0100
CC-RL	RL78 コンパイラ CC-RL ユーザーズマニュアル	R20UT3123JJ0102

目次

1. 概要

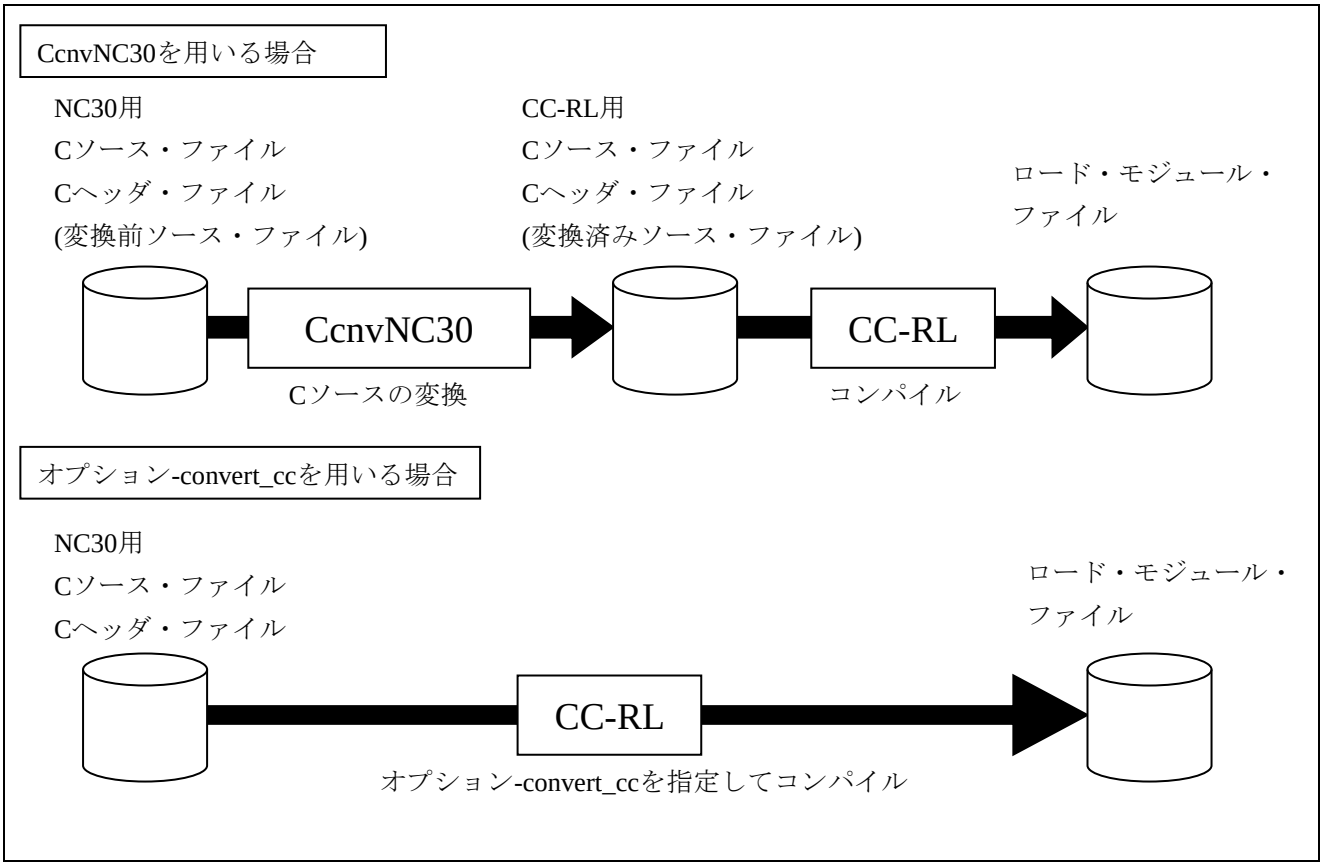
CcnvNC30 は，M16C シリーズ，R8C ファミリ用 C コンパイラである NC30 の開発環境で作成した C ソース・プログラムを，RL78 ファミリ用 C コンパイラである CC-RL で動作する C ソース・プログラムに変換する，C ソースコンバータです。C ソース中に記述した NC30 用拡張機能を，CC-RL 用拡張機能に変換します。

CC-RL には，C ソース中の NC30 用拡張機能をコンパイラ内部で CC-RL 用拡張機能に変換するオプション-convert_cc があります。変換対象のプログラムが保守対象であり，今後の変更も小規模である場合や，プログラムの移行による性能評価を行いたい場合は，CC-RL のオプション-convert_cc を使用してください。

CC-RL のオプション-convert_cc 指定時に手動による C ソースの修正が大量に必要となった場合や，今後もプログラムに機能追加があり，CC-RL 用としての C ソースが必要である場合は，CcnvNC30 を使用してください。

CcnvNC30 は，NC30 用プログラムから CC-RL 用プログラムへの移行を支援するためのソフトウェアです。変換後のプログラムが完全に動作することは保証しません。必ず変換後の C ソースを用いてプログラムの動作確認をしてください。

図 1.1 CcnvNC30 と CC-RL オプション-convert_cc の比較



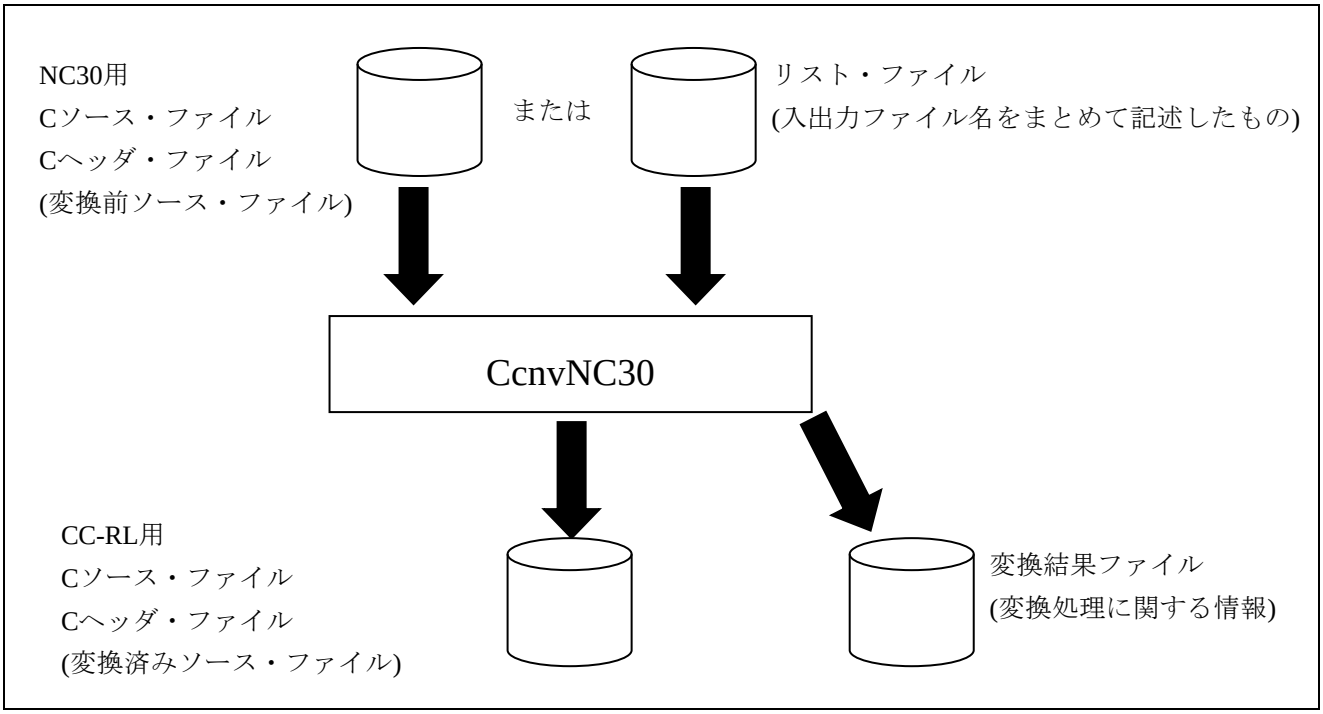
2. コマンド・リファレンス

この章では，CcnvNC30 における処理の流れについて説明します。

2.1 概要

NC30 用の C ソース・プログラムに対して，拡張言語仕様であるマクロ名，予約語，#pragma 指令，拡張機能の記述を，CC-RL の拡張言語仕様に変換し，CC-RL 用の C ソース・プログラムを生成します。

図 2.1 CcnvNC30 における処理の流れ



2.2 入出力ファイル

CcnvNC30 の入出力ファイルを以下に示します。

表 2.1 入出力ファイル

ファイル種別	入出力	拡張子	説明
C ソース・ファイル C ヘッダ・ファイル	入出力	(入力) .c .h (出力) 任意	NC30 用の C ソース・ファイルまたは C ヘッダ・ファイルを入力し，変換した CC-RL 用の C ソース・ファイルまたは C ヘッダ・ファイルを出力します。変換後のファイルは，先頭に CcnvNC30 のバージョン情報をコメントとして挿入し，変換箇所は元の記述をコメントとして残します。 入力ファイルの拡張子は固定です。他の拡張子を持つファイルを指定した場合は，入力ファイルの内容を変換せずそのまま出力します。 変換後のファイルは-o オプションまたは-l オプションで指定します。 変換後のファイルを再入力した場合は，変換せずそのまま出力し，既に変換済みであることを通知します。
リスト・ファイル	入力	任意	入力ファイル名と出力ファイル名を記述したテキスト･ファイルです。 -l オプションでリスト・ファイルを指定することで，複数のソース・ファイルをまとめて変換することができます。リスト・ファイルの書式については，-l オプションを参照してください。
変換結果ファイル	出力	任意	標準出力に出力する変換結果の中で，メッセージについては，-r オプションで指定したファイルに出力することができます。 メッセージの内容については，「メッセージ」を参照してください。

入力ファイルと出力ファイルの例を示します。変換仕様の詳細については「コンバータ変換仕様」を参照してください。

(入力ファイル：input.c)

```
#pragma ADDRESS p0 00E0H /* Port P0 register */
char c;
void main(void)
{
    c = p0;
}
```

(出力ファイル：output.c)

```
/* NC30 C Source Converter Vx.xx.xx.xx [dd Mmm yyyy] */
/***********************************************************************
 DISCLAIMER
 This software is supplied by Renesas Electronics Corporation and is only
 intended for use with Renesas products. No other uses are authorized. This
 software is owned by Renesas Electronics Corporation and is protected under
 all applicable laws, including copyright laws.
 THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES REGARDING
 THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT
 LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
 AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY DISCLAIMED.
 TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
 ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
 FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR
 ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS AFFILIATES HAVE
 BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
 Renesas reserves the right, without notice, to make changes to this software
 and to discontinue the availability of this software. By using this software,
 you agree to the additional terms and conditions found by accessing the
 following link:
 http://www.renesas.com/disclaimer
 Copyright (C) yyyy Renesas Electronics Corporation. All rights reserved.
***********************************************************************/
//[CcnvNC30] #pragma ADDRESS p0 00E0H /* Port P0 register */
#pragma address p0=0x00E0
char c;
void main(void)
{
    c = p0;
}
```

2.3 変換結果

CcnvNC30 は変換結果を標準出力に出力します。出力形式は，以下の通りです。

```
メッセージ

入力ファイル名
    結果
    メッセージ個数
```

-l オプション指定時は，リスト・ファイルで指定したファイル数の分だけ，上記の出力を繰り返します。

「メッセージ」はエラーやワーニングなどがある場合に出力します。メッセージの出力形式については「メッセージ」を参照してください。-r オプション指定時は，メッセージを標準出力ではなく指定したファイルに出力します。
「入力ファイル名」はコマンドラインまたはリスト・ファイルで指定した入力ファイルです。
「結果」は以下のいずれかを表示します。

- 変更箇所がある場合

```
変換しました。
```

- 変更箇所がない場合

```
変更箇所はありませんでした。
```

- 変換したファイルを再度 CcnvNC30 に入力した場合

```
既に変換されています。
```

- エラーが発生した場合

```
変換に失敗しました。
```

「メッセージ個数」は出力したメッセージの個数をメッセージの種別ごとに表示します。

変換結果の例を示します。

(入力ファイル：input.c)

```
#pragma ADDRESS p0 00E0H /* Port P0 register */
char c;
void main(void)
{
    c = p0;
}
```

(標準出力)

```
NC30 C Source Converter Vx.xx.xx.xx [dd Mmm yyyy]

input.c(1):M0593113:[変更]#pragma address を CC-RL の形式に変更しました。
input.c(1):M0593146:[情報]R8C,M16C に依存した言語仕様です。

input.c
                変換しました。
                0 deleted, 0 inserted, 1 changed, 1 information
                Total warning(s) : 0
```

2.4 操作方法

コマンドラインで以下のように入力します。

CcnvNC30[△*option*]…[△*file*][△*option*]…

file : 入力ファイル名
option : オプション名
[ ] : [ ]内は省略可能
… : 直前の[ ]内のパターンの繰り返しが可能
{ } : ｜で区切られた項目を選択
△ : 1 個以上の空白

- 入力ファイル名およびオプションで指定するファイル名は，Windows で認められるものであれば指定可能です。
- 入力ファイル名およびオプションで指定するファイル名は, 絶対パスまたは相対パスでの指定が可能です。入力ファイル名およびオプションで指定するファイル名を，パスなしまたは相対パスで指定する場合は，カレント・フォルダを基準とします。
- 入力ファイル名およびオプションで指定するファイル名に空白を含める(パス名を含む)場合は，パス名を含むファイル名をダブルクォーテーションで囲んでください。
- 入力ファイル名およびオプションで指定するファイル名の長さは，Windows に依存します(259 文字まで)。
- 入力ファイル名を複数指定した場合はエラーとなります。複数の入力ファイル名を指定したい場合は，-l オプションを使用してください。
- 入力ファイルを指定した場合は，必ず出力ファイル名を指定する必要があります。コマンドラインで入力ファイルを指定した場合は，-o オプションで出力ファイルを指定してください。
- 同じオプションを複数回指定した場合はエラーとなります。

2.5 オプション

オプションについて説明します。

- オプションの大文字／小文字は区別します。
- パラメータとしてファイル名を指定する場合は，パス付き(絶対パス，または相対パス)での指定が可能です。パスなし，または相対パスで指定する場合は，カレント・フォルダを基準とします。
- パラメータ中に空白を含める場合(パス名など)は，そのパラメータ全体をダブルクォーテーションで囲んでください。

表 2.2 オプション

オプション	説明
-V	CcnvNC30 のバージョン情報を表示します。
-h	CcnvNC30 のオプションの説明を表示します。
-c	日本語の文字コードを指定します。
-l	リスト・ファイル名を指定します。
-o	出力ファイル名を指定します。
-r	メッセージの出力先を指定します。
-A	ANSI 規定に関する機能を有効にした状態で変換します。
-R8C	R8C 用の C ソースとみなして変換します。

-V

CcnvNC30 のバージョン情報を表示します。

［指定形式］

```
-V
```

・省略時解釈
CcnvNC30 のバージョン情報を表示しません。

［詳細説明］

- CcnvNC30 のバージョン情報を標準エラー出力に表示します。
- 本オプション指定時は変換しません。
- 他のオプションを同時に指定した場合は，他のオプションを無視します。

［使用例］

```
>CcnvNC30  -V
```

-h

CcnvNC30 のオプションの説明を表示します。

［指定形式］

```
-h
```

・省略時解釈
CcnvNC30 のオプションの説明を表示しません。

［詳細説明］

- CcnvNC30 のオプションの説明を標準エラー出力に表示します。
- 本オプション指定時は変換しません。
- 他のオプションを同時に指定した場合は，他のオプションを無視します。
- -V を同時に指定した場合は，-V を優先します。

［使用例］

```
>CcnvNC30  -h
```

-C

日本語の文字コードを指定します。

［指定形式］

```
-c={sjis | euc_jp}
```

・省略時解釈
本オプションのパラメータに sjis を指定したものと見なします。

［詳細説明］

- 入力ファイル中のコメントに対して，使用する文字コードを指定します。
- パラメータを省略した場合は，エラーとなります。
- パラメータに指定可能なものを以下に示します。これ以外のものを指定した場合は，ワーニングを出力し sjis として扱います。なお，入力ファイル中で使用している文字コードと異なるものを指定した場合，動作は保証されません。

sjis	SJIS
euc_jp	EUC(日本語)

［使用例］

```
>CcnvNC30  input.c  -c=euc_jp  -o=output.c
```

-l

リスト・ファイル名を指定します。

［指定形式］

-l=*file*

・省略時解釈
コマンドラインで指定したファイルを変換します。

［詳細説明］

- 複数のファイルを同時に変換する場合に指定します。
- 指定したリスト・ファイルが存在しない場合はエラーとなります。
- 本オプションを指定した場合，コマンドラインで指定したファイル名はワーニングを出力し無視します。
- -o オプションと同時に指定した場合，ワーニングを出力し，-o オプションを無視します。
- パラメータを省略した場合は，エラーとなります。
- リスト・ファイルの書式は以下の通りです。

```
[-c={sjis｜euc_jp}] [-A] 入力ファイル名 出力ファイル名
[-c={sjis｜euc_jp}] [-A] 入力ファイル名 出力ファイル名
(以下略)
```

[ ]　　：[ ]内は省略可能
{ }　　：｜で区切られた項目を選択

- 一行に-c オプション，-A オプション，入力ファイル名，出力ファイル名の順に指定します。
- -c オプション，-A オプションは省略可能です。入出力ファイル名は省略できません。
- 記述可能な入出力ファイル名はコマンドラインでの指定時と同じです。
- ファイル名に空白文字を含む場合は，ファイル名をダブルクォーテーションで囲んでください。
- -c オプションの指定がコマンドラインとリスト・ファイル内とで異なる場合，ワーニングを出力しリスト・ファイルの指定を優先します。
- 出力ファイルが既に存在する場合は，ワーニングを出さずに上書きします。
- 出力ファイル名が，入力ファイル名または-r オプションで指定したファイル名と一致する場合は，エラーとなります。
- リスト・ファイルの日本語文字コードは UTF-8N(BOM なし)のみ，改行コードは CR+LF のみ受容します。

［使用例］

```
>CcnvNC30   -l=listfile.txt
```

・リスト・ファイル(listfile.txt)の中身

```
-c=sjis  input¥file1.c  output¥file1.c

-c=sjis  input¥file2.c  output¥file2.c

-c=sjis  input¥file.h  output¥file.h
```

-o

出力ファイル名を指定します。

［指定形式］

-o=*file*

・省略時解釈
-V, -h, -l オプション指定時を除き，省略できません。省略した場合は，エラーとなります。

［詳細説明］

- 変換後の出力ファイル名を指定します。
- 指定したファイルが既に存在する場合は，ワーニングを出さずに上書きします。
- 出力ファイル名が入力ファイル名または-r オプションで指定したファイル名と一致する場合は，エラーとなります。
- -l オプションと同時に指定した場合，ワーニングを出力し，本オプションを無視します。
- パラメータを省略した場合は，エラーとなります。

［使用例］

```
>CcnvNC30    input.c    -o=output.c
```

-r

メッセージを指定したファイルに出力します。

［指定形式］

-r=*file*

・省略時解釈
メッセージを標準出力に出力します。

［詳細説明］

- メッセージを指定したファイルに出力します。
- 指定したファイルが既に存在する場合は，ワーニングを出さずに上書きします。
- 指定したファイル名が C ソース・ファイルや C ヘッダ・ファイルの入出力ファイル名と一致する場合は，エラーとなります。
- パラメータを省略した場合は，エラーとなります。

［使用例］

```
>CcnvNC30  input.c  -o=output.c  -r=input.txt
```

-A

NC30 の ANSI 規格準拠オプション-fansi を有効とした状態で変換します。

［指定形式］

```
-A
```

・省略時解釈
NC30 の ANSI 規格準拠オプション-fansi を無効とした状態で変換します。

［詳細説明］

- 本オプション指定時は，以下をキーワードと見なさず，変換しません。

```
far,  near,  inline,  asm
```

- 本オプションは，変換前の NC30 開発環境におけるオプション-fansi の有無に合わせてください。

［使用例］

```
>CcnvNC30  input.c  -o=output.c  -A
```

-R8C

R8C 用の C ソースとみなして変換します。

［指定形式］

```
-R8C
```

・省略時解釈
M16C 用の C ソースとみなして変換します。

［詳細説明］

- 本オプション指定時は，_far, far キーワードを_near, near キーワードとして扱います。
- 本オプション指定時は，#pragma SPECIAL を無効とし，メッセージを出力して削除します。
- 本オプションは，変換前の NC30 開発環境に合わせてください。

［使用例］

```
>CcnvNC30    input.c    -o=output.c    -R8C
```

3. コンバータ変換仕様

この章では，CcnvNC30 の変換仕様を示します。

- NC30 の構文的に誤った C ソースを入力した場合は，動作を保証しません。
- コメントおよび文字列・文字定数に含まれている内容については変換しません。
- コメントのネストに対応していません。ネスト構造のコメント文は正常に認識せず，コメントの範囲が不正となります。コメントのネストがないか，変換前に確認してください。
- 変換対象のキーワードが##演算子等で生成される等，キーワードとして見つけることができない場合は，変換できません。そのまま CC-RL でコンパイルすると，コンパイル・エラーとなります。変換キーワードに対する#define，typedef，##演算子がないかどうか，変換前に確認してください。
- デバイスに依存した記述は，変換後に手動で修正してください。
- C ソース中でインクルードしているファイルは変換しません。別途，変換してください。

下記の拡張言語仕様に対して変換します。

- マクロ名
- 予約語
- デフォルト引数
- ワイド文字列の連結および文字定数
- #pragma SPECIAL
- #pragma SECTION
- ASM 文
- 割り込み関数
- 絶対番地配置指定
- 組み込み関数
- その他の#pragma 指令
- 標準ライブラリ関数

3.1 マクロ名

NC30 でサポートしているマクロは，以下の通り変換します。CC-RL に対応するマクロがない場合は，メッセージを出力します。NC30 でのみサポートしている標準ライブラリマクロは変換せず，メッセージも出力しません。CC-RL ではユーザ定義マクロとして扱います。

表 3.1 マクロ名の変換

NC30 マクロ名	変換後	備考
__LINE__	変換しません	そのまま CC-RL で使用できます。
__FILE__	変換しません	そのまま CC-RL で使用できます。
__DATE__	変換しません	そのまま CC-RL で使用できます。
__TIME__	変換しません	そのまま CC-RL で使用できます。
__STDC__	変換しません	そのまま CC-RL で使用できます。
NC30	変換しません	メッセージを出力します。 CC-RL ではユーザ定義マクロとして扱います。
M16C	変換しません	メッセージを出力します。 CC-RL ではユーザ定義マクロとして扱います。
__R8C__	変換しません	メッセージを出力します。 CC-RL ではユーザ定義マクロとして扱います。
__cplusplus	変換しません	メッセージを出力します。 CC-RL ではユーザ定義マクロとして扱います。
MB_LEN_MAX	変換しません	<limits.h> メッセージを出力しません。 CC-RL ではユーザ定義マクロとして扱います。
LC_ALL LC_COLLATE LC_CTYPE LC_MONETARY LC_NUMERIC LC_TIME	変換しません	<locale.h> メッセージを出力しません。 CC-RL ではユーザ定義マクロとして扱います。
SIGABRT SIGFPE SIGILL SIGSEGV SIG_DFL SIG_ERR SIG_IGN	変換しません	<signal.h> メッセージを出力しません。 CC-RL ではユーザ定義マクロとして扱います。

NC30 マクロ名	変換後	備考
_IOFBF _IOLBF _IONBF BUFSIZ FILENAME_MAX FOPEN_MAX SEEK_CUR SEEK_END SEEK_SET TMP_MAX stderr stdin stdout	変換しません	<stdio.h> メッセージを出力しません。 CC-RL ではユーザ定義マクロとして扱います。
MB_CUR_MAX	変換しません	<stdlib.h> メッセージを出力しません。 CC-RL ではユーザ定義マクロとして扱います。
CLOCKS_PER_SEC	変換しません	<time.h> メッセージを出力しません。 CC-RL ではユーザ定義マクロとして扱います。

3.2 予約語

予約語に対する変換仕様を示します。

表 3.2 予約語の変換

NC30 の予約語	変換後	備考
wchar_t	変換しません	ファイルの先頭に typedef unsigned short wchar_t を出力します。最初に記述した wchar_t に対してメッセージを出力します。
_inline	__inline	
inline	__inline	-A オプションが無効の場合のみ変換します。
restrict	削除	
_ext4mptr	削除	
_near	__near	
near	__near	-A オプションが無効の場合のみ変換します。
_far	__far (-R8C なし) __near (-R8C あり)	
far	__far (-R8C なし) __near (-R8C あり)	-A オプションが無効の場合のみ変換します。

3.3 デフォルト引数

NC30 では C++の機能と同様に関数の引数にデフォルト値を定義できますが，CC-RL ではデフォルト引数を指定できません。CcnvNC30 はデフォルト引数を変換しませんが，メッセージを出力します。ただし，関数宣言時の型名に typedef 名を使用していて，記憶域クラス指定子，型指定子，型修飾子のいずれも検出できない場合は，デフォルト引数を正しく認識できません。

［例］

パターン	例	備考
パターン 1	`int func1(int a, char b=1);`	メッセージを出力します。
パターン 2 (変数名省略)	`int func2(int, int=2);`	メッセージを出力します。
パターン 3 (型指定子省略)	`int func3(unsigned a=3);`	メッセージを出力します。
パターン 4 (構造体)	`struct S {int i;} s;` `int func4(struct S param=s);`	メッセージを出力します。
パターン 5 (typedef 使用)	`typedef int INT16;` `int func5(INT16 param=2);`	デフォルト引数と認識できません。 メッセージを出力しません。

3.4 ワイド文字列の連結および文字定数

単純文字列リテラル(例："abc")とワイド文字列リテラル(例：L"def")が隣り合っている場合，NC30 と CC-RL とでは定数の扱いが異なります。また，2 文字以上の文字定数(例：'ab'，L'あい')や，ワイド文字に対する単純文字定数(例：'あ')は，NC30 と CC-RL とでは定数の扱いが異なります。これらの場合に CcnvNC30 はメッセージを出力しますが，変換はしませんので，メッセージ出力箇所に対して手動で修正してください。

- 単純文字列リテラルとワイド文字列リテラルが隣り合っている場合

〈NC30〉

それぞれの型に合わせることなく，そのまま結合します。

L"abc""def"	00H	61H	00H	62H	00H	63H	64H	65H	66H	00H				
"abc"L"def"	61H	62H	63H	00H	64H	00H	65H	00H	66H	00H	00H			

〈CC-RL〉

単純文字列リテラルとワイド文字列リテラルを結合した場合，ワイド文字列とします。

L"abc""def"	00H	61H	00H	62H	00H	63H	00H	64H	00H	65H	00H	66H	00H	00H
"abc"L"def"	00H	61H	00H	62H	00H	63H	00H	64H	00H	65H	00H	66H	00H	00H

- 2 文字以上の文字定数

文字定数の値が異なります。

	NC30	CC-RL
'ab'	0x61	0x6162
'あ'	0xa0	0x82a0
L'ab'	0x6162	0x0061
L'あい'	0x82a2	0x82a0

3.5 #pragma SPECIAL

スペシャルページサブルーチン呼び出し関数は，callt 関数に置き換えます。

NC30 の書式は以下の通りです。

#pragma SPECIAL **呼び出し番号 関数名** **または** #pragma SPECIAL **関数名**(vect=**呼び出し番号**)

CC-RL の書式は以下の通りです。

#pragma callt [( ] 関数名 [, …] [ )]

- -R8C オプション指定時は，#pragma SPECIAL を削除します。
- callt 関数は最大 32 個指定できます。#pragma SPECIAL の指定が 32 個を超える場合は，変換後にコンパイル・エラーとなります。

［例］

パターン 1 （-R8C なし）	**変換前**	#pragma SPECIAL 20 func
	変換後	#pragma callt func
パターン 2 （-R8C なし）	**変換前**	#pragma SPECIAL func(vect=20)
	変換後	#pragma callt func
パターン 3 （-R8C あり）	**変換前**	#pragma SPECIAL func(vect=20)
	変換後	//[CcnvNC30] #pragma SPECIAL func(vect=20)

3.6 #pragma SECTION

#pragma SECTION は，NC30 と CC-RL とでセクション名が異なるため，セクション名の変換処理が必要になります。ただし，セクションによっては CC-RL 側に対応するセクションが存在しないケースもあるため，変換が不可能なセクションも存在します。また，変換は可能であっても機能が少し異なるものもあります。CcnvNC30 ではいくつかのセクションの変換時にメッセージを出力します。詳細はセクション名対応表を参照してください。

NC30 の書式は以下の通りです。

#pragma SECTION **セクション名 変更後セクション名**

CC-RL の書式は以下の通りです。

#pragma section [{text ｜ const ｜ data ｜ bss}] [**変更後セクション名**]

- CC-RL の#pragma section におけるセクション名は「変更後セクション名+"_n"」または「変更後セクション名+"_f"」，saddr 領域用のセクション名は「変更後セクション名+"_s"」となります。詳細は CC-RL のユーザーズ・マニュアルを参照してください。
- CC-RL 側に対応するセクションがないため変換できない場合は，メッセージを出力し，変換しません。CC-RL ではワーニングを出力し，#pragma 指令を無視します。後述のセクション名対応表に従い，C ソースを修正してください。

[例]

パターン 1 **(正常に置換)**	**変換前**	#pragma SECTION rom MY_DATA
	変換後	#pragma section const MY_DATA
パターン 2 **(置換不可能)**	**変換前**	#pragma SECTION interrupt MY_CODE
	変換後	#pragma section interrupt MY_CODE
	対応	**CC-RL に対応するセクションがないため，変換せず出力します。** **セクション名対応表に従い，修正してください。**

表 3.3 セクション名対応表

NC30 セクション名	説明	CC-RL セクション 種別	CcnvNC30 の動作 変換後の対応
program	コード部用セグメント	text	対応したセクション種別に変更します。
			不要です。 CC-RL でのセクション名は「変更後セクション名+"_n"」 または「変更後セクション名+"_f"」となります。
data	初期値ありデータ用セグメント	data	変換しません。
			CC-RL でのセクション名は「変更後セクション名+"_n"」 または「変更後セクション名+"_f"」となります。 ROM から RAM へマップするセクションは，リンク・オプション-ROm で指定してください。
bss	初期値なしデータ用セグメント	bss	変換しません。
			不要です。 CC-RL でのセクション名は「変更後セクション名+"_n"」 または「変更後セクション名+"_f"」となります。
rom	ROM データ用セグメント	const	対応したセクション種別に変更します。
			不要です。 CC-RL でのセクション名は「変更後セクション名+"_n"」 または「変更後セクション名+"_f"」となります。
interrupt	互換用	なし	変換しません。
			#pragma を削除してください。 CC-RL ではセクション名を変更できません。

3.7 ASM 文

NC30 では_asm( ), asm( )関数や#pragma asm～#pragma endasm を用いて関数中にアセンブリ記述をしますが, CC-RL では#pragma inline_asm で宣言したアセンブリ記述関数をインライン展開します。CcnvNC30 では，_asm( ), asm( )や#pragma asm～#pragma endasm 内のアセンブラ命令を実行する inline_asm 関数をファイル先頭に作成し，アセンブラ命令の記述箇所ではこの関数を呼び出すように変換します。

NC30 の書式は以下の通りです。

```
#pragma asm
  :     /* アセンブリ記述 */
#pragma endasm
```

または

```
_asm(/* コメント */);
_asm("アセンブリ記述");
_asm("アセンブリ記述", 引数 1);
_asm("アセンブリ記述", 引数 1, 引数 2);
```

CC-RL の書式は以下の通りです。

```
#pragma inline_asm [(] 関数名 [, … ] [)]
関数宣言 {
  :     /* アセンブリ記述 */
}
```

- R8C や M16C と RL78 は命令セットや命令の仕様が異なるため，手動によるアセンブラ記述の修正が必要です。変換時にメッセージを出力します。
- inline_asm 関数内のアセンブリ記述に，インデントとしてタブ文字を追加します。
- 作成する関数名は__inline_asm_func_00000～__inline_asm_func_99999 とし，関数の数が 100000 件を超える場合は，エラーとなります。
- _asm( )は常に変換します。asm( )は-A オプションを指定しない場合に変換します。
- #pragma asm～#pragma endasm や_asm( ), asm( )の中にラベルがある場合には，メッセージを出力します。CC-RL にて，#pragma inline_asm を指定した関数の中にラベルを書くと，コンパイル時にエラーとなります。このため CcnvNC30 では，#pragma asm～#pragma endasm や_asm( ), asm( )の中にラベルがある場合には，メッセージを出力します。コンパイル・エラーを回避するには，アセンブリ記述のラベルをローカル・ラベルに変更する必要があります。詳細は，CC-RL のユーザーズ・マニュアルを参照してください。
- 以下のようにダブルクォートを含めて#define マクロの変換対象とする場合は，_asm( )関数から inline_asm 関数を生成できません。このような場合は，メッセージを出力し，入力ファイルの内容を変換せずそのまま出力します。予めマクロを展開した状態に修正してから変換してください。

  例） #define MAC "nop"
  _asm(MAC);

- _asm( )内の文字列に'¥n'や'¥t'といった制御文字を含む場合は，変換後にアセンブル・エラーとなります。予め制御文字を削除してから変換してください。
- #pragma asm～#pragma endasm 内のアセンブリ記述のコメント(";")にC記述のコメント("/*")を含む場合は，変換後にコメントの範囲が不正となります。予めコメントを削除してから変換してください。

［例］

パターン 1	変換前	void func(void)
		{
		  _asm("nop");
		}

	変換後	#pragma inline_asm __inline_asm_func_00000
		static void __inline_asm_func_00000(void)
		{
		  nop
		}
		
		void func(void)
		{
		  __inline_asm_func_00000();
		}

パターン 2	変換前	void func(void)
		{
		  #pragma asm
		  nop
		  #pragma endasm
		}

	変換後	#pragma inline_asm __inline_asm_func_00001
		static void __inline_asm_func_00001(void)
		{
		  nop
		}
		
		void func(void)
		{
		  __inline_asm_func_00001();
		}

パターン 3	変換前	#define ASM_NOP _asm("nop");

	変換後	#pragma inline_asm __inline_asm_func_00002
		static void __inline_asm_func_00002(void)
		{
		  nop
		}
		#define ASM_NOP __inline_asm_func_00002();

パターン 4
(変換後にエラー)
	変換前	void func()
		{
		  _asm("¥tnop");
		}

	変換後	#pragma inline_asm __inline_asm_func_00003
		static void __inline_asm_func_00003(void)
		{
		  ¥tnop
		}
		
		void func()
		{
		  __inline_asm_func_00003();
		}

3.8 割り込み関数

NC30 の#pragma interrupt を，CC-RL の#pragma interrupt に変換します。

NC30 の割り込み関数の書式は以下の通りです。

```
#pragma interrupt [/B | /E | /V] [割り込みベクタ番号] 関数名
または
#pragma interrupt [/B | /E] 関数名[(vect=割り込みベクタ番号)]
```

CC-RL の割り込み関数の書式は以下の通りです。

```
#pragma interrupt [(]関数名[([vect=アドレス][,bank=レジスタバンク][,enable={true|false}])][)]
関数宣言
```

- 「/B」は「bank=RB1」に，「/E」は「enable=true」に変換します。「/B」と「/E」を同時に指定した場合は，#pragma interrupt を削除します。
- 「/V」を指定した場合は，#pragma interrupt を削除します。
- 割り込みベクタ番号を指定した場合は，「(vect=アドレス)」に変換し，#include "iodefine.h"を別途出力します。NC30 の割り込みベクタ番号の値はそのまま CC-RL のアドレスとなりますが，デバイス変更により正しく動作しないため，メッセージを出力します。手動で値を修正してください。
- NC30 では同一関数に対し#pragma interrupt を複数記述できますが，CC-RL ではコンパイル・エラーとなります。重複する#pragma 指令を手動で削除してください。

［例］

パターン 1	変換前	#pragma interrupt /V func
	変換後	//[CcnvNC30] #pragma interrupt /V func
パターン 2	変換前	#pragma interrupt /B /E 10 func
	変換後	//[CcnvNC30] #pragma interrupt /B /E func(vect=10)
パターン 3	変換前	#pragma interrupt /E 10 func void func(void) { }
	変換後	#pragma interrupt func(vect=10, enable=true) void func(void) { }
パターン 4	変換前	#pragma interrupt /B func(vect=10) void func(void) { }
	変換後	#pragma interrupt func(vect=10, bank=RB1) void func(void) { }
パターン 5 （変換なし）	変換前	#pragma interrupt func
	変換後	#pragma interrupt func

3.9 絶対番地配置指定

NC30 の#pragma address を CC-RL の#pragma address に変換します。

NC30 の書式は以下の通りです。

#pragma address 変数名 配置アドレス

配置アドレスは 2,8,10,16 進数の表記が可能
2 進数なら数値の後ろに'B'または'b'を，8 進数なら数値の後ろに'O'または'o'を，
16 進数なら数値の後ろに'H'または'h'を付加
配置アドレスは「変数のアドレス+オフセット」のような式を記述可能

CC-RL の書式は以下の通りです。

#pragma address 変数名=配置アドレス

配置アドレスは C 言語表記の 2,8,10,16 進数を表記可能，式は記述不可

- 配置アドレスは，変換前と同じ基数で，C 言語表記に変換します。
- 配置アドレスに式を記述した場合は，メッセージを出力し変換しません。
- M16C,R8C と RL78 ではメモリマップが異なるため，メッセージを出力します。RL78 のメモリマップに合わせて手動で修正してください。
- デバイスが異なるため，SFR のアクセスは見直す必要があります。手動で修正してください。

［例］

パターン 1	変換前	#pragma address i 01010101b
	変換後	#pragma address i= 0b01010101
パターン 2	変換前	#pragma address i 002000O
	変換後	#pragma address i=02000
パターン 3	変換前	#pragma address i 500
	変換後	#pragma address i=500
パターン 4	変換前	#pragma address i 400h
	変換後	#pragma address i=0x400
パターン 5	変換前	#pragma address i 0400H + _OFFSET
	変換後	#pragma address i 0400H + _OFFSET

3.10 組み込み関数

NC30 の組み込み関数は CC-RL と機能が一致しないため，メッセージを出力し CC-RL の組み込み関数に変換しません。CC-RL ではユーザ関数として扱います。

表 3.4 NC30 の組み込み関数

NC30 組み込み関数	備考
abs_b, abs_w, dadc_b, dadc_w, dadd_b, dadd_w, div_b, div_w, divu_b, divu_w, divx_b, divx_w, mod_b, mod_w, modu_b, modu_w, not_b, not_w, neg_b, neg_w, dsbb_b, dsbb_w, movll, movlh, movhl, movhh, rmpa_b, rmpa_w, smovf_b, smovf_w, sha_b, sha_w, sha_l, shl_b, shl_w, shl_l, smovb_b, smovb_w, sstr_b, sstr_w, rolc_b, rolc_w, rorc_b, rorc_w, rot_b, rot_w	いずれも変換しません。 メッセージを出力します。

3.11 その他の#pragma 指令

その他の#pragma 指令に対する変換仕様を示します。

表 3.5 その他の#pragma 指令の変換

NC30 #pragma 指令	変換後	備考
#pragma rom	削除します	CC-RL ではサポートしていません。
#pragma struct	削除します	CC-RL ではサポートしていません。
#pragma ext4mptr	削除します	CC-RL ではサポートしていません。
#pragma bitaddress	削除します	CC-RL ではサポートしていません。
#pragma intcall	削除します	CC-RL ではサポートしていません。
#pragma parameter	削除します	CC-RL ではサポートしていません。
#pragma __asmmacro	削除します	CC-RL ではサポートしていません。
#pragma jsra	削除します	CC-RL ではサポートしていません。
#pragma jsrw	削除します	CC-RL ではサポートしていません。
#pragma page	削除します	CC-RL ではサポートしていません。
#pragma stacksize	削除します	CC-RL ではサポートしていません。
#pragma istacksize	削除します	CC-RL ではサポートしていません。
#pragma creg	削除します	CC-RL ではサポートしていません。
#pragma sectaddress	削除します	CC-RL ではサポートしていません。
#pragma entry	削除します	CC-RL ではサポートしていません。
#pragma almhandler	削除します	CC-RL ではサポートしていません。
#pragma inthandler	削除します	CC-RL ではサポートしていません。
#pragma handler	削除します	CC-RL ではサポートしていません。
#pragma cychandler	削除します	CC-RL ではサポートしていません。
#pragma task	削除します	CC-RL ではサポートしていません。
#pragma bit	削除します	CC-RL ではサポートしていません。
#pragma sbda	削除します	CC-RL ではサポートしていません。

3.12　標準ライブラリ関数

NC30 にはCC-RL ではサポートしていない標準ライブラリ関数があります。いずれの関数に対しても，CcnvNC30 はメッセージを出力せず，変換しません。CC-RL ではユーザ関数として扱います。

- NC30 用標準ライブラリのヘッダ・ファイルを CcnvNC30 で変換して，CC-RL で用いないでください。CC-RL 用標準ライブラリのヘッダ・ファイルを使用してください。
- CC-RL の標準ライブラリは引数や戻り値の型に__near/__far が指定されているため，変換後に引数や戻り値の型が合わなくなる場合があります。CC-RL のユーザーズ・マニュアルをご確認の上，手動で修正してください。

表 3.6 標準ライブラリ関数の変換

NC30 関数名	変換後	備考
<locale.h> setlocale, localeconv	いずれも変換しません	CC-RL ではサポートしていません。 CC-RL ではユーザ関数として扱います。
<signal.h> signal, raise	いずれも変換しません	CC-RL ではサポートしていません。 CC-RL ではユーザ関数として扱います。
<stdio.h> fflush, fprintf, fscanf, vfprintf, vsprintf, fgetc, fgets, fputc, fputs, getc, putc, ungetc, fread, fwrite, clearerr, feof, ferror	いずれも変換しません	CC-RL ではサポートしていません。 CC-RL ではユーザ関数として扱います。
<stdlib.h> exit, mblen, mbtowc, wctomb, mbstowcs, wcstombs	いずれも変換しません	CC-RL ではサポートしていません。 CC-RL ではユーザ関数として扱います。
<string.h> strcoll, strxfrm	いずれも変換しません	CC-RL ではサポートしていません。 CC-RL ではユーザ関数として扱います。
bcopy, bzero, memicmp, stricmp, strnicmp	いずれも変換しません	CC-RL ではユーザ関数として扱います。

3.13 CC-RL オプション-convert_cc との変換仕様差分

CC-RL にて NC30 のソースコードをコンパイルためのオプション「-convert_cc=nc30」指定時と，CcnvNC30 による変換とで動作が異なる拡張機能について，差分を示します。

表 3.7 CC-RL オプション-convert_cc=nc30 と異なる動作

NC30 の拡張機能	オプション-convert_cc=nc30 指定時の動作	CcnvNC30 の変換
#pragma interrupt	CC-RLと異なる書式で記述した場合は，#pragma指令を削除し警告メッセージを出力します。 手動で割り込みベクタ番号をRL78用に修正する必要があります。	CC-RL の書式に変換します。デバイス間で割り込みベクタの仕様が異なるため，メッセージを出力します。 手動で割り込みベクタ番号を RL78 用に修正する必要があります。
#pragma asm ： #pragma endasm	#pragma指令を削除し警告メッセージを出力します。	#pragma inline_asm と関数定義を出力し，#pragma asm～#pragma endasm は新しく生成した関数呼び出しに変換します。 M16C，R8C と RL78 では命令セットが異なるため，メッセージを出力します。手動でアセンブラ記述を RL78 用に修正する必要があります。
_asm( ), asm( )	通常の関数呼び出しと認識します。 手動で inline_asm 関数に修正する必要があります。 手動でアセンブラ記述を RL78 用に修正する必要があります。	各_asm( )と asm( )に対して#pragma inline_asm と関数定義を出力します。 _asm( )および asm( )呼び出しは，新しく生成した関数呼び出しに変換します。 M16C，R8C と RL78 では命令セットが異なるため，メッセージを出力します。手動でアセンブラ記述を RL78 用に修正する必要があります。
NC30 M16C __R8C__	マクロが有効となります。(空白を定義)	変換せず，メッセージを出力します。

4. メッセージ

この章では，CC-RL が出力するメッセージについて説明します。

4.1 出力形式

メッセージの出力形式は，以下の通りです。

・ファイル名と行番号を含む場合

- メッセージ番号の種別がインフォメーションの場合

```
ファイル名(行番号):メッセージ番号:[情報種別]メッセージ
```

情報種別は，変更・追加・削除・情報のいずれかとなります

- メッセージ番号の種別がインフォメーション以外の場合

```
ファイル名(行番号):メッセージ番号:メッセージ
```

・ファイル名と行番号を含まない場合

```
メッセージ番号:メッセージ
```

メッセージ番号は，1 文字の英字(メッセージ種別)，0593，3 桁の数字が連続した文字列として出力されます。

4.2 メッセージ種別

メッセージ種別は，以下のように分類されています。

表 4.1 メッセージ種別

種別	先頭の文字	説明
内部エラー	C	処理を中止します。 変換後 C ソースを出力しません。
エラー	E	処理を中止します。 変換後 C ソースを出力しません。
ワーニング	W	処理を継続します。 変換後 C ソースを出力します。
インフォメーション	M	処理を継続します。 変換後 C ソースを出力します。

4.3 情報種別

メッセージ番号の種別がインフォメーションの場合，情報種別は以下のように分類されています。

表 4.2 情報種別

情報種別	説明
変更	CC-RL 用の C ソースにするため，内容を変更しました。
追加	CC-RL 用の C ソースにするため，内容を追加しました。
削除	CC-RL では記述しないため，内容を削除しました。
情報	NC30 と CC-RL との仕様の違いにより，十分な変換ではない場合があります。 個別に確認してください。

4.4 メッセージ一覧

CcnvNC30 が出力するメッセージは，以下の通りです。

4.4.1 内部エラー

表 4.3 内部エラー

番号	メッセージ	説明
C0593*nnn*	内部エラーが発生しました	特約店，または当社までご連絡ください。

nnn は 3 桁の 10 進数字です。

4.4.2 エラー

表 4.4 エラー

番号	メッセージ	説明
E0593001	複数の入力ファイルが指定されています。	入力ファイルは 1 つしか指定できません。 入力ファイルを複数指定したい場合は，リスト・ファイルを使用してください。
E0593002	*option* オプションに引数は指定できません。	引数を指定できないオプションに引数を指定しています。
E0593003	*option* オプションに引数を指定してください。	引数が必要なオプションに引数を指定していません。
E0593004	*option* オプションが複数回指定されています。	オプションは同時に 1 つしか指定できません。
E0593005	出力ファイルが指定されていません。	入力ファイルに対応する出力ファイルが指定されていません。
E0593006	入力ファイル *file* の読込みに失敗しました。	フォルダ名またはファイル名に誤りがある可能性があります。リスト・ファイルに次のファイルが指定されている倍は，次の変換に移ります。
E0593007	変換結果ファイル *file* の書込みに失敗しました。	フォルダ名に誤りがある可能性があります。
E0593008	出力ファイル *file* の書込みに失敗しました。	フォルダ名に誤りがある可能性があります。
E0593009	リスト・ファイル *file* の読込みに失敗しました。	フォルダ名に誤りがある可能性があります。
E0593010	リスト・ファイル *file* の構文が認識できません。	リスト・ファイルの記述が正しくありません。
E0593011	ファイル名が重複しています。	入力ファイル，出力ファイル，変換結果出力ファイルのファイル名が重複しています。
E0593012	ファイル名が不正です。	コマンドライン指定時の入力ファイル名，もしくは，リスト・ファイル指定時の入出力ファイル名のいずれかが 260 文字を超えています。
E0593013	*option* オプションに指定された引数が不正です。	引数の指定が不正です。 または，指定したファイル名が 260 文字を超えています。

番号	メッセージ	説明
E0593101	*string* の書式が不正です。	NC30 で許可されていない文法があるため，変換できません。入力ファイルを修正してください。
E0593102	アセンブラ埋め込みインライン関数をこれ以上追加できません。	アセンブラ埋め込みインライン関数の作成上限数を超えました。入力ファイルを修正してください。
E0593103	テンポラリ・ファイルの削除に失敗しました。	テンポラリ・ファイルの削除に失敗しました。テンポラリ・ファイルを削除してください。

4.4.3 ワーニング

表 4.5 ワーニング

番号	メッセージ	説明
W0593051	"-l"オプションが指定されたので，入力ファイルの指定は無視されます。	リスト・ファイル指定時は，コマンドラインで入力ファイルを同時に指定できません。-l オプションで指定されたリスト・ファイルの変換が実行され，入力ファイルを無視します。
W0593052	"-c"の指定が，リスト・ファイルとコマンドラインで異なります。コマンドラインの指定は無視されます(*file*)。	リスト・ファイル中で指定された入力ファイル *file* に対応する-c オプションの指定がリスト・ファイルとコマンドラインで異なります。リスト・ファイルの指定で変換します。
W0593053	不正な *option* オプションが指定されています。	不正なオプションが指定されています。オプションを無視します。
W0593054	*option* オプションに指定された引数(*argument*)が不正です。	オプションに指定された引数が不正です。 -c オプションの引数が不正な場合，デフォルト時の指定で処理します。
W0593055	入力ファイル名が指定されていませんでした。この変換を無視します。	-l オプションで指定されたリスト・ファイルに入力ファイルの指定がない箇所があります。
W0593151	*string* を CC-RL の形式に変更できませんでした。	*string* を CC-RL の形式に変更できませんでした。入力ファイルを修正してください。

4.4.4 インフォメーション

表 4.6 インフォメーション

番号	情報種別	メッセージ	説明
M0593111	変更	*string1* を *string2* に変換しました。	字句を変換しました。
M0593113	変更	*string* を CC-RL の形式に変更しました。	記述形式が NC30 と CC-RL とで異なるため，CC-RL の記述形式に変更しました。
M0593123	追加	*string* を生成しました。	CC-RL の形式に合わせて，記述を追加しました。
M0593124	追加	アセンブラ埋め込みインライン関数 *string* を生成しました。	アセンブラ埋め込みインライン関数を生成しました。
M0593131	削除	*string* を削除しました。	CC-RL にない記述形式です。記述を削除しました。
M0593142	情報	対応するセクションが無いため，変換を行いません(*section*)。	CC-RL に対応するセクションが存在しないため，変換できませんでした。
M0593144	情報	対応するマクロが存在しないので，変換しません(*MACRO*)。	CC-RL に対応するマクロが存在しないため，変換できませんでした。
M0593145	情報	アセンブラ命令中にラベルの使用が検出されました。ラベルの変換はできませんので，適切な内容に修正してください。	CC-RL のアセンブリ記述関数内にはローカル・ラベルしか書けません。ラベルを適切な内容に修正してください。
M0593146	情報	R8C,M16C に依存した言語仕様です。	R8C, M16C から RL78 へのデバイス変更の際に見直しが必要です。手動で修正してください。

5. 注意事項

下記の項目に該当する場合, 変換後のC ソースはCC-RL で正しくコンパイルできないことがあります。

表 5.1 注意事項

項番	項目	CcnvNC30 の動作	変換結果に対するCC-RL の動作	参照先
1	ネスト構造のコメント文がある場合	正常に変換できない場合があります。	コメントの範囲が不正となります。	3 章
2	##演算子等の使用により，キーワードを検出できない場合	メッセージを出力せず，変換しません。	エラーE0520065 等になります。	3 章
3	#pragma SECTION のセクション名に, CC-RL には存在しないセクション名を指定している場合	メッセージを出力せず，変換しません。	ワーニングW0523037 を出力し，#pragma 指令を無視します。セクションの配置に失敗し，期待した動作とならない可能性があります。	3.6 節
4	_asm("文字列"), asm("文字列")中の文字列に¥n や¥t を使用している場合	制御文字をそのまま出力します。	エラーE0550249 になります。	3.7 節
5	#pragma asm～#pragma endasm 内のアセンブリ記述のコメント(";"以降の記述)内に「/*」が含まれている場合	アセンブリ記述のコメントをそのまま出力します。	アセンブリ記述のコメント(";")よりも C 記述のコメント("/*")を優先し，コメントの範囲が不正となります。	3.7 節
6	_asm( ), asm( )，または #pragma asm～#pragma endasm のアセンブリ記述にラベルが含まれる場合	メッセージを出力します。	エラーE0550213 になります。	3.7 節 CC-RL ユーザーズ・マニュアル #pragma inline_asm の[制限]
7	ASM 文を記述した場合	メッセージを出力します。#pragma inline_asm 関数内にそのまま出力します。	命令セットが異なるため，アセンブル・エラーになります。	3.7 節

項番	項目	CcnvNC30 の動作	変換結果に対する CC-RL の動作	参照先
8	割り込み関数に割り込みベクタ番号を指定した場合	メッセージを出力し，「(vect=アドレス)」に変換します。	R8C や M16C と RL78 とで割り込みの仕様が異なるため，期待した動作とならない可能性があります。	3.8 節
9	#pragma address を使用した場合	メッセージを出力し，CC-RL の書式に変換します。	R8C や M16C と RL78 とでメモリマップが異なるため，期待した動作とならない可能性があります。	3.9 節

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2016.04.01	ー	初版発行

CcnvNC30 ユーザーズマニュアル

発行年月日　2016年4月1日　　Rev.1.00

発行　　ルネサス エレクトロニクス株式会社
〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

Colophon 3.0

CcnvNC30

ルネサス エレクトロニクス株式会社

R20UT3685JJ0100