RL78 Software Migration Guide

Source Code Migration from Assembly Language to C Language CC-RL

R01AN3954EJ0101
Rev. 1.01
Jan. 23, 2018

Introduction

This application note describes how to migrate the program in the assembly language for the CS+, which is the integrated development environment (IDE), to the inline assembler functions in the C language.

As a migration example, the sample program covered in the application note RL78/G10 Timer Array Unit (Interval Timer) CC-RL (R01AN3074E) is used.

Target Device

RL78 Family

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Procedure for Migrating Source Code from Assembly Language to C Language

The following describes the procedure for migrating the program in the assembly language for the IDE CS+ to the inline assembler functions in the C language. First of all, create a new project by using a code generation tool of the IDE CS+ C compiler CC-RL. Replace the constants, variables, and functions of the assembly source code with the constants, variables, and inline functions of the C language code, respectively.

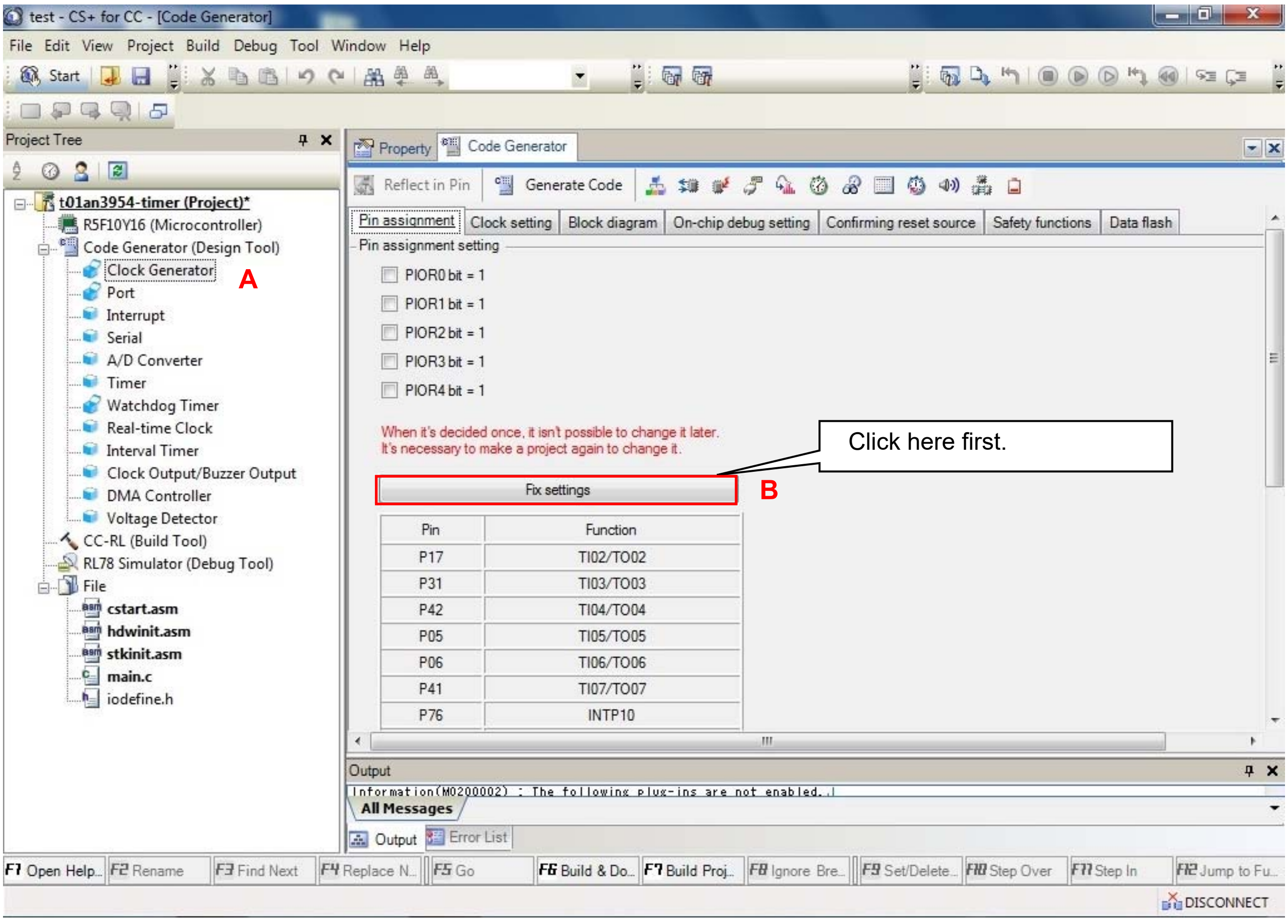
1.1 Automatic Source Code Generation

Source code can be automatically generated by using the code generation tool of the IDE CS+ C compiler CC-RL. Refer to the assembly source code to be replaced and set the code generation tool.

(1) Click "Clock Generator" under Code Generator (Design Tool) on the Project Tree pane (A in Figure 1.1).

(2) Perform "Pin assignment" and click the [Fix settings] button (B in Figure 1.1).

Note: To set the other functions, Pin assignment needs to be performed. Once Pin assignment is decided, it cannot be changed later.



Pin assignment setting

☐ PIOR0 bit = 1

☐ PIOR1 bit = 1

☐ PIOR2 bit = 1

☐ PIOR3 bit = 1

☐ PIOR4 bit = 1

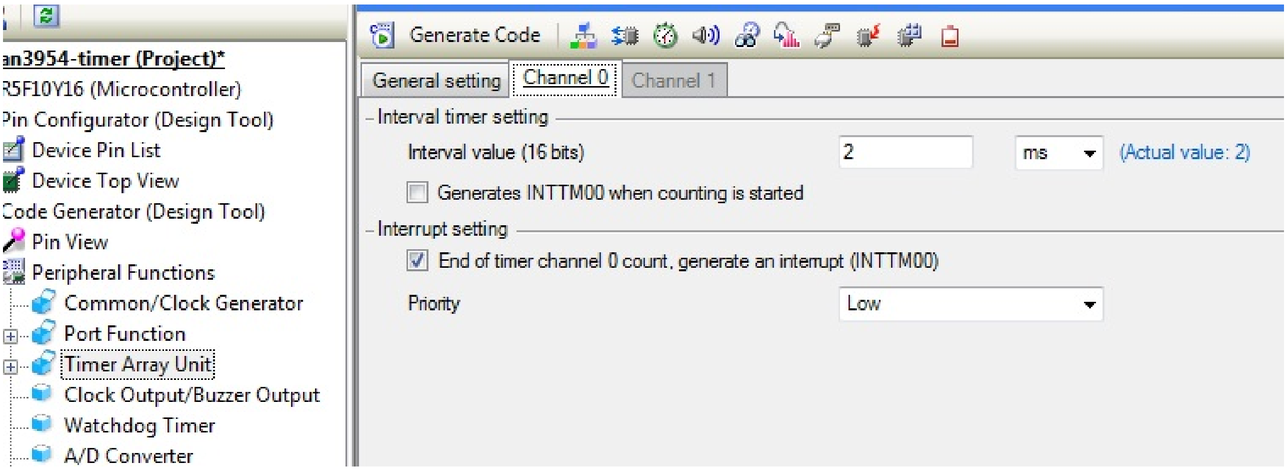
When it's decided once, it isn't possible to change it later.
It's necessary to make a project again to change it.

Pin	Function
P17	TI02/TO02
P31	TI03/TO03
P42	TI04/TO04
P05	TI05/TO05
P06	TI06/TO06
P41	TI07/TO07
P76	INTP10

```
Information(M0200002) : The following plug-ins are not enabled.
```

Figure 1.1 Code Generator Setting Window (1)

(3) Refer to the assembly source code to be replaced and set the other functions.



Interval timer setting

Interval value (16 bits) 2 ms (Actual value: 2)

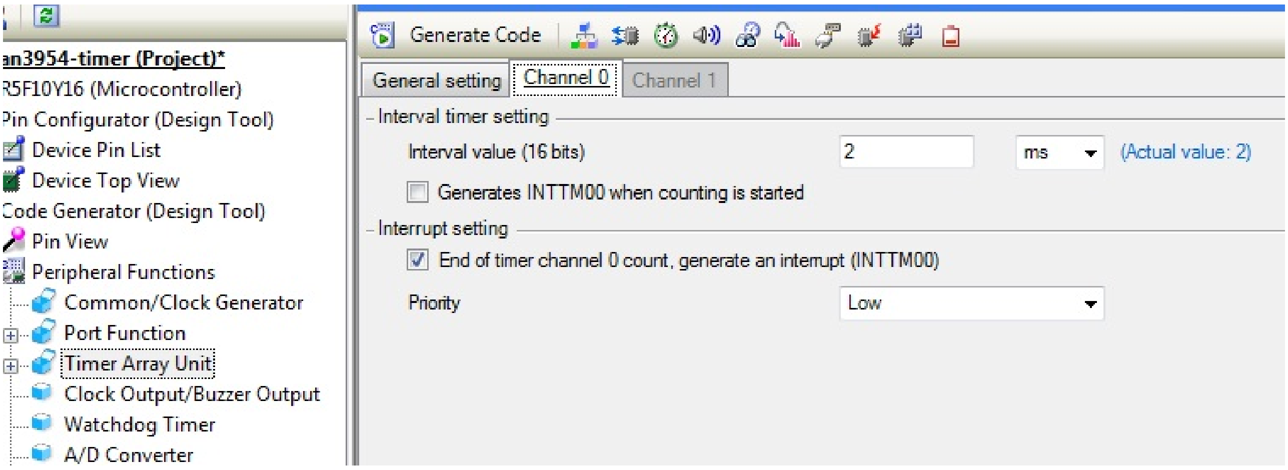
□ Generates INTTM00 when counting is started

Interrupt setting

☑ End of timer channel 0 count, generate an interrupt (INTTM00)

Priority Low

Figure 1.2 Code Generator Setting Window (2)



Interval timer setting

Interval value (16 bits) 2 ms (Actual value: 2)

□ Generates INTTM00 when counting is started

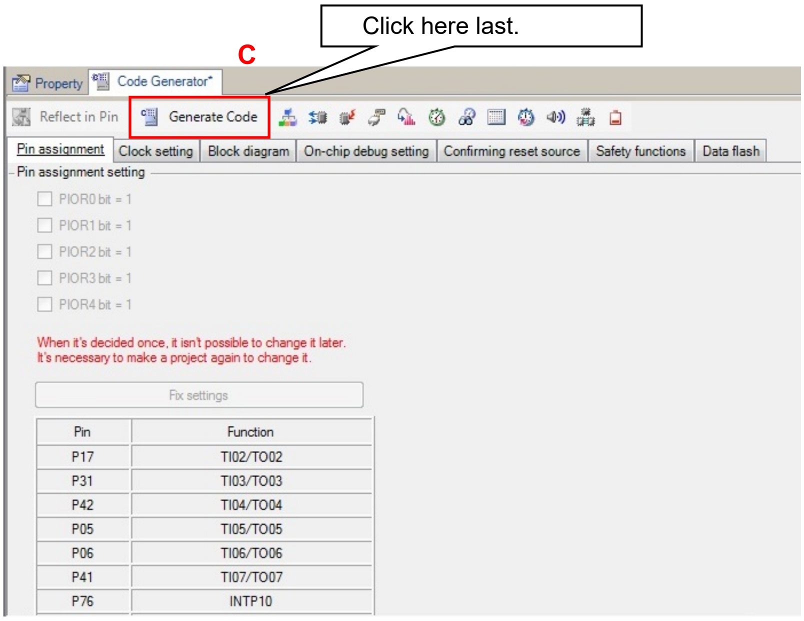
Interrupt setting

☑ End of timer channel 0 count, generate an interrupt (INTTM00)

Priority Low

Figure 1.3 Code Generator Setting Window (3)

When settings of all the functions are completed, click the [Generate Code] button on the upper part of the window to activate code generation (automatic generation of the source code) (C in Figure 1.4).



Pin assignment setting

□ PIOR0 bit = 1

□ PIOR1 bit = 1

□ PIOR2 bit = 1

□ PIOR3 bit = 1

□ PIOR4 bit = 1

When it's decided once, it isn't possible to change it later.
It's necessary to make a project again to change it.

Pin	Function
P17	TI02/TO02
P31	TI03/TO03
P42	TI04/TO04
P05	TI05/TO05
P06	TI06/TO06
P41	TI07/TO07
P76	INTP10

Figure 1.4 Code Generator Setting Window (4)

1.2 Definition of Constants and Variables

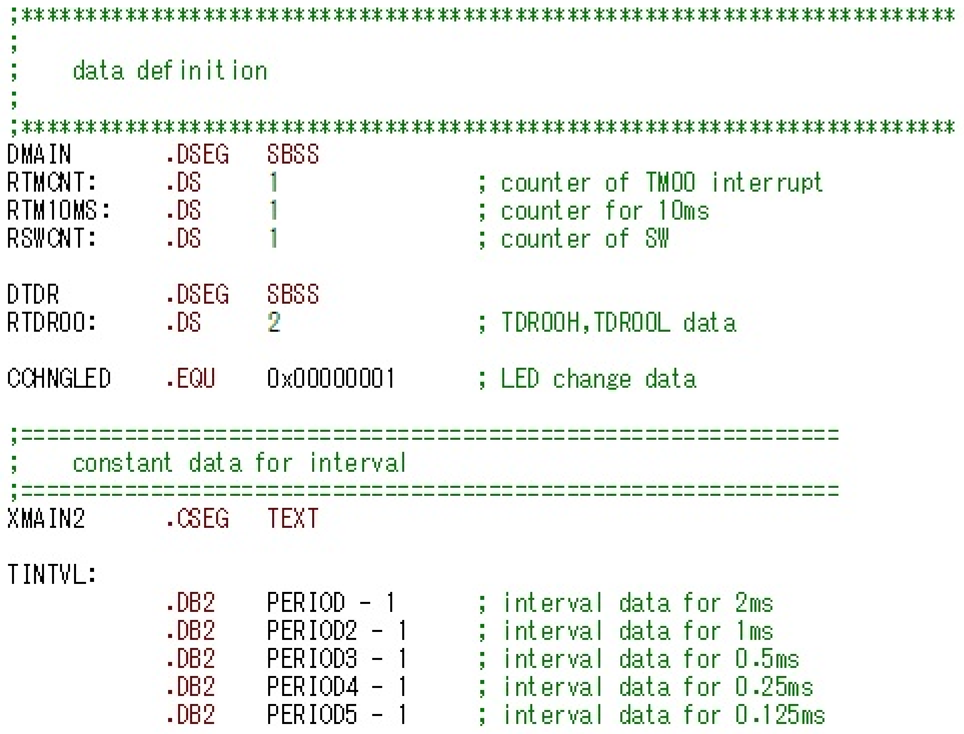
Since sections cannot be defined in the inline assembler functions, newly define the constants and variables in the C language. (Refer to Table 1.1 and Table 1.2.)

Table 1.1 Changing Constants

Constant Name in Assembly Source	Constant Name in C Source	Contents
TINTVL	g_tdr00_data[]	Value to be set to TDR00 each time the switch is pressed for the specified number of times
T10MSWAIT	g_10ms_count[]	10-ms count value by the timer each time the switch is pressed for the specified number of times

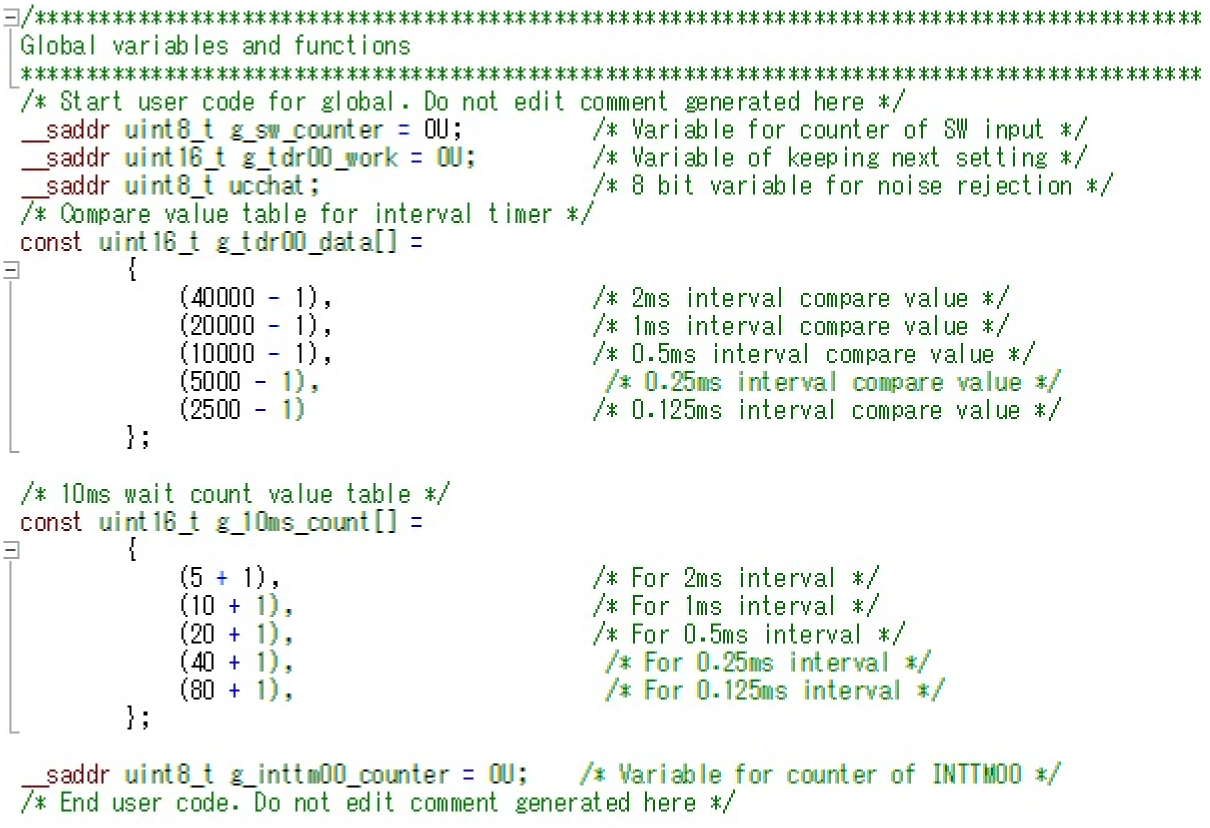
Table 1.2 Changing Global Variables

Constant Name in Assembly Source	Constant Name in C Source	Contents
RSWCNT	g_sw_counter	Counter for counting the number of times the switch is pressed
RTMCNT	g_inttm00_counter	Counter for counting the number of times the timer interrupt is generated
RTDR00	g_tdr00_work	Value to be set to TDR00 each time the timer interrupt is generated 250 times



```
;*************************************************************************
;
;    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
```

Figure 1.5 Definition Part in Assembly Source Code



```
/******************************************************************************************
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 */
```

Figure 1.6 Definition Part in C Language Source Code

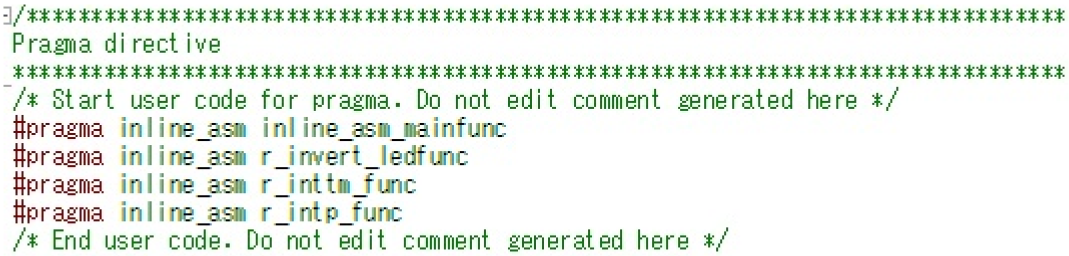
1.3 Definition of Inline Assembler Functions

To replace the functional units in the assembly source code with the corresponding inline assembler functions, define the inline assembler functions.

When using the inline assembler functions, define them with "#pragma inline_asm".

Table 1.3 List of Functions (Subroutines) Used

Function Name	Outline
Inline_asm_mainfunc	MAIN processing
r_invert_ledfunc	Counts the number of INTTM00 generated and reverses the LED display every 250 times.
r_inttm_func	Processes INTTM00 interrupt generated.
r_intp_func	Processes INTP0 interrupt generated.



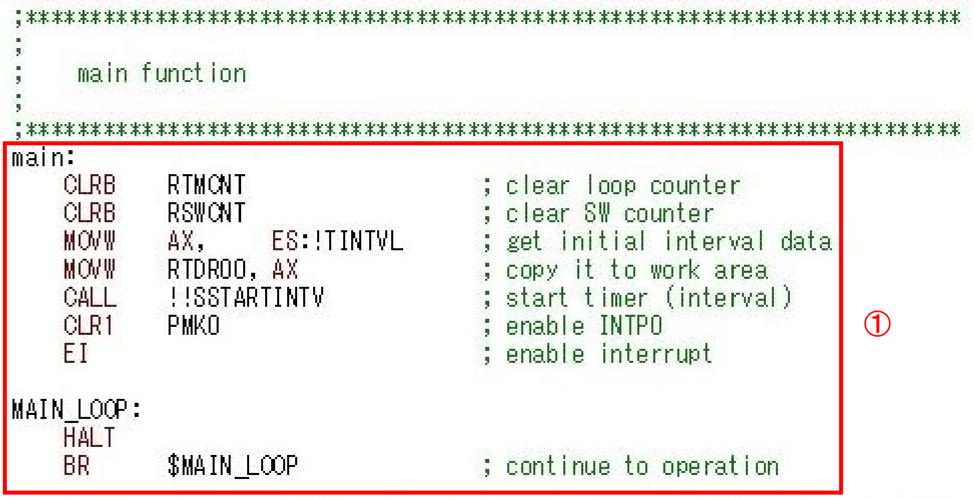
```
/*******************************************************************************
Pragma directive
*******************************************************************************/
/* Start user code for pragma. Do not edit comment generated here */
#pragma inline_asm inline_asm_mainfunc
#pragma inline_asm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

Figure 1.7 Example of Function Definition

1.4 Migration of Processes in Inline Assembler Functions

Migrate the functional units in the assembly source code to the corresponding functions having been defined in 1.3, Definition of Inline Assembler Functions.

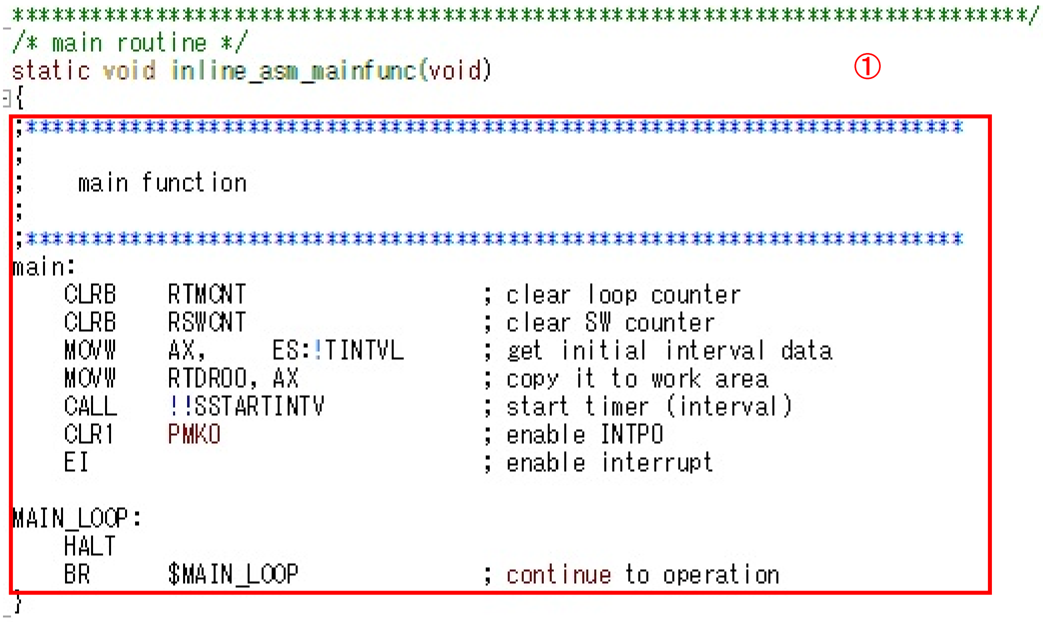
(1) Migrate certain functional units in the assembly source code (① in Figure 1.8 and ② in Figure 1.10) to the corresponding inline assembler functions (① in Figure 1.9 and ② in Figure 1.11).



```
;**********************************************************************
;
;    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

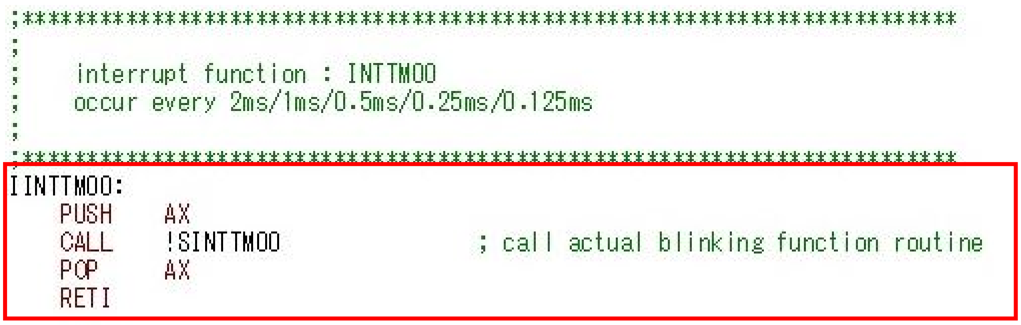
Figure 1.8 Assembly Source Code to be Migrated ①



```
***********************************************************************/
/* 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

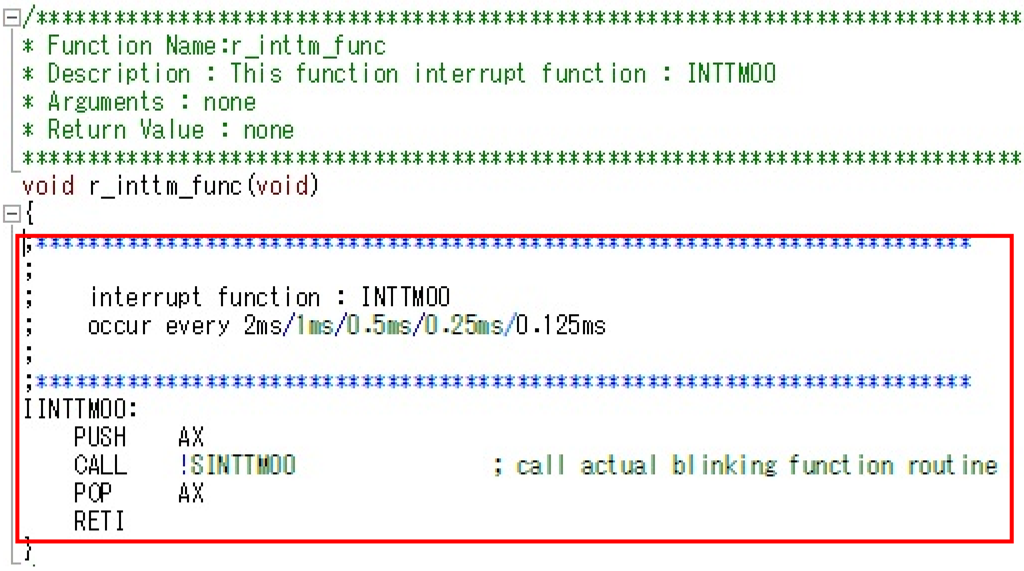
Figure 1.9 C Source Code after Migration ①



```
;***************************************************************************
;
;    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

Figure 1.10 Assembly Source Code to be Migrated 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
}
```

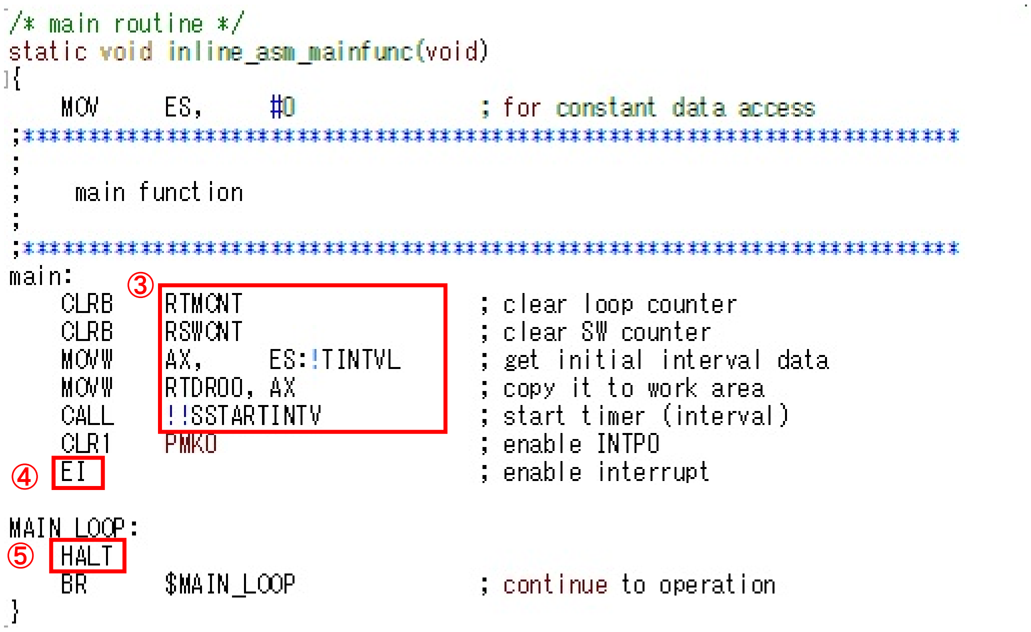
②

Figure 1.11 C Source Code after Migration ②

(2) Modify the names of the variables, constants, and functions of the inline assembler functions to the newly defined descriptions in C (③ in Figure 1.12 and ③ in Figure 1.13).

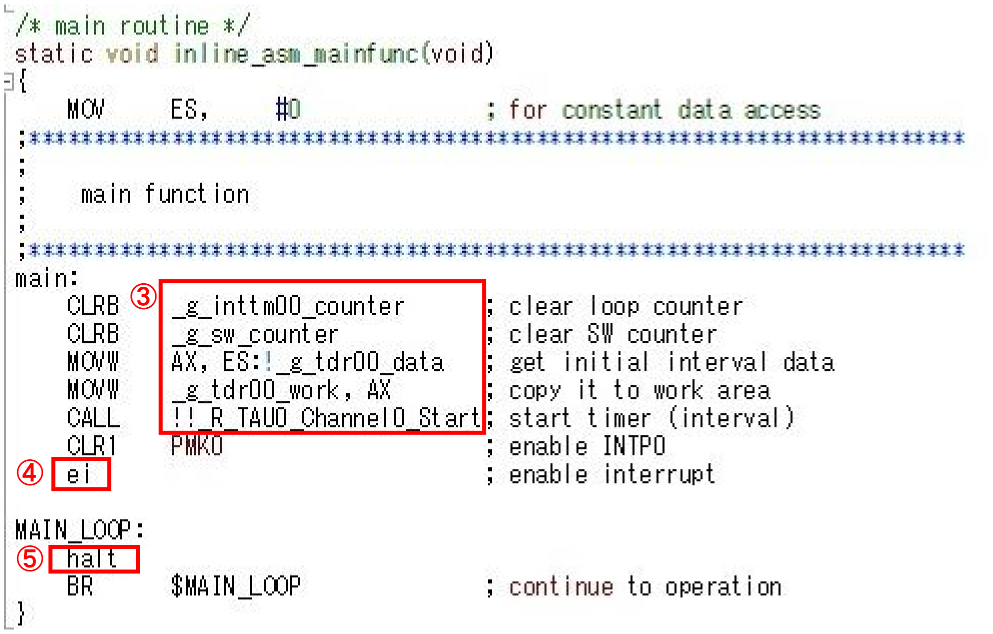
(3) Replace the CPU control instructions as described below (④ and ⑤ in Figure 1.12 and ④ and ⑤ in Figure 1.13).

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:
    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
}
```

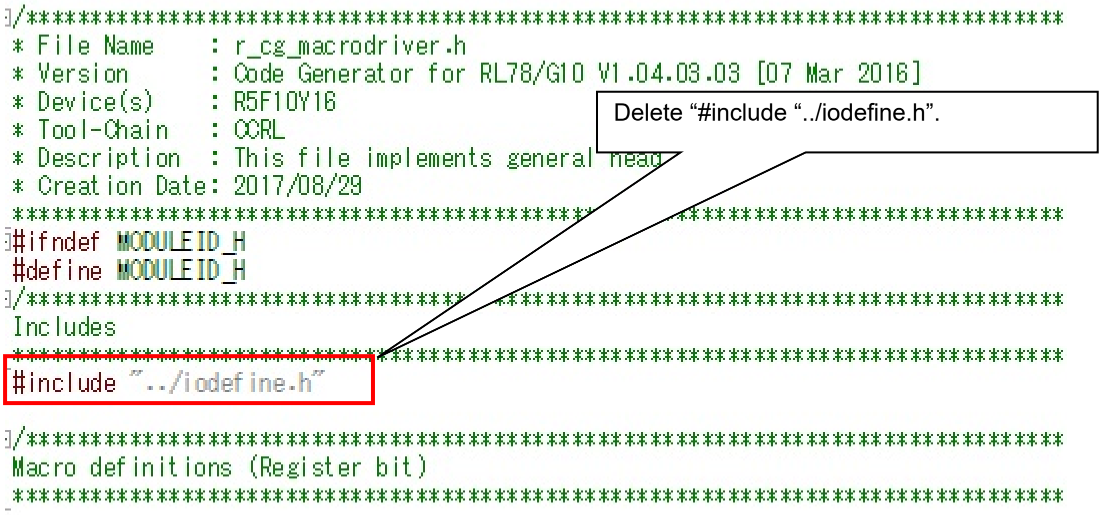
Figure 1.12 C Source Code before Modification



```
/* main routine */
static void inline_asm_mainfunc(void)
{
    MOV     ES,     #0              ; for constant data access
;***********************************************************************
;
;    main function
;
;***********************************************************************
main:
    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
    ei                              ; enable interrupt

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

Figure 1.13 C Source Code after Modification

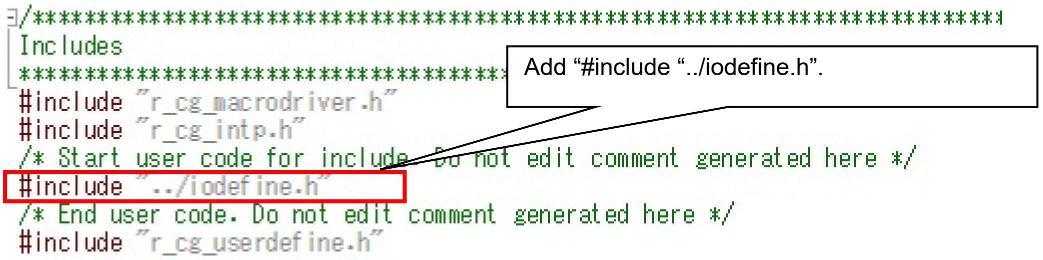
(4) When accessing the special function register (SFR) in the inline assembler functions, first exclude "iodefine.h" included in the r_cg_macrodriver.h. Then, include "iodefine.h" in each of the C files in which the SFR is accessed.



```
/*******************************************************************************
* File Name    : r_cg_macrodriver.h
* Version      : Code Generator for RL78/G10 V1.04.03.03 [07 Mar 2016]
* Device(s)    : R5F10Y16
* Tool-Chain   : CCRL
* Description  : This file implements general head
* Creation Date: 2017/08/29
*******************************************************************************/
#ifndef MODULEID_H
#define MODULEID_H
/*******************************************************************************
Includes
*******************************************************************************/
#include "../iodefine.h"

/*******************************************************************************
Macro definitions (Register bit)
*******************************************************************************/
```

Delete “#include “../iodefine.h”.

Figure 1.14 Deleting "iodefine.h" (r_cg_macrodiver.h)

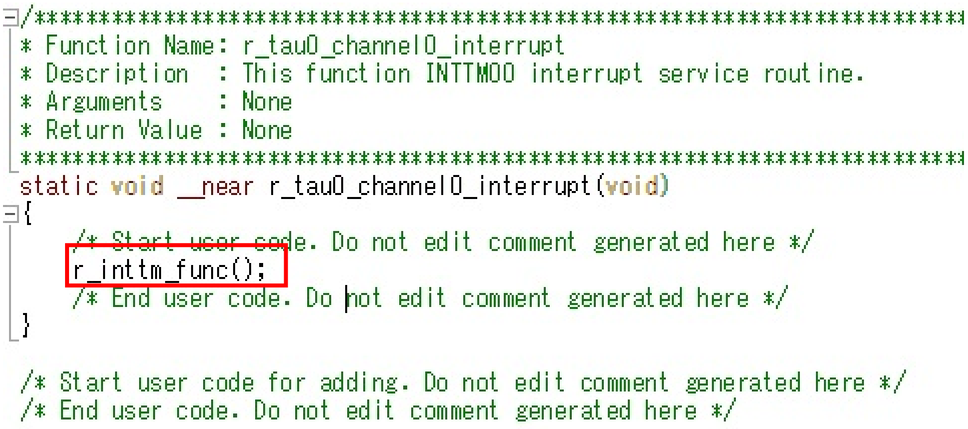


```
/*******************************************************************************
Includes
*******************************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_intp.h"
/* Start user code for include. Do not edit comment generated here */
#include "../iodefine.h"
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"
```

Add “#include “../iodefine.h”.

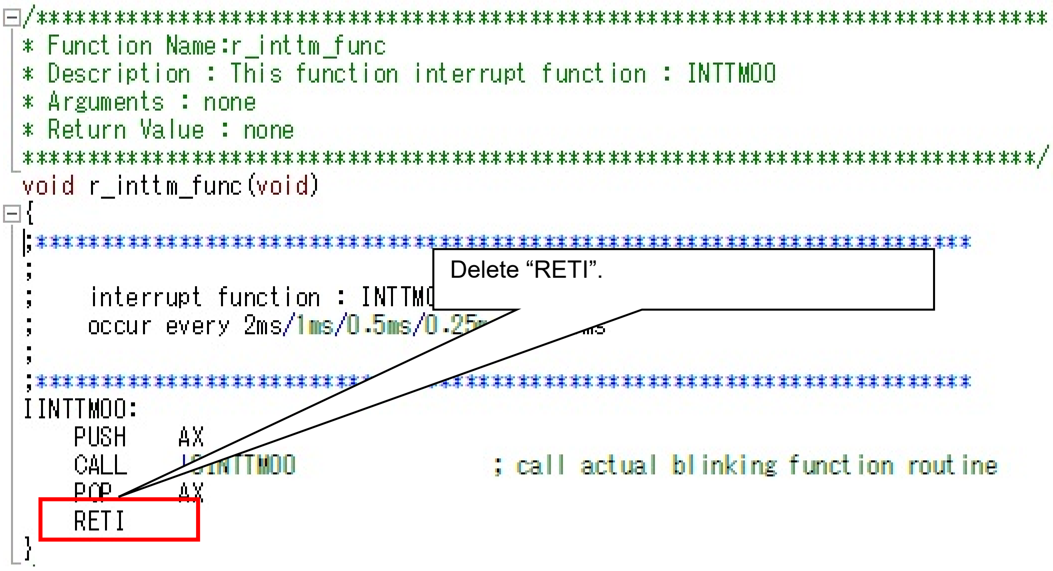
Figure 1.15 Adding "iodefine.h" (r_cg_intp.c)

(5) When calling the inline assembler function in the interrupt process, delete the RETI instruction from the inline assembler function to prevent redundancy of the return instructions from the interrupt process.



```
/***********************************************************************
* 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 */
```

Figure 1.16 Interrupt Process which has Called Inline Assembler Function (r_cg_tau_user.c)



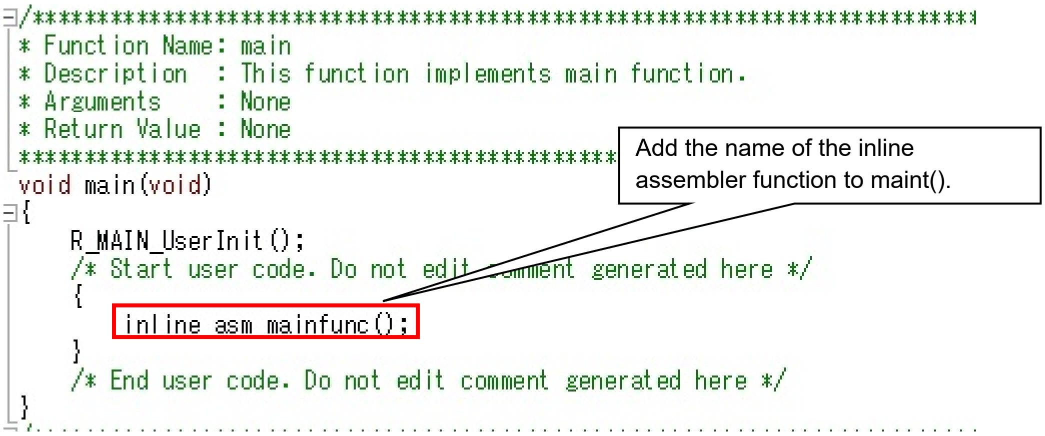
```
/***********************************************************************
* Function Name:r_inttm_func
* Description : This function interrupt function : INTTM00
* Arguments : none
* Return Value : none
***********************************************************************/
void r_inttm_func(void)
{
;*******************************************************************
;
;    interrupt function : INTTM0
;    occur every 2ms/1ms/0.5ms/0.25                ms
;
;*******************************************************************
IINTTM00:
    PUSH    AX
    CALL    !_INTTM00               ; call actual blinking function routine
    POP     AX
    RETI
}
```

Delete “RETI”.

Figure 1.17 Deleting "RETI" (r_cg_main.c)

1.5 Calling Inline Assembler Function from main Function

Add the created inline assembler function (inline_asm_mainfunc()) to the main function (main()).



```
/***********************************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************************/
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    {
        inline_asm_mainfunc();
    }
    /* End user code. Do not edit comment generated here */
}
```

Add the name of the inline assembler function to maint().

Figure 1.18 r_cg_main.c

After completing the above steps, you are ready to migrate the source code from the assembly language to the C language.

1.6 Building a Project

Select "Build Project (B)" from the CS+ Build (B) menu to build a project.

If the following message is displayed on the output window, the project has been successfully built.

"========== Ended(Success:1 Projects, Failed:0 Projects)(Tuesday, xxx xx, 2017 xx:xx:xx AM) =========="

If an error message is displayed, debug the project according to the error message displayed.

2. Sample Code

The sample code is available on the Renesas Electronics website.

3. Reference Documents

User’s Manual:

RL78/G10 Initialization CC-RL (R01AN2668E) Application Note
RL78/G10 Timer Array Unit (Interval Timer) CC-RL (R01AN3074E) Application Note
RL78/G10 User's Manual: Hardware (R01UH0384E)
RL78 Family User's Manual: Software (R01US0015E)
The latest version can be downloaded from the Renesas Electronics website.

Technical Updates/Technical News

The latest information can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact

Revision Record	RL78 Software Migration Guide Migration from Assembly Source Code to C Assembly Source Code CC-RL

Rev.	Date	Description	
		Page	Summary
1.00	Sept. 30, 2017	—	First edition issued
1.01	Jan. 23, 2018	3	Modification Figure 1.1.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   ¾ The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   ¾ The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   ¾ The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   ¾ When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   ¾ The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0