R01AN0765JS0102
Rev.1.02
2013.03.22

RX62Nグループ
OS アダプタレイヤー: RI-600 と FreeRTOS™の点滅デモンストレーション

要旨

本アプリケーションノートでは、RI-600 用アダプタレイヤーを利用するサンプルプログラムを示しています。アダプタレイヤーにより、RI-600 プラットフォームを対象としたアプリケーションを再プログラミングする必要なしに FreeRTOS™ 上で実行できるようになります。

対象デバイス

- RX62N グループ MCU（製品番号: R5FF562N8BDBG）

動作確認ボード

- RX62N ルネサススタータキット ＋ （製品番号: R0K5562N0C000BE）

ドキュメント

- RI-600/4 V.1.00 ユーザーズマニュアル（製品番号: REJ10J2052-0100)
- アダプタレイヤーユーザーマニュアルおよび API 仕様書
- FreeRTOS（www.FreeRTOS.org）

注:　FreeRTOS™ は Real Time Engineers Ltd の商標です。

目次

1. はじめに

1.1 適用条件

- MCU: RX62N グループ
- 評価ボード: RX62N ルネサススタータキット ＋ （製品番号: R0K5562N0C000BE）
- 動作周波数:
  入力クロック: 12 MHz
  システムクロック（ICLK）: 96 MHz
  周辺モジュールクロック（PCLK）: 48 MHz
  外部バスクロック（BCLK）とSDRAMクロック（SDCLK）: 24 MHz
- 動作モード: シングルチップモード
  統合開発環境: ルネサスエレクトロニクス High-performance Embedded Workshop、Ver.4.09.01.007
- ルネサス RX ツールチェーン Version 1.0 に対する C コンパイラの設定は次のとおりです。
  - コンパイラオプション:

  リトルエンディアンオペレーション:
  -cpu=rx600 -
  include="$(PROJDIR)","$(WORKSPDIR)¥FreeRTOS","$(WORKSPDIR)¥Source¥Applications¥LED_Blinky¥include","$(WORKSPDIR)¥Source¥hew_files¥include","$(WORKSPDIR)¥Source¥FreeRTOS","$(WORKSPDIR)¥Source¥FreeRTOS¥Adaptor","$(WORKSPDIR)¥Build¥RX62N_RI600_Adaptor_Layer¥Debug" -
  define=__ADAPTOR_FREE_RTOS__ -output=obj="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -nostuff -speed –nologo

- ルネサス RX ツールチェーン Version 1.0 に対する最適化リンケージエディタの設定は次のとおりです。
  - リンカオプション:

  -noprelink -rom=D=R,D_1=R_1,D_2=R_2 -nomessage -list="$(CONFIGDIR)¥$(PROJECTNAME).map" -nooptimize -
  start=B_1,R_1,B_2,R_2,B,R,SU,SI/01000,PResetPRG/0FFFF8000,C_1,C_2,C,C$*,D*,P,PIntPRG,W*/0FFFF8100,FIXEDVECT/0FFFFFFD0 -nologo -output="$(CONFIGDIR)¥$(PROJECTNAME).abs" -end -
  input="$(CONFIGDIR)¥$(PROJECTNAME).abs" -form=stype -
  output="$(CONFIGDIR)¥$(PROJECTNAME).mot" –exit

- ルネサス RX ツールチェーン Version 1.2 に対する C コンパイラの設定は次のとおりです。
  - コンパイラオプション:

  Little Endian operation:
  - cpu=rx600 -include="$(PROJDIR)" -include="$(WORKSPDIR)¥FreeRTOS" -
  include="$(WORKSPDIR)¥Source¥Applications¥LED_Blinky¥include" -
  include="$(WORKSPDIR)¥Source¥hew_files¥include" -include="$(WORKSPDIR)¥Source¥FreeRTOS" -
  include="$(WORKSPDIR)¥Source¥FreeRTOS¥Adaptor" -
  include="$(WORKSPDIR)¥Build¥RX62N_RI600_Adaptor_Layer¥Debug" -define=__ADAPTOR_FREE_RTOS__ -output=obj="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -section=L=C -nostuff -speed -nologo

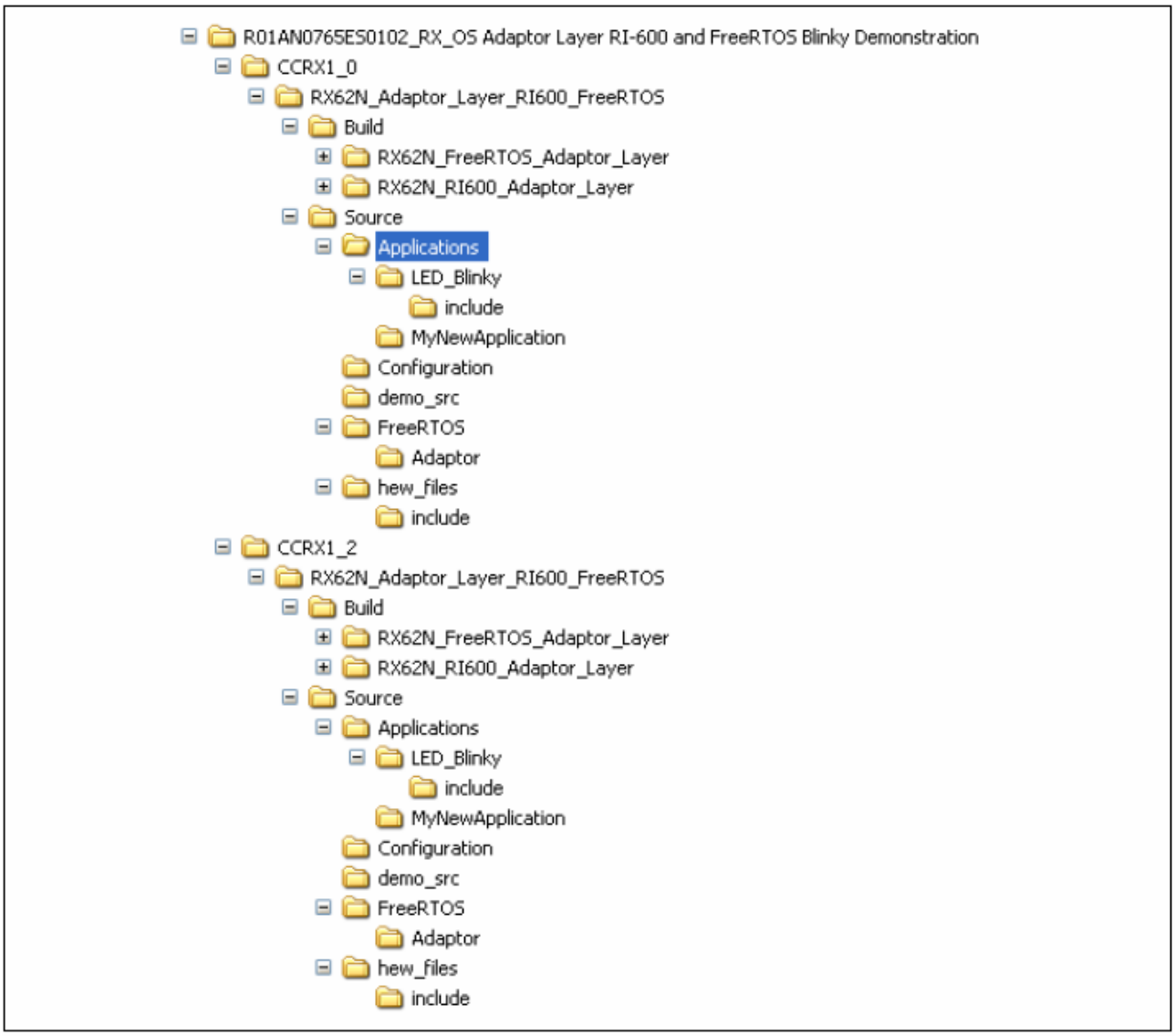
- ルネサス RX ツールチェーン Version 1.2 に対する最適化リンケージエディタの設定は次のとおりです。
  - リンカオプション:

  -noprelink -rom=D=R,D_1=R_1,D_2=R_2 -nomessage -list="$(CONFIGDIR)¥$(PROJECTNAME).map" -nooptimize -
  start=B_1,R_1,B_2,R_2,B,R,SU,SI/01000,PResetPRG/0FFFF8000,C_1,C_2,C,C$*,D*,P,PIntPRG,W*/0FFFF8100,FIXEDVECT/0FFFFFFD0 -nologo -output="$(CONFIGDIR)¥$(PROJECTNAME).abs" -end -
  input="$(CONFIGDIR)¥$(PROJECTNAME).abs" -form=stype -
  output="$(CONFIGDIR)¥$(PROJECTNAME).mot" -exit

2. ディレクトリ構造

2.1 概要

サンプルプロジェクトワークスペースのディレクトリには、以下が含まれています。



CCRXn_w

このフォルダには、ルネサスコンパイラのバージョン V1.0 と V1.2 用のワークスペースが含まれています。この理由は、コンパイラ V1.2 で生成される追加のセクション「L」を管理するためです。

¥RX62N_Adaptor_Layer_RI600_FreeRTOS

このフォルダは、Build および Source という 2 つのサブディレクトリを備えたメインワークスペースです。

¥Build

このディレクトリには、プロジェクトディレクトリが含まれます。1 つは RI-600 用で、もう 1 つは FreeRTOS™ 用です。

¥Source

Source ディレクトリには、サンプルアプリケーションのすべてのソースコードが含まれます。このディレクトリ内の点滅サンプルに関連するすべてのファイルは、モジュール分割のため、LED_Blink ディレクトリに置かれています。

¥Source¥FreeRTOS

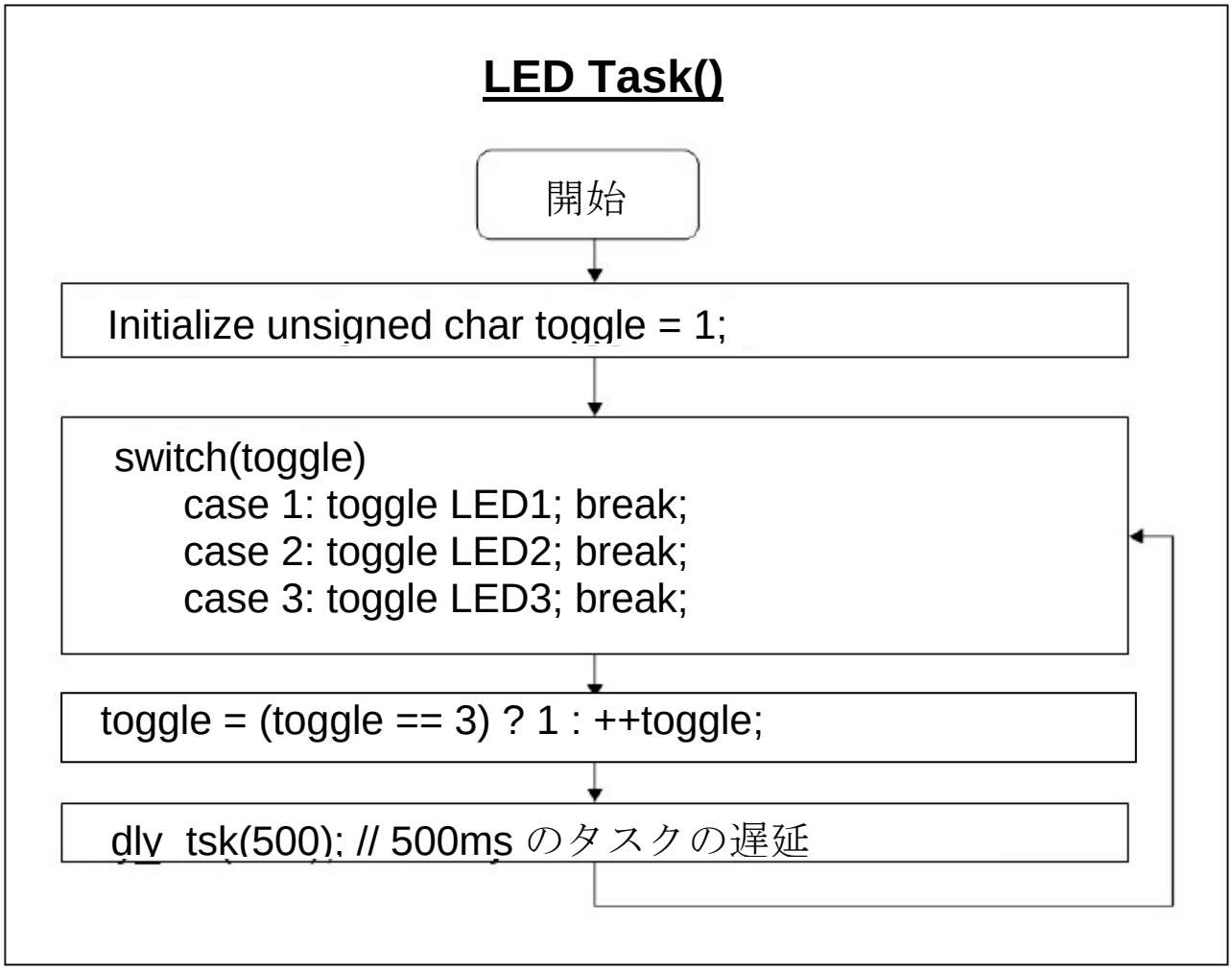
このディレクトリには、FreeRTOS™ に必要なすべてのファイルが含まれます。

¥Source¥FreeRTOS¥Adaptor

このフォルダには、アダプタレイヤーコードとコンフィグレーションファイルが含まれます。

3.　コンパイルおよびアプリケーションの実行

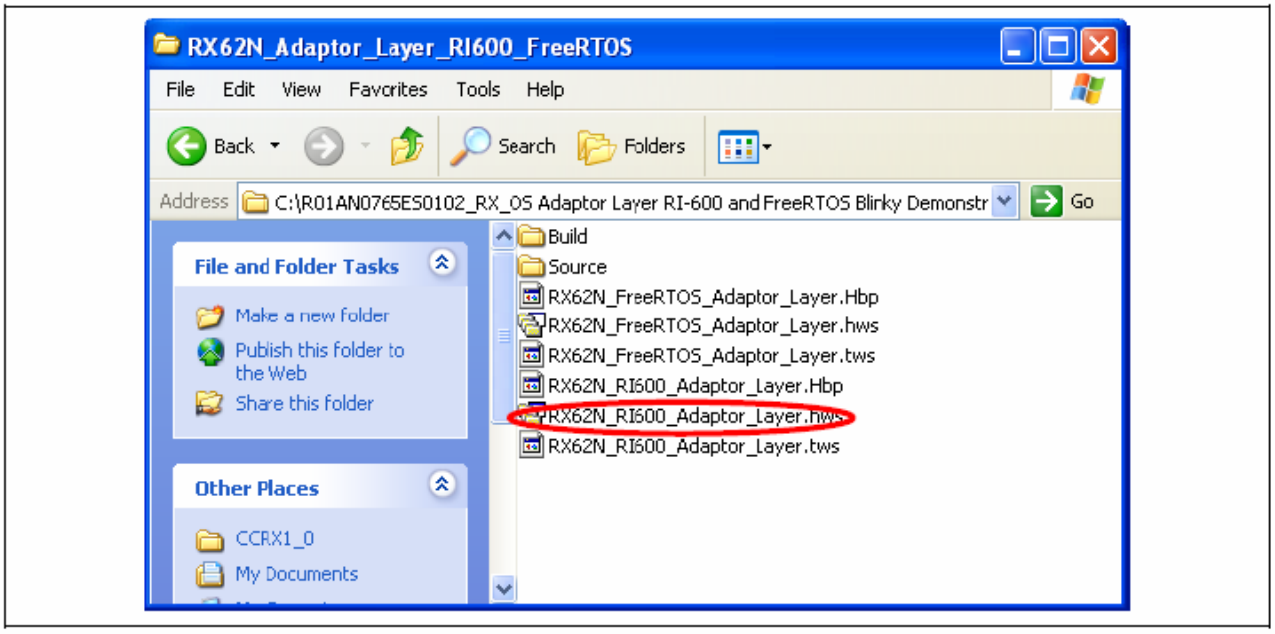
この項では、以下のプログラム実行フローを備えた LED 点滅アプリケーションを段階的に説明しています。



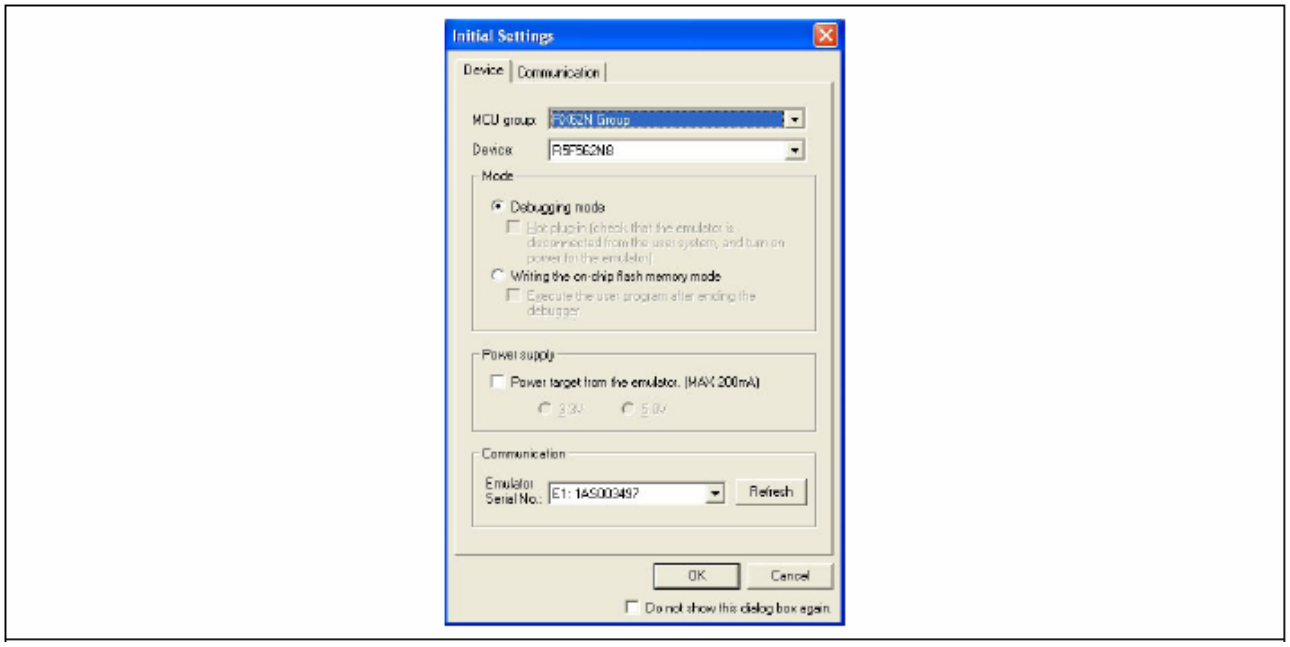
3.1　RI600

3.1.1　RI-600 プロジェクトワークスペースを開く

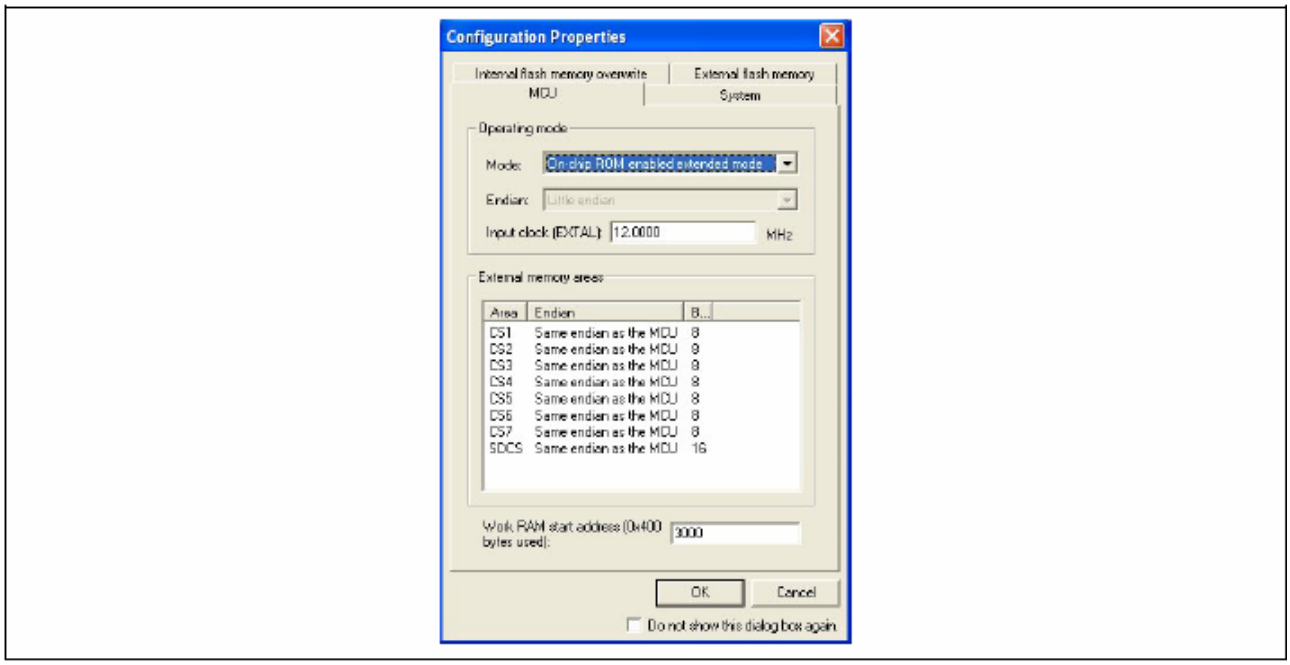
展開したディレクトリフォルダに置かれているプロジェクトワークスペースファイル（RX62N_RI600_Adaptor_Layer.hws）を見つけます。これは、次のように表示されています。



以下の設定で RX62N RSK に接続します。

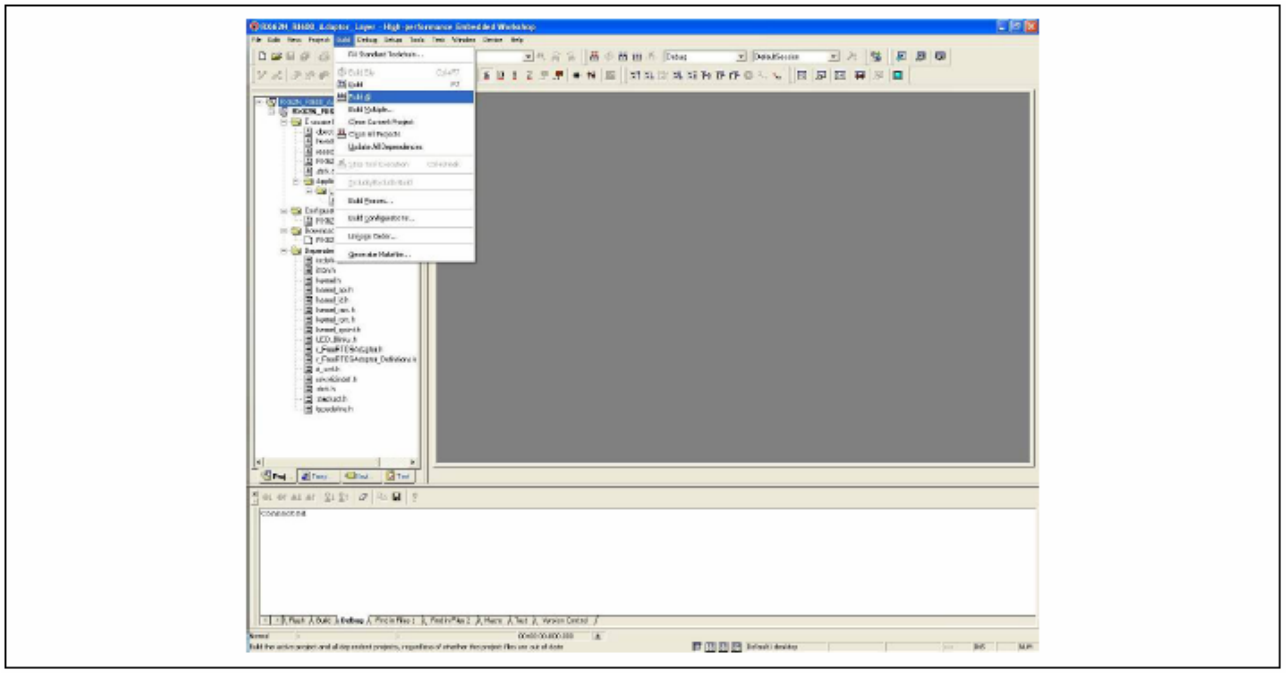


次の画面で、以下のように設定して接続を完了します。

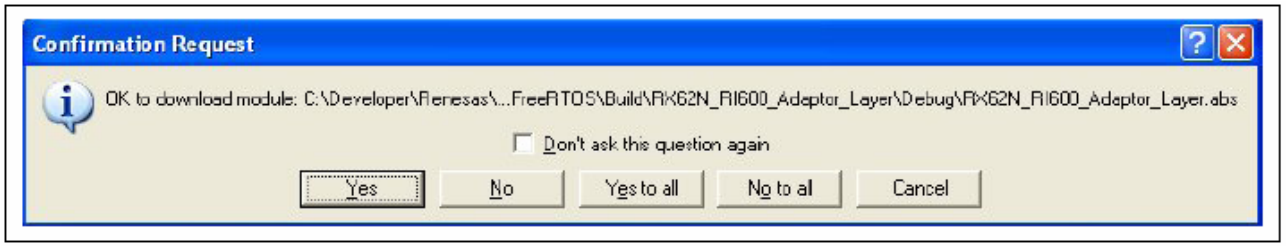


3.1.2　　コンパイルおよびビルド

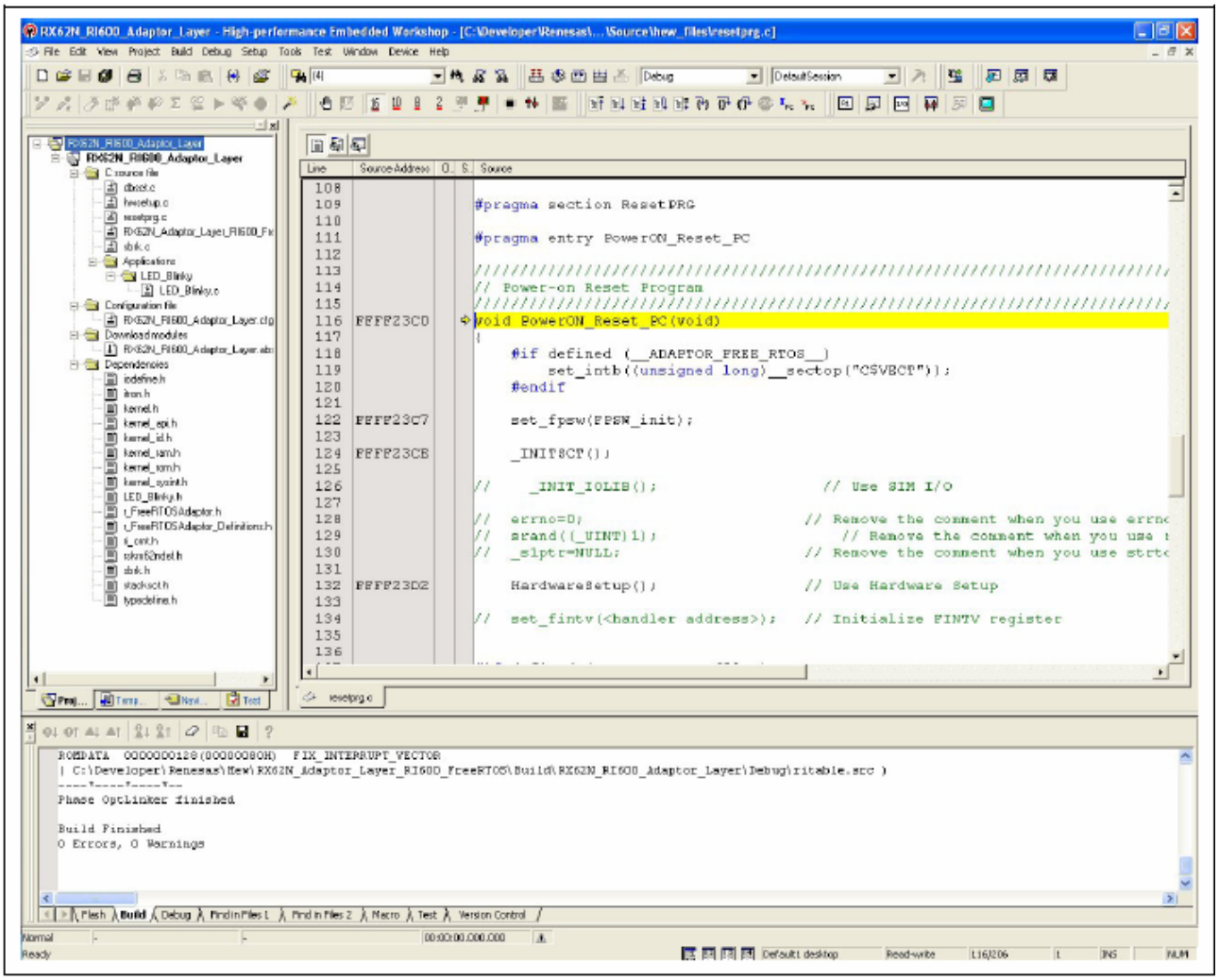
以下に示すように、[Build] → [Build All] を選択して、ワークスペースプロジェクトをコンパイルしてビルドします。



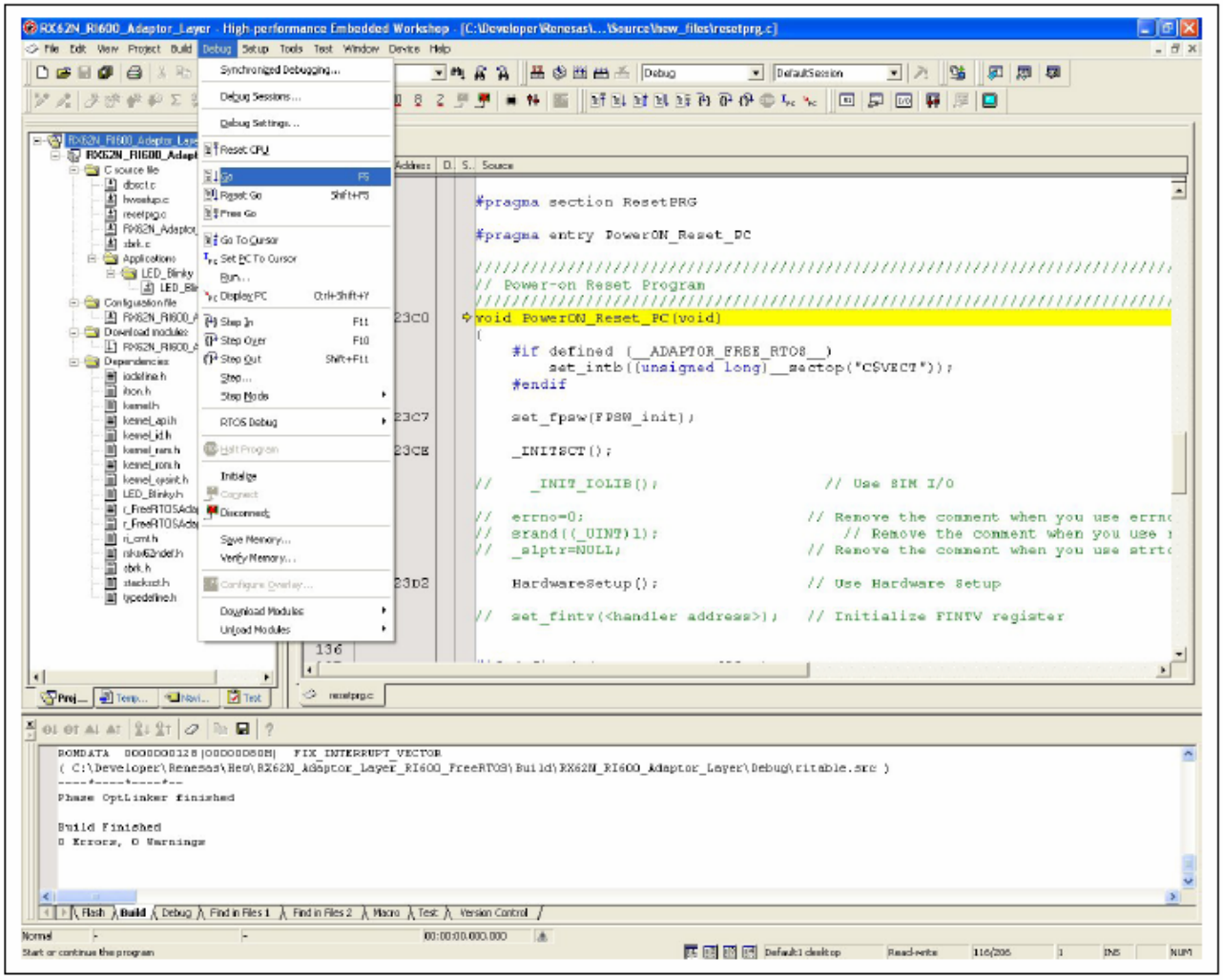
ビルドの完了後、モジュールをダウンロードするための確認要求が表示される場合があります。[Yes to all] が選択されていることを確認してください。



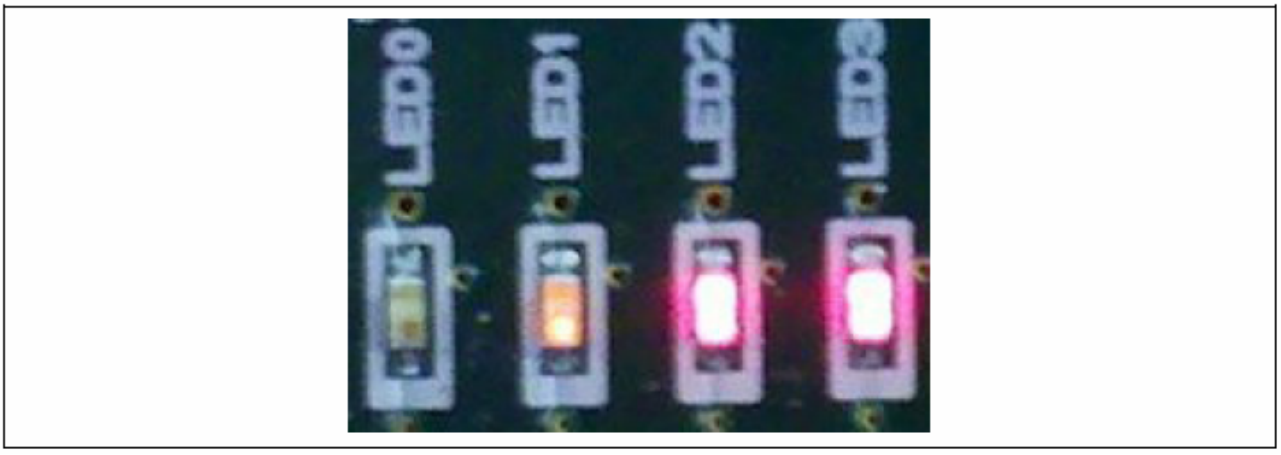
モジュールをロードした後、以下に示すように、カーソルを PowerON_Reset_PC() 関数にセットしてください。



ショートカットキー「F5」を押してプログラムを実行します。オプションで、以下に示すように、[Debug] → [Go] を選択することもできます。



RX62N RSK ボード上の LED の点滅は、次のようになります。

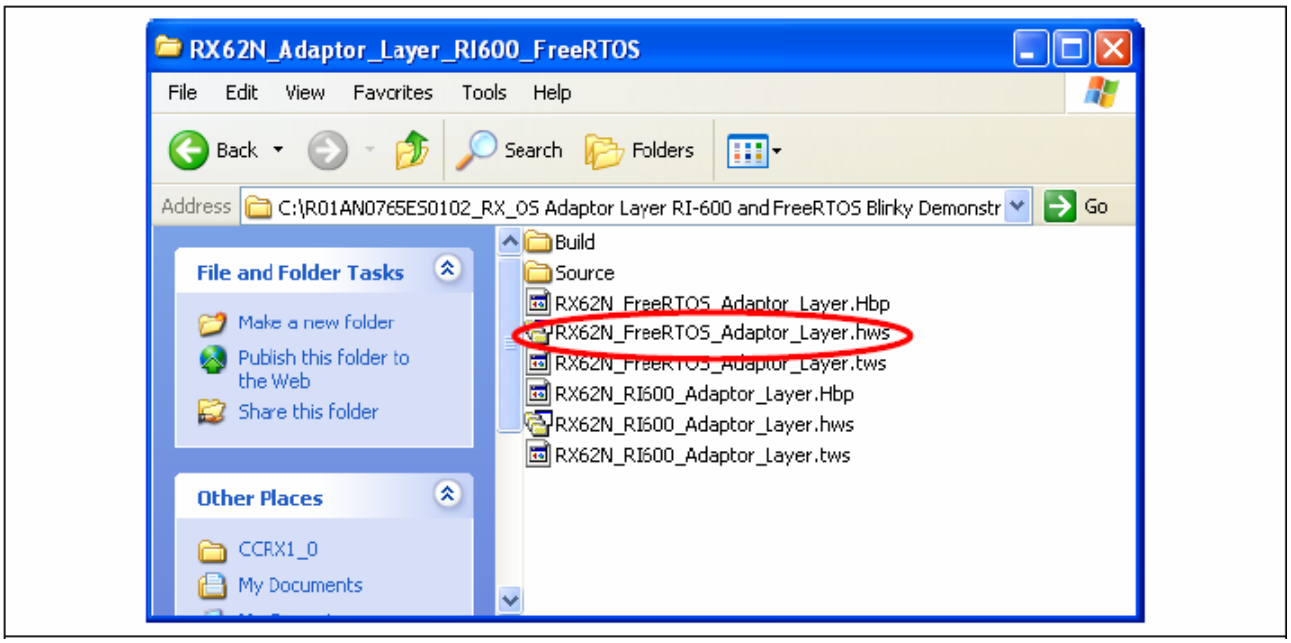


3.2　FreeRTOS™

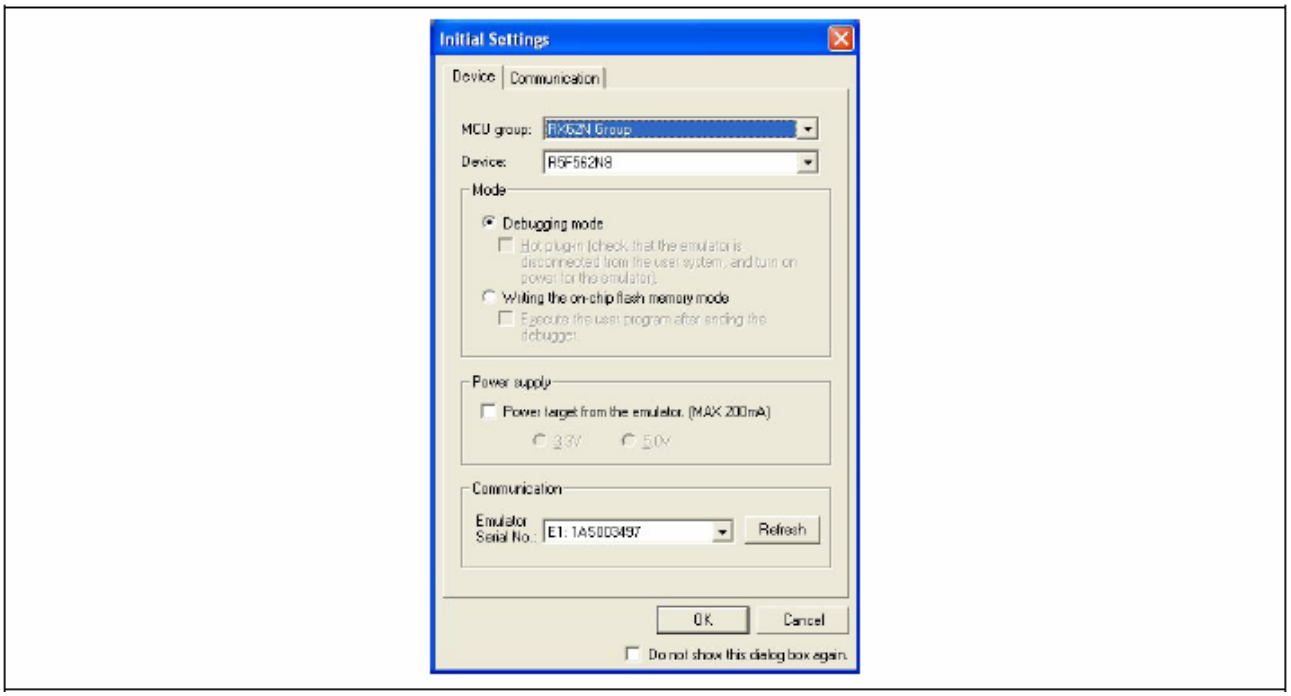
3.2.1　FreeRTOS™ プロジェクトワークスペースを開く

展開したディレクトリフォルダに置かれているプロジェクトワークスペースファイル（RX62N_FreeRTOS_Adaptor_Layer.hws）を見つけます。

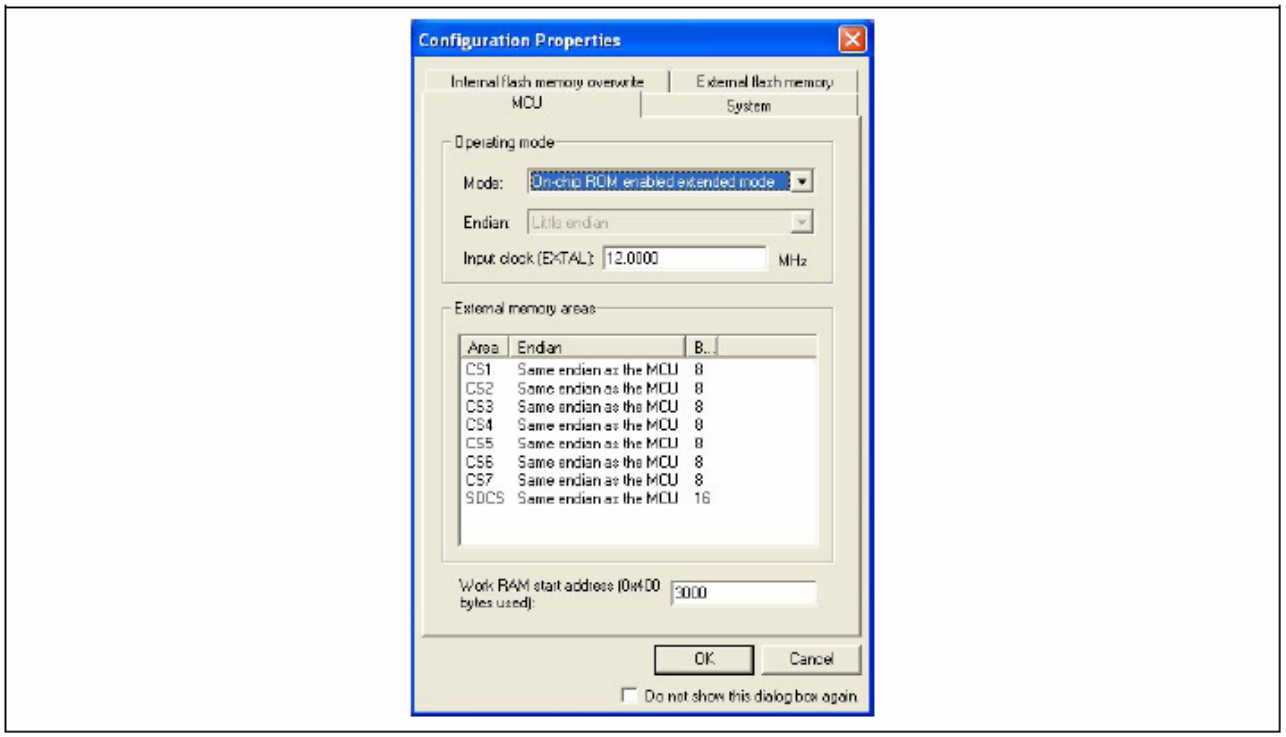
これは次のように表示されています。



以下の設定で RX62N RSK に接続します。

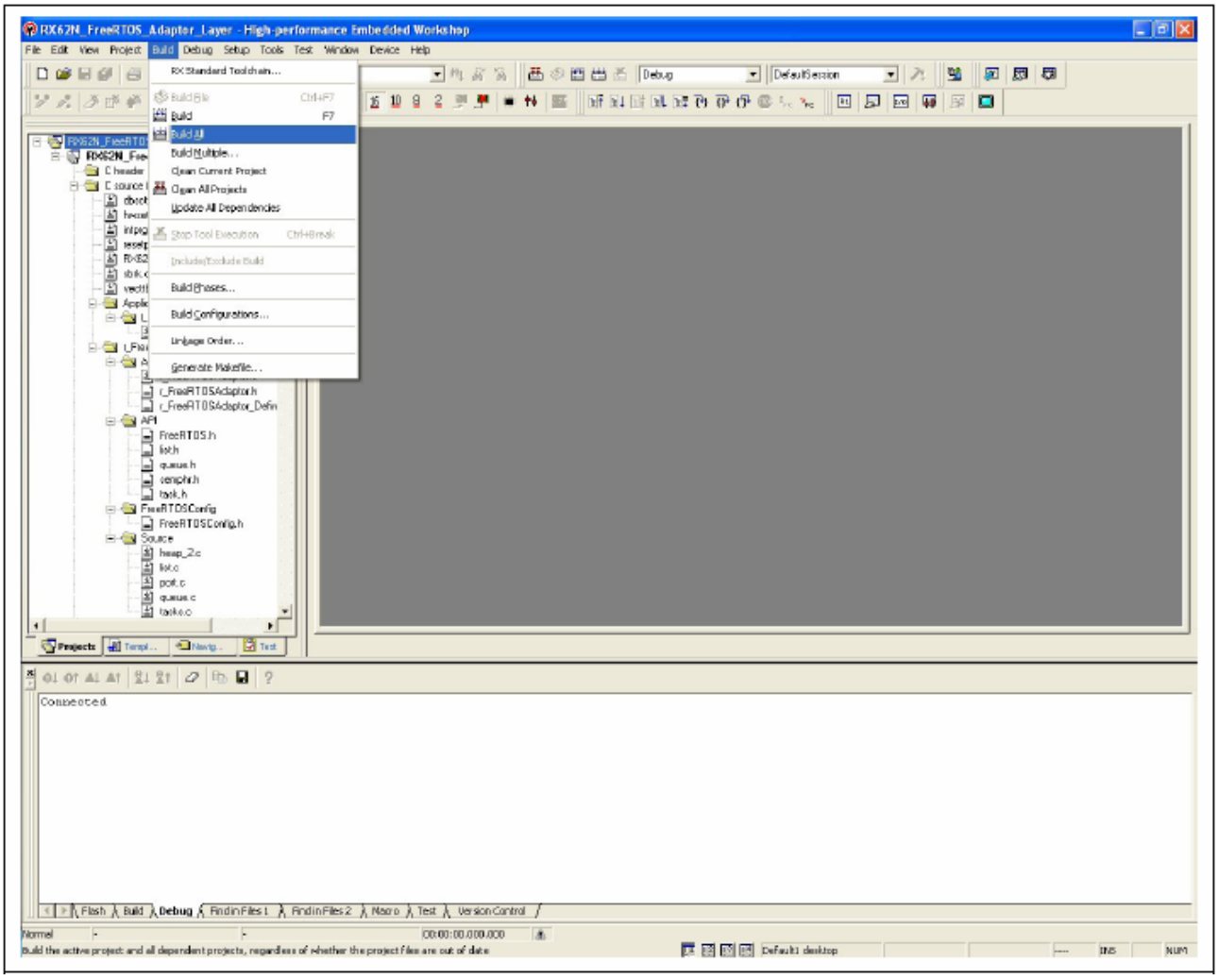


次の画面で、以下のように設定して接続を完了します。

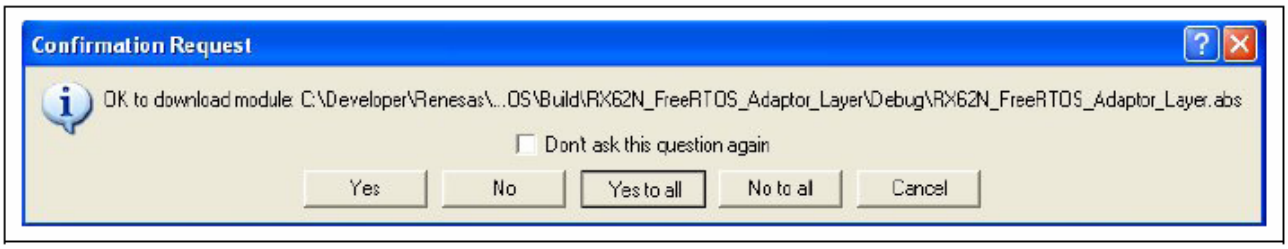


3.2.2　　コンパイルおよびビルド

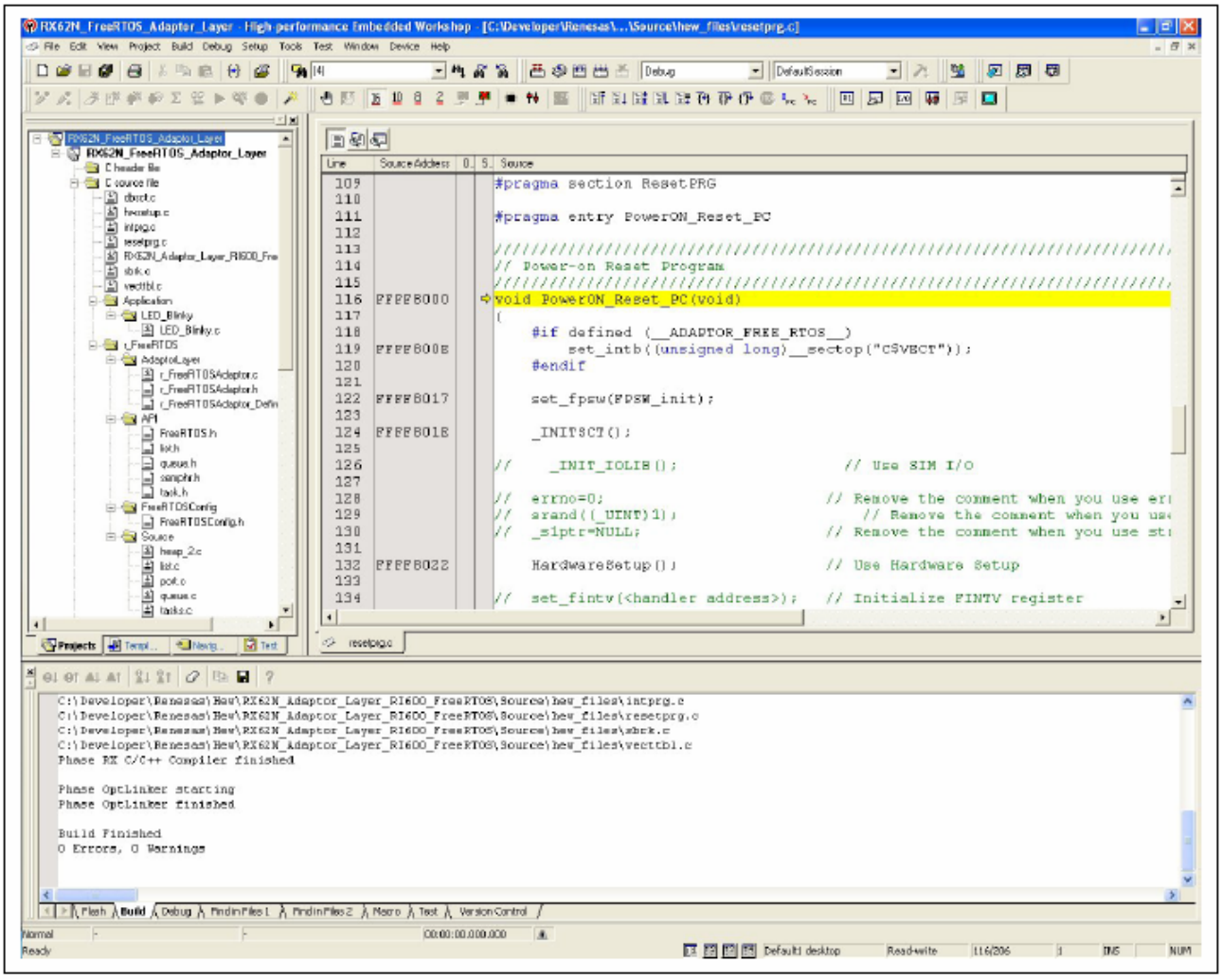
以下に示すように、[Build] → [Build All] を選択して、ワークスペースプロジェクトをコンパイルしてビルドします。



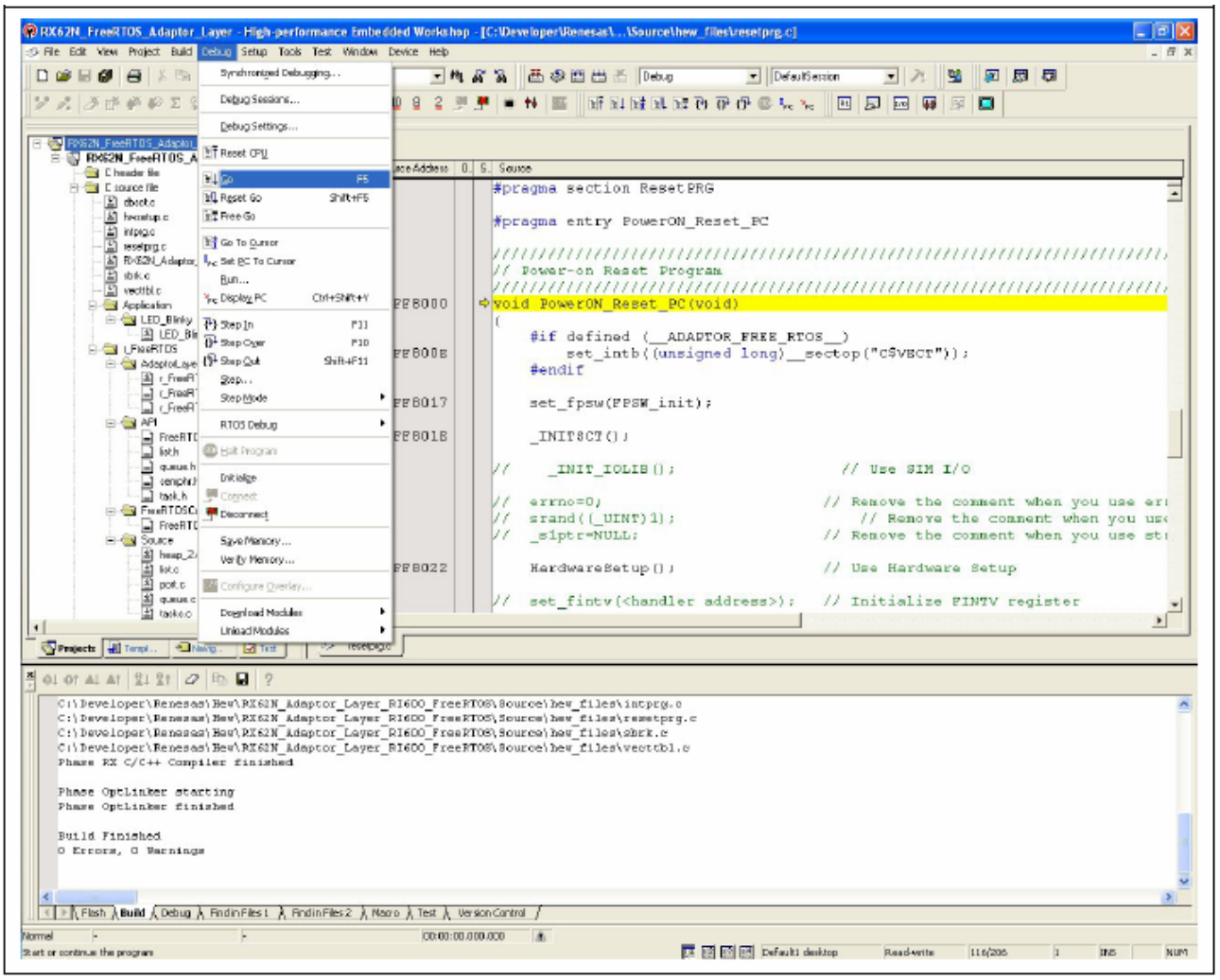
ビルドの完了後、モジュールをダウンロードするための確認要求が表示される場合があります。[Yes to all] が選択されていることを確認してください。



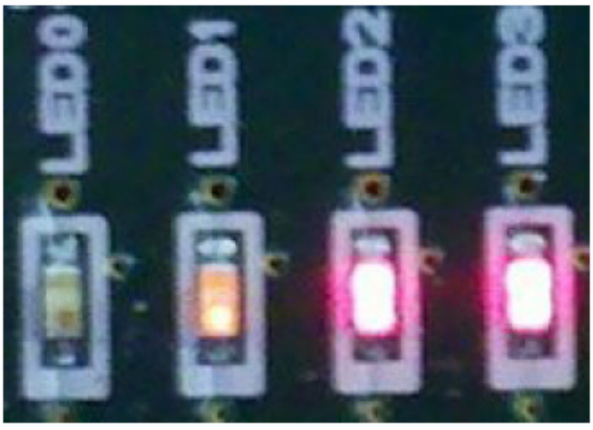
モジュールをロードした後、以下に示すように、カーソルを PowerON_Reset_PC() 関数にセットしてください。


RX62N_FreeRTOS_Adaptor_Layer - High performance Embedded Workshop - [C:\Developer\Renesas\...\Source\hew_files\resetprg.c]
#pragma section ResetPRG
#pragma entry PowerON_Reset_PC
// Power-on Reset Program
void PowerON_Reset_PC(void)
#if defined (__ADAPTOR_FREE_RTOS__)
set_intb((unsigned long)__sectop("C$VECT"));
#endif
set_fpsw(FPSW_init);
_INITSCT();
// _INIT_IOLIB();
// errno=0;
// srand((_UINT)1);
// _s1ptr=NULL;
HardwareSetup();
// set_fintv(<handler address>);
C:\Developer\Renesas\Hew\RX62N_Adaptor_Layer_RI600_FreeRTOS\Source\hew_files\intprg.c
C:\Developer\Renesas\Hew\RX62N_Adaptor_Layer_RI600_FreeRTOS\Source\hew_files\resetprg.c
C:\Developer\Renesas\Hew\RX62N_Adaptor_Layer_RI600_FreeRTOS\Source\hew_files\sbrk.c
C:\Developer\Renesas\Hew\RX62N_Adaptor_Layer_RI600_FreeRTOS\Source\hew_files\vecttbl.c
Phase RX C/C++ Compiler finished
Phase Optlinker starting
Phase Optlinker finished
Build Finished
0 Errors, 0 Warnings

ショートカットキー「F5」を押してプログラムを実行します。オプションで、以下に示すように、[Debug] → [Go] を選択することもできます。



RX62N RSK ボード上の LED の点滅は、次のようになります。



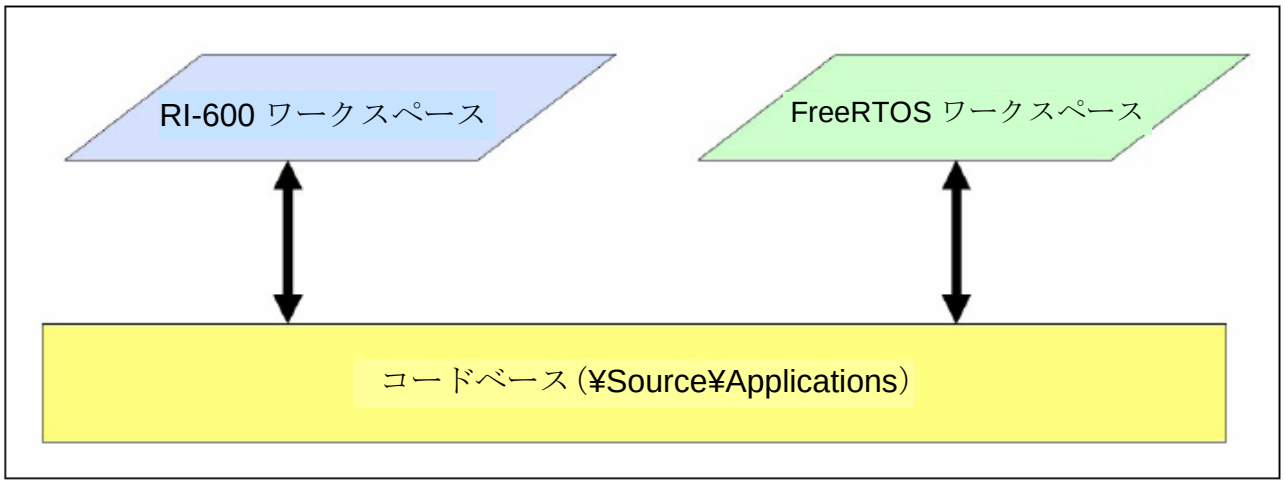
4. ワークスペースからアプリケーションを構築する

4.1 アプリケーションの開発

アプリケーションのコード開発と API は、RI-600 仕様のアプリケーションコードの開発と API に従う必要があります。つまり、アプリケーションは、RI-600 プラットフォームを対象としているかのように記述する必要があるということです。

4.2 アプリケーションコードベース

アプリケーションは提供されるワークスペースのいずれを使用しても開発できます。これにより、RI-600 と FreeRTOS™ の両方のプラットフォームで簡単にアプリケーションの開発およびテストを行うことができるようになります。ワークスペースのディレクトリとファイルは、以下に示すようにリンクされています。



4.3　アプリケーションメインプログラム

アプリケーションメインプログラムファイルは、いずれのワークスペースディレクトリにおいても以下のソースコードファイルにあります。

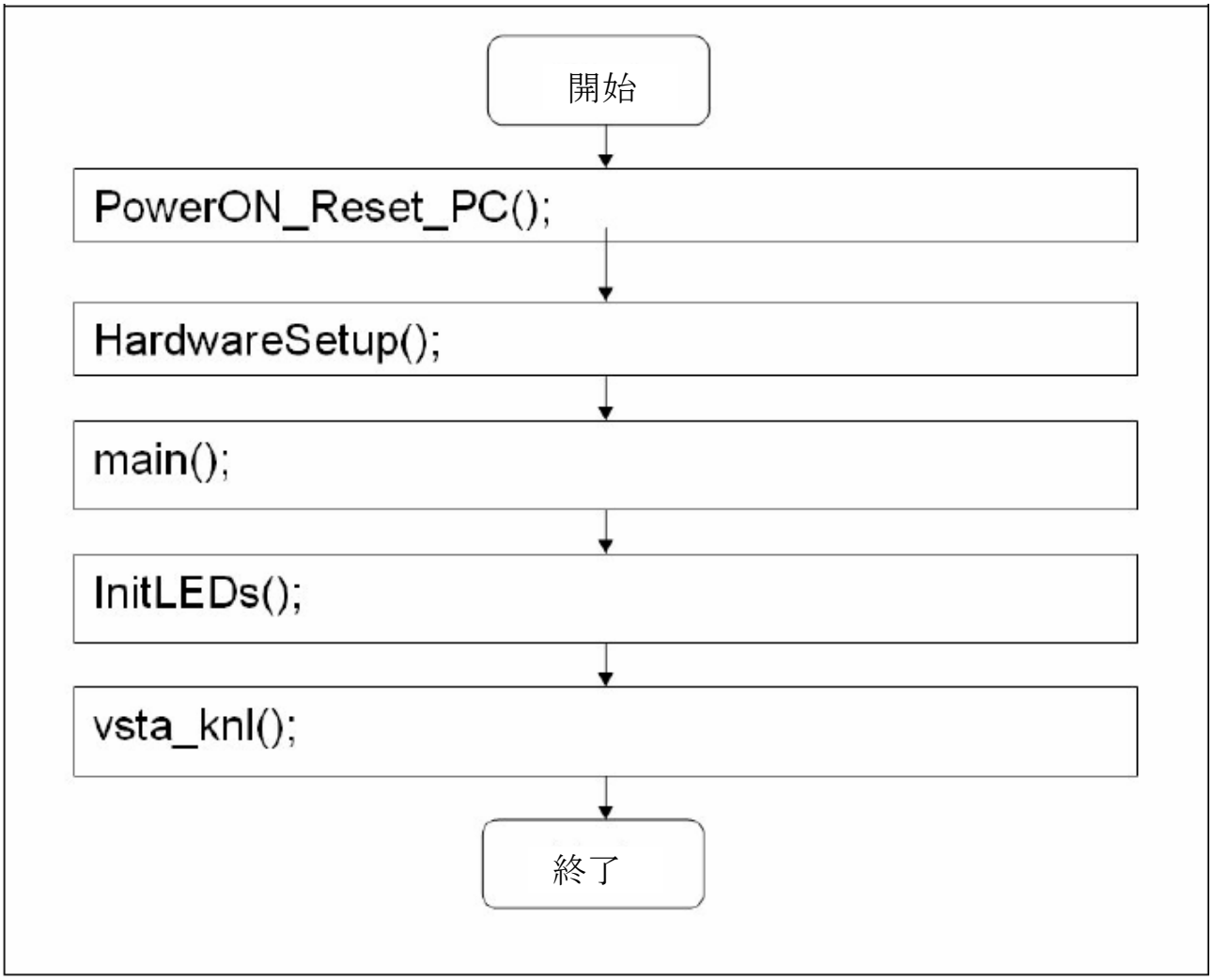
デフォルトのソースコードファイルには、以下が含まれています。

```
#include "LED_Blinky.h"


void main(void)
{
        // Initialize LEDs
    InitLEDs();

        // Start Kernel
    vsta_knl();

        // Should never get here.
    while(1)
    {
        nop();
    }
}
```

5. 点滅デモアプリケーションフロー

5.1 アプリケーションフロー

点滅アプリケーションフロー図は、次のようになります。



5.2　アプリケーションリソース

5.2.1　ハードウェアリソース

点滅デモアプリケーションのハードウェア設定は、次のようになります。

入力クロック: 12 MHz
システムクロック（ICLK）: 96 MHz
周辺モジュールクロック（PCLK）: 48 MHz
外部バスクロック（BCLK）と SDRAM クロック（SDCLK）: 24 MHz

5.2.2　アプリケーションリソース

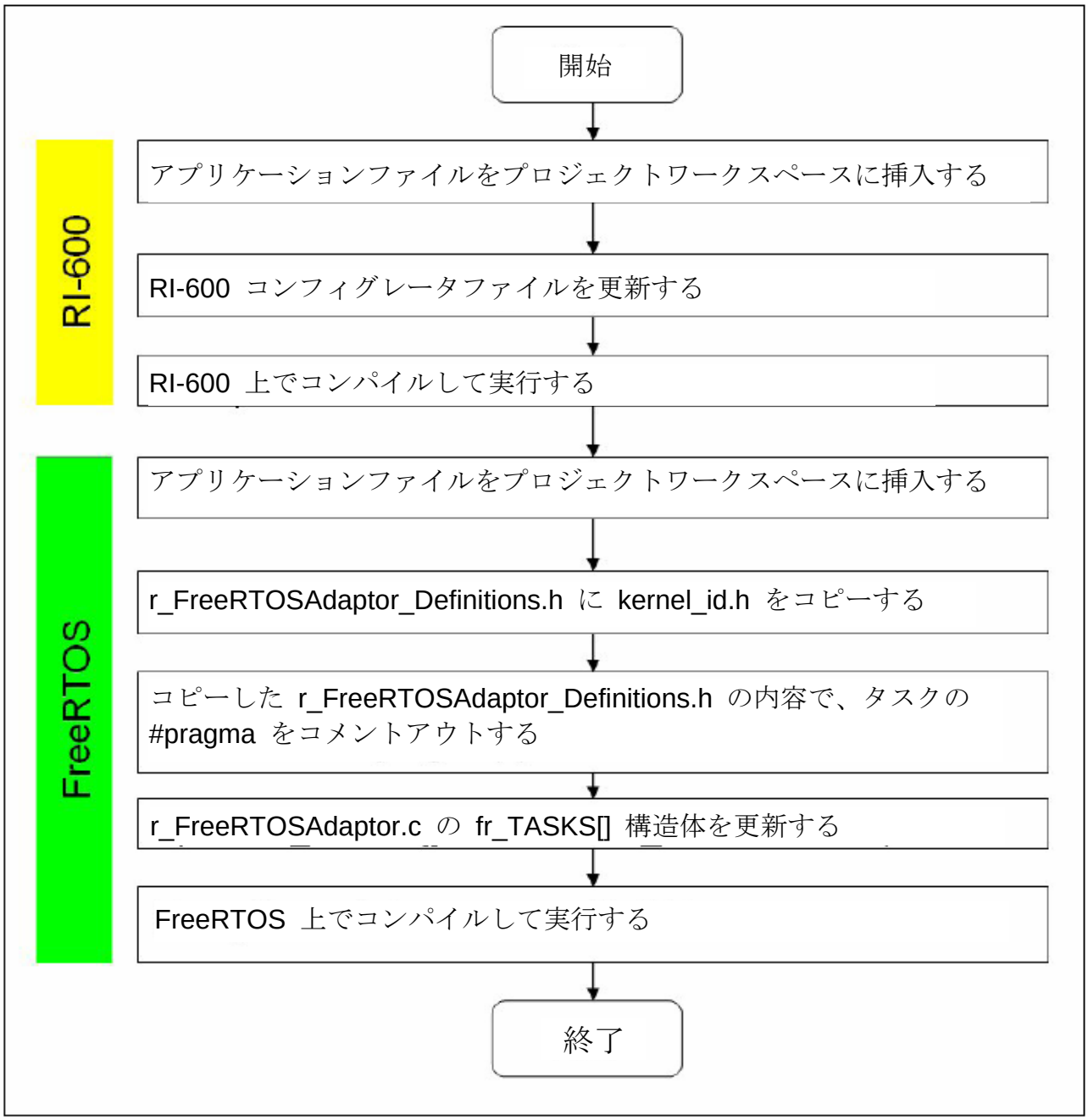
点滅デモアプリケーションの特性は、次のとおりです。

タスク数: 1（LED_TASK）
使用する RTOS API:vsta_knl()、dly_tsk()

これらの関数の詳細については、アダプタレイヤーユーザーマニュアルおよび API 仕様書に記載されています。

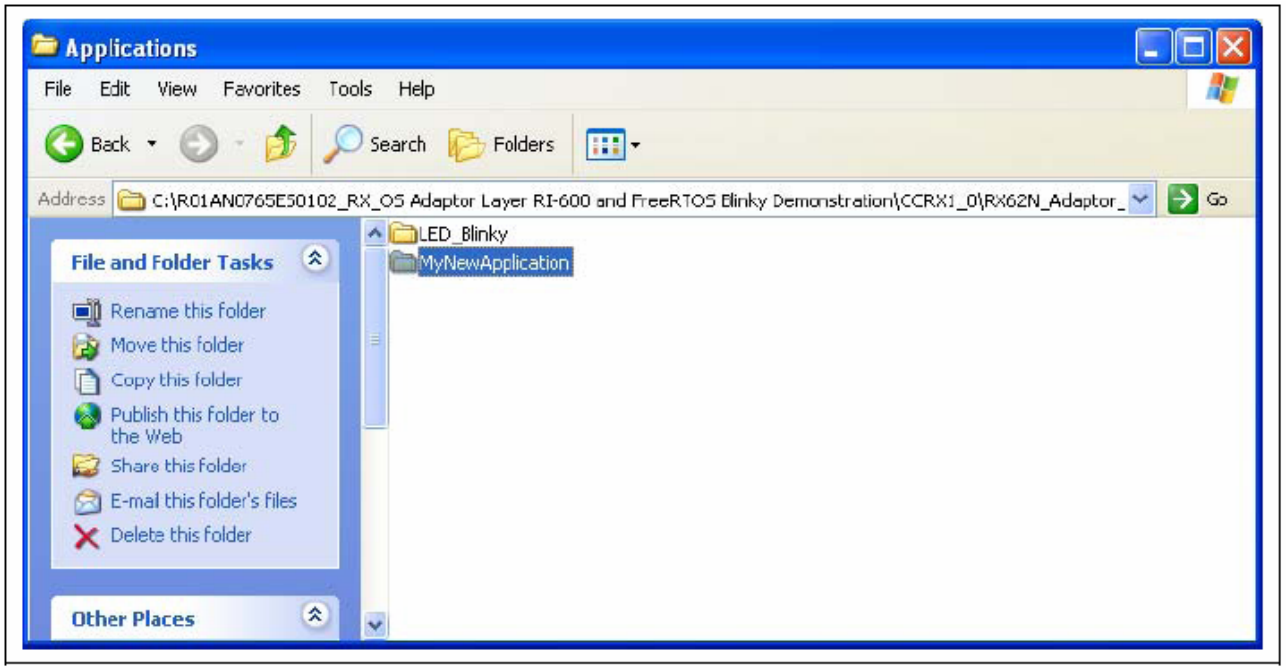
6. アプリケーション作成プロセスのワークフロー

- 以下の手順は、両方の RTOS で動作するアプリケーションの作成の概要について説明しています。

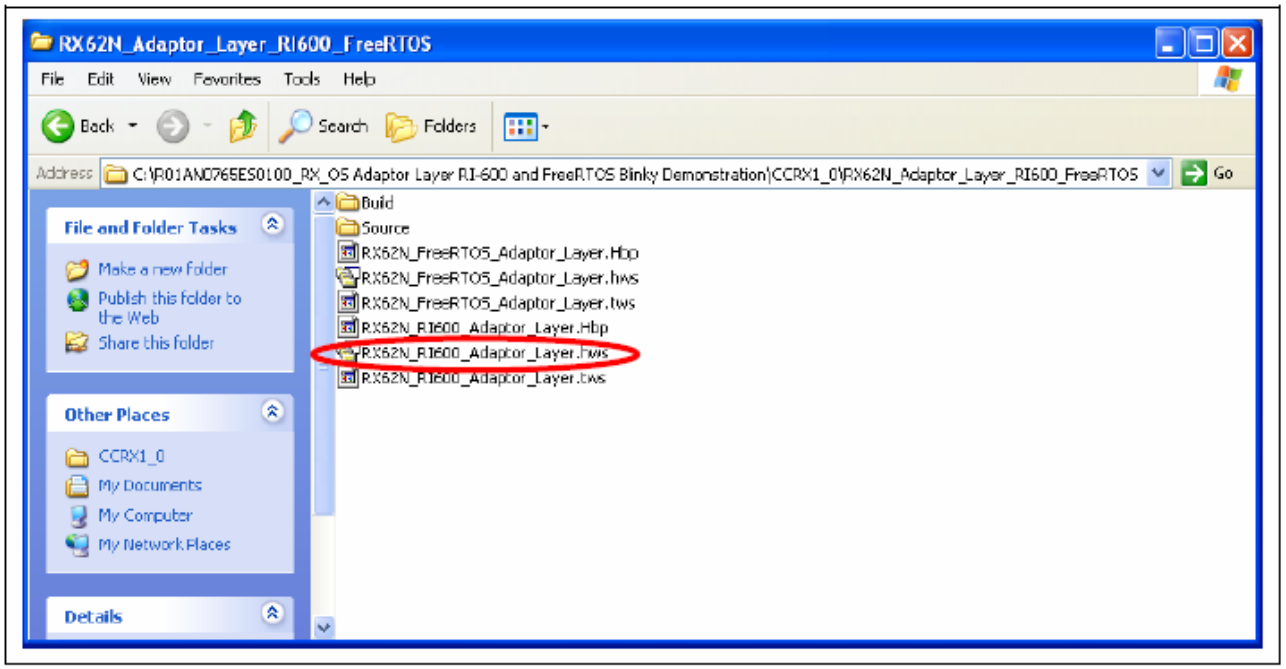


6.1　RI-600: RI-600 用アプリケーションを作成する

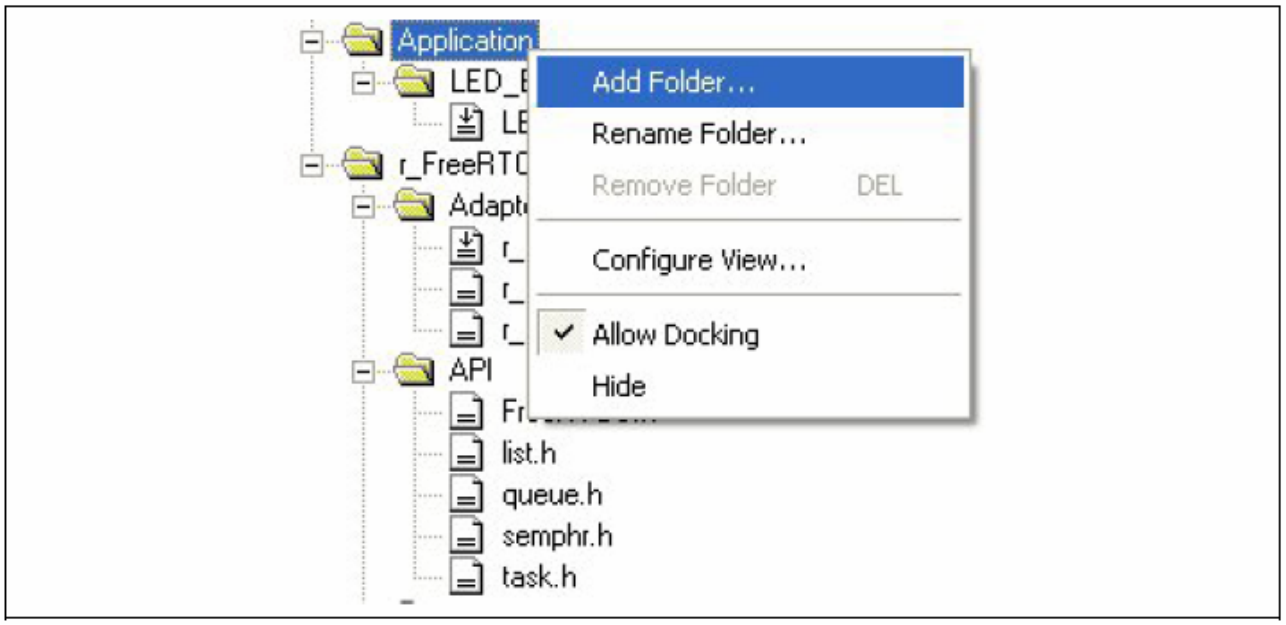
- 最初に、ワークスペースディレクトリの ¥Source¥Applications の下にアプリケーションディレクトリを作成します。



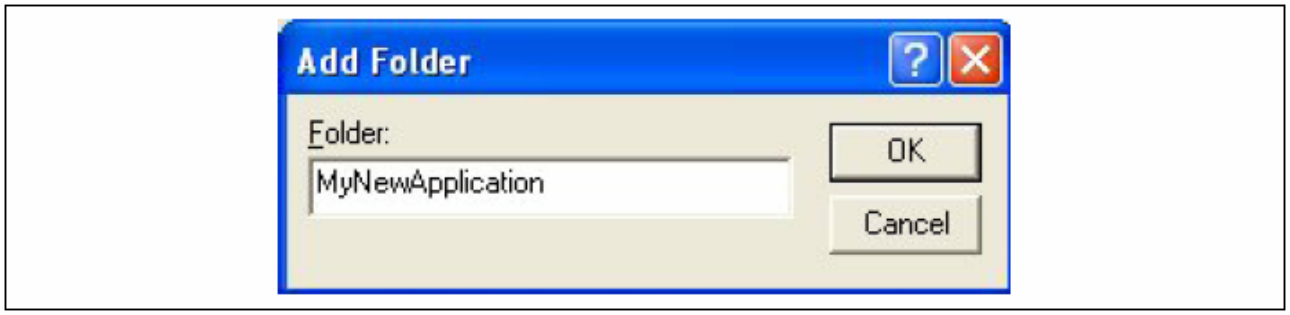
- 以下に示すように、RX62N_RI600_Adaptor_Layer.hws ワークスペースを開きます。



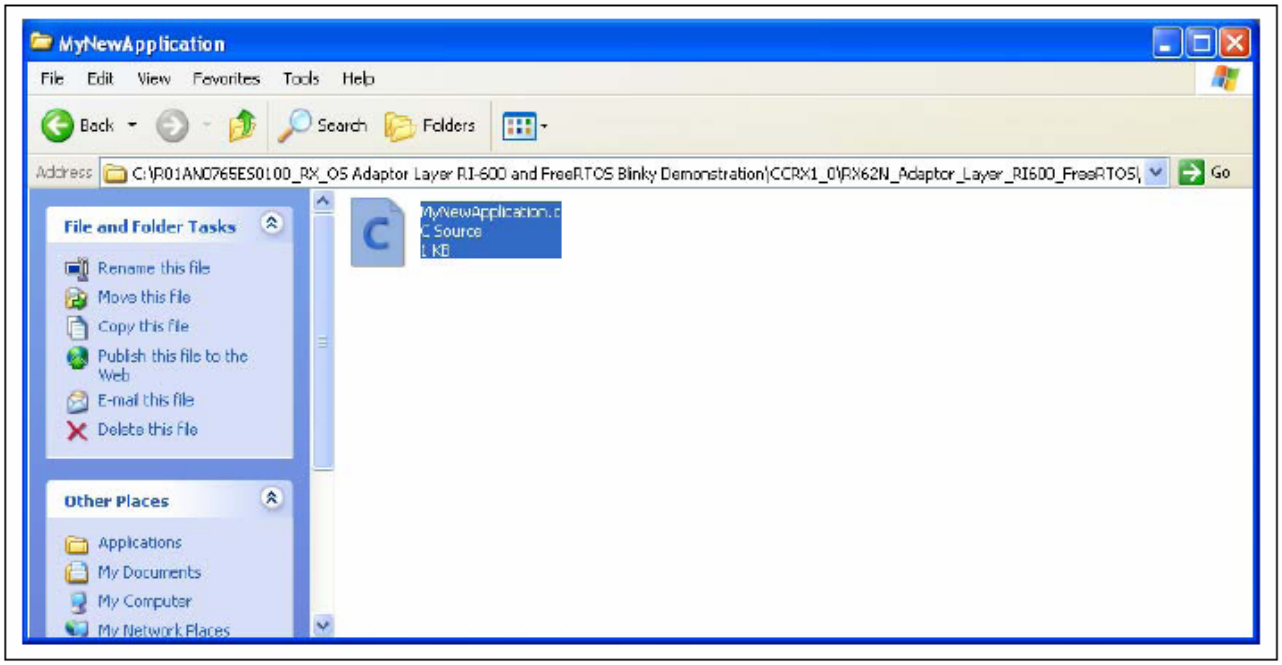
- デフォルトの設定を受け入れてボードに接続し、必要なフォルダをワークスペースに追加します。



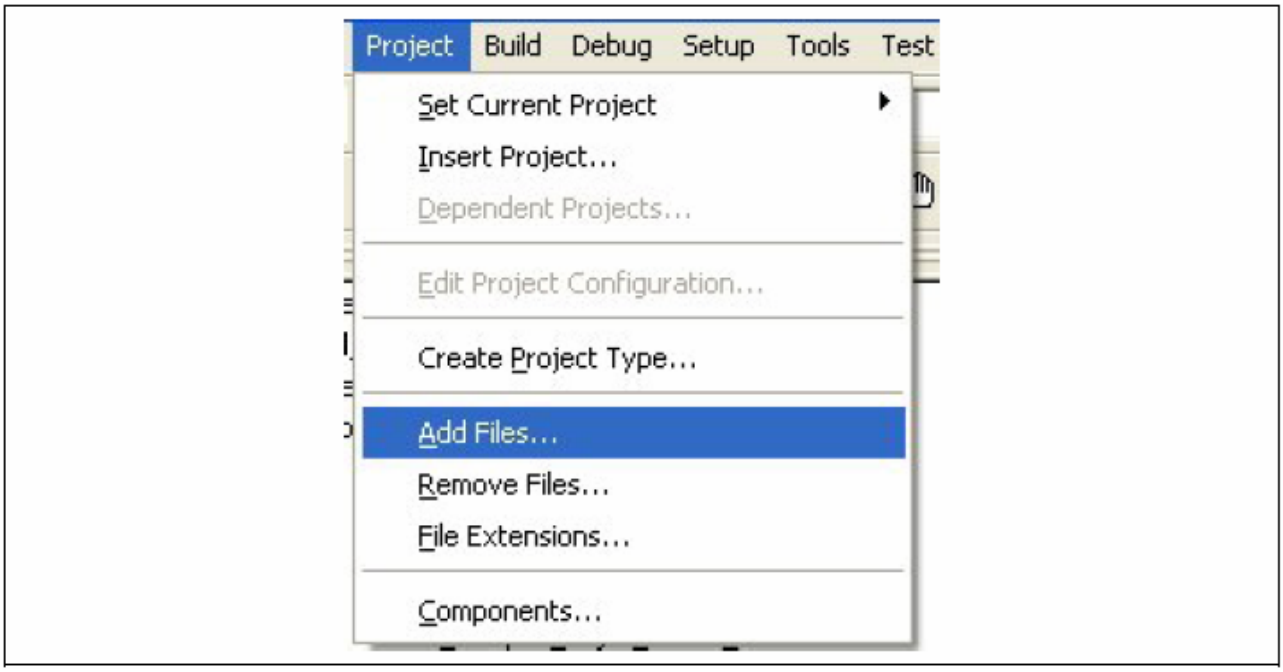
- 適宜、フォルダに名前を付けます。



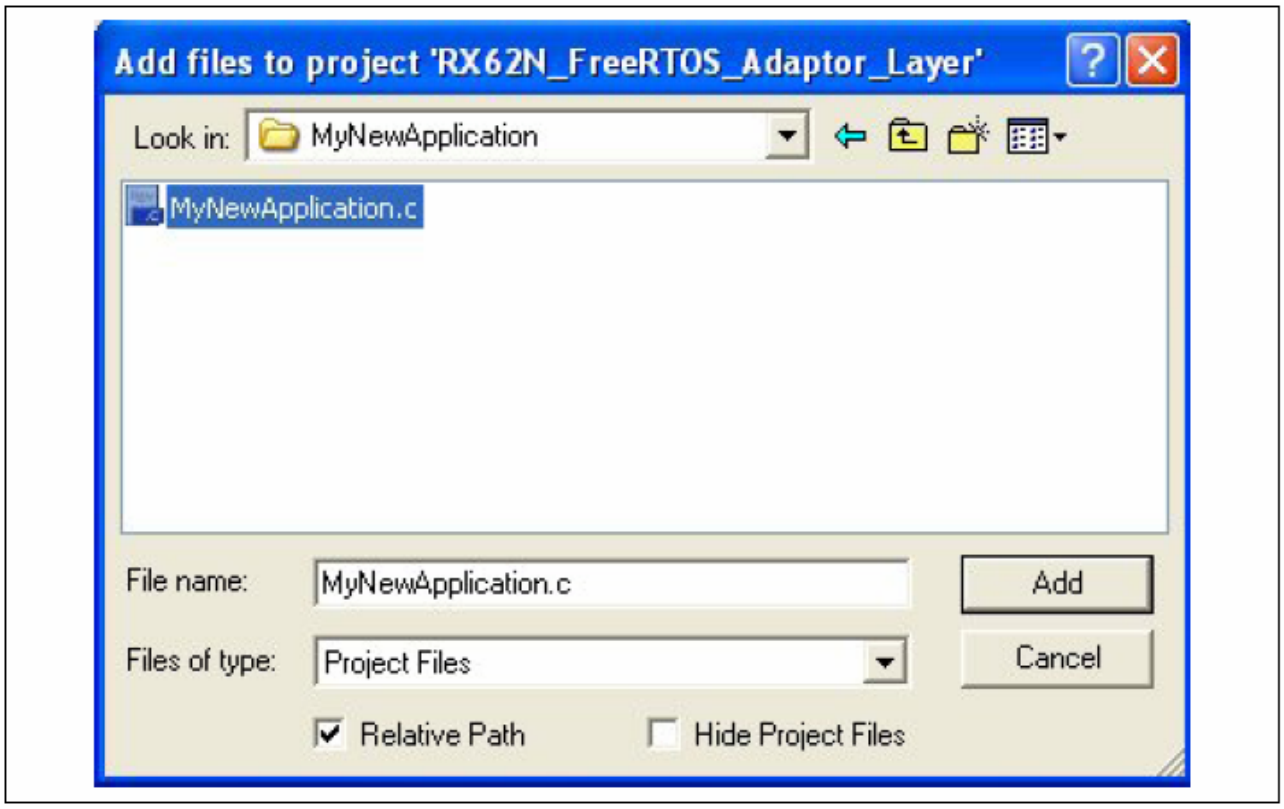
- ¥Source¥Applications¥MyNewApplication フォルダに「MyNewApplication.c」という名前の新しい C コードファイルを作成します。



- 以下のように、この新しい C コードをワークスペースに追加します。



- ファイルを選択して追加します。

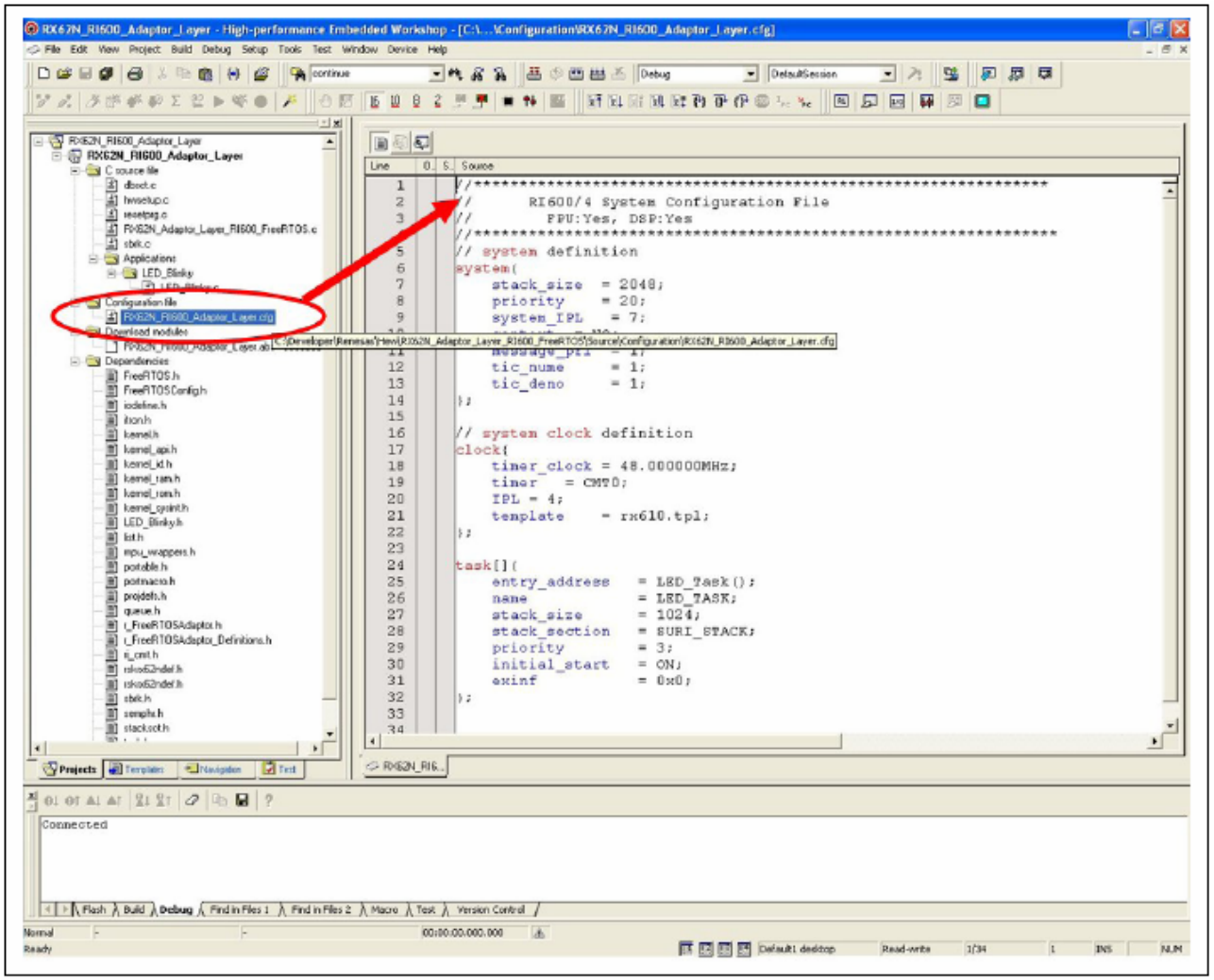


- 以下のように、ワークスペースにコードを配置します。

- MyNewApplication.c を開いて、以下のコードをファイルに追加します。

```
/***********************************************************************************
*           Conditional Compilation Differences.
***********************************************************************************/
#if defined (__ADAPTOR_RI_600__)
      #include "kernel.h"
      #include "kernel_id.h"
      #elif defined (__ADAPTOR_FREE_RTOS__)
      #include "r_FreeRTOSAdaptor.h"
#endif

#include "iodefine.h"
#include "rskrx62ndef.h"

/***********************************************************************************
* Function Name    : LED_Task2();
* Description            : This is the LED Task that will execute the Application
*                                code.
* Argument               : None.
* Return Value           : None.
***********************************************************************************/
void LED_Task2(VP_INT param)
{
      for (;;){
            LED0 = ~LED0;

            dly_tsk(200);
      }
}
```

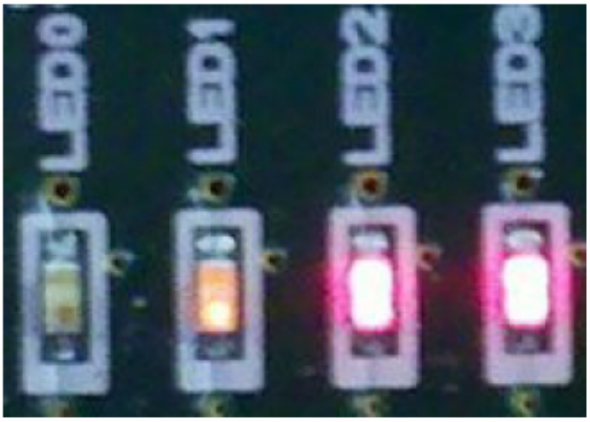
- 以下に示すように、コンフィグレータファイル: RX62N_RI600_Adaptor_Layer.cfg を開きます。



- 以下に示すように、次のコンフィグレーションコードを付け加えます。

```
task[]{
      entry_address     = LED_Task2();
      name              = LED_TASK2;
      stack_size        = 1024;
      stack_section     = SURI_STACK;
      priority          = 3;
      initial_start     = OFF;
      exinf             = 0x0;
};
```

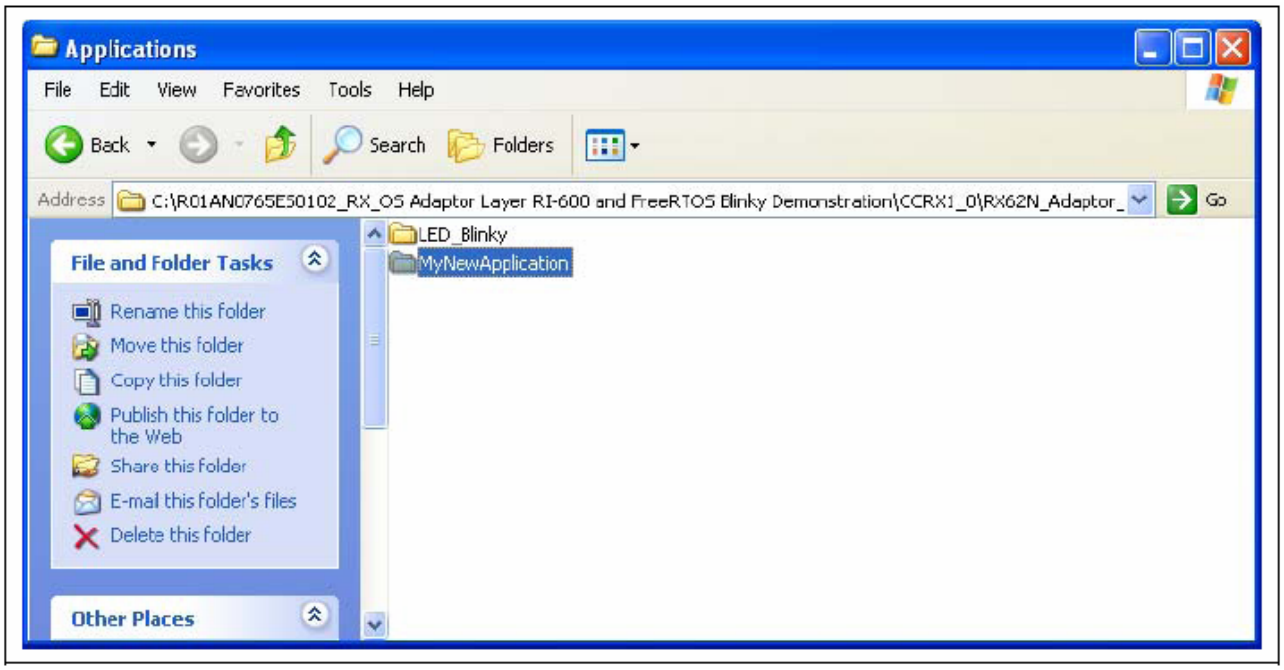
- [Build] → [Build All] を選択して、ワークスペースプロジェクトをコンパイルしてビルドします。次に、モジュールをダウンロードして実行します。RX62N RSK ボード上の LED の点滅は、次のようになります。



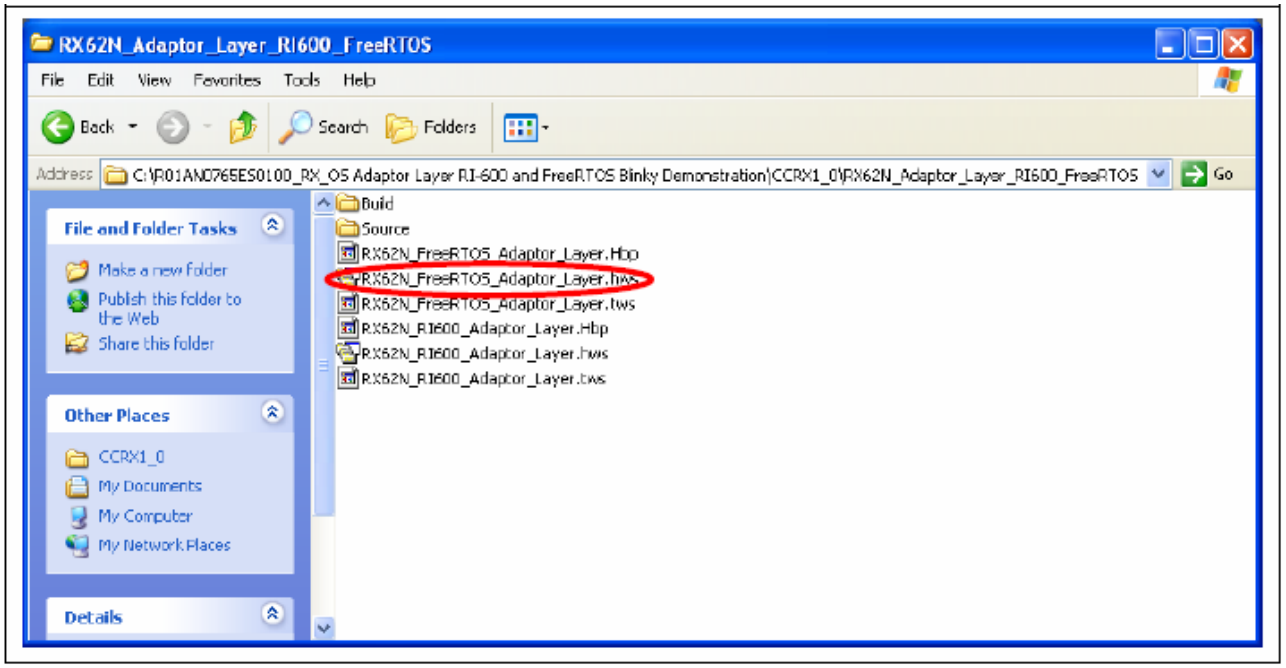
このとき、LED0 は 200ms の遅れで点滅し、LED1 ～ LED 3 は 500ms の遅れで点滅します。

6.2　FreeRTOS™: FreeRTOS™ 用アプリケーションを作成する

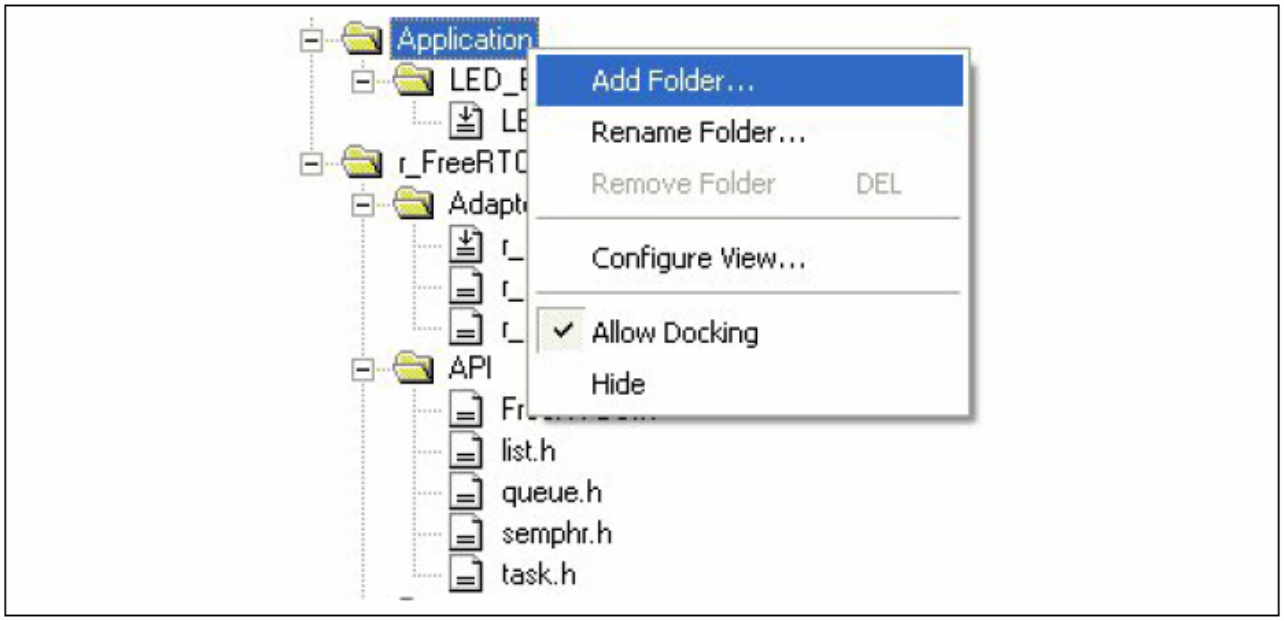
- 最初に、ワークスペースディレクトリの ¥Source¥Applications の下にアプリケーションディレクトリを作成します。



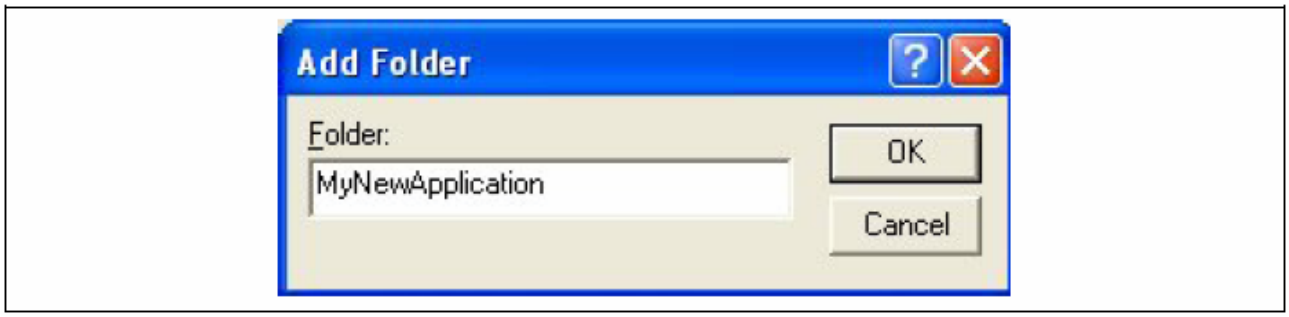
- 以下に示すように、RX62N_FreeRTOS_Adaptor_Layer.hws ワークスペースを開きます。



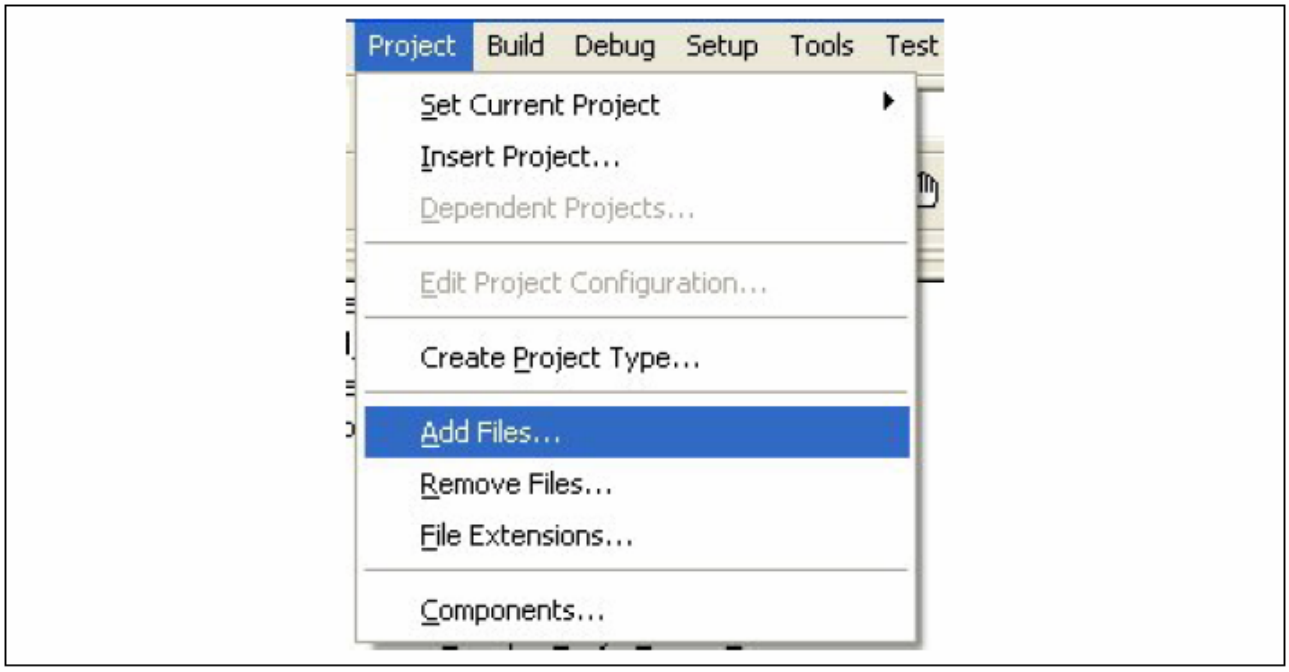
- デフォルトの設定を受け入れてボードに接続します。必要なフォルダをワークスペースに追加します。



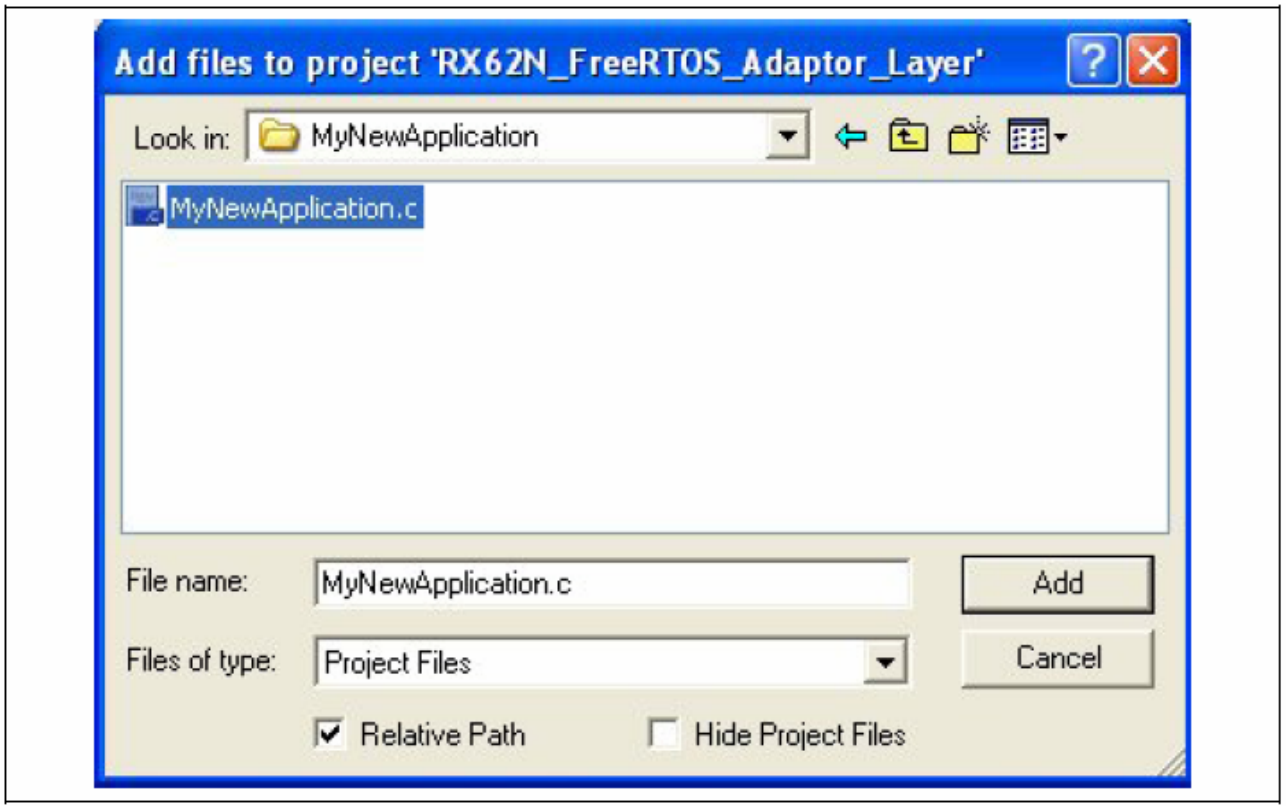
- 適宜、フォルダに名前を付けます。



- 以下のように、¥Source¥Applications¥MyNewApplication フォルダに「MyNewApplication.c」という名前の新しい C コードファイルを作成し、ワークスペースに追加します。



- ファイルを選択して追加します。



- 以下のように、ワークスペースにコードを配置します。

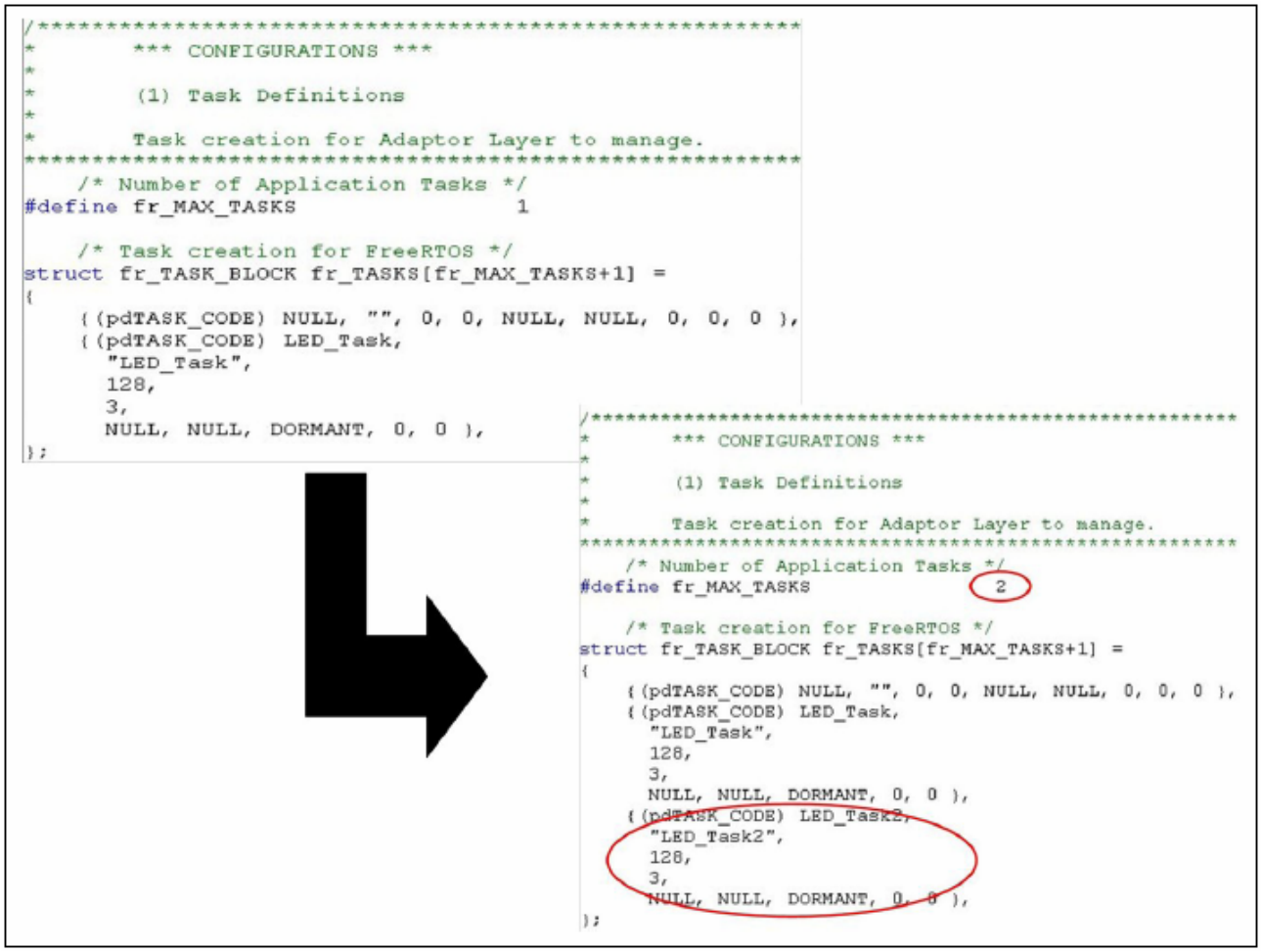
- kernel_id.h が生成したすべてのコードをコピーし、¥Source¥FreeRTOS¥Adaptor¥r_FreeRTOSAdaptor_Definitions.h に貼り付けます。これは、次のように表示されます。

- r_FreeRTOSAdaptor_Definitions.h に貼り付けたコード内で、関数タスクコードを見つけて、以下に示すように、すべてのコンパイラ指示文 #pragma をコメントアウトします。

```
/*-----------------------------------------------------*/
#define _RI_MAX_DTQ       0
#define  VTMAX_DTQ        _RI_MAX_DTQ
/*-----------------------------------------------------*/
#define _RI_STK_SIZE_LED_TASK     0x400UL
#define _RI_STK_SIZE_LED_TASK2    0x400UL
/*-----------------------------------------------------*/
void LED_Task( VP_INT );
//#pragma task    LED_Task

void LED_Task2( VP_INT );
//#pragma task    LED_Task2

/*-----------------------------------------------------*/
/*-----------------------------------------------------*/
/*-----------------------------------------------------*/
/*-----------------------------------------------------*/
/*-----------------------------------------------------*/
```

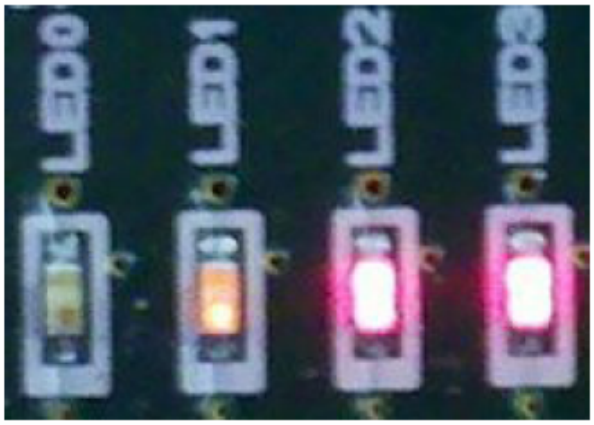
FreeRTOS™ はタスクにこのようなコンパイラ指示文を使用しないため、このステップが必要になります。

- r_FreeRTOSAdaptor.c ファイルを開き、(1) Task Definitions セクションまでスクロールし、以下のように fr_TASKS[] 構造体を更新します。



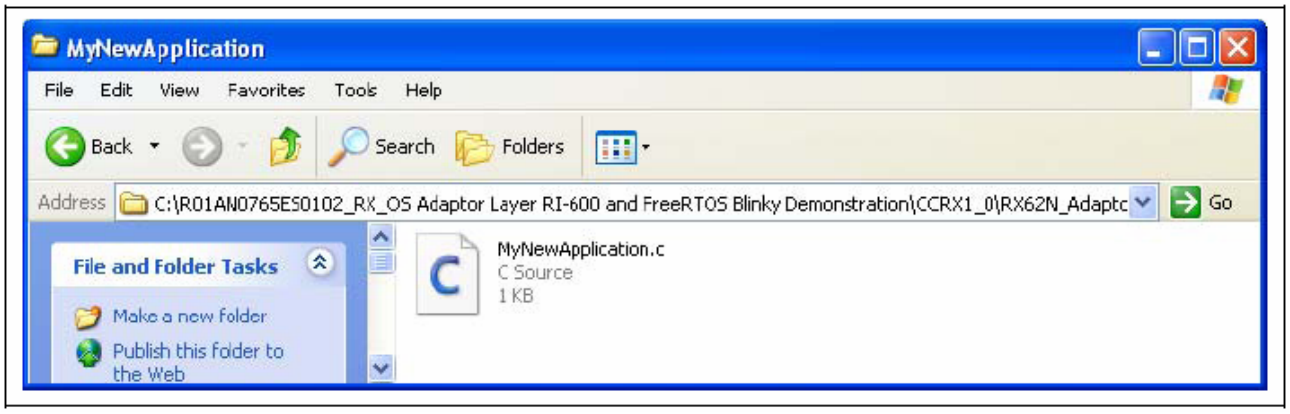
fr_MAX_TASKS が 2 に更新されていて、LED_Task2 の初期化が作成されていることに注意してください。このタスク構造体の初期化は、アダプタレイヤーがタスクを管理するのに必要です。詳細については、アダプタレイヤーユーザーマニュアルおよび API 仕様アプリケーションノートを参照してください。

- [Build] → [Build All] を選択して、ワークスペースプロジェクトをコンパイルしてビルドします。次に、モジュールをダウンロードして実行します。RX62N RSK ボード上の LED の点滅は、次のようになります。



6.3　RI-600: アプリケーションファイルの挿入

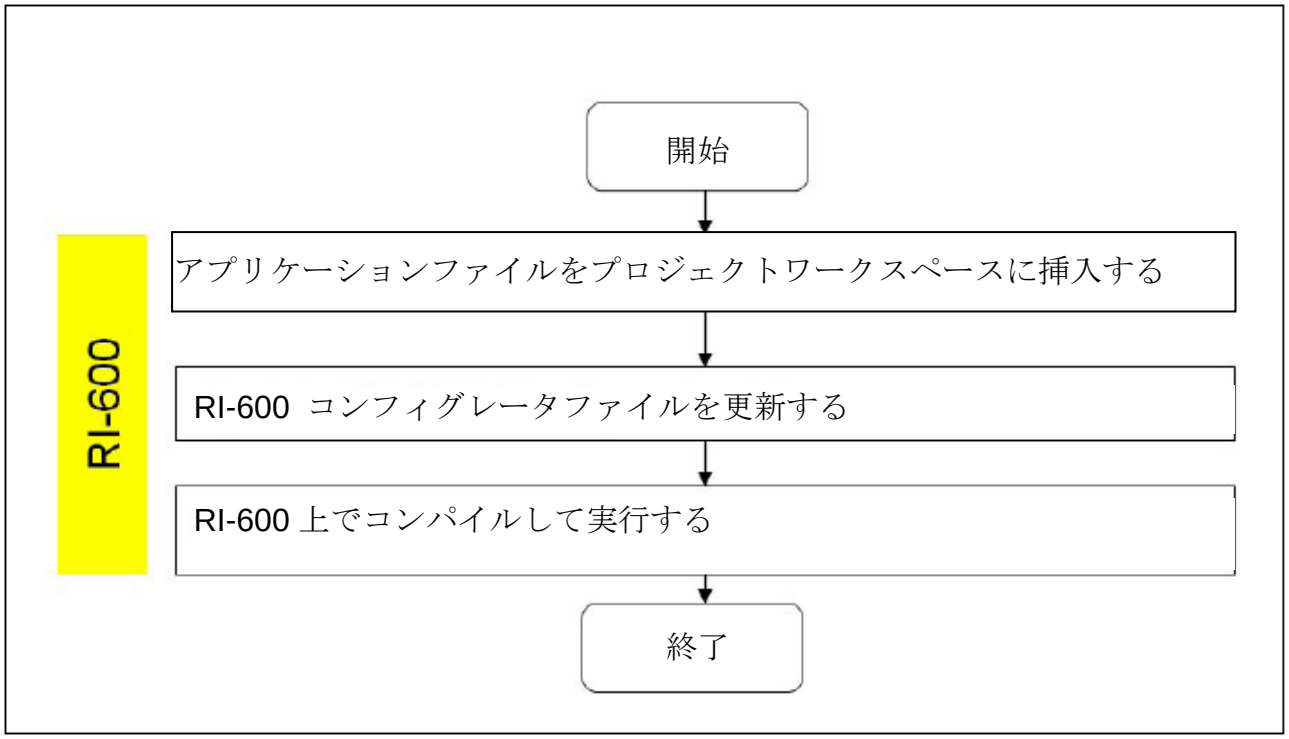
- この節では、¥Source¥Applications¥MyNewApplication にある「MyNewApplication.c」という名前の新しいアプリケーションを挿入する方法を示します。これは、以下に示すとおりです。



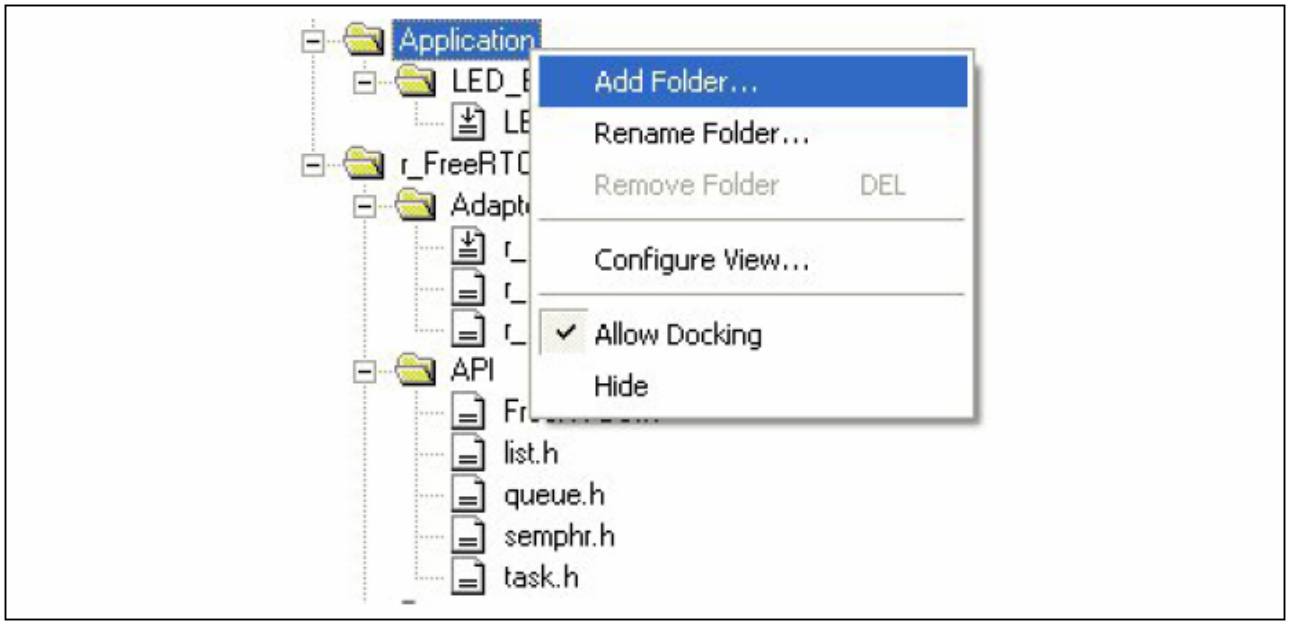
- 「MyNewApplication.c」内のコードは次のとおりです。

```
/***********************************************************************************
*           Conditional Compilation Differences.
***********************************************************************************/
#if defined (__ADAPTOR_RI_600__)

      #include "kernel.h"
      #include "kernel_id.h"

#elif defined (__ADAPTOR_FREE_RTOS__)

      #include "r_FreeRTOSAdaptor.h"

#endif

#include "iodefine.h"
#include "rskrx62ndef.h"

/***********************************************************************************
* Function Name    : LED_Task2();
* Description           : This is the LED Task that will execute the Application
*                              code.
* Argument              : None.
* Return Value          : None.
***********************************************************************************/
void LED_Task2(VP_INT param)
{
      for (;;){

            LED0 = ~LED0;

            dly_tsk(200);
      }
}
```

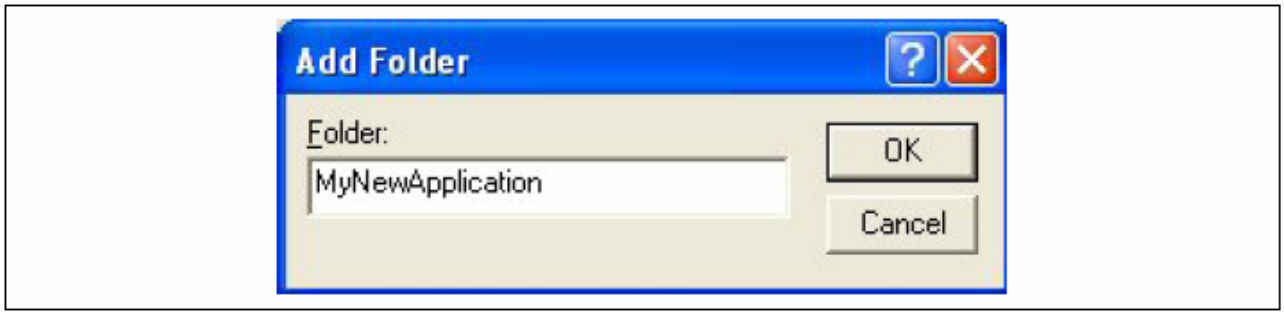
- RI-600 ワークスペースに追加するアプリケーションは、この手順の順序に従う必要があります。



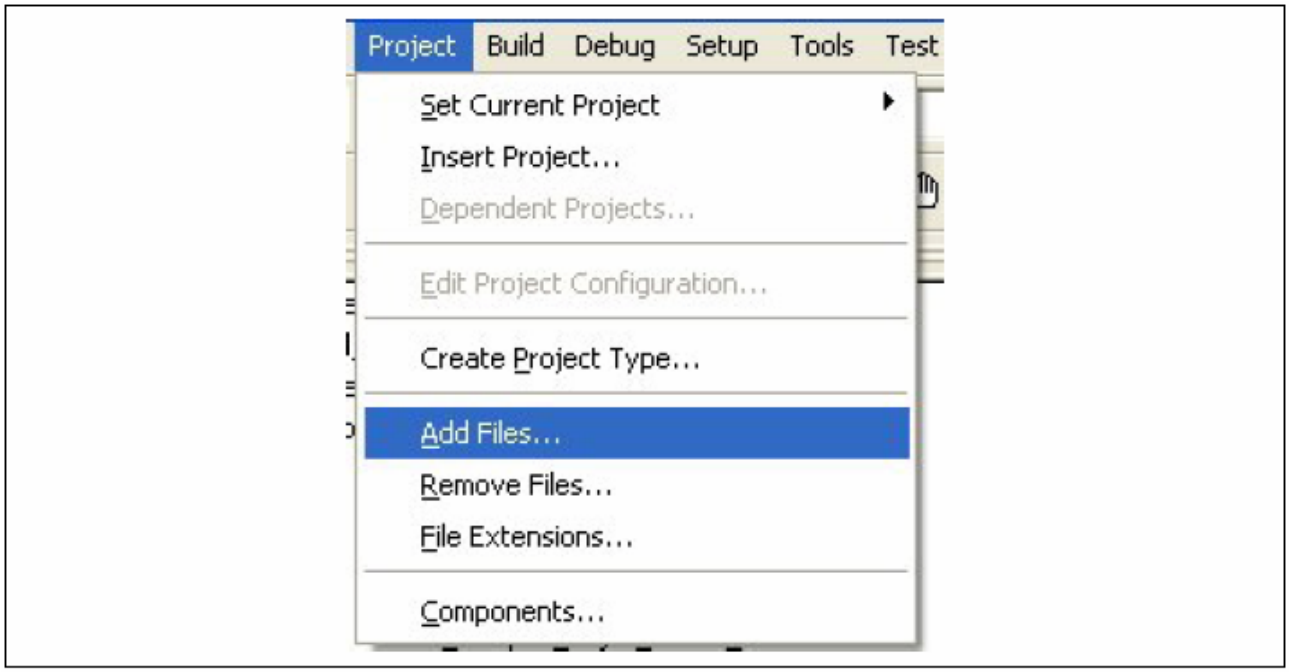
- デフォルトの設定を受け入れてボードに接続し、必要なフォルダをワークスペースに追加します。



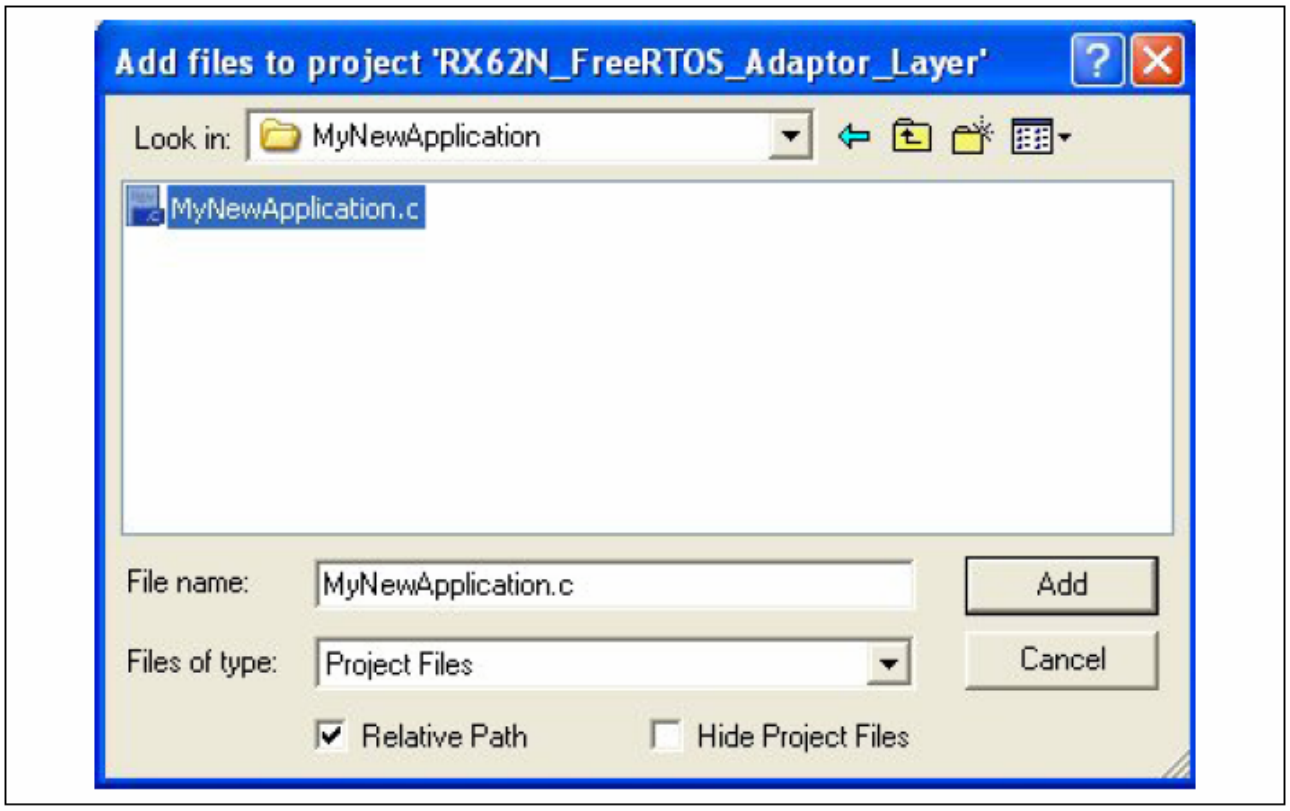
- 適宜、フォルダに名前を付けます。



- 以下のように、¥Source¥Applications¥MyNewApplication フォルダの「MyNewApplication.c」という名前の新しい C コードファイルをワークスペースに追加します。



- ファイルを選択して追加します。



- 以下のように、ワークスペースにコードを配置します。

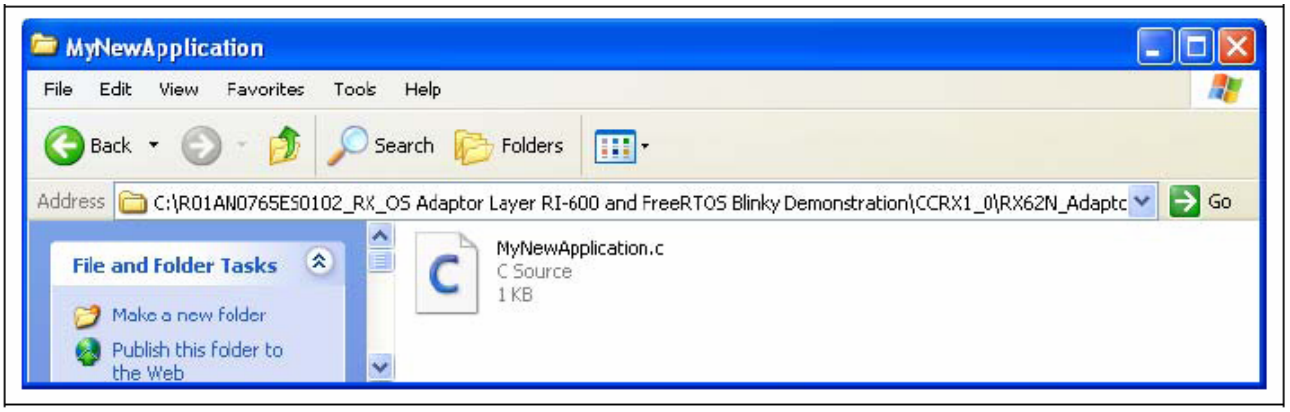
- RX62N_RI600_AdaptorLayer.cfg を開き、以下のコードを付け加えて新しいタスクを作成します。

```
task[]{
      entry_address     = LED_Task2();
      name              = LED_TASK2;
      stack_size        = 1024;
      stack_section     = SURI_STACK;
      priority          = 3;
      initial_start     = OFF;
      exinf             = 0x0;
};
```

- [Build] → [Build All] を選択してワークスペースプロジェクトをコンパイルしてビルドします。次に、モジュールをダウンロードして実行します。

6.4　FreeRTOS™: アプリケーションファイルの挿入

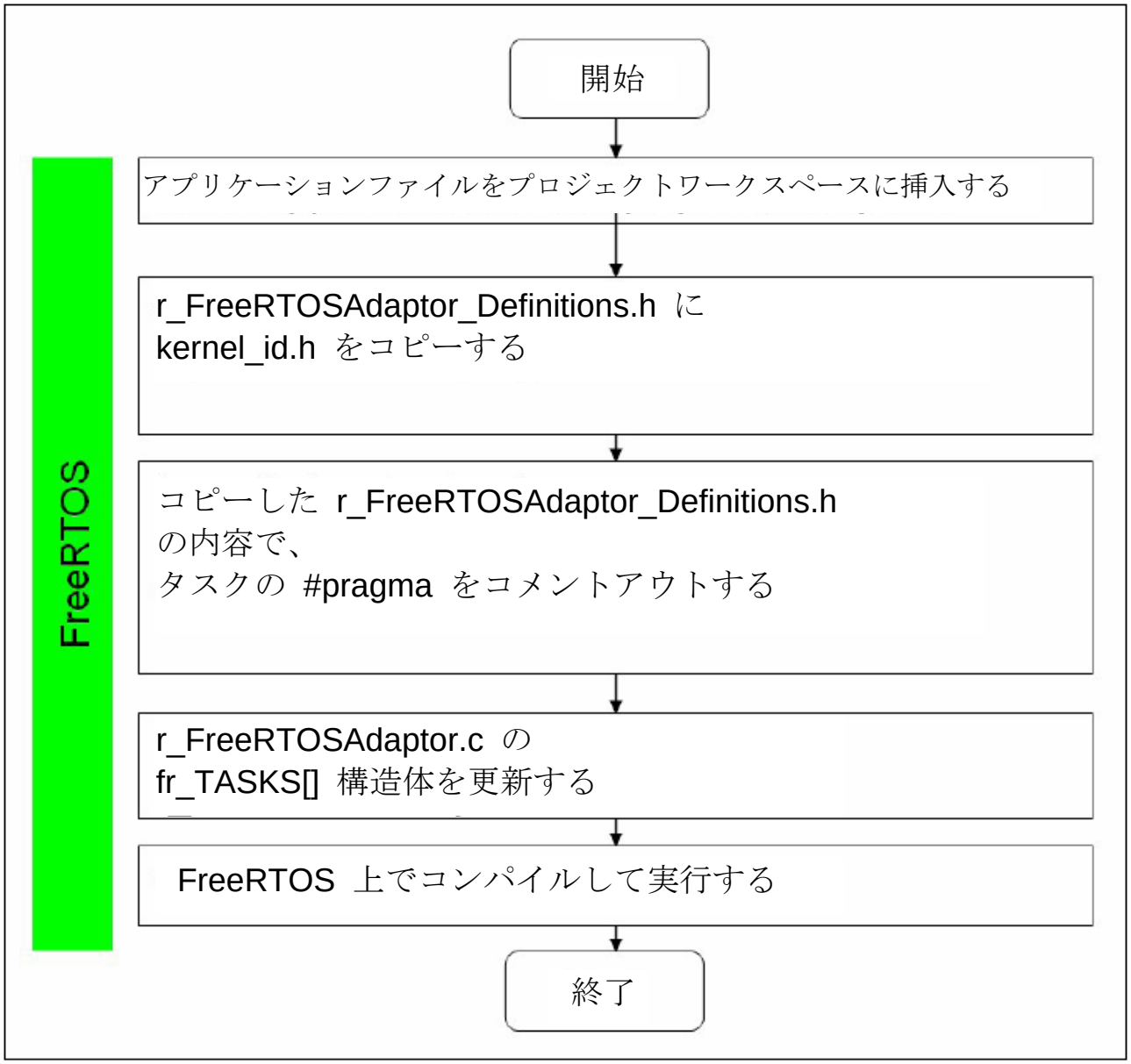
- この節では、¥Source¥Applications¥MyNewApplication にある「MyNewApplication.c」という名前の新しいアプリケーションを挿入する方法を示します。これは、以下に示すとおりです。



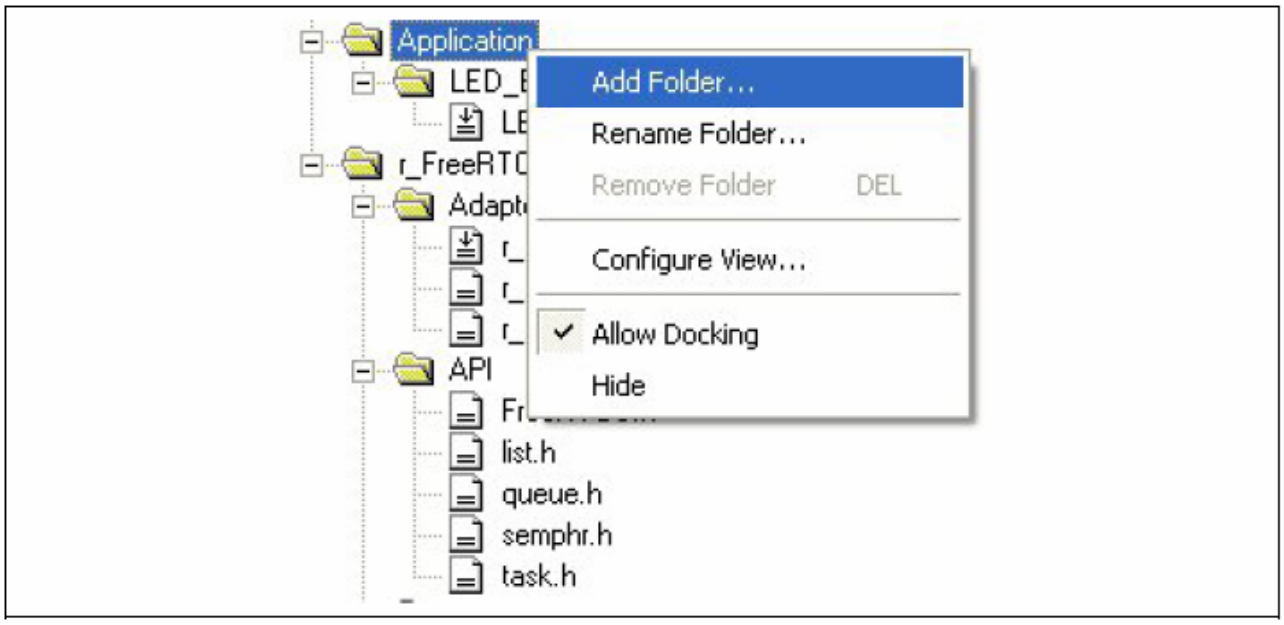
- 「MyNewApplication.c」内のコードは次のとおりです。

```
/***********************************************************************************
*           Conditional Compilation Differences.
***********************************************************************************/
#if defined (__ADAPTOR_RI_600__)

      #include "kernel.h"
      #include "kernel_id.h"

#elif defined (__ADAPTOR_FREE_RTOS__)

      #include "r_FreeRTOSAdaptor.h"

#endif

#include "iodefine.h"
#include "rskrx62ndef.h"

/***********************************************************************************
* Function Name    : LED_Task2();
* Description           : This is the LED Task that will execute the Application
*                              code.
* Argument              : None.
* Return Value          : None.
***********************************************************************************/
void LED_Task2(VP_INT param)
{
      for (;;){

            LED0 = ~LED0;

            dly_tsk(200);
      }
}
```

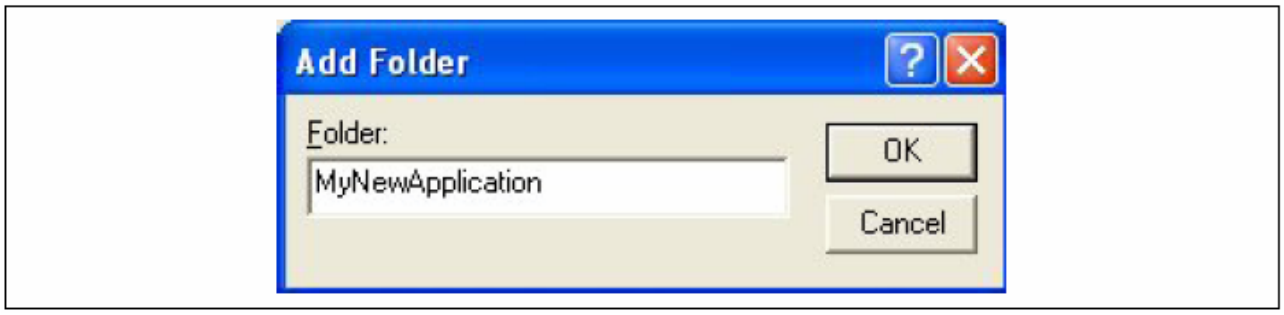
- RI-600 ワークスペースに追加するアプリケーションは、この手順の順序に従う必要があります。



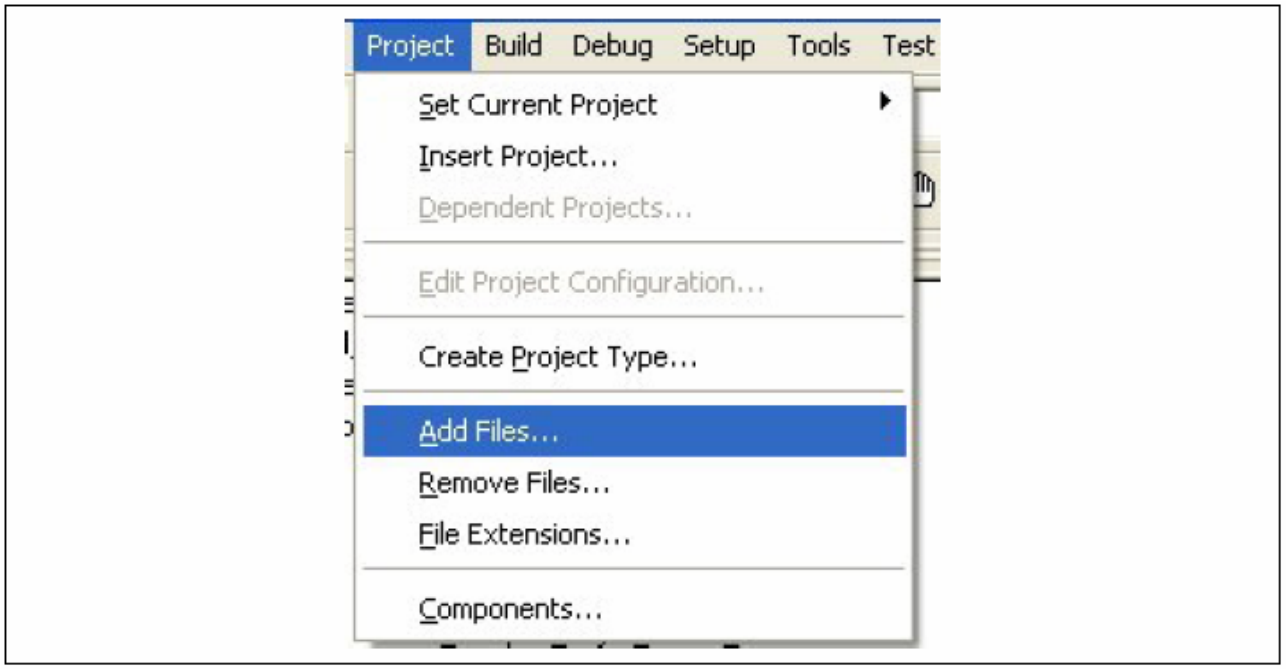
- デフォルトの設定を受け入れてボードに接続し、必要なフォルダをワークスペースに追加します。



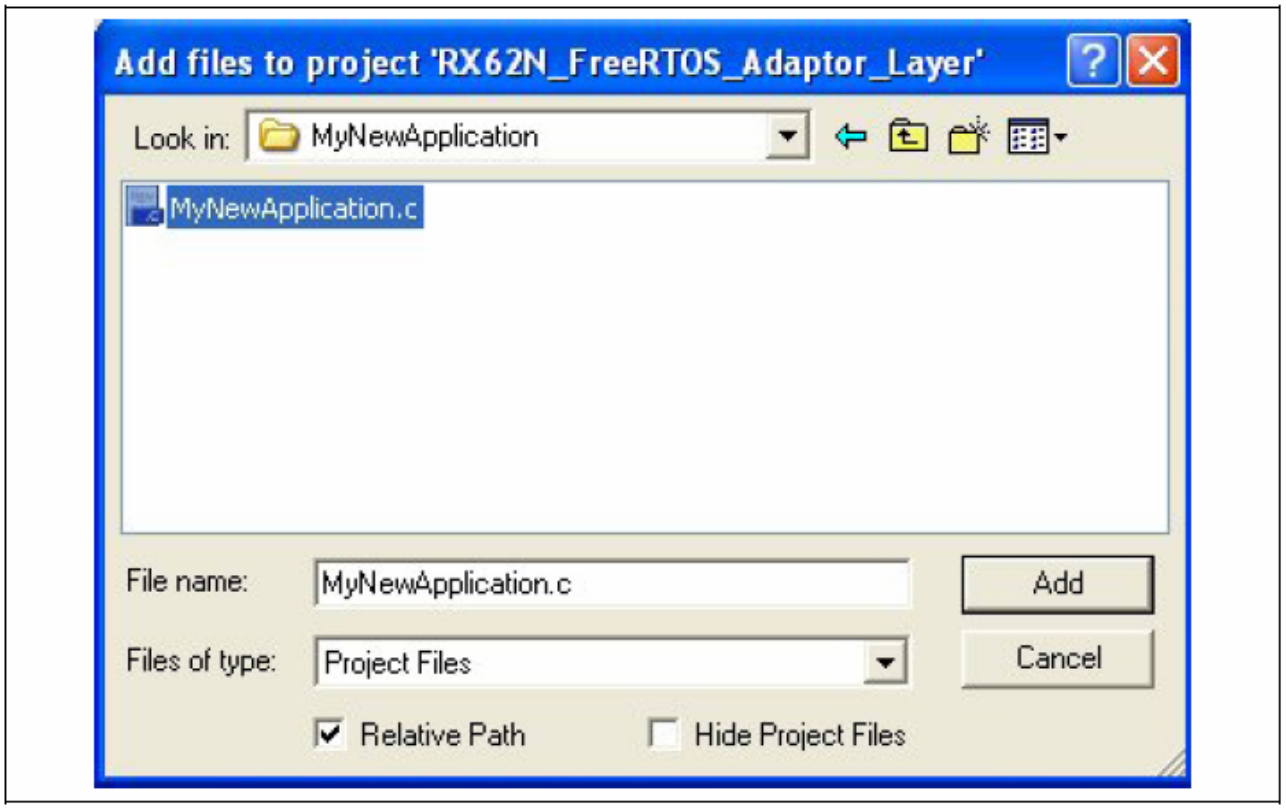
- 適宜、フォルダに名前を付けます。



- 以下のように、¥Source¥Applications¥MyNewApplication フォルダの「MyNewApplication.c」という名前の新しい C コードファイルをワークスペースに追加します。

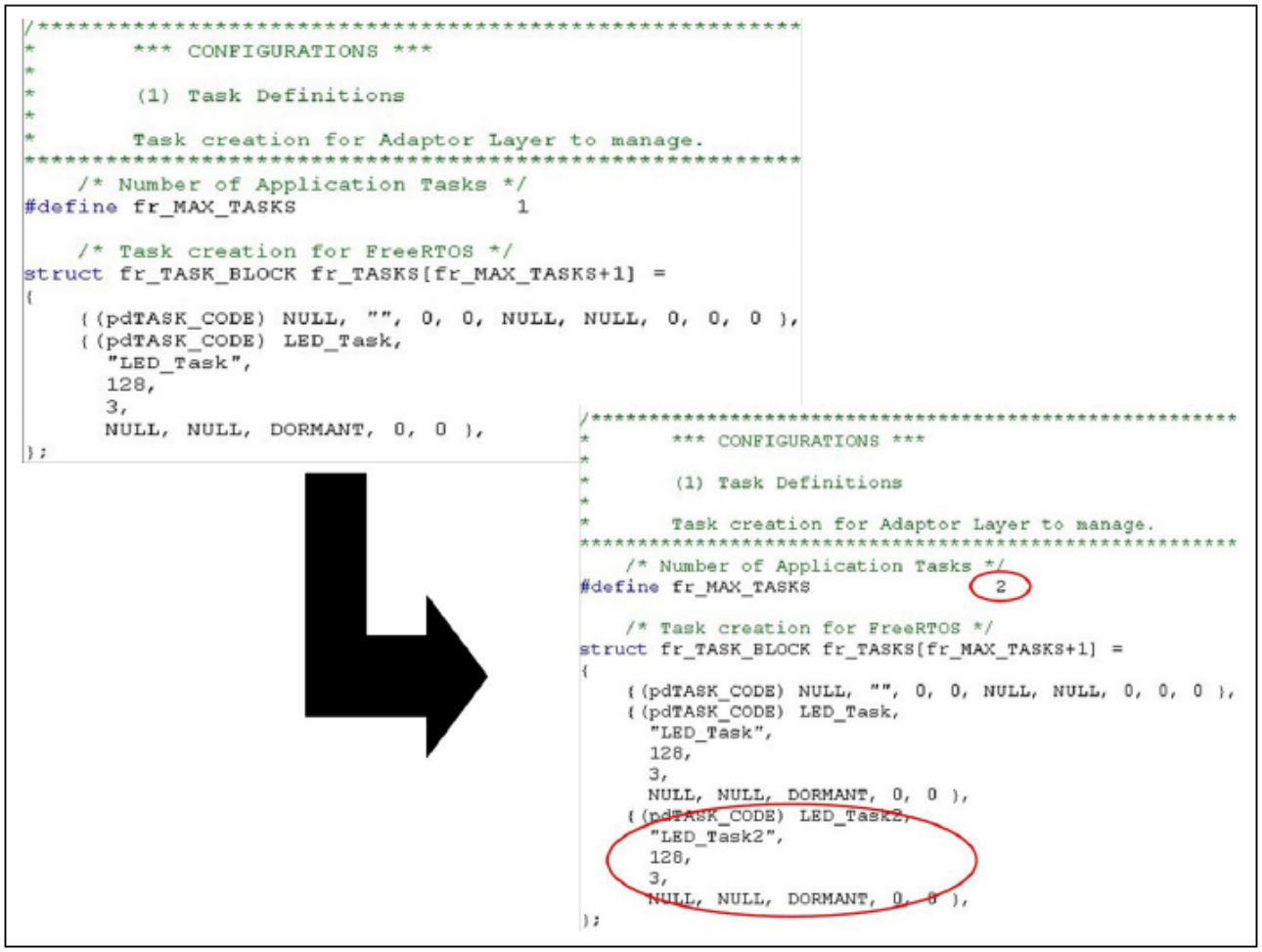


- ファイルを選択して追加します。



- 以下のように、ワークスペースにコードを配置します。

- r_FreeRTOSAdaptor.c ファイルを開き、(1) Task Definitions セクションまでスクロールし、以下のように fr_TASKS[] 構造体を更新します。



```
/*******************************************************
*        *** CONFIGURATIONS ***
*
*        (1) Task Definitions
*
*        Task creation for Adaptor Layer to manage.
*******************************************************
    /* Number of Application Tasks */
#define fr_MAX_TASKS                    1

    /* Task creation for FreeRTOS */
struct fr_TASK_BLOCK fr_TASKS[fr_MAX_TASKS+1] =
{
    {(pdTASK_CODE) NULL, "", 0, 0, NULL, NULL, 0, 0, 0 },
    {(pdTASK_CODE) LED_Task,
      "LED_Task",
      128,
      3,
      NULL, NULL, DORMANT, 0, 0 },
};
```

```
/*******************************************************
*        *** CONFIGURATIONS ***
*
*        (1) Task Definitions
*
*        Task creation for Adaptor Layer to manage.
*******************************************************
    /* Number of Application Tasks */
#define fr_MAX_TASKS                    2

    /* Task creation for FreeRTOS */
struct fr_TASK_BLOCK fr_TASKS[fr_MAX_TASKS+1] =
{
    {(pdTASK_CODE) NULL, "", 0, 0, NULL, NULL, 0, 0, 0 },
    {(pdTASK_CODE) LED_Task,
      "LED_Task",
      128,
      3,
      NULL, NULL, DORMANT, 0, 0 },
    {(pdTASK_CODE) LED_Task2,
      "LED_Task2",
      128,
      3,
      NULL, NULL, DORMANT, 0, 0 },
};
```

新しいタスクの追加に対応できるように、fr_MAX_TASKS は 2 に更新されていることに注意してください。タスクは、以下のように初期化する必要があります。

```
{(pdTASK_CODE) << function label >>,
   << task-name >>,
   << stack depth >>,
   << task priority level >>,
   NULL, NULL, DORMANT, 0, 0 }, //  デフォルトのままにしておく
```

- [Build] → [Build All] を選択して、ワークスペースプロジェクトをコンパイルしてビルドします。

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	RX62N グループ　アプリケーションノート　OS アダプタレイヤー: RI-600 と FreeRTOS の点滅デモンストレーション

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.07.15	—	初版発行
1.01	2012.06.27	—	RX コンパイラのバージョン 1.0 と 1.2 をサポート
		—	コンパイラオプションで、コンパイラの L セクションを C セクションに変換
1.02	2013.03.22	—	HEW と CCRX のコンパイラのバージョンを更新
		—	文書内にその他のディレクトリ構造を追加
		—	一般的注意と注記を更新

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、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。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）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、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お客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。お客様が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. 本資料に記載されている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大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き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、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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。

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。

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・電話番号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 エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■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：http://japan.renesas.com/contact/