R8C ファミリから RL78 ファミリへの置き換えガイド(CcnvNC30)

R01AN3508JJ0100
Rev.1.00
2017.01.20

要旨

本アプリケーションノートでは、R8C 用プログラムを RL78 用プログラムに置き換える方法について説明します。

対象デバイス

R8C ファミリ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. R8C ファミリから RL78 ファミリへのプログラムの置き換え方法 3

2. CcnvNC30 によるプログラム変換 3
2.1 CcnvNC30 について 3
2.2 CcnvNC30 の使用方法 4
2.3 CcnvNC30 を使用しない場合 6

3. 周辺機能のプログラム変換 7
3.1 プログラムの自動生成 7
3.2 プログラムの追加 9
3.3 コード生成ツールを使用しない場合 9

4. 置き換え例 10
4.1 R8C サンプルプログラム(RTC を使用した時計動作) 10
4.1.1 CcnvNC30 を使用した CC-RL へのソース移植 10
4.1.2 プログラムの自動生成 12
4.1.3 プログラム追加 14
4.1.4 その他修正事項 18
4.1.5 置き換え後のサンプルコード 19
4.2 R8C サンプルプログラム(DTC) 20
4.2.1 CcnvNC30 を使用した CC-RL へのソース移植 20
4.2.2 プログラムの自動生成 21
4.2.3 プログラム追加 24
4.2.4 その他修正事項 25
4.2.5 置き換え後のサンプルコード 26
4.3 サンプル・プログラムの動作確認条件 27

5. サンプルコード 27

6. 参考ドキュメント 27

1. R8C ファミリから RL78 ファミリへのプログラムの置き換え方法

R8C ファミリ用プログラムを RL78 ファミリ 用プログラムに置き換える方法について説明します。

最初に、C ソースコンバータ CcnvNC30 を使用して、C コンパイラ NC30 用拡張機能を C コンパイラ CC-RL 用拡張機能に変換します。

次に、統合開発環境 CS+または e2studio でプロジェクトを作成します。R8C ファミリと RL78 ファミリは周辺機能が異なるため R8C ファミリの周辺機能用プログラムを使用せず、RL78 ファミリ用コード生成ツールを利用して RL78 ファミリの周辺機能用プログラムを生成します。

CcnvNC30 で変換したプログラムと上記周辺機能用プログラムを組み合わせて、プログラムを置き換えます。

2. CcnvNC30 によるプログラム変換

2.1 CcnvNC30 について

CcnvNC30 は、NC30 用 C ソース・プログラムに対して、拡張言語仕様であるマクロ名、予約語、#pragma 指令、拡張機能の記述を CC-RL の拡張言語仕様に変換します。

なお、CcnvNC30 は、NC30 用プログラムから CC-RL 用プログラムへの移行を支援するためのソフトウェアです。変換後のプログラムが完全に動作することを保証されていません。必ず、変換後のプログラムでシステムの動作確認をしてください。

また、配置アドレス、SFR へのアクセスおよびアセンブラ記述などデバイスに依存した記述は変換できません。必要に応じて、手動で RL78 ファミリ用に変換してください。

詳細は、「CcnvNC30 C ソースコンバータ ユーザーズマニュアル(R20UT3685J)」を参照してください。

2.2　CcnvNC30 の使用方法

CcnvNC30 を使用したプログラムの変換方法を下記に示します。

(1)　CcnvNC30 (CcnvNC30.exe)と NC30 用プログラムを任意の同じフォルダに置きます。
(2)　Windows のコマンドプロンプトを起動します。
(3)　CcnvNC30 が格納されているフォルダへカレントディレクトリを変更します。

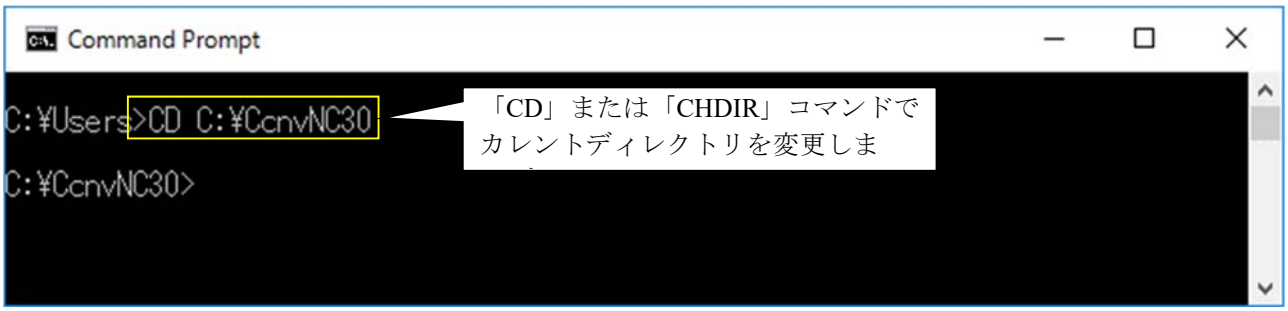

図 2.1　コマンドプロンプト画面

(4)　-o オプションで出力ファイル名を指定して実行します。実行後、CC-RL 用プログラムが出力されます。また、メッセージを指定したファイルに出力する場合は、-r オプションを指定します。

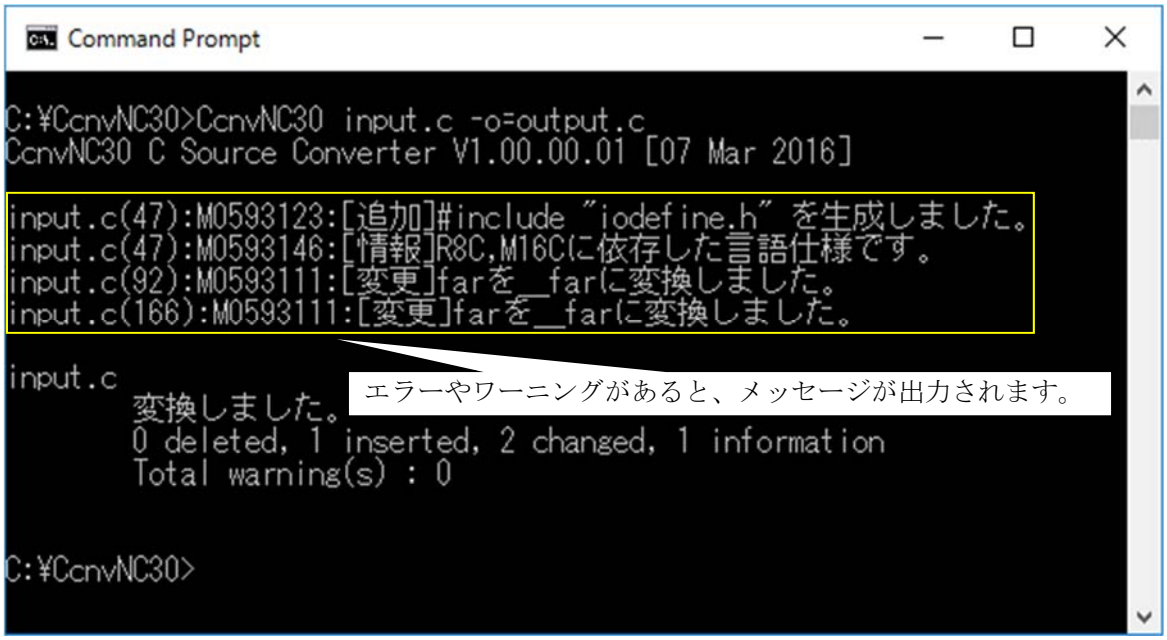

図 2.2　CcnvNC30 実行画面

(5)　複数のファイルを同時に変換する場合は、リスト・ファイルを作成し、-l オプションを実行します。実行後、指定したフォルダに CC-RL 用プログラムが出力されます。

図 2.3　CcnvNC30 実行画面(複数ファイル)

リスト・ファイルの記載例は下記となります。

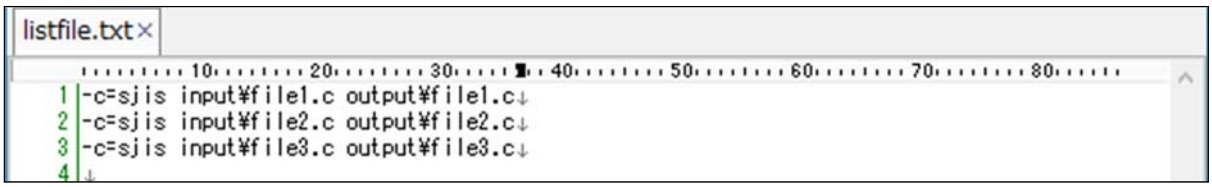

図 2.4　リスト・ファイル記載例

(6)　CcnvNC30 で変換されない箇所について修正します。修正箇所は「CcnvNC30　C ソースコンバータ ユーザーズマニュアル(R20UT3685J)」の「コンバータ変換仕様」を参照してください。

2.3　CcnvNC30 を使用しない場合

CcnvNC30 を使用しない場合は、NC30 用拡張機能を手動で CC-RL 用拡張機能に変換しなければなりません。CC-RL の拡張言語仕様については、「CC-RL コンパイラ ユーザーズマニュアル(R20UT3123J)」を参照してください。

3.　周辺機能のプログラム変換

3.1　　プログラムの自動生成

R8C ファミリで使用していた周辺機能と同種類の RL78 ファミリの周辺機能に対して、統合開発環境 CS+ または e2studio の RL78 ファミリ用コード生成ツールでプログラムの自動生成を行います。コード生成ツールの操作方法については、「CS+コード生成ツール 統合開発環境 ユーザーズマニュアル 周辺機能操作編 (R20UT3104J)」を参照してください。

(1)　プロジェクト・ツリーの中からコード生成(設計ツール)の「クロック発生回路」をクリックし、「端子の割り当て設定」を行います。
　なお、端子割り当て設定を一度確定させると端子割り当て設定は変更できません。

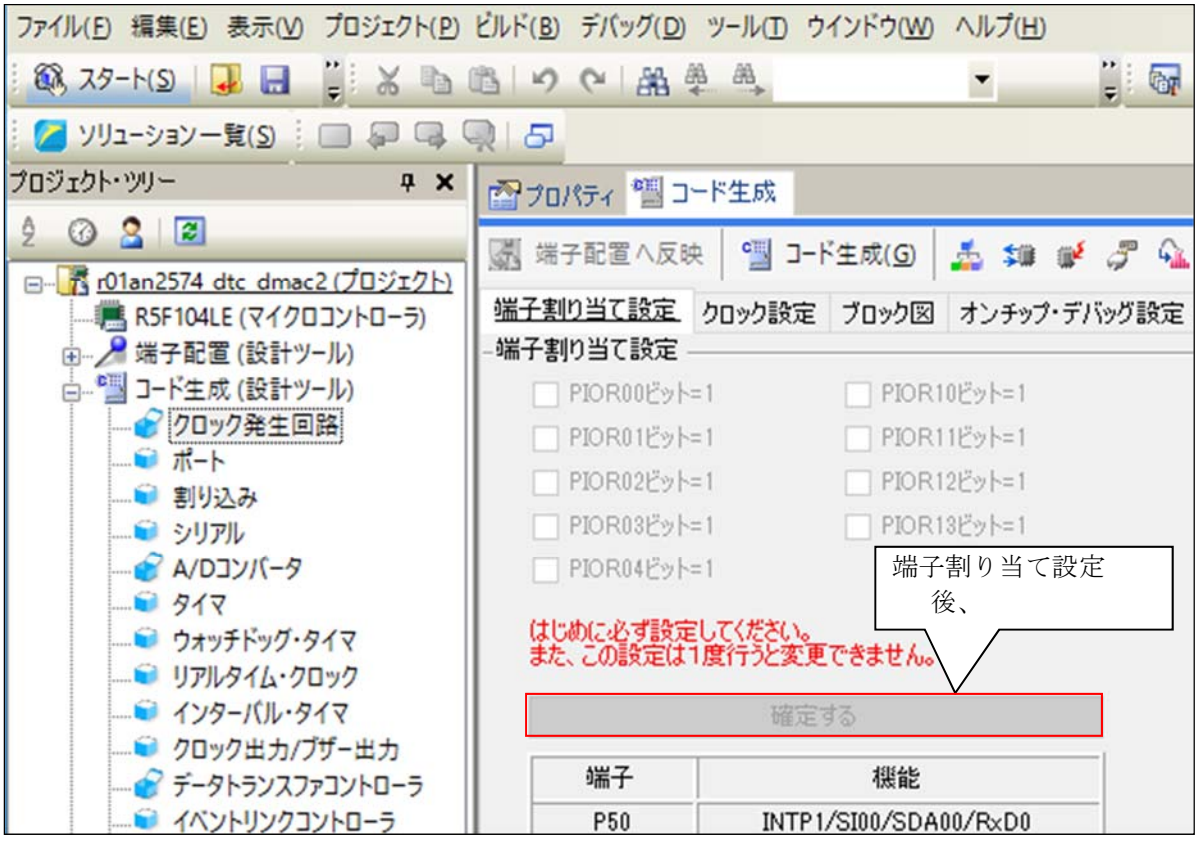

図 3.1　コード生成ツールの設定画面(1)

(2)　R8C ファミリのプログラムを参照して、各機能の設定を行います。

(3)　全ての周辺機能の設定が完了したら、画面上部にある「コード生成(G)」ボタンをクリックして、コード生成(プログラムの自動生成)を行います。自動生成した周辺機能の関数をプログラムの置き換えに使用します。

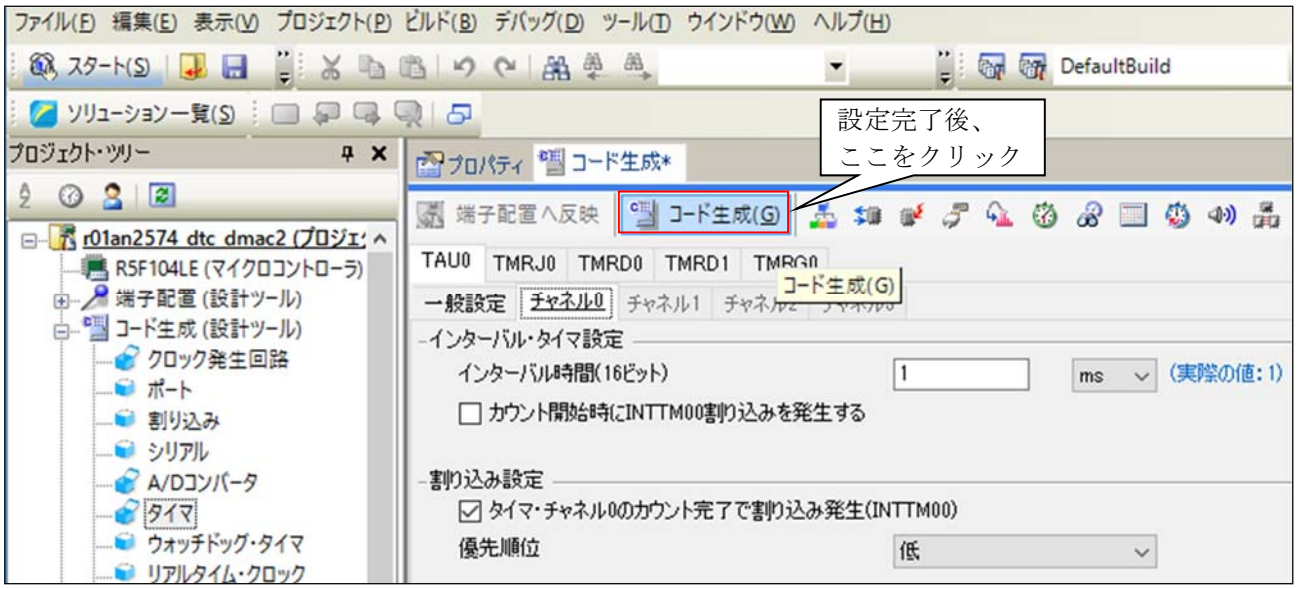

図 3.2　コード生成ツールの設定画面(2)

3.2　プログラムの追加

コード生成ツールで自動生成できないプログラム（main 関数、割り込み関数処理、変数など）を追加します。

自動生成された各ファイルの”/* Start user code for adding. Do not edit comment generated here */”と”/* End user code. Do not edit comment generated here */”の間にプログラムを追加します。プログラム追加は、手動で行う必要があります。なお、上記範囲外に追加したプログラムは、プログラムの自動生成時に、自動的に削除されます。

必ず、追加したプログラムでシステムの動作確認をしてください。

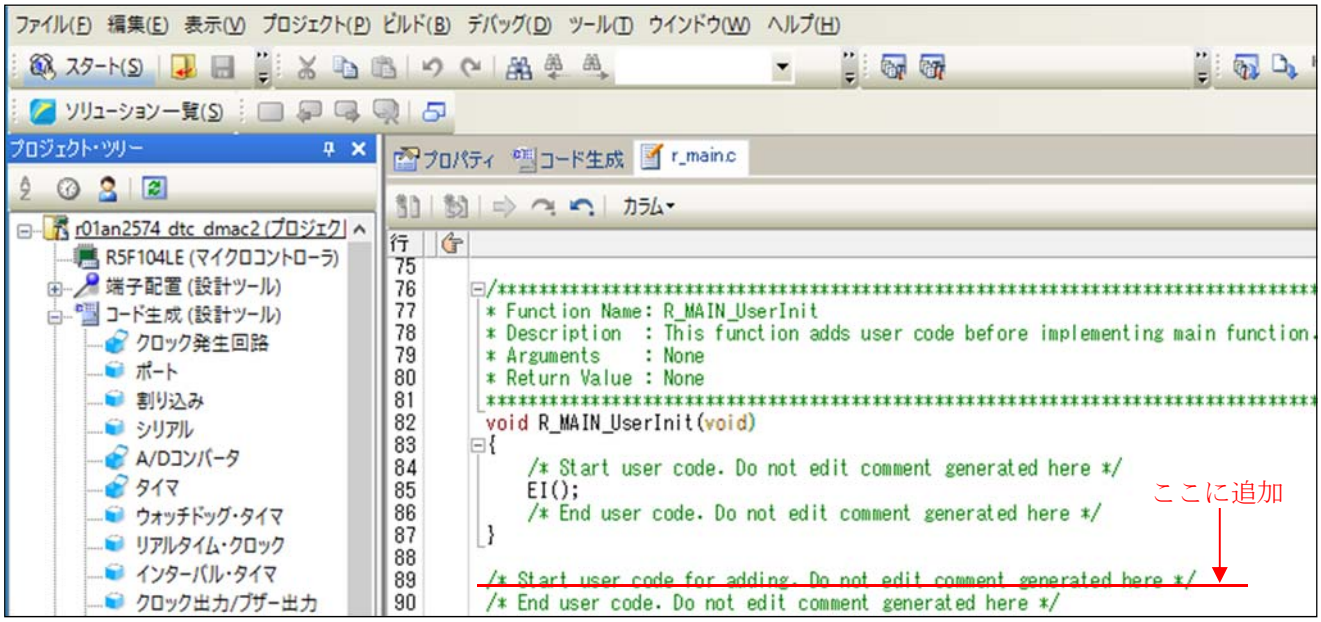

図 3.3　既存のプログラムの追加

3.3　コード生成ツールを使用しない場合

コード生成ツールを使用しない場合は、統合開発環境 CS+または e2studio で新規プロジェクトを作成した後、手動で周辺機能用プログラムを作成しなければなりません。周辺機能の詳細については、RL78 ファミリのユーザーズマニュアルを確認してください。

4.　置き換え例

4.1　　R8C サンプルプログラム(RTC を使用した時計動作)

「R8C/35A グループ RTC を使用した時計動作のプログラム」(R01AN0079J)を RL78/G14 用プログラムに置き換えます。置き換えた後のプロジェクトファイルは「r01an3508_rl78g14_rtc」です。

このプログラムでは、RL78/G14 のリアルタイム・クロック(RTC)を使用して、2009 年 1 月 1 日(木) 0 時 0 分 0 秒(初期設定)からカウントを開始します。CPU クロックは高速オンチップ・オシレータの 16MHz です。

4.1.1　　CcnvNC30 を使用した CC-RL へのソース移植

(1)　リスト・ファイルを作成し、変換する C ソース・ファイルを指定します。

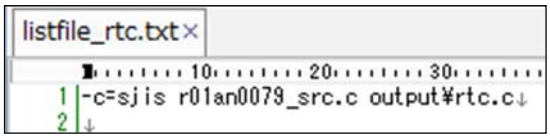

図 4.1　リスト・ファイルの記載例（RTC を使用した時計動作のプログラム）

(2)　コマンドプロンプトを起動し、リスト・ファイルで指定した C ソース・ファイルを変換します。また、出力した変換結果ファイルに変更箇所が記載されます。

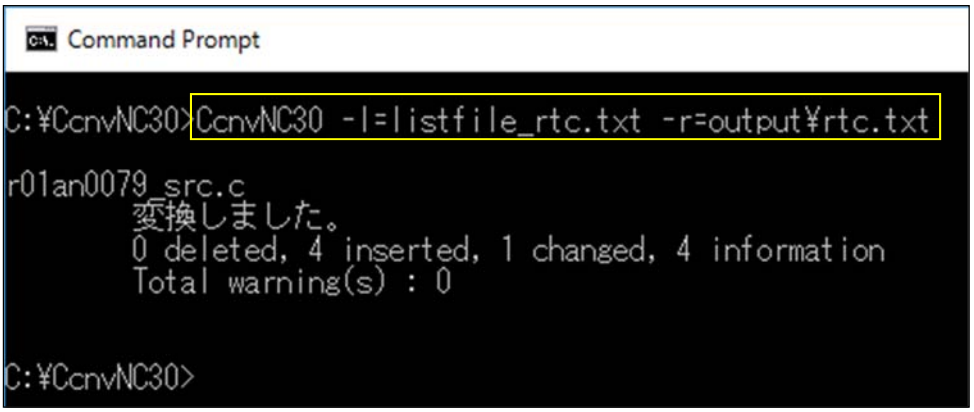

図 4.2　CcnvNC30 実行画面 (RTC を使用した時計動作のプログラム)

変換結果ファイルには下記のように変換結果が記載されます。変換結果の詳細については、「C ソースコンバータ CcnvNC30 ユーザーズマニュアル(R20UT3685J)」を参照してください。

図 4.3　変換結果の詳細（RTC を使用した時計動作のプログラム）

(3)　変換した C ソース・ファイルを修正します。

C コンパイラ CC-RL で不必要な割り込み関数宣言が定義されている可能性があります。C コンパイラ NC30 の記述では C コンパイラ CC-RL でエラーが発生するため、CcnvNC30 が変換した#pragma 宣言を削除します。

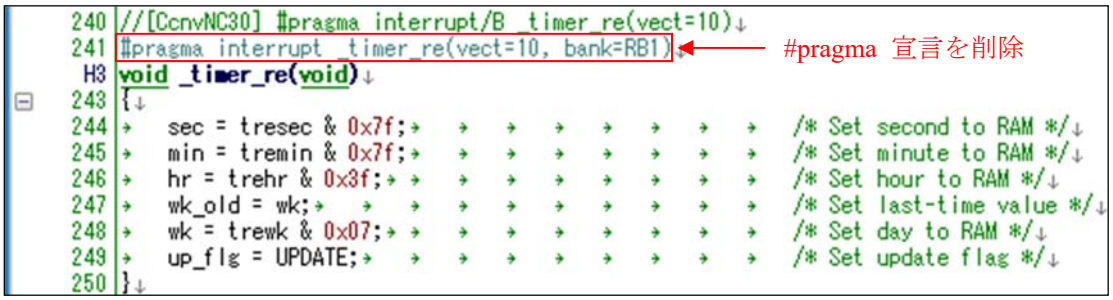

図 4.4　割り込み関数宣言の変更

4.1.2　　プログラムの自動生成

(1)　統合開発環境 CS+または e2studio　で新規プロジェクトを作成します。

(2)　コード生成ツールで各機能を設定します。

CPU クロックを高速オンチップ・オシレータクロック　16MHz に設定します。

RTC、インターバル・タイマ動作クロックを 32.768kHz に設定します。

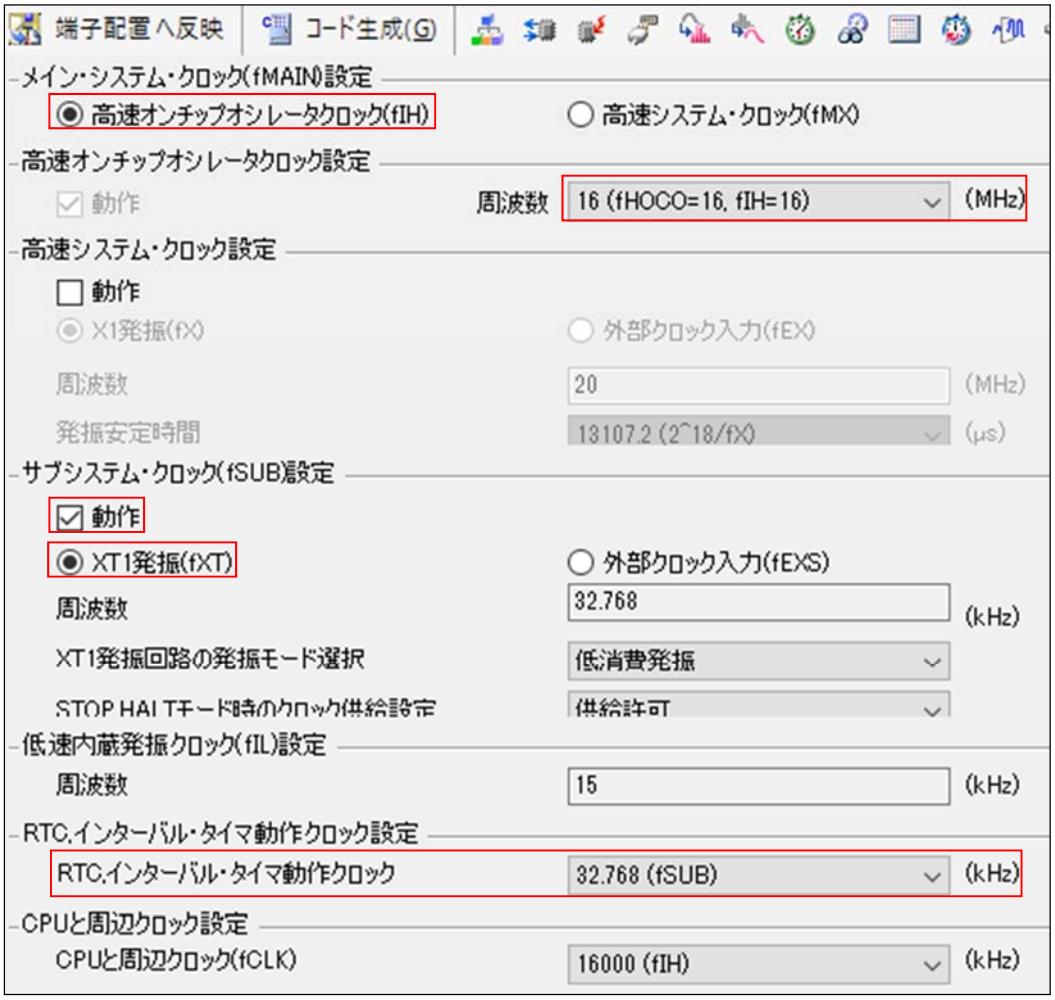

図 4.5　コード生成ツール設定画面（クロック）

R8C ファミリのタイマ RE リアルタイムクロックモード と同等の機能を有するリアルタイム・クロック(RTC)を設定します。

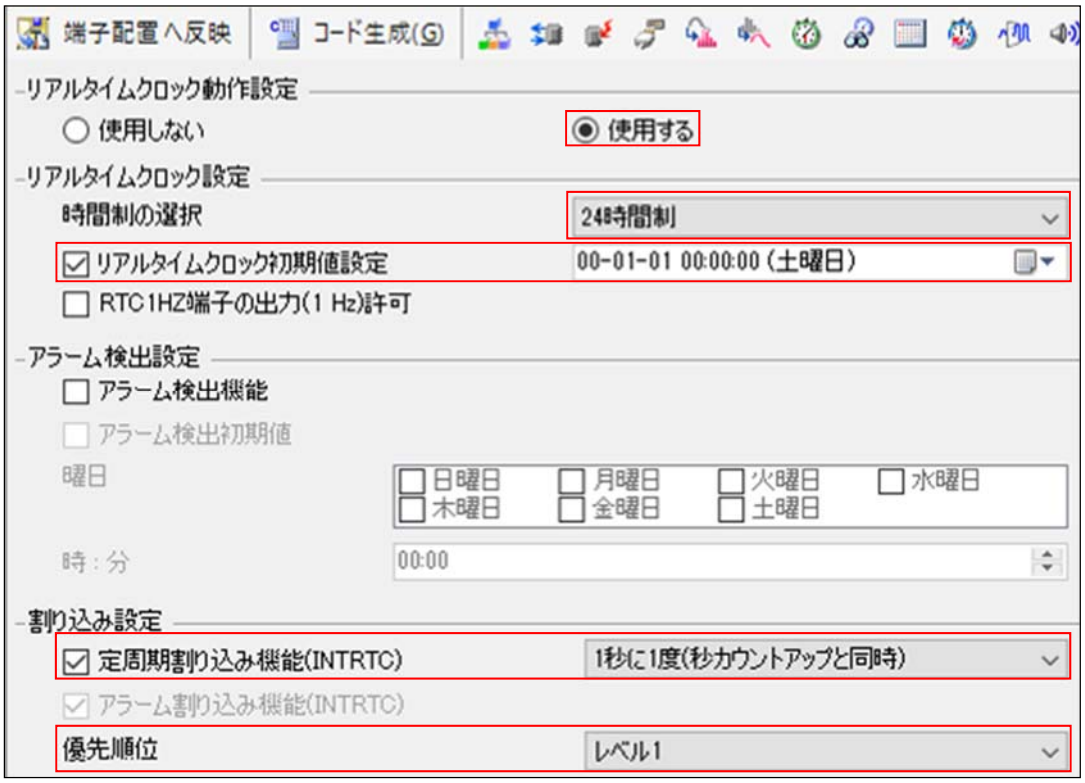

図 4.6　コード生成ツール設定画面(リアルタイム・クロック)

(3)　その他の「ポート」、「ウォッチドッグ・タイマ」、「電圧検出回路」をそれぞれ設定します。
(4)　「コード生成(G)」をクリックし、ファイルを生成します。

4.1.3　　プログラム追加

コード生成したプログラムにシンボル定義と main 関数の処理を追加します。その他のプログラム（クロックやリアルタイム・クロックの設定）については、コード生成したプログラムを使用します。

・シンボル定義

シンボル定義を r_cg_rtc.h に追加します。

R8C 用プログラム

```
94 /************************************************************************************/
95 /* DEFINE                                                                           */
96 /************************************************************************************/
97 #define NO_UPDATE    0    /* No update */
98 #define UPDATE       1    /* Update */
99 #define COMM         0    /* Common year */
100 #define LEAP         1    /* Leap year */
101 #define DEC          0x12 /* December */
102 #define WEEK         0x04 /* A day of the week(thursday) */
```

赤枠に対応する内容を手動で追記

RL78/G14 の r_cg_rtc.h ファイル

```
146     /* Start user code for function. Do not edit comment generated here */
147     #define NO_UPDATE       0               /* No update */
148     #define UPDATE          1               /* Update */
149     #define COMM            0               /* Common year */
150     #define LEAP            1               /* Leap year */
151     #define DEC             0x12            /* December */
```

図 4.7　シンボル定義の置き換え

・main 関数

RL78/G14 用コード生成ツールを使用すると、main 関数実行前に R_Systeminit 関数を実行します。R_Systeminit 関数では、クロックや RTC の初期設定を行います。このため、赤枠の処理のみ手動でプログラムを追加します。R_RTC_Start 関数は、RTC 動作を開始します。R8C/35A は、タイマ RE のリアルタイムクロックモードを使用しました。RL78/G14 は、リアルタイム・クロック(RTC)を使用します。

「R8C/35A グループ RTC を使用した時計動作のプログラム」(R01AN0079J)のメイン関数では、タイマ RE の初期設定と動作開始を実行します。そのため、r01an3508_rl78g14_rtc のメイン関数では、RTC の初期設定(1 秒に 1 度、定周期割り込みを発生する)と RTC の動作開始を実行します。

R8C 用プログラム

```
H3 void main(void)
104 {
105     asm("FCLR I");                  /* Interrupt disabled */
106
107     mcu_init();                     /* MCU initialize */
108
109     timer_re_init();                /* TimerRE initialize */
110
111     asm("FSET I");                  /* Interrupt enabled */
112
113     leap_flg = leap_chk();          /* Leap year check */
114
115     tstart_trecr1 = 1;              /* TRE1 count start */
116     while(tcstf_trecr1== 0);        /* If TRE1 count start? else wait */
117
118     while(1){
119         if(up_flg == UPDATE){
120             update();               /* Update processing */
121         }
122     }
123 }
```

赤枠に対応する内容を
手動で追記

RL78/G14 の r_main.c ファイル

```
59  void main(void)
60  {
61      R_MAIN_UserInit();
62      /* Start user code. Do not edit comment generated here */
63      leap_flg = leap_chk();                          /* Leap year check */
64
65      R_RTC_Set_ConstPeriodInterruptOn(SEC);          /* Interrupt turned on */
66      R_RTC_Start();                                  /* Enabled */
67
68      while (1U)
69      {
70          if(up_flg == UPDATE){
71              update();                               /* Update processing */
72          }
73      }
74      /* End user code. Do not edit comment generated here */
75  }
```

図 4.8　main 関数の置き換え

また、main 関数で使用する update 関数と leap_chk 関数を main.c に追加します。また、2 つの関数が使用する変数を main.c に追加します。

R8C 用プログラム

```
/************************************************************************/
/* RAM                                                                  */
/************************************************************************/
unsigned short year = 0x2009;      /* Year */
unsigned char month = 0x01;        /* Month */
unsigned char day = 0x01;          /* Day */
unsigned char wk = WEEK;           /* Sun,Mon,Tue,Wed,Thu,Fri,Sat */
unsigned char hr = 0x00;           /* Hour */
unsigned char min = 0x00;          /* Minute */
unsigned char sec = 0x00;          /* Second */

/* Work area */
unsigned char wk_old = 0x00;       /* Last-time value of 'wk' */

/* Flags */
unsigned char up_flg = NO_UPDATE;  /* Update flag */
unsigned char leap_flg = COMM;     /* Leap flag */

/************************************************************************/
/* ROM                                                                  */
/************************************************************************/
unsigned char const MONTH_DAYS[13] = {     /* This table contains the number of days in different months */
    0x00,0x31,0x28,0x31,0x30,0x31,0x30,0x31,0x31,0x30,0x31,0x30,0x31
};

unsigned char const MONTH_DAYS_L[13] = {   /* This table contains the number of days in different months (leap year) */
    0x00,0x31,0x29,0x31,0x30,0x31,0x30,0x31,0x31,0x30,0x31,0x30,0x31
};
```

赤枠に対応する内容を手動で追記

RL78/G14 の r_main.c ファイル

```
/***********************************************************************************************************************
Global variables and functions
***********************************************************************************************************************/
/* Start user code for global. Do not edit comment generated here */
unsigned short year = 0x2009;                   /* Year */
unsigned char month = 0x01;                     /* Month */
unsigned char day = 0x01;                       /* Day */
unsigned char wk = WEEK;                        /* Sun,Mon,Tue,Wed,Thu,Fri,Sat */
unsigned char hr = 0x00;                        /* Hour */
unsigned char min = 0x00;                       /* Minute */
unsigned char sec = 0x00;                       /* Second */

unsigned char wk_old = 0x00;                    /* Last-time value of 'wk' */

unsigned char up_flg = NO_UPDATE;               /* Update flag */
unsigned char leap_flg = COMM;                  /* Leap flag */

unsigned char const MONTH_DAYS[13] = {          /* This table contains the number of days in different months */
        0x00,0x31,0x28,0x31,0x30,0x31,0x30,0x31,0x31,0x30,0x31,0x30,0x31
};

unsigned char const MONTH_DAYS_L[13] = {        /* This table contains the number of days in different months (leap year) */
        0x00,0x31,0x29,0x31,0x30,0x31,0x30,0x31,0x31,0x30,0x31,0x30,0x31
};
```

図 4.9　関数の追加

R8C 用プログラム

```
/*""FUNC COMMENT""*************************************************
* Outline                 : Update processing
* Declaration             : void update(void)
* Description             : Update check
* Argument                : none
* Return Value            : none
*""FUNC COMMENT END""*************************************************/
void update(void)
{
    unsigned char  num = 0;

    up_flg = NO_UPDATE;                 /* Update flag clear */

    if(wk != wk_old){                   /* Change day ? */
    }
}

/*""FUNC COMMENT""*************************************************
* Outline                 : Leap year check processing
* Declaration             : unsigned char leap_chk(void)
* Description             : Leap year check
* Argument                : none
* Return Value            : none
*""FUNC COMMENT END""*************************************************/
unsigned char leap_chk(void)
{
    unsigned char  chk = 0;             /* Result of check */
    unsigned short dec = 0;
    unsigned short work = 0;

    work = year;                        /* Copy year */

    dec  = (((work & 0xF000) >> 12) * 1000);    /* Conversion to decimal */
    dec += (((work & 0x0F00) >>  8) *  100);
    dec += (((work & 0x00F0) >>  4) *   10);
    dec += (((work & 0x000F) >>  0) *    1);

    chk = COMM;                         /* Set Common year */

    if((dec % 4) == 0){
    }
    return(chk);
}
```

赤枠に対応する内容を手動で追記

RL78/G14 の r_main.c ファイル

```
/* Start user code for adding. Do not edit comment generated here */
/***********************************************************************
* Function Name             : update
* Description               : Update check
* Arguments                 : None
* Return Value              : None
***********************************************************************/
void update(void)
{ ... }

/***********************************************************************
* Function Name             : leap_chk
* Description               : Leap year check
* Arguments                 : None
* Return Value              : None
***********************************************************************/
unsigned char leap_chk(void)
{ ... }
/* End user code. Do not edit comment generated here */
```

図 4.10　変数の追加

4.1.4　その他修正事項

コード生成ツールで RTC を設定すると、r_cg_rtc_user.c に RTC の割り込み処理が自動生成されます。割り込み処理内に処理を追加します。

R8C 用プログラム

```
/*""FUNC COMMENT""*************************************************
* Outline                 : Timer RE interrupt processing
* Declaration             : void _timer_re(void)
* Description             : Timer RE interrupt
* Argument                : none
* Return Value            : none
*""FUNC COMMENT END""*********************************************/
#pragma interrupt/B _timer_re(vect=10)
void _timer_re(void)
{
    sec = tresec & 0x7f;                /* Set second to RAM */
    min = tremin & 0x7f;                /* Set minute to RAM */
    hr = trehr & 0x3f;                  /* Set hour to RAM */
    wk_old = wk;                        /* Set last-time value */
    wk = trewk & 0x07;                  /* Set day to RAM */
    up_flg = UPDATE;                    /* Set update flag */
}
```

赤枠に対応する内容を手動で追記

RL78/G14 の r_cg_rtc_user.c ファイル

```
/***********************************************************************************************************************
* Function Name: r_rtc_interrupt
* Description  : This function is INTRTC interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
static void __near r_rtc_interrupt(void)
{
    if (1U == RIFG)
    {
        RTCC1 &= (uint8_t)~_08_RTC_INTC_GENERATE_FLAG;    /* clear RIFG */
        r_rtc_callback_constperiod();
    }
}

/***********************************************************************************************************************
* Function Name: r_rtc_callback_constperiod
* Description  : This function is real-time clock constant-period interrupt service handler.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
static void r_rtc_callback_constperiod(void)
{
    /* Start user code. Do not edit comment generated here */
    sec = tresec & 0x7f;                                  /* Set second to RAM */
    min = tremin & 0x7f;                                  /* Set minute to RAM */
    hr = trehr & 0x3f;                                    /* Set hour to RAM */
    wk_old = wk;                                          /* Set last-time value */
    wk = trewk & 0x07;                                    /* Set day to RAM */
    up_flg = UPDATE;                                      /* Set update flag */
    /* End user code. Do not edit comment generated here */
}
```

図 4.11　割り込み処理の追加

また、r_cg_rtc_user.c 以外で定義された変数を参照する場合、変数の外部参照を行います。

```
    /* Start user code for global. Do not edit comment generated here */
    extern unsigned short year;
    extern unsigned char month;
    extern unsigned char day;
    extern unsigned char wk;
    extern unsigned char hr;
    extern unsigned char min;
    extern unsigned char sec;

    extern unsigned char wk_old;

    extern unsigned char up_flg;
    /* End user code. Do not edit comment generated here */
```

図 4.12　外部参照の追加

4.1.5　　置き換え後のサンプルコード

ルネサス エレクトロニクスホームページからサンプルコード「an-r01an3508je0100-rl78-migrate.zip」を入手してください。「workspace」フォルダ内の「rl78g14_migrate_rtc」が「R8C/35A グループ RTC を使用した時計動作のプログラム」(R01AN0079J)を置き換えたサンプルコードとなります。

4.2　R8C サンプルプログラム(DTC)

「R8C/35C グループ DTC(チェイン転送)」(R01AN0372E)を RL78/G14 用プログラムに置き換えます。置き換えた後のプロジェクトファイルは「r01an3508_rl78g14_dtc」です。

このプログラムでは、RL78/G14 のデータ・トランスファ・コントローラ(DTC)のチェイン転送を使用します。

4.2.1　CcnvNC30 を使用した CC-RL へのソース移植

(1)　リスト・ファイルを作成し、変換する C ソース・ファイルを指定します。

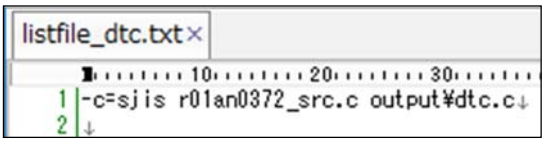

図 4.13 リスト・ファイルの記載例（DTC のチェイン転送）

(2)　コマンドプロンプトを起動し、リスト・ファイルで指定した C ソース・ファイルを変換します。また、出力した変換結果ファイルに変更箇所が記載されます。

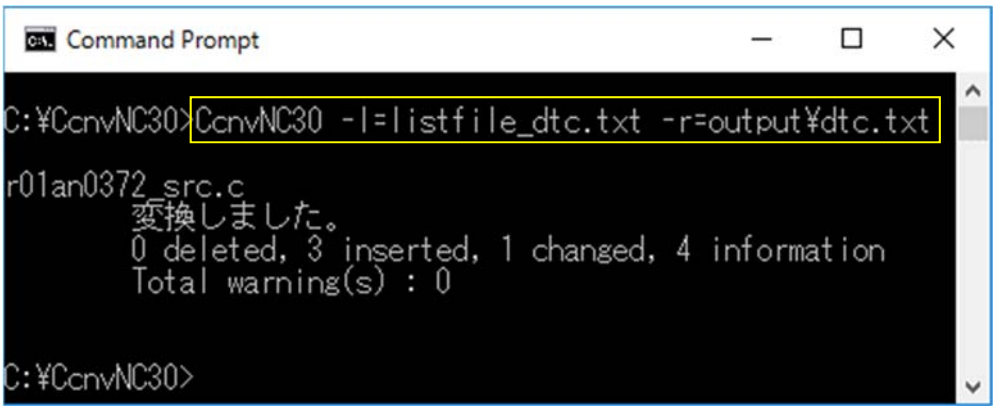

図 4.14 CcnvNC30 実行画面 (DTC のチェイン転送)

変換結果ファイルには下記のように変換結果が記載されます。変換結果の詳細については、「C ソースコンバータ CcnvNC30 ユーザーズマニュアル(R20UT3685J)」を参照してください。

図 4.15 変換結果の詳細（DTC のチェイン転送）

(3)　変換した C ソース・ファイルを修正します。

C コンパイラ CC-RL で不必要な割り込み関数宣言が定義されている可能性があります。C コンパイラ NC30 の記述では C コンパイラ CC-RL でエラーが発生するため、CcnvNC30 が変換した#pragma 宣言を削除します。

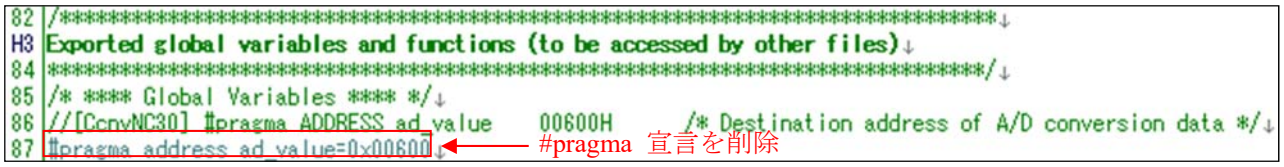

図 4.16 割り込み関数宣言の変更

4.2.2　プログラムの自動生成

(1)　統合開発環境 CS+または e2studio で新規プロジェクトを作成します。

(2)　コード生成ツールで各機能を設定します。

CPU クロックを高速オンチップ・オシレータクロック 16MHz に設定します。

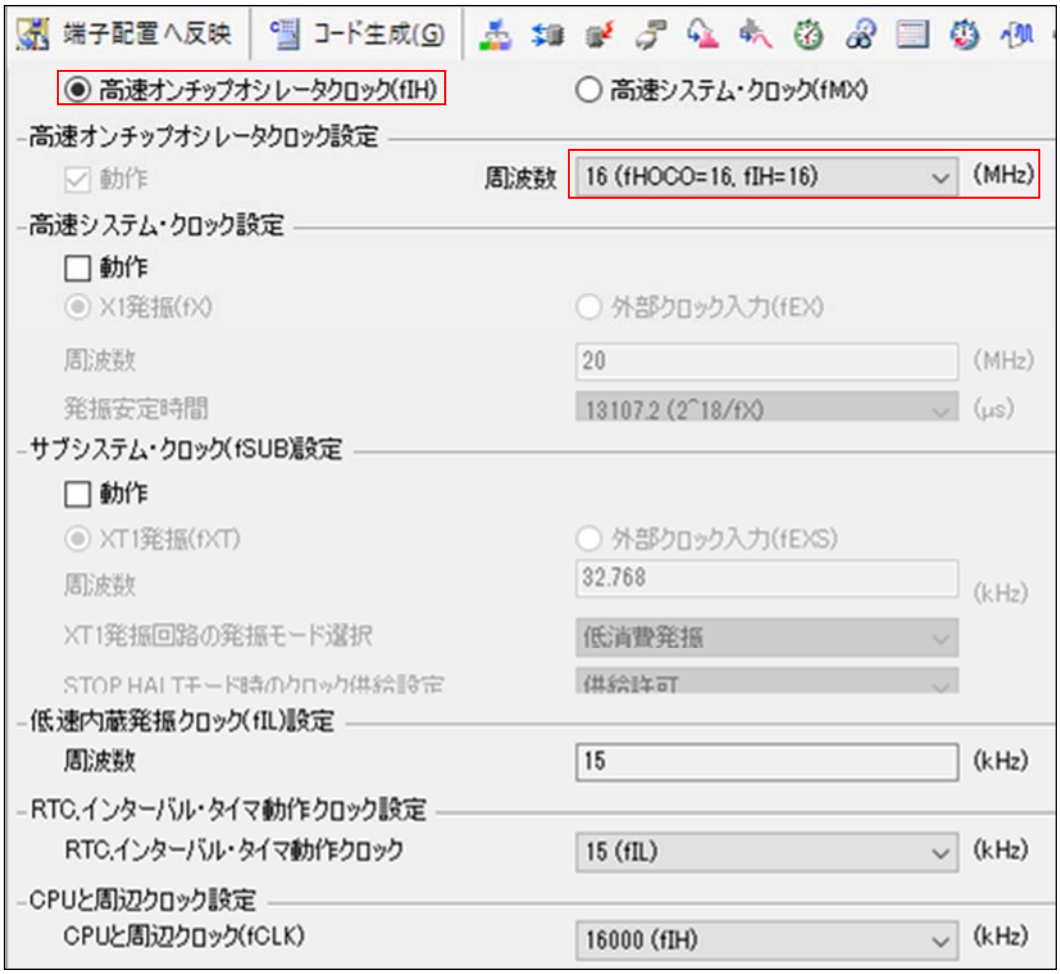

図 4.17　コード生成ツール設定画面（クロック）

R8C ファミリの A/D コンバータと同等機能である A/D コンバータ(ADC)を設定します。

A/D コンバータは、ソフトウエア・トリガ・モード、スキャン・モード、ワンショット変換モードを設定します。

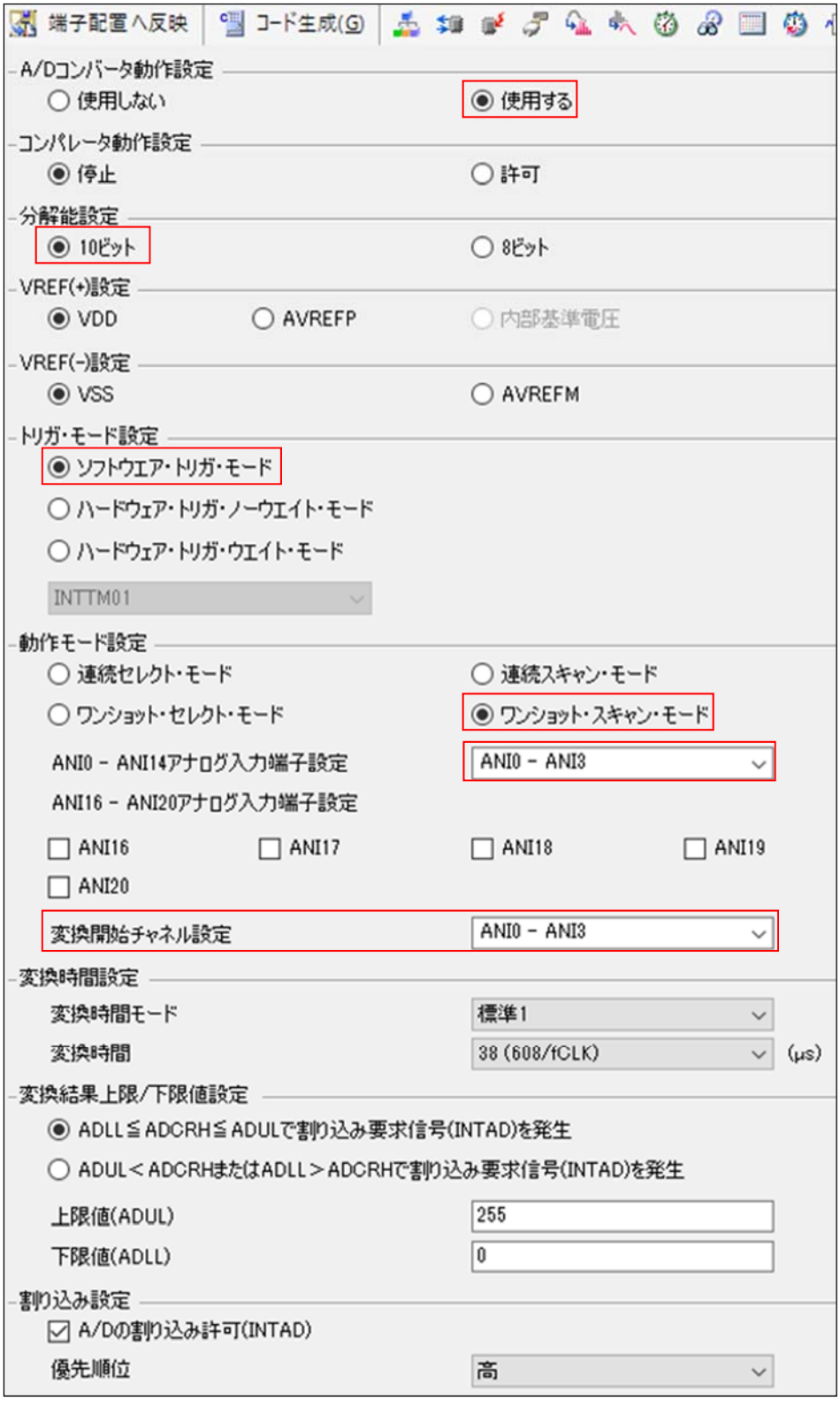

図 4.18　コード生成ツール設定画面(A/D コンバータ)

R8C ファミリの DTC と同等の機能を有するデータ・トランスファ・コントローラ(DTC)を設定します。「R8C/35C グループ DTC(チェイン転送)」(R01AN0372E)では、DTC のチェイン転送を用いて A/D コンバータの A/D 変換結果が格納されている A/D レジスタ(AD0～AD3)の値を RAM 領域へ転送します。

しかし、RL78/G14 では A/D 変換結果レジスタは 1 つしかありません。そのため、DTC 転送(ノーマルモード、チェイン転送を使用しない)を用いて、A/D 変換結果を RAM 領域へ転送します。A/D コンバータのスキャン・モードを用いて、ANI0 端子から ANI3 端子のアナログ入力電圧の A/D 変換を実行して、DTC 転送を用いて A/D 変換結果を RAM 領域へ転送します。

各端子の A / D 変換を連続して行います。各端子の A/D 変換が完了すると、A/D 変換結果は 10 ビット A/D 変換結果レジスタ(ADCR)に転送され、A/D コンバータの割り込み要求が発生します。A/D コンバータの割り込み要求を受けて、DTC が起動して 10 ビット A/D 変換結果レジスタ(ADCR)の値を RAM 領域へ転送します。ANI0 端子から ANI3 端子の A/D 変換と DTC 転送が完了すると、A/D 変換終了割り込み要求信号が発生します。

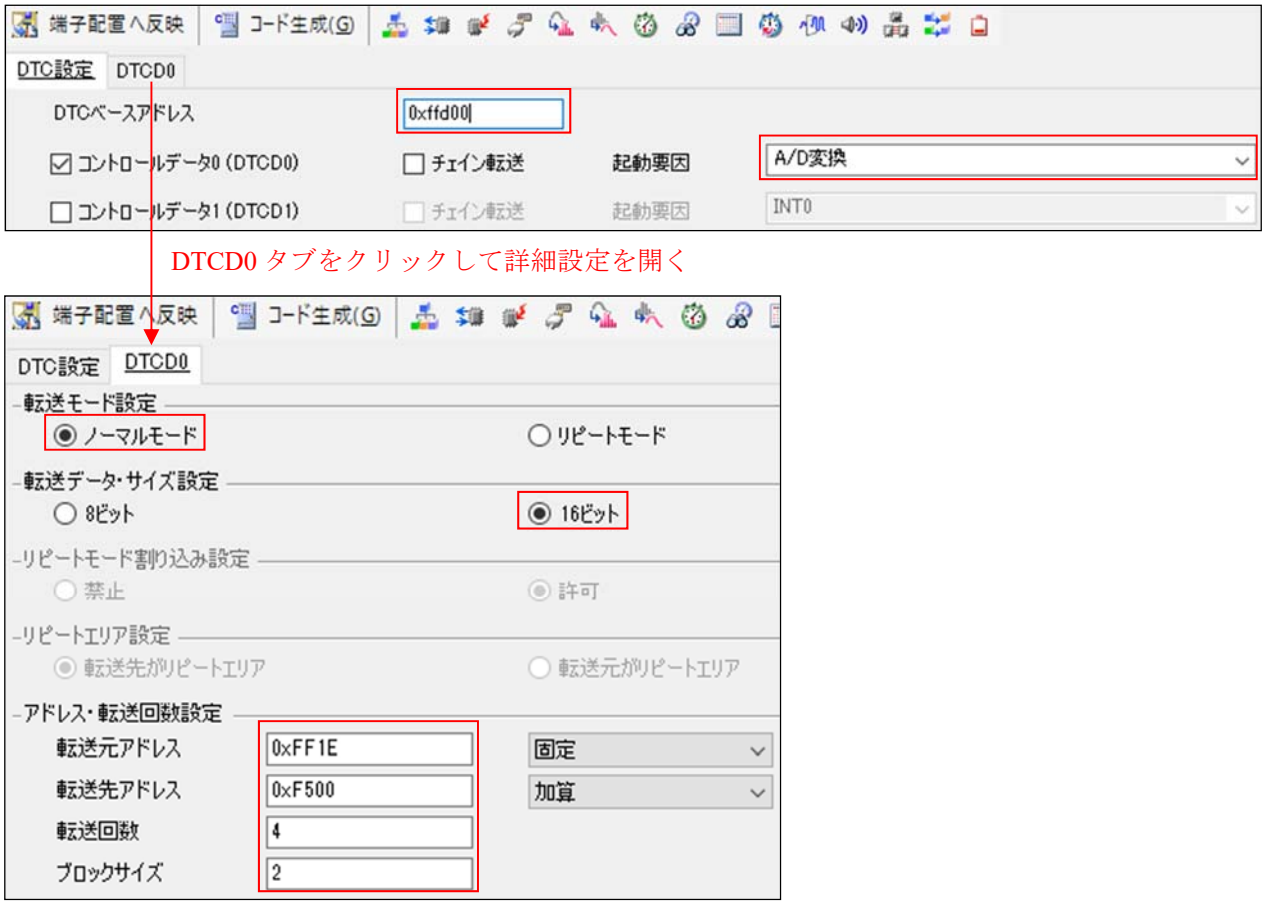

図 4.19　コード生成ツール設定画面（DTC）

(3)　その他の「ポート」、「ウォッチドッグ・タイマ」、「電圧検出回路」をそれぞれ設定します。
(4)　「コード生成(G)」をクリックし、ファイルを生成します。

4.2.3　　プログラム追加

コード生成したプログラムにシンボル定義と main 関数の処理を追加します。その他のプログラム（クロック、A/D コンバータ、DTC の設定）については、コード生成したプログラムを使用します。

・main 関数

RL78/G14 用コード生成ツールを使用すると、main 関数実行前に R_Systeminit 関数を実行します。R_Systeminit 関数では、クロックや A/D コンバータ、DTC の初期設定を行います。このため、赤枠の処理のみ手動でプログラムを追加します。R_DTCD0_Start function 関数は、DTC 起動を許可します。R_ADC_Start function 関数は、A/D コンバータの動作を開始します。

R8C 用プログラム

```
H3 void main(void)
76 {
77     /* ==== Interrupt disabled ==== */
78     asm("FCLR I");
79
80     /* ==== Set High-speed on-chip oscillator clock to System clock ==== */
81     mcu_init();
82
83     /* ==== A/D converter initialize ==== */
84     ad_converter_enable();
85
86     /* ==== DTC initialize ==== */
87     dtc_enable();
88
89     /* ==== Interrupt enabled ==== */
90     asm("FSET I");
91
92     dtcen16 = 1;                    /* Activation of A/D interruption enable */
93
94     adst = 1;                       /* A/D conversion starts */
95
96     /* ==== Main loop ==== */
97     while(1){
98     }
99 }
```

赤枠に対応する内容を手動で追記

RL78/G14 の r_main.c ファイル

```
60  void main(void)
61  {
62      R_MAIN_UserInit();
63      /* Start user code. Do not edit comment generated here */
64      R_DTCD0_Start();                    /* DTC enabled */
65      R_ADC_Set_OperationOn();
66      R_ADC_Start();                      /* A/D converter enabled */
67
68      while (1U)
69      {
70          NOP();
71      }
72      /* End user code. Do not edit comment generated here */
73  }
```

図 4.20　　main 関数の置き換え

C 言語で”while(1U)”と記述した場合、コンパイル結果によって DTC 保留命令が繰り返され DTC 転送が行われない可能性があります。DTC 保留命令が繰り返し実行されることを防ぐため、”while(1U)”文の中に”NOP()”を追加します。

4.2.4　　その他修正事項

コード生成ツールで A/D コンバータを設定すると、割り込み関数が自動生成されます。割り込み関数内に処理を追加します。

R8C 用プログラム

```
/* **** Global Variables **** */
#pragma ADDRESS ad_value    00600H      /* Destination address of A/D conversion data */
unsigned short ad_value[4];             /* A/D data from AN8 to AN11 addressed from */
                                        /* 0x0600 to 0x0607 */

unsigned short an8_value;               /* AN8 value */
unsigned short an9_value;               /* AN9 value */
unsigned short an10_value;              /* AN10 value */
unsigned short an11_value;              /* AN11 value */
```

赤枠に対応する内容を手動で追記

RL78/G14 の r_cg_adc_user.c ファイル

```
#pragma address ad_value=0xFF500
unsigned short ad_value[4];             /* A/D data from ANI0 to ANI3 addressed from */
                                        /* 0xFF500 to 0xFF507 */

unsigned short an0_value;               /* ANI0 value */
unsigned short an1_value;               /* ANI1 value */
unsigned short an2_value;               /* ANI2 value */
unsigned short an3_value;               /* ANI3 value */
```

図 4.21　変数の追加

R8C 用プログラム

```
H8 void ad_converter_int(void)
263 {
264     /* ==== Setting A/D conversion data ==== */
265     an8_value = (ad_value[0]&0x03FF);   /* Setting AN8 value */
266     an9_value = (ad_value[1]&0x03FF);   /* Setting AN9 value */
267     an10_value = (ad_value[2]&0x03FF);  /* Setting AN10 value */
268     an11_value = (ad_value[3]&0x03FF);  /* Setting AN11 value */
269
270     /* ==== A/D converstion starts ==== */
271     dtcct0 = 1;                         /* One-time is set to transfer */
272     dtcct1 = 1;                         /* One-time is set to transfer */
273     dtcen16 = 1;                        /* Activation of A/D interruption enable */
274     adst = 1;                           /* A/D conversion starts */
275 }
```

赤枠に対応する内容を手動で追記

RL78/G14 の r_cg_adc_user.c ファイル

```
57      static void __near r_adc_interrupt(void)
58      {
59          /* Start user code. Do not edit comment generated here */
60          an0_value = (ad_value[0U] & 0xFFC0U) >> 6U;
61          an1_value = (ad_value[1U] & 0xFFC0U) >> 6U;
62          an2_value = (ad_value[2U] & 0xFFC0U) >> 6U;
63          an3_value = (ad_value[3U] & 0xFFC0U) >> 6U;
64          /* ==== A/D converstion starts ==== */
65          R_DTC_Create();
66          R_DTCD0_Start();            /* DTC enabled */
67          R_ADC_Start();              /* A/D converter enabled */
68          /* End user code. Do not edit comment generated here */
69      }
```

図 4.22　割り込み処理の追加

4.2.5　　置き換え後のサンプルコード

ルネサス エレクトロニクスホームページからサンプルコード「an-r01an3508je0100-rl78-migrate.zip」を入手してください。「workspace」フォルダ内の「rl78g14_migrate_dtc」が「R8C/35C グループ DTC(チェイン転送)」(R01AN0372J)を置き換えたサンプルコードとなります。

4.3　サンプル・プログラムの動作確認条件

置き換え後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 4.1 動作確認条件

項目	内容
使用マイコン	RL78/G14 (R5F104PJ)
統合開発環境 (CS+)	ルネサス エレクトロニクス製 CS+ for CC V4.01.00
C コンパイラ (CS+)	ルネサス エレクトロニクス製 CC-RL V1.03.00
統合開発環境 (e^2 studio)	ルネサス エレクトロニクス製 e^2 studio V5.2.0.020
C コンパイラ (e^2 studio)	ルネサス エレクトロニクス製 CC-RL V1.03.00
使用ボード	ルネサス エレクトロニクス製 オリジナルボード

5.　サンプルコード

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。

6.　参考ドキュメント

ユーザーズマニュアル

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

RL78 コンパイラ CC-RL ユーザーズマニュアル(R20UT3123J)

CS+コード生成ツール 統合開発環境ユーザーズマニュアル 周辺機能操作編(R20UT3104J)

CcnvNC30 C ソースコンバータ ユーザーズマニュアル(R20UT3685J)

アプリケーションノート

R8C/35A グループ　RTC を使用した時計動作(R01AN0079J)

R8C/35C グループ　DTC(チェイン転送) (R01AN0372J)

（最新版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<website and support,ws>

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	R8C ファミリから RL78 ファミリへの置き換えガイド(CcnvNC30)

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017.01.20	ー	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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部ROM、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、ノイズ輻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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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、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、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製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。
5. 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、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、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できません。たとえ、意図し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。
6. 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。
7. 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。
8. 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。
9. 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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   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。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
11. 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。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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その総株主の議決権の過半数を直接または間接に保有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.3.0-1　2016.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/