RL78/G1M, RL78/G1N

RL78/G10 のコード生成ツールを使用したソフトウェア開発

要旨

本アプリケーションノートでは, RL78/G10 用コード生成ツールを使用した RL78/G1M または RL78G1N 向けプログラム開発方法を説明します。

コード生成ツールとは, C コンパイラ CC-RL 使用時の RL78 コード生成プラグイン (CG) または IAR 社 C コンパイラ使用時の“AP4 for RL78 (AP4)”を指します。

動作確認デバイス

RL78/G1M, RL78/G1N

目次

1.　仕様

1.1　仕様概要

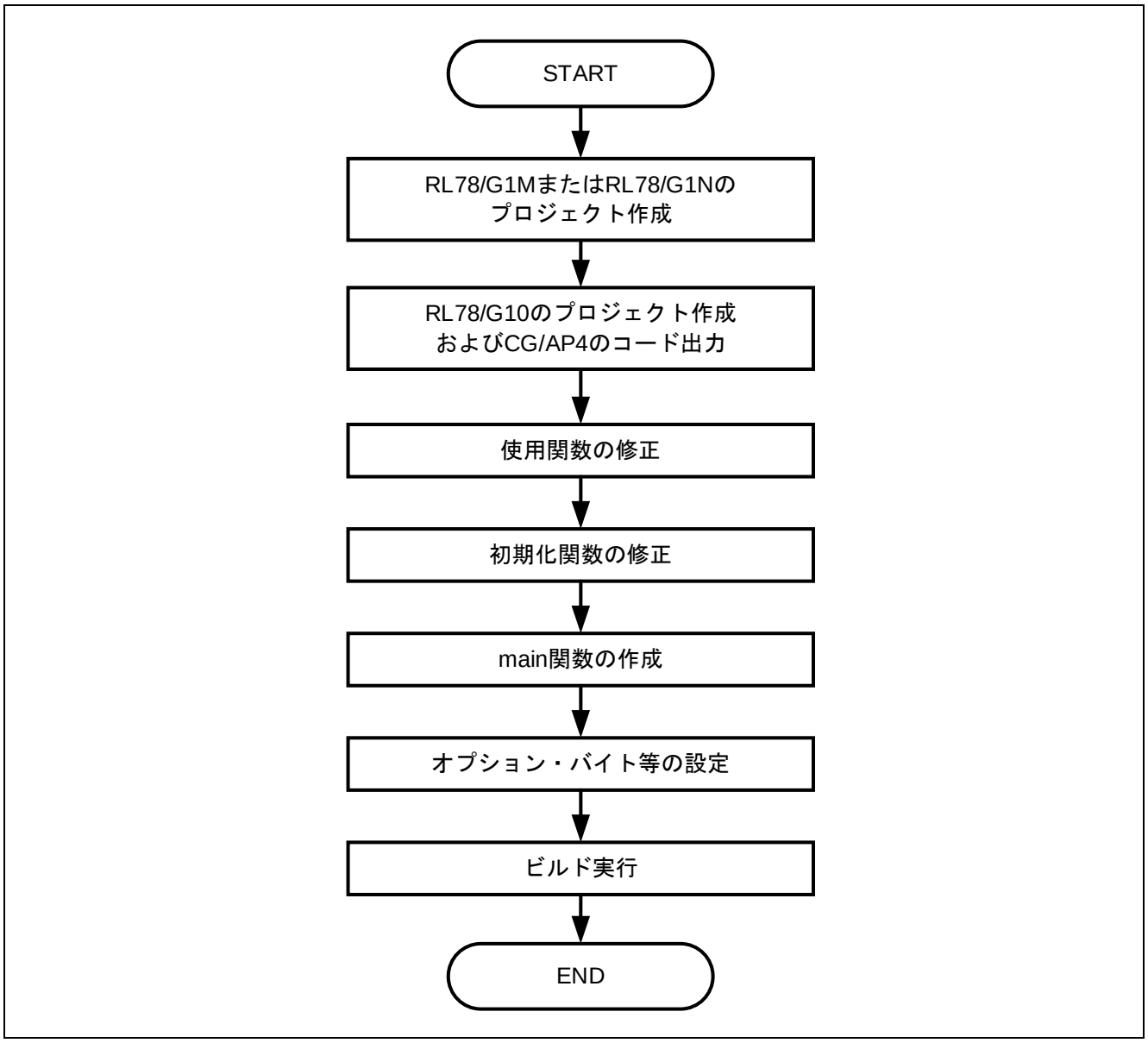
本アプリケーションノートでは, RL78/G10 のコード生成ツール CG または AP4 を使って RL78/G1M または RL78/G1N 向けプログラムを開発する方法を説明します。

RL78/G1M および RL78/G1N では, CG または AP4 はサポートされていません。しかし, RL78/G10 用 CG または AP4 で出力された関数を流用することで，簡単にプログラムを開発することができます。

1.2　開発手順概要

図 1.1 に RL78/G10 用 CG/AP4 を使用した RL78/G1M または RL78/G1N プログラムの開発フローを示します。

図 1.1　　RL78/G10 用 CG/AP4 を使用したプログラム開発フロー



2.　開発手順の詳細

2.1　RL78/G1M, RL78/G1N のプロジェクト作成

CC-RL の場合，統合開発環境を起動して RL78/G1M または RL78/G1N のプロジェクトを作成します。新規プロジェクトを作成すると，プロジェクトを生成したフォルダ内に cg_src フォルダが生成されます。cg_src フォルダには, cstart.asm, hdwinit.asm, main.c, iodefine.h が出力されています。

IAR 社コンパイラを使用する場合, IAR Embedded Workbench を起動して RL78/G1M または RL78/G1N のプロジェクトを生成します。プロジェクトの生成は，「プロジェクト」のメニューから「新規プロジェクトの作成」を選択します。つぎに，「プロジェクトテンプレート」から「C」を選択して「OK」ボタンを押下します。「名前をつけて保存」画面で「ファイル名」にプロジェクト名を入力し，「保存」ボタンを押下します。保存ボタンを押すと, main.c がプロジェクトに登録されます。さらに, IAR がインストールされているディレクトリ[注]内の ior5f11w68.h および ior5f11w68_ext.h をプロジェクトに登録します。

注. 「C:¥Program Files (x86)¥IAR Systems¥Embedded Workbench 8.5¥rl78¥inc」など

2.2　RL78/G10 のプロジェクト作成および CG/AP4 のコード出力

RL78/G1M または RL78/G1N で使用する機能を RL78/G10 用 CG/AP4 でコード出力します。CG/AP4 が出力したファイルを「2.1 RL78/G1M, RL78/G1N のプロジェクト作成」で作成したプロジェクトに追加登録します。

本サンプルコードでは，プロジェクトに追加登録するファイルを下記のフォルダに保存しています。

CG の場合:
..¥workspace¥CS+/ e2studio¥G1x¥cg_src¥
..¥workspace¥e2studio¥G1x¥cg_src¥

AP4 の場合:
..¥workspace¥IAR¥G1x¥cg_src¥

注意．G1x = G1M or G1N

2.3　使用関数の変更

次に「2.2 RL78/G10 のプロジェクト作成および CG/AP4 のコード出力」で登録したファイルの内容を変更します。変更箇所は, RL78/G10 と RL78/G1M または RL78/G1N で機能差がある箇所です。表 2.1 に機能差分を示します。

ここでは, RL78/G10 (R5F10Y47: 16 ピン, ROM 4KB) の CG/AP4 出力ファイルを使用しています。

表 2.1　RL78/G10 (R5F10Y47) と RL78/G1M または RL78/G1N の機能差分

機能	RL78/G10 との 機能差分	RL78/G10 用 CG/AP4 利用	本資料の 説明箇所
CPU コア	なし	利用しない	―
ポート機能	あり	利用しない	2.3.1
クロック発生回路	あり	利用 (変更必要)	2.3.2
タイマ･アレイ･ユニット	なし	利用 (変更が必要な場合あり)	2.3.3
12 ビット･インターバル･タイマ	なし	利用 (変更なし)	2.3.4
クロック出力/ブザー出力制御回路	あり	利用 (変更必要)	2.3.5
ウォッチドッグ･タイマ	なし	利用 (変更なし)	2.3.6
A/D コンバータ	あり	利用 (変更必要)	2.3.7
シリアル･アレイ･ユニット	あり	利用 (変更必要)	2.3.8
リアルタイム出力制御回路(RL78/G1M のみ)	あり (RL78/G10 非搭載)	使用しない	2.3.9
割り込み機能	あり	利用 (変更必要)	2.3.10
キー割り込み機能	あり	利用 (変更必要)	2.3.11
スタンバイ機能	なし	利用しない	―
リセット機能	なし	利用しない	―
セレクタブル・パワーオン･リセット機能	なし	利用しない (オプション・バイトで設定)	―
オプション・バイト	なし	利用しない[注] (IDE で設定)	―
オンチップ・デバッグ機能	なし	利用しない[注] (IDE で設定)	―
10 進補正回路	なし	利用 (変更不要)	―

―:　説明箇所なし
IDE:　Integrated Development Environment
注.　AP4 の場合, r_cg_main.c で設定します。
備考. 本サンプルコードの workspace フォルダにある CS+/e2studio/IAR フォルダ以下の G1M または G1N フォルダ内に RL78/G1M または RL78/G1N のサンプルコードを用意しています。

2.3.1　ポート機能

RL78/G10 用 CG/AP4 は使用せず, 下記サンプルコード (port) を利用します。サンプルコード (port)ではポートのリセット値を設定しています。シリアル通信など兼用機能を使用する場合, 兼用機能の Create 関数で再びポート設定が行われます。

サンプルコード (port) では, コメントにレジスタ説明を記述しています。ポート設定を変更する場合は, 使用する RL78 製品のハードウエアマニュアルを参照してください。

サンプルコード (port)

・r_cg_port.h: 関数宣言のみ。変更せずに使用する。
・r_cg_port.c: 必要に応じてポートの入出力設定を変更する。

サンプルコード変更例: P00 を出力ポートとし，ハイ・レベルを出力する。(赤字箇所)

```
void R_PORT_Create(void)
{
/* Port register*/
/* Set output latch value of each 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   ※0x00 から変更する。
   P1 = 0x00; //After Reset Value
   P4 = 0x00; //After Reset Value

/*Port mode registers*/
/* "1": input mode    "0": output mode */
/* 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    ※0xff から変更する。
   PM1 = 0xff;    //After Reset Value
   PM4 = 0xff;    //After Reset Value
              :
}
```

2.3.2　クロック発生回路

RL78/G10 用 CG/AP4 を利用できます。RL78/G1M または RL78/G1N は X1 発振回路がないため, 12 ビット･インターバル・タイマの動作クロック供給の部分のみを利用します。

サンプルコード (CGC)

・**r_cg_cgc.h**:　　サンプルコードの r_cg_cgc.h を使用する。
・**r_cg_cgc.c**:　　サンプルコードの r_cg_cgc.c を使用する。
・**r_cg_cgc_user.c**: RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

サンプルコードの変更例: 12 ビット･インターバル・タイマの動作クロック供給

```
void R_CGC_Create(void)
{

/*   ★   OSMC selection    "_10_CGC_IT_CLK_FIL" or "_00_CGC_IT_CLK_NO" */
      OSMC = _10_CGC_IT_CLK_FIL;    ※fILを供給する。
      //OSMC = _00_CGC_IT_CLK_NO;  ※fIL を供給しない。
}
```

2.3.3　タイマ・アレイ・ユニット

RL78/G10 用 CG/AP4 を利用できます。ただし，タイマ入出力端子の兼用ポートが異なる場合は，端子設定を変更する必要があります。インターバル・タイマなどポート入出力を使用しない場合は，端子設定の変更は必要ありません。

また，タイマ・アレイ・ユニットの動作クロックは, CPU／周辺ハードウエア・クロック周波数 (fCLK) の分周になります。RL78/G10 用 CG/AP4 を利用する場合は, RL78/G1M または RL78/G1N で使用する fCLK と同じ設定にする必要があります。fCLK は，ユーザ・オプション・バイトおよび高速オンチップ・オシレータ周波数選択レジスタ (HOCODIV) で設定します。

表 2.2　TO0x 端子の兼用ポート

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

() :　PIOR リダイレクション設定時

サンプルコード (tau): fCLK = 20MHz 時の周期 100us の方形波出力
・**r_cg_tau.h**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
・**r_cg_tau.c**:　　RL78/G10 用 CG/AP4 出力から端子設定を変更する。
・**r_cg_tau_user.c**:　RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

サンプルコードの変更例

```
void R_TAU0_Create(void)
{
  /* ★TAU setting: Copy CG output code for RL78/G10 to this area */  ※変更せずに使用する。
     TAU0EN = 1U;       /* Enables input clock supply. */
     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 */   ※TO00を使用する。
           P1    &= 0xFDU;/*Clear bit 2 */   ※TO00を使用する。
           PM1   &= 0xFDU;/*Clear bit 2 */   ※TO00を使用する。

     /*★ TI02 pin setting */
     /*TI02 = P16 case*/
     //      PMC1 &= 0xBFU; /*Clear bit 6 */   ※TI02を使用しない場合，コメントアウトする。
     //      PM1 |= 0x40U; /*Set bit 6 */   ※TI02 を使用しない場合，コメントアウトする。
```

2.3.4　12 ビット･インターバル・タイマ

RL78/G10 用 CG/AP4 の出力コードを変更せずに使用できます。

サンプルコード (it): 100ms 毎に割り込み要求信号を発生させるインターバル・タイマ
- **r_cg_it.h**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
- **r_cg_it.c**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
- **r_cg_it_user.c**:　RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

2.3.5　クロック出力/ブザー出力制御回路

RL78/G10 用 CG/AP4 を利用できます。ただし，兼用ポートが異なるため，端子設定を変更する必要があります。兼用ポートは, PIOR 設定 (r_cg_systemin.c) によっても変更されます。

サンプルコード (pclbuz): fMAIN/ (2^4)を選択。fCLK = 20MHz 時の 1.25 MHz の出力クロック。
- **r_cg_pclbuz.h**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
- **r_cg_pclbuz.c**:　　サンプルコードの r_cg_pclbuz.c を使用する。必要に応じて設定を変更する。
- **r_cg_pclbuz_user.c**:　RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

サンプルコード変更例:

```
void R_PCLBUZ0_Create(void)
{
    PCLOE0 = 0U; /* disable PCLBUZ0 operation */
/* ★PCLBUZ0 output clock selection (CCS02 - CCS00) */   ※使用する設定を選択します。
//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*/※P10出力する場合は, P40出力を無効にし，つぎの設定を有効にします。
/*P10 output case*/
//    PIOR &= 0xFEU;    /* after RESET*/
    PMC1 &= 0xFEU;
    POM1 &= 0xFEU;
    P1 &= 0xFEU;
    PM1 &= 0xFEU;

/*P40 output case*/  ※P40出力する場合は, P10出力を無効にし，つぎの設定を有効にします。
//    PIOR |= 0x01U;
//    P4 &= 0xFEU;
//    PM4 &= 0xFEU;
```

2.3.6　ウォッチドッグ･タイマ

RL78/G10 用 CG/AP4 の出力コードを変更せずに使用できます。
ただし，オプション・バイトで設定が必要です。サンプルコード(wdt)ではオーバーフロー時間を最大値に設定しています。

サンプルコード(wdt)

・**r_cg_wdt.h**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
・**r_cg_wdt.c**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
・**r_cg_wdt_user.c**: RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

2.3.7　A/D コンバータ

RL78/G10 用 CG/AP4 を利用できます。ただし，アナログ入力端子の兼用ポートが異なるため，端子設定を変更する必要があります。

表 2.3　ANIx 端子の兼用ポート

ANIx	ADS 設定値	RL78/G10	RL78/G1M, RL78/G1N
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

―:　兼用ポートなし

サンプルコード(adc)

・**r_cg_adc.h**:　　サンプルコードの r_cg_adc.h を利用する。
・**r_cg_adc.c**:　　サンプルコードの r_cg_adc.c を利用する。必要に応じて設定を変更する。
・**r_cg_adc_user.c**: RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

サンプルコード変更例

```
void R_ADC_Create(void)
{
    ADCEN = 1U;    /* Enables input clock supply.*/
    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;
  /* ★ANI pin selection*/  ※使用するアナログ入力端子を有効にします。
  /*ANI0/P07 case*/
      PMC0 |= 0x80U;
      PM0  |= 0x80U;
   /*ANI1/P10 case*/
      // PMC1 |= 0x01U;
      // PM1  |= 0x01U;
   /*ANI2/P11 case*/
      // PMC1 |= 0x02U;
      // PM1  |= 0x02U;
   /*ANI3/P12 case*/
      // PMC1 |= 0x04U;
      // PM1  |= 0x04U;
```

```
    /*ANI4/P13 case*/
      // PMC1 |= 0x08U;
      // PM1   |= 0x08U;
    /*ANI5/P14 case*/
      // PMC1 |= 0x10U;
      // PM1   |= 0x10U;
    /*ANI6/P15 case*/
      // PMC1 |= 0x20U;
      // PM1   |= 0x20U;
    /*ANI7/P16 case*/
      // PMC1 |= 0x40U;
      // PM1   |= 0x40U;


  /* ★AD converter mode register 0 (ADM0) */   ※動作モードの設定を選択します。
  /*_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 8 bits*/
  /*_02_AD_TIME_MODE_NORMAL_2            (0x02U) | normal 2 mode 17(15)fAD ():when 10 bits*/
  ADM0 = _00_AD_CONVERSION_CLOCK_8 | _00_AD_TIME_MODE_NORMAL_1;/*Conv time 9.2us */

  /* ★AD resolution selection (ADTYP) */   ※どちらかを選択します。
  /*_00_AD_RESOLUTION_10BIT (0x00U) | 10 bits */
  /*_01_AD_RESOLUTION_8BIT    (0x01U) | 8 bits */
  ADM2 = _00_AD_RESOLUTION_10BIT;

  /* ★ADI Channel selection*/ ※使用するアナログ入力ポートを選択します。
  /*_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 = 1/0 (enable/disable) */※A/D変換待機状態にします。
    ADCE = 1U;
}
```

2.3.8　シリアル・アレイ・ユニット

RL78/G10 用 CG/AP4 を利用できます。ただし，シリアル入出力端子の兼用ポートが異なるため，端子設定を変更する必要があります。兼用ポートは, PIOR 設定 (r_cg_systemint.c) によっても変更されます。

また, RL78/G10 には, RL78/G1M および RL78/G1N に搭載されていない CSI01 と IIC00 が搭載されています。

表 2.4　シリアル入出力端子の兼用ポート

機能		RL78 /G10	RL78/G1M		RL78/G1N			
			PIOR7=0	PIOR7=1	PIOR6=0 PIOR5=0 PIOR4=0	PIOR6=0 PIOR5=0 PIOR4=1	PIOR4=0 PIOR5=1 PIOR6=0	PIOR6=1 PIOR5=0 PIOR4=0
CSI00	SO00	P00	P06	P10	P06	P01	P01	P16
	SI00	P01	P07	P15	P07	P137	P137	P137
	SCK00	P02	P10	P16	P10	P00	P16	P10
UART0	TxD0	P00	P06	P10	P06	P01	P01	P16
	RxD0	P01	P07	P15	P07	P137	P137	P137

注意 1. PIOR0 = 1, PIOR1 = 1 の同時設定は禁止です。
2. PIOR3 = 1, PIOR4 = 1 の同時設定は禁止です。
3. PIOR4, PIOR5, PIOR6 のうち，複数を 1 に設定することは禁止です。

サンプルコード(sau)は, UART 用と CSI 用があります。

サンプルコード(sau)
UART 使用時: ¥G1M¥cg_src¥SAU¥UART フォルダ内
・**r_cg_sau.h**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
・**r_cg_sau.c**:　　RL78/G10 用 CG/AP4 出力から端子設定を変更する。
・**r_cg_sau_user.c**: RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

CSI 使用時: ¥G1M¥cg_src¥SAU¥CSI フォルダ内
・**r_cg_sau.h**:　　RL78/G10 用 CG/AP4 出力を変更せずに使用する。
・**r_cg_sau.c**:　　RL78/G10 用 CG/AP4 出力から端子設定を変更する。
・**r_cg_sau_user.c**: RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

サンプルコード変更例: P07, P06 をそれぞれ RxD0, TxD0 として使用する。

```
void R_SAU0_Create(void)
{
    SAU0EN = 1U;       /* Enables input clock supply. */
    NOP();
    NOP();
    NOP();
    NOP();
 /* ★SAU0 Clock setting*/     ※変更せずに使用する。
    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, Copy CG output code for RL78/G10 to this area */ ※変更せずに使用する。
       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*/
                                                                ※UART 端子で使用する端子を選択します。
```

RL78/G1M の場合

```
      /*P07/RxD0 & P06/TxD0 case */
         PIOR &= 0x7FU;  ※PIOR7=0 に変更する。
         PMC0 &= 0x3FU;  ※P06,P07 をデジタルポートに設定をします。
         PM0 |= 0x80U;

         P0 |= 0x40U;        ※P06 のポート設定をします。
         PM0 &= 0xBFU;

      /*P15/RxD0 & P10/TxD0 case */
         //PIOR |= 0x80U;  ※PIOR7=1 に変更する。
         //PM1 |= 0x20U;

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

RL78/G1N の場合

```
      /*P07/RxD0 & P06/TxD0 case */
         PIOR &= 0x7FU;  ※PIOR7=0 に変更する。
         PMC0 &= 0x3FU;  ※P06,P07 をデジタルポートに設定をします。
         PM0 |= 0x80U;

         P0 |= 0x40U;        ※P06 のポート設定をします。
         PM0 &= 0xBFU;

      /*P137/RxD0 & P01/TxD0 case for RL78/G1N*/
         //PIOR |= 0x10U;  ※PIOR4=1,PIOR5,6=0 に変更します。
         //PIOR &= 0x9FU;

         //P0 |= 0x02U;  ※P01 のポート設定をします。P137 端子は設定不要
         //PM0 &= 0xFDU;

      /*P137/RxD0 & P16/TxD0 case for RL78/G1N*/
         //PIOR |= 0x40U;  ※PIOR6=1,PIOR4,5=0 に変更します。
         //PIOR &= 0xCFU;

         //PMC1 &= 0xBFU;  ※P01 のポート設定をします。P137 端子は設定不要
         //P1 |= 0x40U;
         //PM1 &= 0xBFU;

}
```

2.3.9　リアルタイム出力制御回路 (RL78/G1M のみ)

RL78/G10 では搭載していない機能のため，新たにコードを作成する必要があります。

必要に応じてサンプルコードを変更してください
リアルタイム出力に使用するタイマ出力初期化関数は「2.3.3 タイマ・アレイ・ユニット」で作成する必要があります。

サンプルコード(rto)
・**r_cg_rto.h** (R_RTO_Create 関数宣言のみ)
・**r_cg_rto.c (**R_RTO_Create)

2.3.10 外部割り込み機能 (INTPn)

RL78/G10 用 CG/AP4 を利用できます。ただし, INTPn 端子の兼用ポートが異なる場合は, 端子設定を変更する必要があります。さらに, INTP4, INTP5 を利用する場合は, 新たにコードを作成する必要があります。

表 2.5　INTPn 端子の兼用ポート

INTPn	RL78/G10	RL78/G1M, RL78/G1N
INTP0	P137	P137
INTP1	P00 (P03)	P06 (P11)
INTP2	P41 (P122)	P15
INTP3	P06 (P121)	P14
INTP4	未搭載	P01
INTP5	未搭載	P00

注意. () は PIOR によるリダイレクション設定時。

サンプルコード(intp)

・**r_cg_intp.h**:　　サンプルコードの r_cg_intp.h を利用する。必要に応じて設定を変更する。
・**r_cg_intp.c**:　　サンプルコードの r_cg_intp.c を利用する。必要に応じて設定を変更する。
・**r_cg_intp_user.c**: サンプルコードの r_cg_intp.user.c を利用する。必要に応じて設定を変更する。

サンプルコード変更例

r_cg_intp.h

```
/*★INTPn setting*/ ※使用する INTPn 端子を有効にします。
void R_INTC_Create(void); ※INTPn を使用する場合
void R_INTC0_Start(void);   ※INTP0 を使用する場合
void R_INTC0_Stop(void);   ※INTP0 を使用する場合
//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);
```

r_cg_intp.c

```
 /*★Priority setting*/   ※必要に応じて, 設定割り込みの優先順位を変更する。
 /* 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) */

   /* Priority level setting for INTP0 */ ※設定変更しない場合は, “Level 3” になります。
      //     PPR10 = 1U;
      //     PPR00 = 1U;
   /* Priority level setting for INTP1 */
      //     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 = 1: Both rising and falling edges*/

※不要なコード部分を削除します。
   EGN0 = _01_INTP0_EDGE_FALLING_SEL ※INTP0に対して立ち下がりエッジを有効の場合
   //EGP0 = _01_INTP0_EDGE_RISING_SEL | _02_INTP1_EDGE_RISING_SEL |・・・・
                  :

/*★INTPn port setting*/ ※使用するINTPn端子の端子設定をします。
     /* Set INTP0 to P137 */
        //none ※設定不要
     /* Set INTP1 to P06 or P11*/
     /*P06 case, PIOR2 = 0*/
        //    PIOR &= 0xFBU    /* after RESET*/
        //PM0 |= 0x40U;
                  :

/*★INTP0*/ ※使用するINTPn端子の関数設定を有効します。
void R_INTC0_Start(void)     ※INTP0を使用する場合
{
      PIF0 = 0U;       /* clear INTP0 interrupt flag */
      PMK0 = 0U;       /* enable INTP0 interrupt */
}

void R_INTC0_Stop(void)     ※INTP0を使用する場合
{
      PMK0 = 1U;       /* disable INTP0 interrupt */
      PIF0 = 0U;       /* clear INTP0 interrupt flag */
}
                        :
```

r_cg_intp_user.c

CC-RLの場合

```
/*★INTPn Pragma*/ ※使用するINTPn端子の設定を有効します。
#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)
                                    :

/*★*/ ※使用するINTPn端子の関数設定を有効します。
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 *///}
```

AP4 for RL78 の場合

```
#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 キー割り込み機能

RL78/G10 用 CG/AP4 を利用できます。ただし, KRn 端子の兼用ポートが異なるため, 端子設定を変更する必要があります。さらに, KR6, KR7 を利用する場合は, 新たにコードを作成する必要があります。

表 2.6　RL78/G10（R5F10Y47）と RL78/G1M,RL78/G1N キー割り込み機能の差分

KRn	RL78/G10	RL78/G1M, RL78/G1N
KR0	P40	P40 (P00 注)
KR1	P125	P125
KR2	P01	P07
KR3	P02	P10
KR4	P03	P11
KR5	P04	P12
KR6	未搭載	P13
KR7	未搭載	P16

注.　RL78/G1N のみ
注意.　() は PIOR によるリダイレクション設定時

サンプルコード(key)
・**r_cg_key.h**:　　サンプルコードの r_cg_key.h を利用する。
・**r_cg_key.c**:　　サンプルコードの r_cg_key.c を利用する。必要に応じて設定を変更する。
・**r_cg_key_user.c**: RL78/G10 用 CG/AP4 出力を使用する。必要に応じてユーザコード記述する。

サンプルコード変更例

```
void R_KEY_Create(void)
{
    volatile uint8_t w_count;
/* ★KRn setting */  ※使用するKRn端子を有効します。
/* Set KR0 to P40 or P00 (RL78/G1N only)*/
   /*P40 case, PIOR3 = 0*/
     //   PIOR &= 0xF7U    /* after RESET*/
     //   PU4 |= 0x01U;
     //   PM4 |= 0x01U;
              :

/*★Priority setting */  ※必要に応じて，設定割り込みの優先順位を変更する。
    KRPR1 = 1U;
    KRPR0 = 1U;
              :

/*★Detect KRn*/
  /* IF detect KRn Change code */
/* KR0:_01_KR0_SIGNAL_DETECT_ON,  ※KR0を使用する。
   KR1:_02_KR1_SIGNAL_DETECT_ON,  ※KR1を使用する。
                    :                                */

※KR1を使用する場合: OFF設定をON設定に変更します。
    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　初期設定関数の変更

「2.3　使用関数の変更」設定した各機能の初期設定関数を有効にします。
・必要に応じて，サンプルコードの R_Systeminit()関数の PIOR 設定を変更する。
・R_Systeminit()関数で使用する関数を有効にする（コメントアウトを外す）。
・有効にした関数用 include file を有効にする（コメントアウトを外す）。

サンプルコード
r_cg_systeminit.c:　サンプルコードの r_cg_systeminit.c を利用する。

サンプルコード変更例

```
/*★Activate the required include file */
#include "r_cg_port.h"  ※Port機能を有効にする。
//#include "r_cg_cgc.h"
//#include "r_cg_tau.h"
          :
```

G1M の場合

```
/*★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 | */
                   :
```

G1N の場合

```
/*★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   ※必要に応じて，設定を変更する。
                   :

/*★Activate the required functions*/
     R_PORT_Create();  ※Port機能を有効にする。
 //   R_CGC_Get_ResetSource();
 //   R_CGC_Create();
 //   R_TAU0_Create();
                   :
```

2.5　main 関数の作成

使用する関数用 include file を有効し、ユーザコードを記述します。

サンプルコード
r_cg_main.c:　サンプルコードの r_cg_main.c を利用する。

サンプルコード変更例

```
#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"  ※外部割り込み機能を有効にする（コメントアウトを外す）。
```

2.6　option byte の設定

RL78/G1M, G1N ユーザーズマニュアル ハードウェア編を確認し，新たに設定します。CS+, e2studio の IDE では，リンク・オプションでオプション・バイトを設定してください。

IAR の場合、AP4 for RL78 から出力される AP4 コードの r_cg_main.c ファイル内 に出力されますのでソース上で変更する必要があります。

```
/* 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　ユーザ・プログラムの作成時の注意事項

サンプルコードは，割り込み機能の使用を前提としています。割り込み機能を使用しないで割り込み要求フラグを使用する場合は，使用しない割り込み機能を無効（コメントアウト）にしてください。無効にする場合は，r_cg_XXX_user.c の割り込みベクタ定義や割り込み処理関数をプロジェクト登録から外してください。

サンプルコード変更例

r_cg_adc.c ファイル

```
void R_ADC_Start(void)
{
    ADIF = 0U;    /* clear INTAD interrupt flag */
//    ADMK = 0U;    /* enable INTAD interrupt */  ※無効にする。
    ADCS = 1U;    /* enable AD conversion */
}
```

3.　アプリケーション開発

G1M フォルダ内にあるサンプルコードを使用し，アプリケーションを開発します。100ms ごとに ANI0 端子電圧の A/D 変換結果を UART 送信するアプリケーションを開発します。

3.1　動作確認条件

本アプリケーションノートのサンプルコードは，下記の条件で動作を確認しています。

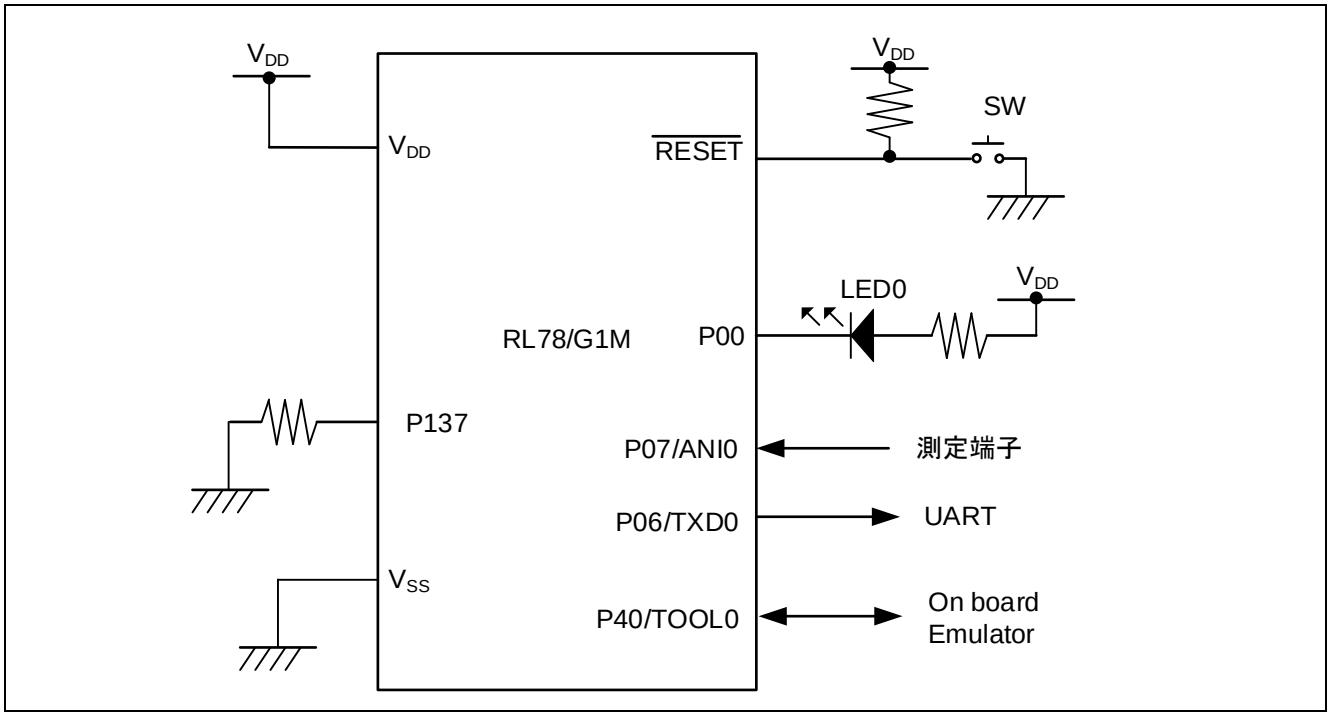
表 3.1　動作確認条件

項目	内容
使用マイコン	RL78/G1M（R5F11W68）
使用ボード	RL78/G1M Fast Prototyping Board (RTK5RLG230CLG000BJ)
動作周波数	高速オンチップ・オシレータ・クロック: 20MHz CPU／周辺ハードウエア・クロック: 20MHz
動作電圧	3.3V (3.02V～5.5V で動作可能) SPOR 検出電圧: リセット・モード 　立ち上がり時 TYP. 2.90 V (2.76 V ～ 3.02 V) 　立ち下がり時 TYP. 2.84 V (2.70 V ～ 2.96 V)
統合開発環境 (CS+)	ルネサス エレクトロニクス製 CS+ V8.07.00
C コンパイラ (CS+)	ルネサス エレクトロニクス製 CC-RL V1.11.00
統合開発環境 (e2studio)	ルネサス エレクトロニクス製 2022-01 (22.1.0)
C コンパイラ (e2studio)	ルネサス エレクトロニクス製 CC-RL V1.11.00
統合開発環境 (IAR)	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 V4.21.2
C コンパイラ (IAR)	
AP4 for RL78(IAR 用)	ルネサス エレクトロニクス製 AP4 (2.10.06.01) RL78/G10 V1.05.04.01

3.2　ハードウェア構成例

本アプリケーションで使用するハードウェア構成例を示します。

図 3.1 ハードウェア構成



注意 1　この回路イメージは接続の概要を示す為に簡略化しています。実際に回路を作成される場合は，端子処理などを適切に行い，電気的特性を満たすように設計してください（入力専用ポートは個別に抵抗を介して V_{DD} 又は V_{SS} に接続して下さい）。

2　V_{DD} は SPOR にて設定したリセット解除電圧（V_{SPOR}）以上にしてください。

3.3　使用端子一覧

表 3.2 に使用端子と機能を示します。

表 3.2　使用端子と機能

端子名	入出力	内容
P06/SO00/TxD0/INTP1/RTIO06	出力	データ送信用端子
P07/ANI0/SI00/RxD0/KR2/RTIO07	入力	AD 測定端子

3.4　ソフトウェア説明

3.4.1　オプション・バイト

表 3.3 にオプション・バイトの設定を示します。

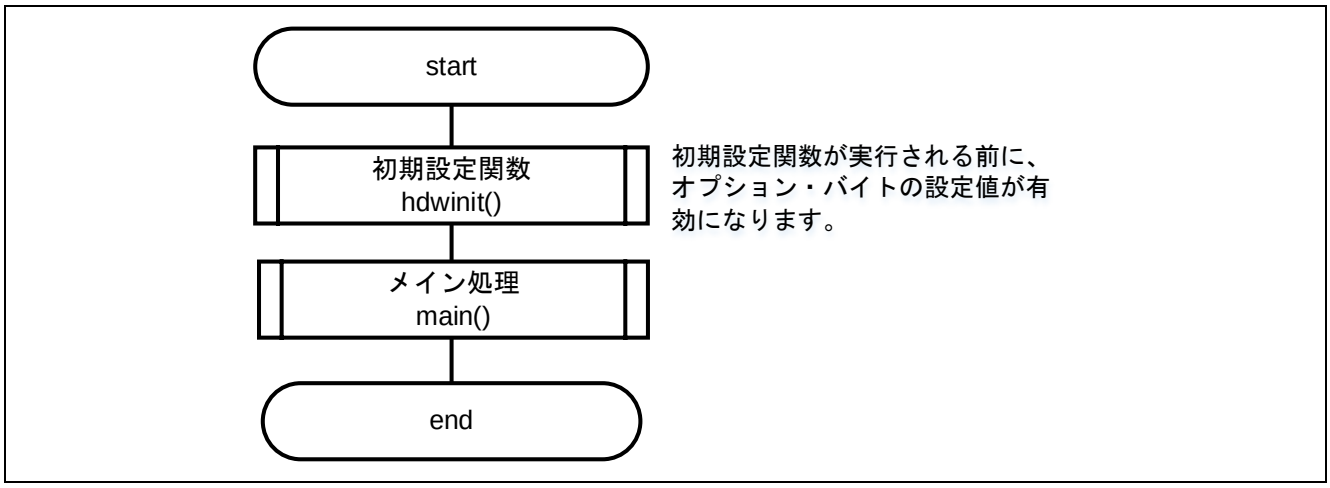
表 3.3　オプション・バイト設定

アドレス	設定値	内容
000C0H	11101110B	ウォッチドッグ・タイマ 動作停止 （リセット解除後，カウント停止）
000C1H	11110111B	LVD 動作：リセット・モード 立ち上がり時 TYP. 2.90 V (2.76 V ～ 3.02 V) 立ち下がり時 TYP. 2.84 V (2.70 V ～ 2.96 V)
000C2H	11111001B	HS モード， 高速オンチップ・オシレータ（HOCO）クロック：20MHz
000C3H	10000101B	オンチップ・デバッグ許可

3.4.2　フローチャート

図 3.2 に全体フローを示します。

図 3.2　全体フロー



注 初期設定関数の前後でスタートアップ・ルーチンが実行されます。

図 3.3 に初期設定フローを示します。

図 3.3　初期設定関数

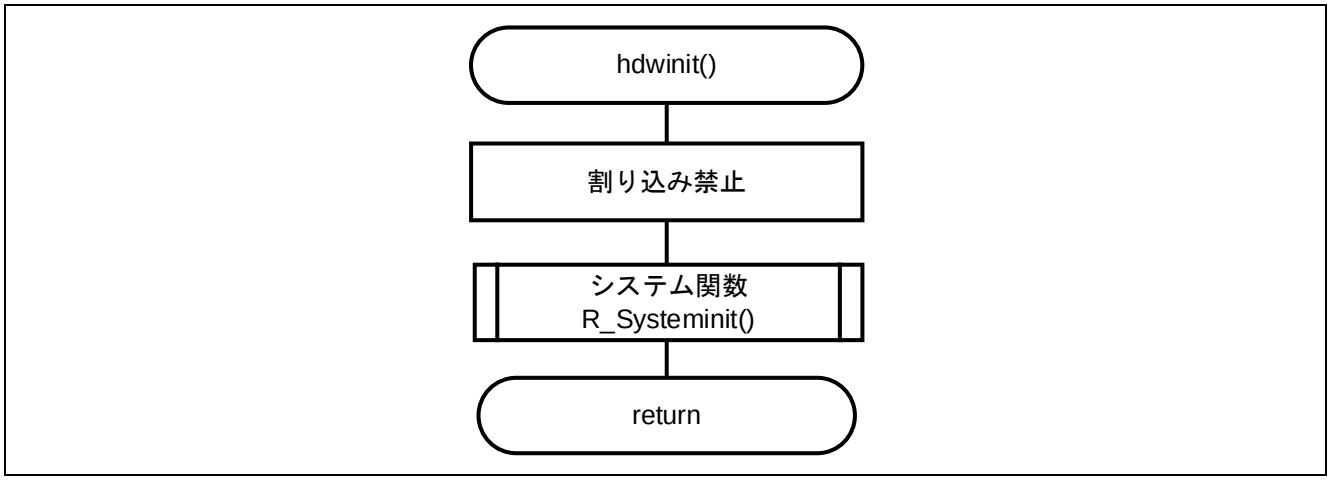


図 3.4 にシステム関数のフローを示します。

図 3.4　システム関数

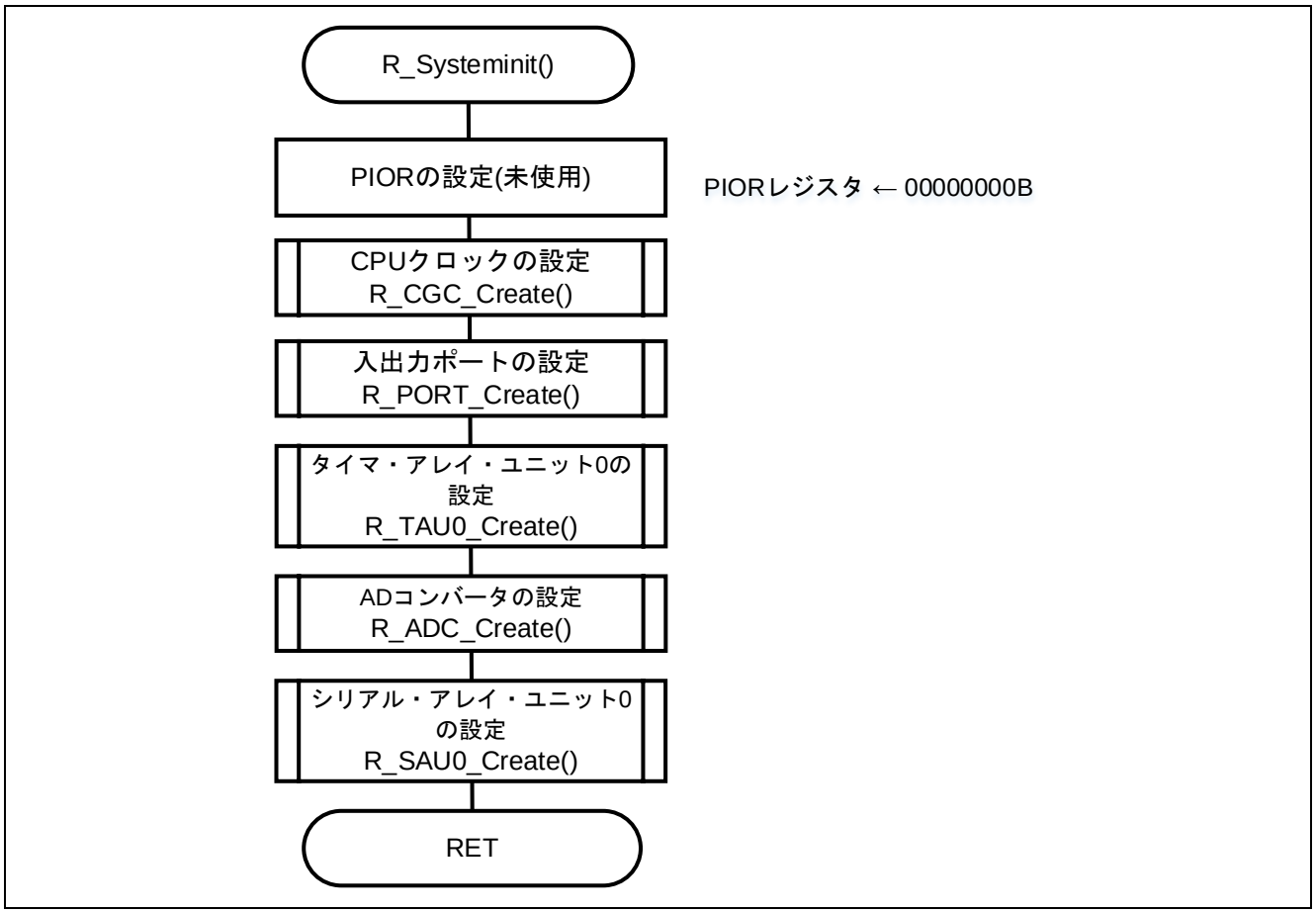


図 3.5 にメイン関数のフローを示します。

図 3.5　メイン関数

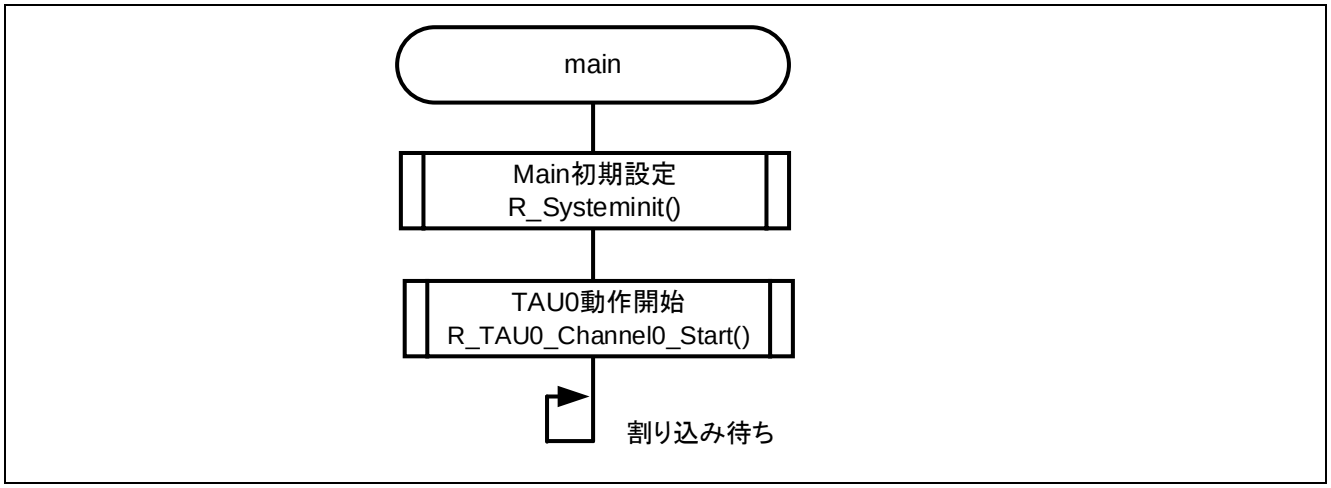
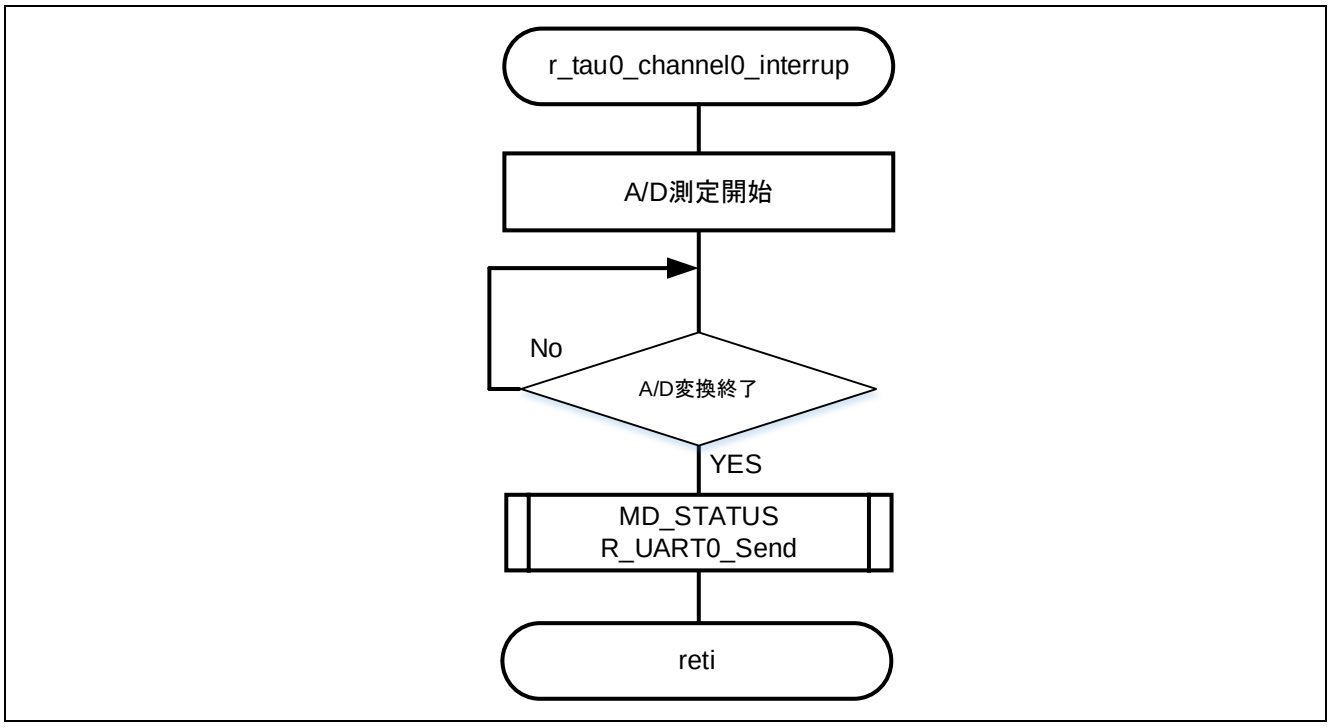


図 3.6 にユーザ関数のフローを示します。

図 3.6　ユーザ関数



3.5　開発手順

最初に G1M フォルダ内のすべてのフォルダおよびファイルを任意のフォルダにコピーします。 IDE 上で新たなプロジェクト名 (例: AD_UART) に変更します。

次に「3.4.2 フローチャート」に合わせてコードを作成します。以下に開発手順例を示します。ただし，ユーザ関数に関しては説明を省略します。

3.5.1　初期選定関数 (r_cg_systeminit.c)の開発

図 3.3 の初期設定フローで使用する初期設定関数を開発します。

① r_cg_systeminit.c 内で必要なインクルードファイルを有効にします。ポート設定，クロック設定，タイマ, AD コンバータ，シリアル通信の関連ファイルを有効にします。

```
#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"
```

② void R_Systeminit(void)関数内の PIOR を設定します。

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

③ 必要な初期設定関数を有効にします。

```
/*★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　システム関数の開発

次にシステム関数を開発します。図 3.4 で示している xxx_create 関数を開発します。

①　R_PORT_Create()の開発

PORT_Create()は r_cg_port.c にある関数です。LED を制御するために, P00 端子をデジタル出力ポートに設定します。LED の初期状態を消灯にするため, P0 に「1」を設定します。

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

②　R_CGC_Create()の開発

R_CGC_Create()は r_cg_cgc.c にある関数です。12 ビット・インターバル・タイマは使用しないので, クロック供給停止を選択します。

```
// OSMC = _10_CGC_IT_CLK_FIL;
  OSMC = _00_CGC_IT_CLK_NO;  ※fILを供給しない。
```

③　R_TAU0_Create()の開発

インターバル・タイマとして使用するため, 端子設定は不要です。RL78/G10 用 CG/AP4 の出力コードを変更せずに使用できます。RL78/G10 用 CG/AP4 で, 100ms ごとに INTTM00 割り込みが発生するように設定してコード出力します。開発中のプロジェクト (AD_UART) の r_cg_tau.c, r_cg_tau.h, r_cg_tau_user.c を RL78/G10 用 CG/AP4 の出力コード r_ch_tau.c, r_cg_tau.h, r_cg_tau_user.c に差し替えます。

④　R_ADC_Create()の開発

R_ADC_Create()は r_cg_adc.c にある関数です。ANI0 使用, 8 ビットモード, 変換クロック(fCLK/8)に設定します。

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

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

/*_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;  ※ANI0 端子を選択する。

/*★ AD comparator ADCE=1:enable ADCE=0:disable */   ※A/D変換待機状態
ADCE = 1U;
```

⑤　R_SAU0_Create()および R_UART0_Create()の開発

R_SAU0_Create()および R_UART0_Create()は r_cg_sau.c にある関数です。RL78/G10 用 CG/AP4 で，シングル転送モード，8 ビット長，LSB ファースト，ストップビット 1 ビット，非反転出力，転送レート 9600bps に設定してコード出力します。

シリアル入出力端子の兼用ポートが異なるため，端子設定を変更します。

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*/    ※RxD0は使用しないため, TxD0兼用ポート設定のみします。
      /*P07/RxD0 & P06/TxD0 case for RL78/G1M & /G1N*/

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

3.5.3　メイン関数（r_cg_main.c）とユーザ・プログラム(r_cg_tau_user.c.c)の開発

メイン関数を開発します。図 3.5 の処理を記述します。

①r_cg_main.c 内で必要なインクルードファイルを有効にします。

```
/*★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"
```

②次にメインプログラムを開発します。

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 */
    }
```

③最後にユーザ・プログラムを開発します。

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　オプション・バイト設定とビルド

コード変更が完了したら,プロジェクトから不要ファイルを外します。次に，リンク・オプションでオプション・バイトを設定します。

「ビルドツール」→「リンク・オプション」→「デバイス」

最後に，ビルドを実行します。

IAR 社コンパイラ使用時の追加設定

プログラムソース上でオプション・バイト設定します。r_cg_main.c の以下部分を適宜変更してください。

```
/* 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;
```

・macrodriver.h の変更

AP4 から出力される macrodriver.h は G10 用のデバイスファイルをインクルードしています。ターゲットデバイス用に変更します。以下は G1M（8K ROM 品）の場合です。

また、IAR 用の各デバイス用のインクルードファイルは IAR インストールフォルダ以下
IAR Systems¥Embedded Workbench 8.5¥rl78¥inc からコピーしてご使用ください。

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

4.　サンプルコード

以下のサンプルコードを用意しています。

G1M、G1N フォルダのソース：G10CG/AP4 出力から G1M,G1N 用の開発元となるソースコード

AD_UART：G1M フォルダ内ソースから 3 章の手順で開発したサンプルプロジェクト一式

サンプルコードは，ルネサスエレクトロニクスホームページから入手してください。

5.　参考ドキュメント

RL78/G1M,G1N　ユーザーズマニュアルハードウェア編(R01UH0904J)

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

RL78　ファミリユーザーズマニュアルソフトウエア編（R01US0015J）

コード生成ツール ユーザーズマニュアル RL78 API リファレンス編

テクニカルアップデート／テクニカルニュース

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

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com

お問合せ先

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2022.5.31		初版発行

製品ご使用上の注意事項

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

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。