Renesas e² studio

スマート・コンフィグレータ Application Examples: Ethernet

R20AN0495JS0100
Rev.1.00
2018.03.27

要旨

スマート・コンフィグレータ（SC）は、コード生成機能と、ドライバ、ミドルウェア、端子の設定機能を持つ、GUI ベースのツールです。スマート・コンフィグレータはルネサスの MCU ファミリそれぞれに適したコードを生成し、FIT モジュールによって生成されたコードをインポートする機能を持ちます。

このアプリケーションノートは、ユーザが e² studio 上のスマート・コンフィグレータを使用して、Ethernet コンポーネントを設定しコードを生成するためのガイドです。

ホスト PC の OS は以下をサポートします。

- Windows 7　32 ビット/64 ビット
- Windows 8.1　32 ビット/64 ビット
- Windows 10　32 ビット/64 ビット

対象デバイス

- RX64M グループ
- RX65N グループ

ソフトウェアコンポーネント

スマート・コンフィグレータは、2 種類のソフトウェアコンポーネント（コード生成（CG）と、Firmware Integration Technology (FIT)）に対応します。それぞれのソフトウェアが対応するドライバとミドルウェアは、以下のとおりです。

- ベーシックドライバ：CG ドライバ（CMT、A/D コンバータ、SCI など）、
  FIT モジュール（CMT、DTC、DMAC、RSPI、SCIFA など）
- ミドルウェア：FIT モジュール（USB、Ethernet、フラッシュメモリ（内蔵フラッシュメモリ書き換え）など）

ベーシックドライバは、CMT や A/D コンバータ、SCI などマイコンの周辺機能の制御プログラムで、コード生成（CG）機能を使ったソフトウェアコンポーネント（CG ドライバ）の組み込みが便利です。また、USB や Ethernet、フラッシュメモリ（内蔵フラッシュメモリ書き換え）などのミドルウェアを含んだ FIT モジュールをソフトウェアコンポーネントとして組み込むことが可能です。

略称一覧：

CMT: Compare Match Timer

DTC: Data Transfer Controller

DMAC: Direct Memory Access Controller

RSPI: Serial Peripheral Interface

SCIFA: FIFO Embedded Serial Communications Interface

目次

1. 概要

1.1 本ドキュメントの目的

スマート・コンフィグレータで、Ethernet FIT モジュールを使用したエコーサーバプログラムを作成する方法をガイドします。

1.2 動作環境

対象デバイス	RX64M グループ RX65N、RX651 グループ
評価ボード	RX65N-2MB 用 Renesas Starter Kit+ (RX65N R5F565NEDxFC)
デバッガ	E1 /E2 Lite
IDE	e^2 studio V6.2.0 以降
ツールチェーン	RX ファミリ用ルネサス C/C++コンパイラパッケージ

1.3　スマート・コンフィグレータプロジェクトの基本操作フロー

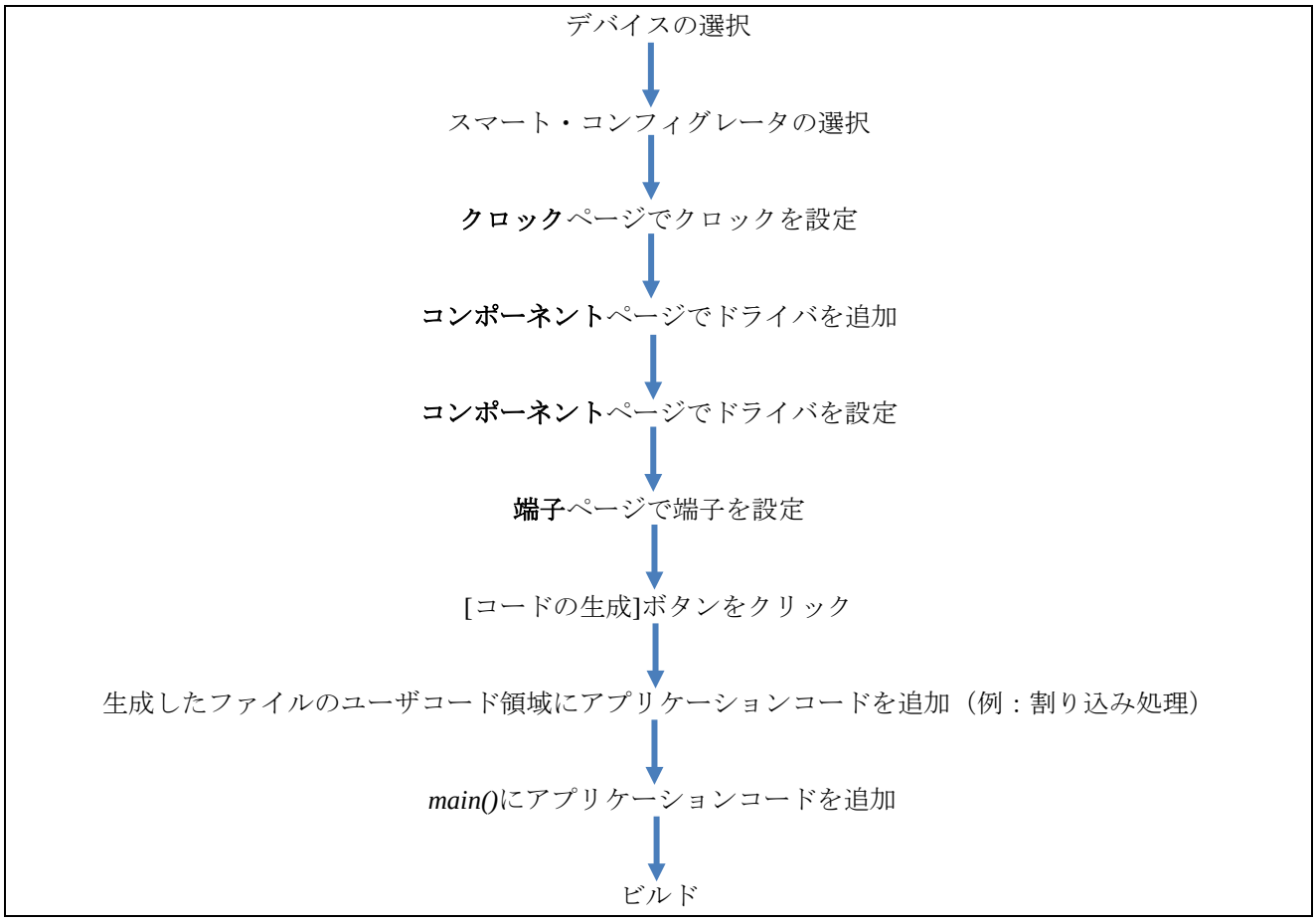

図 1-1 基本操作

スマート・コンフィグレータの詳細な操作については、“スマート・コンフィグレータ　ユーザーガイド”を参照してください。

1.4　モジュール構造

この章では、エコーサーバサンプルが使用する FIT モジュールの構造について説明します。

FIT モジュールの使用方法に関するアプリケーションノートは、各モジュールの”doc”フォルダ内のプロジェクトツリーにあります。たとえば、Ethernet ドライバのアプリケーションノート R01AN2009 は、\src\smc_gen\r_ether_rx\doc フォルダにあります。

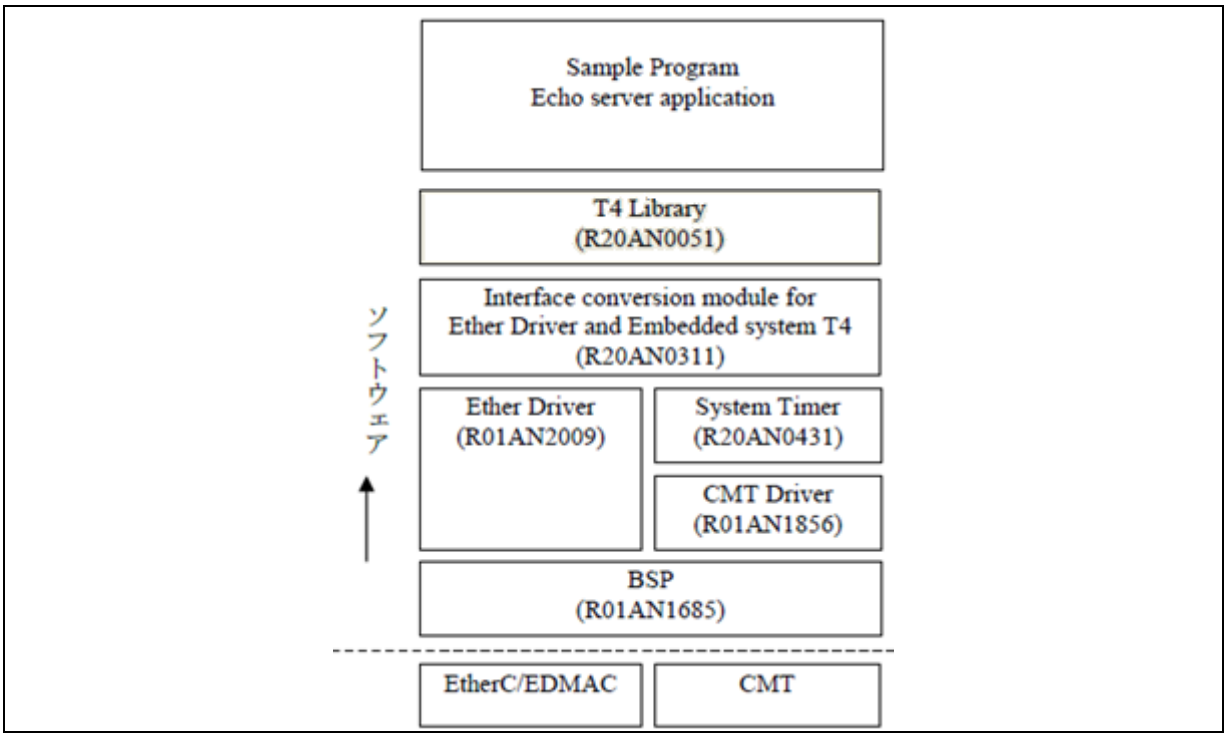

図 1-2　モジュール構造

表 1-1 に、設定される FIT モジュールを示します。

表 1-1　FIT モジュール

タイプ	モジュール	SC ソフトウェアコンポーネント名	バージョン
ミドルウェア	T4 ライブラリ（組み込み用 TCP/IP M3S-T4-Tiny）	r_t4_rx	2.07
インタフェース	Ether ドライバと組み込み用システム T4 のインタフェース変換モジュール	r_t4_driver_rx	1.06
デバイスドライバ	Ether ドライバ	r_ether_rx	1.14
ミドルウェア	システムタイマ	r_sys_time_rx	1.00
デバイスドライバ	CMT（コンペアマッチタイマ）ドライバ	r_cmt_rx	3.10
BSP	BSP（ボードサポートパッケージ）	r_bsp	3.60

1.5　Ethernet ドライバの端子設定

この Ethernet Application Example では、MII Ethernet コントロールモードを使用します。以下は、FIT を使用した Ethernet モジュールのアプリケーションノート（\src\smc_gen\r_ether_rx\doc にある R01AN2009JJ0114）からの抜粋です。

MII モードを使用する場合	RMII モードを使用する場合	I/O ポート
ET0_TX_CLK		PC4
ET0_RX_CLK	REF50CK0	P76
ET0_TX_EN	RMII0_TXD_EN	P80
ET0_ETXD3		PC6
ET0_ETXD2		PC5
ET0_ETXD1	RMII0_TXD1	P82
ET0_ETXD0	RMII0_TXD0	P81
ET0_TX_ER		PC3
ET0_RX_DV		PC2
ET0_ERXD3		PC0
ET0_ERXD2		PC1
ET0_ERXD1	RMII0_RXD1	P74
ET0_ERXD0	RMII0_RXD0	P75
ET0_RX_ER	RMII0_RX_ER	P77
ET0_CRS	RMII0_CRS_DV	P83
ET0_COL		PC7
ET0_MDC		P72
ET0_MDIO		P71
ET0_LINKSTA		P34*1
ET0_EXOUT		-*2
ET0_WOL		-*2

【注】 *1 ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は設定不要です。
【注】 *2 イーサネット FIT モジュールでは使用しない端子なので設定不要です。

図 1-3　MII Ethernet コントロールモードでの端子使用方法

対応する端子を、以下の RX65N-2MB 用 Renesas Starter Kit+の回路図に示します。

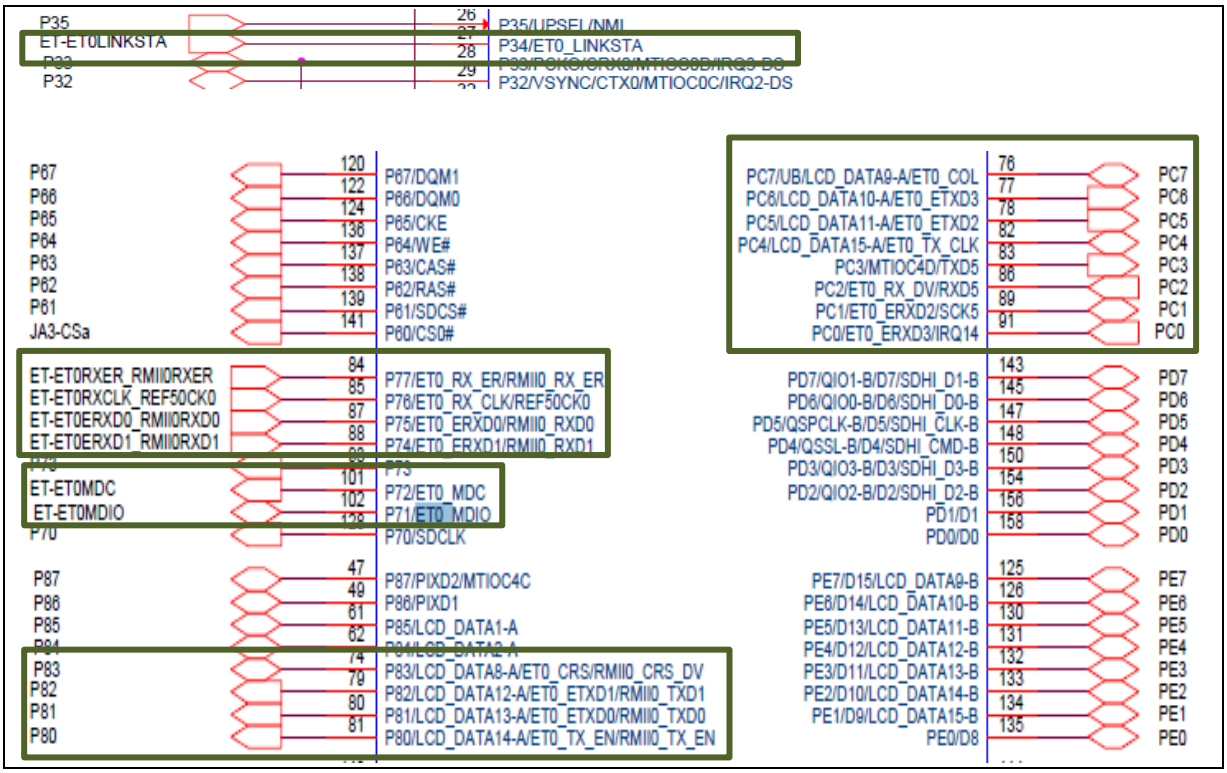

図 1-4　RX65N-2MB 用 Renesas Starter Kit+における MII Ethernet 端子の配置

RX65N-2MB 用 Renesas Starter Kit++の端子を MII Ethernet コントロールモードで動作させるため、表 1-2 で示すとおりに設定してください。この端子の配置は、2.5 章のコンポーネントの端子設定で行います。

表 1-2　端子の配置

機能	ポート	端子番号
ET0_COL	PC7	76
ET0_CRS	P83	74
ET0_ERXD0	P75	87
ET0_ERXD1	P74	88
ET0_ERXD2	PC1	89
ET0_ERXD3	PC0	91
ET0_ETXD0	P81	80
ET0_ETXD1	P82	79
ET0_ETXD2	PC5	78
ET0_ETXD3	PC6	77
ET0_LINKSTA	P34	27
ET0_MDC	P72	101
ET0_MDIO	P71	102
ET0_RX_CLK	P76	85
ET0_RX_DV	PC2	86
ET0_RX_ER	P77	84
ET0_TX_CLK	PC4	82
ET0_TX_EN	P80	81
ET0_TX_ER	PC3	83

1.6　メイン・クロック・ソース

下記の回路図では、RX65N メインクロックは 24MHz の水晶発振子に接続されています。このクロックは、3.3 章のデバッグ構成設定で、メイン・クロック・ソースとして設定されます。

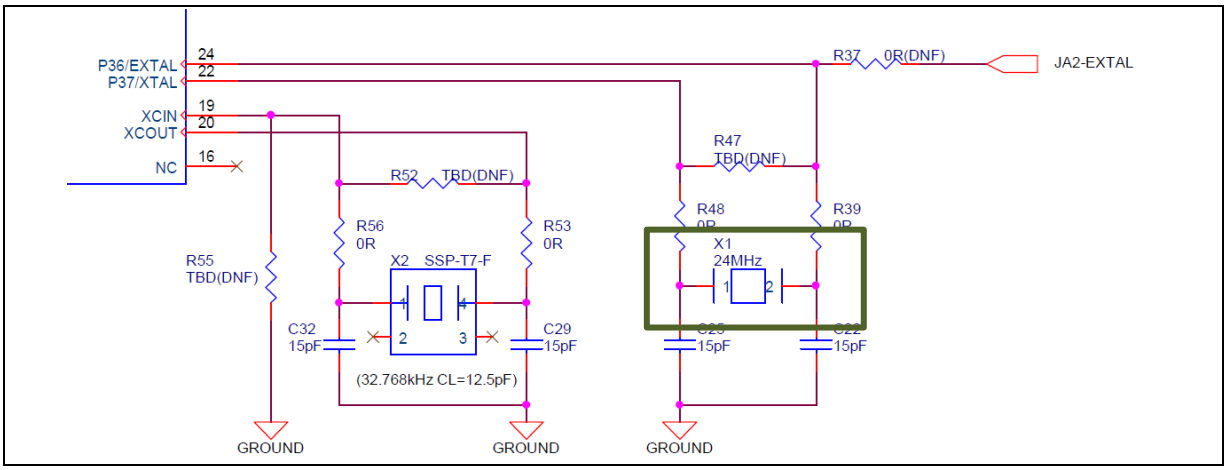

図 1-5　RX65N-2MB 用 Renesas Starter Kit+の回路図

2. Application Example（スマート・コンフィグレータを使用した Ethernet プログラム作成）

2.1 プログラム作業フローチャート

プログラムフローチャートは、以下の通りです。

a) main 関数：

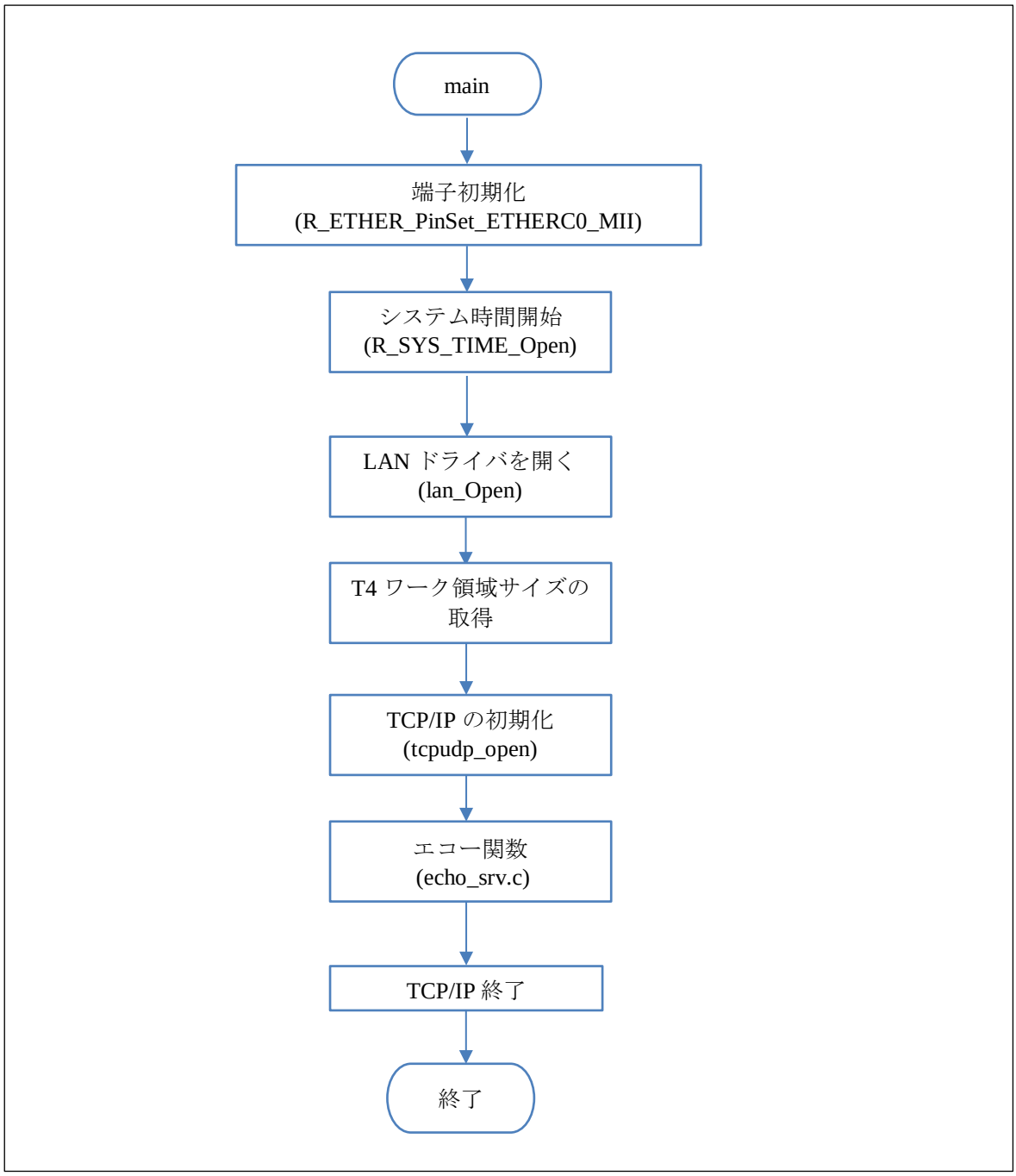

図 2-1　main フローチャート

b)　echo_srv 関数：

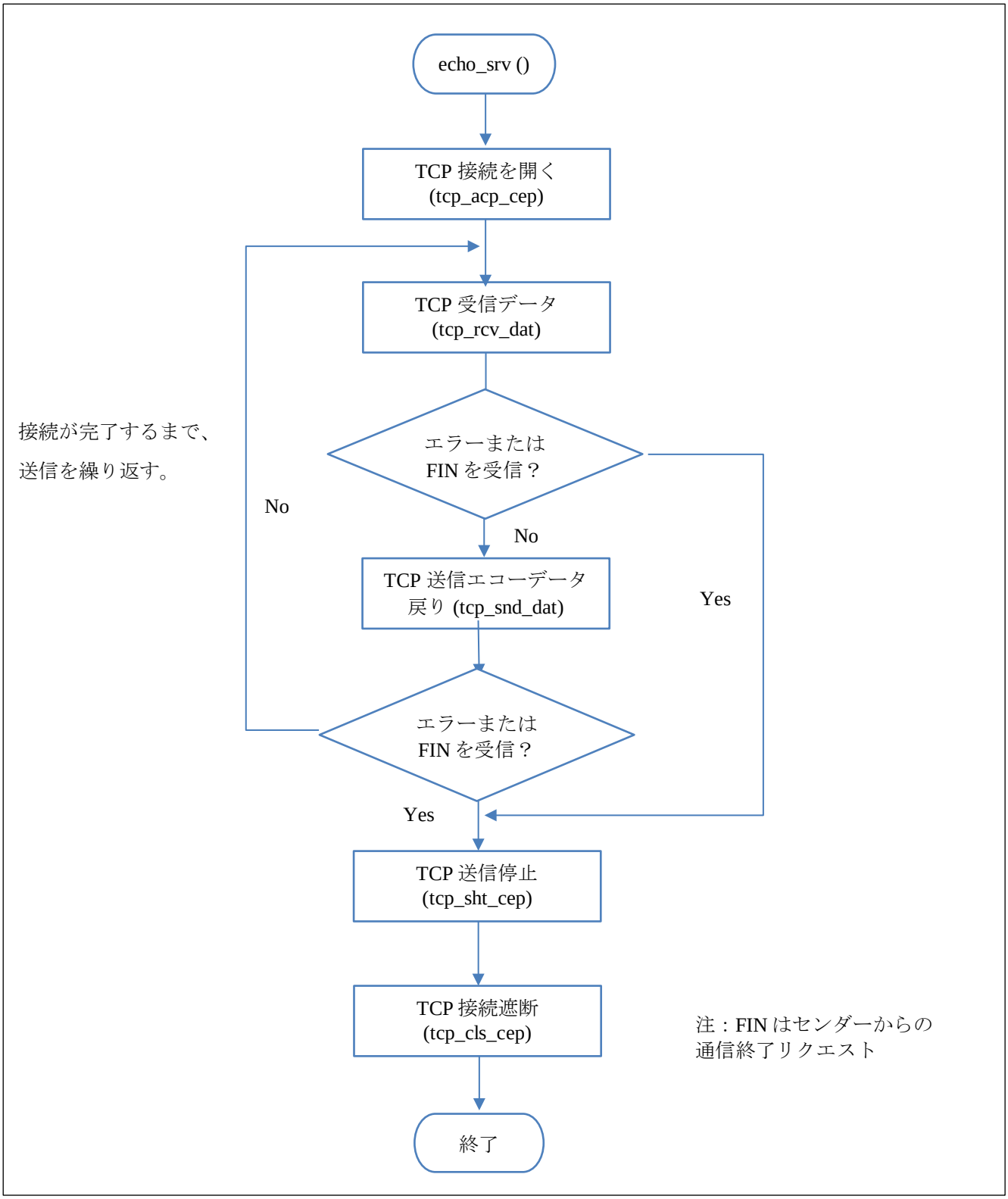

図 2-2　エコー関数フローチャート

2.2　ワークスペースの作成

1) Windows®スタートメニューから、e²studio を起動します。デフォルトのワークスペースフォルダを使用し、[OK]をクリックします。

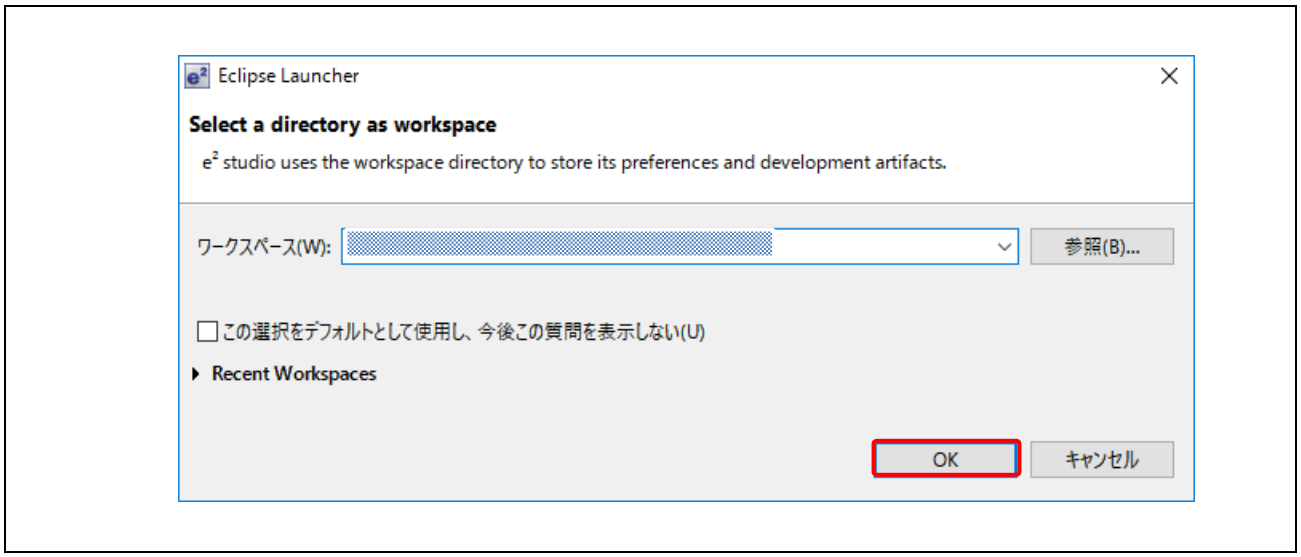

図 2-3　ワークスペースランチャー

2.3　プロジェクトの作成

1) e²studio で、新しい C プロジェクトを作成します。

[ファイル] → [新規] → [C/C++Project] を開き、新しいプロジェクトを作成します。

図 2-4　ファイルメニューからプロジェクトを作成

2)　[Renesas RX] → [Renesas CC-RX C/C++Executable Project]を選択し、[次へ]をクリックします。

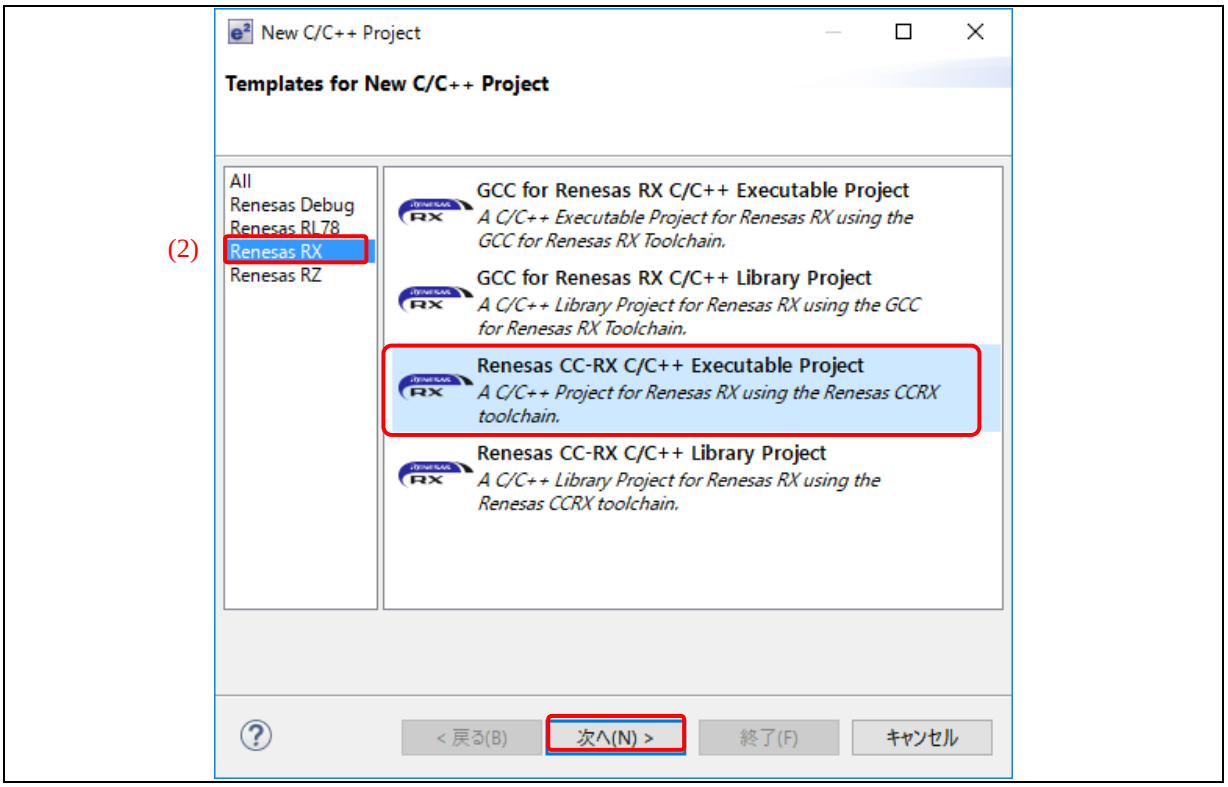

図 2-5　ファイルメニューからプロジェクトを作成

3)　プロジェクト名を入力し(ここでは“Smart_Configurator_Example”とします)、[次へ]に進みます。

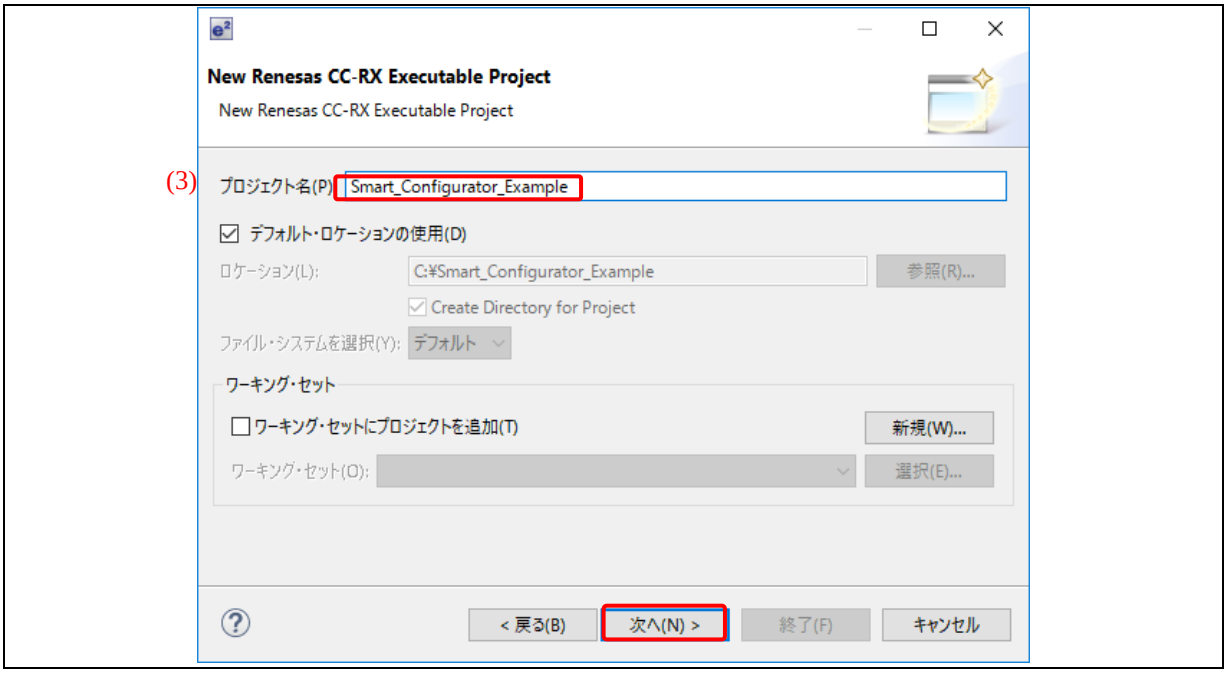

図 2-6　ファイルメニューからプロジェクトを作成

4) 言語は“C”を選択します。
5) ツールチェーンは“Renesas CCRX”を選択します。
6) ツールチェーン・バージョンを選択します。（例：v2.08.00）
7) ターゲット・デバイスを選択します：

   RX65N-2MB の場合 “RX600 > RX65N > RX65N - 176pin > R5F565NEDxFC”

   RX64M の場合 “RX600 > RX64M > RX64M - 176pin > R5F564MLCxFC”
8) [Hardware Debug 構成を生成]にチェックを入れ、エミュレータを選択します。（例：E1(RX)）
9) [次へ]をクリックします。

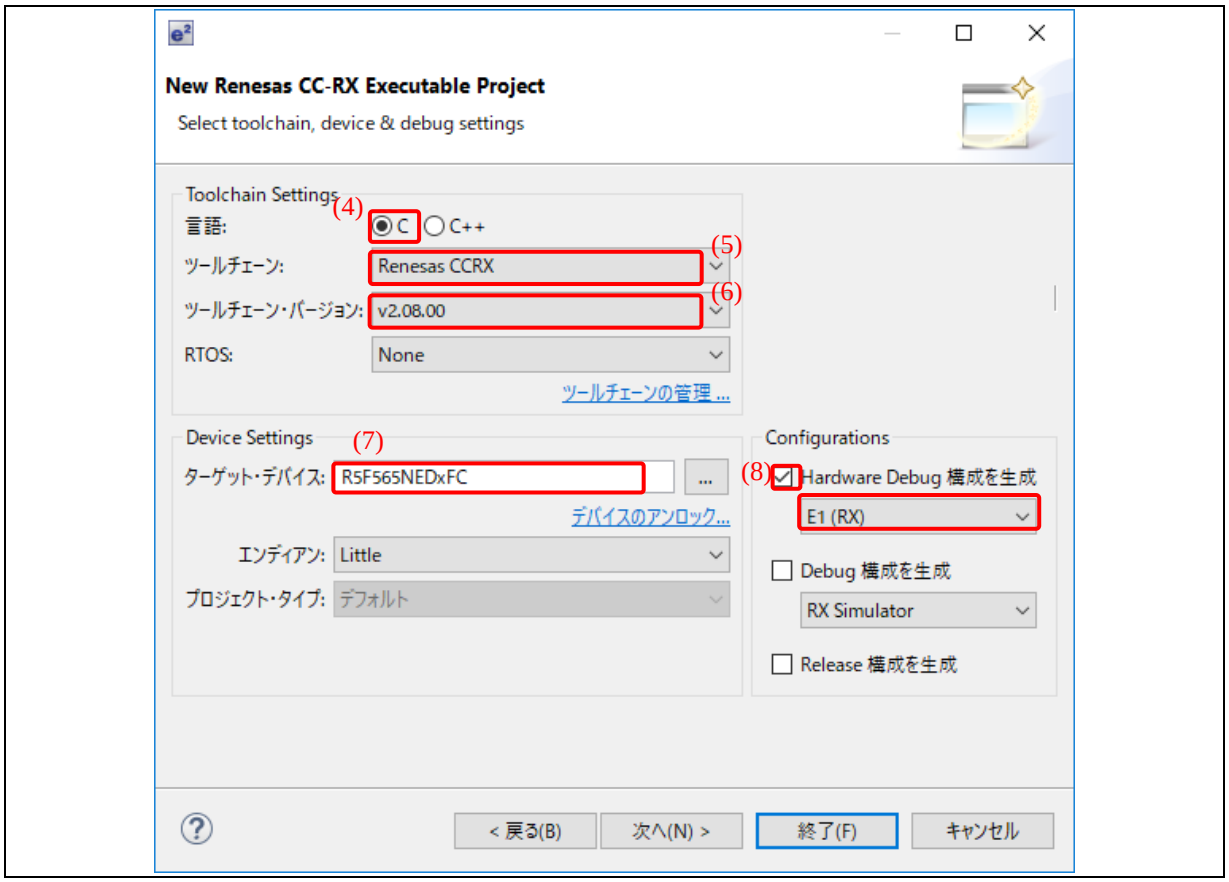

図 2-7　C プロジェクト：E1 エミュレータを使用する場合の設定例

10) “コーディング・アシストツールの選択”ダイアログボックスで、“スマート・コンフィグレータを使用する”にチェックを入れます。

11) [終了]をクリックします。

図 2-8　コーディング・アシストツールの選択

2.4　クロック設定

以下に示す手順で、スマート・コンフィグレータ・パースペクティブを開始します。

1) Smart_Configurator_Example.scfg 画面で、[クロック]ページをクリックします。

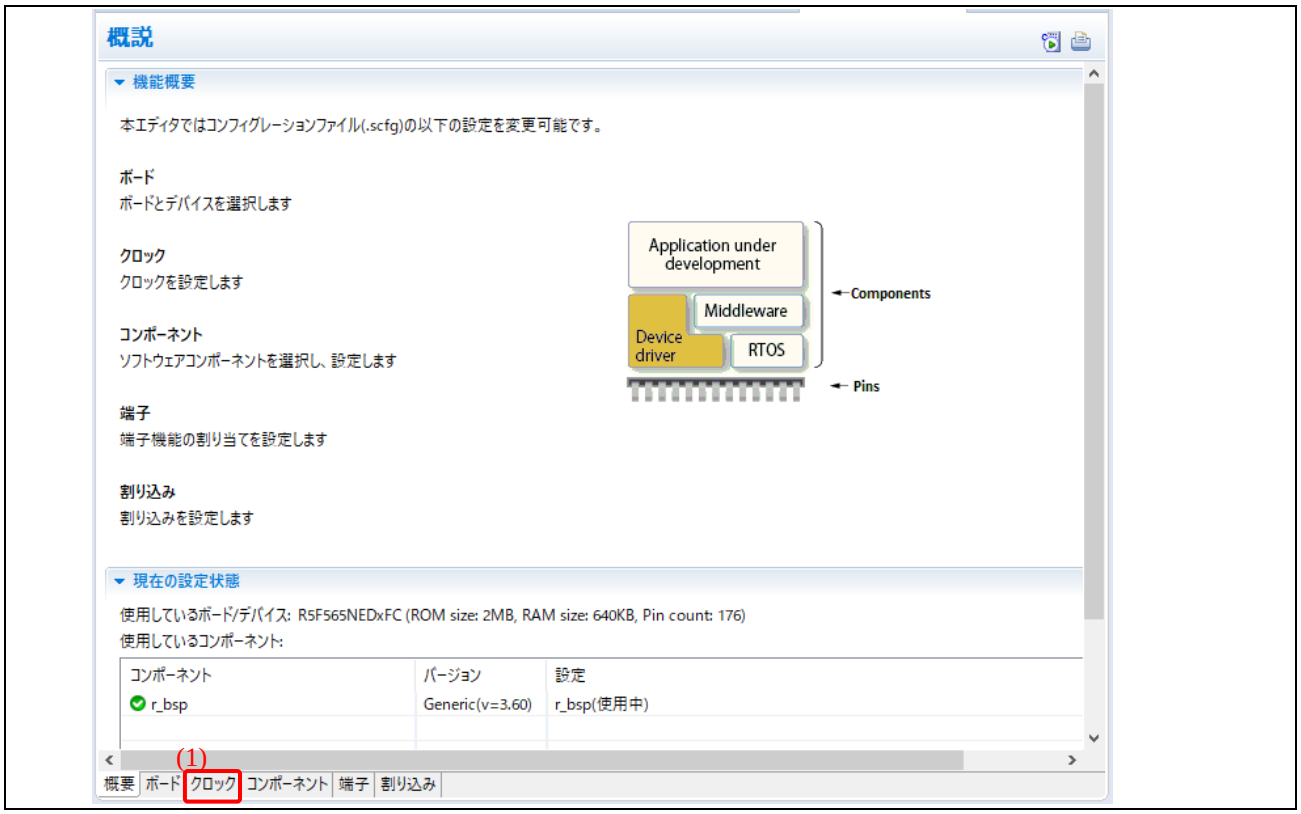

図 2-9　スマート・コンフィグレータ・パースペクティブ

2) 24MHz の水晶発振子が RX65N のメインクロックに接続されているため（1.6 章参照）、メインクロックの周波数が 24MHz に設定されていることを確認してください。他のクロック設定は、デフォルトのままにしておいてください。

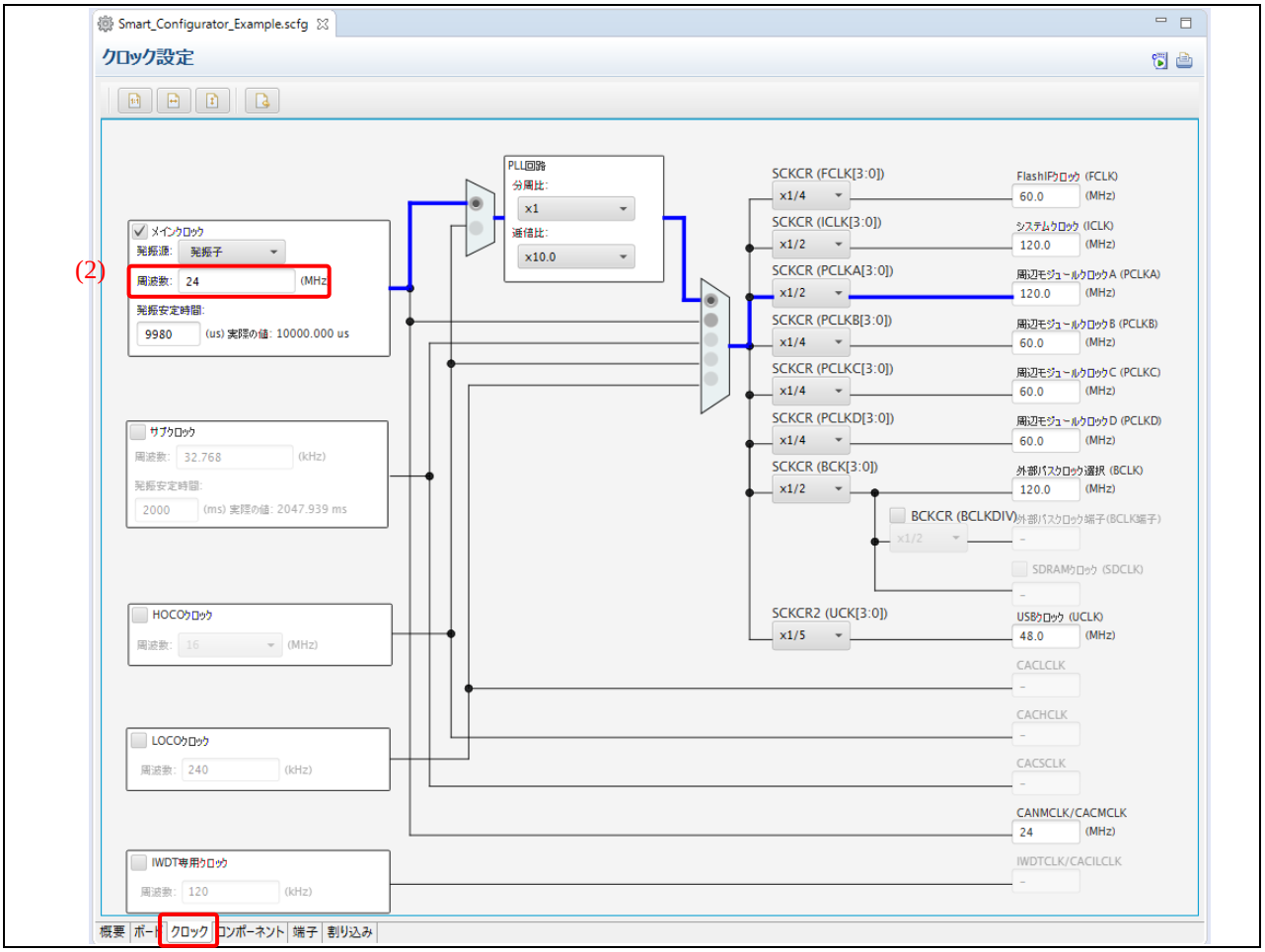

図 2-10　スマート・コンフィグレータでのクロック設定

2.5　ソフトウェアコンポーネントの追加

1)　Smart_Configurator_Example.scfg 画面で、[コンポーネント]ページをクリックしてください。

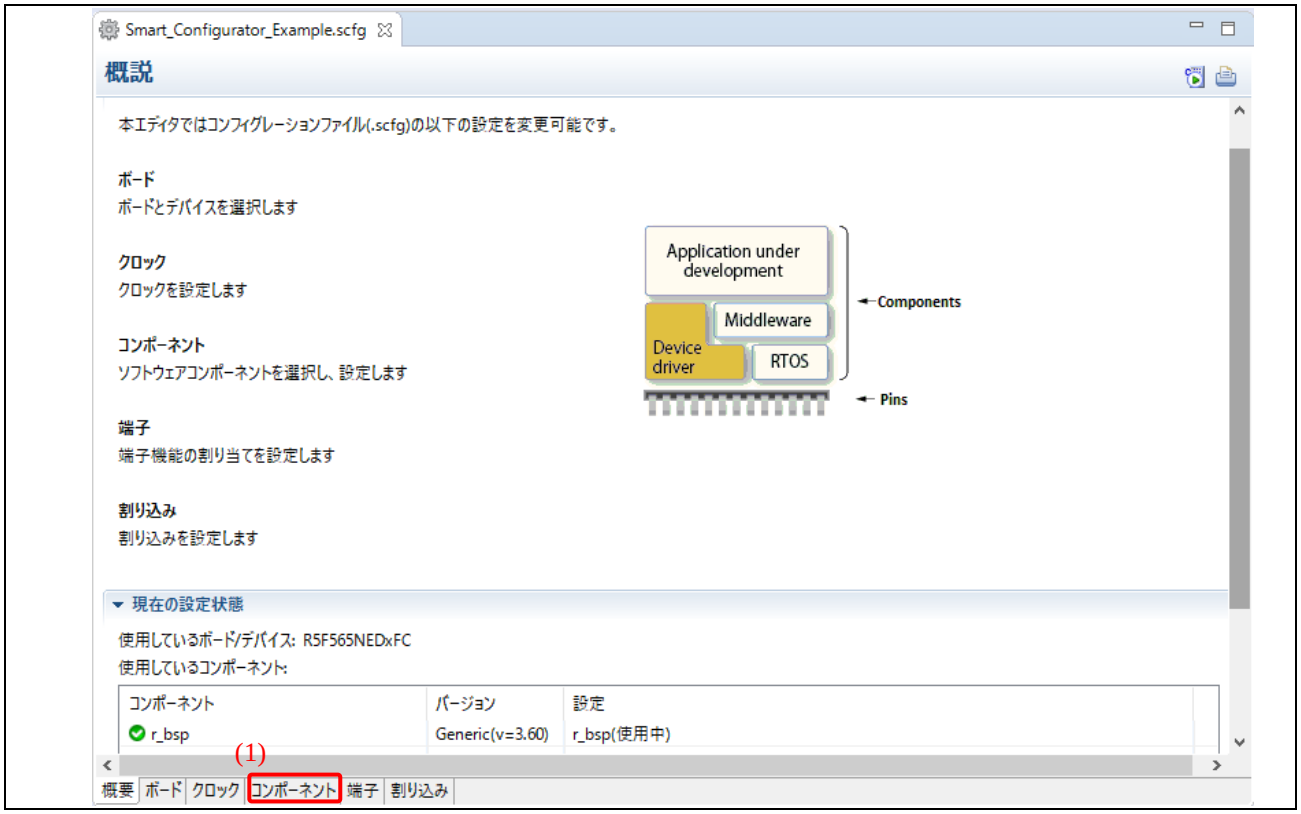

図 2-11　スマート・コンフィグレータ・パースペクティブ

2)　をクリックし、新しいコンポーネントを追加します。

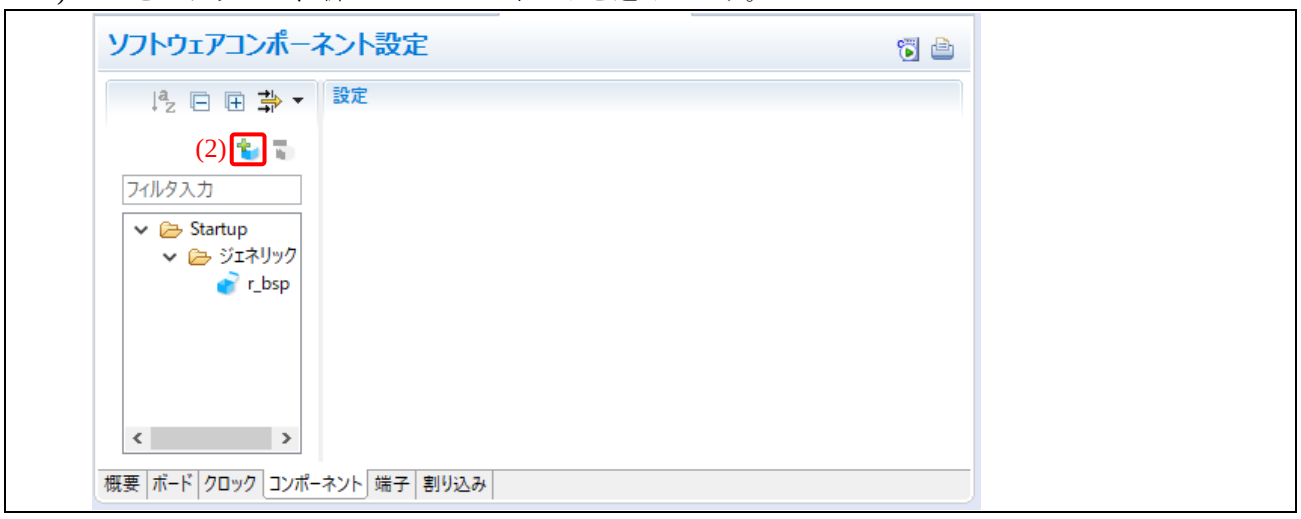

図 2-12　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) FIT モジュールをプロジェクトに追加します。
   a. コンポーネントリストを表示し、“*r_cmt_rx*”モジュールを選択します。
   b. Ctrl キーを押したまま、以下の FIT モジュールをクリックします。
      r_ether_rx
      r_sys_time_rx
      r_t4_driver_rx
      r_t4_rx
      注：これらの FIT モジュールがリストにない場合、[他のソフトウェアコンポーネントをダウンロードする]をクリックして、FIT モジュールをダウンロードしてください。
   c. [終了]をクリックします。

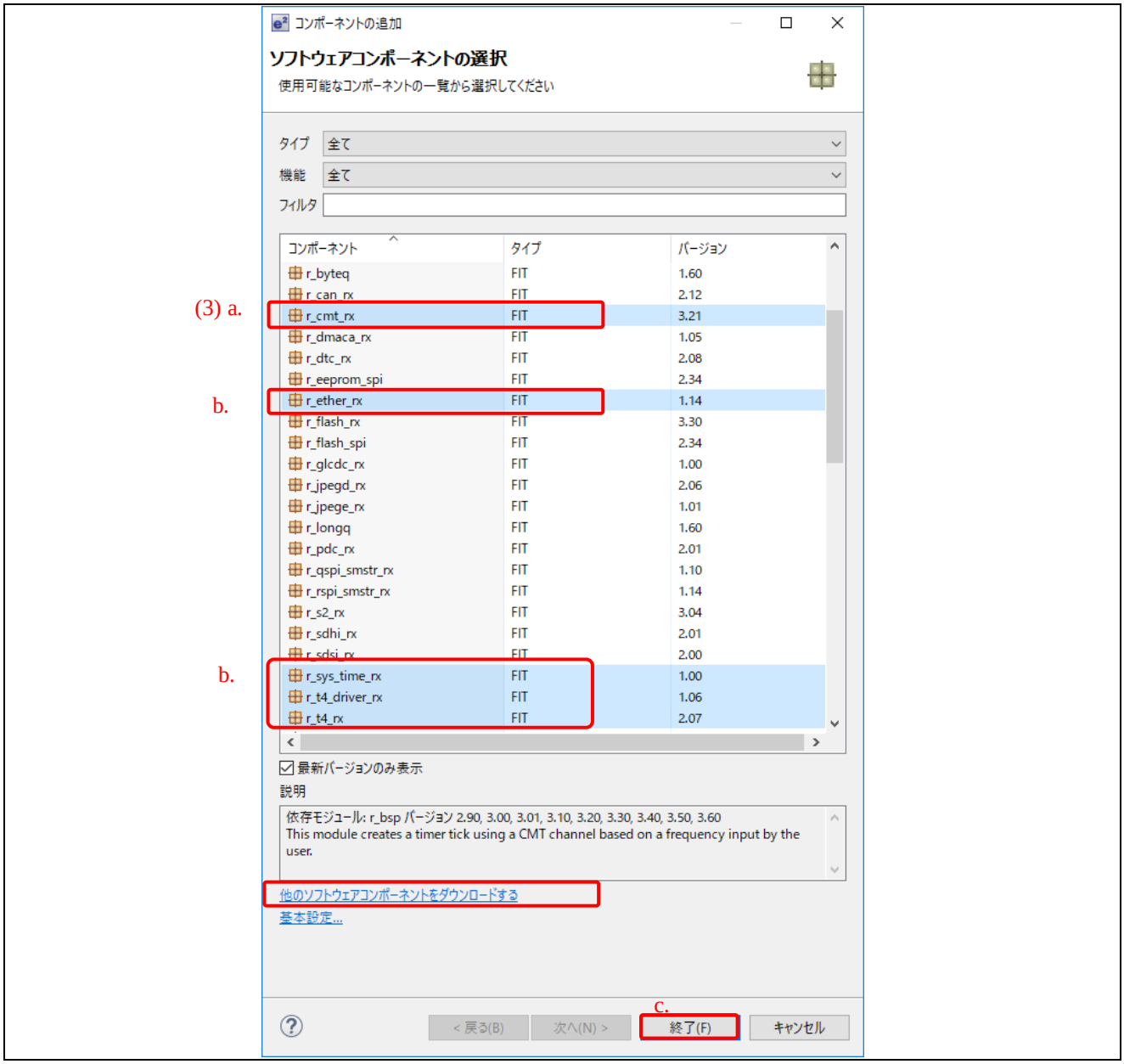

図 2-13　ソフトウェアコンポーネントの選択

4)　新しいソフトウェアコンポーネントが、[コンポーネント]ページに表示されます。

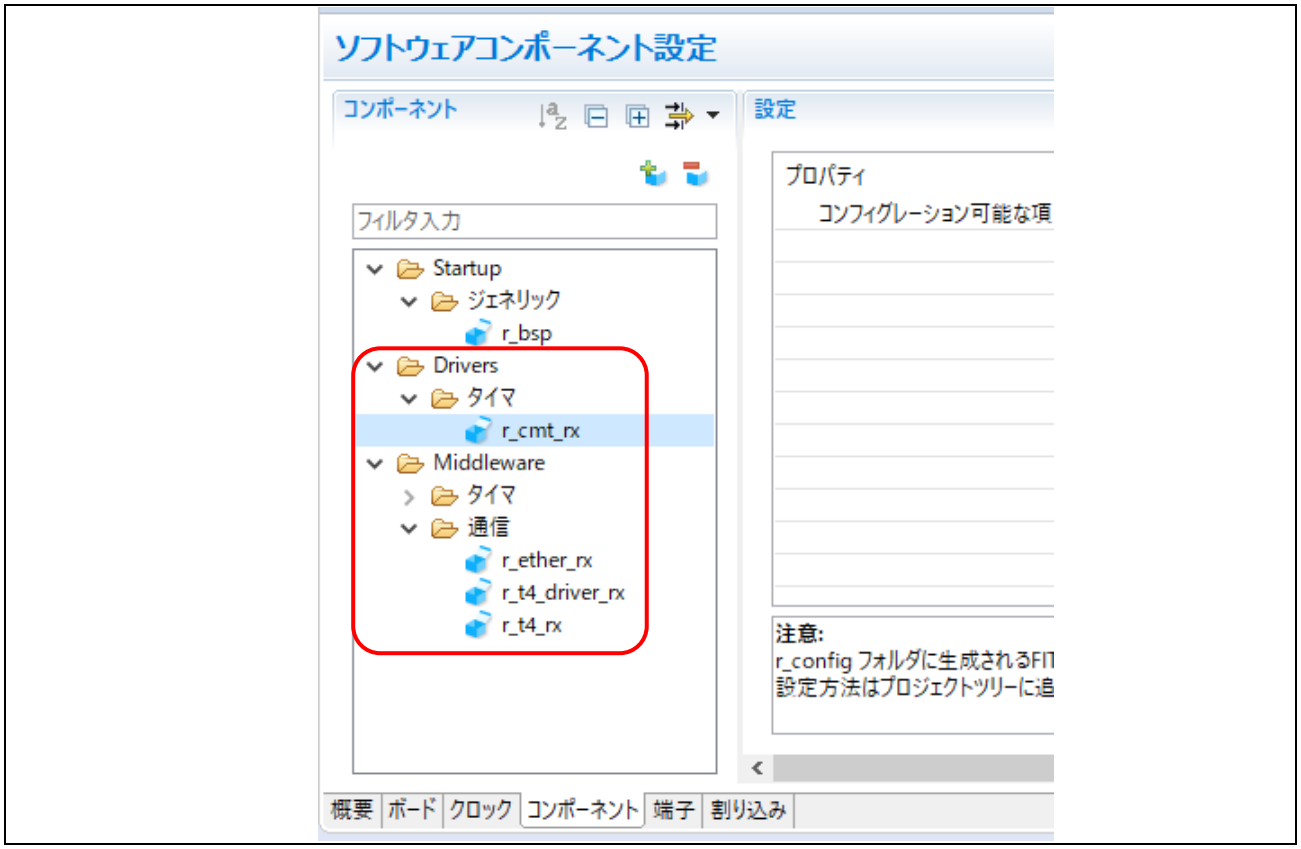

図 2-14　スマート・コンフィグレータでのソフトウェアコンポーネント設定

5) [コンポーネント]ページで、以下を設定します。
   a. r_t4_rx を選択します。
   b. Configurations にある、以下の設定を行います。
      Channel number your system has.　　1
      Enable/Disable DHCP Function.　　0
      SYSTEM callback function use　　0

注：これらの設定は、コード生成後に\src\smc_gen\r_t4_rx\src フォルダの“config_tcpudp.c”ファイルで参照することができます。

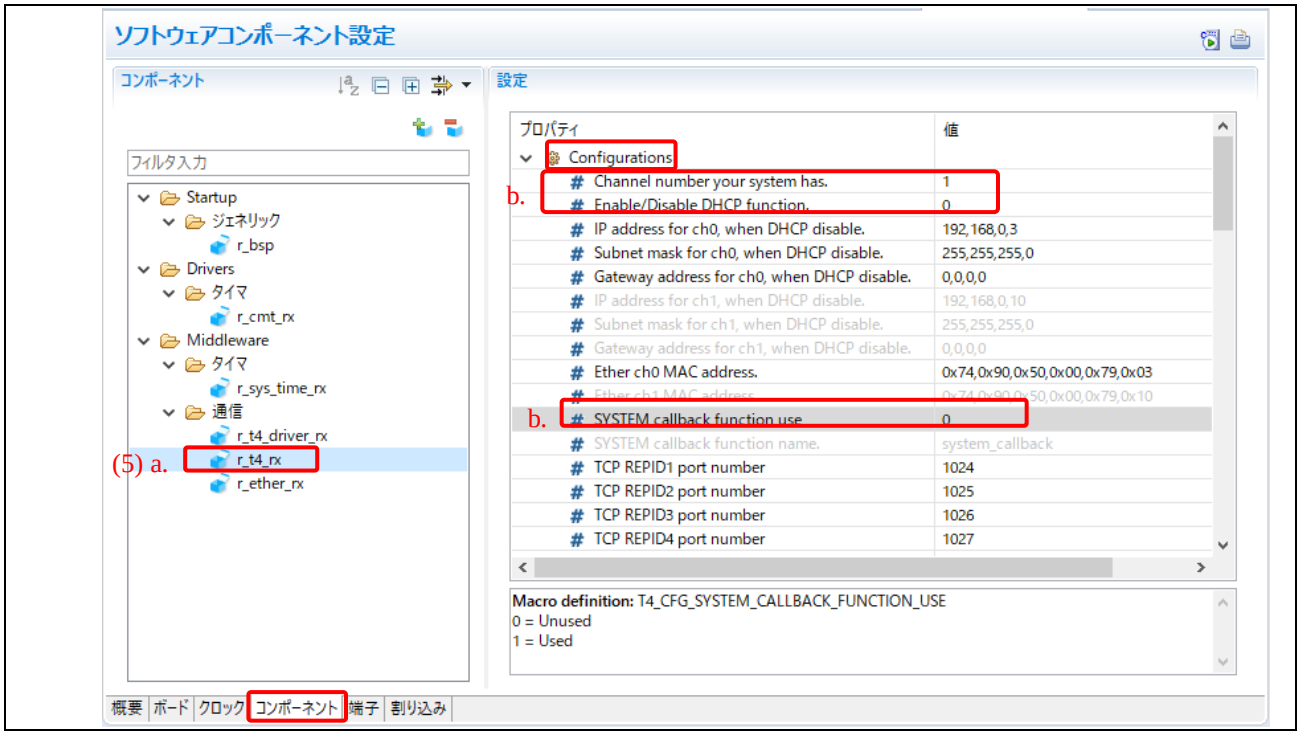

図 2-15　コンポーネント設定

6)　[コンポーネント]ページで、以下を設定します。

a.　r_ether_rx を選択します。

b.　Configurations にある、以下の設定を行います。

Ethernet interface	MII
PHY-LSI address setting for ETHER 0	30
PHY-LSI address setting for ETHER 1	1
The register bus of PHY0 for ETHER0/1	Use ETHER0
The register bus of PHY1 for ETHER0/1	Use ETHER0

注：これらの設定は、コード生成後に\src\smc_gen\r_config フォルダの“r_ether_rx_config.h”ファイルで参照することができます。

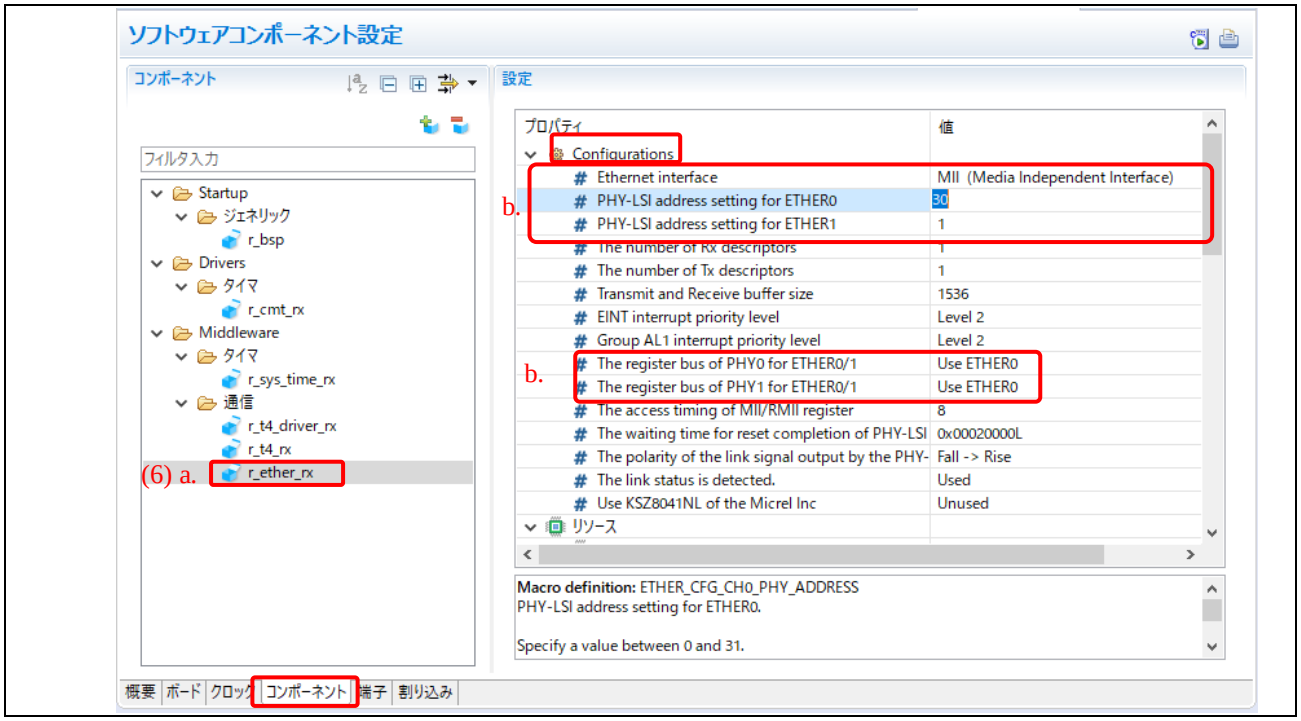

図 2-16　Ethernet 端子設定

7) [コンポーネント]ページで、以下を設定します。
   a. *r_ether_rx* を選択します。
   b. “リソース”で、以下の図のようにリソースのチェックボックスにチェックを入れてください。選択したリソースを、このサンプルコードで使用します。

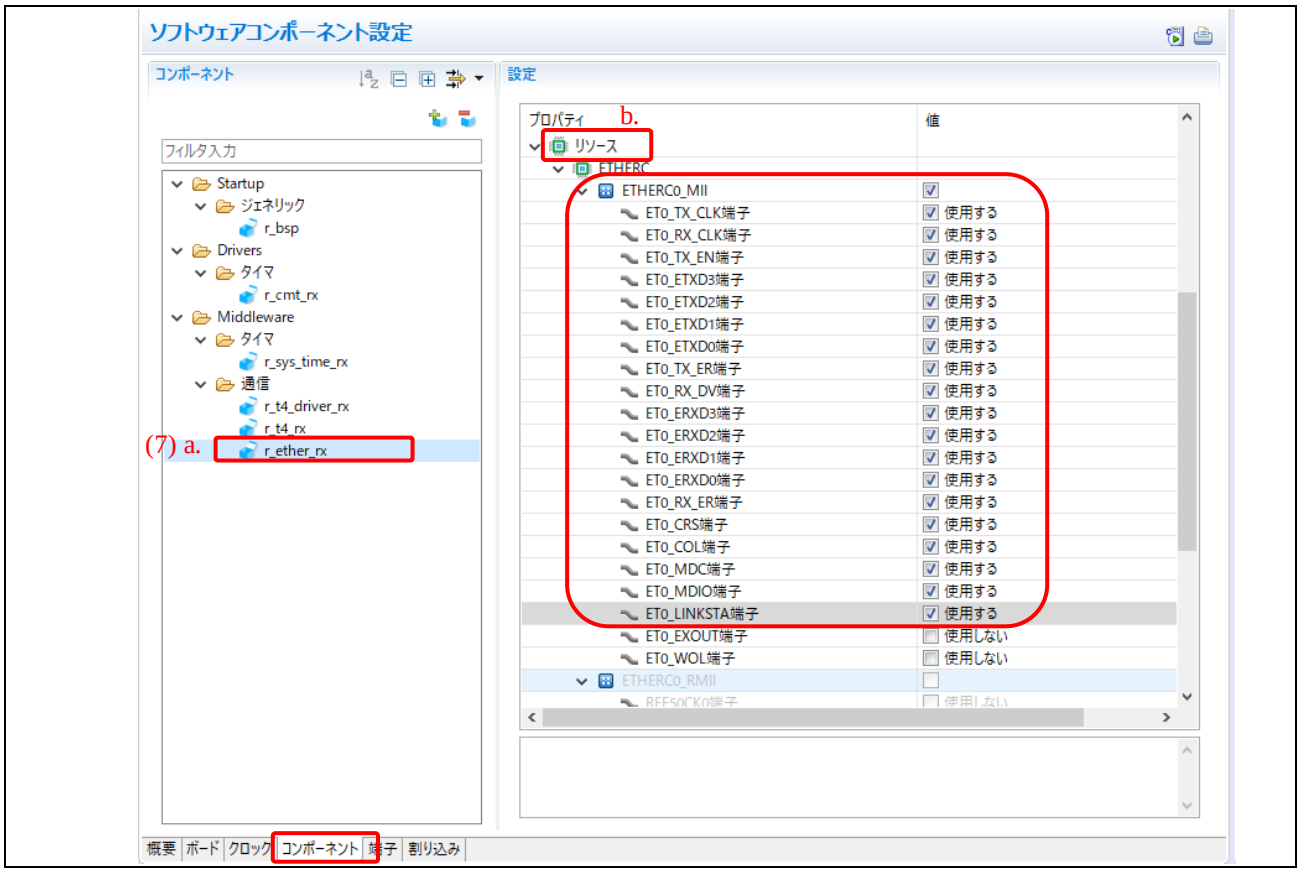

図 2-17　Ethernet 端子設定

8) [端子]ページで  ボタンをクリックし、ツリーをソフトウェアコンポーネントビューに切り替えます。

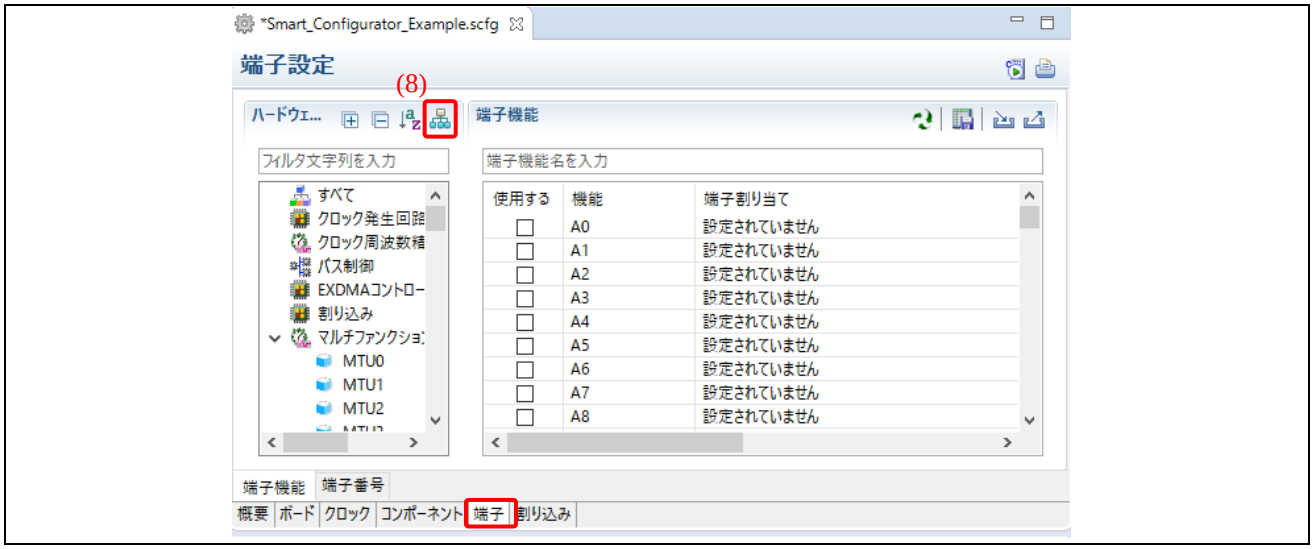

図 2-18　端子ページ

9) ツリーにある *r_ether_rx* をクリックし、端子設定を表示します。

10) 以下の機能が（1.5 章で説明したとおりに）対応する端子に割り当てられていることを、確認します。
例：[使用する]のフラグをチェックし、機能“ET0_CRS”の割り当てを PB7（デフォルト設定）から P83 に変更する

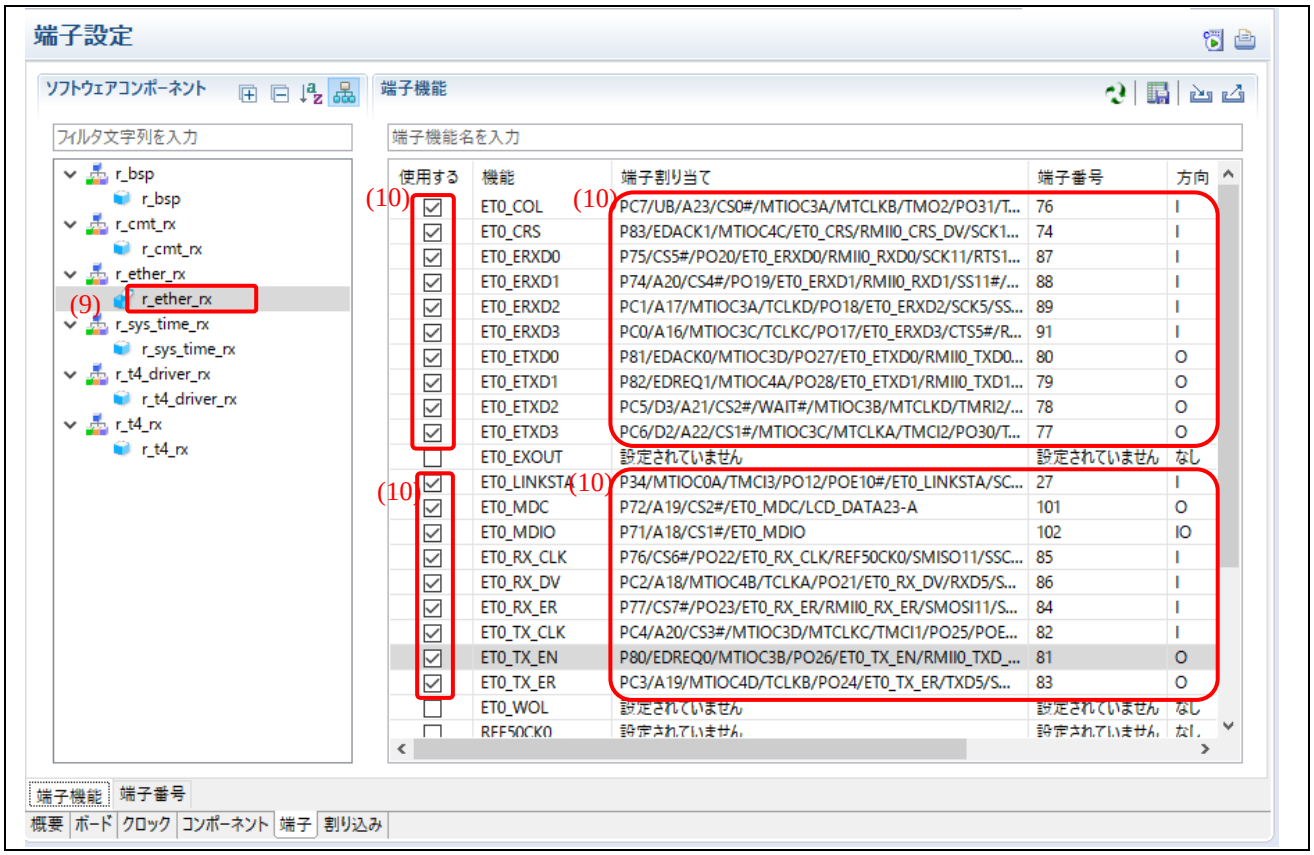

図 2-19　*r_ether_rx* の端子設定

2.6　MCU パッケージ

すべての端子設定が完了すると、以下の図に示すとおり MCU パッケージの端子配置が自動的に更新されます。

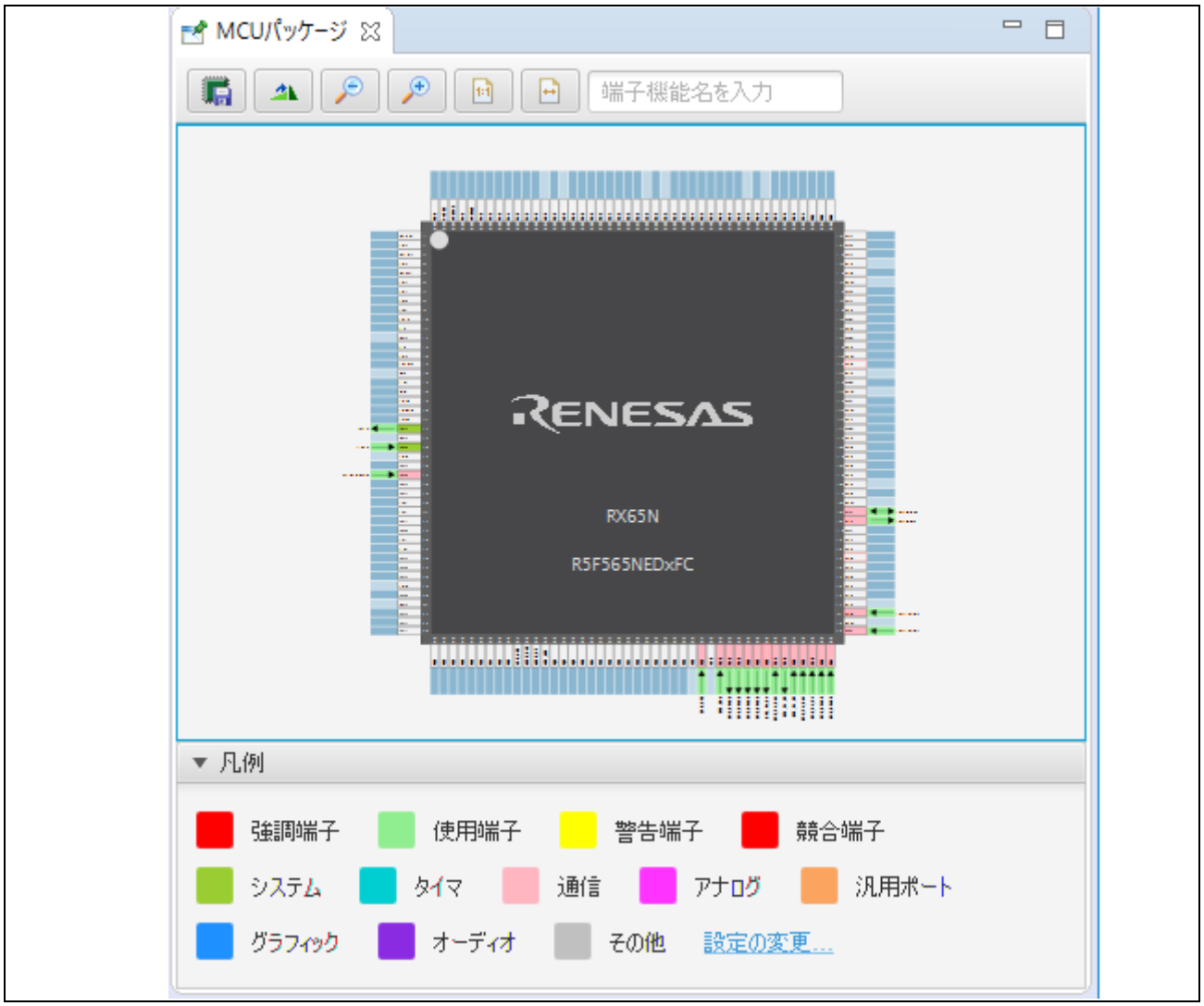

図 2-20　MCU パッケージの端子配置

2.7　コード生成

1)　  をクリックし、コードを生成します。

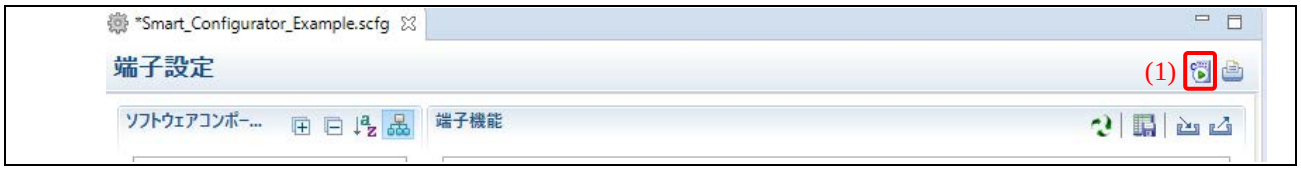

図 2-21　コード生成

2)　“コード生成の終了”というメッセージがコンソールに表示されます。

3)　プロジェクトの\src\smc_gen フォルダにファイルが生成されます。

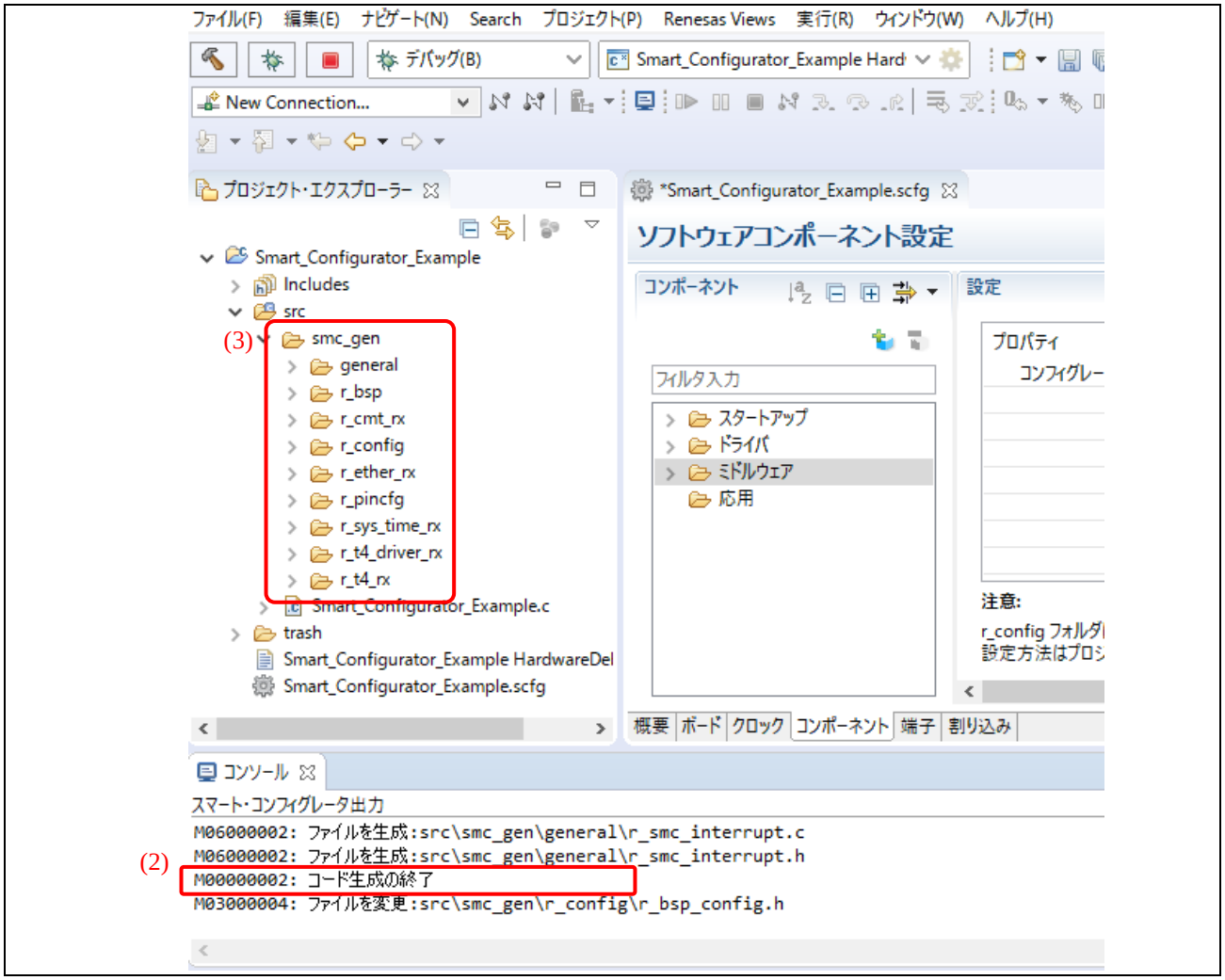

図 2-22　正常に終了したコード生成

2.8　src フォルダにソース・ファイルを追加

1)　プロジェクト・エクスプローラーのツリーで、[src]フォルダを右クリックし、[新規] → [ソース・ファイル]を選択します。

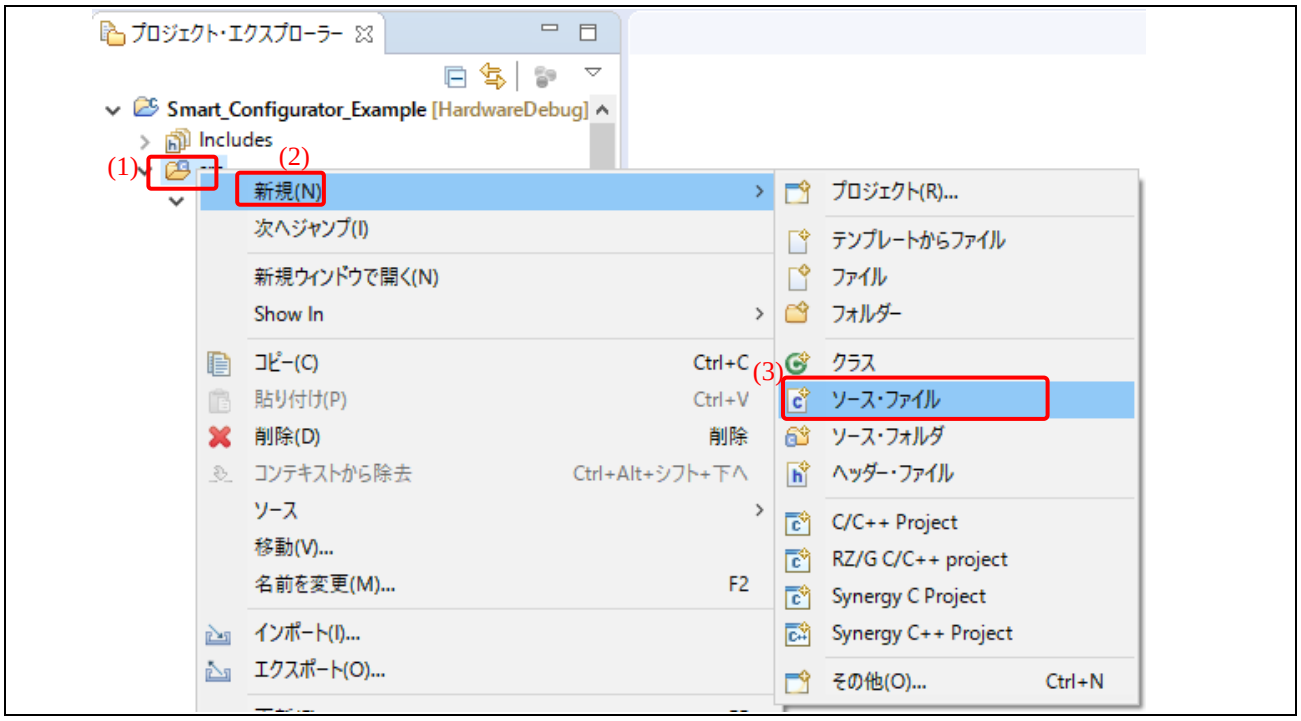

図 2-23　ソース・ファイルの追加

2)　ヘッダファイル名（例 *echo_srv.c*）を入力し、[終了]をクリックします。

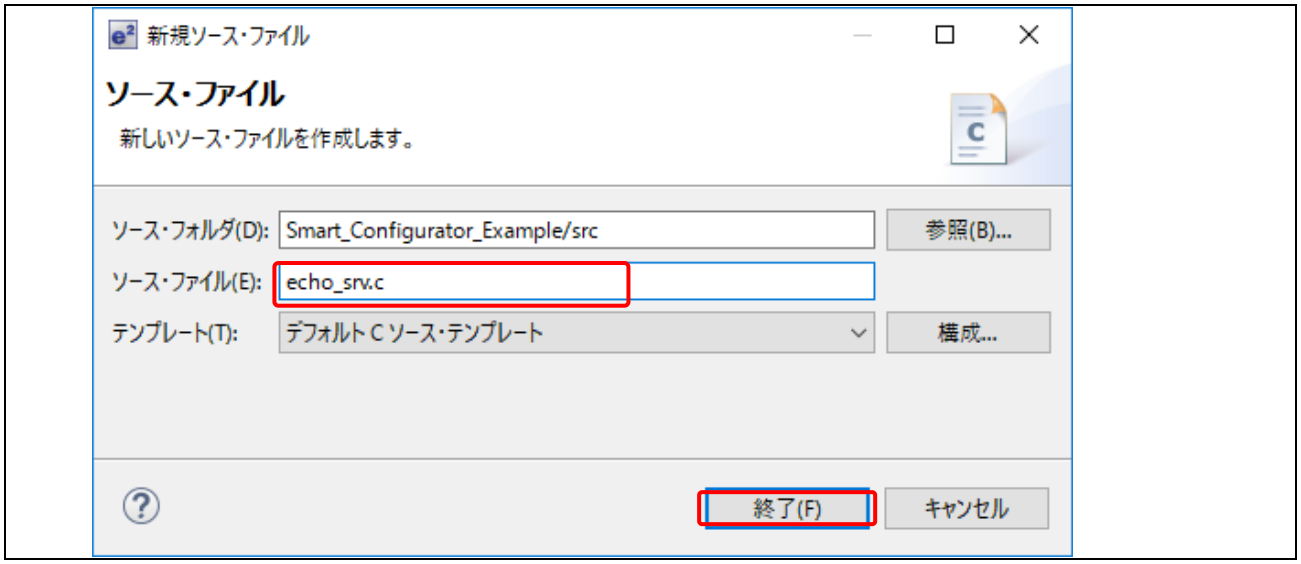

図 2-24　ソース・ファイルの追加

3) \src フォルダにある“*echo_srv.c*”ファイルを開きます。

この Application Example では、以下のコードを“*echo_srv.c*”に追加します。

```
#include "r_t4_itcpip.h"
/***********************************************************************
Macro definitions
***********************************************************************/
/* Size of Ethernet  receive buffer,refer to tcp_ccep[].rbufsz  */
#define BUFFER_SIZE             (1460)

/*********************echo server main function*********************************/
void echo_srv(void)
{
    ID          cepid=1; /*ID of a TCP communication end point  ("1"~ "30") */
    ID          repid=1; /*ID of a TCP reception point ("1"~ "30") */
    T_IPV4EP    dst_addr; /*destination IP address (PC)*/
    UB          rcv_buf[BUFFER_SIZE ];/*receive buffer*/
    ER          ercd;/*error code*/

    /* Make one TCP connection on Ethernet channel.
     Ethernet 0: 192.168.0.3 - Port: 1024 (refer to config_tcpudp.c)*/

     while (1)
    {
        /* TCP connection open */
        ercd = tcp_acp_cep(cepid,repid, &dst_addr,TMO_FEVR);

        if(E_OK == ercd)
        {
                /* process of echo server */
                while(1)
                {
                        /* TCP receive data */
                        ercd = tcp_rcv_dat(cepid, rcv_buf, sizeof(rcv_buf), TMO_FEVR);
                        if(ercd <= 0)
                        {
                                break;
                        }

                        /* TCP transmit echo data back */
                        ercd = tcp_snd_dat(cepid, rcv_buf, ercd, TMO_FEVR);
                        if(ercd < 0)
                        {
                                break;
                        }
                }

                /* Close TCP connection */
                tcp_sht_cep(cepid); /*TCP transmission shutdown */
                tcp_cls_cep(cepid, TMO_FEVR);/*TCP connection disconnect  */

            }
        }

} /* End of function echo_srv */
```

ソース・ファイルはこのようになります。

```
 * echo_srv.c
#include "r_t4_itcpip.h"
/**********************************************************************************
Macro definitions
**********************************************************************************/
/* Size of Ethernet  receive buffer,refer to tcp_ccep[].rbufsz  */
#define BUFFER_SIZE             (1460)
/*********************echo server main function*********************************/
void echo_srv(void)
{
    ID          cepid=1; /*ID of a TCP communication end point  ("1"~ "30") */
    ID          repid=1; /*ID of a TCP reception point ("1"~ "30") */
    T_IPV4EP    dst_addr; /*destination IP address (PC)*/
    UB          rcv_buf[BUFFER_SIZE ];/*receive buffer*/
    ER          ercd;/*error code*/
    /* Make one TCP connection on Ethernet channel.
     Ethernet 0: 192.168.0.3 - Port: 1024 (refer to config_tcpudp.c)*/
     while (1)
    {
         /* TCP connection open */
         ercd = tcp_acp_cep(cepid,repid, &dst_addr,TMO_FEVR);
         if(E_OK == ercd)
         {
             /* process of echo server */
             while(1)
             {
                 /* TCP receive data */
                 ercd = tcp_rcv_dat(cepid, rcv_buf, sizeof(rcv_buf), TMO_FEVR);
                 if(ercd <= 0)
                 {
                     break;
                 }

                 /* TCP transmit echo data back */
                 ercd = tcp_snd_dat(cepid, rcv_buf, ercd, TMO_FEVR);
                 if(ercd < 0)
                 {
                     break;
                 }
             }

             /* Close TCP connection */
             tcp_sht_cep(cepid); /*TCP transmission shutdown */
             tcp_cls_cep(cepid, TMO_FEVR);/*TCP connection disconnect  */

            }
        }

} /* End of function echo_srv */
```

図 2-25　echo_srv.c

2.9 main()にアプリケーションコードを追加

1) Smart_Configurator_Example.c で、コード行[#include "r_smc_entry.h"]の後に、以下のコードを追加または上書きします。

```
#include <string.h>
#include "r_t4_itcpip.h"
#include "r_sys_time_rx_if.h"

/***********************************************************************
Macro definitions
***********************************************************************/
/* T4 work memory area size is 4.5 KB,refer to application note R20AN0051EJ0206 */
#define T4_WORK_SIZE (4608)
/***********************************************************************
Private global variables and functions
***********************************************************************/
static UW tcpudp_work[T4_WORK_SIZE / sizeof(UW) + 1];

/***********************************************************************
Imported global variables and functions (from other files)
***********************************************************************/
extern void echo_srv(void);
extern void R_ETHER_PinSet_ETHERC0_MII();
void main(void)
{
    ER   ercd;    /*error code*/

    sys_time_err_t systime_ercd;  /*system time error code*/
    char    ver[128];

    /* Initializes pins for r_ether_rx module */
         R_ETHER_PinSet_ETHERC0_MII();

    /* Get the version of T4 */
    strcpy(ver, (char*)R_t4_version.library);

    /* start system time */
    systime_ercd = R_SYS_TIME_Open();
    if (systime_ercd != SYS_TIME_SUCCESS)
    {
        while (1);
    }

    /* start LAN controller */
    ercd = lan_open();
    if (ercd != E_OK)
    {
        while (1)
        {
          /* Cannot open LAN controller */
        };
    }


    /* Initialize the TCP/IP */
    ercd = tcpudp_open(tcpudp_work);
    if (ercd != E_OK)
    {
        while (1)
        {
          /* Cannot open TCP/IP */
        };
    }

    /* start echo server */
    echo_srv();

    /* end TCP/IP */
    tcpudp_close();
    lan_close();
    R_SYS_TIME_Close();

}/* End of function main() */
```

ソース・ファイルはこのようになります。

```
#include "r_smc_entry.h"

#include <string.h>
#include "r_t4_itcpip.h"
#include "r_sys_time_rx_if.h"

/***********************************************************************************
Macro definitions
***********************************************************************************/
/* T4 work memory area size is 4.5 KB,refer to application note R20AN0051EJ0206 */
#define T4_WORK_SIZE (4606)
/***********************************************************************************
Private global variables and functions
***********************************************************************************/
static UW tcpudp_work[T4_WORK_SIZE / sizeof(UW) + 1];

/***********************************************************************************
Imported global variables and functions (from other files)
***********************************************************************************/
extern void echo_srv(void);
extern void R_ETHER_PinSet_ETHERC0_MII();
void main(void)
{
    ER   ercd;    /*error code*/
    sys_time_err_t systime_ercd;  /*system time error code*/
char    ver[128];

    /* Initializes pins for r_ether_rx module */
    R_ETHER_PinSet_ETHERC0_MII();

    /* Get the version of T4 */
    strcpy(ver, (char*)R_t4_version.library);

    /* start system time */
    systime_ercd = R_SYS_TIME_Open();
    if (systime_ercd != SYS_TIME_SUCCESS)
    {
        while (1);
    }

    /* start LAN controller */
    ercd = lan_open();
    if (ercd != E_OK)
    {
        while (1)
        {
            /* Cannot open LAN controller */
        };
    }


    /* Initialize the TCP/IP */
    ercd = tcpudp_open(tcpudp_work);
    if (ercd != E_OK)
    {
        while (1)
        {
            /* Cannot open TCP/IP */
        };
    }

    /* start echo server */
    echo_srv();

    /* end TCP/IP */
    tcpudp_close();
    lan_close();
    R_SYS_TIME_Close();

}/* End of function main() */
```

図 2-26　main.c

3.　操作の検証

3.1　マクロの定義

1)　プロジェクト・エクスプローラーの[Smart_Configurator_Example]を右クリックし、[プロパティ]をクリックします。

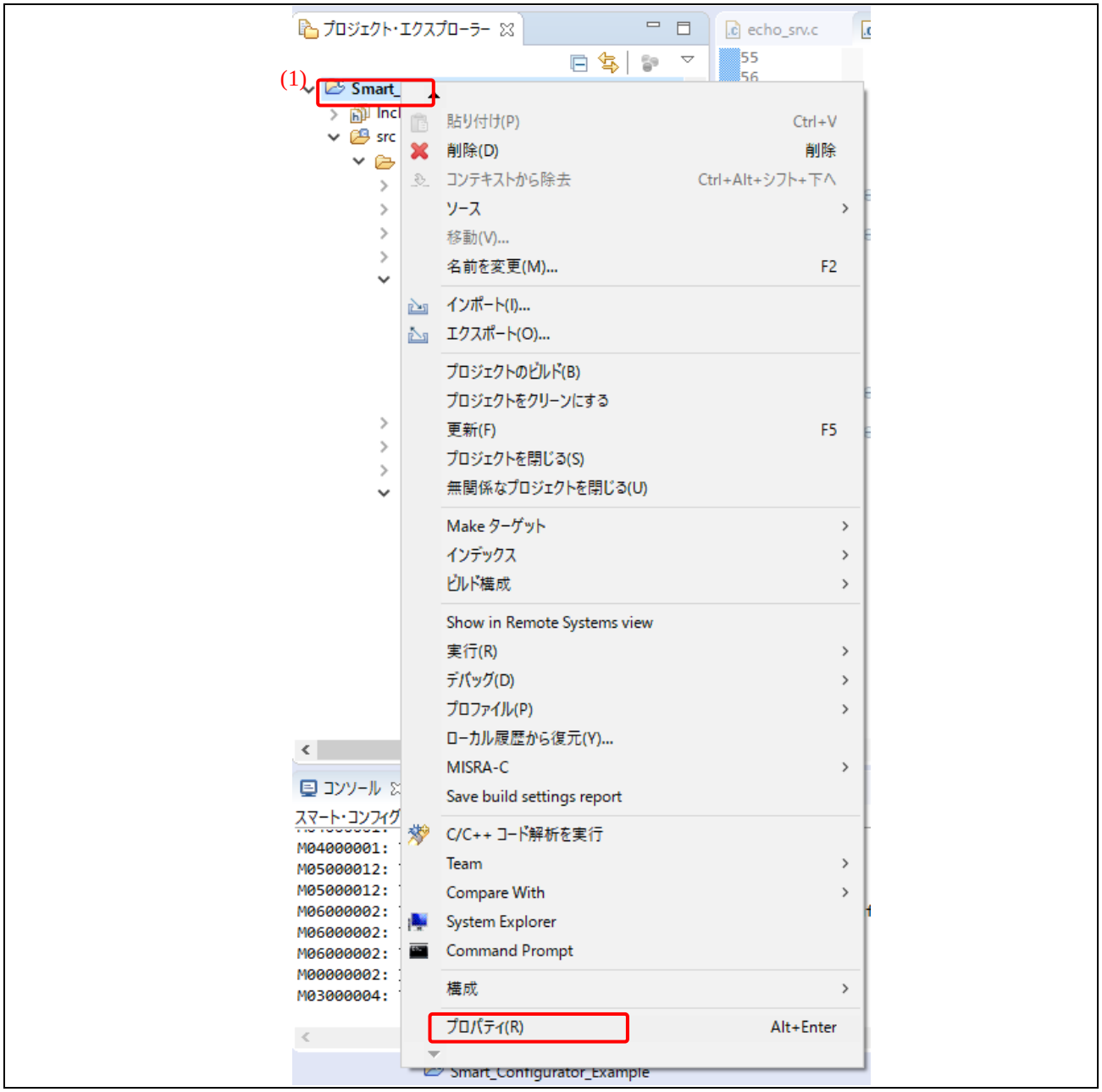

図 3-1　プロパティを開く

2) プロパティウィンドウの設定：
   a. [C/C++ビルド] → [設定]をクリックします。
   b. [ツール設定]タブで、[Compiler] → [ソース]を選択します。
   c. [プリプロセッサ・マクロの定義]の  ボタンをクリックします。
   d. ポップアップウインドウに、‘__RX’と入力します。
   e. [OK]をクリックし、[値を入力してください]ウインドウを閉じます。
   f. [OK]をクリックし、すべての設定を承認します。

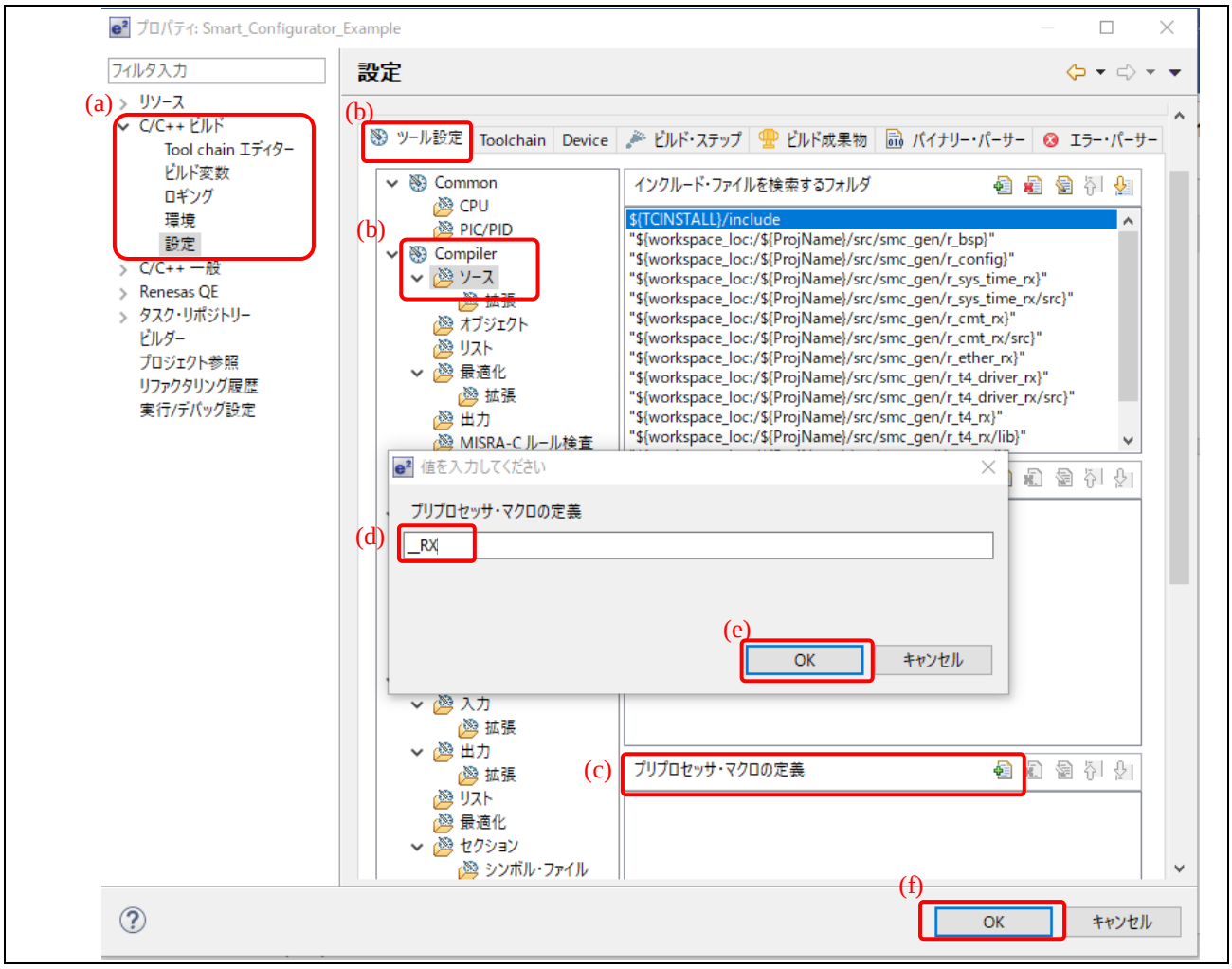

図 3-2　プロパティの変更

3.2 ボードと PC の設定

1) ターゲットボードのハードウェア設定が表 3-1 に従っていることを確認してください。

表 3-1 ボード設定

機能	設定	
	RSK+ RX65N-2MB	RSK+ RX64M
LAN ケーブル	ETHERNET コネクタ	ETHERNET0 コネクタ
MII モード	J8: Open (do not connect)	-
ET0LINKSTA	-	SW6.5: ON; SW6.6: OFF
ET0MDIO	-	J3: Pin1-2
ET0MDC	-	J4: Pin1-2
ET0ERXD1	-	SW6.7: ON; SW6.8: OFF
ET0RXCLK	-	J10: Pin1-2; R48 有効; R57 無効;
ET0RXER	-	SW6.9: ON; SW6.10: OFF
ET0TXEN	-	SW7.1: ON; SW7.2: OFF; SW7.3: OFF
ET0ETXD0	-	SW7.4: ON; SW7.5: OFF; SW7.6: OFF
ET0ETXD1	-	SW7.7: ON; SW7.8: OFF
ET0CRS	-	SW7.9: ON; SW7.10: OFF
ET0ERXD3	-	J11: Pin2-3; SW5.1: ON
ET0ERXD2	-	SW5.2: ON; SW5.3: OFF
ET0RXDV	-	SW5.4: ON; SW5.5: OFF
ET0TXER	-	SW5.6: ON; SW5.7: OFF; SW5.8: OFF
ET0TXCLK	-	SW5.9: ON; SW5.10: OFF
ET0ETXD2	-	J13: Pin2-3; SW6.1: ON
ET0ETXD3	-	J14: Pin2-3; SW6.2: ON
ET0COL	-	J12: Pin2-3; SW6.3: ON; SW6.4: OFF

2)　PC に対象ボードを接続します。

a.　RSK+ RX65N 2MB ボードと E1/ E2 Lite エミュレータを接続し、E1/ E2 Lite エミュレータを PC に接続します。

b.　LAN ケーブル（クロスまたはストレートケーブル）を 1 本使用し、RSK+ RX65N 2MB ボードを PC に接続します。

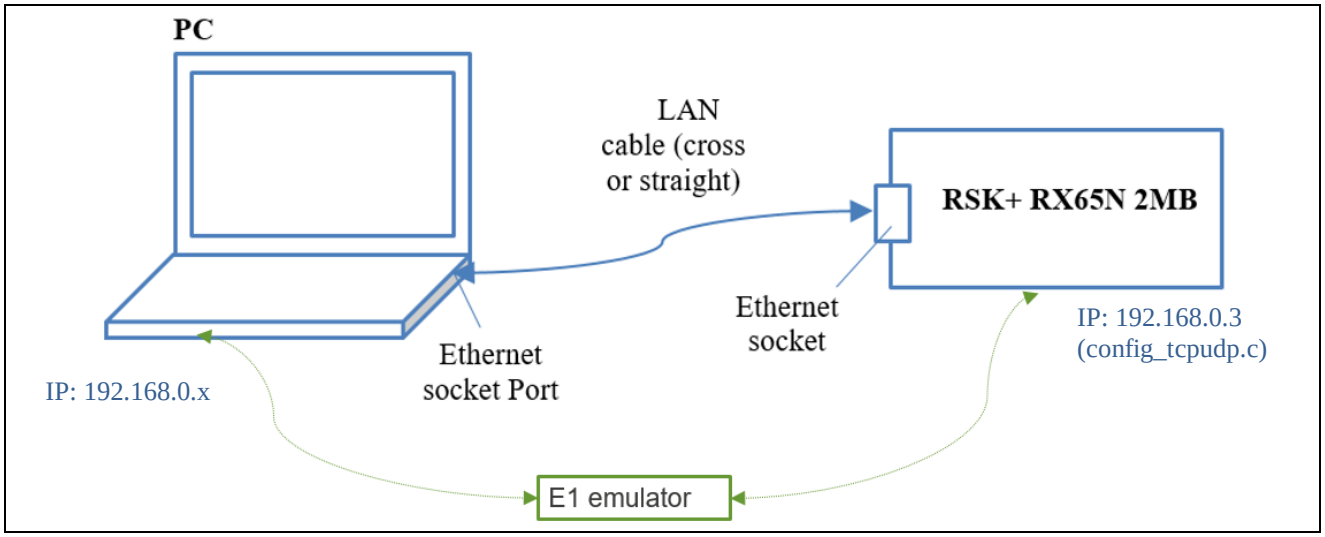

図 3-3　PC と RSK+ RX65N 2MB の接続

3)　PC に IP アドレスを設定します。

a.　Windows 10 OS 環境の[ネットワークとインターネット]設定で、以下を行います。

（a.1）[プロキシ]をクリックし、設定ページを開きます。

（a.2）[手動プロキシセットアップ]がオフになっていることを確認します。

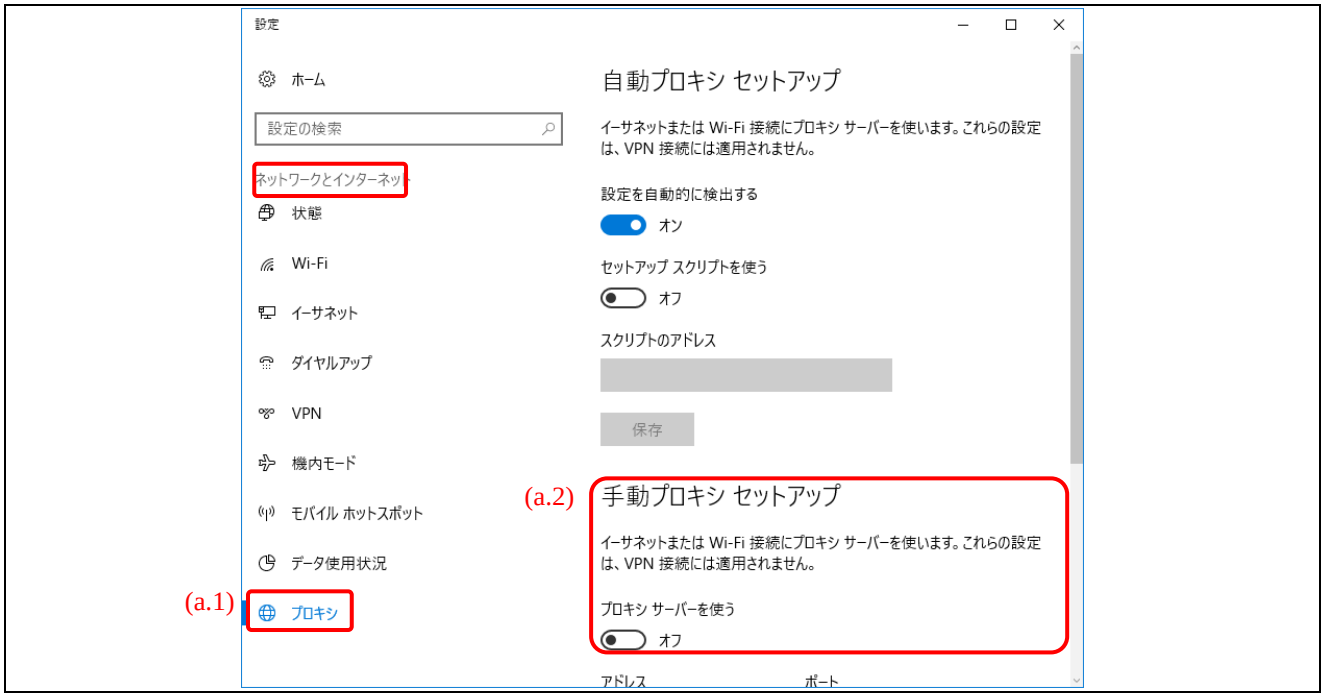

図 3-4　PC の IP アドレス設定

b.　[ネットワークとインターネット]設定で以下を行います。

（b.1）[イーサネット]をクリックし、設定ページを開きます。

（b.2）[アダプターのオプションを変更する]をクリックします。

（b.3）ポップアップウインドウで、“イーサネット”を右クリックし、[プロパティ]を選択します。

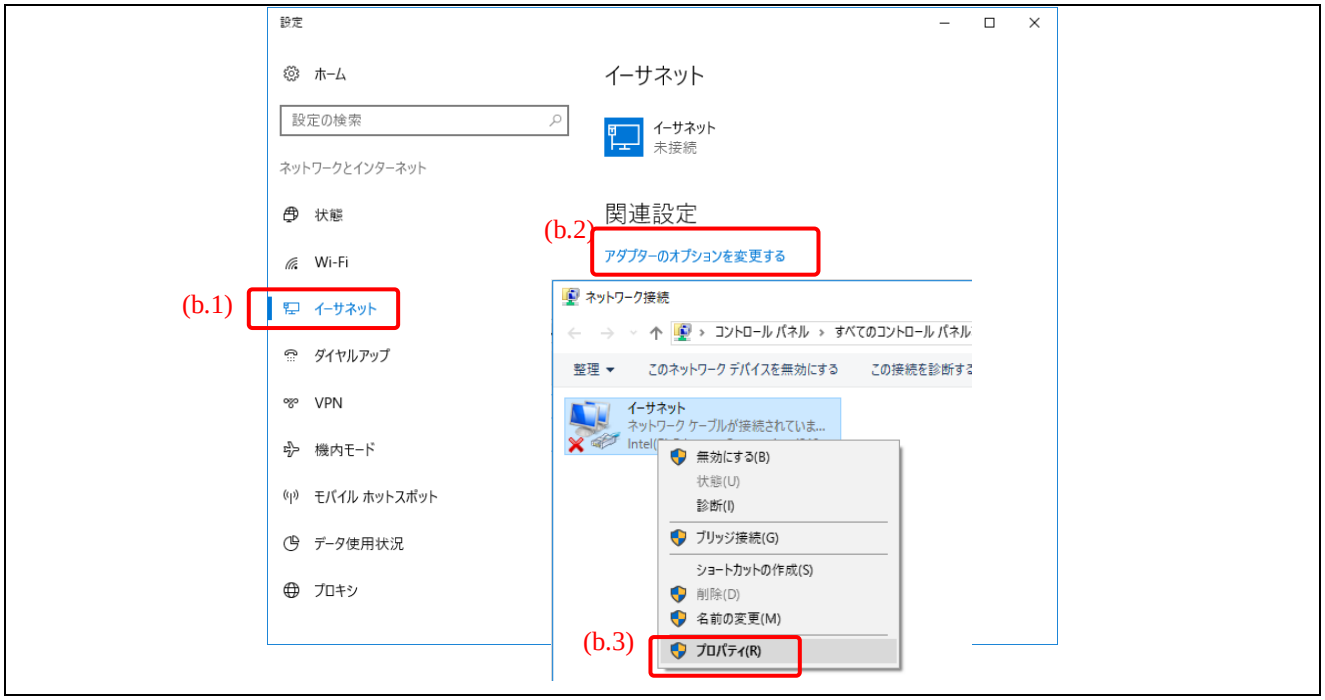

図 3-5　PC の IP アドレス設定

c.　ポップアップウインドウで、以下を設定します。

（c.1）[インターネットプロトコルバージョン 4（TCP/IPv4）]を選択します。

（c.2）[プロパティ]をクリックします。

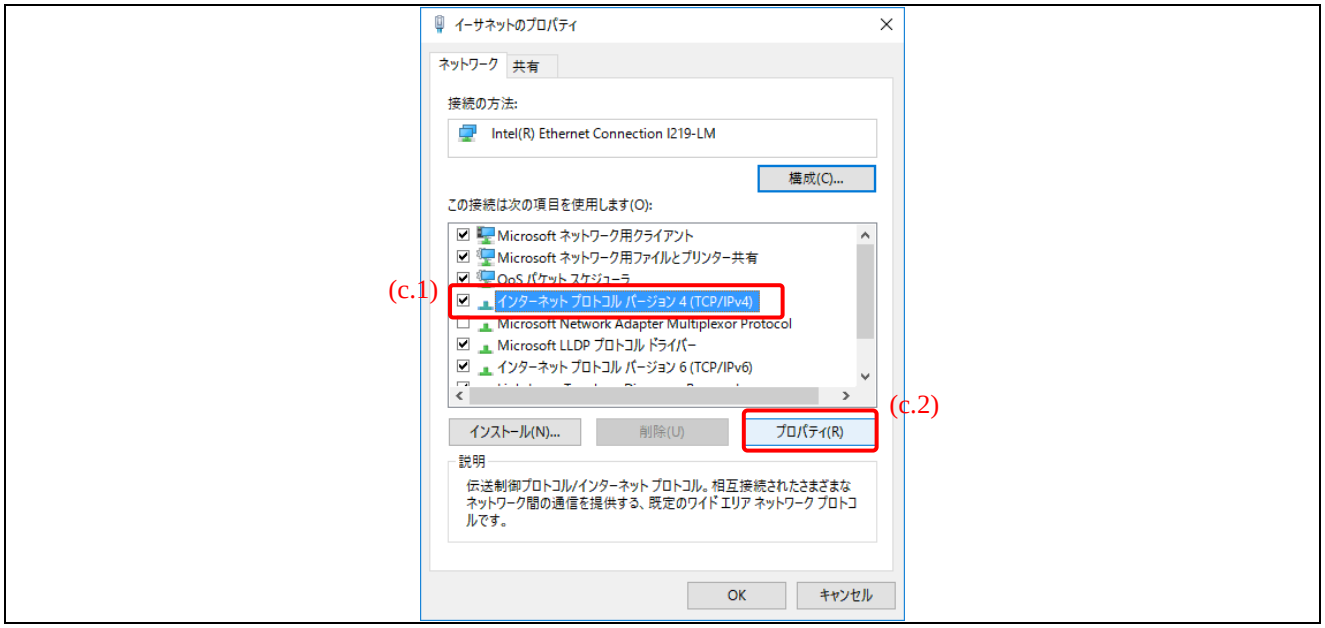

図 3-6　PC の IP アドレス設定

d.　ポップアップウインドウで、以下を行います。

（d.1）[次の IP アドレスを使う]を選択し、以下を設定します。

IP アドレス：192.168.0.100

（192.168.0.x の x には、ターゲットボードの IP アドレスとの競合を防ぐため、1 から 254 の中の 3 以外の数字を入れてください）

サブネットマスク：255.255.255.0

（d.2）[次の DNS サーバーのアドレスを使う]を選択します。

（d.3）[OK]をクリックし、TCP/IPv4 プロパティの設定を承認します。

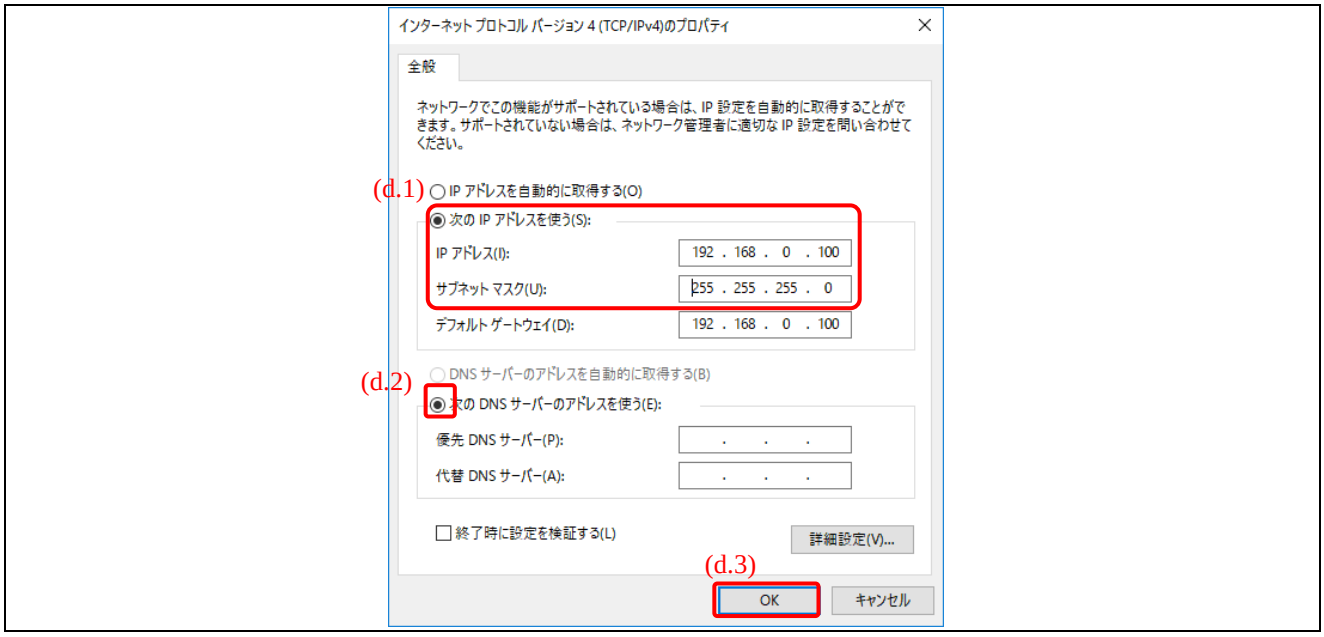

図 3-7　PC での IP アドレス設定

4) PC の“Telnet クライアント”を使用可能にします。

Windows 10 OS 環境の[プログラムと機能]で、以下を行います。

a. [Windows の機能の有効化または無効化]をクリックし、設定ページを開きます。

b. ポップアップウインドウの[Telnet クライアント]のチェックボックスにチェックを入れます。

c. [OK]をクリックし、設定を承認します。

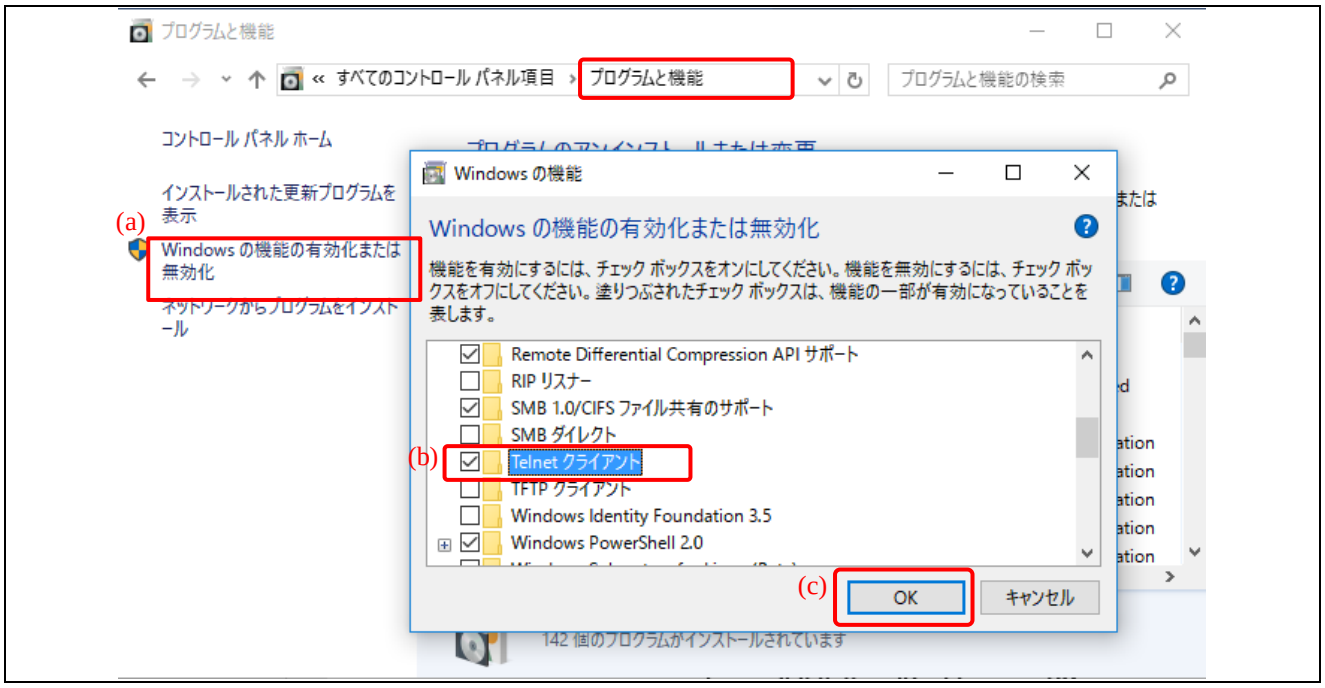

図 3-8　PC の“Telnet クライアント”の有効化

3.3　プロジェクトのビルドとデバッグ

1)　[プロジェクト] から[プロジェクトのビルド]を選択し、プロジェクトをビルドします。

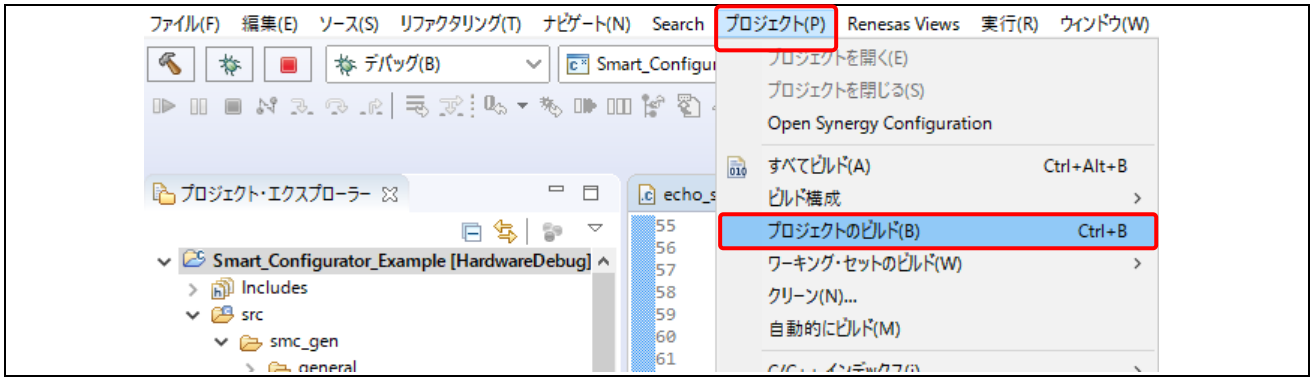

図 3-9　プロジェクトのビルド

2)　コードをデバッグします。

[実行]から[デバッグの構成…]または、  アイコンの横にある下向きの矢印から[デバッグの構成…]を選択し、“デバッグ構成”ウインドウを開きます。

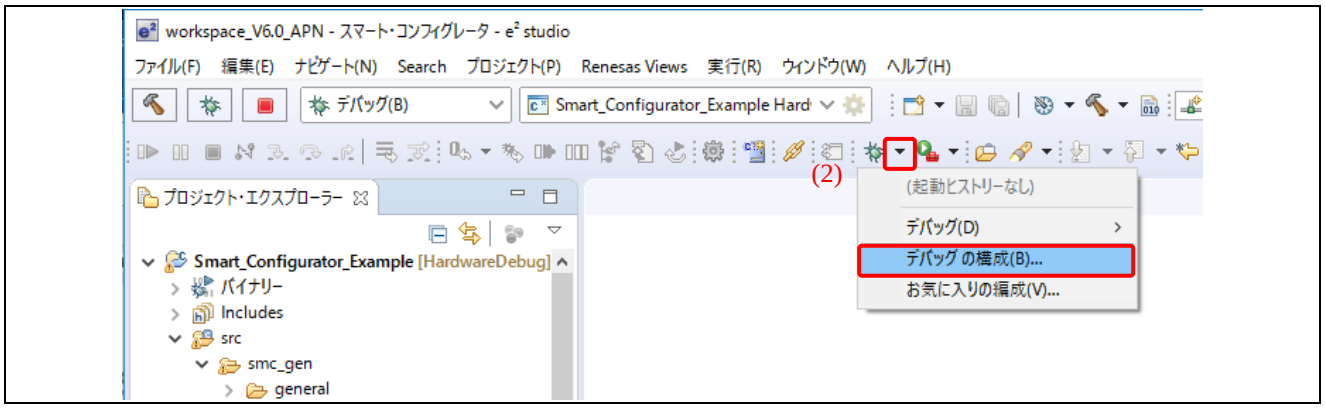

図 3-10　デバッグ構成ウインドウを開く

3) [Renesas GDB Hardware Debugging]を拡張し、[Smart_Configurator_Example_HardwareDebug]をクリックします。[Debugger]タブ上で[Connection Settings]をクリックし、以下のように設定してください。

a. Debug Hardware：E1 (RX)または E2 Lite (RX)を選択

b. Target Device：R5F565NE (RSK+65N-2MB)

R5F564ML (RSK+64M)

c. メイン・クロック・ソース：EXTAL

d. Extal 周波数[MHz]：24MHz

e. 内部フラッシュメモリ書き換え時にクロックソースの変更を許可する：はい

f. エミュレータから電源を供給する(MAX 200mA)：はい

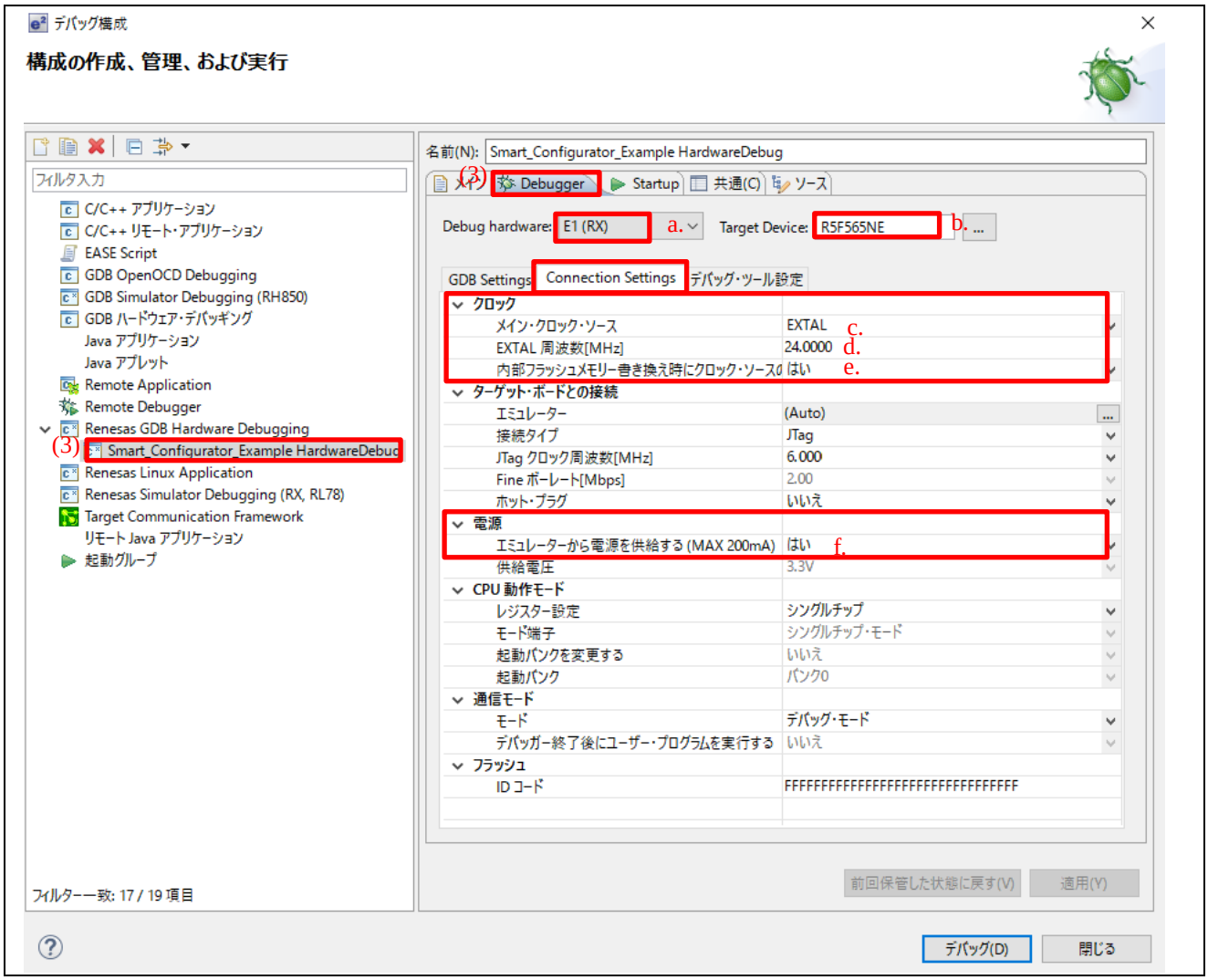

図 3-11　E1 エミュレータ接続時のデバッグコンフィグレーション例

4) “Startup”タブをクリックし、“ブレークポイント設定先： main”のチェックを外します。

5) [デバッグ]をクリックし、デバッグを開始します。

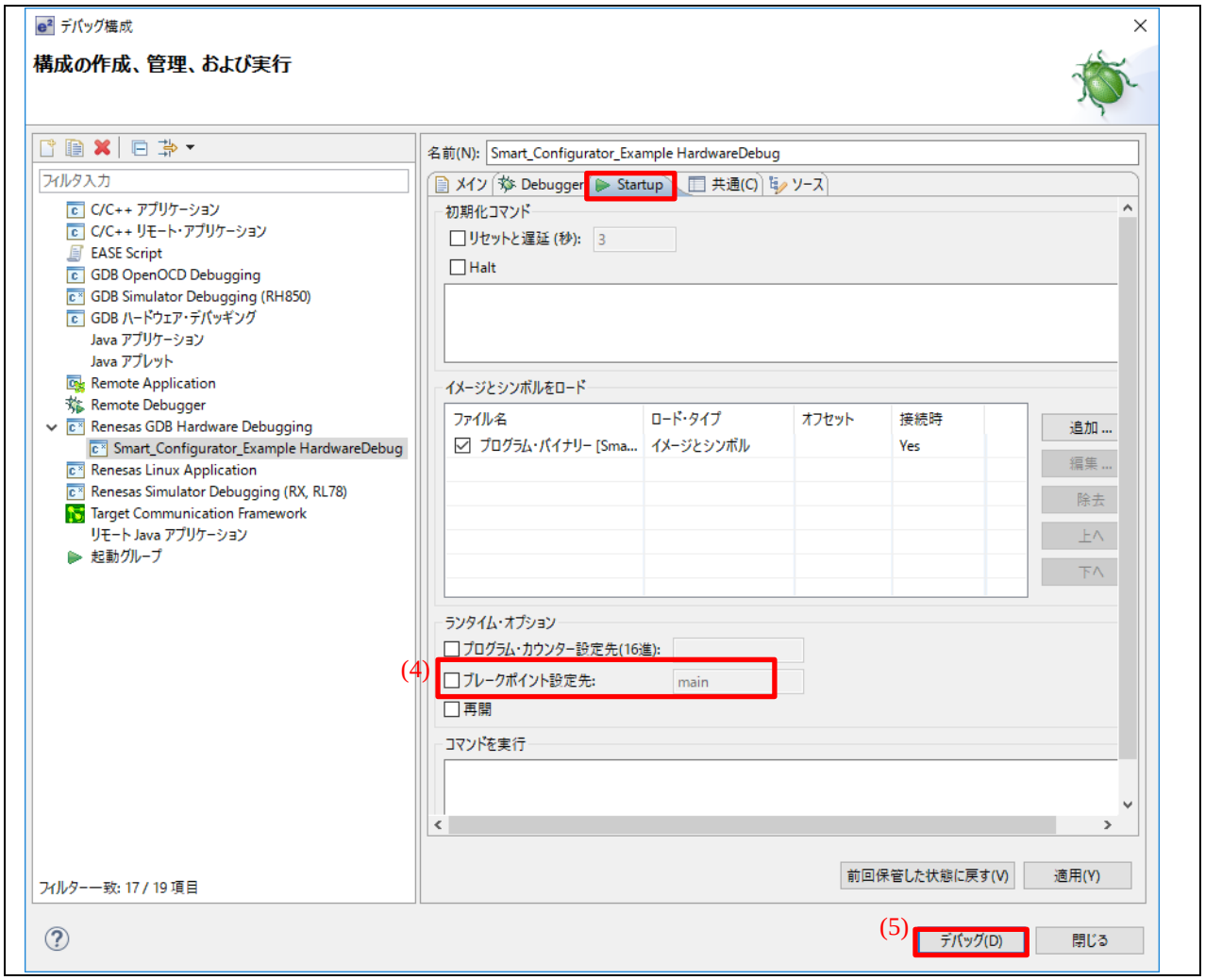

図 3-12　main のブレークポイント解除

6) ‘パースペクティブ切り替えの確認’ダイアログが表示されたら、[はい]をクリックして続けます。

7) をクリックし、プログラムを実行します。

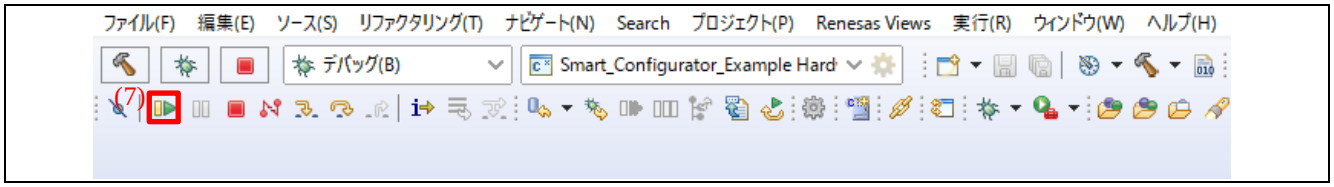

図 3-13　実行開始

次に、デバッグウィンドウでプログラムの中断やデバッグを停止する方法について説明しますが、ここでは実行しないでください

1) 中断ボタンをクリックし、プログラムの実行を一時停止します。

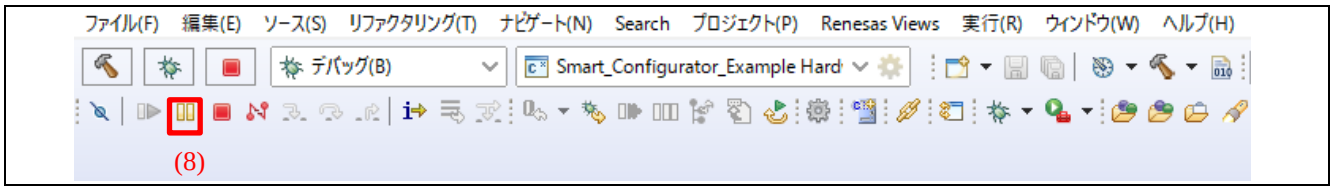

図 3-14　実行の中断

2) プログラムの実行を停止するには、切断ボタンをクリックし、デバッグセッションを終了します。

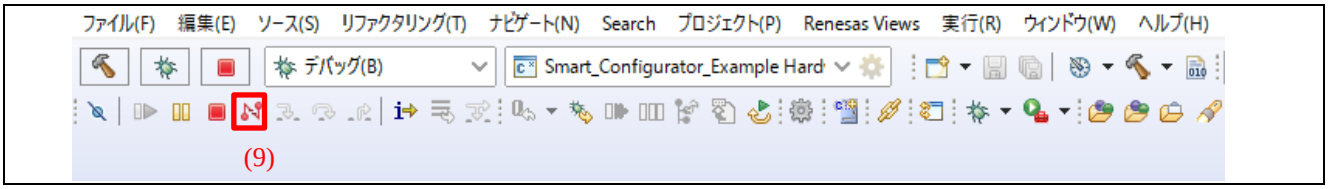

図 3-15　デバッグの停止

3.4　エコーサーバの操作

1)　Windows OS 環境の[コマンドプロンプト]ターミナルを開きます。

a.　コマンドを入力します：ping 192.168.0.3

b.　ターゲットボードからの応答（192.168.0.3）が表示されます。これは、ボードと PC 間の Ethernet 接続が成功したことを意味します。

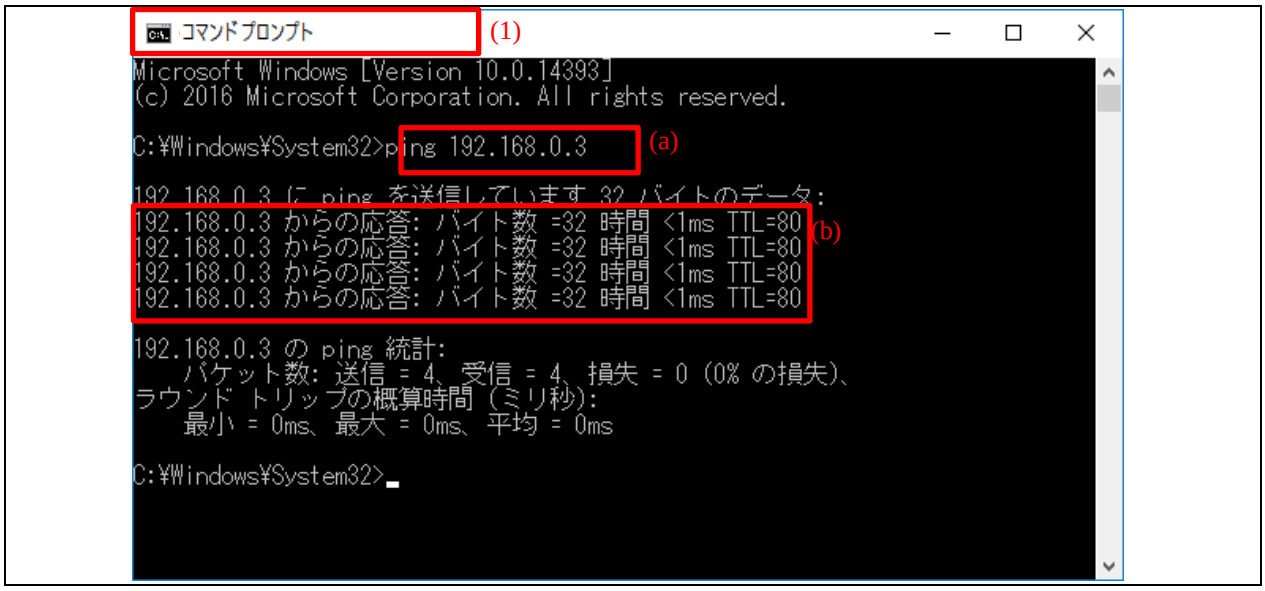

図 3-16　コマンドプロンプト

2)　次に、コマンド「telnet 192.168.0.3 1024」を入力します。

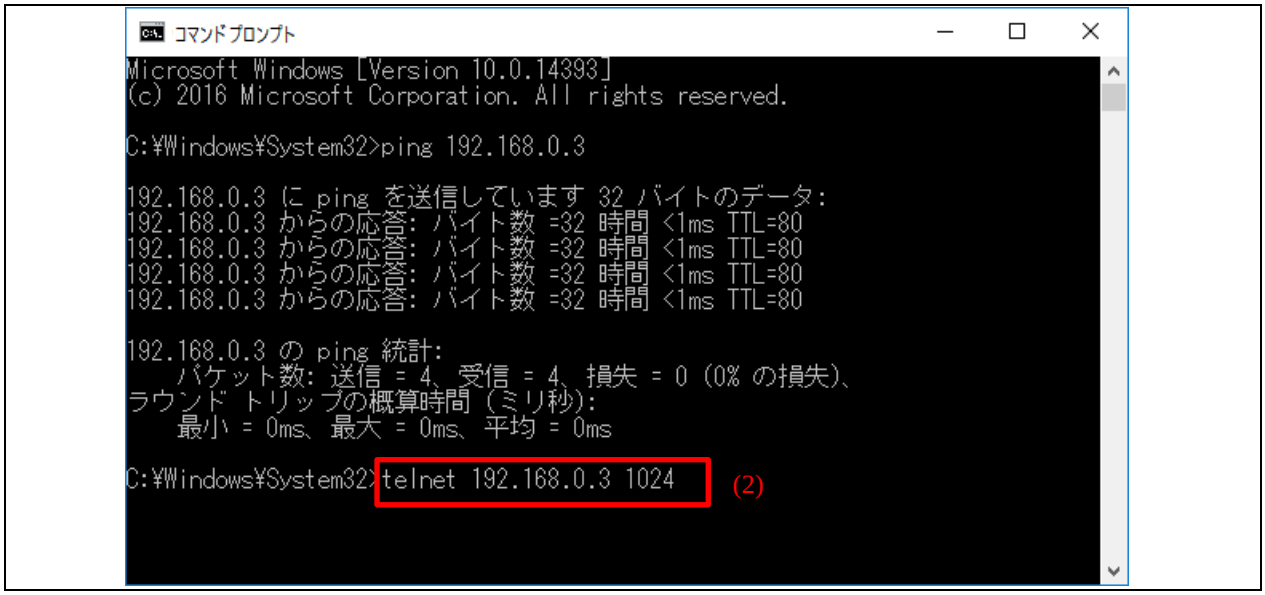

図 3-17　コマンドプロンプト

3)　ポップアップウインドウ[Telnet 192.168.0.3]で、以下を行います。

a.　任意の文字を入力し、エコーバックメッセージを確認します。

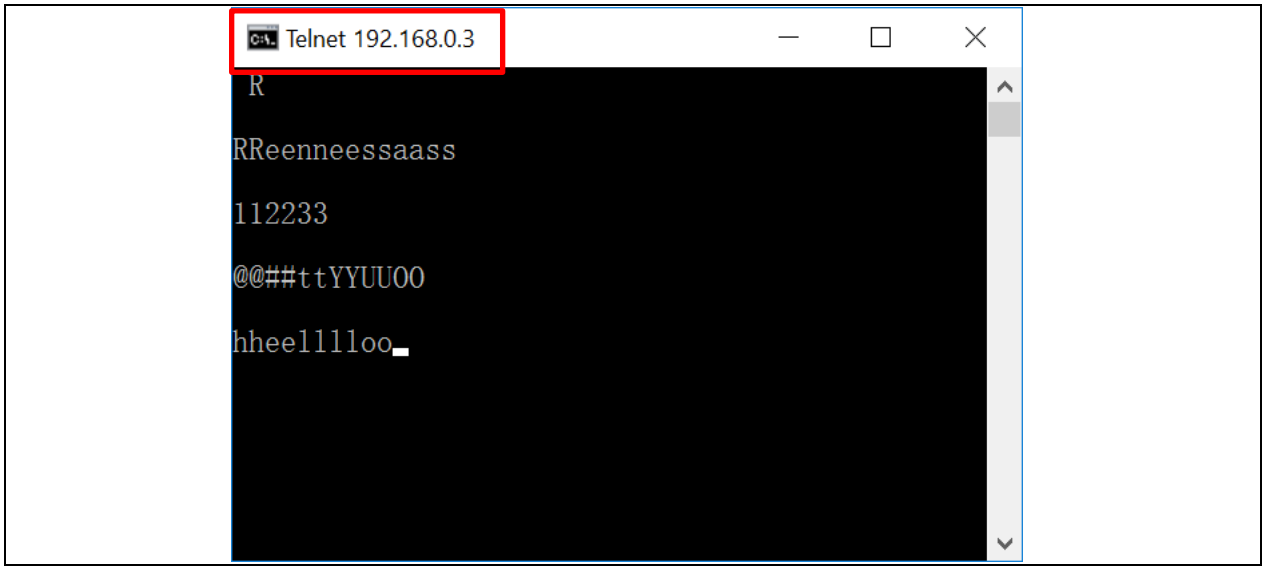

図 3-18　Telnet 192.168.0.3

3.5　開発支援ツール（QE）

開発の効率向上を支援するため、各種アプリケーション用の開発ツールとして、ルネサスは豊富な Quick and Effective Tool Solution (QE)を提供しています。TCP/IP を組み込む場合には、QE for TCP/IP の使用をお勧めします。QE for TCP/IP は、TCP/IP アプリケーションのデバッグを支援する機能を持っています。

ルネサス IDE でサポートするアプリケーションや、QE については、以下のリンクを参照してください。

QE:

https://www.renesas.com/qe

QE for TCP/IP:

https://www.renesas.com/qe-tcpip

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://www.renesas.com/

お問合せ先
http://www.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2018.03.27	—	新規作成

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、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。
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、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。
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、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。
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。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。
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.4.0-1 2017.11)

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/