APPLICATION NOTE

Ref:	**FR-191-AN-RB-003**
Date:	**15th October 2010**
To:	**General distribution**
From:	**Richard Barry – Real Time Engineers Ltd.**
Subject:	**A simple FreeRTOS demo for the LPCXpresso LPC1768**

INTRODUCTION

This application note is intended to assist in building, running and understanding the accompanying simple FreeRTOS demo that targets the LPCXpresso LPC1768 hardware.

The scope of the demo is deliberately kept small and at a basic level with the intention of ensuring it is understandable to those who have no previous RTOS experience. Further and more detailed reading and reference material can be found in the following places:

1. The FreeRTOS web site (http://www.FreeRTOS.org)

   This provides a lot more information on the FreeRTOS project, including full licensing terms and full API documentation. The Quick Start Guide is a good place to visit after the home page (http://www.FreeRTOS.org/FreeRTOS-quick-start-guide.html). There is also a download link to the latest FreeRTOS release, which may have been updated since this package was put together.

2. The LPC17xx version of the FreeRTOS book (http://www.FreeRTOS.org/Documentation)

   This provides a tutorial style course on using real time kernels in microcontroller applications. It comes with a further 16 simple example projects that also target the LPCXpresso LPC1768 hardware.

3. The accompanying PDF memo (document number and file name FR-201-MO-RB-003)

   This contains an introductory overview of FreeRTOS and the FreeRTOS project. It provides a summary description of the license terms and the support options available, with links to relevant pages within the FreeRTOS.org site.

BUILDING AND RUNNING THE PROVIDED DEMO

Obtaining the Build Tools

The LPCXpresso IDE is a free tool but does require registration to obtain a license. Instructions are provided on the download page: http://www.code-red-tech.com/lpcxpresso.

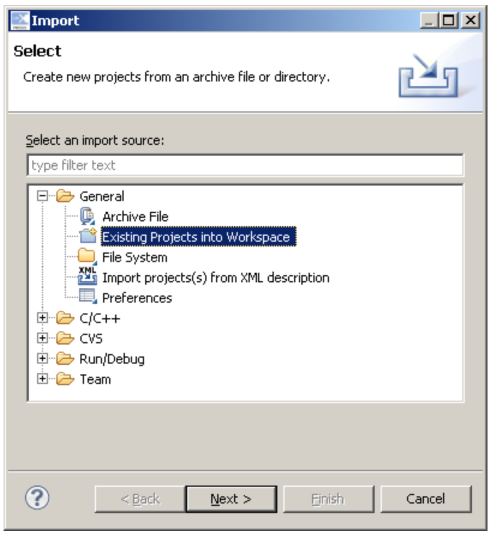
Importing the FreeRTOS Demo Project

The example project is provided in a single .zip file archive. Do not extract the files manually from the archive. Instead, follow the procedure defined below.

1. Start the LPCXpresso IDE. You will be prompted to select a workspace.

2. Select an existing workspace (if you have one) or create a new one by entering the path to where you would like the new workspace to be created.

3. Select 'Import' from the 'File' menu. The dialog box shown in Figure 1 will open.


Import

Select

Create new projects from an archive file or directory.

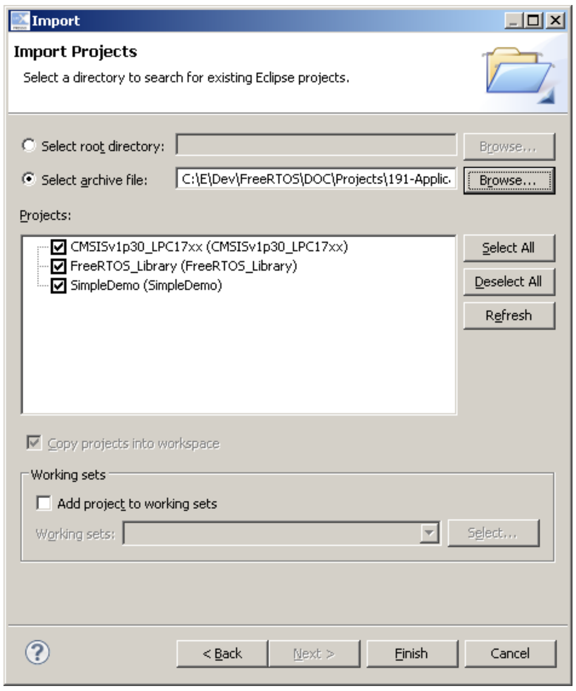
Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Import projects(s) from XML description
  - Preferences
- C/C++
- CVS
- Run/Debug
- Team

Figure 1. Selecting the option to import existing projects into the workspace

4. Select 'Existing Projects into Workspace' and click 'Next'. The dialog box shown in Figure 2 will open.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory:

Select archive file: C:\E\Dev\FreeRTOS\DOC\Projects\191-Applic

Projects:

- [✓] CMSISv1p30_LPC17xx (CMSISv1p30_LPC17xx)
- [✓] FreeRTOS_Library (FreeRTOS_Library)
- [✓] SimpleDemo (SimpleDemo)

[✓] Copy projects into workspace

Working sets

[ ] Add project to working sets

Working sets:

Figure 2. Selecting the .zip archive

5. Choose the 'Select archive file' option, then browse to and select the .zip file that came with this application note.

6. Click ‘Finish’. The three projects shown in Figure 2 will be copied into the workspace directory.

Building the Demo

The Project Explorer window in the LPCXpresso IDE will list one executable project named SimpleDemo, and two library projects named CMSISv1p30_LPC17xx and FreeRTOS_Library respectively. This is shown in Figure 3.

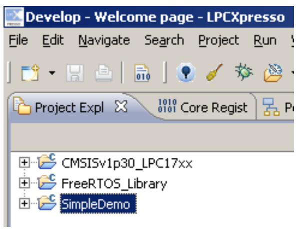

Develop - Welcome page - LPCXpresso
File Edit Navigate Search Project Run
Project Expl
Core Regist
CMSISv1p30_LPC17xx
FreeRTOS_Library
SimpleDemo

Figure 3. The projects listed in the Project Explorer

To build the example, highlight SimpleDemo in the Project Explorer, then select ‘Build Project’ from the IDE ‘Project’ menu. Figure 3 shows SimpleDemo highlighted.

The demo project depends on the library projects; therefore, the libraries will build automatically the first time the demo is built. The FreeRTOS code itself is contained in the FreeRTOS_Library project.

Starting a Debug Session

Ensure the LPCXpresso hardware is connected via USB to the computer running the LPCXpresso IDE, and that SimpleDemo is still selected in the Project Explorer – then click the ‘Debug’ speed button. Figure 4 shows the location of the debug speed button within the IDE.

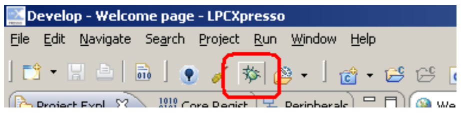

Develop - Welcome page - LPCXpresso
File Edit Navigate Search Project Run Window Help

Figure 4. Locating the 'Debug' speed button within the LPCXpresso IDE

THE SIMPLE DEMO PROJECT

The simple demo project demonstrates task and queue usages only. It is not intended to perform any useful functionality, or demonstrate how best to enhance application design by introducing multi-tasking. Details of other FreeRTOS features (API functions, tracing features, configuration options, diagnostic hook functions, memory management, etc.) can all be found on the FreeRTOS web site and/or in the FreeRTOS book – links to both of which are provided in the introductory section of this application note.

All the functions described below are defined in the main.c source file of the SimpleDemo project. Additional details can also be found in the comments within the source code itself.

The main() Function

main() creates one queue and two tasks before starting the scheduler. It does not execute past the call to start the scheduler.

The queue is used to pass a data value from one task (the queue send task) to the other task (the queue receive task). The queue receive task toggles the LPCXpresso LED each time a data value that equals 100 is received. The LED will toggle every 500 milliseconds while the demo is executing.

The Queue Send Task

The queue send task is implemented by the prvQueueSendTask() function.

prvQueueSendTask() sits in a loop that causes it to continuously block for 500 milliseconds before sending the value 100 to the queue that was created within main().

The Queue Receive Task

The queue receive task is implemented by the prvQueueReceiveTask() function.

prvQueueReceiveTask() sits in a loop that causes it to continuously attempt to read data from the queue that was created within main(), before checking the value of the received data, and toggling the LED if the received value equals 100.

The ‘block time’ parameter passed to the queue receive function specifies that the calling queue receive task should be held in the Blocked state indefinitely to wait for data to be available on the queue. The queue receive task will only leave the Blocked state when the queue send task writes to the queue. Because the queue send task writes to the queue every 500 milliseconds, the queue receive task leaves the blocked state every 500 milliseconds, which in turn also causes the LED to toggle every 500 milliseconds.