RL78/G1M, RL78/G1N

Software Development Using Code Generation tools of RL78/G10

Introduction

This application note shows how to develop RL78G1M orRL78/G1N programs using RL78/G10 code generation.

Code generation tools refer to the RL78 code generation plug-in (CG) when using the C compiler CC-RL or "AP4 for RL78 (AP4)" when using IAR's C compiler.

Target Device

RL78/G1M, RL78G1N

Contents

1. Specifications 3
1.1 Specification Outline 3
1.2 Development Procedure Overview 3
2. Development Procedure Details 4
2.1 Create RL78/G1M or RL78/G1N Project 4
2.2 Create a G10 Project and generate CG output 4
2.3 Modify Functions to be Used 5
2.3.1 Port Configuration Function 5
2.3.2 Clock Generator 6
2.3.3 Timer array unit 7
2.3.4 12-Bit Interval Timer 8
2.3.5 Clock Output/Buzzer Output Control Circuit 8
2.3.6 Watchdog Timer 9
2.3.7 A/D converter 9
2.3.8 Serial Array Unit 11
2.3.9 Real-Time Output Control (G1M Only) 13
2.3.10 Interrupt Functions (INTPx) 14
2.3.11 Key Interrupt Function 17
2.4 Modify Initial Setting Functions 18
2.5 Create the main function 19
2.6 Option Byte Setting 19
2.7 Precautions When Creating User Programs 19
3. Application Development Example 20
3.1 Conditions for Operation Confirmation 20
3.2 Hardware Configuration Example 21
3.3 List of Pins Used 21
3.4 Software Description 22
3.4.1 Option Bytes 22
3.4.2 Flowchart 22
3.5 Development Procedure 25
3.5.1 Developing the initial setting functions (r_cg_systeminit.c) 25
3.5.2 Developing System Functions 26
3.5.3 Developing the main Function (r_cg_main.c) 28
3.5.4 Build 29
4. Sample codes 30
5. Reference 30
Revision History 30

1. Specifications

1.1 Specification Outline

This application note describes how to develop programs for RL78/G1M or G1N using the code generation features (here after CG) of RL78/G10.

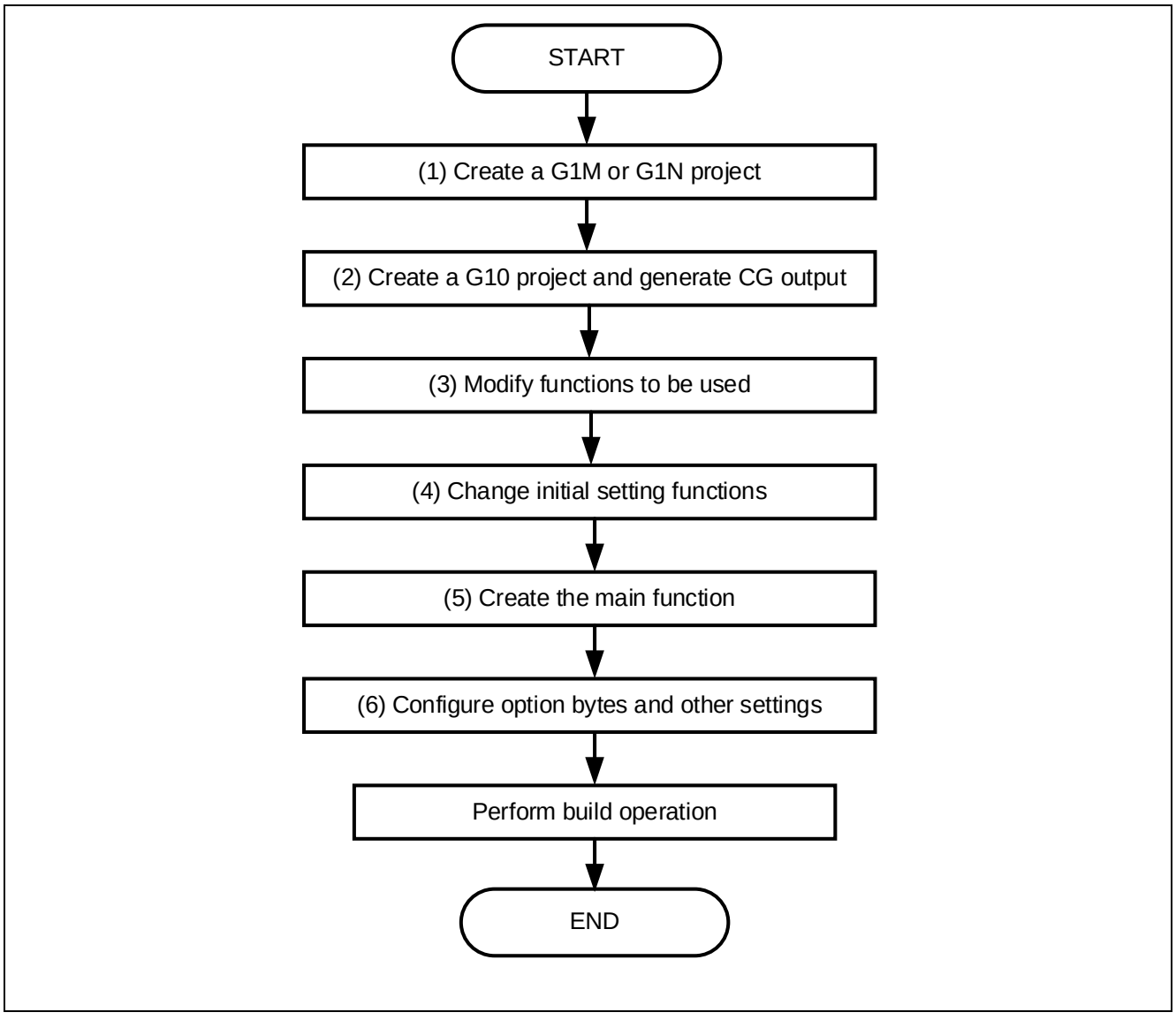
Although the RL78/G1M and RL78/G1N groups of devices do not support CS+ CG features, you can easily develop programs by reusing functions generated by the CG features of RL78/G10 (the device group from which these device groups are derived). This application note provides development procedures to take advantage of all the features of G1M/G1N. Sample codes are also included to facilitate program modification during development.

1.2 Development Procedure Overview

Figure 1.1 shows the G1M/G1N program development flow.

This flow utilizes G10 CG output. The next chapter details specific procedures.

Figure 1.1 /G1M/G1N Program Development Flow



2. Development Procedure Details

2.1 Create RL78/G1M or RL78/G1N Project

For CC-RL, start the Integrated Development Environment and create a project for RL78/G1M or RL78/G1N. When you create a new project, cg_src folder is generated in the folder where the project is generated. cg_src folder contains cstart.asm, hdwinit.asm, main.c and iodefine.h output files.

For IAR's compiler, start IAR Embedded Workbench and generate a project for RL78/G1M or RL78/G1N. To generate a project, select "Create New Project" from the "Project" menu. Then, select "C" from "Project Template" and click "OK" button. In the "Save As" window, enter a project name in the "File Name" field and click the "Save" button. After clicking the "Save" button, main.c is registered in the project. In addition, register the files "ior5f11w68.h" and "ior5f11w68_ext.h" in the directory where IAR is installed.

Note. C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.5\rl78\inc etc.

2.2 Create a G10 Project and generate CG output

CG/AP4 for RL78/G10 outputs the code for the functions used in RL78/G1M or RL78/G1N, and registers the file output by CG/AP4 to the project created in "2.1 Create RL78/G1M or RL78/G1N Project".

In this sample code, the files to be added to the project are saved in the following folder.

For CG:
..\workspace\CS+/ e2studio\G1x\cg_src\
..\workspace\e2studio\G1x\cg_src\

For AP4:
..\workspace\IAR\G1x\cg_src\

Remark. G1x = G1M or G1N

2.3 Modify Functions to be Used

Then, modify the CG output file created in step (2). The sample code uses a CG output file of RL78/G10 (R5F10Y47 16-pin FROM 4K). The G10 CG output codes can be modified or used as is. The portions that need to be modified are those where there are functional differences. Table 2.1 shows the functional differences and whether changes are necessary. For details on the changes, see the corresponding sections describing the CG source.

"-" in the CG function column indicates that the functionality does not require a function generated by CG.

Table 2.1 Difference between RL78/G10 (R5F10Y47) and RL78/G1M, /G1N

Functionality of /G1M, /G1N	Functional difference from G10	Whether G10 CG/AP4 can be reused	Chapters explainig
CPU Core	None	No	-
Port functions	Yes	No	2.3.1
Clock Generator	Yes	Yes (Requires some changes)	2.3.2
Timer Array Unit	None	Yes (May requires some changes)	2.3.3
12-bit Interval Timer	None	Yes(No change)	2.3.4
Clock output /Buzzer output controller	Yes	Yes (Requires some changes)	2.3.5
Watchdog Timer	None	Yes(No change)	2.3.6
A/D Converter	Yes	Yes (Requires some changes)	2.3.7
Serial Array Unit	Yes	Yes (Requires some changes)	2.3.8
Real-Time output controller	Not equipped with G10	No	2.3.9
Interrupt functions	Yes	Yes (Requires some changes)	2.3.10
Key Interrupt function	Yes	Yes (Requires some changes)	2.3.11
Standby function	None	No	-
Reset function	None	No	-
Selectable Power-On-Reset circuit	None	No (Option byte setting)	-
Option Byte	None	No[Note](Set in IDE)	-
On-Chip Debug functions	None	No[Note](Set in IDE)	-
BCD Correction circuit	None	Yes(No change)	-

-:No description
IDE: Integrated Development Environment

Note: For AP4, set in r_cg_main.c

Remark. RL78/G1M or RL78/G1N sample code is available in the G1M or G1N folder under the CS+/e2studio/IAR folder in the workspace folder of this sample code

2.3.1 Port Configuration Function

The G10 CG output code cannot be used. Instead, a configuration source similar to the CG output is provided as a sample code.

Ports used for each functionality (such as input/output pin configuration used for serial communication) are set by the configuration function for the respective functionality, so use this function as the initial pin settings on the system.

R_PORT_Create in the sample code is set to the value after reset.

Register descriptions are provided in comments in the source. Before using the code, modify it by referring to the HUM or comments.

Sample codes
- r_cg_port.h: Function declaration only. To be used as is.
- r_cg_port.c: This code changes input/output settings according to the system.

Sample code modification example (when the initial output is high and output port is P00)

```
void R_PORT_Create(void)
{
/* Port register*/
/* These registers set the output latch value of a port.*/
/* P0  format  P07| P06| P05| P04| P03| P02| P01| P00| */
/* P1  format    0| P16| P15| P14| P13| P12| P11| P10| */
/* P4  format    0|   0|   0|   0|   0|   0|   0| P40| */
/* P12 format      0|   0|P125|   0|   0|   0|   0|   0| */
/* P13 format  P137|   0|   0|   0|   0|   0|   0|   0| */
  P0 = 0x01; //After Reset Value   →Change from 0x00
  P1 = 0x00; //After Reset Value
  P4 = 0x00; //After Reset Value
        :
        :
/*Port mode registers*/
/* "1":input mode  "0":outputmode   */
/* PM0  format  PM07| PM06| PM05| PM04| PM03| PM02| PM01| PM00| */
/* PM1  format     1| PM16| PM15| PM14| PM13| PM12| PM11| PM10| */
/* PM4  format     1|    1|    1|       1|    1|    1|    1| PM40| */
  PM0 = 0xfe;  //After Reset Value   →Change from 0xff
  PM1 = 0xff;  //After Reset Value
  PM4 = 0xff;  //After Reset Value
        :
        :
}
```

2.3.2 Clock Generator

You can reuse the RL78/G10 CG/AP4 output. However, RL78/G1M or RL78/G1N does not have X1 oscillation circuit, only the part of 12-bit interval timer operation clock supply is used.

Sample codes
- **r_cg_cgc.h** (G10 CG output from which unnecessary portions have been removed): To be used as is.
- **r_cg_cgc.c** (G10 CG output from which unnecessary portions have been removed): To be modified only when the fIL supply is stopped.
- **r_cg_cgc_user.c** (G10 CG output without modification): Write a user code as needed.

Sample code modification example

```
void R_CGC_Create(void)
{

/*   ★   OSMC Set select   "_10_CGC_IT_CLK_FIL" or "_00_CGC_IT_CLK_NO" */
    OSMC = _10_CGC_IT_CLK_FIL; Supply fIL clock
      //OSMC = _00_CGC_IT_CLK_NO; In the case of fIL Stop, change to this setting.
}
```

2.3.3 Timer array unit

You can reuse the RL78/G10 CG/AP4 output. However, if the pins used for timer input/output are different, you need to modify the pin setting portion. No change is necessary if pin inputs/outputs are not used, such as in the case of interval timers.

The operating clock of the timer array unit is divided by the CPU/peripheral hardware clock frequency (fCLK). fCLK must be set to the same setting as that used in the RL78/G1M or RL78/G1N if CG/AP4 for RL78/G10 is used. The fCLK is set by the user option byte and the high-speed on-chip oscillator frequency select register (HOCODIV).

Table 2.2 Alternate port of TO0x pin

TO0x	RL78/G10	RL78/G1M,RL78/G1N
TI00	P137	P137
TI01	P04(P40)	P12(P40)
TI02	P05	P16
TII03	P41	P14
TO00	P03	P11
TO01	P04(P40)	P12(P40)
TO02	P05	P16
TO03	P07	P13

Description enclosed in parentheses () is for when PIOR redirection is set.

Sample codes (tau)+: **Square wave output with period 100us at fCLK = 20MHz**
- **r_cg_tau.h** (G10 CG/AP4 output without modification)
- **r_cg_tau.c** (G10 CG/AP4 output Modification required.) To be modified pin setting
- **r_cg_tau_user.c** . (G10 CG/AP4 output without modification) To be used as is.

Sample code modification example

```
void R_TAU0_Create(void)
{
  /* ★TAU setting: Copy CG output code for RL78/G10 to this area */ Use without modification.
    TAU0EN = 1U;    /* supplies input clock */
    TPS0 = _00_TAU_CKM0_FCLK_0 | _00_TAU_CKM1_FCLK_0;
                :
                :
   /* ★TAU setting: End of copy area for CG output code for RL78/G10 */

    /*★ TO00 pin setting */
     /* TO00 = P11 case*/
        PMC1 &= 0xFDU;/*Clear bit 2 */ Use TO00
        P1    &= 0xFDU;/*Clear bit 2 */ Use TO00
        PM1  &= 0xFDU;/*Clear bit 2 */ Use TO00

       /*★ TI02 pin setting */
       /*TI02 = P16 case*/
     //     PMC1 &= 0xBFU; /*Clear bit 6 */ Uncomment in TI02 not use
     //    PM1 |= 0x40U; /*Set bit 6 */ Uncomment in TI02 not use
```

2.3.4 12-Bit Interval Timer

You can use the RL78/G10 CG/AP4 output files without modification.
However, since the alternate ports are different, it is necessary to change the pin settings.

Sample codes(it) 100 ms interval timer function.
- **r_cg_it.h** (G10 CG/AP4 output without modification)
- **r_cg_it.c** (G10 CG/AP4 output without modification)
- **r_cg_it_user.c** (G10/AP4 CG output without modification) To be used as is.

2.3.5 Clock Output/Buzzer Output Control Circuit

You can reuse the RL78/G10 CG/AP4 output, but you need to modify the used pin portion because the output port is different from that of G10.
Note that the output port also depends on the PIOR setting (r_cg_systemin.c).
The sample code is a function with 1250 kHz (fMain/16) output.

Sample codes(pclbuz) Select fMAIN/ (2^4). fCLK = 1.25 MHz output clock at 20 MHz
- **r_cg_pclbuz.h** (G10 CG/AP4 output without modification): To be used as is.
- **r_cg_pclbuz.c** (Use sample code): To be used as is
- **r_cg_pclbuz_user.c** (G10 CG/AP4 output without modification): To be used as is.

Sample code modification example

```
void R_PCLBUZ0_Create(void)
{
    PCLOE0 = 0U;    /* disable PCLBUZ0 operation */
/* ★PCLBUZ0 output clock selection (CCS02 - CCS00) */ Select a division ratio from the above.
// CKS0 =_00_PCLBUZ_OUTCLK_fMAIN0  (0x00U)    /* fMAIN */
// CKS0 =_01_PCLBUZ_OUTCLK_fMAIN1  (0x01U)    /* fMAIN/2 */
// CKS0 =_02_PCLBUZ_OUTCLK_fMAIN2  (0x02U)    /* fMAIN/2^2 */
// CKS0 =_03_PCLBUZ_OUTCLK_fMAIN3  (0x03U)    /* fMAIN/2^3 */
CKS0 =_04_PCLBUZ_OUTCLK_fMAIN4  (0x04U)    /* fMAIN/2^4 */
// CKS0 =_05_PCLBUZ_OUTCLK_fMAIN5  (0x05U)    /* fMAIN/2^11 */
// CKS0 =_06_PCLBUZ_OUTCLK_fMAIN6  (0x06U)    /* fMAIN/2^12 */
// CKS0 =_07_PCLBUZ_OUTCLK_fMAIN7  (0x07U)    /* fMAIN/2^13 */

/*★ PCLBUZ0 pin setting */ For P10 output, disable P40 output and activate the following settings.
/*P10 Output case*/
//  PIOR &= 0xFEU;  /* after RESET*/
   PMC1 &= 0xFEU;
   POM1 &= 0xFEU;
   P1 &= 0xFEU;
   PM1 &= 0xFEU;

/*P40 Output case*/ For P40 output, disable P10 output and activate the following settings.
//   PIOR |= 0x01U;
//   PMC4 &= 0xFEU;
//   P4 &= 0xFEU;
//   PM4 &= 0xFEU;
```

2.3.6 Watchdog Timer

You can reuse the RL78/G10 CG/AP4 output as is.
Option byte settings are required. The sample code is a CG output with the maximum overflow value.
Use the three RL78/G10 CG/AP4 output files without modification.

Sample codes
- **r_cg_wdt.h** (G10 CG/AP4 output without modification)
- **r_cg_wdt.c** (G10 CG/AP4 output without modification)
- **r_cg_wdt_user.c** (G10/AP4 CG output without modification) :To be used as is.

2.3.7 A/D converter

You can reuse the RL78/G10 CG output, but the conversion pins are different from those of G10.
The table shows the differences. Before using the code, change the CG output pin settings.

Table 2.3 A/D Converter Difference between RL78/G10 (R5F10Y47) and RL78/G1M or G1N.

ANIx	ADS Value	RL78/G10	RL78/G1M, RL78G1N
ANI0	00H	P01	P07
ANI1	01H	P02	P10
ANI2	02H	P03	P11
ANI3	03H	P04	P12
ANI4	04H	P05	P13
ANI5	05H	P06	P14
ANI6	06H	P07	P15
ANI7	07H	—	P16

Sample codes
- **r_cg_adc.h** (Use sample code)
- **r_cg_adc.c** (Use sample code): To be used as is
- **r_cg_adc_user.c** (G10 CG/AP4 output without modification): To be used as is.

Sample code modification example

Change the portions marked with ★.

```
void R_ADC_Create(void)
{
    ADCEN = 1U;  /* supply AD clock */
    ADM0 = _00_AD_ADM0_INITIALVALUE;  /* disable AD conversion and clear ADM0 register */
    ADMK = 1U;  /* disable INTAD interrupt */
    ADIF = 0U;  /* clear INTAD interrupt flag */
    /* Set INTAD low priority */
    ADPR1 = 1U;
    ADPR0 = 1U;
  /* ★Set ANI pin select*/ Change the register setting of the ANIx pin to be used to active.
  /*ANI0 case set P07*/
     PMC0 |= 0x80U;
     PM0  |= 0x80U;
   /*ANI1 case set P10*/
     // PMC1 |= 0x01U;
     // PM1  |= 0x01U;
   /*ANI2 case set P11*/
     // PMC1 |= 0x02U;
     // PM1  |= 0x02U;
   /*ANI3 case set P12*/
     // PMC1 |= 0x04U;
     // PM1  |= 0x04U;
   /*ANI4 case set P13*/
```

```
      // PMC1 |= 0x08U;
     // PM1  |= 0x08U;
   /*ANI5 case set P14*/
     // PMC1 |= 0x10U;
     // PM1  |= 0x10U;
   /*ANI6 case set P15*/
     // PMC1 |= 0x20U;
     // PM1  |= 0x20U;
   /*ANI7 case set P16*/
     // PMC1 |= 0x40U;
     // PM1  |= 0x40U;


   /* ★AD converter mode register 0 (ADM0) */ Select the operation mode setting.
   /*_00_AD_ADM0_INITIALVALUE       (0x00U) */
   /* AD conversion operation control (ADCS)  */
   /*_80_AD_CONVERSION_ENABLE       (0x80U) enable AD conversion operation control */
   /*_00_AD_CONVERSION_DISABLE      (0x00U) disable AD conversion operation control */
   /* AD conversion clock selection (FR1, FR0) */
   /*_00_AD_CONVERSION_CLOCK_8      (0x00U) | fCLK/8 */
   /*_08_AD_CONVERSION_CLOCK_4      (0x08U) | fCLK/4 */
   /*_10_AD_CONVERSION_CLOCK_2      (0x10U) | fCLK/2 */
   /*_18_AD_CONVERSION_CLOCK_1      (0x18U) | fCLK/1 */
   /* Specification AD conversion time mode (LV0) */
   /*_00_AD_TIME_MODE_NORMAL_1      (0x00U) | normal 1 mode 23(21)fAD ():when 8bit*/
   /*_02_AD_TIME_MODE_NORMAL_2      (0x02U) | normal 2 mode 17(15)fAD ():when 10bit*/
   ADM0 = _00_AD_CONVERSION_CLOCK_8 | _00_AD_TIME_MODE_NORMAL_1;/*Conv time 9.2us */

   /* ★    AD resolution selection (ADTYP) */ Set either one of them.
   /*_00_AD_RESOLUTION_10BIT (0x00U) | 10 bits */
   /*_01_AD_RESOLUTION_8BIT  (0x01U) | 8 bits */
   ADM2 = _00_AD_RESOLUTION_10BIT;

   /* ★ Select ADI Channel */ Select a conversion channel.
   /*_00_AD_INPUT_CHANNEL_0  (0x00U) | ANI0 */
   /*_01_AD_INPUT_CHANNEL_1  (0x01U) | ANI1 */
   /*_02_AD_INPUT_CHANNEL_2  (0x02U) | ANI2 */
   /*_03_AD_INPUT_CHANNEL_3  (0x03U) | ANI3 */
   /*_04_AD_INPUT_CHANNEL_4  (0x04U) | ANI4 */
   /*_05_AD_INPUT_CHANNEL_5  (0x05U) | ANI5 */
   /*_06_AD_INPUT_CHANNEL_6  (0x06U) | ANI6 */
   /*_07_AD_INPUT_CHANNEL_7  (0x07U) | ANI7 */
    ADS = _00_AD_INPUT_CHANNEL_0;

   /*★ AD comparator ADCE=1:enable ADCE=0:disable  */ Enable (or disable) ADCE at the initial setting.
     ADCE = 1U;
}
```

2.3.8 Serial Array Unit

You can reuse the RL78/G10 CG/AP4 output. However, since the alternate ports for serial I/O pins are different, the pin settings must be changed. The alternate ports are also changed by the PIOR setting (r_cg_systemint.c).
In addition, RL78/G10 has CSI01 and IIC00 which are not included in RL78/G1M and RL78/G1N.

Table 2.4 Alternate ports for Serial I/O

Function		RL78/G10	RL78/G1M		RL78/G1N	
			PIOR7=0	PIOR7=1	PIOR4,5,6=0	PIOR4=1 注1 or PIOR5=1 or PIOR6=1
CSI00	SO01	P00	P06	P10	P06	P10
	SI00	P01	P07	P15	P07	P15
	SCK00	P02	P10	P16	P10	P16Note2
UART0	TXD0	P00	P06	P10	P06	P10
	RXD0	P01	P07	P15	P07	P15

Note 1: Simultaneous setting of PIOR0 = 1 and PIOR1 = 1 is prohibited
2: Simultaneous setting of PIOR3 = 1 and PIOR14= 1 is prohibited
3: Setting PIOR4, PIOR5, and PIOR6 to multiple 1s is prohibited.

Sample code (sau) is available for UART and CSI.

Sample codes(sau)
When using UART: \G1M\cg_src\SAU\UART folder
- **r_cg_sau.h** (G10 CG/AP4 output without modification)
- **r_cg_sau.c** (Modification required from G10 CG/AP4 output): Port settings
- **r_cg_sau_user.c** (G10 CG/AP4 output without modification)

When using CSI: \G1M\cg_src\SAU\CSI folder
- **r_cg_sau.h** (G10 CG/AP4 output without modification)
- **r_cg_sau.c** (Modification required from G10 CG/AP4 output): Port settings required.
- **r_cg_sau_user.c** (G10 CG/AP4 output without modification)

For UART:
Sample code modification example (UART using P06 and P07)

```
void R_SAU0_Create(void)
{
    SAU0EN = 1U;    /* supply SAU0 clock */
    NOP();
    NOP();
    NOP();
    NOP();
 /* ★Set SAU0 Clock */ Copy from the G10 CG output.
    SPS0 = _04_SAU_CK00_FCLK_4 | _40_SAU_CK01_FCLK_4;
 /*     */
    R_UART0_Create();
}
                    :
                    :
```

```
void R_UART0_Create(void)
{

  /* ★UART Function */
  /* ★IF UART setting change,  G10CG output code copy to this area */  Copy from the G10 CG output.
    ST0 |= _02_SAU_CH1_STOP_TRG_ON | _01_SAU_CH0_STOP_TRG_ON;    /* disable UART0 receive
and transmit */
          :
           :
/* ★Port setting for UART: End of copy area for CG output code for RL78/G10*/
                                                                              Select use pin for UART
```

For RL78/G1M

```
    /*P07/RxD0 & P06/TxD0 case */
      PIOR &= 0x7FU;  Change to PIOR7=0
      PMC0 &= 0x3FU;  Set the P06 and P07 are digital port.
      PM0 |= 0x80U;

      P0 |= 0x40U;        Set the P06
      PM0 &= 0xBFU;

    /*P15/RxD0 & P10/TxD0 case */
      //PIOR |= 0x80U;  Change to PIOR7=1
      //PM1 |= 0x20U;

        //P1 |= 0x01U;
        //PM1 &= 0xFEU;
```

For RL78/G1N

```
    /*P07/RxD0 & P06/TxD0 case */
      PIOR &= 0x7FU;  Change to PIOR7=0
      PMC0 &= 0x3FU;  Set the P06 and P07 are digital port.
      PM0 |= 0x80U;

      P0 |= 0x40U;     Set the P06
      PM0 &= 0xBFU;

    /*P137/RxD0 & P01/TxD0 case for RL78/G1N*/
      //PIOR |= 0x10U;  Change to PIOR4=1,PIOR5,6=0
      //PIOR &= 0x9FU;

      //P0 |= 0x02U;    Set the P01, Not requireP137 setting
      //PM0 &= 0xFDU;

    /*P137/RxD0 & P16/TxD0 case for RL78/G1N*/
      //PIOR |= 0x40U;    Change to PIOR6=1,PIOR4,5=0
      //PIOR &= 0xCFU;

      //PMC1 &= 0xBFU;  Set the P01, Not requireP137 setting
      //P1 |= 0x40U;
      //PM1 &= 0xBFU;

}
```

2.3.9 Real-Time Output Control (G1M Only)

RL78/G10 does not have this functionality. Therefore, you need to create this functionality.

Prepare sample codes for setting register values according to the setup procedure. Make changes as needed.
The timer output initialization function used for real-time output must be created in “2.3.3 Timer Array Unit”.

Sample codes
- **r_cg_rto.h** (R_RTO_Create function declaration only.)
- **r_cg_rto.c (**R_RTO_Create)

2.3.10 Interrupt Functions (INTPx)

You can reuse the RL78/G10 CG/AP4 output, However, if the ports for the INTPn pins are different, it is necessary to change the pin settings. In addition, if INTP4 and INTP5 are used, a new code must be created.

Table 2.5 Alternate port of INTPn

INTPn	RL78/G10	RL78/G1M,RL78/G1N
INTP0	P137	P137
INTP1	P00 (P03)	P06(P11)
INTP2	P41 (P122)	P15
INTP3	P06 (P121)	P14
INTP4	None	P01
INTP5	None	P00

Note: Description enclosed in parentheses () is for when PIOR redirection is set.

Sample codes

- **r_cg_intp.h** (Use sample code) :modified as necessary
- **r_cg_intp.c** (Use sample code) :modified as necessary
- **r_cg_intp_user.c** (Use sample code): modified as necessary

Changes to r_cg_intp.h

r_cg_intp.h

```
/*★INTPn setting*/ Activate the INTPn pin to be used.
void R_INTC_Create(void); When using INTPn
void R_INTC0_Start(void); When using INTP0
void R_INTC0_Stop(void); When using INTP0
//void R_INTC1_Start(void);
//void R_INTC1_Stop(void);
//void R_INTC2_Start(void);
//void R_INTC2_Stop(void);
//void R_INTC3_Start(void);
//void R_INTC3_Stop(void);
//void R_INTC4_Start(void);
//void R_INTC4_Stop(void);
//void R_INTC5_Start(void);
//void R_INTC5_Stop(void);
```

Changes to r_cg_intp.c

The portions that need to be changed are marked with ★.

```
/*★Priority setting*/Change the priority of interrupts, if necessary.
/* PPR1x = 0, PPR0x = 0: Level 0 (higher priority level) */
/* PPR1x = 0, PPR0x = 1: Level 1 */
/* PPR1x = 1, PPR0x = 0: Level 2 */
  /* PPR1x = 1, PPR0x = 1: Level 3 (lower priority level) */


  /* Set INTP0 low priority */
    //   PPR10 = 1U; The default setting is Level 3.
    //   PPR00 = 1U;
  /* Set INTP1 low priority */
    //   PPR11 = 1U;
    //   PPR01 = 1U;
         :
```

```
/*★Interrupt edge setting*/
/* EGPx = 0, EGNx = 0: Edge detection disabled */
/* EGPx = 0, EGNx = 1: Falling edge */
/* EGPx = 1, EGNx = 0: Rising edge */
/* EGPx = 1, EGNx = 0: Both rising and falling edges*/

*Delete unnecessary code sections.
  EGN0 = _01_INTP0_EDGE_FALLING_SEL :Eenable the falling edge of INTP0
  //EGP0 = _01_INTP0_EDGE_RISING_SEL | _02_INTP1_EDGE_RISING_SEL |・・・・

/*★INTPn port setting*/ Set the INTPn pins to be used.
   /* Set INTP0 to P137 */
     //none  No need to set
   /* Set INTP1 to P06 or P11*/
   /*P06 case, PIOR2 = 0*/
         //  PIOR &= 0xFBU  /* after RESET*/
     //PM0 |= 0x40U;
                  :

/*★INTP0*/ Activate the function setting of the INTPn pin to be used.
void R_INTC0_Start(void)   When using INTP0
{
    PIF0 = 0U;    /* clear INTP0 interrupt flag */
    PMK0 = 0U;    /* enable INTP0 interrupt */
}

void R_INTC0_Stop(void) When using INTP0
{
    PMK0 = 1U;    /* disable INTP0 interrupt */
    PIF0 = 0U;    /* clear INTP0 interrupt flag */
}
```

r_cg_intp_user.c

In CC-RL case

```
/*★INTPn Pragma*/ Activate the function setting of the INTPn pin to be used.
#pragma interrupt r_intc0_interrupt(vect=INTP0)
//#pragma interrupt r_intc2_interrupt(vect=INTP1)
//#pragma interrupt r_intc2_interrupt(vect=INTP2)
//#pragma interrupt r_intc3_interrupt(vect=INTP3)
//#pragma interrupt r_intc4_interrupt(vect=INTP4)
//#pragma interrupt r_intc5_interrupt(vect=INTP5)
                                    :

/*★*/ Activate the function setting of the INTPn pin to be used.
static void __near r_intc0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}
//static void __near r_intc1_interrupt(void)
//{
    /* Start user code. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here *///}
```

In AP4 for RL78 case

```
#pragma vector = INTP0_vect
__interrupt static void r_intc0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}

//#pragma vector = INTP1_vect
//__interrupt static void r_intc0_interrupt(void)
//{
    /* Start user code. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
//}
```

2.3.11 Key Interrupt Function

You can reuse the RL78/G10 CG/AP4 output, However, you need to change the pin settings for the KRn pins because the ports used for the KRn pins are different. In addition, if you want to use KR6 and KR7, you need to create a new code.

Table 2.6 Key Interrupt Function Difference between RL78/G10 (R5F10Y47) and RL78/G1M or G1N

KRn	RL78/G10	RL78/G1M, RL78/G1N
KR0	P40	P40(P00Note)
KR1	P125	P125
KR2	P01	P07
KR3	P02	P10
KR4	P03	P11
KR5	P04	P12
KR6	None	P13
KR7	None	P16

Note:RL78/G1N only
Caution: () is case of PIOR redirection set.

Sample codes(key)
- **r_cg_key.h** (Use sample code): To be used as is
- **r_cg_key.c** (Use sample code): To be used as is
- **r_cg_key_user.c** (G10 CG/AP4 output without modification): To be used as is.

Changes to r_cg_key.c

```
void R_KEY_Create(void)
{
    volatile uint8_t w_count;
 /* KRn setting */ Uncomment the port settings portion of the KRn pin to be used.
/* Set KR0 to P40 or P00 (RL78/G1N only)*/
   /*P40 case, PIOR3 = 0*/
        //  PIOR &= 0xF7U  /* after RESET*/
        //  PU4 |= 0x01U;
        //  PM4 |= 0x01U;
                  :
                  :
/*★Set INTKR low priority */ Change interrupt priority as needed.
    KRPR1 = 1U;
    KRPR0 = 1U;
                  :
                  :
/*★Detect KRn*/
 /* IF detect KRn  Change code */
/* KR0:_01_KR0_SIGNAL_DETECT_ON, KR0 detection setting
   KR1:_02_KR1_SIGNAL_DETECT_ON, KR1 detection setting
            :
            :                    */
When using KR1: Change the OFF setting to the ON setting.
    KRM0 = _00_KR0_SIGNAL_DETECT_OFF | _02_KR1_SIGNAL_DETECT_ON |
                  _00_KR2_SIGNAL_DETECT_OFF | _00_KR3_SIGNAL_DETECT_OFF |
                  _00_KR4_SIGNAL_DETECT_OFF | _00_KR5_SIGNAL_DETECT_OFF |
          _00_KR6_SIGNAL_DETECT_OFF | _00_KR7_SIGNAL_DETECT_OFF;
```

2.4 Modify Initial Setting Functions

Activate the Initial functions for each function set in "2.3 Modify Functions to be Used"

Change the PIOR setting of the R_Systeminit() function in the sample code,if necessary.

Activate the functions used in the R_Systeminit() function.

Activate the include file for the enabled functions.

Sample codes
r_cg_systeminit.c (Use sample code)

Changes to - r_cg_systeminit.c

```
/*★Activate the required include file */Activated the port function
#include "r_cg_port.h"
//#include "r_cg_cgc.h"
//#include "r_cg_tau.h"
          :
```

For RL78/G1M

```
/*★Set PIOR                Setting Value */
/*                          |   0 |   1 | */
/*----------------------------------- */
/* PIOR7 note1  SO00/TxD0      | P06 | P10 | */
/*              SI00/RxD0      | P07 | P15 | */
/*              SCK00          | P10 | P16 | */
/* PIOR6 note2                      0      */
/* PIOR5 note2                      0      */
/* PIOR4 note2                      0      */
/* PIOR3                      0      */
/* PIOR2        INTP1         | P06 | P11 | */
/* PIOR1        TI01/TO01           | P12 | P40 | */
/* PIOR0        PCLBUZ0             | P10 | P40 | */
                    :
```

For RL78/G1N

```
/*★Set PIOR            Setting Value */
/*                          |   0 |   1 | */
/*----------------------------------- */
/* PIOR7 note1                0      */
/* PIOR6 note2  TxD0               | P06 | P16 | */
/*              RxD0           | P07 | P137| */
/* PIOR5 note2  SO00/TxD0      | P06 | P01 | */
/*              SI00/RxD0      | P07 | P137| */
/*              SCK00          | P10 | P16 | */
/* PIOR4 note2  SO00/TxD0      | P06 | P01 | */
/*              SI00/RxD0      | P07 | P137| */
/*              SCK00          | P10 | P00 | */
/* PIOR3 note2  KR0                | P40 | P00 | */
/* PIOR2        INTP1          | P06 | P11 | */
/* PIOR1        TI01/TO01           | P12 | P40 | */
/* PIOR0        PCLBUZ0             | P10 | P40 | */
 PIOR = 0x00U; //After reset value     Make changes as needed.
                    :
/*★Activate the required functions*/
     R_PORT_Create(); Activate the port function.
 //  R_CGC_Get_ResetSource();
 //  R_CGC_Create();
 //  R_TAU0_Create();
```

2.5 Create the main function

Uncomment the include files for the functions with the required functionality and write the user program.

Sample codes
r_cg_main.c (Use sample code):To be used as is

Changes to - r_cg_main.c

```
        :
#include "r_cg_macrodriver.h"
/*★Activate the required include file*/
//#include "r_cg_cgc.h"
//#include "r_cg_tau.h"

#include "r_cg_intp.h" Uncomment the include files for the functions to be used.
```

2.6 Option Byte Setting

For CS+ and e2studio IDEs, set the option byte in the link option.
In the case of IAR, the Option byte is output in the r_cg_main.c file of the AP4 code output from AP4 for RL78, so you need to change it in the source code.

```
/* Set option bytes */
#pragma location = "OPTBYTE"
__root const uint8_t opbyte0 = 0xEEU;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte1 = 0xF7U;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte2 = 0xF9U;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte3 = 0x85U;
```

2.7 Precautions When Creating User Programs

The start and stop routines for each functionality are CG output assuming that interrupt routines are used. If you do not use interrupt routines (for example, in the case of AD converters or other devices), use CG output code without setting interrupt use, or delete or comment out the codes related to INT.

Changes to sample code

```
void R_ADC_Start(void)
{
    ADIF = 0U;  /* clear INTAD interrupt flag */
//   ADMK = 0U;  /* enable INTAD interrupt */ Change to disable
   ADCS = 1U;  /* enable AD conversion */
}
```

3. Application Development Example

Using the sample code in the G1M folder, develop an application that sends the A/D conversion result of the ANI0 pin voltage to UART every 100ms.

3.1 Conditions for Operation Confirmation

The sample code with this application note runs properly under the condition below.

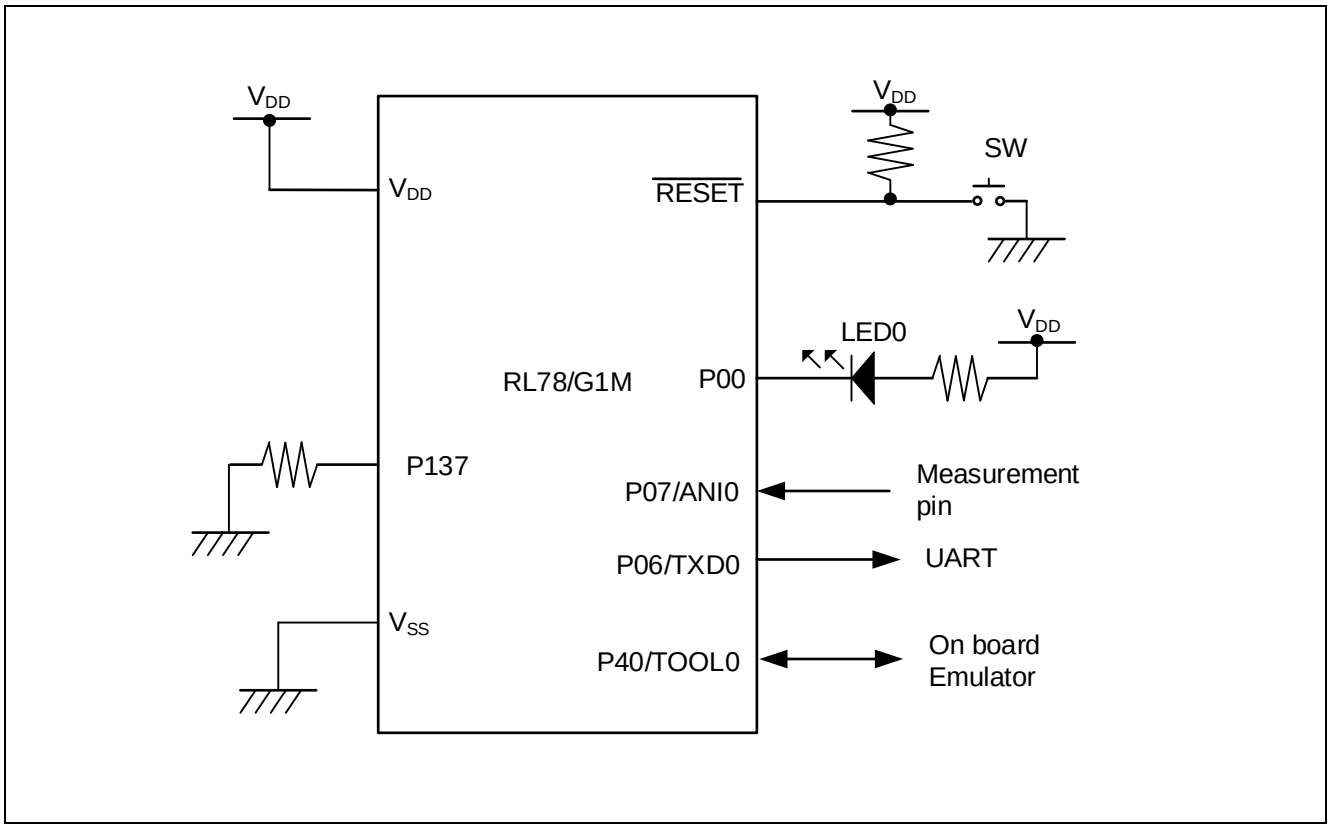
表 3.1 Operation Confirmation Conditions

Items	Contents
MCU	RL78/G1M（R5F11W68）
Bord	RL78/G1M Fast Prototyping Board (RTK5RLG230CLG000BJ)
Operating frequencies	• High-speed on-chip oscillator clock: 20 MHz • CPU/peripheral hardware clock: 20 MHz
Operating voltage	3.3V (can be operated at 3.02V～5.5V) SPOR operations: Reset mode Rising edge TYP. 2.90 V (2.76V ～3.02 V) Falling edge TYP. 2.86 V (2.70 V ～ 2.96 V)
Integrated development environment (CS+)	CS+ V8.07.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.11.00 from Renesas Electronics Corp.
Integrated development environment (e2 studio)	e2 studio 2022-01 (22.01.0) from Renesas Electronics Corp.
C compiler (e2 studio)	CC-RL V1.11 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.2 from IAR Systems
C compiler (IAR)	

3.2 Hardware Configuration Example

Figure 3.1 shows an example of hardware configuration that is used for this application note.

Figure 3.1 Hardware Configuration



Note 1: This circuit diagram is simplified to show an overview of the connections. When actually creating a circuit, design the circuit to meet the electrical characteristics by properly handling the pins, etc. (Input-only ports should be individually connected to VDD or VSS via resistors.)

2: VDD must be equal to or greater than the reset release voltage (VSPOR) set by SPOR.

3.3 List of Pins Used

Table 3.2 shows Pins Used and Their Functionalities.

Table 3.2 Pins Used and Their Functionalities

Pin name	Input/Output	Function
P06/SO00/TxD0/INTP1/RTIO06	Output	UART data output pin
P07/ANI0/SI00/RXD0/KR2/RTIO07	Input	A/D converter analog input pin

3.4 Software Description

3.4.1 Option Bytes

Table 3.3 shows the option byte settings.

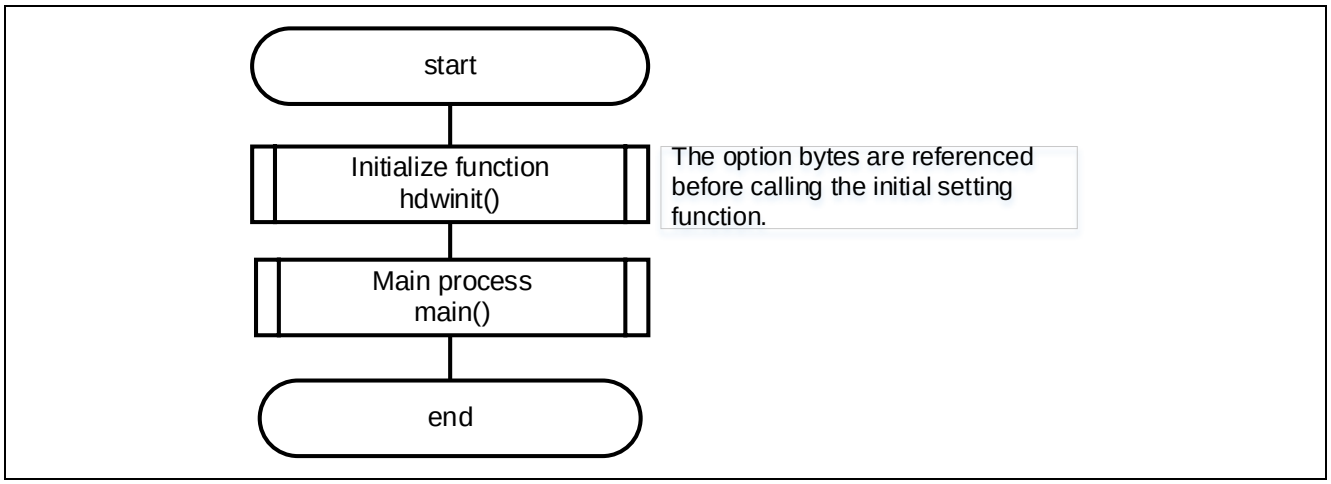
Table 3.3 Option Byte Settings

Address	Setting Value	Contents
000C0H	11101110B	Operation of Watchdog timer is stopped (Counting is stopped after reset)
000C1H	11111011B	SPOR operating mode：reset mode Detection voltage Rising edge TYP. 2.90 V (1.84 V ～ 1.95 V) Falling edge TYP. 2.84 V (1.80 V ～ 1.91 V)
000C2H	11111001B	Flash operating mode: HS mode High-speed on-chip oscillator clock: 20MHz
000C3H	10000101B	On-chip debugging is enabled

3.4.2 Flowchart

Figure 3.1 shows the overall flow.

Figure 3.1 Overall Flow



Note: The startup routine is executed before and after the initial setting function.

Figure 3.2 shows the initial setting flow.

Figure 3.2 Initial Setting Function

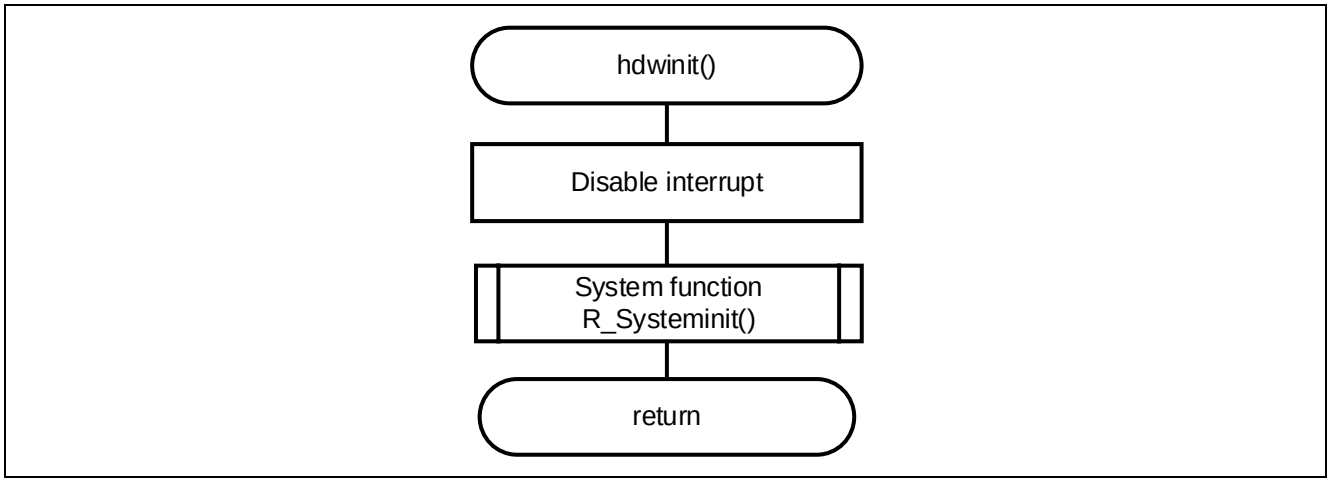


Figure 3.3 shows the system function flow.

Figure 3.3 System Function

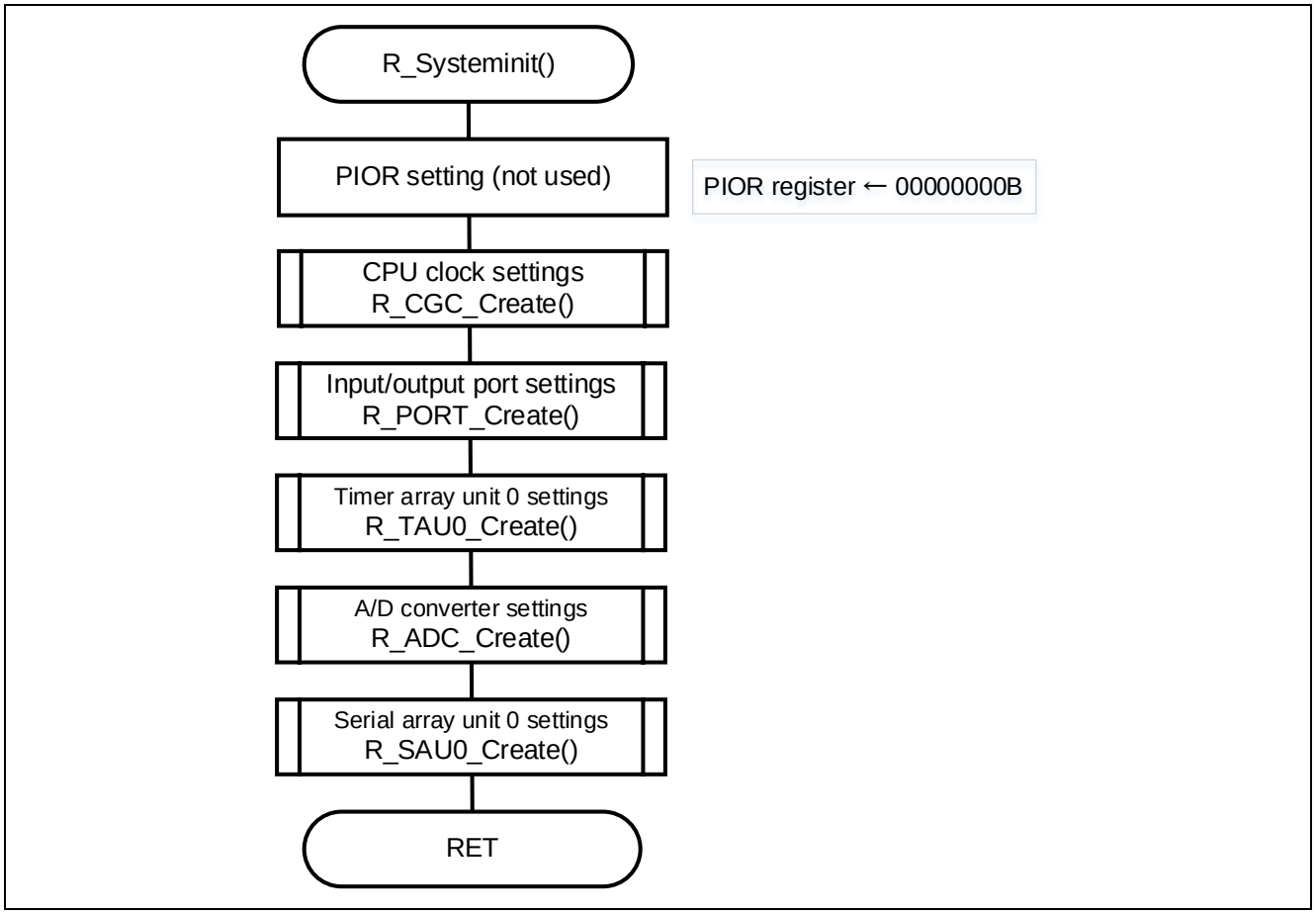


Figure 3.4 shows the main function flow.

Figure 3.4 Main function

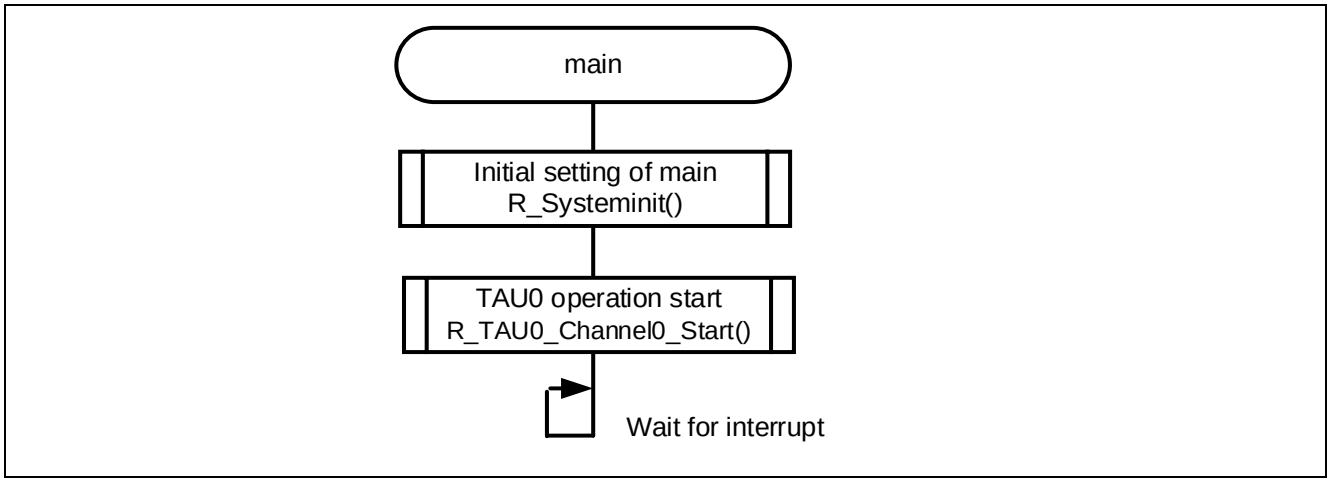
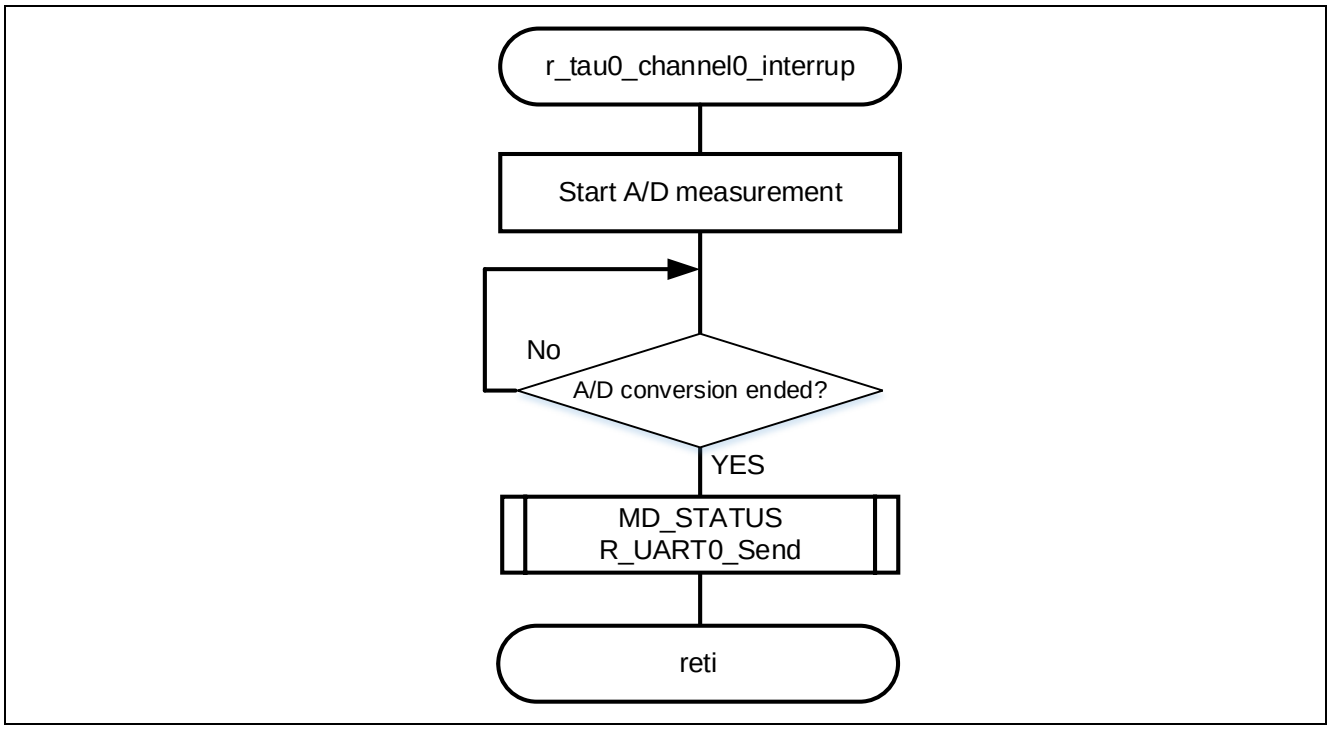


Figure 3.5 shows the flow of the user function.

Figure 3.5 User Function



3.5 Development Procedure

Develop sample source code based on the development procedures described in Chapter 2.
First, copy the G1M_sample project and give it an arbitrary project name (e.g., r01an5984).

Next, create each code based on the flowcharts shown in 3.5.2. The following sections describe an example development procedure.
User functions are not described here.

3.5.1 Developing the initial setting functions (r_cg_systeminit.c)

Develop the initial setting functions used in the initial setting flow in Figure 3.3.
Perform the following steps (i) to (iii).

(i) Uncomment the necessary include files in r_cg_systeminit.c.
Specifically, activate files related to port settings, clock settings, timers, A/D converter, and serial communication settings.

```
#include "r_cg_macrodriver.h"
/*★Activate the required include file */
#include "r_cg_port.h"
#include "r_cg_cgc.h"
#include "r_cg_tau.h"
//#include "r_cg_it.h"
//#include "r_cg_pclbuz.h"
//#include "r_cg_wdt.h"
#include "r_cg_adc.h"
#include "r_cg_sau.h"
//#include "r_cg_intp.h"
//#include "r_cg_rto.h"
//#include "r_cg_key.h"
```

(ii) Next, set the PIOR value in the void R_Systeminit(void) function.

```
PIOR = 0x00U; //After reset value
```

(iii) Finally, uncomment the necessary initial setting function.

```
/*★Activate the required functions*/
R_PORT_Create();
//  R_CGC_Get_ResetSource();
R_CGC_Create();
R_TAU0_Create();
//  R_PCLBUZ0_Create();
//  R_WDT_Create();
R_ADC_Create();
R_SAU0_Create();
//  R_INTC_Create();
//  R_KEY_Create();
//  R_RTO_Create();
//  R_IT_Create();
```

3.5.2 Developing System Functions

Next, develop system functions. Develop the ***_create functions used in Figure 3.4.

(i) Developing R_PORT_Create()
PORT_Create() is a function in r_cg_port.c. To control LEDs, set the P00 pin to the digital output port. And set P0 to "1" to turn off the LED initially.

```
P0 = 0x01;
PM0 = 0xfe;
```

(ii) Developing R_CGC_Create()
R_CGC_Create() is a function in r_cg_cgc.c.
Because fIL is not used, change to “fIL Stop”.

```
// OSMC = _10_CGC_IT_CLK_FIL;
 OSMC = _00_CGC_IT_CLK_NO; //fIL Stop
```

(iii) Developing R_TAU0_Create()
Since it is used as an interval timer, no pin setting is required; the output code of CG/AP4 for RL78/G10 can be used without modification; the CG/AP4 for RL78/G10 is set to generate INTTM00 interrupt every 100ms for code output. Replace r_cg_tau.c, r_cg_tau.h, and r_cg_tau_user.c in the project under development (AD_UART) with the CG/AP4 for RL78/G10 output codes r_ch_tau.c, r_cg_tau.h, and r_cg_tau_user.c

(iv) Developing R_ADC_Create()
R_ADC_Create() is a function in r_cg_adc.c. It uses ANI0, 8-bit mode, and sets the conversion clock (fCLK/8).

```
/* ★Set ANI pin select*/
   /*ANI0 case set P07*/
  PMC0 |= 0x80U;
  PM0  |= 0x80U;
```

Do not change the setting because the conversion clock (fCLK/8) is used.

```
/* ★AD converter mode register 0 (ADM0)    */
 ADM0 = _00_AD_CONVERSION_CLOCK_8 | _00_AD_TIME_MODE_NORMAL_1;/*Ctime9.2us */
```

Change the AD resolution to 8 bits.

```
/*_00_AD_RESOLUTION_10BIT (0x00U) | 10 bits */
/*_01_AD_RESOLUTION_8BIT  (0x01U) | 8 bits */
ADM2 = _01_AD_RESOLUTION_8BIT;

/* ★ Select ADI Channel */
 ADS = _00_AD_INPUT_CHANNEL_0;

/*★ AD comparator ADCE=1:enable ADCE=0:disable   */
ADCE = 1U;
```

(v) Developing R_SAU_Create()

R_SAU_Create() is a function in r_cg_sau.c.

This function is used with the following configuration: transmission mode setting is single shot mode, data bit length is 8 bits, data transfer direction is LSB, stop bit length is 1, transmit data level setting is standard, baud rate is 9600 bps.

r_cg_sau.c

```
void R_SAU0_Create(void)
{

/* ★SAU0 Clock setting*/
    SPS0 = _04_SAU_CK00_FCLK_4 | _40_SAU_CK01_FCLK_4;


void R_UART0_Create(void)
{

/* ★UART Function */
/* ★IF UART setting change, Copy CG output code for RL78/G10 to this area */
     ST0 |= _01_SAU_CH0_STOP_TRG_ON;    /* UART0 transmit disable */
    STMK0 = 1U;    /* disable INTST0 interrupt */
    STIF0 = 0U;    /* clear INTST0 interrupt flag */
    SRMK0 = 1U;    /* disable INTSR0 interrupt */
    SRIF0 = 0U;    /* clear INTSR0 interrupt flag */
    SREMK0 = 1U;   /* disable INTSRE0 interrupt */
    SREIF0 = 0U;   /* clear INTSRE0 interrupt flag */
    /* Set INTST0 low priority */
    STPR10 = 1U;
    STPR00 = 1U;
    SMR00L = _20_SAU_SMRMN_INITIALVALUE | _02_SAU_MODE_UART | _00_SAU_TRANSFER_END;
    SMR00H = _00_SAU_CLOCK_SELECT_CK00 | _00_SAU_TRIGGER_SOFTWARE;
    SCR00L = _80_SAU_LSB | _10_SAU_STOP_1 | _07_SAU_LENGTH_8;
    SCR00H = _80_SAU_TRANSMISSION | _00_SAU_INTSRE_MASK | _00_SAU_PARITY_NONE;
    SDR00H = _80_UART0_TRANSMIT_DIVISOR;
    SO0 |= _01_SAU_CH0_DATA_OUTPUT_1;
    SOL0 |= _00_SAU_CHANNEL0_NORMAL;    /* output level normal */
     SOE0 |= _01_SAU_CH0_OUTPUT_ENABLE;    /* enable UART0 output */
   /* ★UART Function G10 code copy end*/

     /* ★RxD0 & TxD0 pin setting*/ Since RxD0 is not used, only TxD0 port setting is required.
    /*P07/RxD0 & P06/TxD0 case for RL78/G1M & /G1N*/

     PIOR &= 0x7FU;
    //  PMC0 &= 0x7FU;
    //  PM0 |= 0x80U;
      P0 |= 0x40U;
       PM0 &= 0xBFU;
```

3.5.3 Developing the main Function (r_cg_main.c)

Develop the main function. Write a code for the process shown in Figure 3.5.

(i) Uncomment the necessary include files.

```
/*★Activate the required include file*/
#include "r_cg_cgc.h"
#include "r_cg_tau.h"
//#include "r_cg_it.h"
//#include "r_cg_wdt.h"
#include "r_cg_adc.h"
#include "r_cg_sau.h"
```

(ii) Next, develop a main program.

```
r_cg_main.c
 void main(void)
{
  R_MAIN_UserInit();
 /* Start user code. Do not edit comment generated here */
  R_TAU0_Channel0_Start();/*Interval Timer Start*/
 while (1U)
 {
;
 }
 /* End user code. Do not edit comment generated here */
}
```

(iii)Finally, the user program is developed.
グローバル変数の記載とユーザ・メインプログラムを記載します。

r_cg_tau_user.c.c

```
 /* Start user code for global. Do not edit comment generated here */
 uint8_t AD_DATA; /* AD Value */
 MD_STATUS g_uart0_tx_end = 0U;
 /* End user code. Do not edit comment generated here */
                                  :
static void __near r_tau0_channel0_interrupt(void)
{
  /* Start user code. Do not edit comment generated here */
 R_ADC_Start();
 R_UART0_Start();
 while(ADIF!=1){}
 AD_DATA = ADCRH;
 ADIF=0U;
 g_uart0_tx_end = R_UART0_Send(&AD_DATA, 1U);
  /* End user code. Do not edit comment generated here */
}
```

3.5.4 Build

So far, you have made changes to the code of the necessary files. Finally, remove unnecessary files from the project to complete the development.

Review the option bytes to be set from [Build Tool] > [Link Options] > [Device], and then perform build operation.

Additional settings when using IAR's compiler

Set the option bytes on the program source.

```
/* Set option bytes */
#pragma location = "OPTBYTE"
__root const uint8_t opbyte0 = 0xEEU;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte1 = 0xF7U;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte2 = 0xF9U;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte3 = 0x85U;
```

・Modification of macrodriver.h

The macrodriver.h output from AP4 includes the device file for G10. Change it for the target device. The following is the case of G1M (8K ROM product).

Also, include files for each device for IAR are located under the IAR installation folder

Please copy and use them from IAR Systems\Embedded Workbench 8.5\rl78\inc.

```
/***********************************************************************************************************************
Includes
***********************************************************************************************************************/
#include "ior5f11w68.h"
#include "ior5f11w68_ext.h"
#include "intrinsics.h"
```

4. Sample codes

We provide the following two types of projects and sample codes:

Source for G1M and G1N folders: Source code developed from G10CG/AP4 output for G1M and G1N
AD_UART: A set of sample projects developed from the sources in the G1M folder according to the procedures in Chapter 3.

Obtain the sample codes from the Renesas Electronics website.

5. Reference

RL78/G1M,G1N User's Manual: Hardware (R01UH0904E)

RL78/G10 User's Manual: Hardware (R01UH0384E)

RL78 Family User's Manual: Software（R01US0015E）

Code Generator User's Manual: RL78 API Reference(R20UT4323E)

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

Technical Update / Technical News

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

Website and Support

Renesas Electronics Website

http://www.renesas.com

Inquires

http://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2022.5.31		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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.