RL78 Software Migration Guide

Migrating from CA78K0R to CC-RL (CS+)

R01AN3100EJ0100
Rev.1.00
Feb. 26, 2016

Introduction

This application note describes how to replace the source codes created by the CA78K0R C compiler for the integrated development environment CS+ with the source codes supported by the CC-RL C compiler for the integrated development environment CS+.

The applicable C compiler versions are as follows.

- CA78K0R V1.20 and later
- CC-RL V1.01.00

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. Methods for Migrating Projects from CA78K0R to CC-RL

Two methods are available to replace the source codes created by the CA78K0R C compiler for the integrated development environment CS+ with the source codes supported by the CC-RL C compiler for the integrated development environment CS+.

In the first method, create a new project with the integrated development environment CS+, then manually port the source codes created by the CA78K0R C compiler for the integrated development environment CS+, and finally create a project supported by the CC-RL C compiler for the integrated development environment CS+. In the second method, use the porting support function in the integrated development environment CS+ to change the source codes created by the CA78K0R C compiler for the integrated development environment CS+ to a new project supported by the CC-RL C compiler for the integrated development environment CS+.

Section 2 explains the manual migration method. Section 3 explains the migration method using the porting support function.

2. Manual Migration Method

2.1 Generating Source Codes Automatically

Source codes are automatically generated using the code generator tool in the CC-RL C compiler for the integrated development environment CS+. Set the code generator tool by referring to the existing source codes that were created by the CA78K0R C compiler for the integrated development environment CS+.

(1) Under [Project Tree], click [Clock Generator] in [Code Generator (Design Tool)]. (Figure 2.1 A).

(2) Perform “Pin assignment” and click the [Fix settings] button. (Figure 2.1 B)

Note: To set other functions, it is necessary to set the pin assignment. When the pin assignment setting is decided once, it is not possible to change it later.

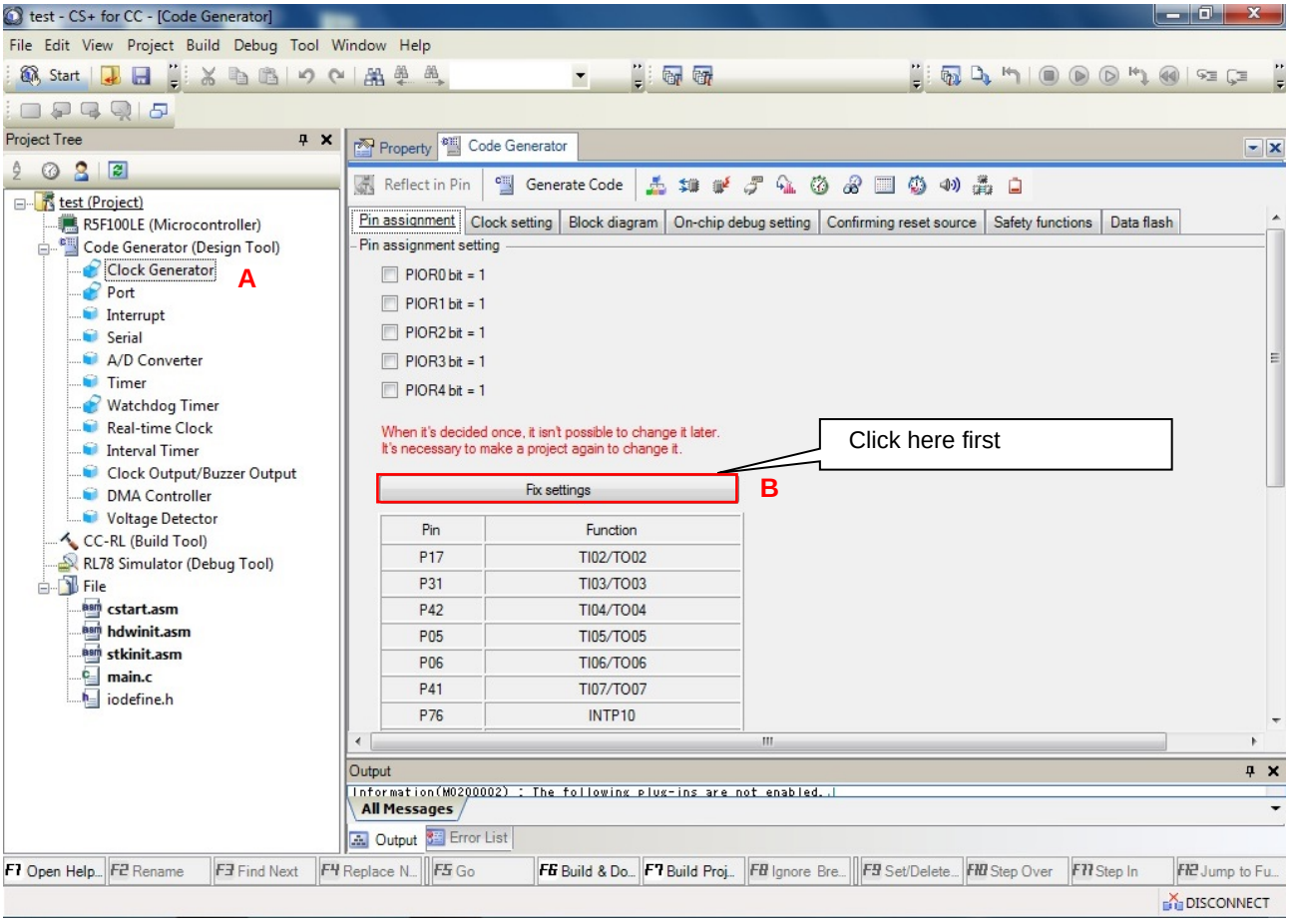


Figure 2.1 Code Generator Setting Window (1)

(3) Refer to the existing source codes that were created by the CA78K0R C compiler for the integrated development environment CS+ and set each function.

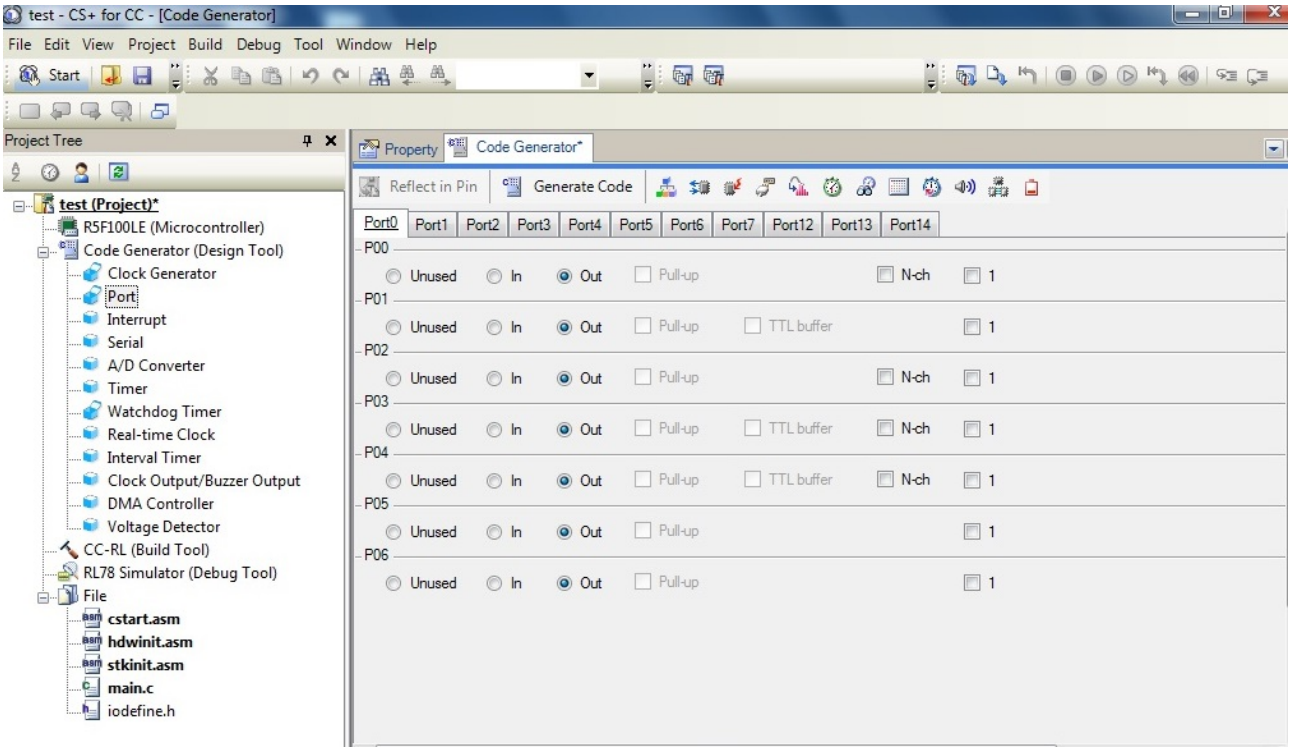


Figure 2.2 Code Generator Setting Window (2)

(4) On completion of all the function settings, click the [Generate Code] button at the top of the window (Figure 2.3 C) to generate codes (automatic source code generation).

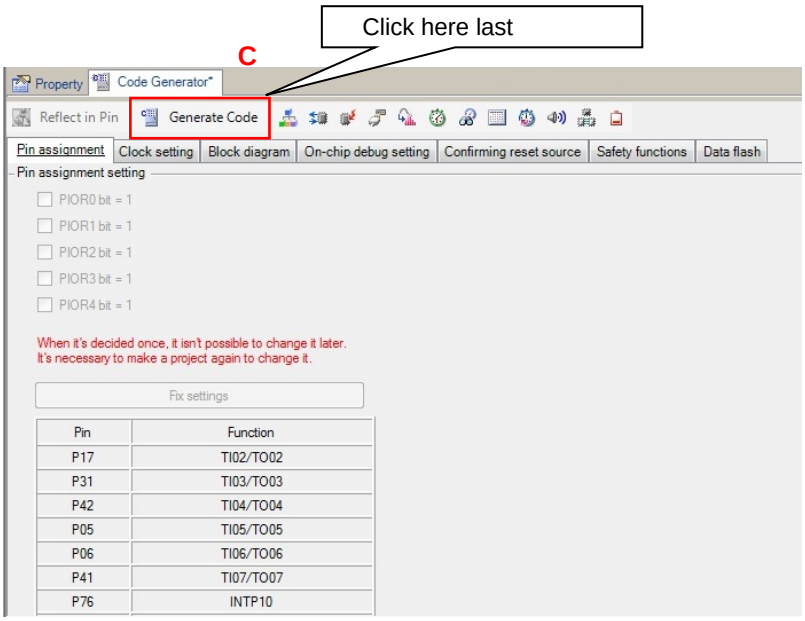


Figure 2.3 Code Generator Setting Window (3)

2.2 Adding Source Codes Other Than Automatically Generated Source Codes

2.2.1 Adding Source Codes

Add the source codes required for the source codes generated automatically with the source generator tool.

First, check the difference between the source code created with CA78K0R and the automatically generated source code. To check the difference, use the software that can compare multiple text files.

Next, add the difference to the automatically generated source code. Include the source code between "/* Start user code for include. Do not edit comment generated here */" and "/* End user code. Do not edit comment generated here */".

If the source codes are added to the position other than the above, automatically generating source code again by pressing the [Generate Code] button in the automatic generator tool will clear the source codes added to the position other than the above. To prevent the codes from being cleared, change the setting as shown below in the code generator tool.

As indicated in the red box in Figure 2.4, change [Generate file] in [Generate File Mode] from [Merge file] to [Do nothing if file exists].

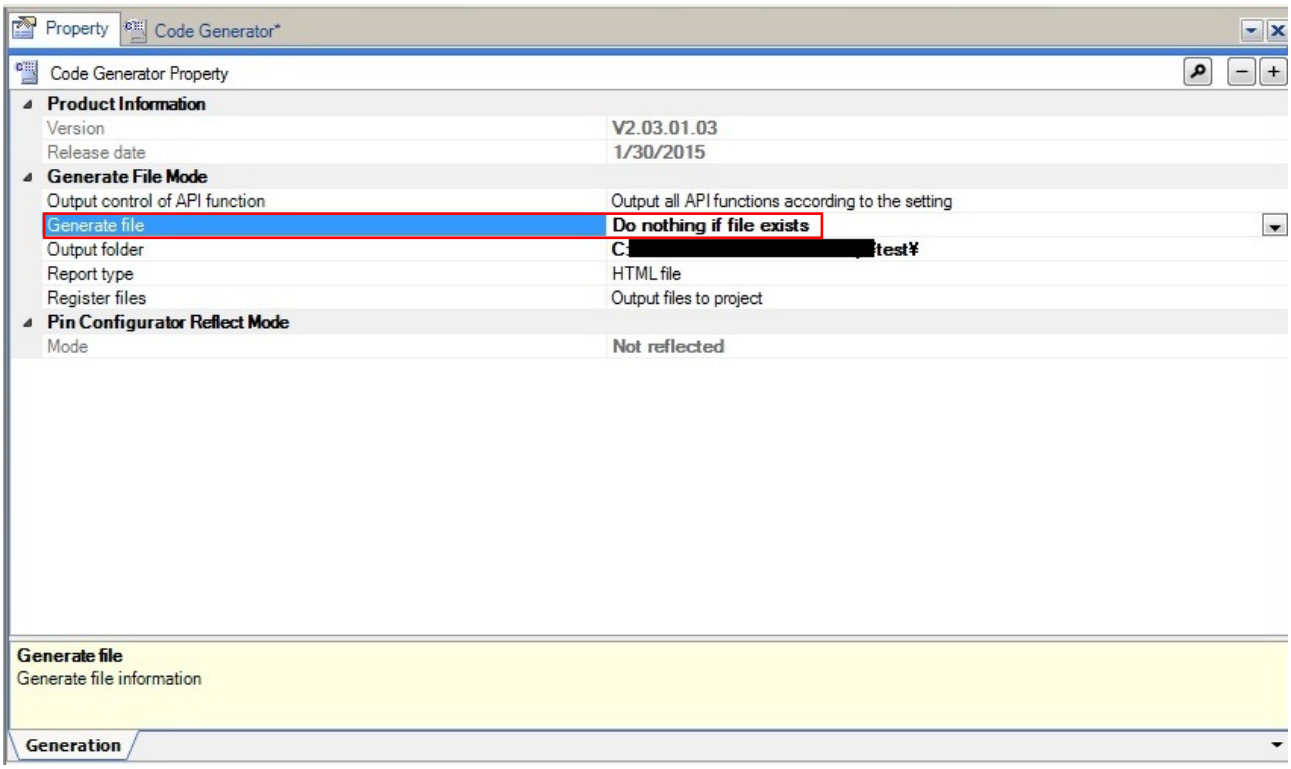


Figure 2.4 Code Generator Property

2.2.2 Adding User Initialization Function

The function for user initialization process (R_×××_Create_UserInit (××× is the function name)) created by the code generator tool in CA78K0R is not automatically generated by the code generator tool in CC-RL. The process of calling the user initialization function within the function initialization function is not also automatically generated.

r_cg_dmac.c for CA78K0R

```
44
45      /***********************************************************************************************
46      *·Function·Name:·R_DMAC0_Create
47      *·Description··:·This·function·initializes·the·DMA0·transfer.
48      *·Arguments···:·none
49      *·Return·Value·:·none
50      ***********************************************************************************************/
51      void·R_DMAC0_Create(void)
52      {
53      ····DRC0·=·_80_DMA_OPERATION_ENABLE;
54      ····NOP();
55      ····NOP();
56      ····DMAMK0·=·1U;····/*·disable·INTDMA0·interrupt·*/
57      ····DMAIF0·=·0U;····/*·clear·INTDMA0·interrupt·flag·*/
58      ····/*·Set·INTDMA0·low·priority·*/
59      ····DMAPR10·=·1U;
60      ····DMAPR00·=·1U;
61      ····DMC0·=·_00_DMA_TRANSFER_DIR_SFR2RAM·|·_20_DMA_DATA_SIZE_16·|·_01_DMA_TRIGGER_AD;
62      ····DSA0·=·_1E_DMA0_SFR_ADDRESS;
63      ····DRA0·=·_FE20_DMA0_RAM_ADDRESS;
64      ····DBC0·=·_0000_DMA0_BYTE_COUNT;
65      ····DEN0·=·0U;····/*·disable·DMA0·operation·*/
66      (1)··R_DMAC0_Create_UserInit();
67      }
68      /***********************************************************************************************
69      End·of·function·R_DMAC0_Create
70      ***********************************************************************************************/
71
```

r_cg_dmac.c for CC-RL

```
49
50      /***********************************************************************************************
51      *·Function·Name:·R_DMAC0_Create
52      *·Description··:·This·function·initializes·the·DMA0·transfer.
53      *·Arguments····:·None
54      *·Return·Value·:·None
55      ***********************************************************************************************/
56      void·R_DMAC0_Create(void)
57      {
58      ····DRC0·=·_80_DMA_OPERATION_ENABLE;
59      ····NOP();
60      ····NOP();
61      ····DMAMK0·=·1U;·/*·disable·INTDMA0·interrupt·*/
62      ····DMAIF0·=·0U;·/*·clear·INTDMA0·interrupt·flag·*/
63      ····/*·Set·INTDMA0·low·priority·*/
64      ····DMAPR10·=·1U;
65      ····DMAPR00·=·1U;
66      ····DMC0·=·_00_DMA_TRANSFER_DIR_SFR2RAM·|·_20_DMA_DATA_SI      ·|·_01_DMA_TRIGGER_AD;
67      ····DSA0·=·_1E_DMA0_SFR_ADDRESS;
68      ····DRA0·=·_FE20_DMA0_RAM_ADDRESS;
69      ····DBC0·=·_0000_DMA0_BYTE_COUNT;
70      ····DEN0·=·0U;·/*·disable·DMA0·operation·*/
71      (2)
72
73      /***********************************************************************************************
74      *·Function·Name:·R_DMAC0_Start
```

The process of calling the user initialization function "R_DMAC0_Cereate_UserInit()" is not automatically generated.

Figure 2.5 Difference between CA78K0R and CC-RL Code Generator Tools

An example of using the function for DMA0 user initialization (R_DMA0_Create_UserInit) is shown to describe how to add the user initialization function.

(1) Copy “R_DMA0_Create_UsetInit” included in r_cg_dmac_user.c that is the source code for CA78K0R to r_cg_dmac_user.c that is the source code for CC-RL.

r_cg_dmac_user.c for CA78K0R

```
/*·End·user·code.·Do·not·edit·comment·generated·here·*/
#include·"r_cg_userdefine.h"

/***********************************************************************************
*·Function·Name:·R_DMAC0_Create_UserInit
*·Description··:·This·function·adds·user·code·after·initializing·DMA0.
*·Arguments··:·none
*·Return·Value··:·none
***********************************************************************************/
void·R_DMAC0_Create_UserInit(void)
{
⟶/*·Start·user·code.·Do·not·edit·comment·generated·here·*/
⟶DEN0⟶=·1U;⟶⟶⟶⟶⟶⟶⟶/*·Enable·DMA0·operation·*/
⟶DRA0⟶=·(uint16_t)&g_AdResult;⟶/*·Set·destination·RAM·address·*/
⟶DBC0⟶=·ADC_USED_CH_NUM·*·ADC_EXEC_TIMES;
⟶DEN0⟶=·0U;⟶⟶⟶⟶⟶⟶⟶/*·Disable·DMA0·operation·*/
⟶/*·End·user·code.·Do·not·edit·comment·generated·here·*/
}
/***********************************************************************************
End·of·function·R_DMAC0_Create_UserInit
***********************************************************************************/
```

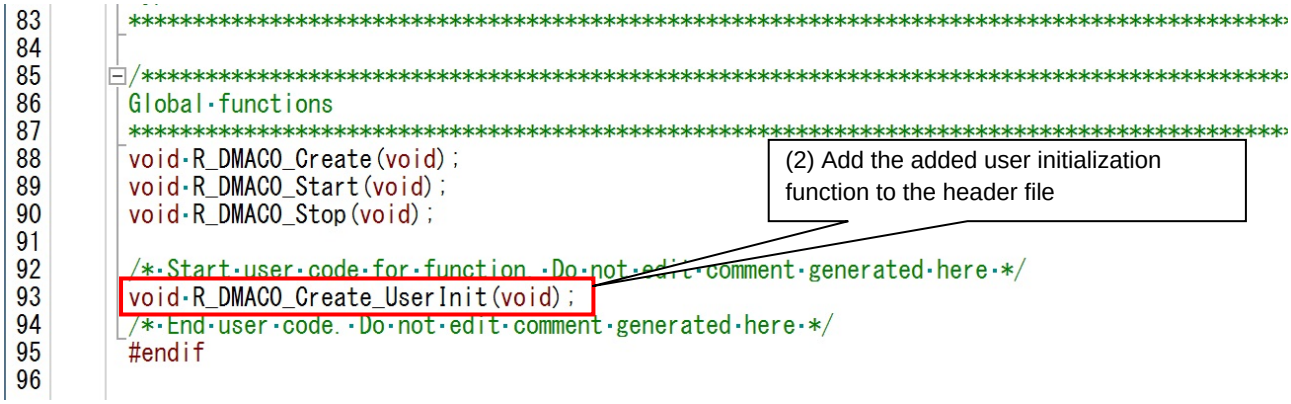
r_cg_dmac_user.c for CC-RL

Paste the entire “R_DMAC0_Create_UserInit”.

```
/*·Start·user·code·for·adding.·Do·not·edit·comment·generated·here·*/
/***********************************************************************************
*·Function·Name:·R_DMAC0_Create_UserInit
*·Description··:·This·function·adds·user·code·after·initializing·DMA0.
*·Arguments··:·none
*·Return·Value··:·none
***********************************************************************************/
void·R_DMAC0_Create_UserInit(void)
{
····DEN0···=·1U;·····················/*·Enable·DMA0·operation·*/
····DRA0···=·(uint16_t)&g_AdResult;····/*·Set·destination·RAM·address·*/
····DBC0···=·ADC_USED_CH_NUM·*·ADC_EXEC_TIMES;
····DEN0···=·0U;·····················/*·Disable·DMA0·operation·*/
}
/***********************************************************************************
End·of·function·R_DMAC0_Create_UserInit
***********************************************************************************/
/*·End·user·code.·Do·not·edit·comment·generated·here·*/
```

Figure 2.6 Adding User Initialization Function

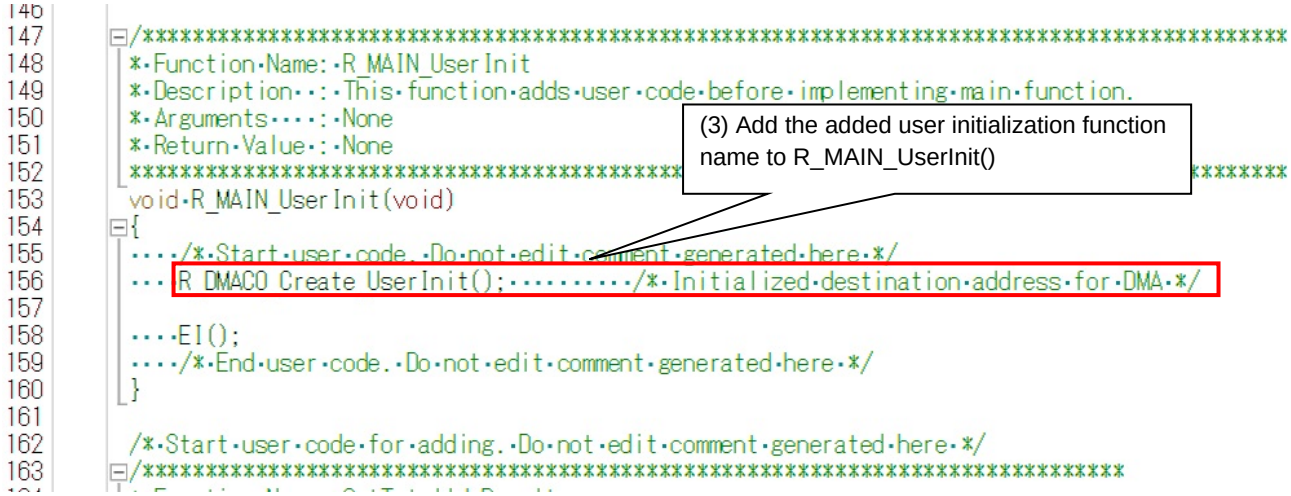
(2) Globally declare the added function.




```
83      ************************************************************************************************
84
85      /************************************************************************************************
86      Global functions
87      ************************************************************************************************
88      void R_DMAC0_Create(void);
89      void R_DMAC0_Start(void);
90      void R_DMAC0_Stop(void);
91
92      /* Start user code for function. Do not edit comment generated here */
93      void R_DMAC0_Create_UserInit(void);
94      /* End user code. Do not edit comment generated here */
95      #endif
96
```

Figure 2.7 Description Example in CC-RL Header File “r_cg_dmac.h”

(3) Add the process to be added to call the user initialization function in the R_MAIN_UserInit() function in the r_main.c file.




```
147     /************************************************************************************************
148     * Function Name: R_MAIN_UserInit
149     * Description  : This function adds user code before implementing main function.
150     * Arguments    : None
151     * Return Value : None
152     ************************************************************************************************
153     void R_MAIN_UserInit(void)
154     {
155         /* Start user code. Do not edit comment generated here */
156         R_DMAC0_Create_UserInit();          /* Initialized destination address for DMA */
157
158         EI();
159         /* End user code. Do not edit comment generated here */
160     }
161
162     /* Start user code for adding. Do not edit comment generated here */
163     /************************************************************************************************
```

Figure 2.8 Adding Process of Calling User Initialization Function for CC-RL

This completes the process of adding the user initialization function.

2.3 Correcting Added Parts

A warning message or an error may occur if the source code added in section 2 is left unchanged. In this case, the description should be corrected according to the CC-RL specifications.

The main differences in the description specifications between CA78K0R and CC-RL are described below.

2.3.1 Accessing Special Function Registers (SFR)

(1) Change the method of accessing the special function registers (SFR).

CA78K0R: #pragma sfr

CC-RL: #include ”iodefine.h”

Because CC-RL does not support #pragma sfr, include the define header file for the sfr access “iodefine.h” that is automatically generated by the code generator tool.

(2) Correct the port register description.

When CA78K0R is used, ".bit number" is added at the end of a register name. When CC-RL is used, "_bit.no bit number" is added at the end of a register name.

r_main.c for CA78K0R

```
88
89          /* AD conversion stop */
90          R_ADC_Stop();
91
92          /* Check result of AD conversion data */
93          if (result == 0x00)
94          {
95              if (testVoltageIndex == 2)   /* AD test all OK */
96              {
97                  /* LED1 turn on */
98                  P6.2 = 0;
99                  while (1U)
100                 {
101                     /* Do Nothing */
102                 }
103             }
104             else                    /* Next AD test */
105             {
106                 ++testVoltageIndex;
107             }
108         }
109         else    /* AD test NG */
110         {
111             /* LED blinks */
112             R_Main_Blink_Led();
113         }
114     }
115     /* End user code. Do not edit comment generated here */
116 }
```

r_main.c for CC-RL

```
83
84          /* Gets the check result of AD conversion data */
85          result = R_Main_Check_AD_Data(testVoltageIndex);
86
87          /* AD conversion stop */
88          R_ADC_Stop();
89
90          /* Check result of AD conversion data */
91          if (result == 0x00U)
92          {
93              if (testVoltageIndex == 2U) /* AD test all OK */
94              {
95                  /* LED1 turn on */
96                  P6_bit.no2 = 0U;
97                  while (1U)
98                  {
99                      /* Do Nothing */
100                 }
101             }
102             else                    /* Next AD test */
```

Figure 2.9 Port Register Descriptions

2.3.2 Enabling Interrupt Functions

Replace the #pragma directive with a function.

1. In case of di

   #pragma di → __DI();

   (When r_cg_macrodriver.h is used, DI(); can also be used.)

2. In case of ei

   #pragma ei → __EI();

   (When r_cg_macrodriver.h is used, EI(); can also be used.)

2.3.3 Enabling CPU Control Instructions

Replace the #pragma directive with a function.

1. In case of halt

   #pragma halt → __halt();

   (When r_cg_macrodriver.h is used, HALT(); can also be used.)

2. In case of stop

   #pragma stop → __stop();

   (When r_cg_macrodriver.h is used, STOP(); can also be used.)

3. In case of brk

   #pragma brk → __brk();

   (When r_cg_macrodriver.h is used, BRK(); can also be used.)

4. In case of nop

   #pragma nop → __nop();

   (When r_cg_macrodriver.h is used, NOP(); can also be used.)

2.3.4 Replacing Absolute Address Specification (__directmap)

To specify absolute addresses with CA78K0R, “__directmap” is used. To specify absolute addresses with CC-RL, “#pragma address” is used.

Change (1) “__directmap type specification variable name = start address;” to (2) “#pragma address variable name = start address” and (3) “type specification variable name;”.

Example)

(1) __directmap uint8_t p130_high = {0xFE900};

(2) #pragma address p130_high = 0xFE900U

(3) uint8_t __near p130_high;

r_main.c for CA78K0R

```
/************************************************************************************************
Global variables and functions
************************************************************************************************/
/* Start user code for global. Do not edit comment generated here */
(1) __directmap uint8_t p130_high            = {0xFE900};
    __directmap uint8_t p130_low             = {0xFE901};
    __directmap uint8_t adc_snooze           = {0xFE902};
    __directmap uint16_t get_adcr[MAX_BUFFER] = {0xFEA00};

uint8_t  buffer_count;                /* buffer counter */
uint16_t result_buffer[MAX_BUFFER];   /* AD converter result buffer */
/* End user code. Do not edit comment generated here */
```

r_main.c for CC-RL

```
/************************************************************************************************
Pragma directive
************************************************************************************************/
/* Start user code for pragma. Do not edit comment generated here */
(2) #pragma address p130_high  = 0xFE900U
    #pragma address p130_low   = 0xFE901U
    #pragma address adc_snooze = 0xFE902U
    #pragma address get_adcr   = 0xFEA00U
/* End user code. Do not edit comment generated here */

/************************************************************************************************
Global variables and functions
************************************************************************************************/
/* Start user code for global. Do not edit comment generated here */
(3) uint8_t __near p130_high;
    uint8_t __near p130_low;
    uint8_t __near adc_snooze;
    uint16_t __near get_adcr[MAX_BUFFER];

uint8_t  buffer_count;                /* buffer counter */
uint16_t result_buffer[MAX_BUFFER];   /* AD converter result buffer */
/* End user code. Do not edit comment generated here */
```

Figure 2.10 Description of __directmap

2.3.5 Replacing Variable saddr Area Allocation (sreg, __sreg)

To allocate a variable to the saddr area with CA78K0R, “sreg (or __sreg)” is used. To allocate a variable to the saddr area with CC-RL, “__saddr” is used.

Change (1) “sreg type specification variable name; (or __sreg type specification variable name;)” to (2) “__saddr type specification variable name;”.

Example)

(1) sreg uint32_t g_PulseWidth[8];

(2) __saddr uinit32_t g_pulse_width[8];

r_main.c for CA78K0R

```
52      /************************************************************************
53      Exported global variables and functions (to be accessed by other files)
54      ************************************************************************/
55  (1) sreg uint32_t g_PulseWidth[8];      /* Store pulse width */
56      uint8_t g_times;                    /* Measurement times counter */
57      /* End user code. Do not edit comment generated here */
58
59      /************************************************************************
```

r_main.c for CC-RL

```
46      /************************************************************************
47      Global variables and functions
48      ************************************************************************/
49      /* Start user code for global. Do not edit comment generated here */
50  (2) __saddr uint32_t g_pulse_width[8];      /* Store pulse width */
51      uint8_t g_times;                        /* Measurement times counter */
52      /* End user code. Do not edit comment generated here */
53      void R_MAIN_UserInit(void);
54
55      /************************************************************************
```

Figure 2.11 Example of Changing sreg Description

2.3.6 near/far Attributes

As for the memory model, the small, medium, and large models are available in CA78K0R, but only the small and medium models are available in CC-RL.

When the small model or the medium model in CA78K0R is used, the same memory model of the small model or the medium model in CC-RL is used.

When the large model in CA78K0R is used, the medium model in CC-RL is used.

In addition, if neither the near area nor the far area is specified for a function or a variable, the function or variable is allocated in the near area. Therefore, the __far type qualifier is used to allocate a function or variable in the far area.

In case of the source code that references the area extending over 64 K such as the source code that references the CRC calculation result data, the pointer attribute should be changed to the far attribute.

Example)

(1) Change “uint16_t *oc_calc_hs_crc;” to “__far uint16_t *oc_calc_hs_crc;”.

(2) Change “oc_calc_hs_crc = (uint16_t *)HIGHSPEED_CALC_ADDR;” to “oc_calc_hs_crc = (__far uint16_t *)HIGHSPEED_CALC_ADDR;”.

r_main.c for CA78K0R

```
79      (1) uint16_t  *oc_calc_hs_crc;  /* Pointer of OC result (High-Speed) */
80      uint16_t  count;
81
82      /* High-speed CRC */
83      /* Get High-speed CRC calculated result that OC output. */
84      (2) oc_calc_hs_crc = (uint16_t*)HIGHSPEED_CALC_ADDR;
85
86      result_hs_crc = R_HighSpeedCRCProc();  /* Process of high-speed CRC. */
87
88      /* The results are compared and it outputs it to LED. */
89      if (result_hs_crc == *oc_calc_hs_crc)  /* High-speed CRC. */
90      {
91          P6.2 = 0;                          /* OK = LED Lighting. */
92      }
```

r_main.c for CC-RL

```
80      ........uint16_t........result_gp_crc = 0U;/* Program result (General-Purpose) */
81      ....(1) __far uint16_t  *oc_calc_hs_crc;    /* Pointer of OC result (High-Speed) */
82      ........uint16_t........count = 0U;
83
84      ......../* High-speed CRC */
85      ......../* Get High-speed CRC calculated result that OC output. */
86      ....(2) oc_calc_hs_crc = (__far uint16_t *)HIGHSPEED_CALC_ADDR;
87      ........result_hs_crc = R_HighSpeedCRCProc();   /* Process of high-speed CRC. */
88
89      ......../* The results are compared and it outputs it to LED. */
90      ........if (result_hs_crc == *oc_calc_hs_crc)    /* High-speed CRC. */
91      ........{
92      ............P6_bit.no2 = 0U;                    /* OK = LED Lighting. */
93      ........}
```

Figure 2.12 Example of Changing Pointer Attribute

3. Migration Method Using Porting Support Function

This section explains how to migrate the existing project of the CA78K0R C compiler for the integrated development environment CS+ to the source codes of a new project of the CC-RL C compiler for the integrated development environment CS+.

3.1 Creating Project by Using Existing Project

(1) Click the [Start] button at the top of the window to display the start menu.

(2) Click the [GO] button in the [Create New Project] item in the start menu.

(3) Select the microcontroller to be used.

(4) Select [Application (CC-RL)] for [Kind of project].

(5) Select the [Pass the file composition of an existing project to the new project] checkbox and enter the project file name to be passed in [Project to be passed:]. Then, select the [Copy composition files in the diverted project folder to a new project folder] checkbox.

(6) Click the [Create] button to create a project.

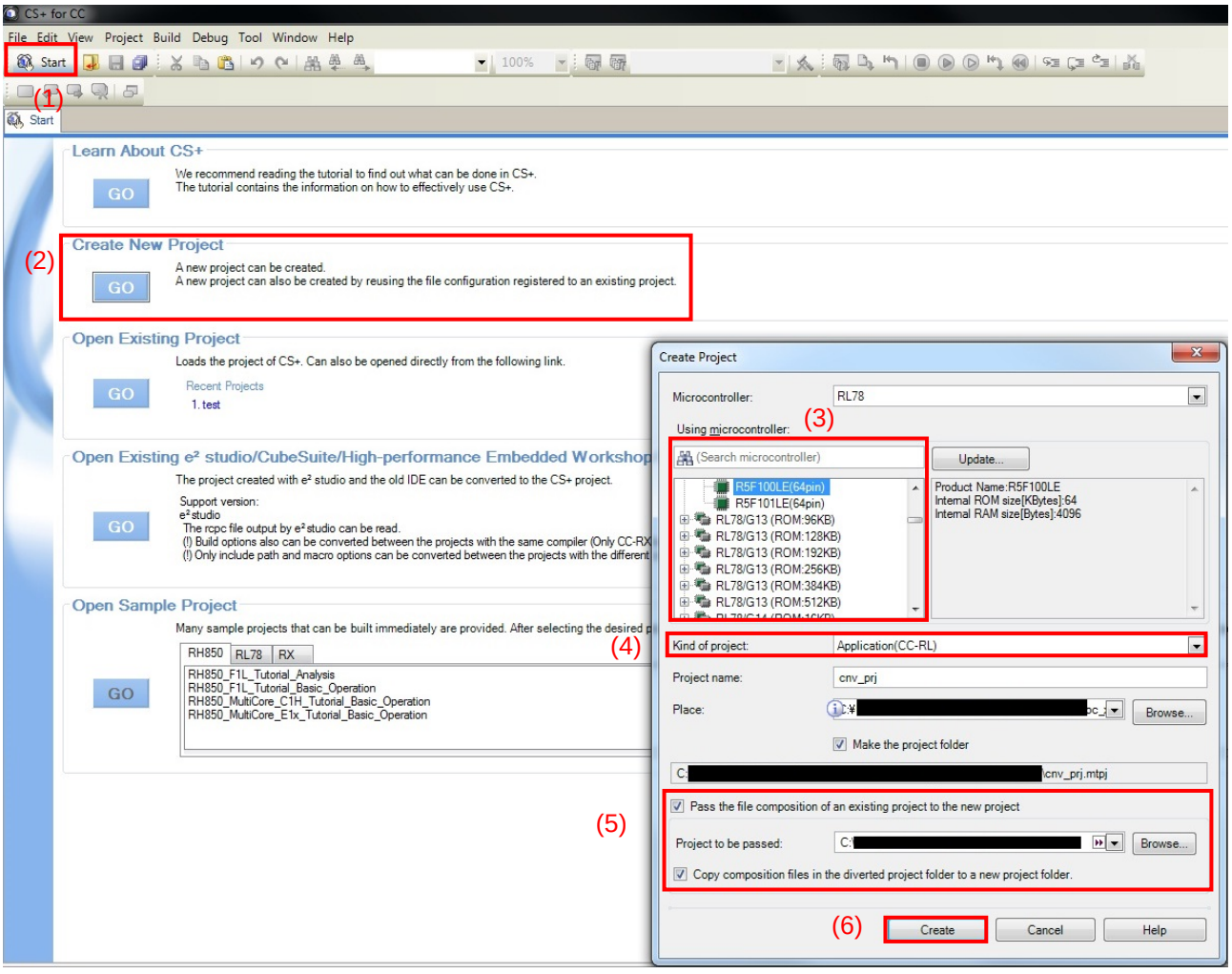


Figure 3.1 Start Menu Window of CS+

3.2 Adding Include File

Click [CC-RL (Build Tool)] in [Project Tree] and open the [Include files at head of compiling units] item in the [Compile Options] tab. Then add “iodefine.h”.

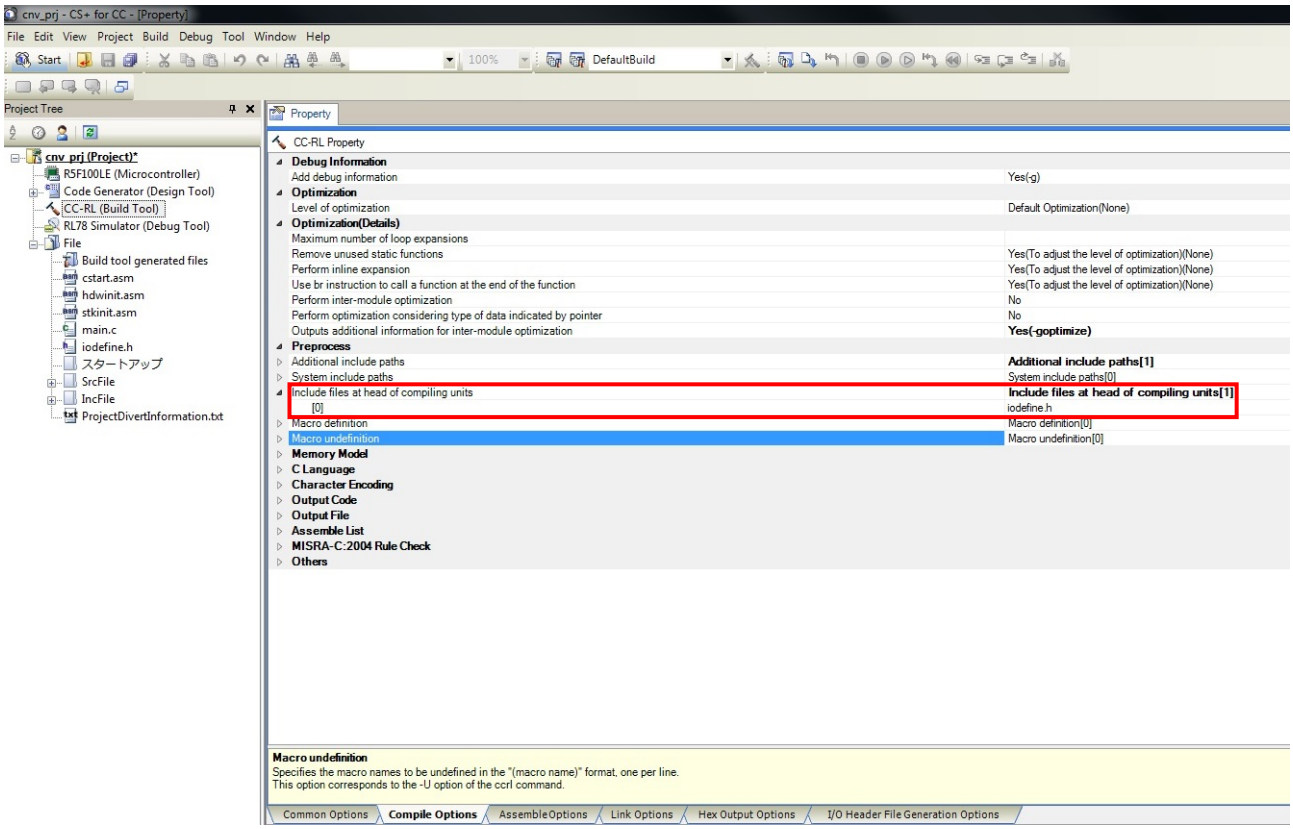


Figure 3.2 Compile Options

3.3 Changing Startup File

When the main function and the hdwinit function are registered in the existing project, use the following procedure to exclude the files that are automatically generated during project creation (main.c and hdwinit.asm) from the target of build.

(1) Right-click [main.c] in [Project Tree] to display the menu.

(2) Select [Property] from the menu.

(3) Change the [Set as build-target] item from [Yes] to [No] in the property of the file. (The same procedure is used to change the setting of hdwinit.asm.)

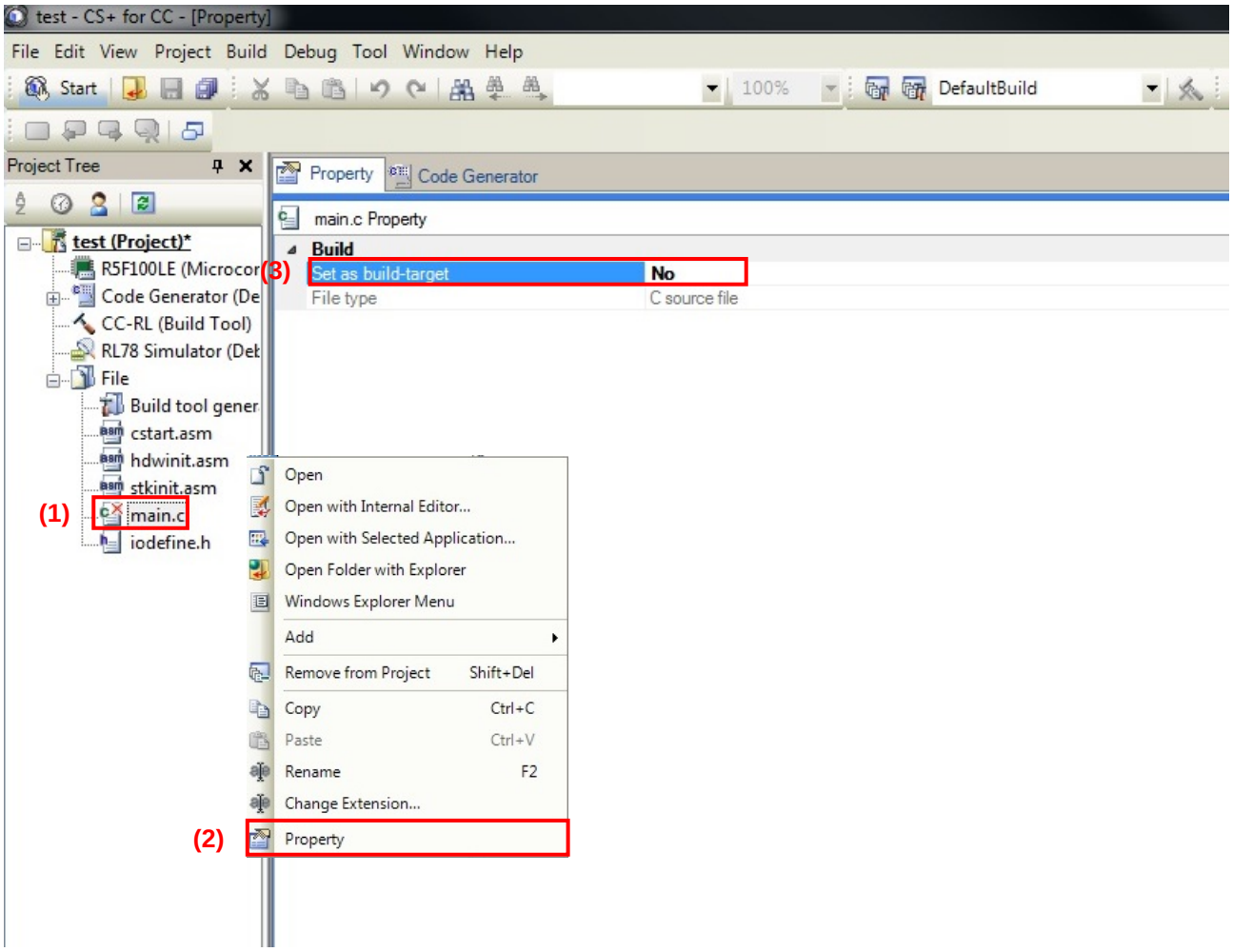


Figure 3.3 Displaying File Property

3.4 Deleting Special Function Registers (SFR) Access Description

Delete the description of “#pragma sfr” in “r_cg_macrodriver.h”.

```
36  /***********************************************************************************
37  Includes
38  ***********************************************************************************
39  #pragma sfr
40  #pragma DI
41  #pragma EI
42  #pragma NOP
43  #pragma HALT
44  #pragma STOP
45
46  /***********************************************************************************
```

Figure 3.1 r_cg_macrodriver.h

Because the porting support function does not support the replacement of the absolute address specification (__directmap), the absolute address specification should be replaced manually according to 2.3.4.

4. ROM Allocation Method

To allocate sections with CA78K0R, a link directive file is used. To allocate sections with CC-RL, the link option section is used for setting. In addition, the -start option can also be used to allocate sections.

Click [CC-RL (Build Tool)] in [Project Tree].

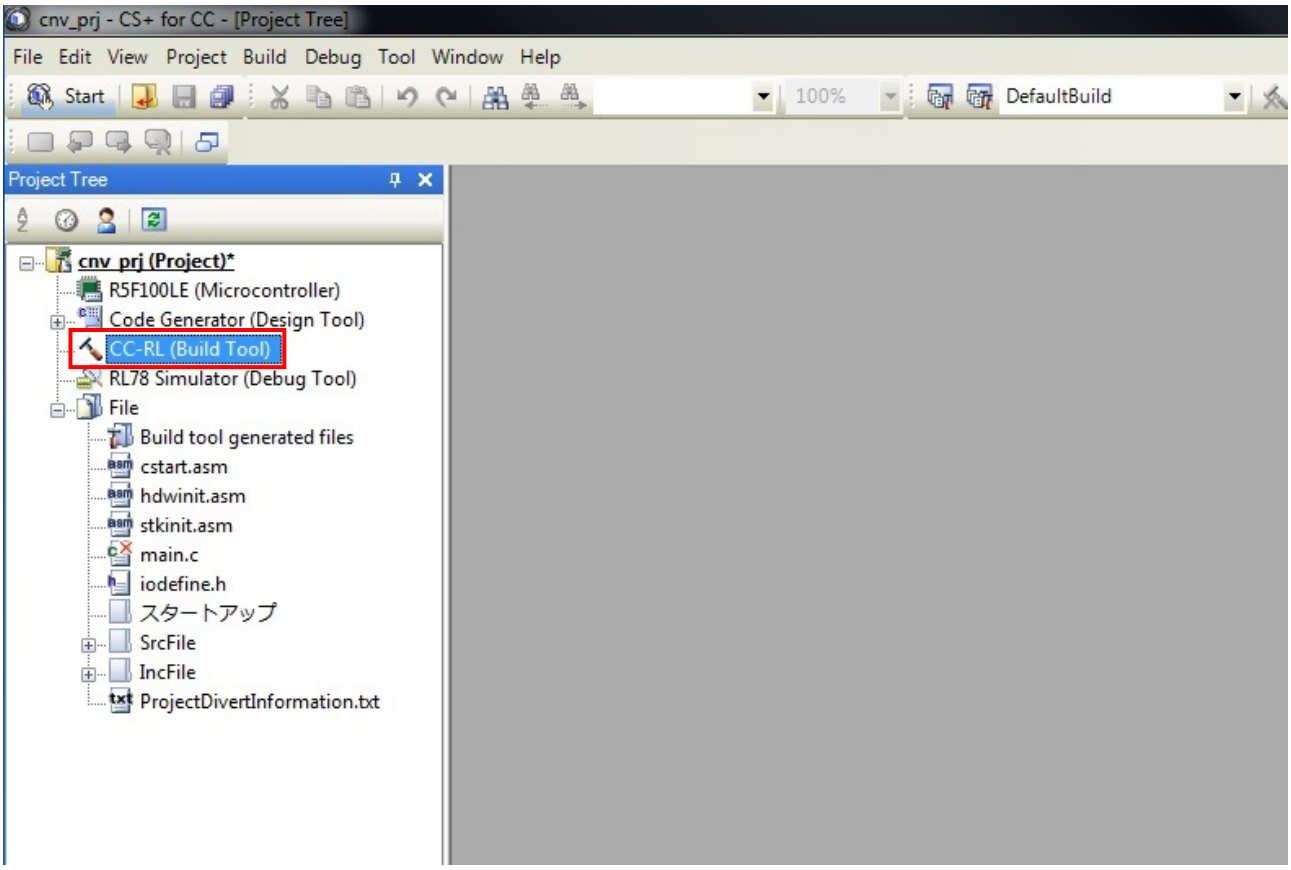


Figure 4.1 Link Option Setting Example in CC-RL (CS+) (1)

(1) Click the [Link Option] tab.

(2) Open the [Section] item.

(3) Change [Layout sections automatically] from [Yes] to [No].

(4) Click the […] button in the [Section start address] item.

(5) Specify the section settings in the section settings window that appears.

(6) Click the [OK] button to finish the section settings.

Important

- The allocation of the ROM area/RAM area in each section cannot be changed.
- The allocation areas for the SFR area, the interrupt vector area (section .vect), and the CALLT function table area (section .callt0) have already been determined and thus are not specified.
- The sections for the saddr area (sections .sdataR and .sbss) should be allocated within the range of the .saddr area.

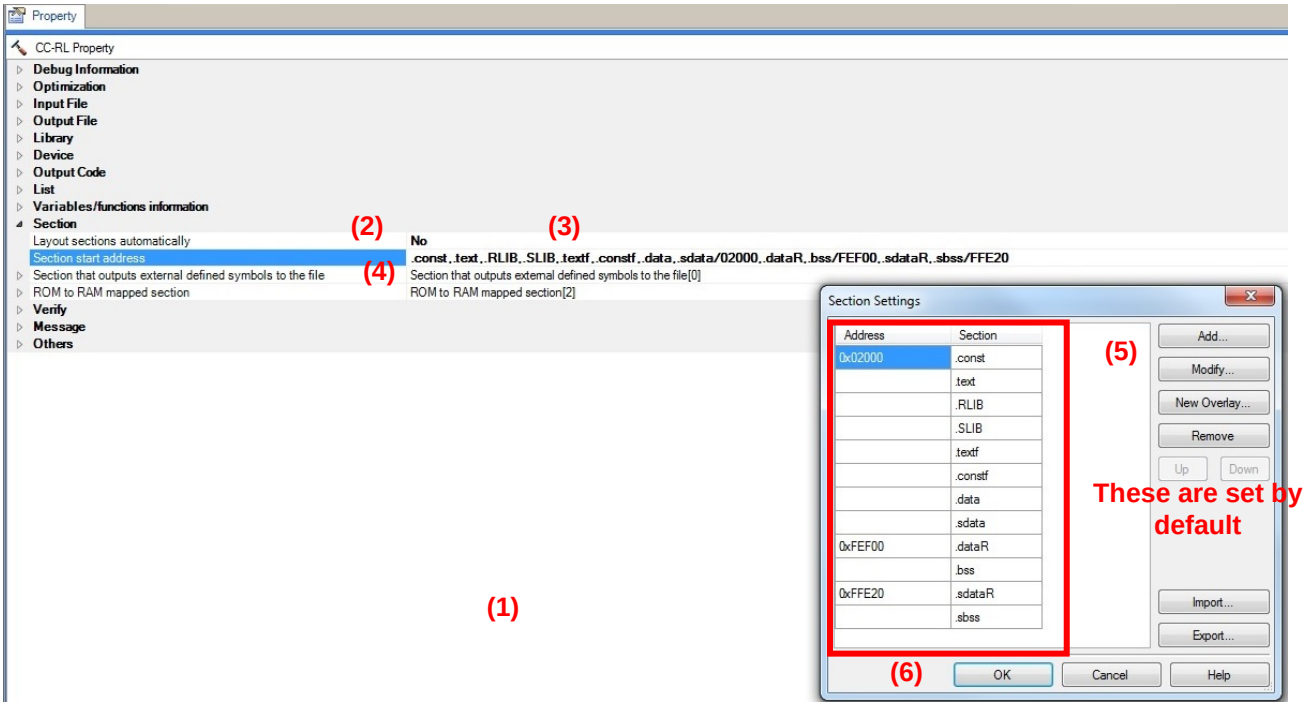


Figure 4.2 Link Option Setting Example in CC-RL (CS+) (2)

5. Sample Code

The sample code is available on the Renesas Electronics website.

6. Reference Documents

RL78Family User's Manual: Software (R01US0015E)

RL78 Compiler CC-RL User's Manual (R20UT3123E)

Integrated Development Environment for the RL78 Family - Migrating from the CA78K0R to the CC-RL

(Project Manipulation) (R20UT3415E)

(Coding) (R20UT3416E)

(Linkage Editor Options) (R20UT3417E)

(Compiler Options and Assembler Options) (R20UT3418E)

CS+ Code Generator Tool Integrated Development Environment User's Manual: RL78 API Reference

(R20UT3102E)

CS+ V3.01.00 Integrated Development Environment User's Manual: Message (R20UT3286E)

CS+ V3.01.00 Integrated Development Environment User's Manual: Project Operation (R20UT3287E)

(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	Migrating from CA78K0R to CC-RL (CS+)

Rev.	Date	Description	
		Page	Summary
1.00	Feb. 26, 2016	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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.

   — 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.

   — 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.

   — 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.

   — 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.

   — The characteristics of an MPU or MCU 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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: +44-1628-585-100, Fax: +44-1628-585-900

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, P. R. China 200333
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 II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0