# RL78 ソフトウェア置き換えガイド

## アセンブリ言語から C 言語へのソースコード移植 CC-RL

R01AN3954JJ0101
Rev. 1.01
2018.01.23

### 要旨

本アプリケーションノートでは、開発統合環境 CS+用のアセンブリ言語で記述されたプログラムを C 言語のインラインアセンブラ関数に置き換える方法について説明します。

移植例として、「RL78/G10　タイマ・アレイ・ユニット（インターバル・タイマ）　CC-RL」のアプリケーションノート（R01AN3074J）のアセンブリ言語で記述されたプログラムを利用します。

### 対象デバイス

RL78 ファミリ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

# 目次

# 1. アセンブリ言語から C 言語へのソースコード移植手順

開発統合環境 CS+用のアセンブリ言語で記述されたプログラムを C 言語のインラインアセンブラ関数に置き換えます。最初に、開発統合環境 CS+ 用 C コンパイラ CC-RL のコード生成ツールを利用して新規にプロジェクトを作成します。アセンブリ・ソースの定数、変数および関数は、それぞれ、C 言語の定数、変数およびインラインアセンブラ関数に置き換えます。

## 1.1 ソースコードの自動生成

開発統合環境 CS+ 用 C コンパイラ CC-RL のコード生成ツールでソースコードの自動生成を行います。置き換え元のアセンブリ・ソースコードを参照して、コード生成ツールを設定します。

(1) プロジェクト・ツリーの中からコード生成（設計ツール）の「クロック発生回路」をクリックします。（図 1.1 の A）

(2) 「端子割り当て設定」を行い、[確定する]ボタンをクリックします。（図 1.1 の B）

注意. 他の機能を設定するためには、端子割り当て設定を行う必要があります。なお、端子割り当て設定を一度確定させると、端子割り当て設定は変更できません。

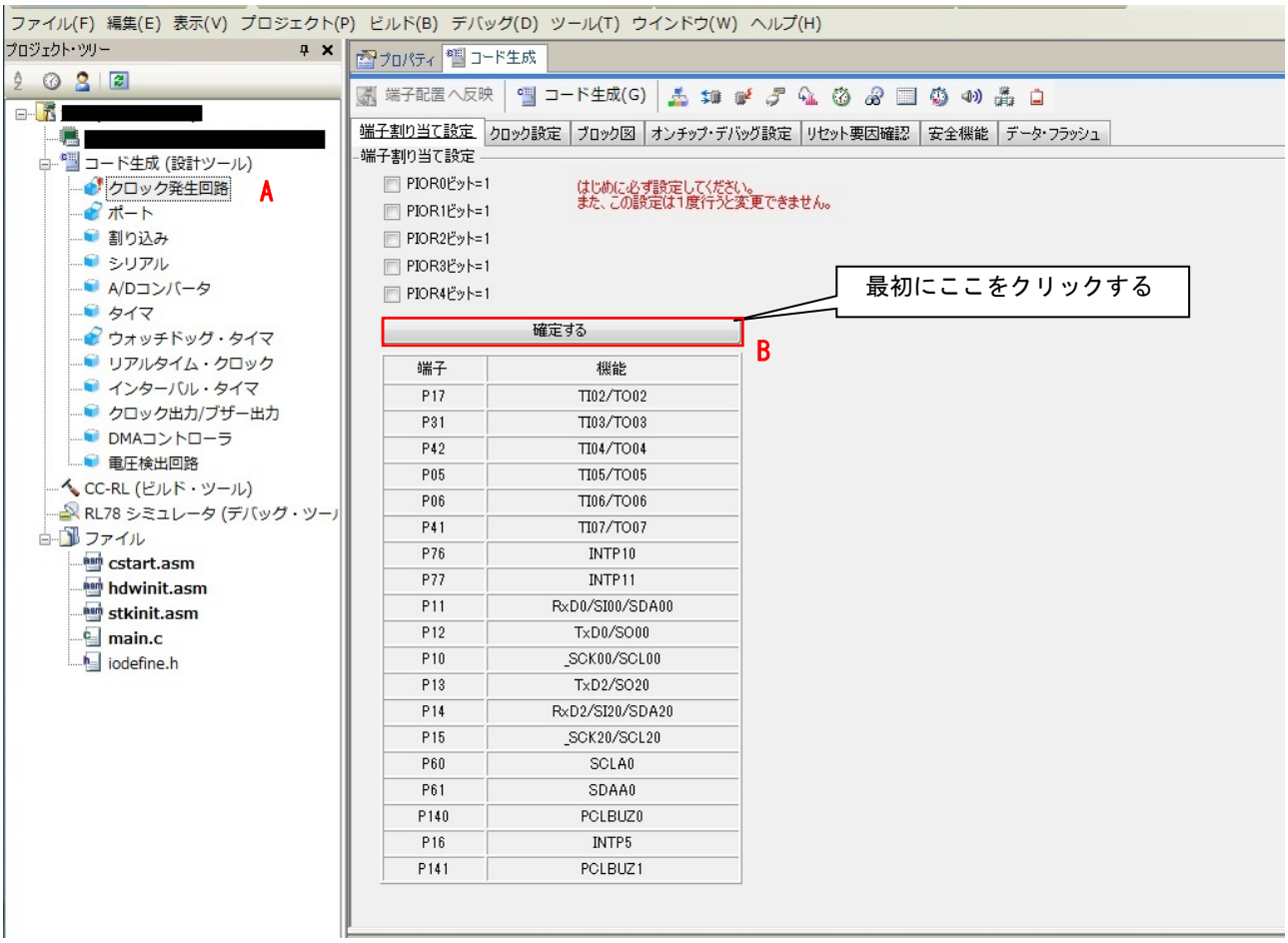


図 1.1 コード生成ツールの設定画面 (1)

(3)　置き換え元のアセンブリ言語で記述されたソースコードを参照して、各機能の設定をします。

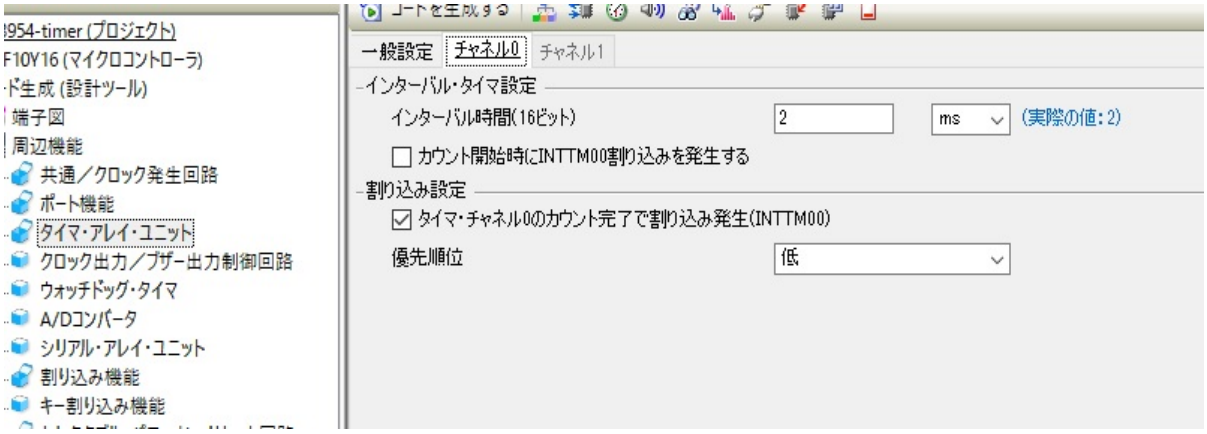


図 1.2　コード生成ツールの設定画面 (2)

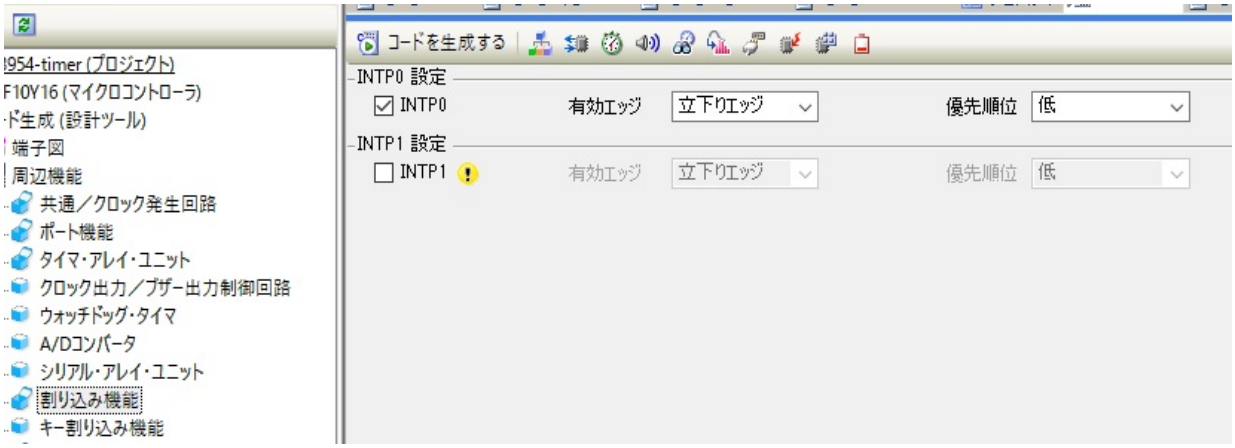


図 1.3 コード生成ツールの設定画面 (3)

(4)　全ての機能の設定を完了したら、画面上部にある[コード生成 (G)]ボタンをクリックして、コード生成（ソースコードの自動生成）を行います。（図 1.4 の C）


図 1.4　コード生成ツールの設定画面 (4)

## 1.2　定数、変数の定義

インラインアセンブラ関数内ではセクション定義が行えないため、C 言語で新たに定数、変数を定義します。（表 1.1、表 1.2　参照）

表 1.1　定数の変更

| アセンブリ・ソース<br>定数名 | C ソース<br>定数名 | 内容 |
|---|---|---|
| TINTVL | g_tdr00_data[] | スイッチ押下回数ごとの TDR00 への設定値 |
| T10MSWAIT | g_10ms_count[] | スイッチ押下回数ごとのタイマによる 10ms カウント値 |

表 1.2　グローバル変数の変更

| アセンブリ・ソース<br>変数名 | C ソース<br>変数名 | 内容 |
|---|---|---|
| RSWCNT | g_sw_counter | スイッチ押下回数カウンタ |
| RTMCNT | g_inttm00_counter | タイマ割り込み発生回数カウンタ |
| RTDR00 | g_tdr00_work | タイマ割り込み発生 250 回ごとの TDR00 への設定値 |

```
;***************************************************************************
;
;    data definition
;
;***************************************************************************
DMAIN       .DSEG   SBSS
RTMCNT:     .DS     1               ; counter of TM00 interrupt
RTM10MS:    .DS     1               ; counter for 10ms
RSWCNT:     .DS     1               ; counter of SW

DTDR        .DSEG   SBSS
RTDR00:     .DS     2               ; TDR00H,TDR00L data

CCHNGLED    .EQU    0x00000001      ; LED change data

;=============================================================
;    constant data for interval
;=============================================================
XMAIN2      .CSEG   TEXT

TINTVL:
            .DB2    PERIOD - 1      ; interval data for 2ms
            .DB2    PERIOD2 - 1     ; interval data for 1ms
            .DB2    PERIOD3 - 1     ; interval data for 0.5ms
            .DB2    PERIOD4 - 1     ; interval data for 0.25ms
            .DB2    PERIOD5 - 1     ; interval data for 0.125ms
```

図 1.5 アセンブリ・ソースの定義部

```
/***********************************************************************************************
Global variables and functions
***********************************************************************************************/
/* Start user code for global. Do not edit comment generated here */
__saddr uint8_t g_sw_counter = 0U;          /* Variable for counter of SW input */
__saddr uint16_t g_tdr00_work = 0U;         /* Variable of keeping next setting */
__saddr uint8_t ucchat;                     /* 8 bit variable for noise rejection */
/* Compare value table for interval timer */
const uint16_t g_tdr00_data[] =
        {
            (40000 - 1),                    /* 2ms interval compare value */
            (20000 - 1),                    /* 1ms interval compare value */
            (10000 - 1),                    /* 0.5ms interval compare value */
            (5000 - 1),                     /* 0.25ms interval compare value */
            (2500 - 1)                      /* 0.125ms interval compare value */
        };

/* 10ms wait count value table */
const uint16_t g_10ms_count[] =
        {
            (5 + 1),                        /* For 2ms interval */
            (10 + 1),                       /* For 1ms interval */
            (20 + 1),                       /* For 0.5ms interval */
            (40 + 1),                       /* For 0.25ms interval */
            (80 + 1),                       /* For 0.125ms interval */
        };

__saddr uint8_t g_inttm00_counter = 0U;    /* Variable for counter of INTTM00 */
/* End user code. Do not edit comment generated here */
```

図 1.6　C ソースの定義部

## 1.3　インラインアセンブラ関数の定義

アセンブリ・ソースを機能毎にインラインアセンブラ関数に置き換えるために、インラインアセンブラ関数を定義します。

インラインアセンブラ関数を利用する場合は、「#pragma inline_asm」を使用して定義します。

表 1.3　使用する関数（サブルーチン）一覧

| 関数名 | 概要 |
|---|---|
| Inline_asm_mainfunc | MAIN 処理 |
| r_invert_ledfunc | INTTM00 回数をカウントして 250 回ごとに LED 表示の反転処理 |
| r_inttm_func | INTTM00 割り込み時処理 |
| r_intp_func | INTP0 割り込み時処理 |

```
/***********************************************************************************
Pragma directive
***********************************************************************************/
/* Start user code for pragma. Do not edit comment generated here */
#pragma inline_asm inline_asm_mainfunc
#pragma inline_asm r_invert_ledfunc
#pragma inline_asm r_inttm_func
#pragma inline_asm r_intp_func
/* End user code. Do not edit comment generated here */
```

図 1.7　関数定義例

## 1.4　インラインアセンブラ関数内処理の移植

アセンブリ言語で記述されたソースコードを機能毎に「1.3　インラインアセンブラ関数の定義」で定義した関数へ移植します。

(1) アセンブリ・ソースのある機能（図 1.8の ①、図 1.10の②）をその機能に対応したインラインアセンブラ関数（図 1.9の①、図 1.11の②）へ移植します。

```
;*******************************************************************************
;
;    main function
;
;*******************************************************************************
main:
    CLRB    RTMCNT                  ; clear loop counter
    CLRB    RSWCNT                  ; clear SW counter
    MOVW    AX,     ES:!TINTVL      ; get initial interval data
    MOVW    RTDR00, AX              ; copy it to work area
    CALL    !!SSTARTINTV            ; start timer (interval)
    CLR1    PMK0                    ; enable INTP0
    EI                              ; enable interrupt

MAIN_LOOP:
    HALT
    BR      $MAIN_LOOP              ; continue to operation
```

①

図 1.8　移植元のアセンブリ・ソース①

```
*******************************************************************************/
/* main routine */
static void inline_asm_mainfunc(void)
{
;*******************************************************************************
;
;    main function
;
;*******************************************************************************
main:
    CLRB    RTMCNT                  ; clear loop counter
    CLRB    RSWCNT                  ; clear SW counter
    MOVW    AX,     ES:!TINTVL      ; get initial interval data
    MOVW    RTDR00, AX              ; copy it to work area
    CALL    !!SSTARTINTV            ; start timer (interval)
    CLR1    PMK0                    ; enable INTP0
    EI                              ; enable interrupt

MAIN_LOOP:
    HALT
    BR      $MAIN_LOOP              ; continue to operation
}
```

①

図 1.9　移植後の C ソースコード①

```
;***************************************************************************
;
;    interrupt function : INTTM00
;    occur every 2ms/1ms/0.5ms/0.25ms/0.125ms
;
;***************************************************************************
IINTTM00:
    PUSH    AX
    CALL    !SINTTM00               ; call actual blinking function routine
    POP     AX
    RETI
```

②

図 1.10　移植元のアセンブリ・ソース②

```
/***************************************************************************
* Function Name:r_inttm_func
* Description : This function interrupt function : INTTM00
* Arguments : none
* Return Value : none
***************************************************************************/
void r_inttm_func(void)
{
;***************************************************************************
;
;    interrupt function : INTTM00
;    occur every 2ms/1ms/0.5ms/0.25ms/0.125ms
;
;***************************************************************************
IINTTM00:
    PUSH    AX
    CALL    !SINTTM00               ; call actual blinking function routine
    POP     AX
    RETI
}
```

②

図 1.11　移植後の C ソースコード②

(2)　インラインアセンブラ関数内の変数、定数、関数名を C 言語で新たに定義した記述に修正します。
（図 1.12 の③、図 1.13の③）

(3)　CPU 制御命令を次のように置き換えます。（図 1.12 の④と⑤、図 1.13の④と⑤）

EI → ei、DI → di、HALT → halt、STOP → stop、NOP → nop

```
/* main routine */
static void inline_asm_mainfunc(void)
{
    MOV     ES,     #0              ; for constant data access
;*******************************************************************************
;
;    main function
;
;*******************************************************************************
main:
③   CLRB    RTMCNT                  ; clear loop counter
    CLRB    RSWCNT                  ; clear SW counter
    MOVW    AX,     ES:!TINTVL      ; get initial interval data
    MOVW    RTDR00, AX              ; copy it to work area
    CALL    !!SSTARTINTV            ; start timer (interval)
    CLR1    PMK0                    ; enable INTP0
④   EI                              ; enable interrupt

MAIN_LOOP:
⑤   HALT
    BR      $MAIN_LOOP              ; continue to operation
}
```

図 1.12 修正前の C ソースコード

```
/* main routine */
static void inline_asm_mainfunc(void)
{
    MOV     ES,     #0              ; for constant data access
;*******************************************************************************
;
;    main function
;
;*******************************************************************************
main:
③   CLRB    _g_inttm00_counter      ; clear loop counter
    CLRB    _g_sw_counter           ; clear SW counter
    MOVW    AX, ES:!_g_tdr00_data   ; get initial interval data
    MOVW    _g_tdr00_work, AX       ; copy it to work area
    CALL    !!R_TAU0_Channel0_Start ; start timer (interval)
    CLR1    PMK0                    ; enable INTP0
④   ei                              ; enable interrupt

MAIN_LOOP:
⑤   halt
    BR      $MAIN_LOOP              ; continue to operation
}
```

図 1.13　修正後の C ソースコード

(4) インラインアセンブラ関数内で特殊機能レジスタ（SFR）をアクセスする場合は、r_cg_macrodriver.h の”iodefine.h”のインクルードを外します。次に、SFR アクセスを使用する各 C ファイルで”iodefine.h” のインクルードを行います。

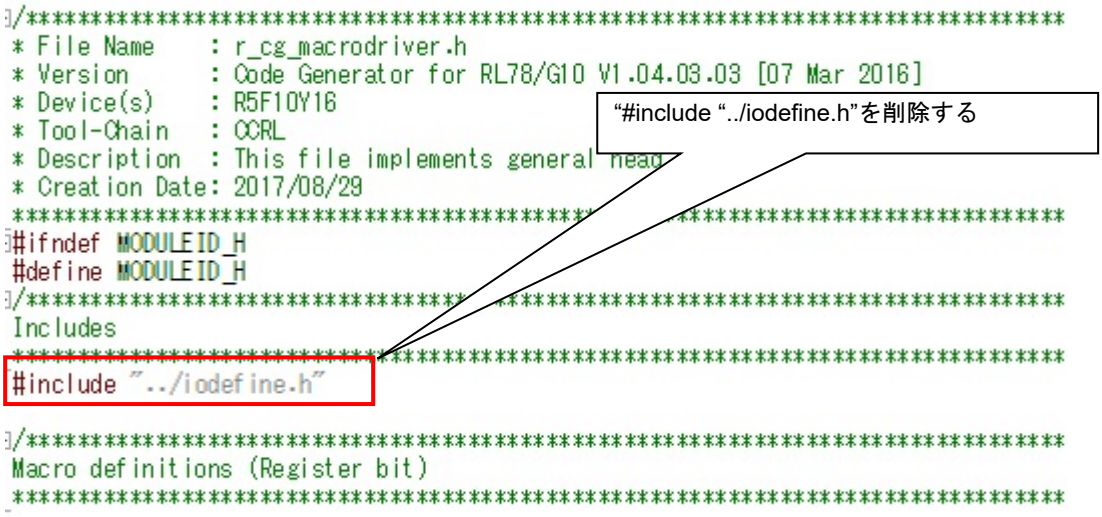


図 1.14 ”iodefine.h”の削除（r_cg_macrodiver.h）

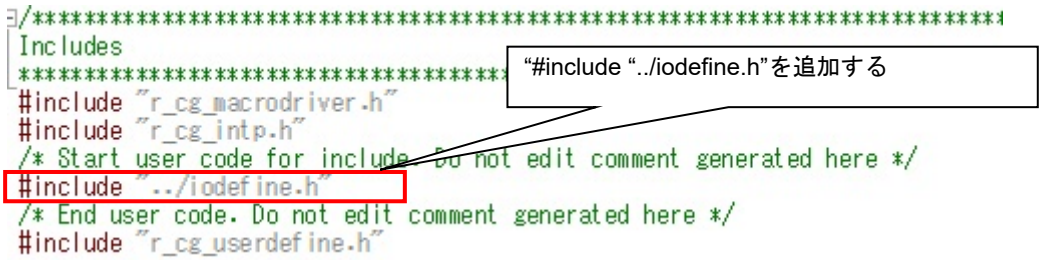


図 1.15 ”iodefine.h”の追加（r_cg_intp.c）

(5) 割り込み処理でインラインアセンブラ関数を呼び出す場合は、割り込みからの復帰命令が 2 重になるため、インラインアセンブラ関数内にある RETI 命令を削除します。

```
/***********************************************************************
* Function Name: r_tau0_channel0_interrupt
* Description  : This function INTTM00 interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void __near r_tau0_channel0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    r_inttm_func();
    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

図 1.16　インラインアセンブラ関数を呼び出した割り込み処理（r_cg_tau_user.c）

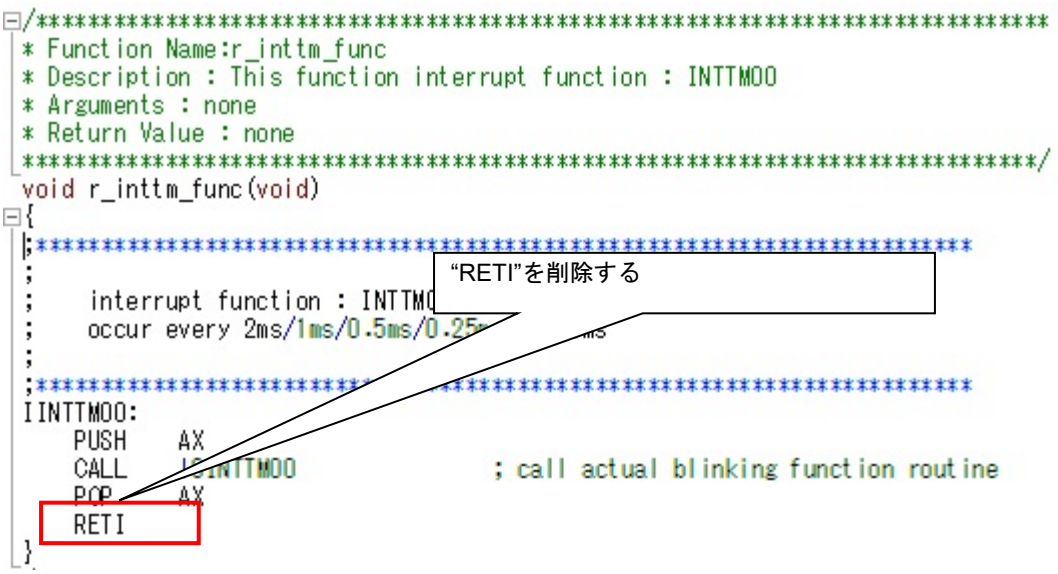


図 1.17　"RETI"の削除（r_cg_main.c）

## 1.5　main 関数からインラインアセンブラ関数を呼び出す

メイン関数（main()）に作成したインラインアセンブラ関数（inline_asm_mainfunc()）を追加します。

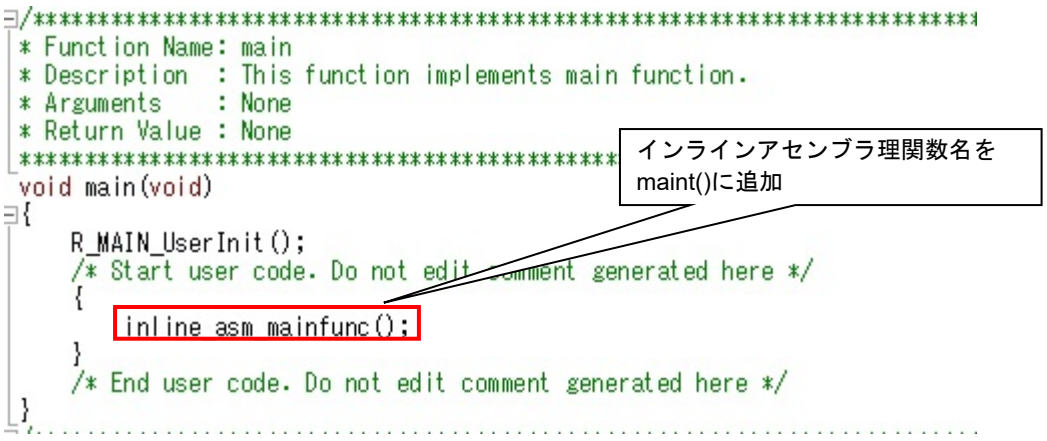


図 1.18 r_cg_main.c

以上で、アセンブリ言語から C 言語へのソースコード移植の準備ができました。

## 1.6　プロジェクトのビルド

CS＋のビルド（B）メニューから「ビルド・プロジェクト（B）」を選択して、プロジェクトのビルドを行ってください。

出力ウインドウに

“========= 終了しました(成功:1 プロジェクト, 失敗:0 プロジェクト)(xx 年 x 月 x 日 xx:xx:xx) =========”

と表示されたらビルド成功です。

もしエラーが出た場合は、エラーメッセージを元にプロジェクトのデバッグを行ってください。

## 2. サンプルコード

ルネサス エレクトロニクスのウェブサイトでサンプルコードを入手してください。

## 3. 参考ドキュメント

RL78/G10 初期設定 CC-RL（R01AN2668J）アプリケーションノート

RL78/G10 タイマ・アレイ・ユニット（インターバル・タイマ）CC-RL （R01AN3074J）アプリケーションノート

RL78/G10 ユーザーズマニュアル ハードウェア編 （R01UH0384J）

RL78 ファミリ ユーザーズマニュアル ソフトウェア編 （R01US0015J）

（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート/テクニカルニュース

（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

## ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com/

お問合せ先

http://japan.renesas.com/contact/

| 改訂記録 | RL78 ソフトウェア置き換えガイド<br>アセンブリ・ソースコードから C ソースコードへの移行 CC-RL |
|---|---|

| Rev. | 発行日 | 改訂内容 | |
|---|---|---|---|
| | | ページ | ポイント |
| 1.00 | 2017.11.30 | — | 初版発行 |
| 1.01 | 2018.01.23 | 9 | 図表のズレを修正 |

## 製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

# 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生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。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. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.4.0-1　2017.11)


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：https://www.renesas.com/contact/