Replacement Guide from 78K0 Family to RL78 Family (CcnvCA78K0)

R01AN3471EJ0100
Rev.1.00
Nov. 08, 2016

Introduction

This application note describes how to replace the programs for 78K0 with the programs for RL78.

Target Device

78K0 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. How to Replace Programs from 78K0 Family to RL78 Family

This section explains how to replace the programs for 78K0 with the programs for RL78.

First, use the C source converter CcnvCA78K0 to convert the extended functions for the C compiler CA78K0 (or CC78K0) to the extended functions for the C compiler CC-RL.

Next, use the integrated development environment CS+ or e2studio to create a project. Because the 78K0 family and the RL78 family have different peripheral functions, use the code generator for the RL78 family to generate programs for peripheral functions of the RL78 family instead of using the programs for peripheral functions of the 78K0 family.

Combine the programs converted with CcnvCA78K0 and the above programs for peripheral functions to replace programs.

2. Program Conversion Using CcnvCA78K0

2.1 About CcnvCA78K0

CcnvCA78K0 converts extended language specifications (such as macro names, reserved words, #pragma directives, and extended functions) in C source programs for CA78K0 (or CC78K0) into extended language specifications for CC-RL.

CcnvCA78K0 is the software that supports the porting of the programs for CA78K0 to the programs for CC-RL. Since we do not guarantee the correct operation of the programs converted by CcnvCA78K0, be sure to check the operation of the program after conversion.

In addition, the device-dependent codes such as location addresses, access to an SFR, and assembly-language codes cannot be converted. Convert these codes manually into the code for the RL78 family as required.

For details, see “CcnvCA78K0 C Source Converter User's Manual (R20UT3684EJ0100)”.

2.2 How to Use CcnvCA78K0

The method of converting a program with CcnvCA78K0 is shown below.

(1) Place CcnvCA78K0 (CcnvCA78K0.exe) and a program for CA78K0 in the same folder of your choice.

(2) Launch Command Prompt in Windows.

(3) Change the current directory to the folder where CcnvCA78K0 is stored.

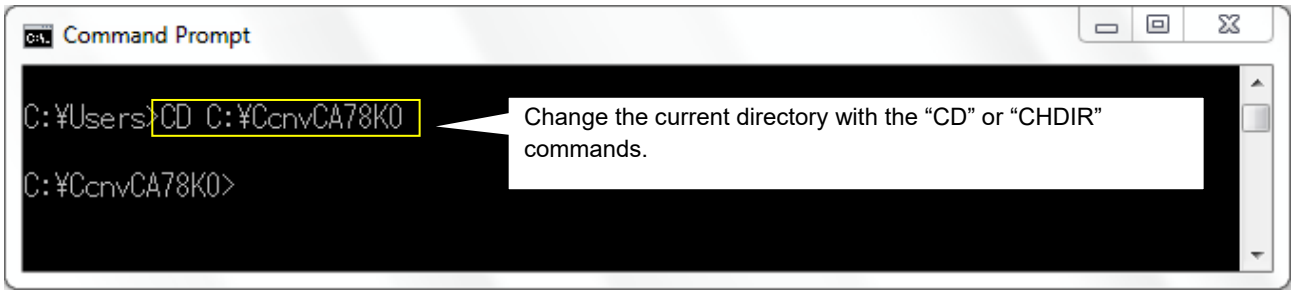


Figure 1.1 Command Prompt Window

(4) Specify an output file name with the -o option before execution. After the execution, a program for CC-RL is output. In addition, when outputting messages in a specified file, use the -r option.

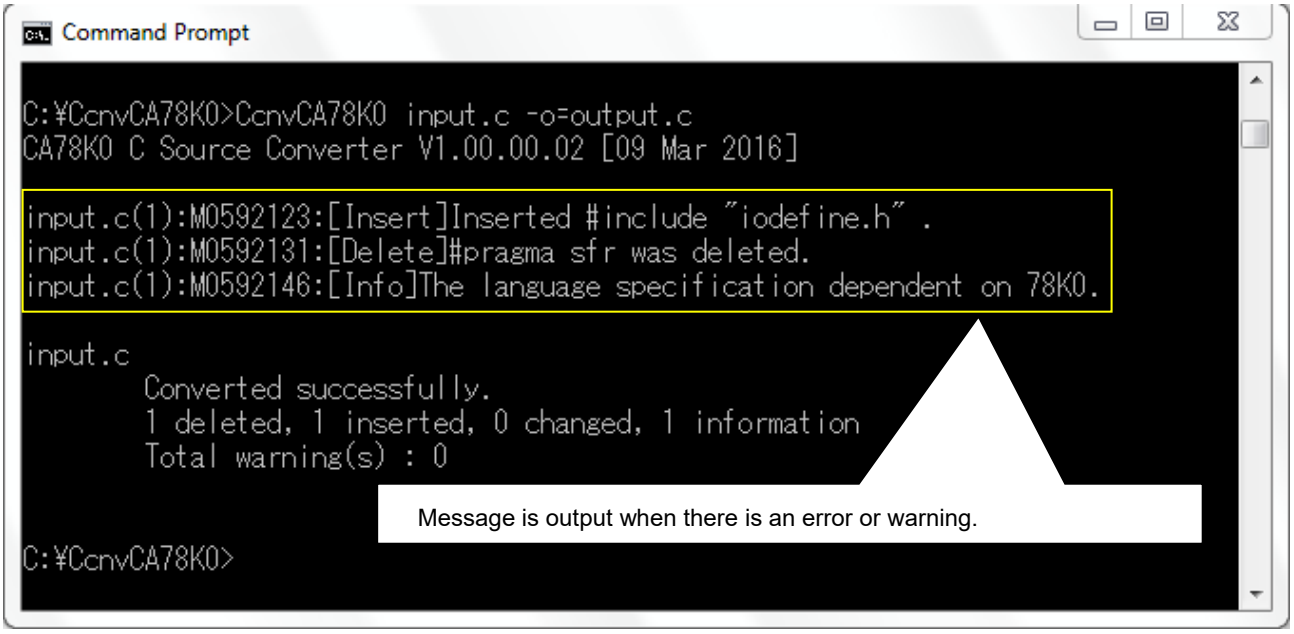


Figure 1.2 CcnvCA78K0 Execution Window

(5) When converting multiple files at the same time, create a list file and execute conversion with the -l option specified. After the execution, programs for CC-RL are output to the specified folder.

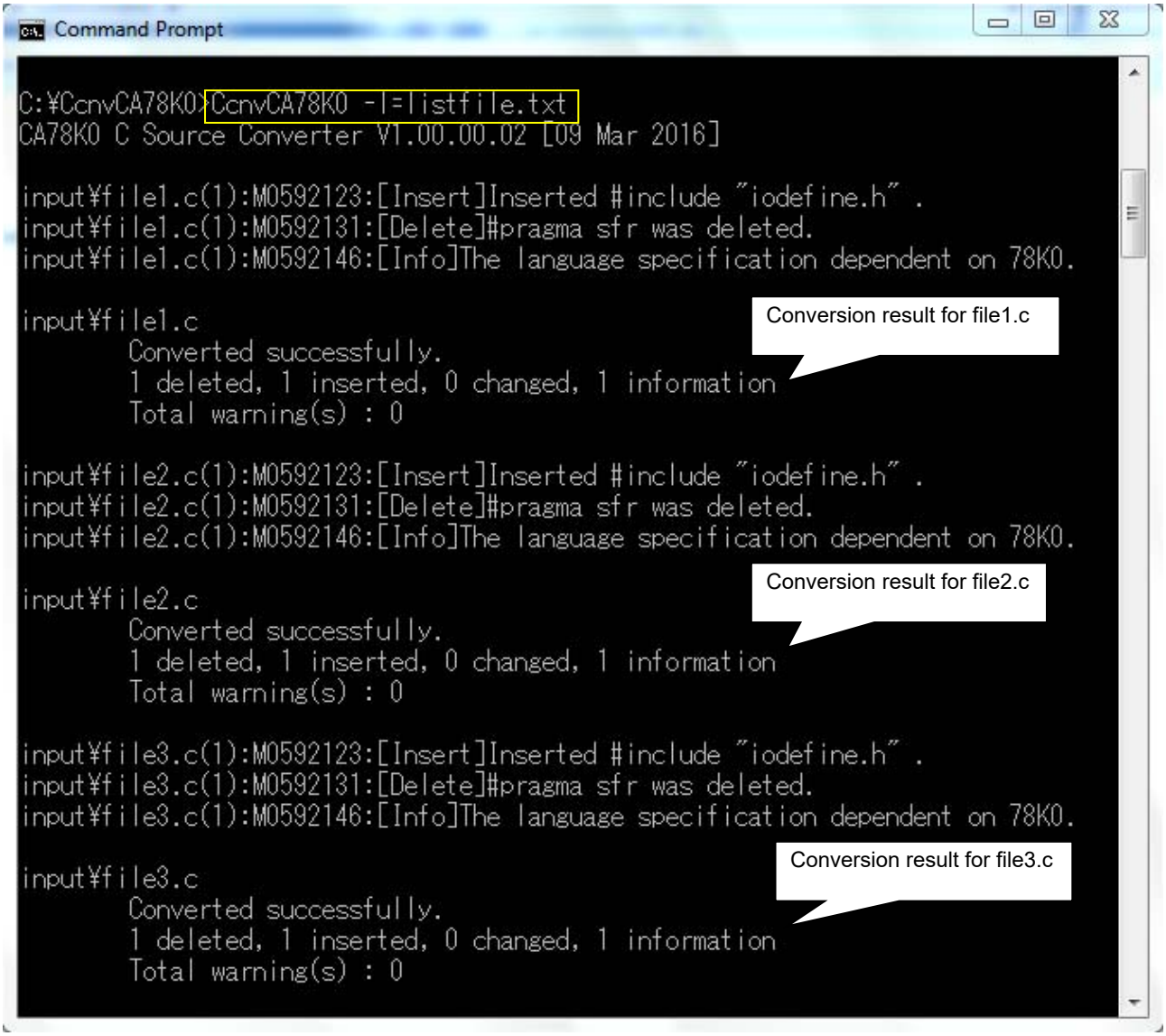


Figure 1.3 CcnvCA78K0 Execution Window (Multiple Files)

The example below shows the description in a list file.

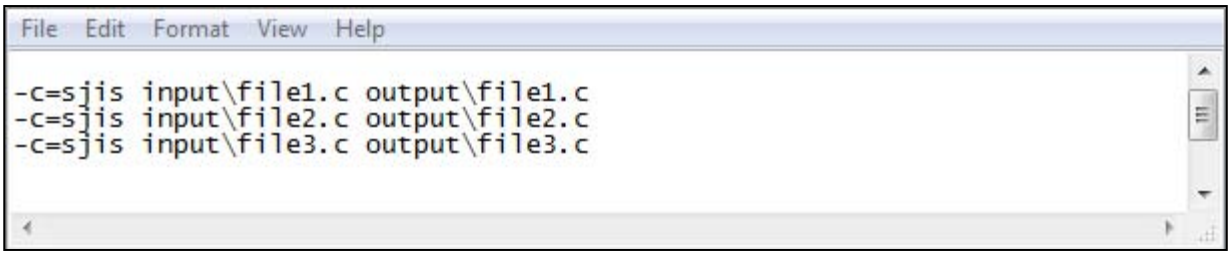


Figure 1.4 Example of Description in List File

(6) Correct the parts that are not converted by CcnvCA78K0. For the parts that require corrections, refer to "CONVERSION SPECIFICATIONS" in "CcnvCA78K0 C Source Converter User's Manual (R20UT3684EJ0100)".

2.3 When CcnvCA78K0 is Not Used

When CcnvCA78K0 is not used, extended functions of CA78K0 (or CC78K0) need to be converted manually into extended functions of CC-RL. For the extended language specifications supported by CC-RL, see "CC-RL Compiler User's Manual (R20UT3123EJ0103)".

3. Converting Programs for Peripheral Functions

3.1 Generating Programs Automatically

Programs are automatically generated for the RL78 family peripheral functions equivalent to the peripheral functions that were used by the 78K0 family by using the code generator for the RL78 family provided in the integrated development environment CS+ or e2studio. For how to use the code generator, see “CS+ Code Generator Integrated Development Environment User’s Manual: Peripheral Function Operation (R20UT3104EJ0100)”.

(1) Under [Project Tree], click [Clock Generator] in [Code Generator (Design Tool)] and perform “pin assignment”. When the pin assignment setting is decided once, it is not possible to change it later.

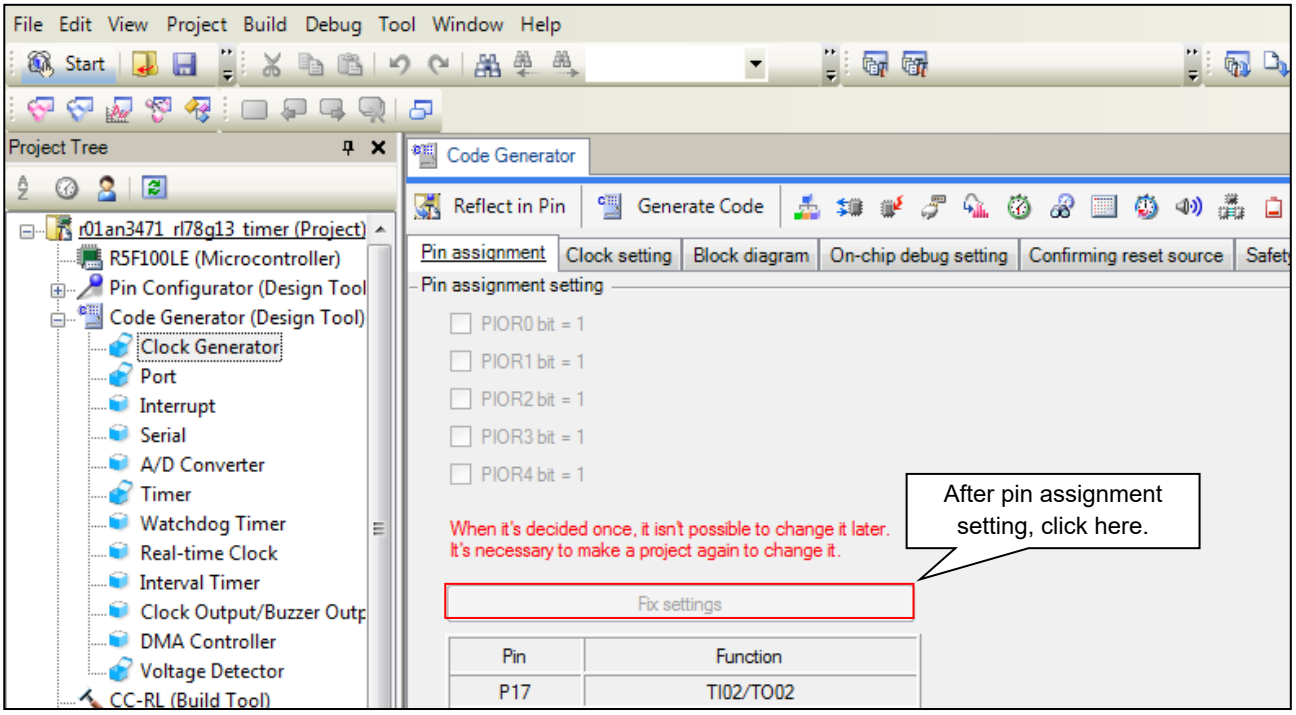


Figure 3.1 Code Generator Setting Window (1)

(2) Refer to the program for the 78K0 family and set each function.

(3) On completion of all the peripheral function settings, click the [Generate Code] button at the top of the window to generate codes (automatic program generation). Use the automatically generated functions for peripheral functions to replace programs.

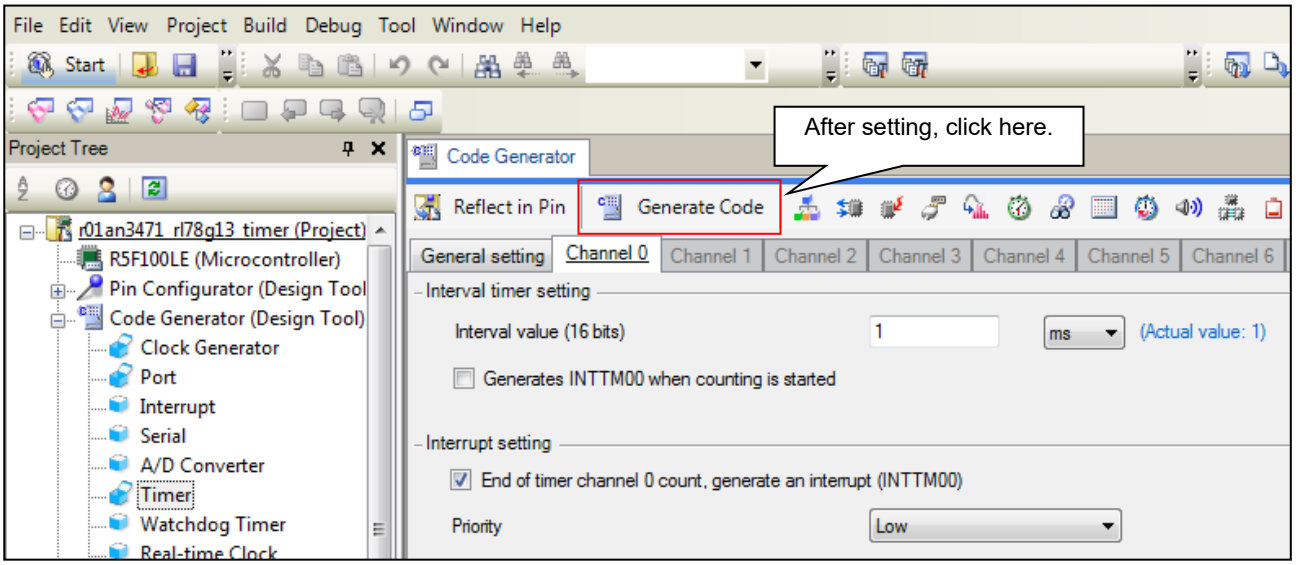


Figure 3.2 Code Generator Setting Window (2)

3.2 Adding Programs

Add the programs that cannot be automatically generated by the code generator (such as main function, interrupt function process, and variables).

Add a program between"/* Start user code for adding. Do not edit comment generated here */" and "/* End user code. Do not edit comment generated here */" in each file that was automatically generated. A program needs to be added manually. Note that any program added outside this range is automatically deleted during automatic generation of a program.

Be sure to confirm the operation of the system using the added programs.

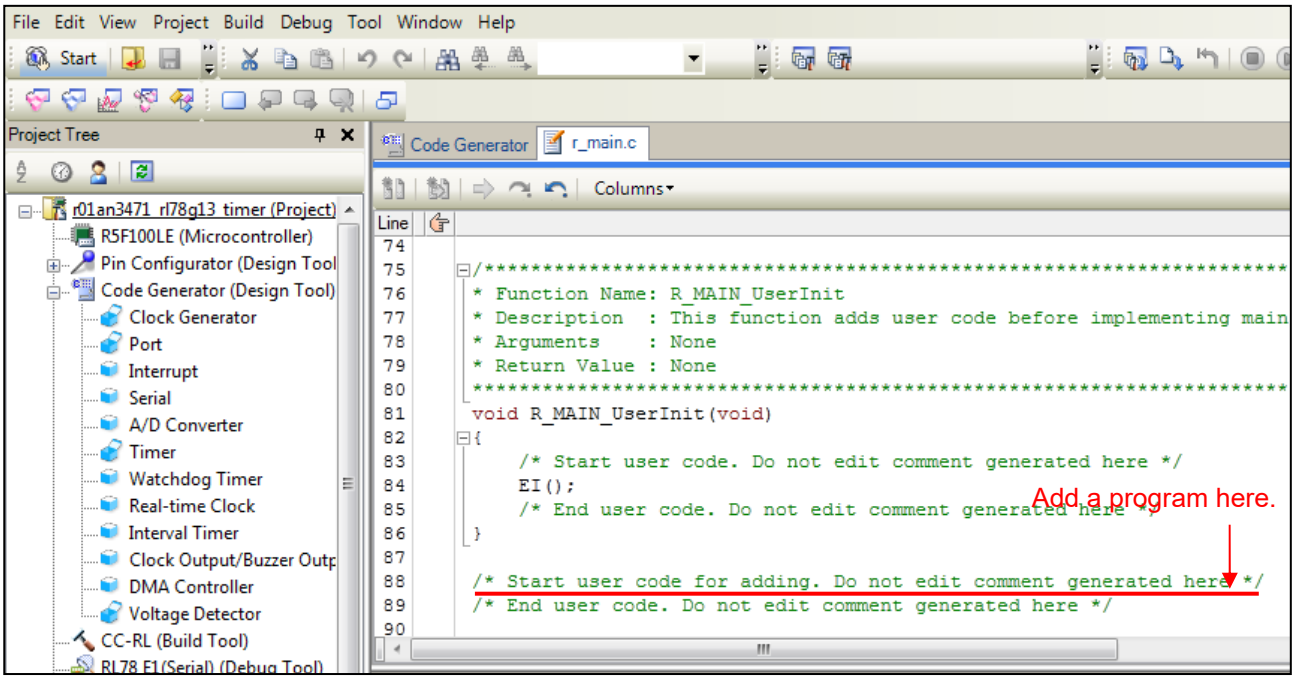


Figure 3.3 Adding Existing Program

3.3 When Code Generator is Not Used

When the code generator is not used, you need to create a new project first with the integrated development environment CS+ or e2studio and then manually create a program for a peripheral function. For details of peripheral functions, see the user's manual for the RL78 family.

4. Replacement Examples

4.1 78K0/Kx1 Sample Program (Serial Interface UART0)

The program for the serial interface UART0 included in “78K0/Kx1,78K0/Kx1+シリアル通信プログラム集” is replaced with the program for RL78/G13. The project file after replacement is “r01an3471_rl78g13_serial”.

This program uses UART0 and repeats data transmission of 0x55 at the transfer speed of 9600 bps at 2-ms intervals. The CPU clock is 10-MHz high-speed system clock.

4.1.1 Porting Source to CC-RL with CcnvCA78K0

(1) Create a list file to specify a C source file to be converted.

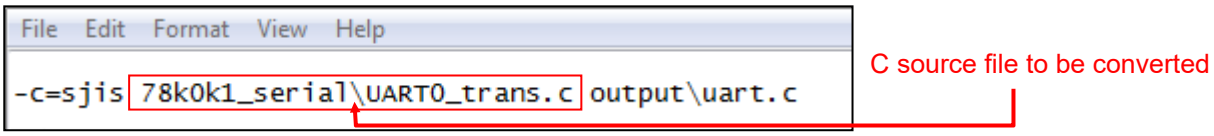


Figure 4.1 Example of Description in List File (Serial Interface UART0)

(2) Launch Command Prompt to convert the C source file specified with the list file.

In addition, the output conversion result file indicates changes.

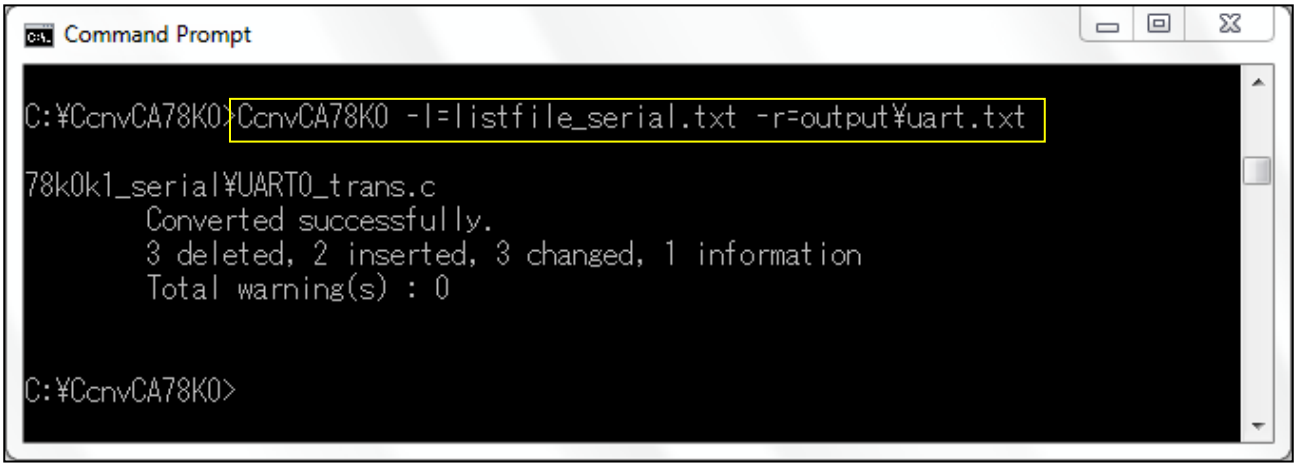


Figure 4.2 CcnvCA78K0 Execution Window (Serial Interface UART0)

The conversion result file indicates the conversion result as shown below. For details of the conversion result, see “CcnvCA78K0 C Source Converter User's Manual (R20UT3684EJ0100)”.

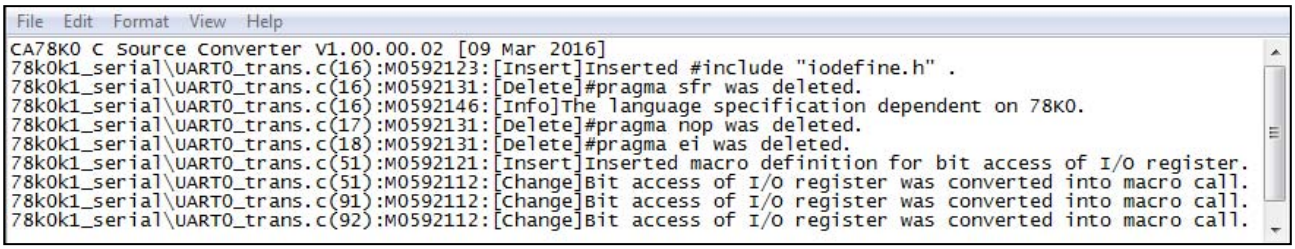
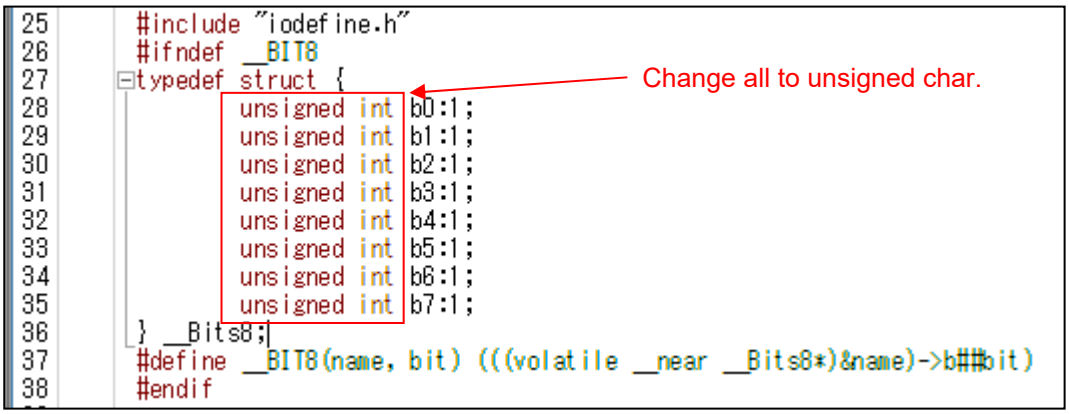


Figure 4.3 Details of Conversion Result (Serial Interface UART0)

(3) Correct the converted C source file.

Bit access to SFRs and the saddr variable are replaced with a type declaration of a bit field and a macro as shown below. When performing bit access to an 8-bit SFR, change unsigned int to unsigned char.




```
25      #include "iodefine.h"
26      #ifndef __BIT8
27      typedef struct {
28              unsigned int b0:1;
29              unsigned int b1:1;
30              unsigned int b2:1;
31              unsigned int b3:1;
32              unsigned int b4:1;
33              unsigned int b5:1;
34              unsigned int b6:1;
35              unsigned int b7:1;
36      } __Bits8;
37      #define __BIT8(name, bit) (((volatile __near __Bits8*)&name)->b##bit)
38      #endif
```

Figure 4.4 Changing Bit Access Description

4.1.2 Generating Programs Automatically

(1) Create a new project with the integrated development environment CS+ or e2studio.

(2) Set each function with the code generator.

Set the CPU clock to 10-MHz high-speed system clock.

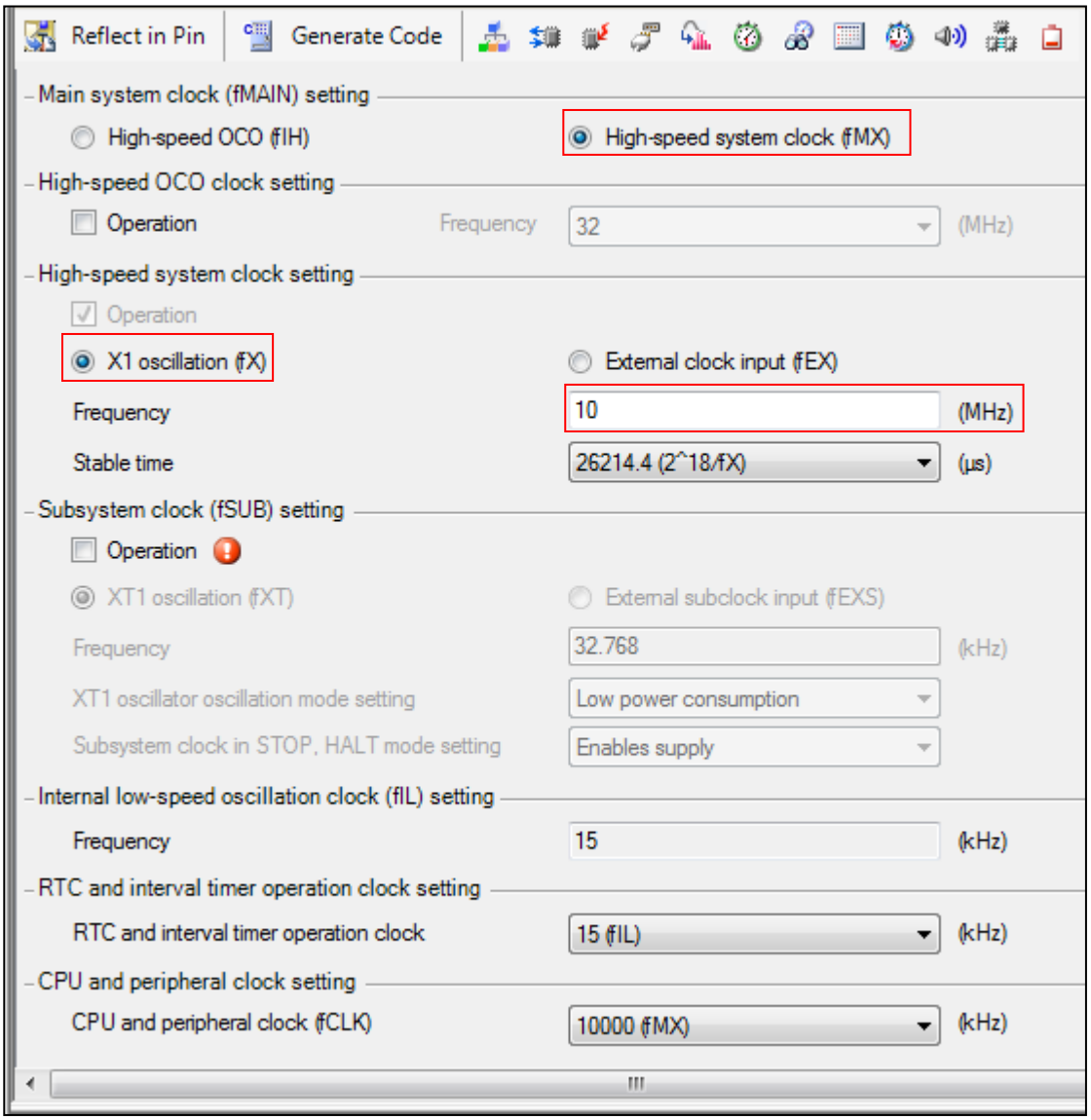


Figure 4.5 Setting Window in Code Generator (Clock)

Set UART0 of the serial array unit which is the equivalent function to the serial interface UART0 of the 78K0 family.

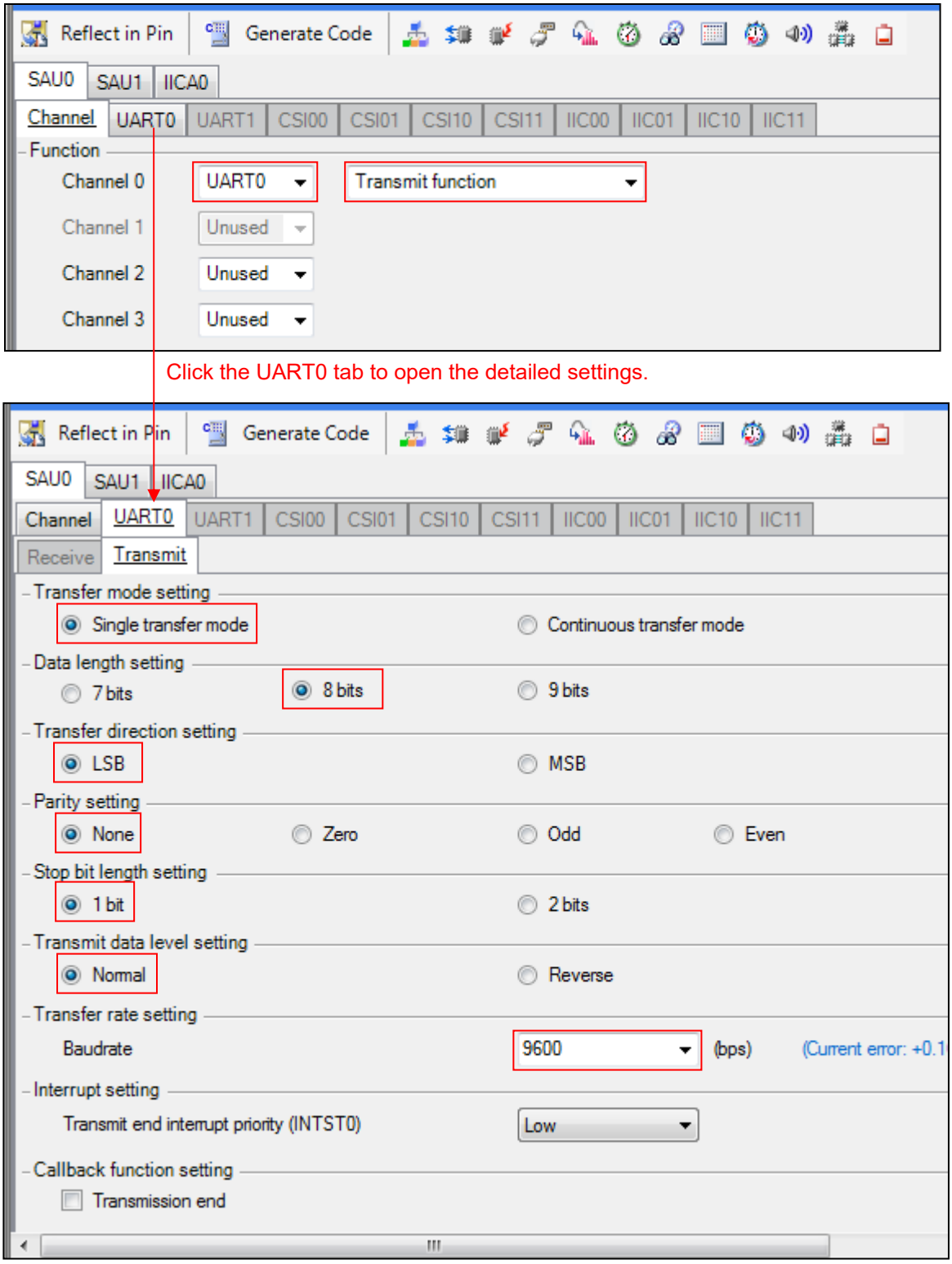


Figure 4.6 Setting Window in Code Generator (Serial Array Unit)

(3) Set ports, watchdog timer, voltage detection circuit, etc. based on your environment.

(4) Click [Generate Code] to generate a file.

4.1.3 Adding Programs

Add the processes of symbol definition and the main function to the program with generated code. Use the programs with generated code for other programs (such as clock setting and UART0 function setting).

- Symbol definition

  Add symbol definition to r_cg_userdefine.h.

Program for 78K0

```
22      ------------------------------------------------------------------------
23          Constants/Variables
24      ------------------------------------------------------------------------
25      */
26
27      #define UART_BAUDRATE_M0     0x3
28      #define UART_BAUDRATE_K0     0x10
29
30      /*status list definition*/
31      #define TRUE              1
32      #define FALSE             0
```

r_cg_userdefine.h file for RL78/G13

```
32      /************************************************************************
33      User definitions
34      ************************************************************************/
35
36      /* Start user code for function. Do not edit comment generated here */
37      #define TRUE              1
38      #define FALSE             0
39      /* End user code. Do not edit comment generated here */
```

Figure 4.7 Replacement of Symbol Definition

- main function

When the code generator for RL78/G13 is used, the R_Systeminit function is executed before the main function is executed. The R_Systeminit function performs the initial setting of the clock and UART0. Thus, only the process indicated in the red box is added manually. The R_UART_Start function starts the operation of UART0.

Program for 78K0

```
43      void main( void )
44      {
45          unsigned short i;
46
47          PCC  = 0x00;                        /* CPU clock: fx */
48          WDTM = 0x77;                        /* Watchdog Timer Stop */
49
50          /* Waiting for oscillation stable time */
51          while( OSTC.0 == 0);
52          MCM0 = 1;                           /* supply clock: X1 */
53          /* Waiting for X1 clock change */
54          while( MCS == 0 );
55
56          UART0_Init();                       /* UART0 initialization function */
57          UART0_Enable();                     /* UART0 enable function */
58
59          while(TRUE)                         /* main loop */
60          {
61              TXS0 = 0x55;
62              /* Waiting for the completion of transmitting */
63              while( DUALIF0 == 0 );
64              DUALIF0 = 0;
65
66              for( i=0 ; i<1000 ; i++ );      /* wait 2 ms */
67          }
68      }
```

r_main.c file for RL78/G13

```
59      void main(void)
60      {
61          R_MAIN_UserInit();
62          /* Start user code. Do not edit comment generated here */
63          {
64              unsigned short i = 0;
65
66              R_UART0_Start();
67
68              while(TRUE)                         /* main loop */
69              {
70                  TXD0 = 0x55;
71                  /* Waiting for the completion of transmitting */
72                  while( STIF0 == 0 );
73                  STIF0 = 0;
74
75                  for( i=0 ; i<4000 ; i++ );      /* wait 2 ms */
76              }
77          }
78          /* End user code. Do not edit comment generated here */
79      }
```

Figure 4.8 Replacement of main Function

4.1.4 Other Items to be Corrected

(1) If UART0 is set with the code generator, interrupt processes are automatically generated. Because interrupts are not used this time, disable interrupt processes.

(2) Because the interrupt function is not used, change “EI();” of the R_MAIN_UserInit function to “DI();”.

(3) Readjust the processing time of the software timer. Because compilers are different, the processing time may vary.

4.1.5 Sample Code After Replacement

Obtain the sample code “an-r01an3471jj0100-rl78-migrate.zip” from the Renesas Electronics Website. “rl78g13_migrate_serial” in the “workspace” folder is the sample code that replaces the program for the serial interface UART0 included in “78K0/Kx1,78K0/Kx1+シリアル通信プログラム集”.

4.2 78K0/Kx2 Sample Program (Interval Timer)

The program included in “78K0/Kx2 サンプル・プログラム インターバル・タイマ編(U19031JJ2V0AN00)” is replaced with the program for RL78/G13. The project file after replacement is “r01an3471_rl78g13_timer”.

This program uses 16-bit timer/event counter 00 to generate an interval interrupt at 1-ms intervals. In the interval interrupt process, P10 is inverted. In addition, P11 is inverted each time 100 interval interrupts are generated.

4.2.1 Porting Source to CC-RL with CcnvCA78K0

(1) Create a list file to specify a C source file to be converted.

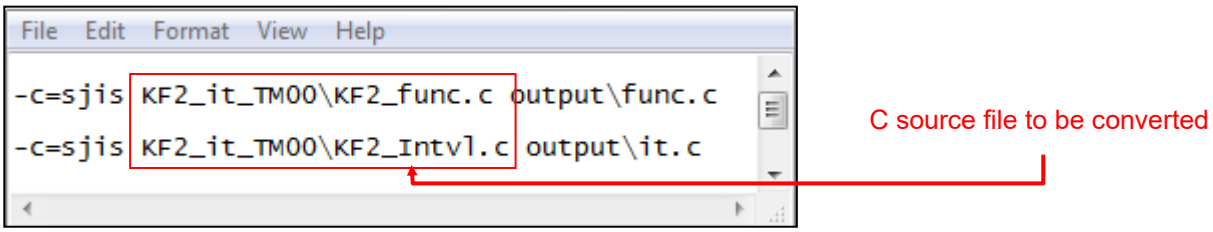


Figure 4.9 Example of Description in List File (Interval Timer)

(2) Launch Command Prompt to convert the C source file specified with the list file.

In addition, the output conversion result file indicates changes.

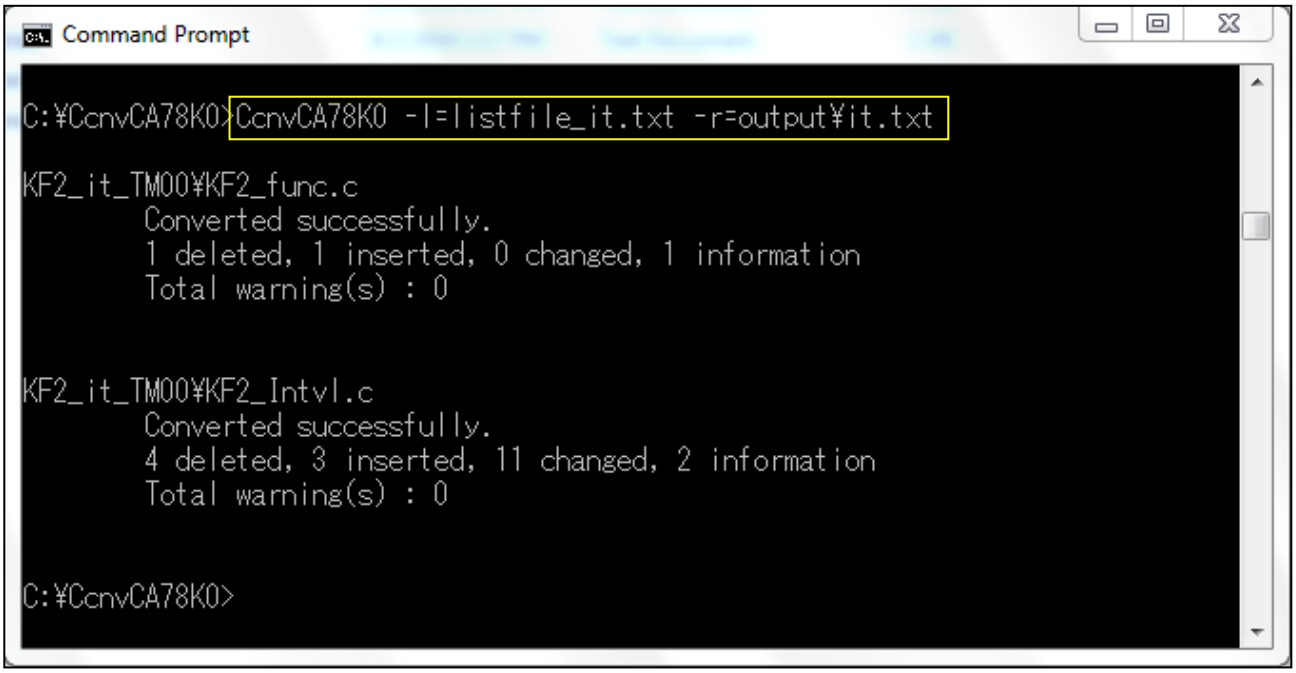


Figure 4.10 CcnvCA78K0 Execution Window (Interval Timer)

The conversion result file indicates the conversion result as shown below. For details of the conversion result, see "CcnvCA78K0 C Source Converter User's Manual (R20UT3684EJ0100)".

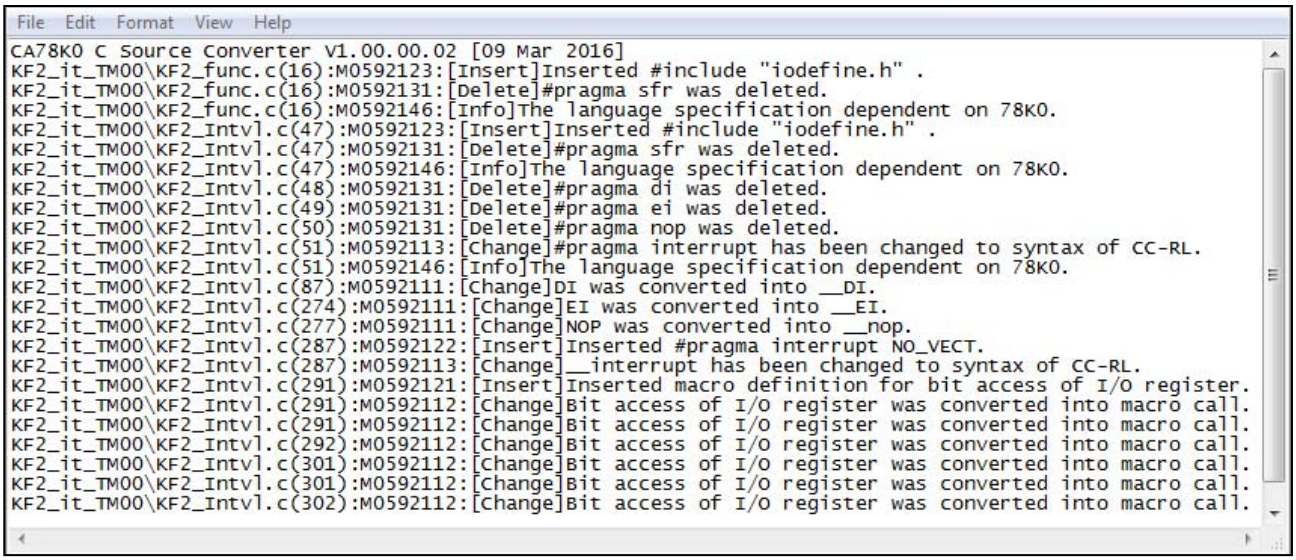


Figure 4.11 Details of Conversion Result (Interval Timer)

(3) Correct the converted C source file.

Bit access to SFRs and the saddr variable are replaced with a type declaration of a bit field and a macro as shown below. When performing bit access to an 8-bit SFR, change unsigned int to unsigned char.

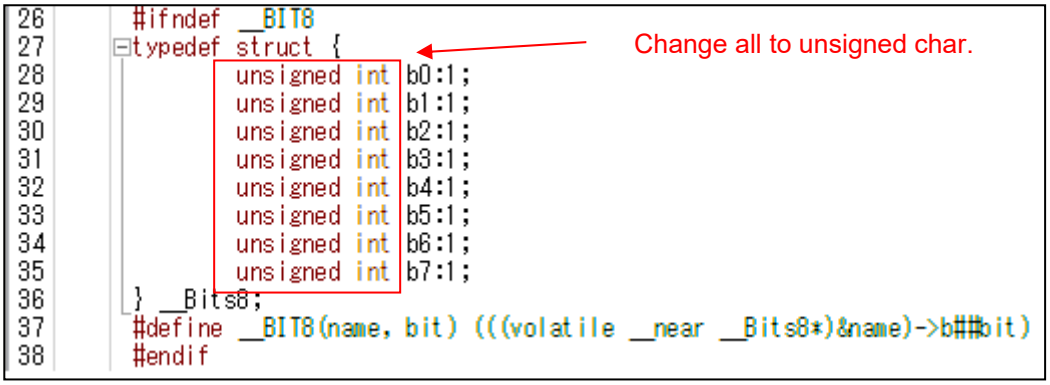


Figure 4.12 Changing Bit Access Description

There may be redundant interrupt function declarations. As CC-RL produces an error in this case, delete the converted #pragma directive.

```
329     //[CcnvCA78K0] __interrupt void fn_intTimerInterval(void)
330     //{
331     #pragma interrupt fn_intTimerInterval
332     void fn_intTimerInterval(void)
333     {
334             g_ucIntCnt++;                                   /* Incremented
335
336
337     //[CcnvCA78K0]  if( P1.0 )       P1.0    =       0;      /* To invert th
338             if( __BIT8( P1, 0 ) )    __BIT8( P1, 0 ) =       0;      /* To i
339     //[CcnvCA78K0]  else             P1.0    =       1;
340             else              __BIT8( P1, 0 ) =       1;
341
342             if(g_ucIntCnt == 100){
343
344     //[CcnvCA78K0]          if( P1.1 )       P1.1    =       0;      /* Inve
345                     if( __BIT8( P1, 1 ) )    __BIT8( P1, 1 ) =       0;
346     //[CcnvCA78K0]          else             P1.1    =       1;
347                     else              __BIT8( P1, 1 ) =       1;
348
349                     g_ucIntCnt              =       0;      /* Initializes
350             }
351     }
```

Delete the #pragma directive.

Figure 4.13 Changing Interrupt Function Declaration

4.2.2 Generating Programs Automatically

(1) Create a new project with the integrated development environment CS+ or e2studio.

(2) Set each function with the code generator.

Set the CPU clock to 8-MHz high-speed on-chip oscillator clock.

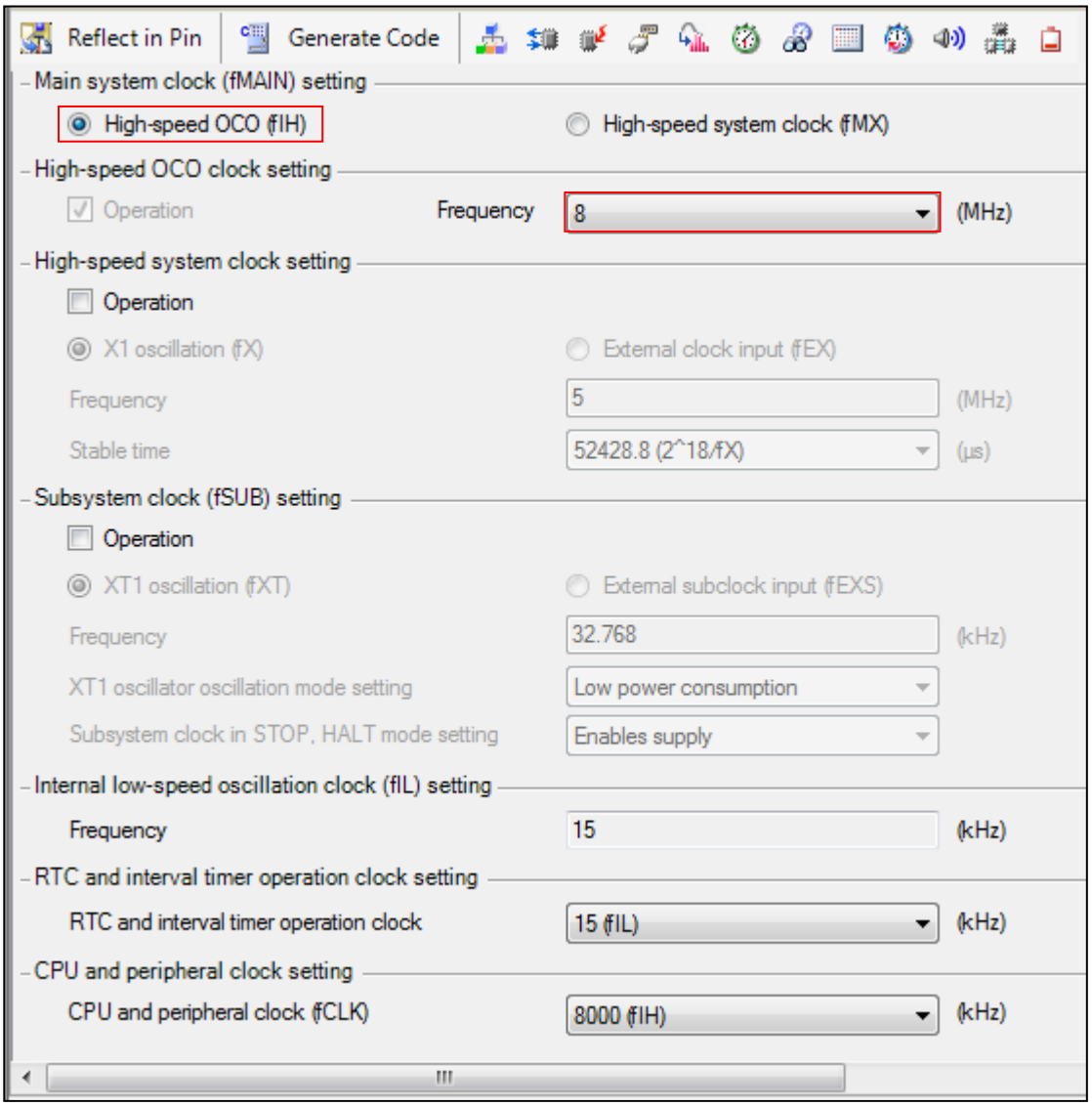


Figure 4.14 Setting Window in Code Generator (Clock)

Set the interval timer function of the timer array unit which is the equivalent function to 16-bit timer/event counter 00 (TM00) of the 78K0 family.

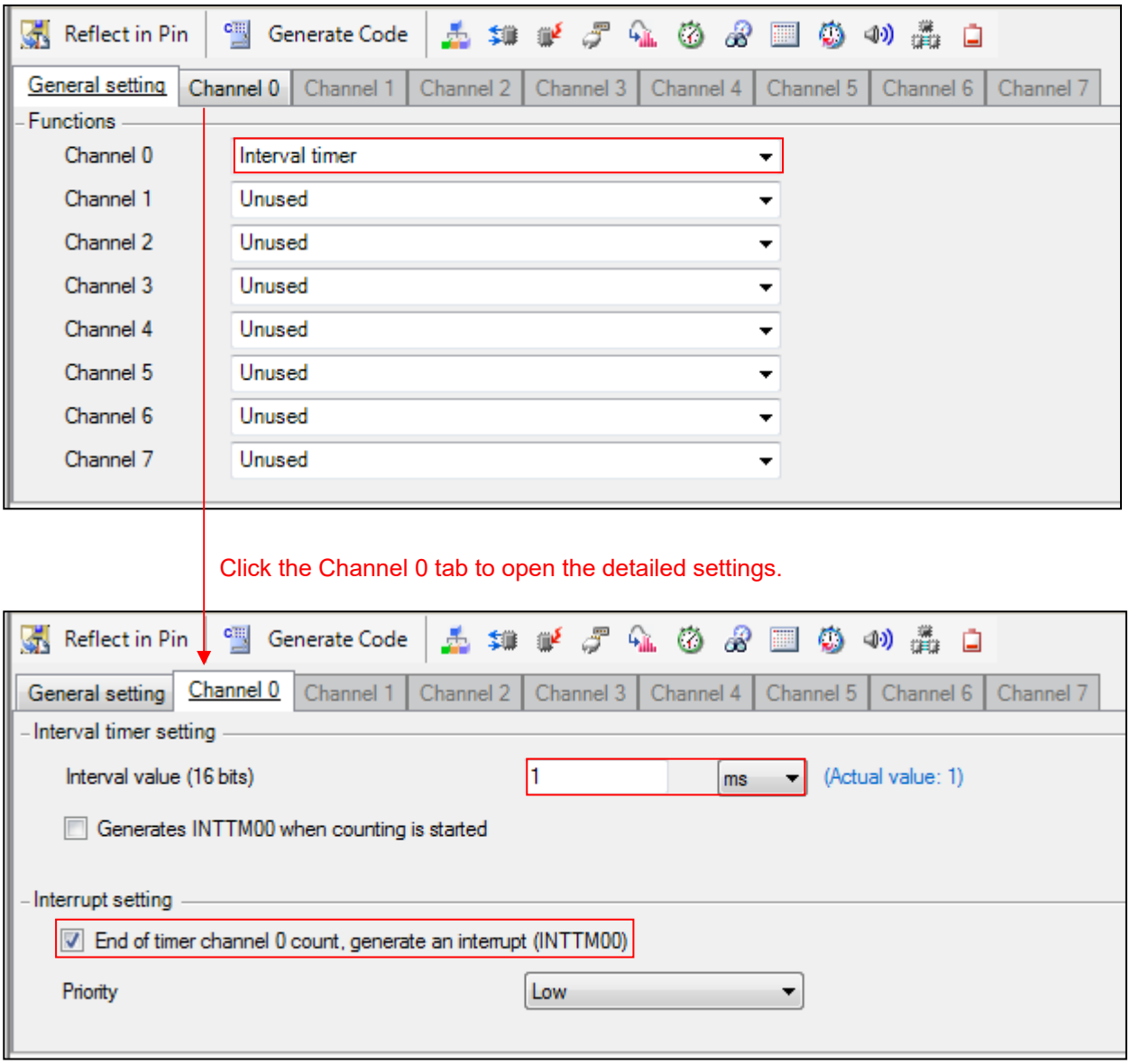


Figure 4.15 Setting Window in Code Generator (Timer Array Unit)

(3) Set ports, watchdog timer, voltage detection circuit, etc. based on your environment.

(4) Click [Generate Code] to generate a file.

4.2.3 Adding Programs

Add the processes of a variable, the main function, and the interrupt function to the program with generated code. Use the programs with generated code for other programs (such as clock setting and timer array unit setting).

- Variable

  Add a variable to r_main.c and r_cg_timer_user.c. Also add a type declaration of a bit field to r_cg_timer_user.c.

Program for 78K0

```
112     /*==========================================================================
113     ;       Global variables and functions
114     ;==========================================================================*/
115     static  unsigned        char    g_ucIntCnt;
```

r_main.c file for RL78/G13

```
46      /***********************************************************************
47      Global variables and functions
48      ***********************************************************************
49      /* Start user code for global. Do not edit comment generated here */
50      unsigned    char    g_ucIntCnt = 0;     /* Count of the interrupt gen
51      /* End user code. Do not edit comment generated here */
```

r_cg_timer_user.c file for RL78/G13

```
45      /***********************************************************************
46      Global variables and functions
47      ***********************************************************************
48      /* Start user code for global. Do not edit comment generated here */
49      #ifndef __BIT8
50      typedef struct {
51          unsigned char b0:1;
52          unsigned char b1:1;
53          unsigned char b2:1;
54          unsigned char b3:1;
55          unsigned char b4:1;
56          unsigned char b5:1;
57          unsigned char b6:1;
58          unsigned char b7:1;
59      } __Bits8;
60      #define __BIT8(name, bit) (((volatile __near __Bits8*)&name)->b##bit)
61      #endif
62
63      extern      unsigned    char    g_ucIntCnt;
64
65      /* End user code. Do not edit comment generated here */
```

Type declaration of bit field

Figure 4.16 Replacement of Variable

- main function

Add the main function process of the program for 78K0 to the main function in r_main.c for RL78/13. For the operation start of the timer array unit, change to R_TAU0_Channel_Start() automatically generated by the code generator for RL78/G13.

Program for 78K0

```
311     void main(void){
312
313             g_ucIntCnt      =       0;              /* Initializes the counter
314             fn_InitTimer();                 /* Initializes the interval timer
315     //[CcnvCA78K0]  EI();                           /* Vector interrupt
316             __EI();                         /* Vector interrupt enable
317
318             while (1){
319     //[CcnvCA78K0]          NOP();
320                     __nop();
321             }
322     }
```

r_main.c file for RL78/G13

```
60      void main(void)
61      {
62          R_MAIN_UserInit();
63          /* Start user code. Do not edit comment generated here */
64          g_ucIntCnt = 0;             /* Initializes the counter */
65          R_TAU0_Channel0_Start();        /* Timer array unit operation start */
66          __EI();                         /* Vector interrupt enable */
67
68          while (1)
69          {
70              __nop();
71          }
72          /* End user code. Do not edit comment generated here */
73      }
```

Figure 4.17 Replacement of main Function

- Interrupt function

  Add an interrupt process to r_tau0_channel0_interrupt() in r_cg_timer_user.c.

Program for 78K0

```
331     #pragma interrupt fn_intTimerInterval
332     void fn_intTimerInterval(void)
333     {
334             g_ucIntCnt++;                                   /* Incremented each time inter
335
336
337     //[CcnvCA78K0]  if( P1.0 )      P1.0    =       0;      /* To invert the P10 each time
338             if( __BIT8( P1, 0 ) )   __BIT8( P1, 0 ) =       0;      /* To invert the P10 e
339     //[CcnvCA78K0]  else            P1.0    =       1;
340             else            __BIT8( P1, 0 ) =       1;
341
342             if(g_ucIntCnt == 100){
343
344     //[CcnvCA78K0]          if( P1.1 )      P1.1    =       0;      /* Inverted each time
345                     if( __BIT8( P1, 1 ) )   __BIT8( P1, 1 ) =       0;      /* Inverted ea
346     //[CcnvCA78K0]          else            P1.1    =       1;
347                     else            __BIT8( P1, 1 ) =       1;
348
349                     g_ucIntCnt              =       0;      /* Initializes the counter */
350             }
351     }
```

r_cg_timer_user.c for RL78/G13

```
73      static void __near r_tau0_channel0_interrupt(void)
74      {
75          /* Start user code. Do not edit comment generated here */
76          g_ucIntCnt++;                   /* Incremented each time interrupt generation
77
78          if( __BIT8( P1, 0 ) ) __BIT8( P1, 0 ) = 0;      /* To invert the P10 each time
79          else __BIT8( P1, 0 ) = 1;
80
81          if(g_ucIntCnt == 100){
82              if( __BIT8( P1, 1 ) ) __BIT8( P1, 1 ) = 0;  /* Inverted each time an inte
83              else __BIT8( P1, 1 ) = 1;
84
85              g_ucIntCnt = 0;         /* Initializes the counter */
86          }
87          /* End user code. Do not edit comment generated here */
88      }
```

Figure 4.18 Replacement of Interrupt Function

4.2.4 Sample Code After Replacement

Obtain the sample code “an-r01an3471jj0100-rl78-migrate.zip” from the Renesas Electronics Website. “rl78g13_migrate_timer” in the “workspace” folder is the sample code that replaces the program included in “78K0/Kx2 サンプル・プログラムインターバル・タイマ編(U19031JJ2V0AN00)”.

4.3 78K0/Kx2 Sample Program (A/D Converter)

The program included in “78K0/Kx2 サンプル・プログラム A/D コンバータ(ZUD-CC-10-0016)” is replaced with the program for RL78/G13. The project file after replacement is “r01an3471_rl78g13_ad”.

This program uses four analog input channels to perform A/D conversion, while switching channels at 1-ms intervals. After a single cycle of 32 ms, in which four channels are sampled eight times, the mean values are saved in the appropriate variables. Depending on the mean values, the LED corresponding to the analog input channel is turned on or off.

4.3.1 Porting Source to CC-RL with CcnvCA78K0

(1) Create a list file to specify a C source file to be converted.

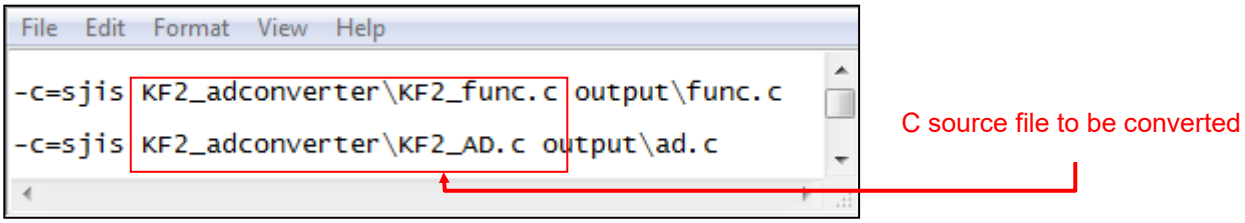


Figure 4.19 Example of Description in List File (A/D Converter)

(2) Launch Command Prompt to convert the C source file specified with the list file.

In addition, the output conversion result file indicates changes.

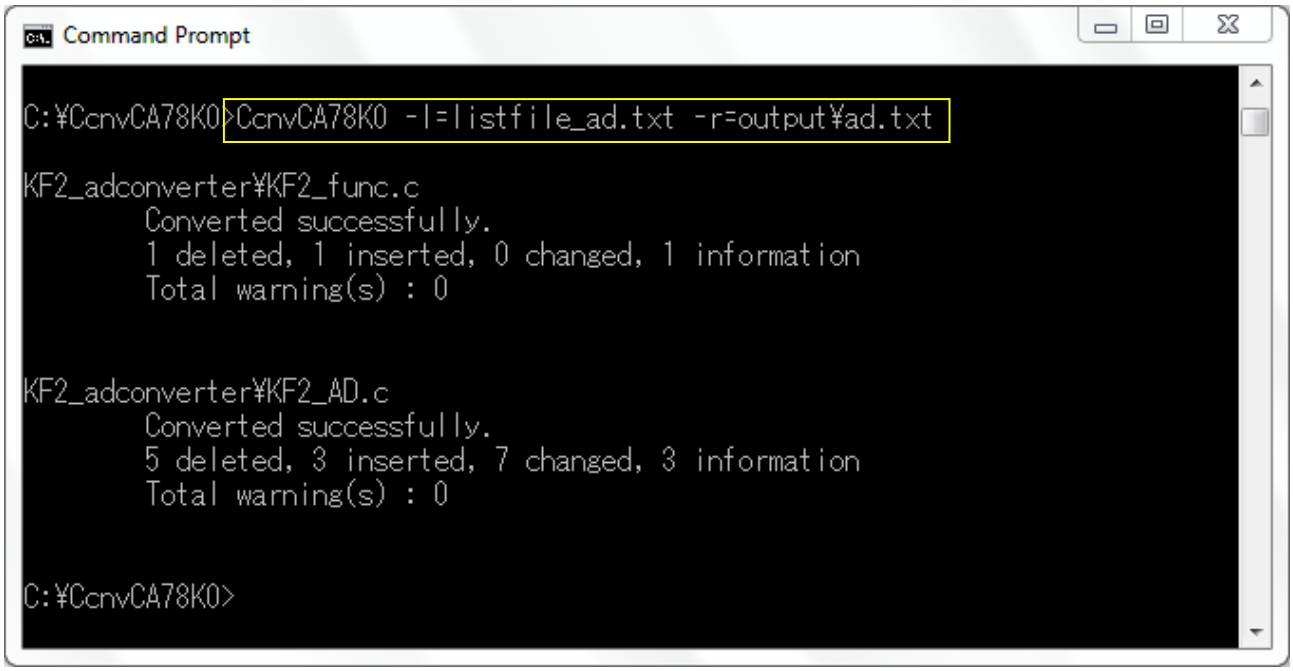


Figure 4.20 CcnvCA78K0 Execution Window (A/D Converter)

The conversion result file indicates the conversion result as shown below. For details of the conversion result, see "CcnvCA78K0 C Source Converter User's Manual (R20UT3684EJ0100)".

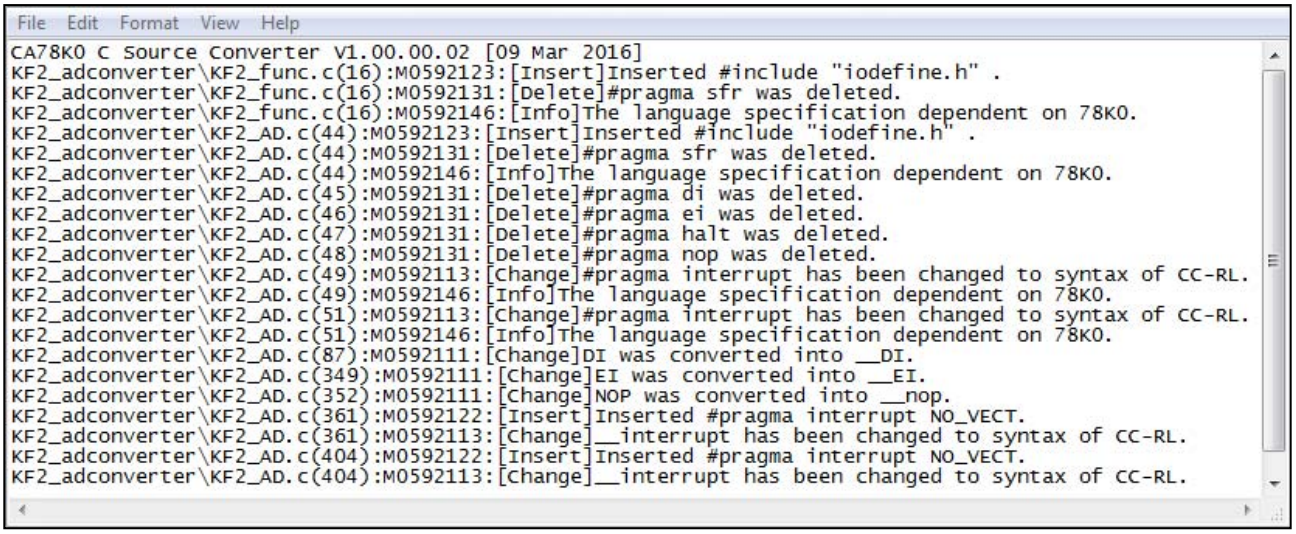


Figure 4.21 Details of Conversion Result (A/D Converter)

(3) Correct the converted C source file.

There may be redundant interrupt function declarations. As CC-RL produces an error in this case, delete the converted #pragma directive.

```
392     //{
393     #pragma interrupt fn_intAdConverter          <-- Delete the #pragma directive.
394     void fn_intAdConverter(void)
395     {
396             if(g_ucAdCnt == 0){
397                     for(g_ucAdCh = 0; g_ucAdCh < 4; g_ucAdCh++){
398                             g_ucAdData[g_ucAdCh] = 0;
399                                                             /* Clear the A / D conversion result buff
400                     }
401                     g_ucAdCh = 0;
402             }
403
404             g_ucAdData[g_ucAdCh] += ADCR >> 6;
405                                                             /* Remove the lower 6 bits of the A / D c
406             g_ucAdCh++;                                     /* Increments the A/D conversion channel
407
408             g_ucAdCh &= 0b00000011;
409             ADS = g_ucAdCh;
410                                                             /* Change the analog input channel*/
411
412             g_ucAdCnt++;                                    /* Increments the A/D conversion counter
413
414             if(g_ucAdCnt >= 32){
415                     for(g_ucAdCh = 0; g_ucAdCh < 4; g_ucAdCh++){
416
417                             g_ucAdCnt = g_ucAdCnt << 1;
418                             g_ucAdData[g_ucAdCh] = g_ucAdData[g_ucAdCh] >> 3;       /* Average the A
419                             if(g_ucAdData[g_ucAdCh] >= 612){                        /* ANI pin more t
420                                     g_ucAdCnt &= 0b11111110;
421                             }
422                             else{                                                   /* ANI pin is les
423                                     g_ucAdCnt |= 0b00000001;
424                             }
425                     }
426                     g_ucAdCnt &= 0b11111111;
427                     P1 = g_ucAdCnt;
428                     g_ucAdCnt = 0;                          /* Clear the A / D conversion counter */
429             }
430             ADCS = 0;                                       /* A/D conversion is stopped */
431             ADIF = 0;
432     }
433     /************************************************************************
434     ;
435     ;       interval timer interrupt processing(INTTM000)
436     ;
437     ************************************************************************/
438     //[CcnvCA78K0] __interrupt void fn_intTimerInterval(void)
439     //{
440     #pragma interrupt fn_intTimerInterval        <-- Delete the #pragma directive.
441     void fn_intTimerInterval(void)
442     {
443             ADCS    =       1;                              /* A/D conversion start */
444     }
```

Figure 4.22 Changing Interrupt Function Declaration

4.3.2 Generating Programs Automatically

(1) Create a new project with the integrated development environment CS+ or e2studio.

(2) Set each function with the code generator.

Set the CPU clock to 8-MHz high-speed on-chip oscillator clock.

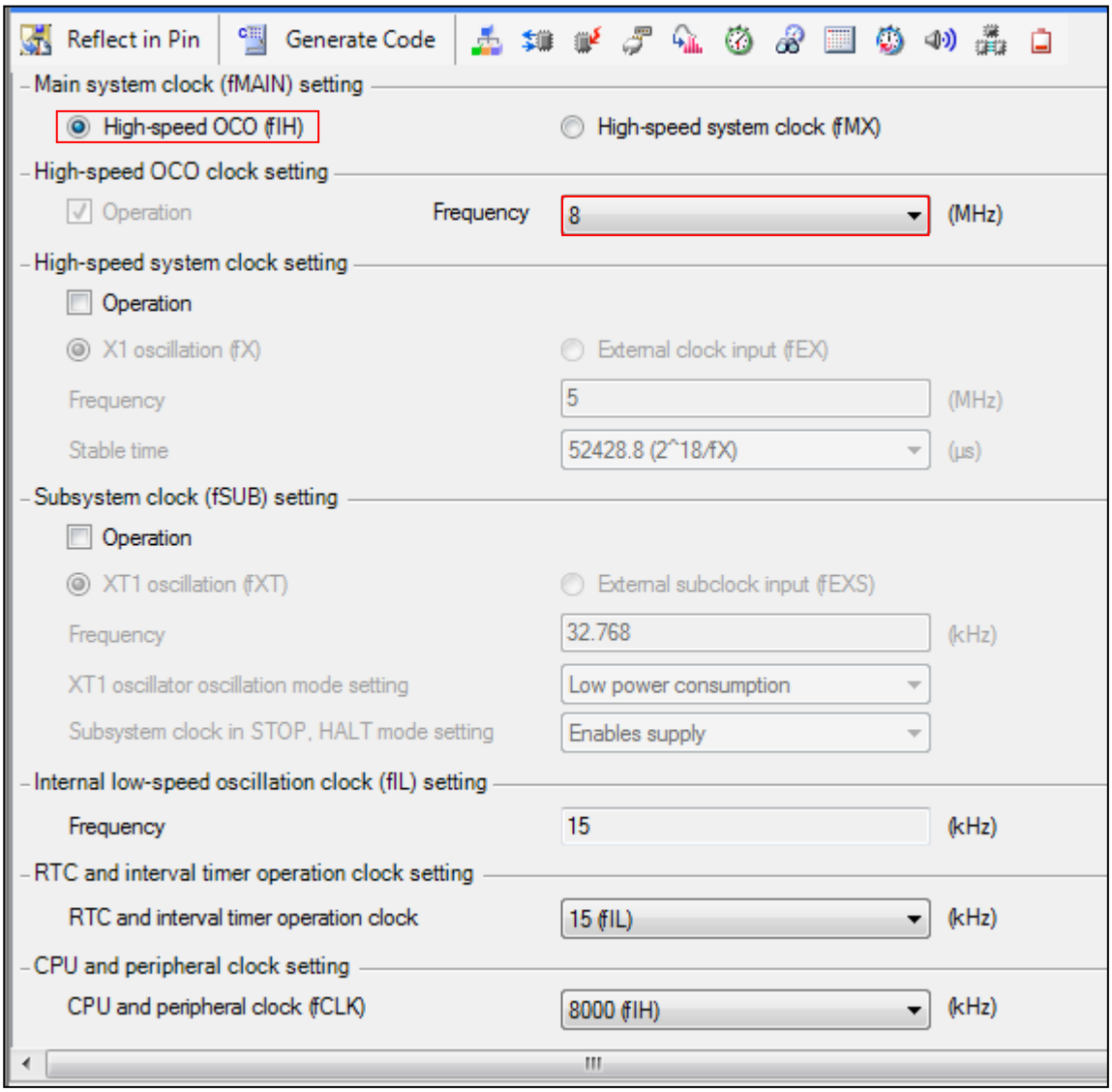


Figure 4.23 Setting Window in Code Generator (Clock)

Set the A/D converter.

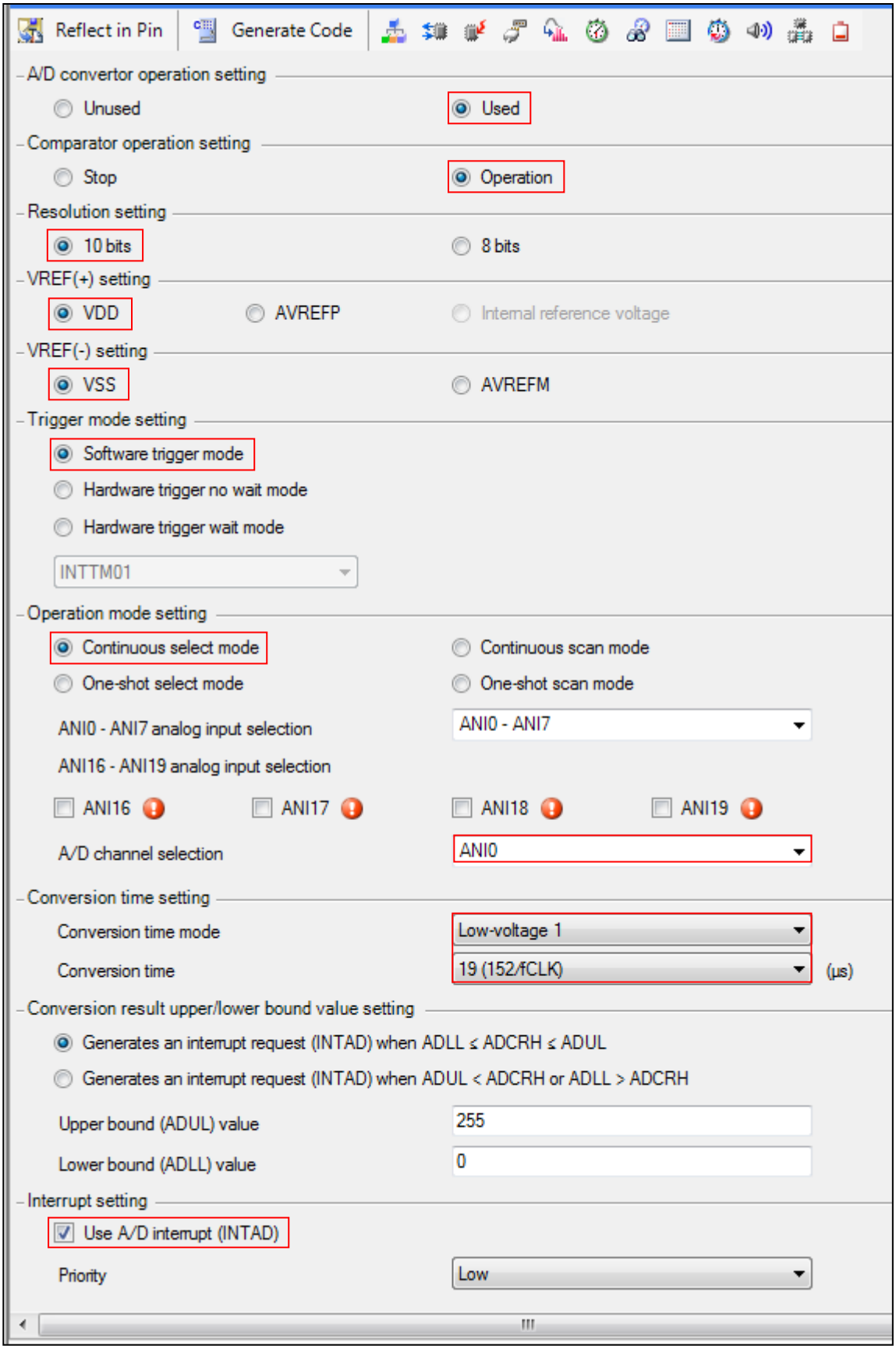


Figure 4.24 Setting Window in Code Generator (A/D Converter)

Set the interval timer function of the timer array unit which is the equivalent function to 16-bit timer/event counter 00 (TM00) of the 78K0 family.

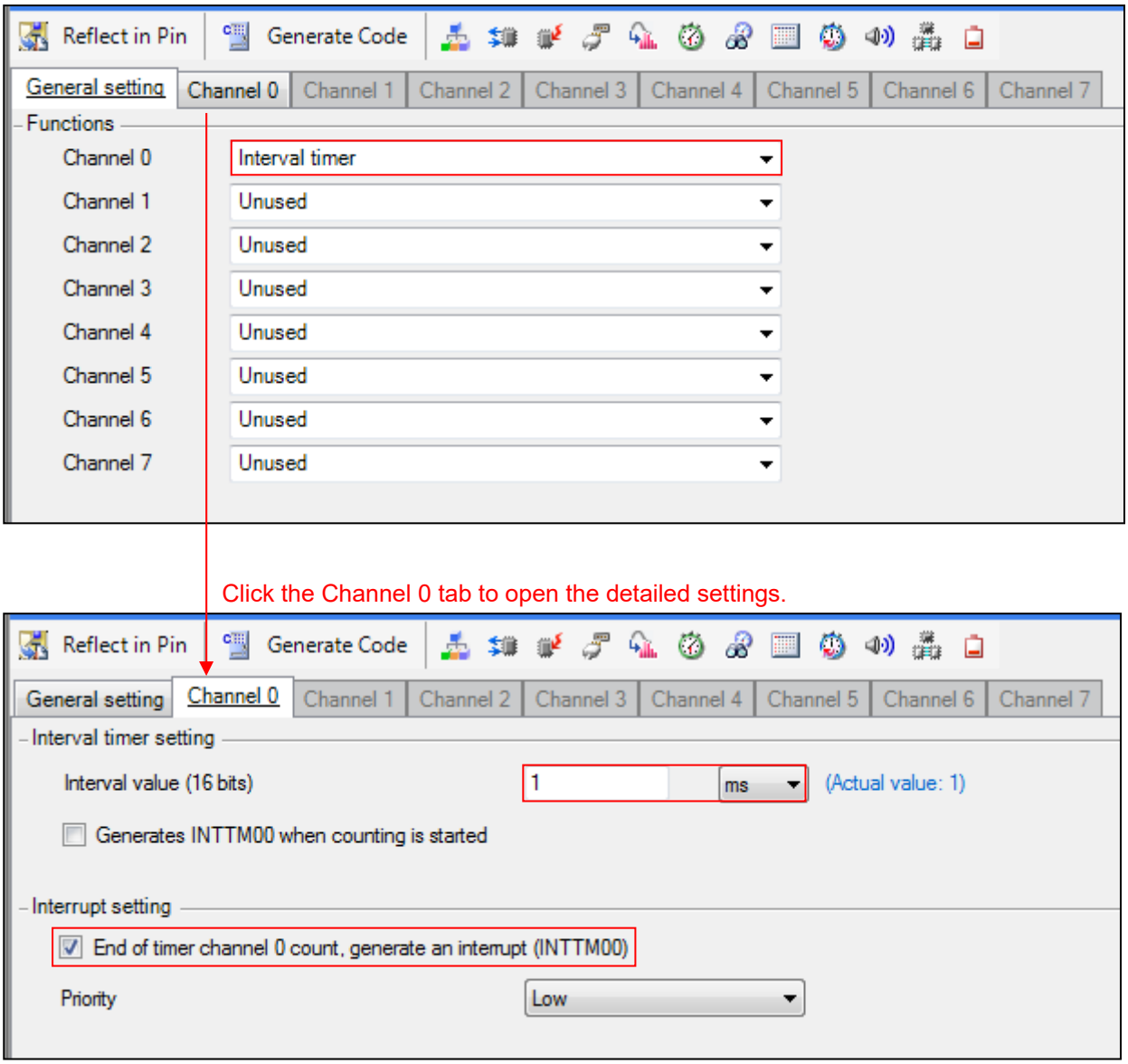


Figure 4.25 Setting Window in Code Generator (Timer Array Unit)

(3) Set ports, watchdog timer, voltage detection circuit, etc. based on your environment.

(4) Click [Generate Code] to generate a file.

4.3.3 Adding Programs

Add the processes of a variable, the main function, and the interrupt function to the program with generated code. Use the programs with generated code for other programs (such as clock setting and A/D converter setting).

- Variable

  Add variables to r_main.c and r_cg_adc_user.c.

Program for 78K0

```
100  /*==========================================================================
101  ;       Global variables and functions
102  ;==========================================================================*/
103  static  unsigned        char    g_ucAdCnt;
104  static  unsigned        char    g_ucAdCh;
105  static  unsigned        short   g_ucAdData[4];
```

r_main.c file for RL78/G13

```
47   /***********************************************************************
48   Global variables and functions
49   ***********************************************************************/
50   /* Start user code for global. Do not edit comment generated here */
51   extern unsigned    char    g_ucAdCnt;
52
53   /* End user code. Do not edit comment generated here */
```

r_cg_adc_user.c file for RL78/G13

```
45   /***********************************************************************
46   Global variables and functions
47   ***********************************************************************/
48   /* Start user code for global. Do not edit comment generated here */
49   unsigned char g_ucAdCnt = 0;
50   unsigned char g_ucAdCh = 0;
51   unsigned short g_ucAdData[4] = {0,0,0,0};
52   /* End user code. Do not edit comment generated here */
```

Figure 4.26 Replacement of Variable

- main function

When the code generator for RL78/G13 is used, the R_Systeminit function is executed before the main function is executed. The R_Systeminit function performs the initial setting of the timer and the A/D converter. The R_TAU0_Channel0_Start function starts the count operation of the timer array unit.

Program for 78K0

```
373     void main(void){
374             fn_InitAd();
375             fn_InitTimer();
376             g_ucAdCnt       =       0;      /* Initialization of variables */
377
378     //[CcnvCA78K0]  EI();
379             __EI();
380
381             while (1){
382     //[CcnvCA78K0]          NOP();
383                     __nop();
384             }
385     }
```

Change

r_main.c file for RL78/G13

```
62      void main(void)
63      {
64          R_MAIN_UserInit();
65          /* Start user code. Do not edit comment generated here */
66          R_TAU0_Channel0_Start();
67          g_ucAdCnt = 0;                  /* Initialization of variables */
68
69          while (1)
70          {
71              __nop();
72          }
73          /* End user code. Do not edit comment generated here */
74      }
```

Figure 4.27 Replacement of main Function

- Interrupt function (timer array unit)

  Add an interrupt function process to r_tau0_channel0_interrupt() in r_cg_timer_user.c.

Program for 78K0

```
439     void fn_intTimerInterval(void)
440     {
441             ADCS    =       1;                      /* A/D conversion start */
442     }
```

r_cg_timer_user.c for RL78/G13

```
57      static void __near r_tau0_channel0_interrupt(void)
58      {
59          /* Start user code. Do not edit comment generated here */
60          ADCS = 1;                       /* A/D conversion start */
61          /* End user code. Do not edit comment generated here */
62      }
```

Figure 4.28 Replacement of Interrupt Function (Timer Array Unit)

- Interrupt function (A/D converter)

  Add an interrupt process to r_adc_interrupt() in r_cg_adc_user.c.

Program for 78K0

```
393     #pragma interrupt fn_intAdConverter
394     void fn_intAdConverter(void)
395     {
396             if(g_ucAdCnt == 0){
397                     for(g_ucAdCh = 0; g_ucAdCh < 4; g_ucAdCh++){
398                             g_ucAdData[g_ucAdCh] = 0;
399                                                             /* Clear the A / D conve
400                     }
401                     g_ucAdCh = 0;
402             }
403
404             g_ucAdData[g_ucAdCh] += ADCR >> 6;
405                                                             /* Remove the lower 6 bi
406             g_ucAdCh++;                                     /* Increments the A/D co
407
408             g_ucAdCh &= 0b00000011;
409             ADS = g_ucAdCh;
410                                                             /* Change the analog inp
411
412             g_ucAdCnt++;                                    /* Increments the A/D co
413
414             if(g_ucAdCnt >= 32){
415                     for(g_ucAdCh = 0; g_ucAdCh < 4; g_ucAdCh++){
416
417                             g_ucAdCnt = g_ucAdCnt << 1;
418                             g_ucAdData[g_ucAdCh] = g_ucAdData[g_ucAdCh] >> 3;
419                             if(g_ucAdData[g_ucAdCh] >= 612){
420                                     g_ucAdCnt &= 0b11111110;
421                             }
422                             else{                           /* ANI pin is less than
423                                     g_ucAdCnt |= 0b00000001;
424                             }
425                     }
426                     g_ucAdCnt &= 0b11111111;
427                     P1 = g_ucAdCnt;
428                     g_ucAdCnt = 0;                  /* Clear the A / D conversion co
429             }
430             ADCS = 0;                                       /* A/D conversion is sto
431             ADIF = 0;
432     }
```

Figure 4.29 Replacement of Interrupt Function (A/D Converter) (1/2)

r_adc_user.c file for RL78/G13

```
 60    static void __near r_adc_interrupt(void)
 61    {
 62        /* Start user code. Do not edit comment generated here */
 63        if(g_ucAdCnt == 0){
 64            for(g_ucAdCh = 0; g_ucAdCh < 4; g_ucAdCh++){
 65                g_ucAdData[g_ucAdCh] = 0;
 66                                        /* Clear the A / D conversion result buffe
 67            }
 68            g_ucAdCh = 0;
 69        }
 70
 71        g_ucAdData[g_ucAdCh] += ADCR >> 6;
 72                                        /* Remove the lower 6 bits of the A / D co
 73        g_ucAdCh++;                     /* Increments the A/D conversion channel *
 74
 75        g_ucAdCh &= 0b00000011;
 76        ADS = g_ucAdCh;
 77                                        /* Change the analog input channel*/
 78
 79        g_ucAdCnt++;                    /* Increments the A/D conversion counter *
 80
 81        if(g_ucAdCnt >= 32){
 82            for(g_ucAdCh = 0; g_ucAdCh < 4; g_ucAdCh++){
 83
 84                g_ucAdCnt = g_ucAdCnt << 1;
 85                g_ucAdData[g_ucAdCh] = g_ucAdData[g_ucAdCh] >> 3;    /* Average the
 86                if(g_ucAdData[g_ucAdCh]    >= 612){                  /* ANI pin mor
 87                    g_ucAdCnt &= 0b11111110;
 88                }
 89                else{                   /* ANI pin is less than 3V */
 90                    g_ucAdCnt |= 0b00000001;
 91                }
 92            }
 93            g_ucAdCnt &= 0b11111111;
 94            P1 = g_ucAdCnt;
 95            g_ucAdCnt = 0;              /* Clear the A / D conversion counter */
 96        }
 97        ADCS = 0;                       /* A/D conversion is stopped */
 98        ADIF = 0;
 99        /* End user code. Do not edit comment generated here */
100    }
```

Figure 4.30 Replacement of Interrupt Function (A/D Converter) (2/2)

4.3.4 Sample Code After Replacement

Obtain the sample code “an-r01an3471jj0100-rl78-migrate.zip” from the Renesas Electronics Website. “rl78g13_migrate_ad” in the “workspace” folder is the sample code that replaces the program included in “78K0/Kx2 サンプル・プログラム A/D コンバータ(ZUD-CC-10-0016)”.

4.4 Conditions for Confirming Operations of Sample Programs

The operations of the sample codes after replacement are confirmed under the following conditions.

Table 4.1 Conditions for Confirming Operations

Item	Description
Microcontroller used	RL78/G13 (R5F100LEA)
Integrated development environment (CS+)	CS+ for CC V4.00.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.02.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^2 studio V4.3.1.001 from Renesas Electronics Corp.
C compiler (e^2 studio)	CC-RL V1.02.00 from Renesas Electronics Corp.
Board used	RL78/G13 target board (QB-R5F100LE-TB) from Renesas Electronics Corp.

5. Sample Code

The sample code is available on the Renesas Electronics website.

6. Reference Documents

RL78 family User's Manual: Software (R01US0015E)

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

CS+ Code Generator Integrated Development Environment User's Manual: Peripheral Function Operation (R20UT3104E)

CcnvCA78K0 C Source Converter User's Manual (R20UT3684E)

78K0/Kx1,78K0/Kx1+ シリアル通信プログラム集

78K0/Kx2 サンプル・プログラム インターバル・タイマ編(U19031JJ2V0AN00)

78K0/Kx2 サンプル・プログラム A/D コンバータ(ZUD-CC-10-0016)

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

Website and Support

Renesas Electronics website
http://japan.renesas.com/

Inquiries
http://japan.renesas.com/inquiry

Revision History	Replacement Guide from 78K0 Family to RL78 Family (CcnvCA78K0)

Rev.	Date	Revision Contents	
		Page	Description
1.00	Nov. 08, 2016	—	First Edition.

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 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.

California Eastern Laboratories, Inc.
4590 Patrick Henry Drive, Santa Clara, California 95054-1817, U.S.A.
Tel: +1-408-919-2500, Fax: +1-408-988-0279

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