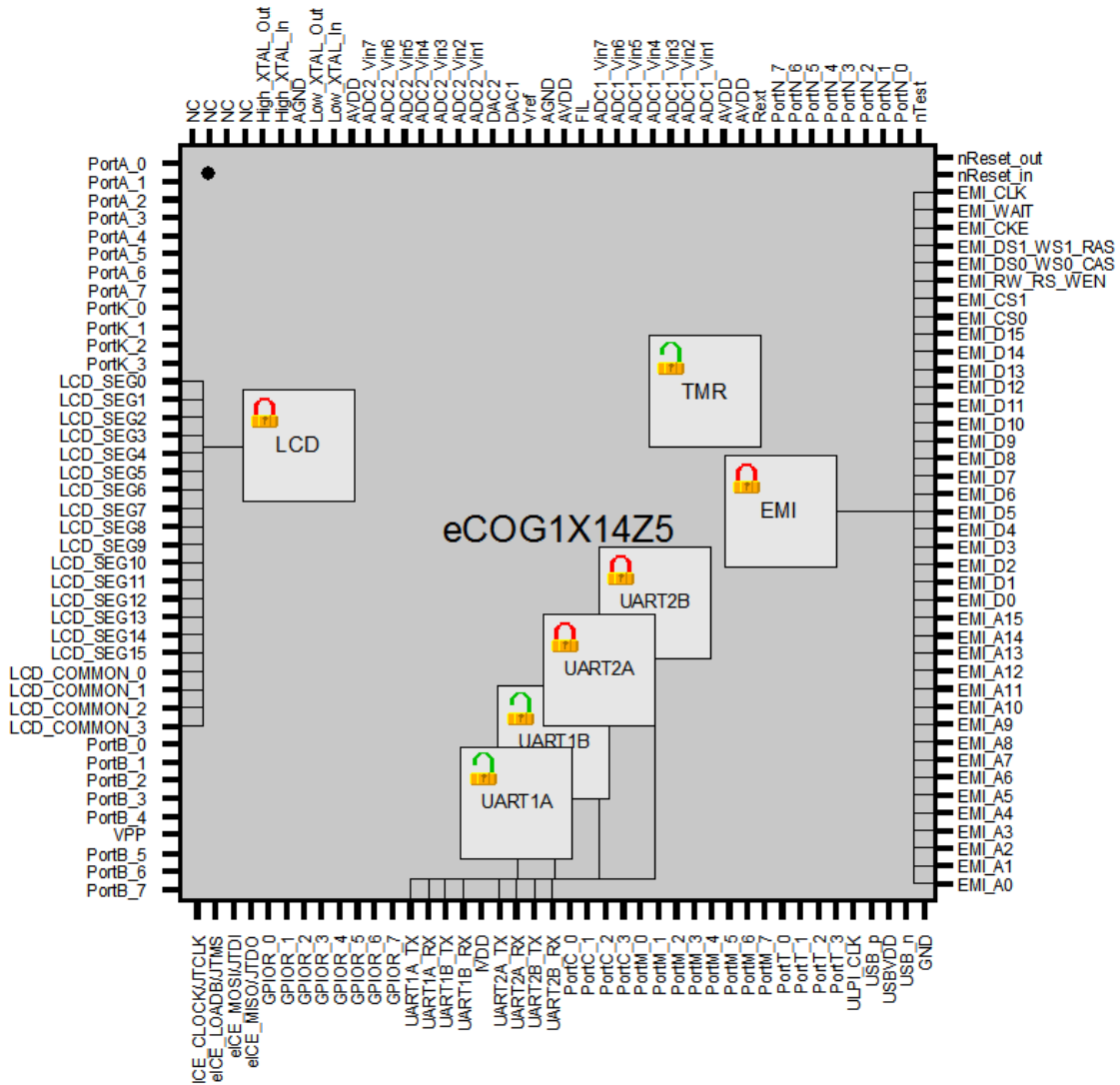




AN072 – Implementation of FreeRTOS on eCOG1X

Version 1.1

This application note describes the implementation of FreeRTOS on the eCOG1X microcontroller.



Confidential and Proprietary Information

©Cyan Technology Ltd, 2009

This document contains confidential and proprietary information of Cyan Technology Ltd and is protected by copyright laws. Its receipt or possession does not convey any rights to reproduce, manufacture, use or sell anything based on information contained within this document.

Cyan Technology™, the Cyan Technology logo and Max-eICE™ are trademarks of Cyan Holdings Ltd. CyanIDE® and eCOG® are registered trademarks of Cyan Holdings Ltd. Cyan Technology Ltd recognises other brand and product names as trademarks or registered trademarks of their respective holders.

Any product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by Cyan Technology Ltd in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Cyan Technology Ltd shall not be liable for any loss or damage arising from the use of any information in this guide, any error or omission in such information, or any incorrect use of the product.

This product is not designed or intended to be used for on-line control of aircraft, aircraft navigation or communications systems or in air traffic control applications or in the design, construction, operation or maintenance of any nuclear facility, or for any medical use related to either life support equipment or any other life-critical application. Cyan Technology Ltd specifically disclaims any express or implied warranty of fitness for any or all of such uses. Ask your sales representative for details.



Revision History

Version	Date	Notes
V1.0	20/03/2008	First release
V1.1	12/01/2009	Software update only.

Contents

1	Introduction	5
2	Glossary	5
3	Licensing	5
4	Installation	6
5	FreeRTOS Source Files	6
6	Using the Application	7
6.1	Creating a New Project.....	7
6.2	Implementing a Task.....	7
6.3	Switching between the Co-operative and Pre-emptive Scheduler	7
6.4	Adding Interrupt Service Routines.....	7
6.5	Application-Specific Configuration.....	7
7	Project Example	8
7.1	Running the Project Example.....	8
8	Testing with the FreeRTOS Demo Application	9
9	Memory Usage	10
10	Changes to the FreeRTOS Source Code	11

1 Introduction

This application note provides an introduction to Cyan's implementation of the FreeRTOS real-time operating system on the eCOG1X microcontroller.

This document is not a tutorial on how to use FreeRTOS or multi-tasking real-time operating systems in general. Further documentation including a full description of the API is available on the FreeRTOS web site at <<http://www.freertos.org>>.

2 Glossary

A table of abbreviations and terms used in this document.

API	Application Programming Interface
CPU	Central Processor Unit
eCOG1X	Cyan Technology target microcontroller.
DUART	Dual Universal Asynchronous Receiver/Transmitter.
I/O	Input/Output
ISR	Interrupt Service Routine
LED	Light-Emitting Diode
RAM	Random Access Memory
RTOS	Real-Time Operating System
UART	Universal Asynchronous Receiver / Transmitter

3 Licensing

FreeRTOS is licensed under a modified GPL and can be used in commercial applications under this license. An alternative commercial license option is also available if required.

The FreeRTOS source code is licensed by the GNU General Public License (GPL) with an exception. The full text of the GPL and the text of the exception are available on the FreeRTOS.org web site.

The exception permits the source code of applications that use FreeRTOS solely through the API published on the FreeRTOS.org web site to remain closed source, thus permitting the use of FreeRTOS in commercial applications without requiring that the whole application be open sourced.

4 Installation

Download the zip file *FreeRTOS471_1X.zip* and extract the contents to the CyanIDE projects directory. This zip file contains:

- FreeRTOS version 4.7.1 kernel files.
- FreeRTOS version 4.7.1 demo application files.
- FreeRTOS project template for CyanIDE.
- CyanIDE project file for the eCOG1X development board.
- Files specific to the eCOG1X port.

Copy the directories *Demo*, *License*, *Source* and *TraceCon*, to a new directory *FreeRTOS* in the CyanIDE installation directory. These directories contain the common FreeRTOS source code. They are shared and do not change for different projects.

The directory *<Demo\ECOG1X_NCC>* contains the example project for the eCOG1X development board.

The FreeRTOS project template allows the user to create a new CyanIDE project and application based on the FreeRTOS kernel. To use this template, copy and paste the *FreeRTOS Project* directory to the *<templates\ecog1XProjects>* sub-directory in the CyanIDE installation directory.

5 FreeRTOS Source Files

The FreeRTOS source files can be obtained from the FreeRTOS.org website.

- Go to <http://www.freertos.org>.
- From the menu frame on the left, click on the “Downloads” item.
- In the new window, click on the link “Download the RTOS source code from here”.
- Download and extract the files.

The FreeRTOS demo application is a test suite for verifying a port to a specific processor. Two types of implementation are provided. The functionalities of the two implementations are similar, but the *Demo\Common\Minimal* (referred to as *minimal* in the rest of this document) uses less memory and does not contain console I/O. The test scenario of the demo application is described in the FreeRTOS.org website section “Demo application -> Standard files”. The *minimal* demo application is used to test the eCOG1X port, as described in section 8.

6 Using the Application

6.1 Creating a New Project

A CyanIDE project template is provided to create new projects based on the FreeRTOS kernel. To create a new project in CyanIDE, from the main menu select *Project->New*. In the New Project dialogue, select *eCOG1X Projects* in the left pane, then select the *FreeRTOS Project* icon in the right pane. Name and create the new project. The relevant files are added to the Navigator view in CyanIDE.

In the Navigator view, the FreeRTOS kernel files and some files specific to the eCOG1X port are located in the subfolders under *FreeRTOS*. The configuration options required to tailor this project to a target application are described later in this section.

6.2 Implementing a Task

Tasks are created by calling *xTaskCreate()* and deleted by calling *vTaskDelete()*. Tasks should be created before calling the *vTaskStartScheduler()* function, as illustrated in the comments in the *main.c* file. A task is a function that takes a *void* pointer as its parameter and returns *void*. For examples on how to use tasks, see the FreeRTOS demo application files.

6.3 Switching between the Co-operative and Pre-emptive Scheduler

The symbol definition *configUSE_PREEMPTION* in the include file *FreeRTOSConfig.h* should be set to '0' to use the co-operative scheduler or '1' to use the pre-emptive scheduler.

6.4 Adding Interrupt Service Routines

Additional interrupt handlers should be declared and added to the interrupt vector table in the file *isr_table.c*, which can be found in the *<C Files\port>* folder of the Navigator pane.

Interrupt service routines are implemented as normal C functions. They are declared as

```
portDeclareISR isr_name(void *pvParameters);
```

The usual CyanIDE form for interrupt service routines

```
void __irq_entry isr_name(void);
```

should not be used.

At the end of an interrupt service routine, use one of the two functions *portExit_ISR()* or *portYield_ISR()*. The *portExit_ISR()* function returns to the task that was running before the interrupt was serviced, while the *portYield_ISR()* function calls the task scheduler before returning and may cause a task switch.

6.5 Application-Specific Configuration

The file *FreeRTOSConfig.h* in the *<H Files\port>* folder of the Navigator pane contains some configurable parameters, including the stack size, heap memory size, CPU clock speed. This allows the FreeRTOS kernel to be tailored to a specific application and target system. These parameters are described in the FreeRTOS.org website section "Customisation". For example, to change the heap memory, redefine the symbol *configTOTAL_HEAP_SIZE* in *FreeRTOSConfig.h* and the symbol *HEAP_SIZE* in the file *cstartup.asm*.

7 Project Example

An example project is provided to demonstrate the use of the FreeRTOS demo application on the eCOG1X development board.

7.1 Running the Project Example

The project file *ECOG1X_NCC.cyp* was developed using CyanIDE version 1.4.4 and is configured to run on the eCOG1X development board with the eCOG1X14Z5 daughter board. The FreeRTOS demo applications are called from *main.c*.

If a serial terminal (such as HyperTerminal) is used, it should be connected to the UART 1A serial port and configured for 115200 baud, 8 data bits, no parity and 1 stop bit.

A loopback connector is required to run the FreeRTOS *comtest* test case. Pins two and three on a standard 9-way 'D' connector should be connected together. The test case serial port does not use any flow control and should be connected to UART 1B.

The project file *ECOG1X_NCC.cyp* should be opened from within CyanIDE. Select *Run (F5)* to build the application and download it to the eCOG1X target system. If the demo applications are executing correctly, the seventh LED on the eCOG1X development board flashes regularly at a period defined by the symbol *mainNO_ERROR_FLASH_PERIOD*. However, if any of the demo applications have not been executed or have reported an error, the flash rate of the LED is increased to *mainERROR_FLASH_PERIOD*. These two constants are defined as 3 seconds and 300 milliseconds respectively in *main.c*.

8 Testing with the FreeRTOS Demo Application

A total of 14 task-based test cases from the *minimal* set are used to test the RTOS port.

When the co-operative scheduler is used, 13 of these 14 test cases are executed, excluding the *blocktim* test case. This test case is not designed for use with the co-operative scheduler. It checks the timing characteristics of tasks of varying priorities with the pre-emptive scheduler.

When the pre-emptive scheduler is used, all 14 test cases are executed, spawning a total of 47 tasks.

An error checking task *vErrorChecks* is created to check whether all the tasks are still running without error. This task determines the flash rate of the seventh LED on the eCOG1X development board.

The flashing first three LEDs on the eCOG1X development board indicate that the *flash* test is running. The fourth and fifth LEDs indicate that the transmitting and receiving tasks of the *comtest* test are running.

Another two co-routine based test cases from the *minimal* set, *crhook* and *crflash*, are also used to test the eCOG1X port. The source files for these co-routine tests are not included in the example project file. To use these two test cases, add the test case source files to the project and set the symbol *configUSE_CO_ROUTINES* in *FreeRTOSConfig.h* to '1'.

9 Memory Usage

The FreeRTOS kernel uses the following amounts of memory:

Code	
Total Memory Size	98304 (Bytes)
Memory Used	10910 (Bytes) (11.1%)
Memory Available	87394 (Bytes) (88.9%)
Constants	
Total Memory Size	32768 (Bytes)
Memory Used	526 (Bytes) (1.605%)
Memory Available	32242 (Bytes) (98.39%)
Variables (no tasks running)	
Total Memory Size	16384 (Bytes)
Memory Used	1402 (Bytes) (8.56%)
Memory Available	14982 (Bytes) (91.44%)
Additional memory per task	568 bytes

The FreeRTOS kernel and the test application together use the following amounts of memory:

Code	
Total Memory Size	98304 (Bytes)
Memory Used	34034 (Bytes) (34.62%)
Memory Available	64270 (Bytes) (65.38%)
Constants	
Total Memory Size	32768 (Bytes)
Memory Used	1140 (Bytes) (3.479%)
Memory Available	31628 (Bytes) (96.52%)
Variables	
Total Memory Size	49152 (Bytes)
Memory Used	36478 (Bytes) (74.21%)
Memory Available	12674 (Bytes) (25.79%)

Note that the total memory available for variables has been increased by adding additional external memory to the system configuration for the test application, as the data space required is too large to fit into internal memory alone.

10 Changes to the FreeRTOS Source Code

Some changes have been made to the FreeRTOS source files to work on the eCOG1X.

1. The definition of the queue used by the scheduler, *xQUEUE*, has to be repeated in the API header file *queue.h*. The queue handler, *xQueueHandle*, is defined as a pointer to the *xQUEUE* structure in the file *queue.c*. The API header file originally defines *xQueueHandle* as a pointer to *void*. Due to the implementation of the CyanIDE V1.4 compiler and the eCOG1X architecture, the *void* pointer is treated as a 17-bit address and the queue handler is a 16-bit address. The use of casting in the program might have resulted in an address error. This was encountered when running a demo application that uses queue management.

This problem should be resolved in a later version of CyanIDE.

2. A *signed char* cast is required when passing the string argument to *xtaskcreate()* in the demo application *recmutex*. Although the CyanIDE compiler property has been configured to treat plain *char* as *signed*, this is ignored by the current version of the C compiler.