RX62N Group

R01AN0765ES0102
Rev.1.02
Sep 05, 2012

OS Adaptor Layer: RI-600 and FreeRTOS™ Blinky Demonstration

Introduction

This application note presents a sample program that makes use of the Adaptor Layer for RI-600. The Adaptor Layer allows applications targeted for the RI-600 platform to be able to run on FreeRTOS™ without the need of reprogramming.

Target Device

- RX62N Group MCU (product number: R5FF562N8BDBG)

Target Board

- RX62N Renesas Start Kit + (product number: R0K5562N0C000BE)

Documents

- RI-600/4 V.1.00 User's Manual (product number: REJ10J2052-0100)
- Adaptor Layer User Manual and API Specification
- FreeRTOS (www.FreeRTOS.org)

Note: FreeRTOS™ is a trademark of Real Time Engineers Ltd

Contents

1. Introduction

1.1 Applicable Conditions

- MCU: RX62N Group
- Evaluation board: RX62N Renesas Starter Kit + (product number: R0K5562N0C000BE)
- Operating frequencies:

  Input clock: 12 MHz

  System clock (ICLK): 96 MHz

  Peripheral module clock (PCLK): 48 MHz

  External bus clock (BCLK) and SDRAM clock (SDCLK): 24 MHz
- Operating mode: S ingle-chip mode
- Integrated development encironment: Renesas Electronics High-performance Embedded Workshop, Ver. 4.09.01.007
- C compiler settings for Renesas RX Toolchain Version 1.0 are as follows:
  - Compile options:

  Little Endian operation:

  -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
- Optimizing linkage editor settings for Renesas RX Toolchain Version 1.0 are as follows:
  - Linker options:

  -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

- C compiler settings for Renesas RX Toolchain Version 1.2 are as follows:
  - Compile options:

  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

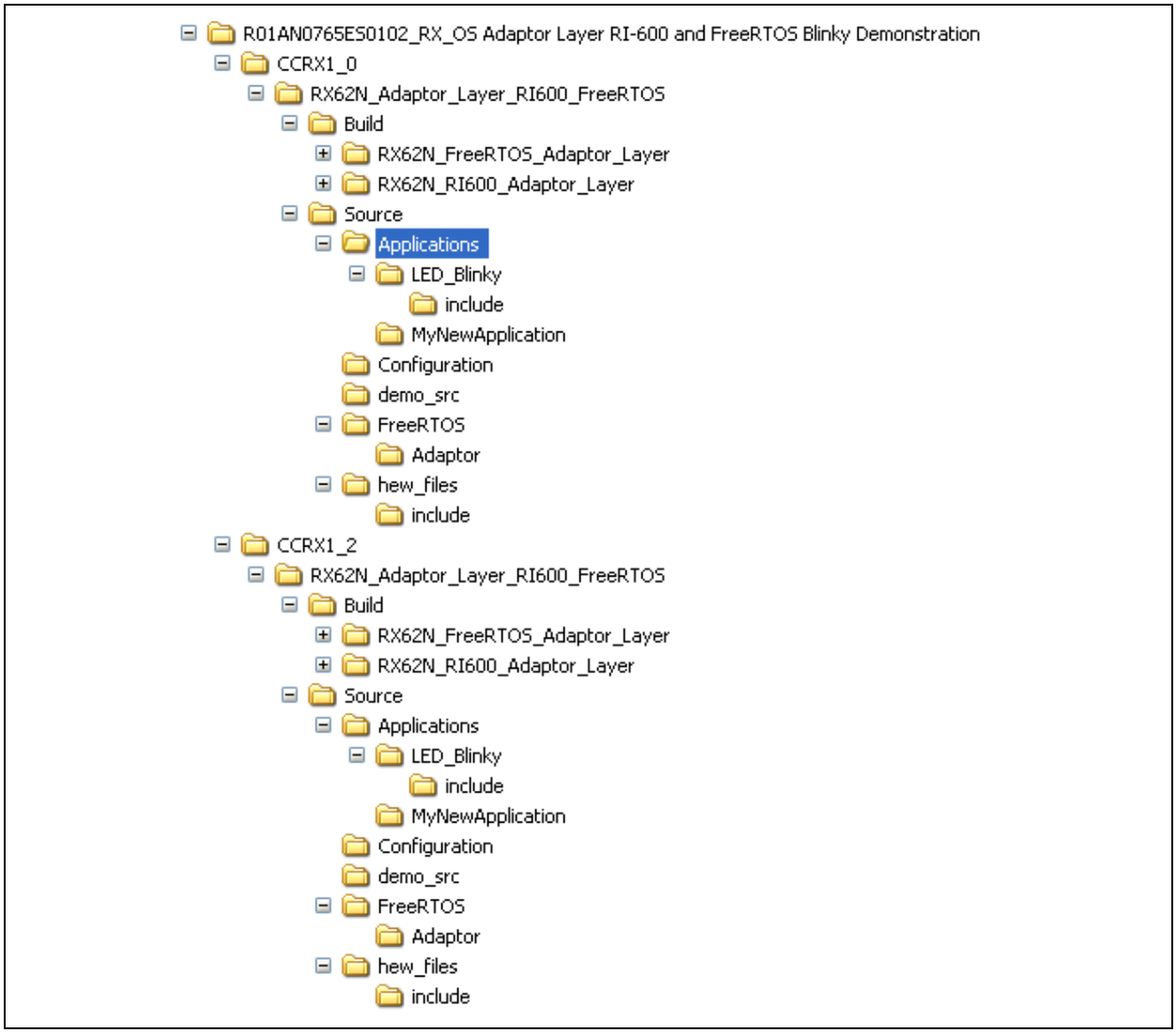
- Optimizing linkage editor settings for Renesas RX Toolchain Version 1.2 are as follows:
  - Linker options:

  -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. Directory Structure

2.1 Overview

The directories of the sample project workspace contain the following:



CCRXn_w

This folder contains the workspaces for Renesas compiler version of V1.0 and V1.2. Reason for this is to manage the additional section “L” generated by in Compiler V1.2

\RX62N_Adaptor_Layer_RI600_FreeRTOS

This folder is the main workspace that has 2 sub-directories namely: Build and Source.

\Build

This directory contains the Project Directories, one for RI-600 and the other for FreeRTOS™.

\Source

The source directory contains all the source code of the sample application. The LED_Blink directory places all files pertaining to the blinky example in this directory for modular separation.

\Source\FreeRTOS

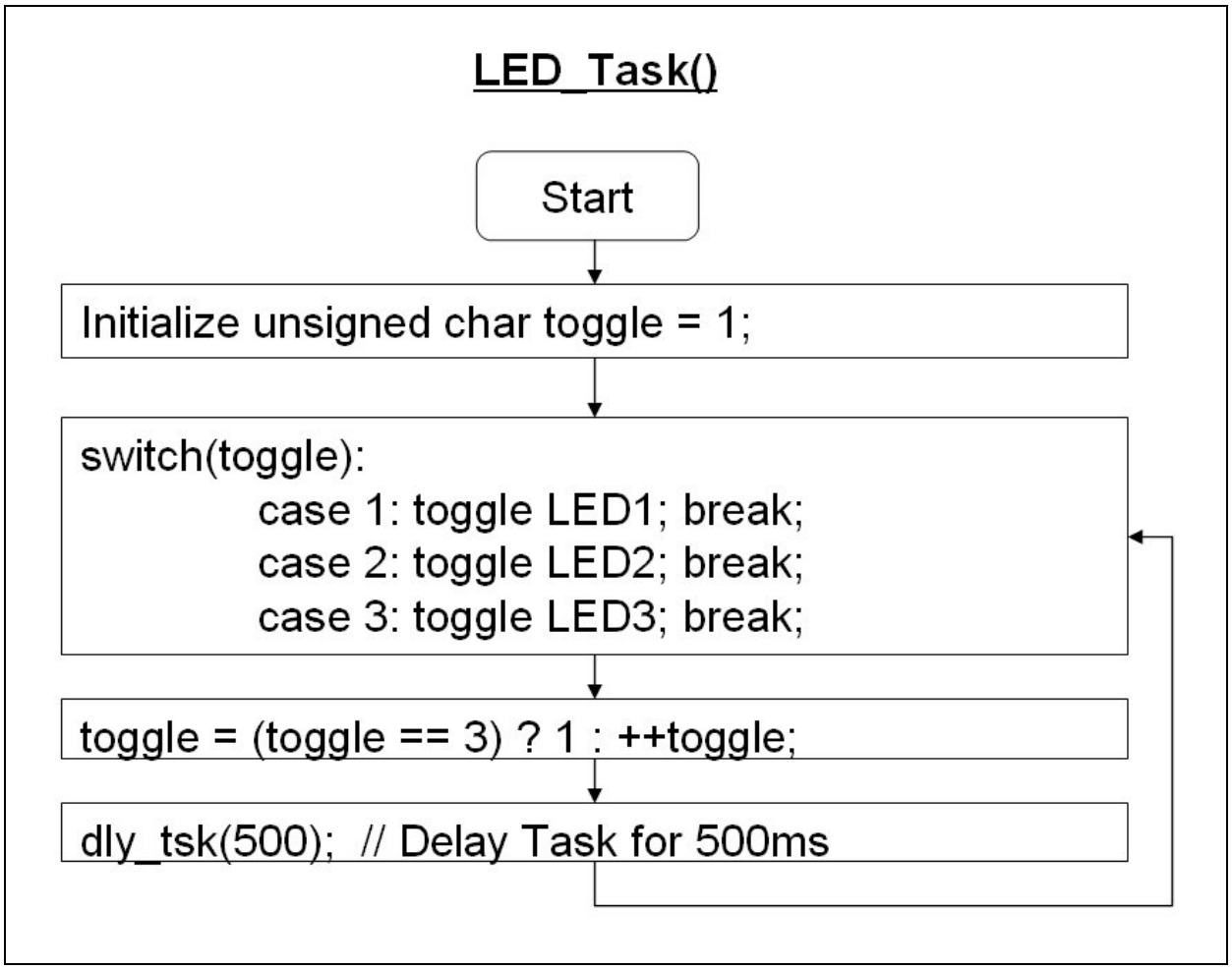
This directory contains all the files that are required for FreeRTOS™.

\Source\FreeRTOS\Adaptor

This folder contains the Adaptor Layer code and configuration files.

3. Compilation and Running Application

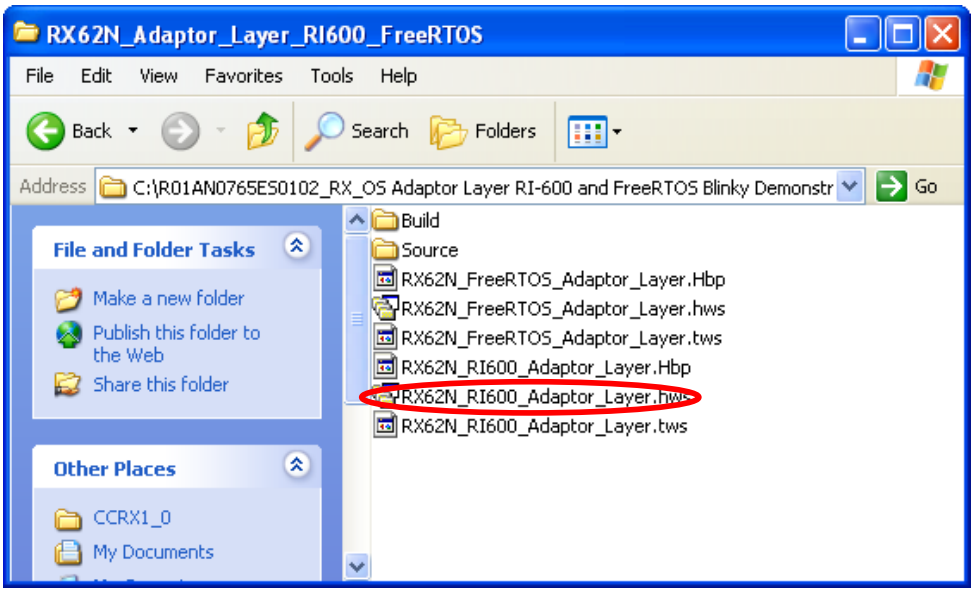
This section is a walkthrough of the LED Blinky Application that has the following program execution flow:



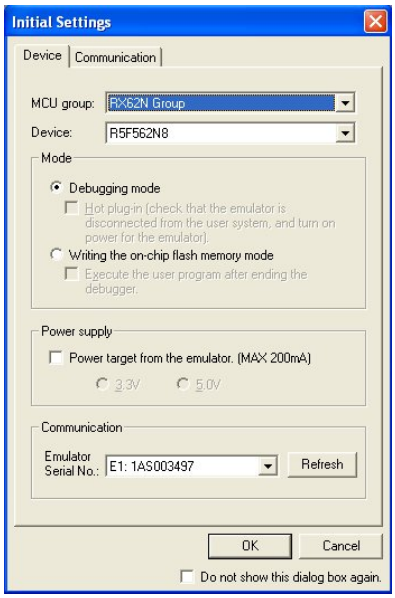
3.1 RI600

3.1.1 Opening RI-600 Project Workspace

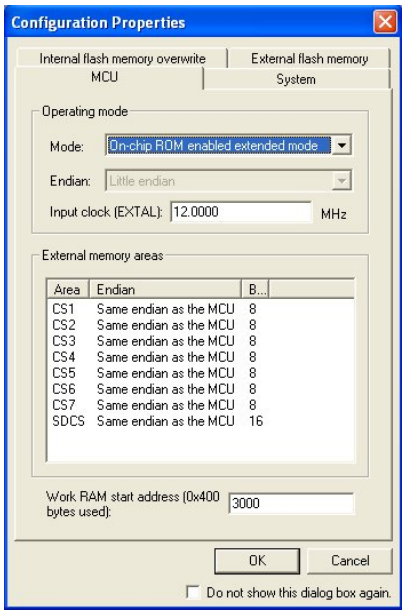
Locate the .Project Workspace File (RX62N_RI600_Adaptor_Layer.hws) located in the extracted directory folder. This can be seen as follows:



Connect to the RX62N RSK with the following settings:

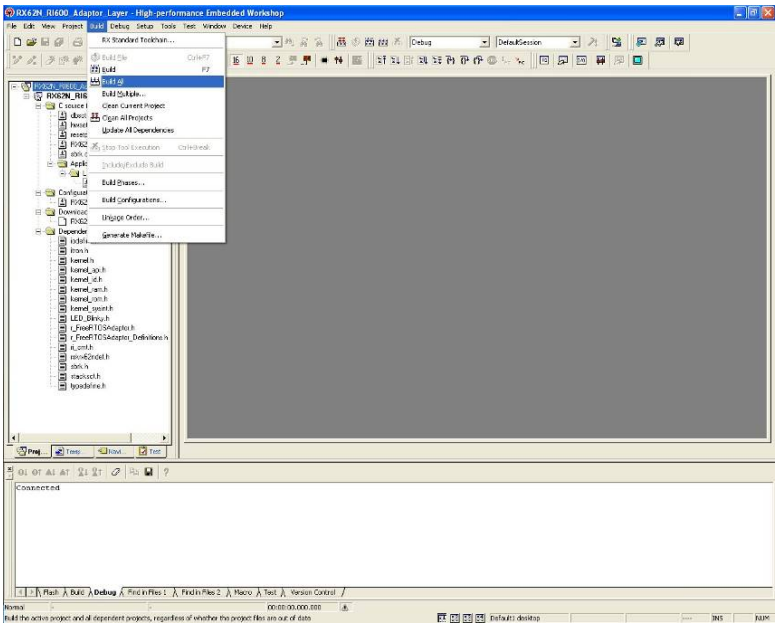


In the next screen, configure as follows to complete the connection:



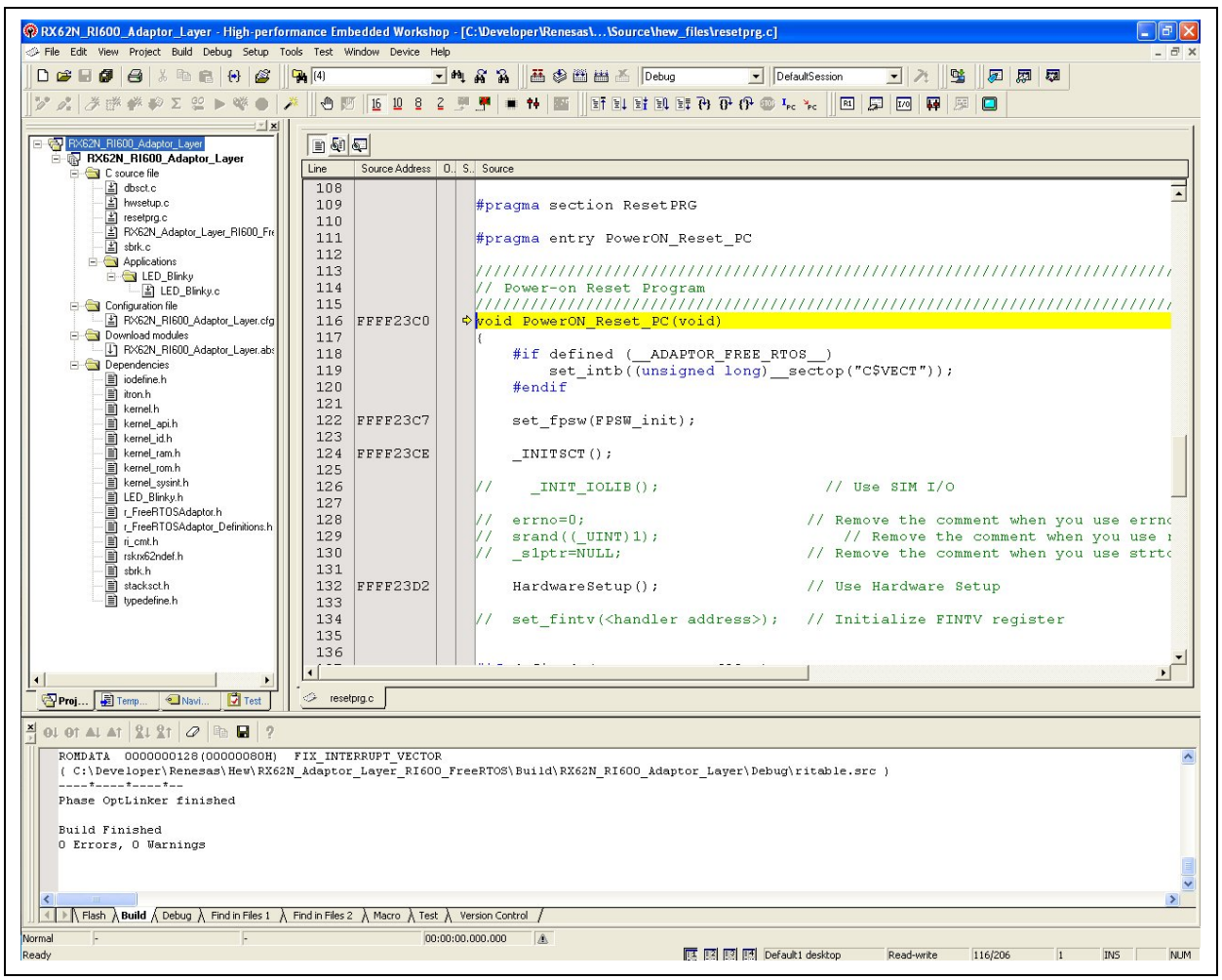
3.1.2 Compile and Build

Compile and build the workspace project by selecting Build→Build All, as shown in the following:

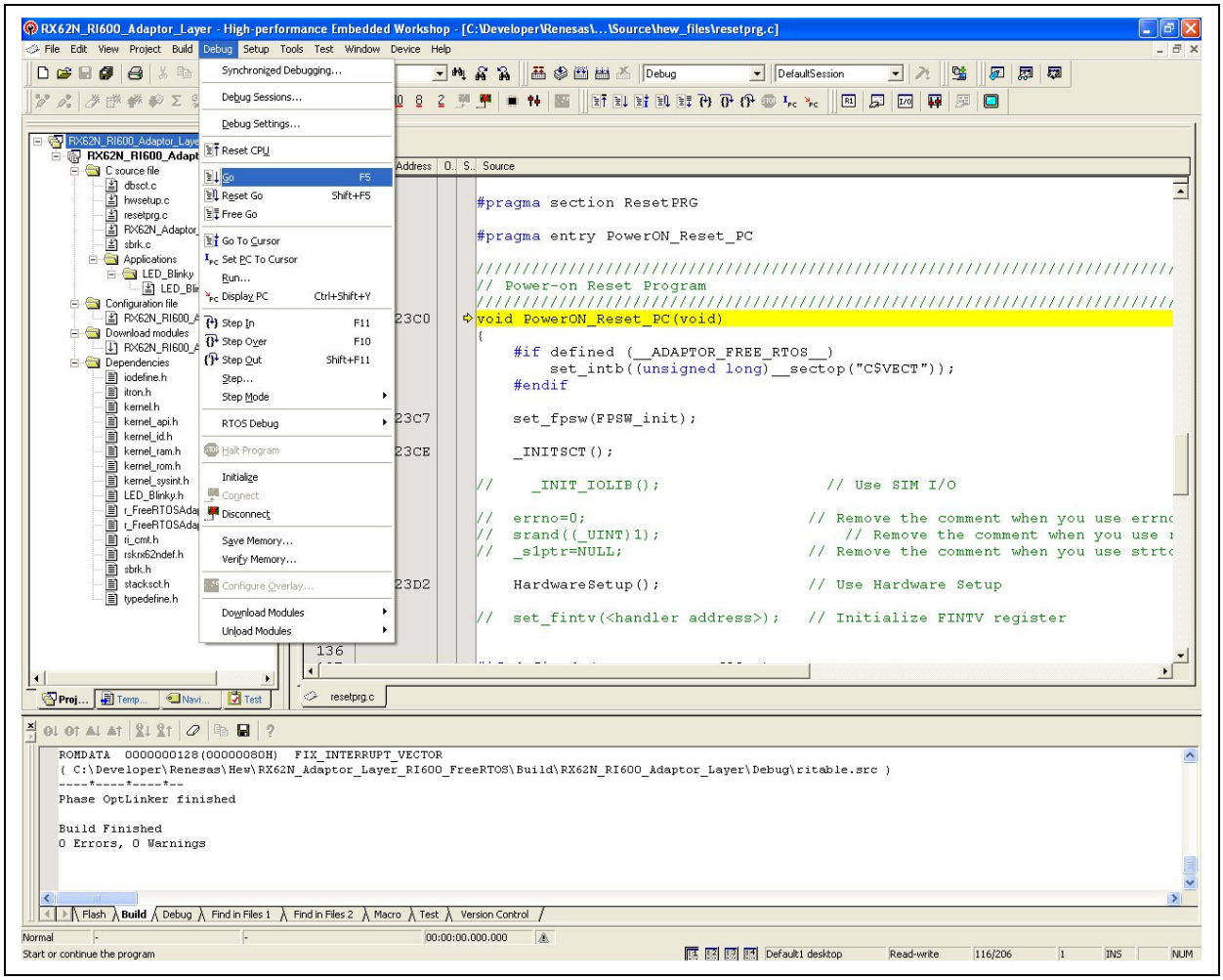


Upon completion of the build, a confirmation request to download the module may appear. Ensure that “Yes to all” is selected:

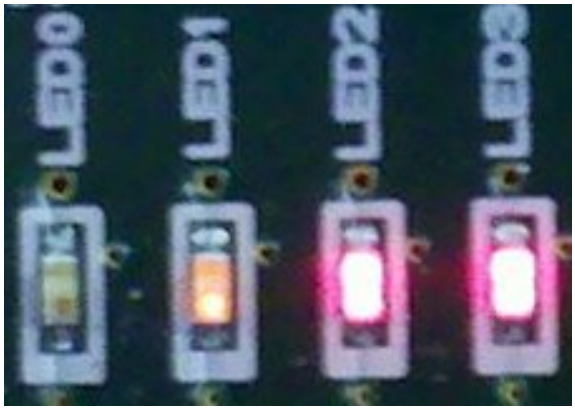
After the module has loaded, the cursor should be set to the PowerON_Reset_PC() function as shown:



Press short-cut key “F5” to allow the program to run. Optionally, Debug → Go may be selected as shown:



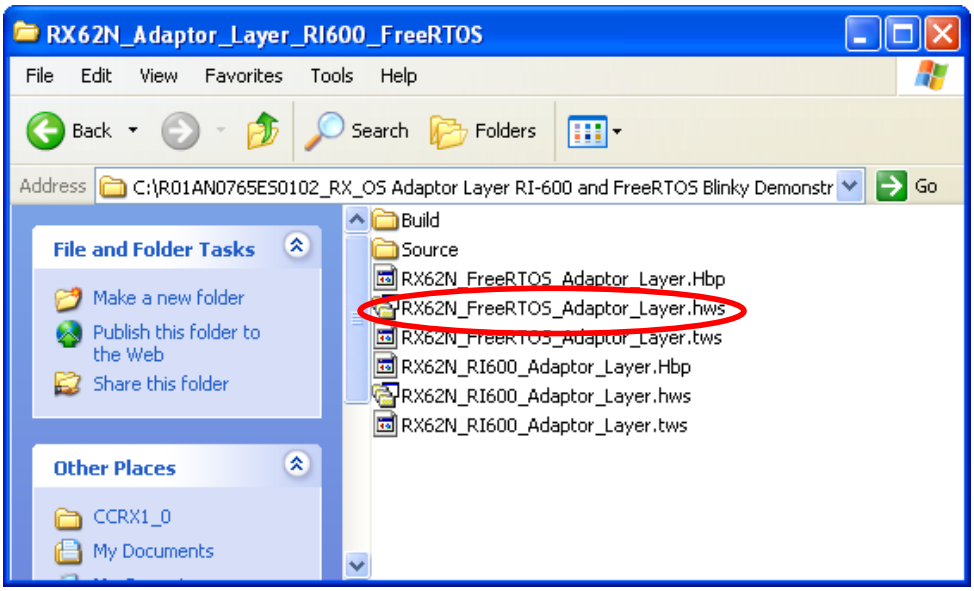
The blinking of the LEDs on the RX62N RSK Board can be seen as follows:



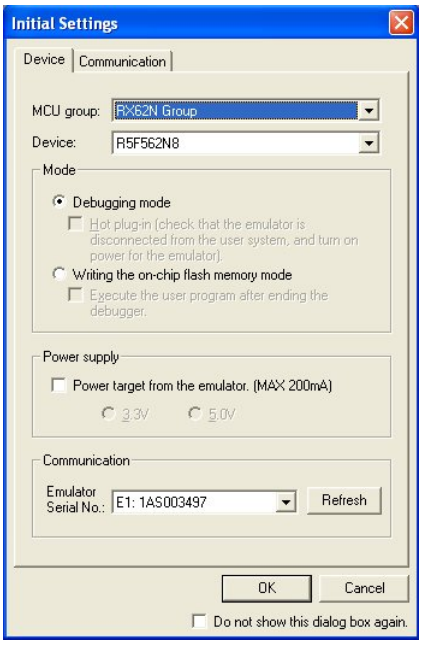
3.2 FreeRTOS™

3.2.1 Opening FreeRTOS™ Project Workspace

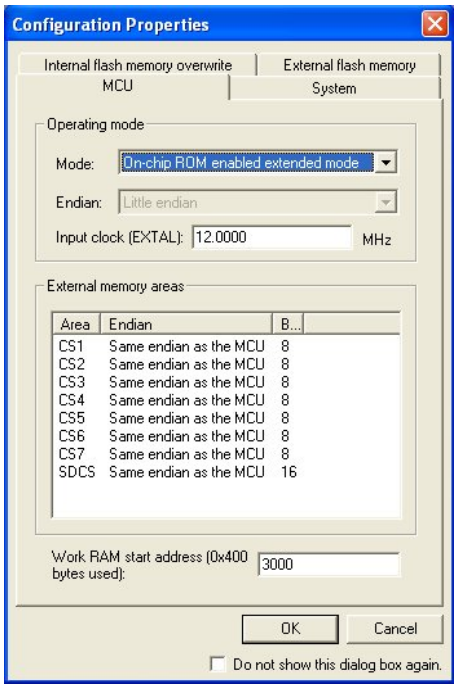
Locate the .Project Workspace File (RX62N_FreeRTOS_Adaptor_Layer.hws) located in the extracted directory folder. This can be seen as follows:



Connect to the RX62N RSK with the following settings:

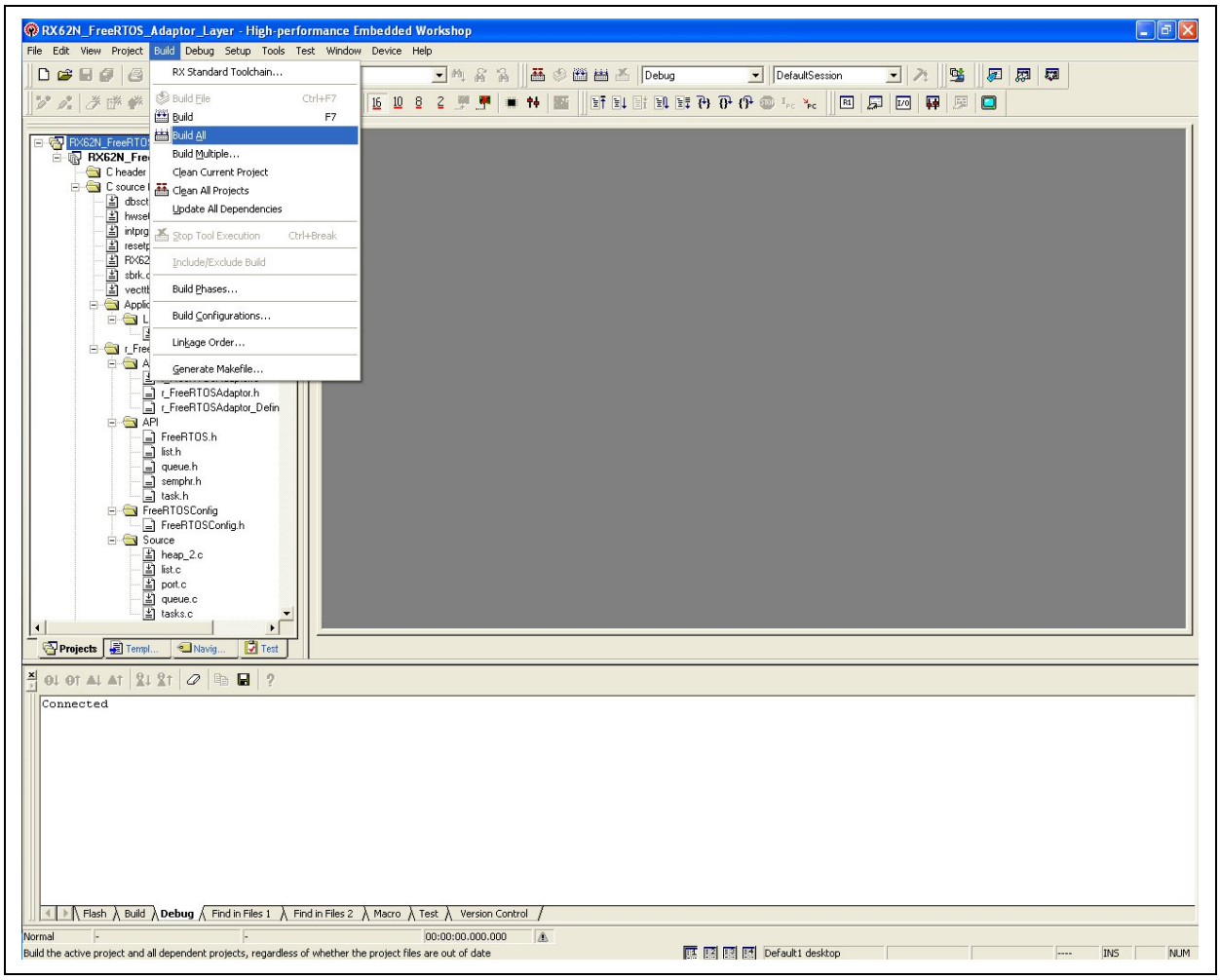


In the next screen, configure as follows to complete the connection:

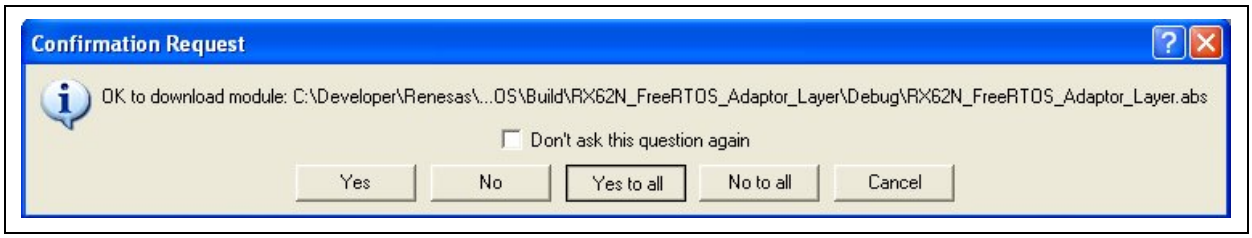


3.2.2 Compile and Build

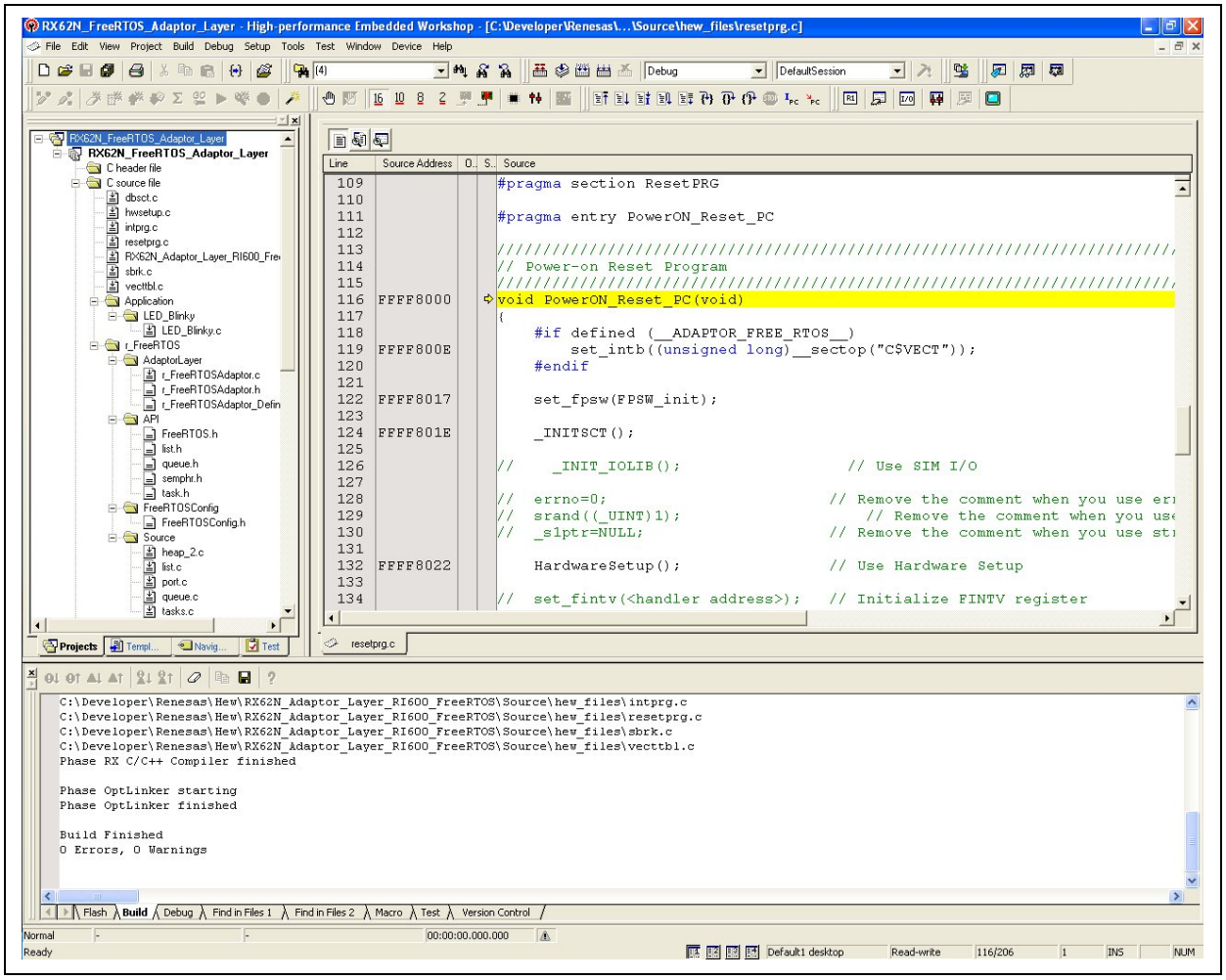
Compile and build the workspace project by selecting Build→Build All, as shown in the following:



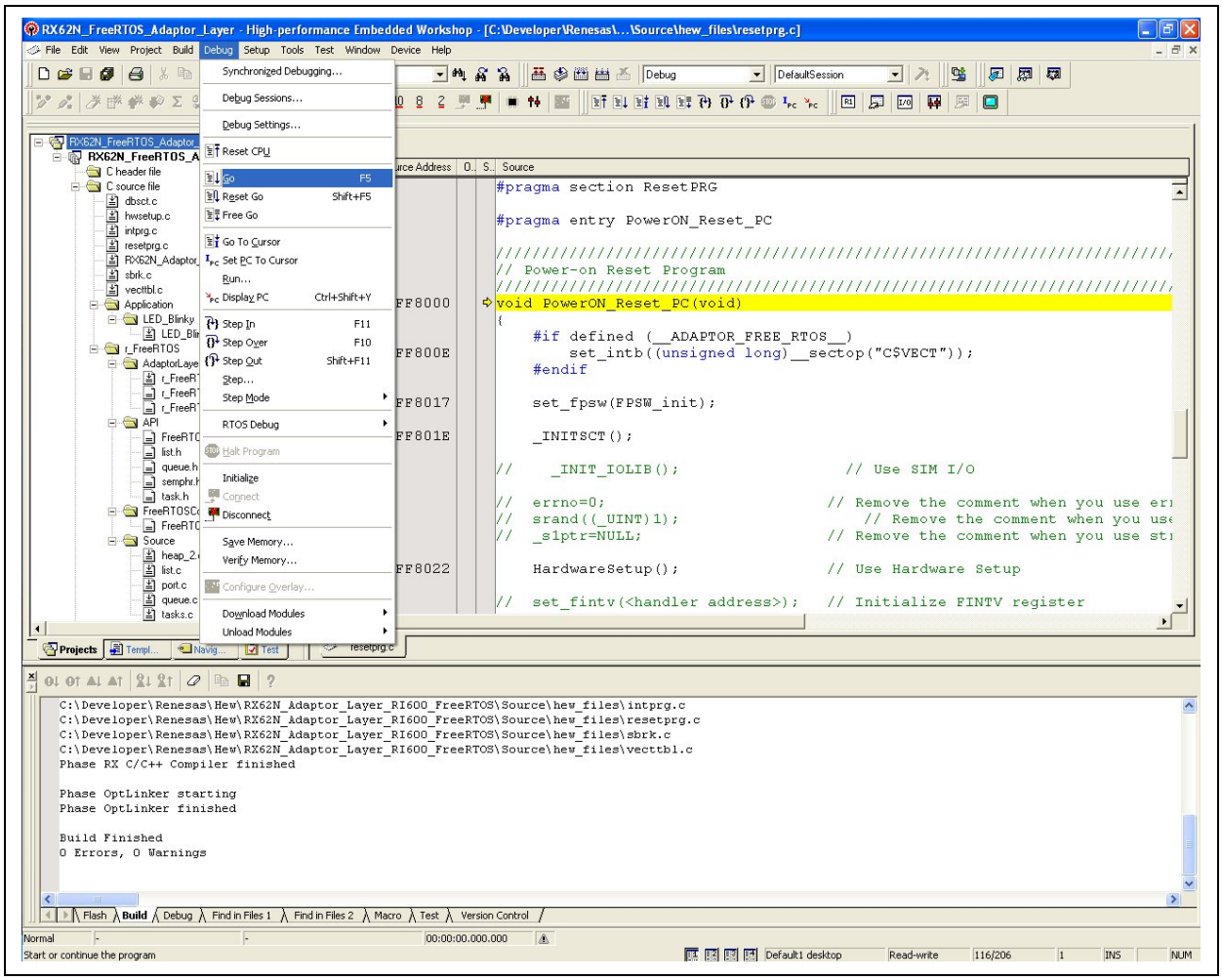
Upon completion of the build, a confirmation request to download the module may appear. Ensure that “Yes to all” is selected:



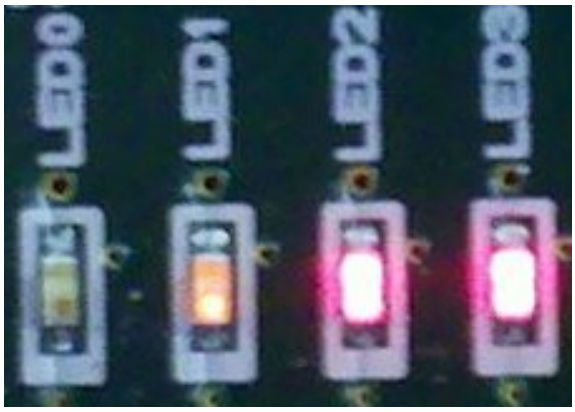
After the module has loaded, the cursor should be set to the PowerON_Reset_PC() function as shown:


RX62N_FreeRTOS_Adaptor_Layer - High-performance Embedded Workshop - [C:\Developer\Renesas\...\Source\hew_files\resetprg.c]
File Edit View Project Build Debug Setup Tools Test Window Device Help
RX62N_FreeRTOS_Adaptor_Layer
C header file
C source file
dbsct.c
hwsetup.c
intprg.c
resetprg.c
RX62N_Adaptor_Layer_RI600_Fre
sbrk.c
vecttbl.c
Application
LED_Blinky
LED_Blinky.c
r_FreeRTOS
AdaptorLayer
r_FreeRTOSAdaptor.c
r_FreeRTOSAdaptor.h
r_FreeRTOSAdaptor_Defin
API
FreeRTOS.h
list.h
queue.h
semphr.h
task.h
FreeRTOSConfig
FreeRTOSConfig.h
Source
heap_2.c
list.c
port.c
queue.c
tasks.c
Projects
Templ...
Navig...
Test
Line
Source Address
O..
S..
Source
109 #pragma section ResetPRG
110
111 #pragma entry PowerON_Reset_PC
112
113 //
114 // Power-on Reset Program
115 //
116 FFFF8000 void PowerON_Reset_PC(void)
117 {
118 #if defined (__ADAPTOR_FREE_RTOS__)
119 FFFF800E set_intb((unsigned long)__sectop("C$VECT"));
120 #endif
121
122 FFFF8017 set_fpsw(FPSW_init);
123
124 FFFF801E _INITSCT();
125
126 // _INIT_IOLIB(); // Use SIM I/O
127
128 // errno=0; // Remove the comment when you use er
129 // srand((_UINT)1); // Remove the comment when you us
130 // _s1ptr=NULL; // Remove the comment when you use st
131
132 FFFF8022 HardwareSetup(); // Use Hardware Setup
133
134 // set_fintv(<handler address>); // Initialize FINTV register
resetprg.c
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
Flash
Build
Debug
Find in Files 1
Find in Files 2
Macro
Test
Version Control
Normal
00:00:00.000.000
Ready
Default1 desktop
Read-write
116/206
1
INS
NUM

Press short-cut key “F5” to allow the program to run. Optionally, Debug → Go may be selected as shown:



The blinking of the LEDs on the RX62N RSK Board can be seen as follows:



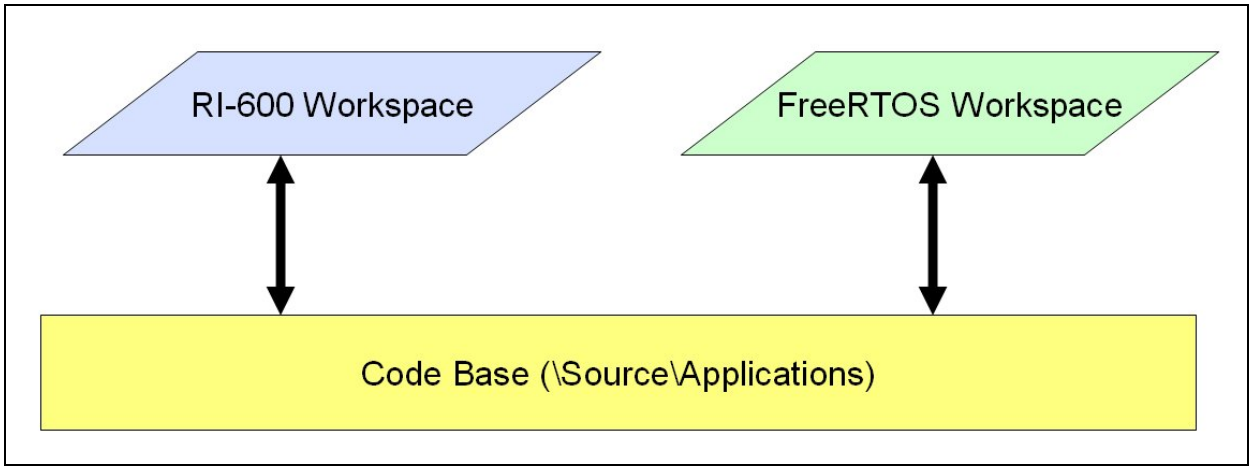
4. Building Applications from the Workspace

4.1 Application Development

Application code development and APIs should conform to that of the RI-600 Specification. This means that applications should be written as though they are targeted for the RI-600 platform.

4.2 Application Code Base

Applications can be developed by using either of the provided workspaces. This allows for ease of application development and test on both the RI-600 and FreeRTOS™ platform. The workspace directories and files are linked as shown:



4.3 Application Main Program

The application main program file can be found in the following source code file in either of the workspace directories:

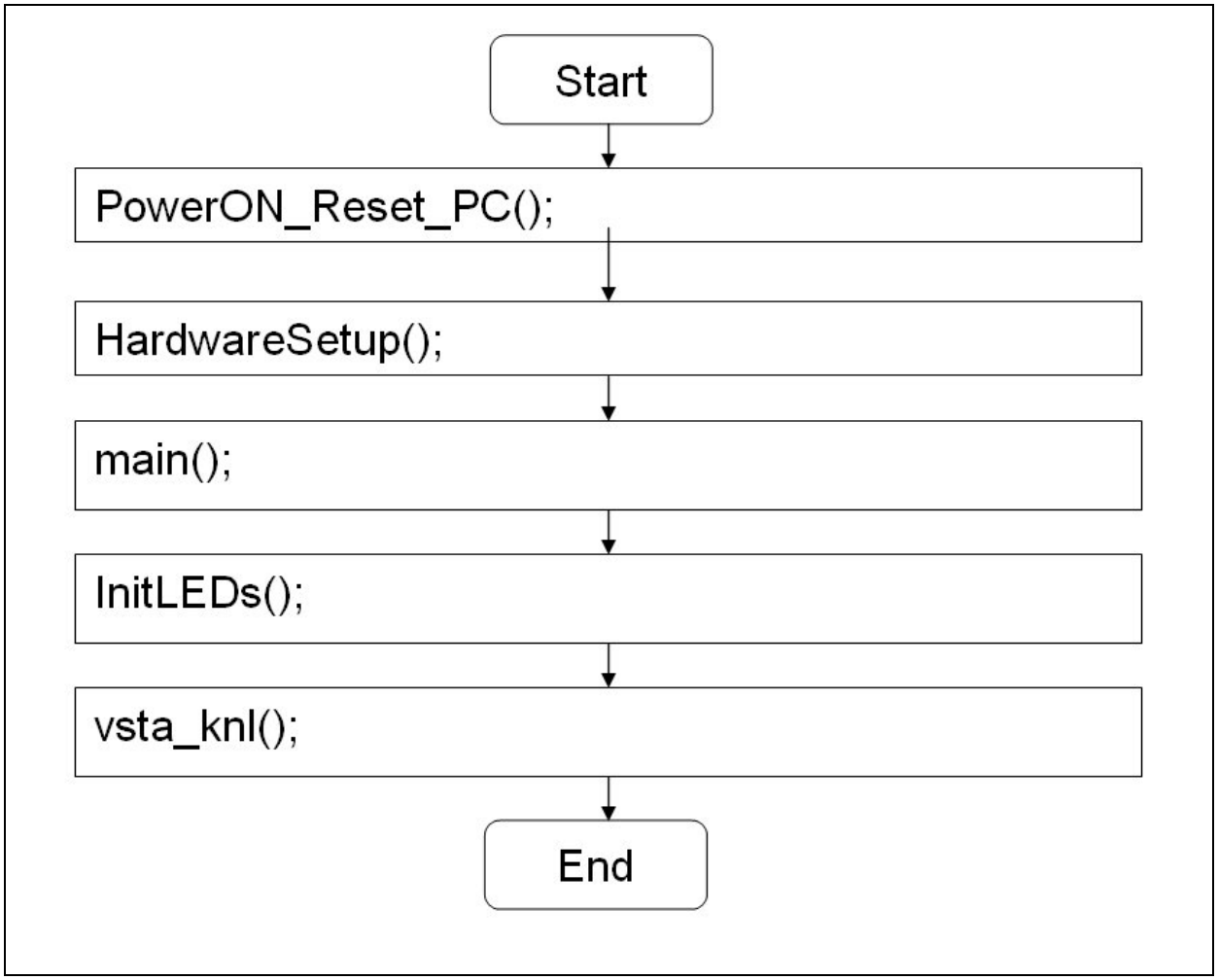
The default source code file contains the following:

```
#include "LED_Blinky.h"


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

        // Start Kernel
    vsta_knl();

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

5. Blinky Demo Application Flow

5.1 Application Flow

The Blinky Application flowchart can be seen as follows:



5.2 Application Resources

5.2.1 Hardware Resources

The Blinky Demo Application has the following hardware setup:

Input clock: 12 MHz
System clock (ICLK): 96 MHz
Peripheral module clock (PCLK): 48 MHz
External bus clock (BCLK) and SDRAM clock (SDCLK): 24 MHz

5.2.2 Application Resources

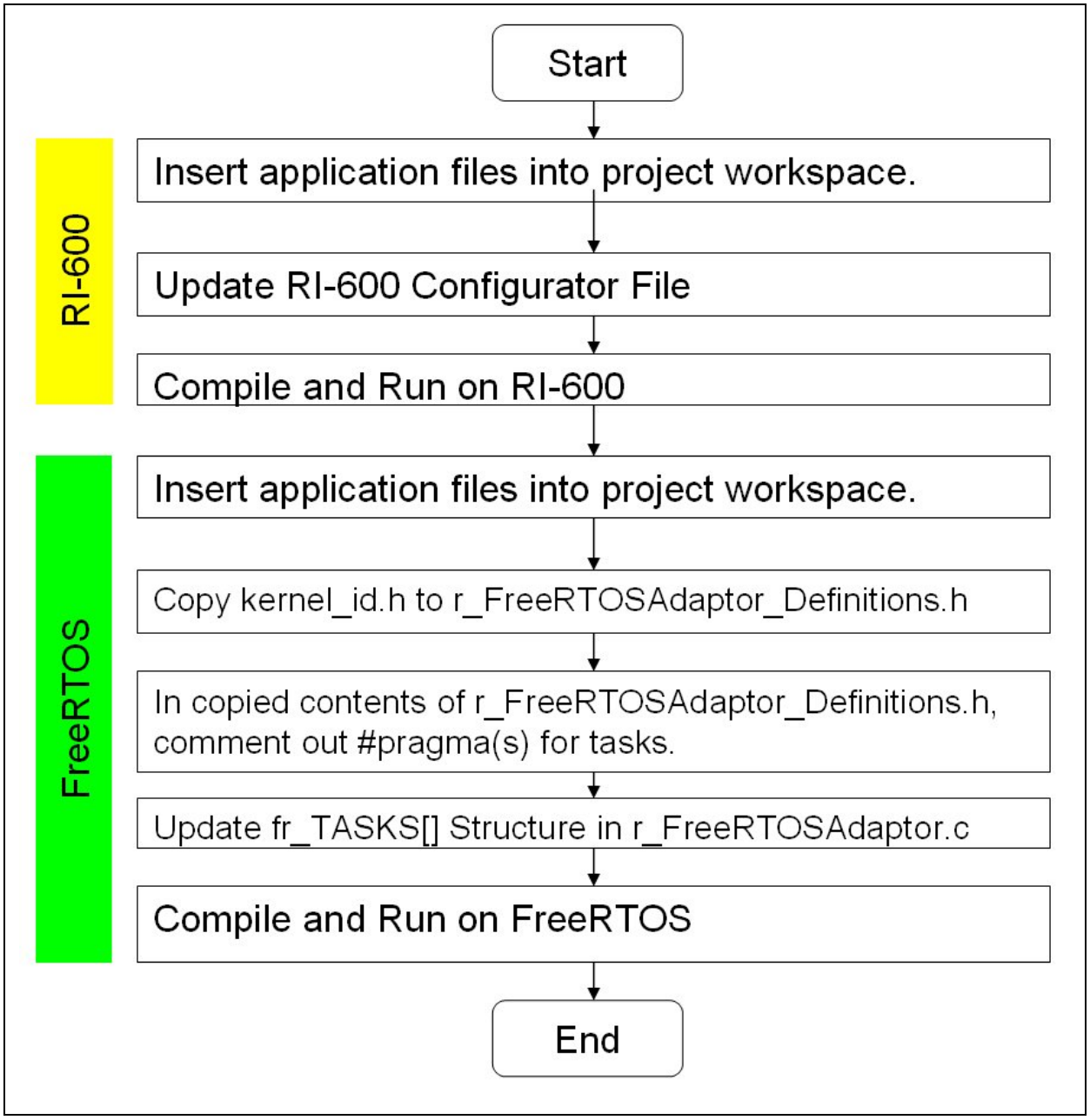
The Blinky Demo Application has the following characteristics:

No. of Tasks:	1 (LED_TASK)
RTOS API used:	vsta_knl(), dly_tsk()

Details of these functions can be found in the Adaptor Layer User Manual and API Specification.

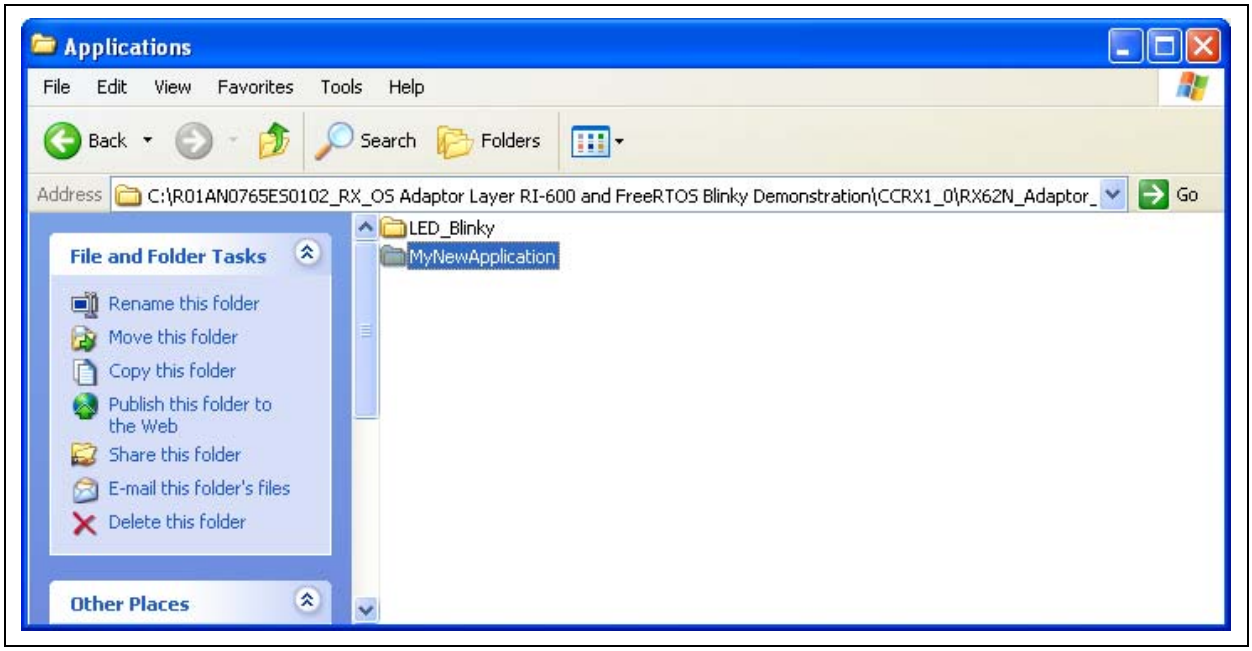
6. Application Creation Process Workflow

- The following steps outline the creation of an application that that runs on both RTOS:

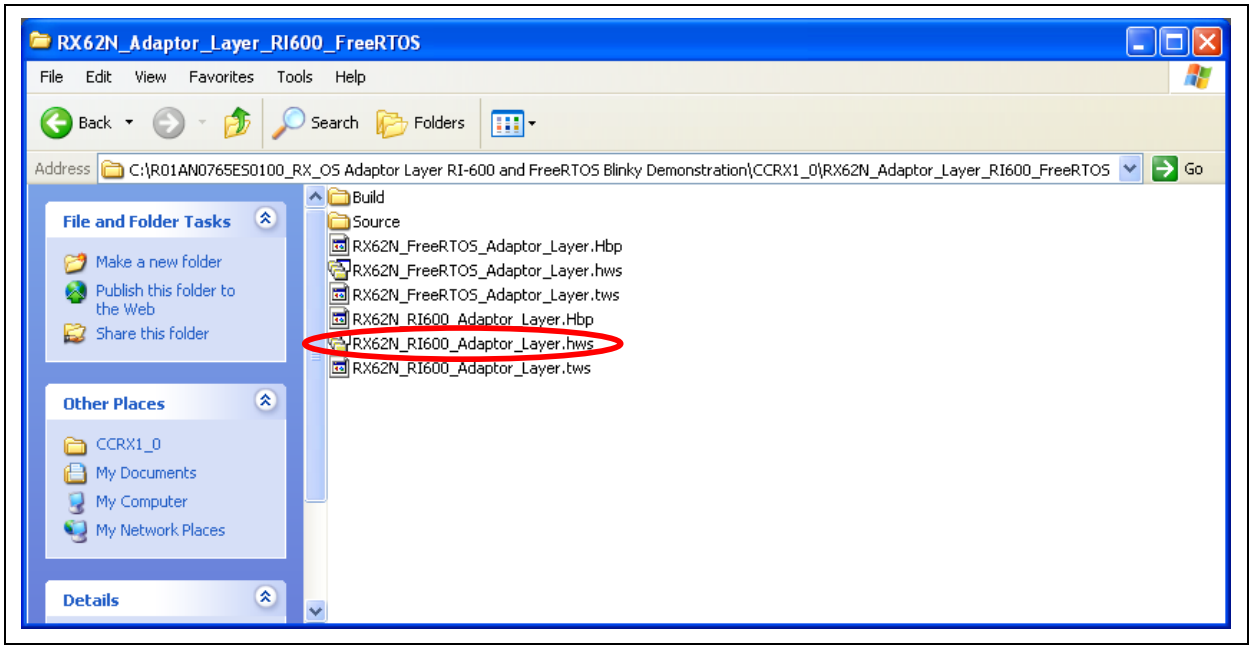


6.1 RI-600: Creating an Application for RI-600

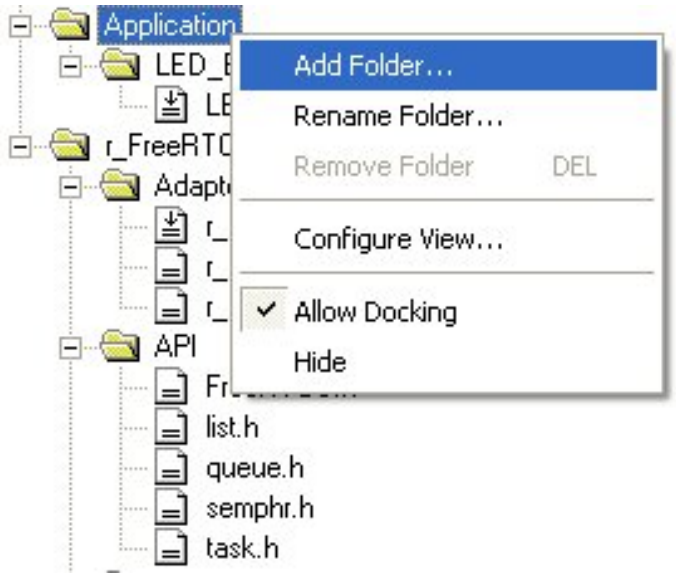
- Being by creating an application directory under \Source\Applications of the Workspace Directory



- Open the RX62N_RI600_Adaptor_Layer.hws Workspace as shown:

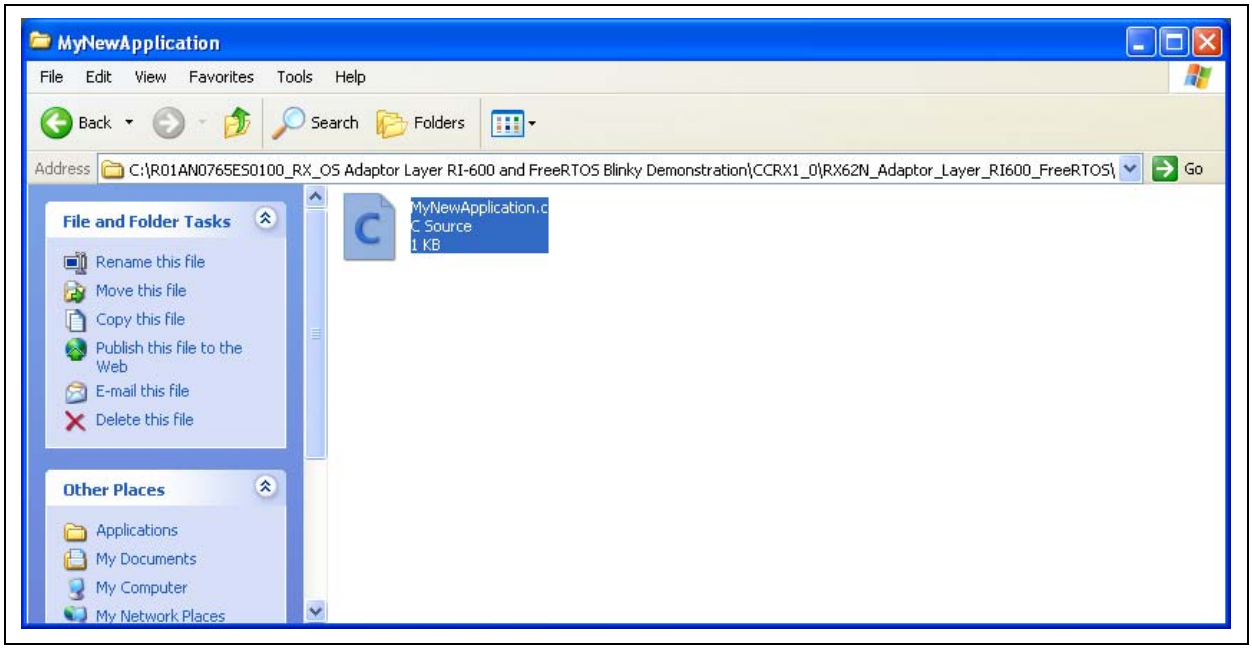


- Connect to the board by accepting the default settings and add the required folder to the Workspace

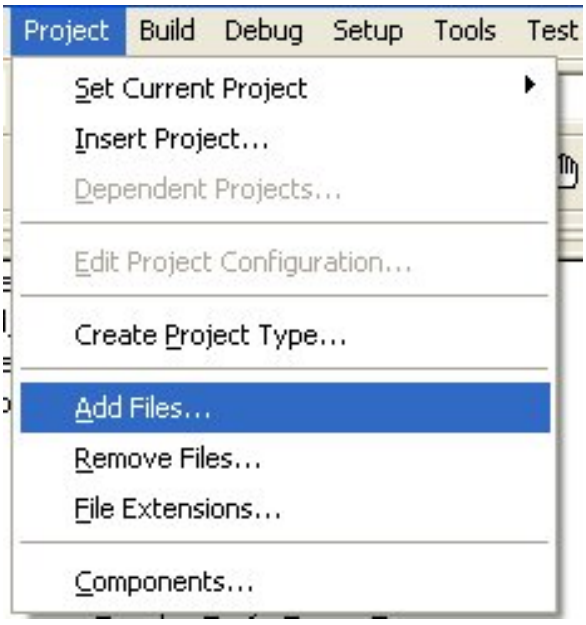


- Name the folder accordingly:

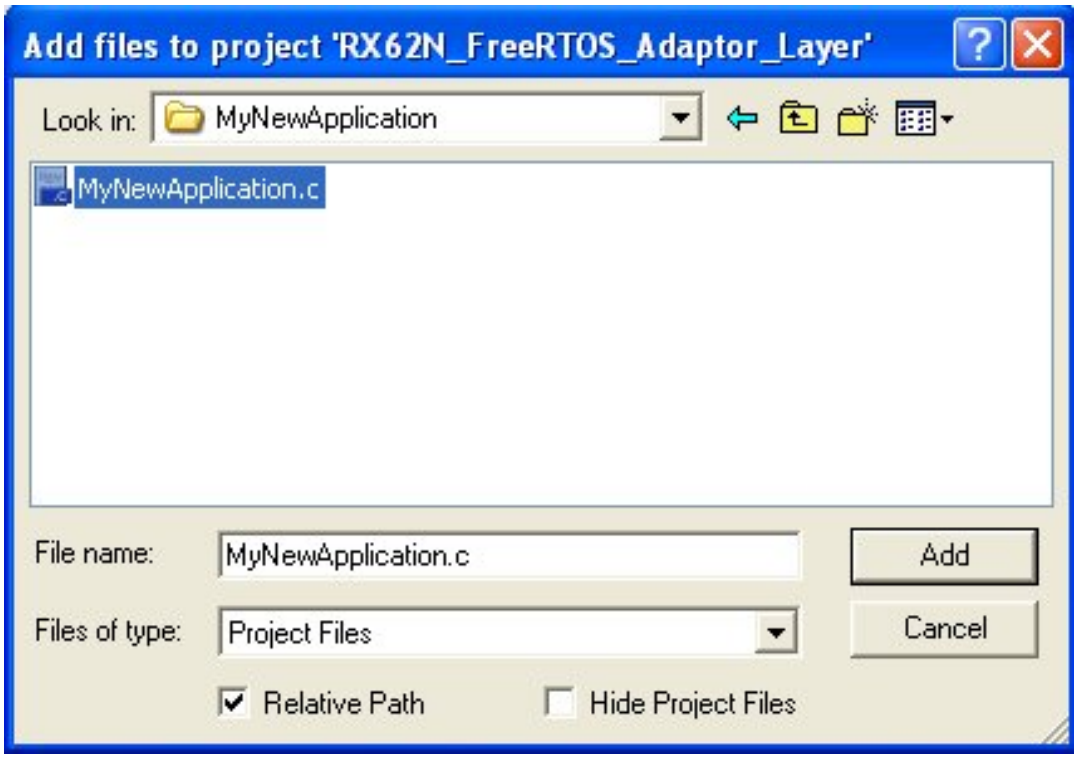
- Create a new C Code file named “MyNewApplication.c” in the \Source\Applications\MyNewApplication folder:



- Add this new C Code to the Workspace as follows:



- Select the file to add

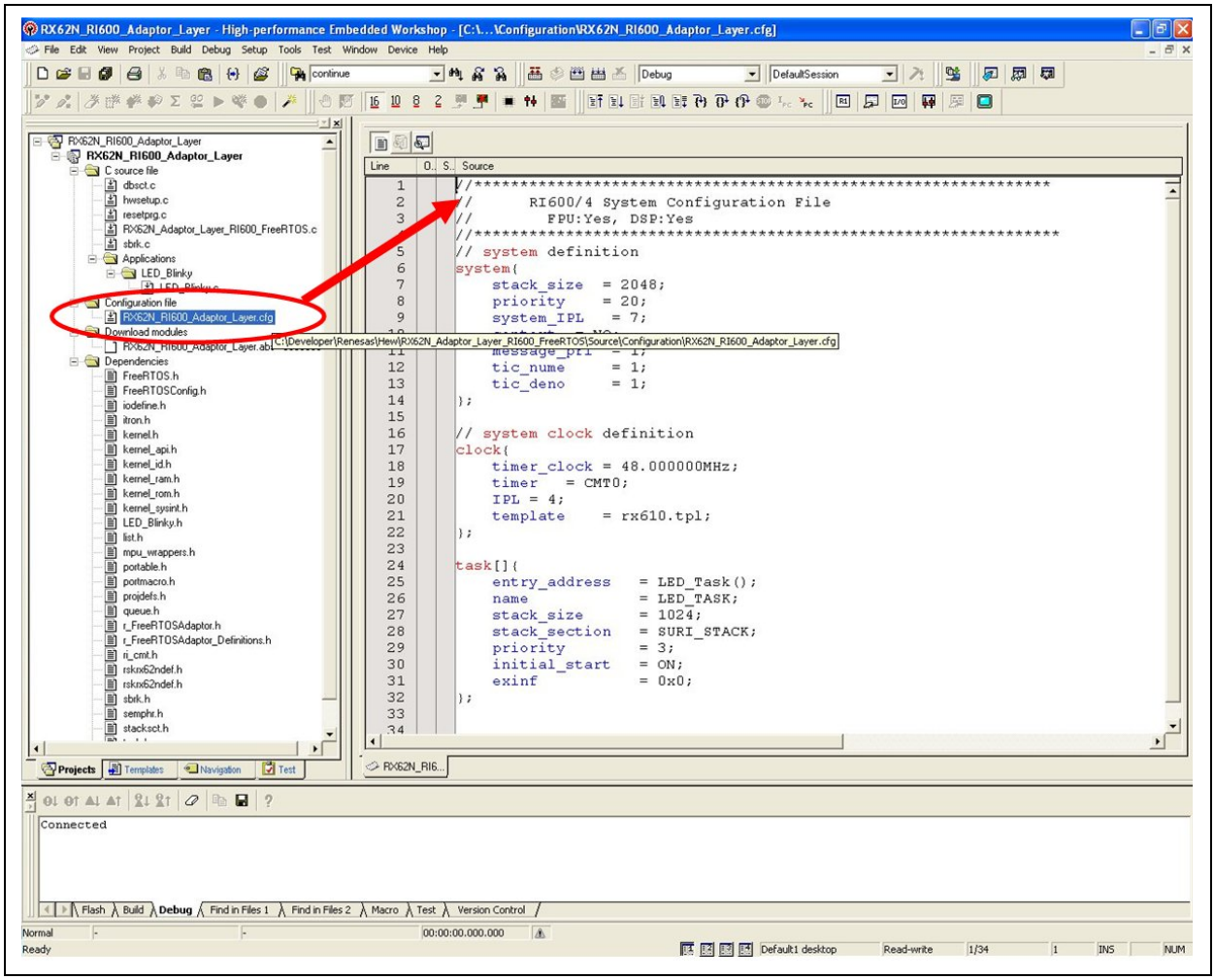


- Arrange the code in the Workspace as follows:

- Open MyNewApplication.c and add the following code to the file:

```
/***********************************************************************
*           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);
      }
}
```

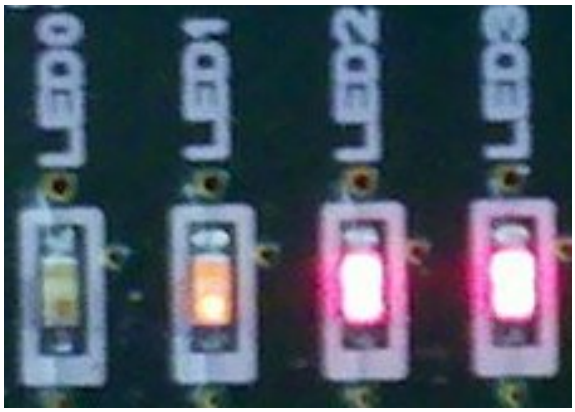
- Open the Configurator file: RX62N_RI600_Adaptor_Layer.cfg RI-600 as shown:



- Append the following configuration code as shown:

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

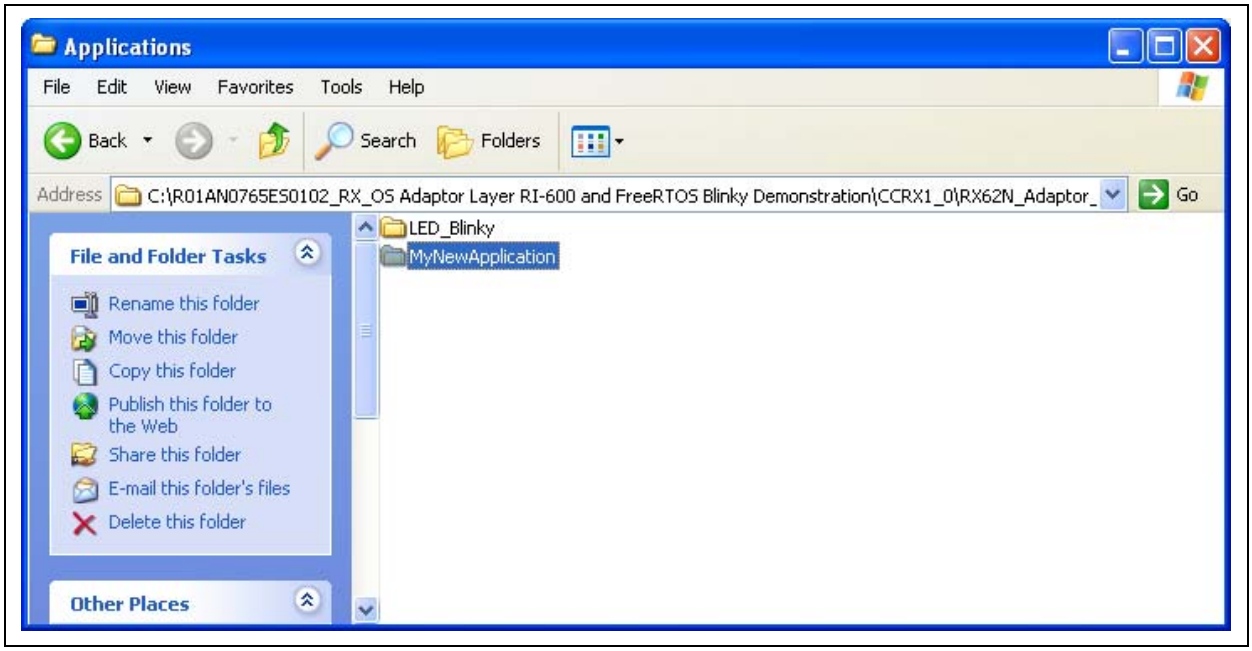
- Compile and build the workspace project by selecting Build→Build All. Subsequently, download and run the module. The blinking of the LEDs on the RX62N RSK Board can be seen as follows:



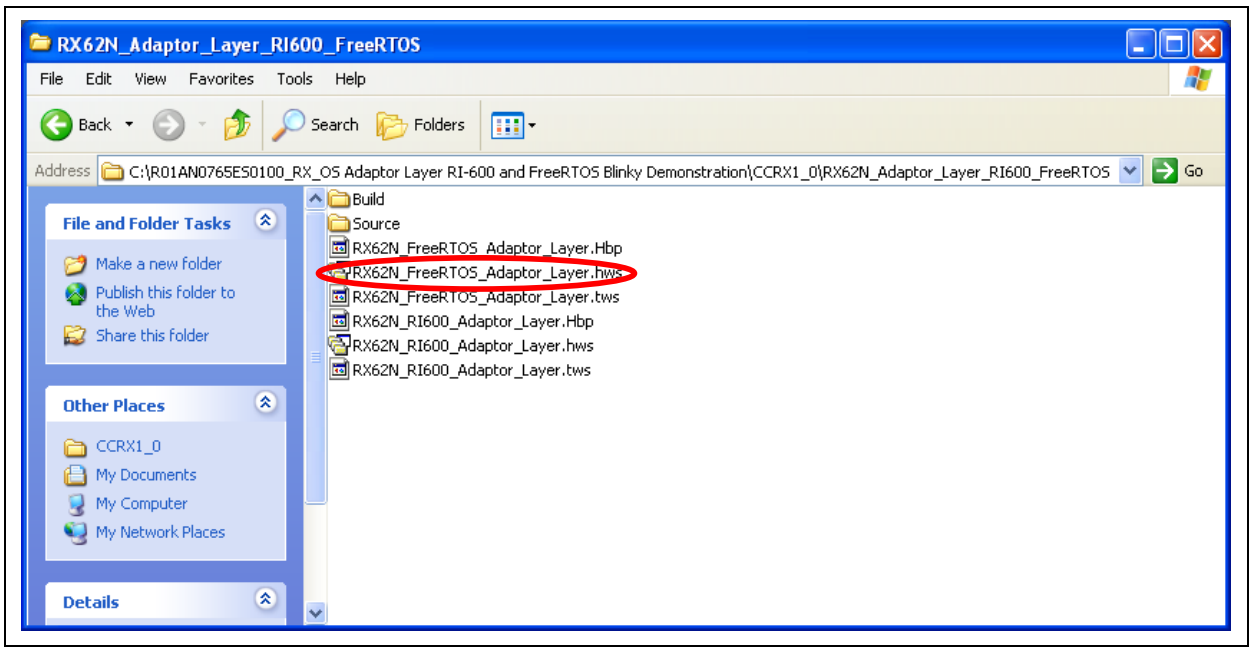
This time, LED0 blinks with a delay of 200ms while LED1 through LED3 blinks with a delay of 500ms.

6.2 FreeRTOS™: Creating an Application for FreeRTOS™

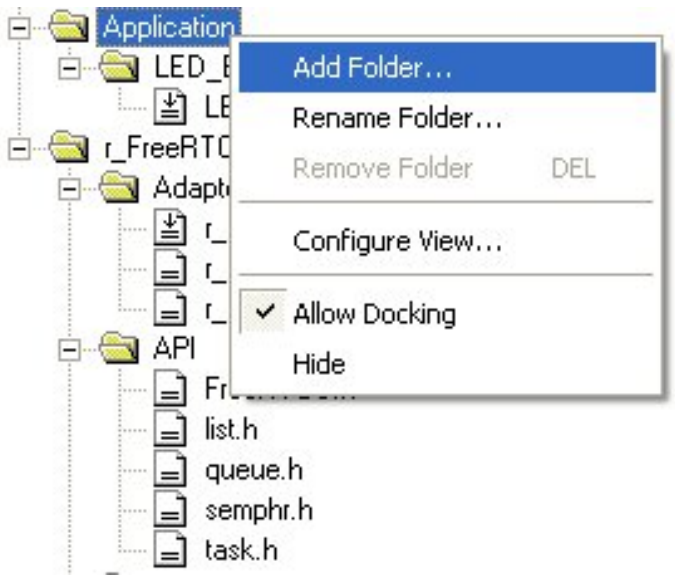
- Being by creating an application directory under \Source\Applications of the Workspace Directory



- Open the RX62N_FreeRTOS_Adaptor_Layer.hws Workspace as shown:

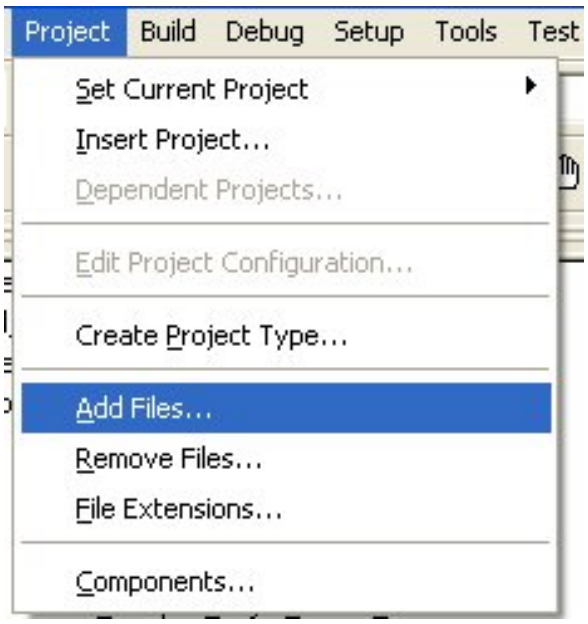


- Connect to the board by accepting the default settings. Add the required folder to the Workspace

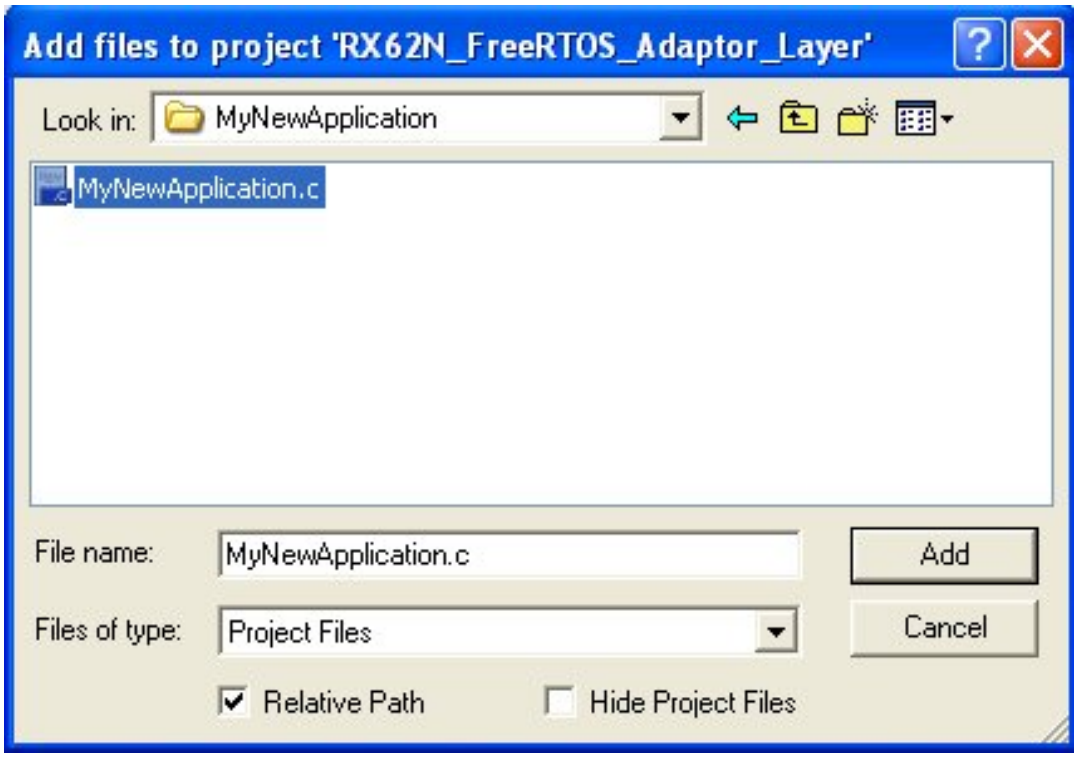


- Name the folder accordingly

- Create a new C Code file named “MyNewApplication.c” in the \Source\Applications\MyNewApplication folder and add it to the Workspace as follows:



- Select the file to add



- Arrange the code in the Workspace as follows:

- Copy all code generated by kernel_id.h and paste it in |\Source\FreeRTOS\Adaptor\r_FreeRTOSAdaptor_Definitions.h. This can be seen as follows:

- In the code pasted in r_FreeRTOSAdaptor_Definitions.h, locate the function Task code and comment out all compiler directive #pragma(s) as shown:

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

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

This step is necessary as FreeRTOS™ does not use such compiler directives for Tasks.

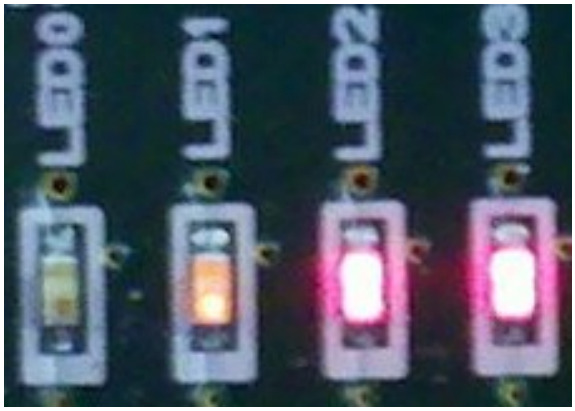
- Open file: r_FreeRTOSAdaptor.c and scroll to section: (1) Task Definitions and update the fr_TASKS[] structure as follows:

```
/*********************************************************
*       *** 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 },
};
```

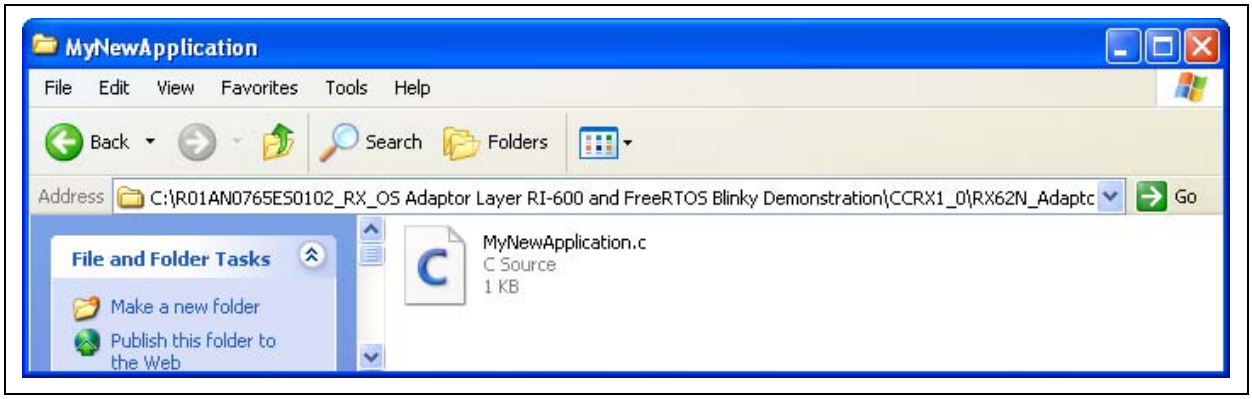
Note that fr_MAX_TASKS has been updated to 2 and an initialization of LED_Task2 has been created. This task structure initialization is necessary for the Adaptor Layer to manage the Tasks. For more information, consult the Adaptor Layer User Manual and API Specification Application Note.

- Compile and build the workspace project by selecting Build→Build All. Subsequently, download and run the module. The blinking of the LEDs on the RX62N RSK Board can be seen as follows:



6.3 RI-600: Application File Insertion

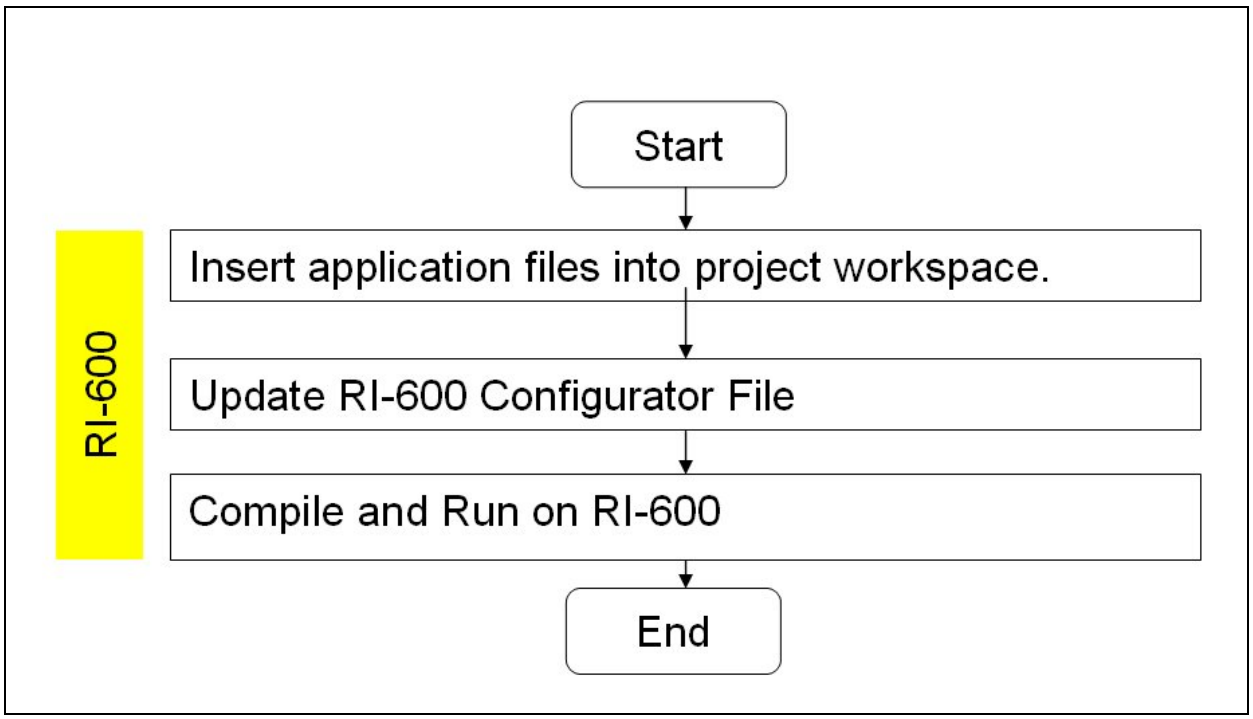
- This section shows how to insert a new application named “MyNewApplication.c” located in \Source\Applications\MyNewApplication. This is shown as follows:



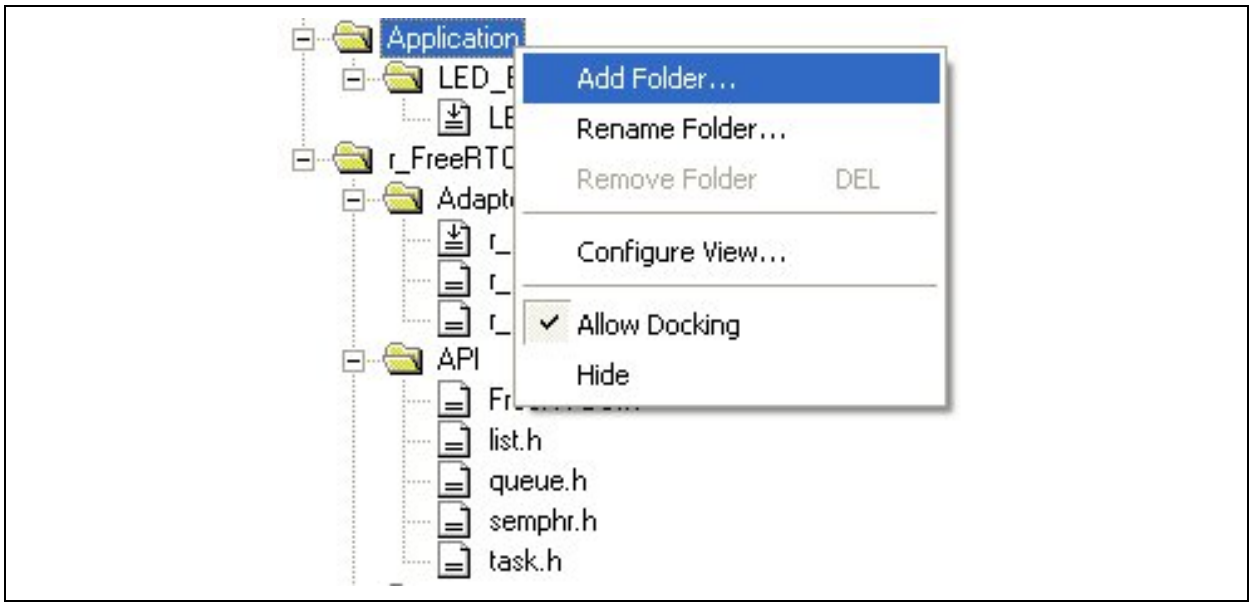
- The code in “MyNewApplication.c” is as follows:

```
/***********************************************************************
*           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);
      }
}
```

- Applications that are to be added to the RI-600 workspace needs to follow this sequence of steps:

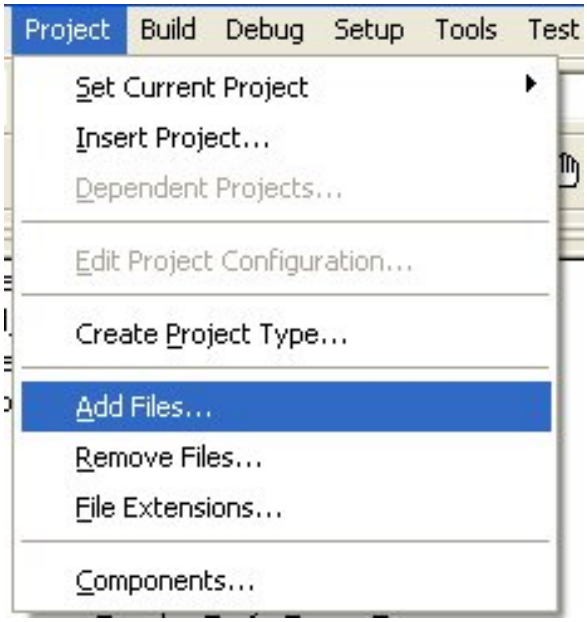


- Connect to the board by accepting the default settings and add the required folder to the Workspace

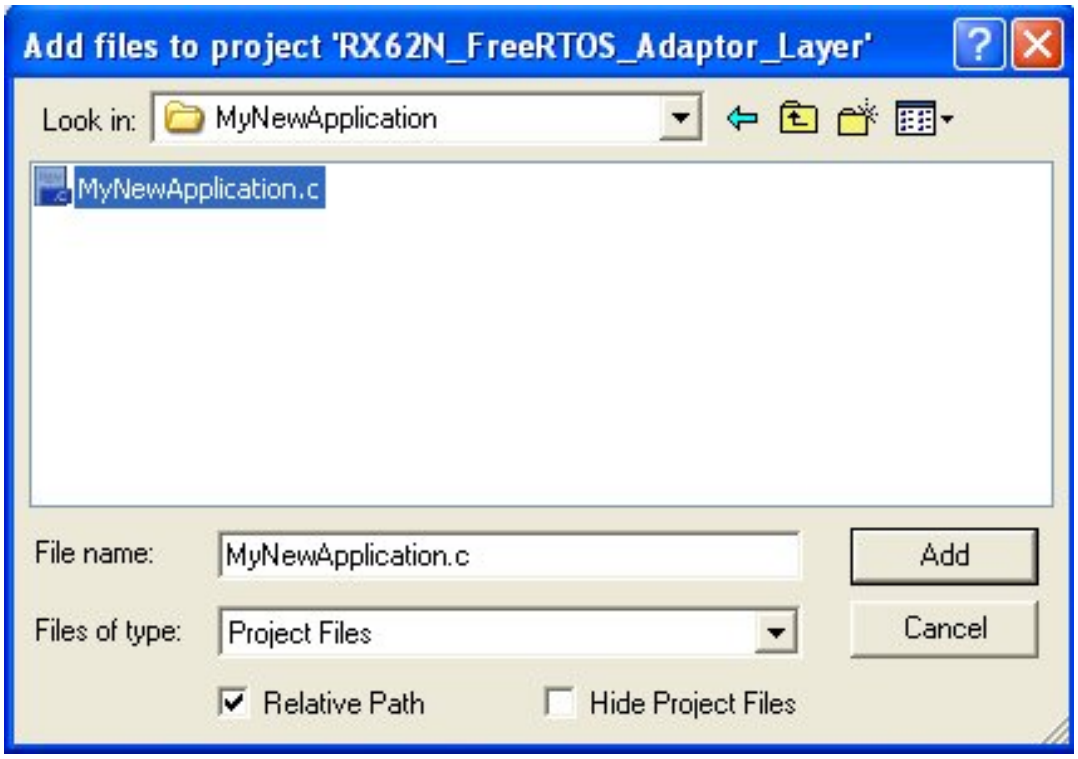


- Name the folder accordingly:

- Add the new C Code file named “MyNewApplication.c” in the \Source\Applications\MyNewApplication folder to the Workspace as follows:



- Select the file to add



- Arrange the code in the Workspace as follows:

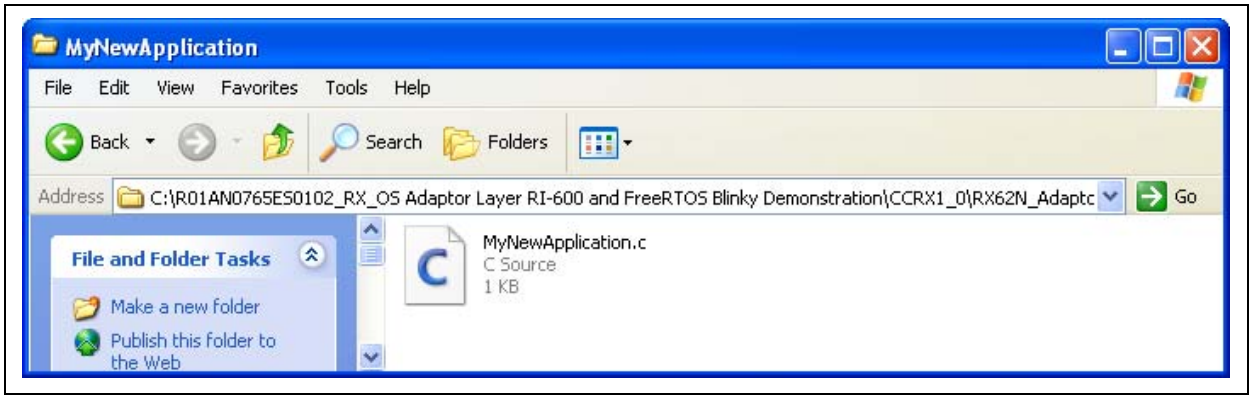
- Open RX62N_RI600_AdaptorLayer.cfg and append the following to create the new Task:

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

- Compile and build the workspace project by selecting Build→Build All. Subsequently, download and run the module.

6.4 FreeRTOS™: Application File Insertion

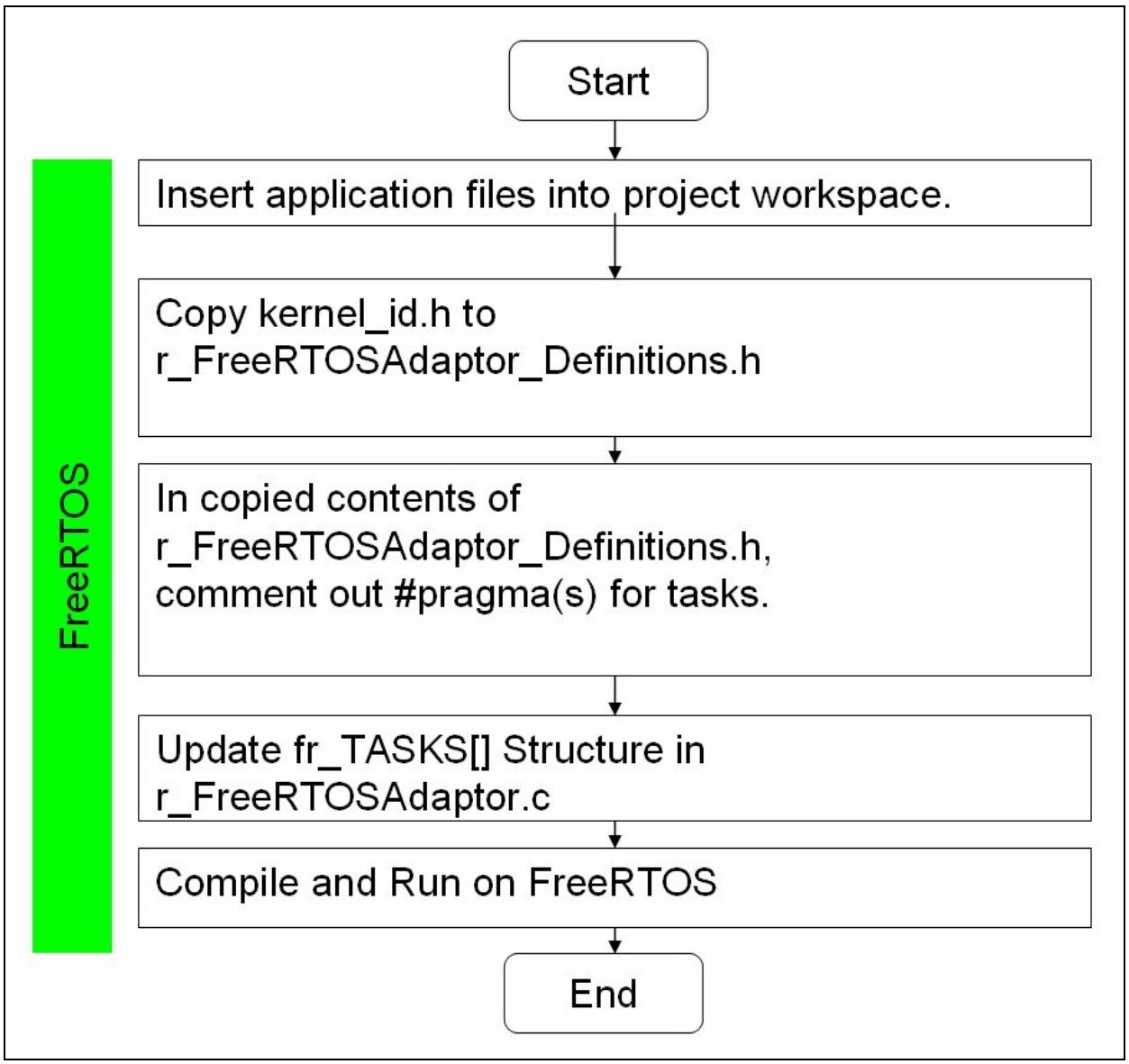
- This section shows how to insert a new application named “MyNewApplication.c” located in \Source\Applications\MyNewApplication. This is shown as follows:



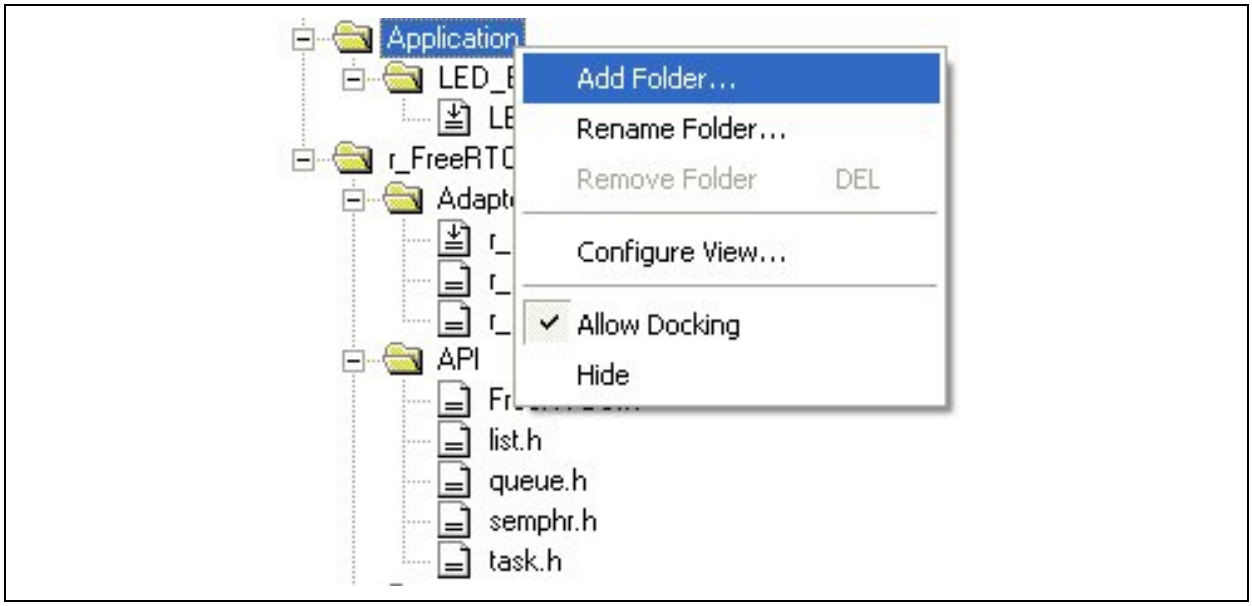
- The code in “MyNewApplication.c” is as follows:

```
/***********************************************************************
*           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);
      }
}
```

- Applications that are to be added to the RI-600 workspace needs to follow this sequence of steps:

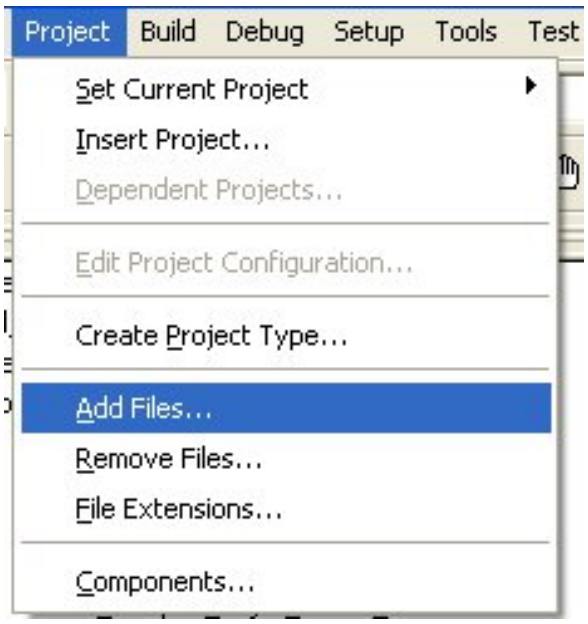


- Connect to the board by accepting the default settings and add the required folder to the Workspace

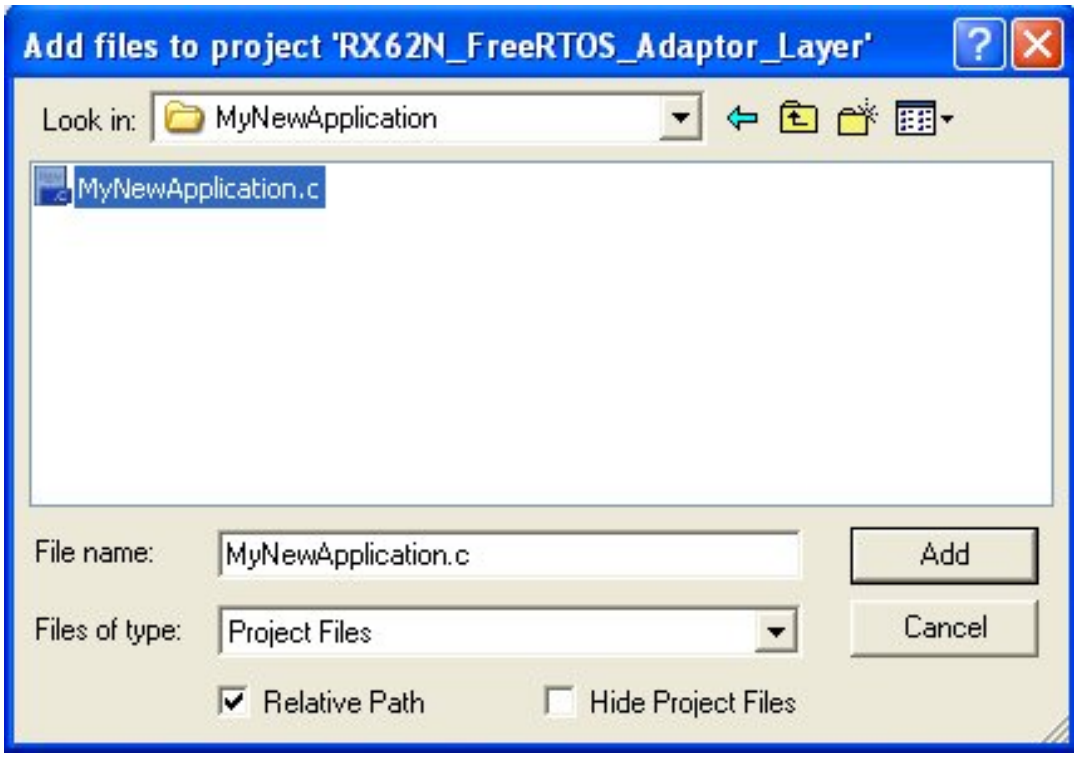


- Name the folder accordingly:

- Add the new C Code file named “MyNewApplication.c” in the \Source\Applications\MyNewApplication folder to the Workspace as follows:



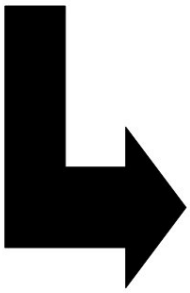
- Select the file to add



- Arrange the code in the Workspace as follows:

- Open file: r_FreeRTOSAdaptor.c and scroll to section: (1) Task Definitions and update the fr_TASKS[] structure as follows:

```
/*********************************************************
*       *** 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 },
};
```

Note that fr_MAX_TASKS has been updated to 2 to accommodate the addition of the new Task. The task should be initialized as:

```
{(pdTASK_CODE) << function label >>,
    << task-name >>,
    << stack depth >>,
    << task priority level >>,
    NULL, NULL, DORMANT, 0, 0 },    // Leave as default
```

- Compile and build the workspace project by selecting Build→Build All.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jul.15.11	—	First edition issued
1.01	Jun.27.12	—	Support RX compiler version 1.0 and Compiler version 1.2
		—	Compiler L Section converted to C Section in Compiler option
1.02	Sep.05.12	—	Updated HEW and CCRX compiler versions
		—	Added Additional Directory structure in document
		—	Updated General Precaution and Notice

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.0