FreeRTOS for Tiva C TM4C123x - Keil Project Setup

Written by: jspicer-ltu
https://github.com/jspicer-ltu

INTRODUCTION

In this article, I'll discuss two methods for setting up a new FreeRTOS project for the TM4C123x device in Keil µVision. Method 1 uses the CMSIS libraries and FreeRTOS software packs provided by µVision, and Method 2 incorporates the FreeRTOS source code from freertos.org directly into a project without using CMSIS. Method 1 requires fewer steps and is the quicker way to get started. However, it requires you to use the CMSIS startup and system code, which may not be preferable for all projects, particularly if you want to use your own startup and driver implementations. Method 2 is better in this regard, but is more manual and requires additional steps to get started. I recommend setting up a project using Method 1 first before trying Method 2. You will then be able reuse the FreeRTOSConfig.h file that Method 1 provides instead of creating this file from scratch. This is the approach I will take when covering Method 2.

METHOD 1 – CMSIS

1. Open µVision and select Manage->Pack Installer from the Project menu. The Pack Installer window should appear.
2. On the Devices tab, choose Texas Instruments|Tiva C Series. The Pack Keil::TM4C_DFP should appear on the right-side pane, under the Packs tab. If you do not have this pack installed or it is not up to date, then install or update it.
3. Under the Generic section of the Packs list, expand ARM::CMSIS-FreeRTOS. For this tutorial I will be using FreeRTOS 10.0.1. If you do not already have this pack installed, go ahead and install it. If you don't see it listed, you may need to update the list of available packs.
4. Close the Pack Installer window and select 'New µVision Project' from the Project menu. Choose a location and Save it.
5. In the 'Select Device' dialog that follows, select Texas Instruments|Tiva C Series|TM4C123x Series|TM4C123GH6PM and click OK. (Please verify that the model ID on your Tiva board is the same, or select differently if this is the case.)
6. In the Manage Run-Time Environment dialog that follows, select the following options: RTOS:
    - Config
    - Core
    - Heap -> Heap_1 (for this tutorial)
    - Timers

   Device:

    - Startup

   CMSIS:

    - CORE

Close the dialog. You should now have a (mostly) functional FreeRTOS project.

7. Right-click the source group folder (e.g. 'Source Group 1') and choose Add New Item. Select 'C File (.c)' and name it main.c. Click Add and close the dialog.
8. Add the following code snipped to main.c:

```
#include <stdint.h>
#include <FreeRTOS.h>
#include <task.h>

void vPeriodicTask(void *pvParameters)
{
	// Establish the task's period.
	const TickType_t xDelay = pdMS_TO_TICKS(1000);
	TickType_t xLastWakeTime = xTaskGetTickCount();

	for (;;) {

		// Block until the next release time.
		vTaskDelayUntil(&xLastWakeTime, xDelay);
	}
}

int main()
{
	xTaskCreate(vPeriodicTask, "My Task", 256, NULL, 1, NULL);

	// Startup of the FreeRTOS scheduler.  The program should block here.
	vTaskStartScheduler();

	// The following line should never be reached.  Failure to allocate enough
	//	memory from the heap would be one reason.
	for (;;);
}
```

9. Open the FreeRTOSConfig.h file under the RTOS project node in the Project tree and add #include "TM4C123GH6PM.h" to the list of include files *at the top of the file*, i.e. just after #include <stdint.h>. This step is necessary so that the configPRIO_BITS macro is defined with __NVIC_PRIO_BITS, which has a value of 3. Any other value will cause a runtime assertion, and the program will appear to hang in xPortStartScheduler() in the port.c file.
10. Build the project. There should be no errors or warnings.
11. Set two breakpoints in the vPeriodicTask function: one on the first line, and the second inside the loop where it calls vTaskDeleteUntil.
12. Download the project to the Tiva board and Start a debug session. The debugger should hit the first breakpoint in vPeriodicTask, and then hit the second breakpoint every second. If it is doing this, then it means everything is working successfully. *Congratulations!*

METHOD 2 – MANUAL SETUP

1. Select 'New µVision Project' from the Project menu. Choose a location and Save it. Remember the location, because you'll need it in subsequent steps.
2. In the Select Device dialog that follows, select Texas Instruments|Tiva C Series|TM4C123x Series|TM4C123GH6PM and click OK. (Please verify that the model ID on your Tiva board is the same, or select differently if this is the case.)
3. The Manage Run-Time Environment dialog will appear next. We will not choose anything from it. Simply click OK. You will now have an empty project.
4. If you haven't done so already, go to the freertos.org website and download the source code for FreeRTOS. We will be using version 10.0.1 for this tutorial. Unzip the source to a folder on your computer, e.g. ~\Downloads\FreeRTOSv10.0.1.
5. Using Windows File Explorer, open the FreeRTOSv10.0.1 source code folder and copy the FreeRTOS subfolder into the project folder that you created in Step #1. In other words, we are copying the FreeRTOS subfolder from the source distribution directly into your Keil project folder.
6. Open your µVision project folder (using Windows File Explorer) and go into the FreeRTOS subfolder that you copied from the previous step. We are going to delete some unused folders and keep the ones we need:
   a. Delete the Demo folder
   b. Inside the Source\portable folder, delete everything except the MemMang and RVDS folders.
   c. Inside the Source\portable\RVDS folder, delete everything except the ARM_CM4F folder.
7. Go back into µVision, and right-click the Target folder (e.g. 'Target 1') in the Project tree. Choose Add Group. Rename the group, 'FreeRTOS'. Right-click on it, and select Add Existing Files. Add the following files from your project root directory:
   a. .\FreeRTOS\Source\tasks.c
   b. .\FreeRTOS\Source\list.c
   c. .\FreeRTOS\Source\queue.c
   d. .\FreeRTOS\Source\timers.c
   e. .\FreeRTOS\Source\portable\RVDS\ARM_CM4F\port.c
   f. .\FreeRTOS\Source\portable\MemMang\heap_1.c
8. Now you will need to provide a FreeRTOSConfig.h file. If you followed Method 1, you can reuse that one here. It will likely be in the .\RTE\RTOS folder of your Method 1 project. Make a copy of it, and paste it into the *root* of your Method 2 project folder using Windows File Explorer. Next, go back into µVision, and right-click the source group folder (e.g. 'Source Group 1'). Choose Add Existing Files and select FreeRTOSConfig.h from the root of your project folder.
9. Open the FreeRTOSConfig.h file in the Project tree. Locate and *remove* the following #include statements:

   ```
   #include "TM4C123GH6PM.h"
   …
   #include "freertos_evr.h"
   ```

   The first line will be near the top of the file (when it was added for Method 1) and the second will be near the bottom. These files are needed for Method 2 and have CMSIS dependencies that will cause compile errors.

   You must also change the configPRIO_BITS value to 3. Since __NVIC_PRIO_BITS is a CMSIS value, it will not be defined here, so you must set it directly in the #else clause of the conditional statement, as shown below:

```
/* Cortex-M specific definitions. */
#ifdef __NVIC_PRIO_BITS
  /* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
  #define configPRIO_BITS                     __NVIC_PRIO_BITS
#else
  /* 8 priority levels */
  #define configPRIO_BITS                     3
#endif
```

If 4 is left in, FreeRTOS will fail with an assertion and hang runtime, so it must be 3.

10. Click the 'Options for Target' button on the toolbar and select the C/C++ tab. Append the following line to the Include Paths:

    ```
    .;.\FreeRTOS\source\include;.\FreeRTOS\source\portable\RVDS\ARM_CM4F
    ```

11. Next, you will need to add a startup file. For this tutorial, I will be using the startup_rvmdk.S file from the TivaWare C Series EK-TM4C123GXL Kit Software, which can be downloaded from the TI website (http://www.ti.com/tool/SW-TM4C). This startup file is located in the examples\project folder. It is a minimal implementation and mainly contains the interrupt table and reset function. Using Windows Explorer, copy this startup_rvmdk.S file from the TI kit to the root of your µVision project folder. Now go back into µVision, and right-click the source group folder (e.g. 'Source Group 1'). Choose Add Existing Files and select startup_rvmdk.S from the root of your project folder.
12. In the startup_rvmdk.S file, add the following EXTERN declarations somewhere before the EXPORT __Vectors statement:

    EXTERN SVC_Handler
    EXTERN PendSV_Handler
    EXTERN SysTick_Handler

    These are the names of interrupt handlers that FreeRTOS has defined in the port.c file. Now, replace the IntDefaultHandler value in the __Vectors table with these handler names. Use the comment at the end of the DCD statement to find the correct handler. The result should look like this:

```
;*****************************************************************************
**
;
; The vector table.
;
;*****************************************************************************
**

        EXTERN SVC_Handler
```

```
        EXTERN PendSV_Handler
        EXTERN SysTick_Handler

        EXPORT  __Vectors

__Vectors
        DCD    StackMem + Stack          ; Top of Stack
        DCD    Reset_Handler             ; Reset Handler
        DCD    NmiSR                     ; NMI Handler
        DCD    FaultISR                  ; Hard Fault Handler
        DCD    IntDefaultHandler         ; The MPU fault handler
        DCD    IntDefaultHandler         ; The bus fault handler
        DCD    IntDefaultHandler         ; The usage fault handler
        DCD    0                         ; Reserved
        DCD    0                         ; Reserved
        DCD    0                         ; Reserved
        DCD    0                         ; Reserved
        DCD    SVC_Handler               ; SVCall handler
        DCD    IntDefaultHandler         ; Debug monitor handler
        DCD    0                         ; Reserved
        DCD    PendSV_Handler            ; The PendSV handler
        DCD    SysTick_Handler           ; The SysTick handler
        …
```

13. Right-click the source group folder (e.g. 'Source Group 1'). Choose Add New Item. Select 'C File (.c)' and name it main.c. Click Add and close the dialog. Insert the code snippet from Method 1 into the main.c file. Near the top of main.c file, just below the #include statements, declare a variable named SystemCoreClock, and assign it the default bus clock frequency of 16MHz:

    ```
    uint32_t SystemCoreClock = 16000000;
    ```

    FreeRTOSConfig.h will refer to this variable to define configCPU_CLOCK_HZ macro. Be sure to change the value to whatever bus frequency you plan to use later on.

14. Build the project. There should be no errors or warnings.
15. Set two breakpoints in the vPeriodicTask function: one on the first line, and the second inside the loop where it calls vTaskDeleteUntil.
16. Download the project to the Tiva board and Start a debug session. The debugger should hit the first breakpoint in vPeriodicTask, and then hit the second breakpoint every second. If it is doing this, then it means everything is working successfully. *Congratulations!*

SUMMARY

By using one of these two methods, you should be able to set up a basic, FreeRTOS project, and then expand on it from there. For more details about creating a new FreeRTOS project from scratch, I recommend reading section 1.4 in the FreeRTOS manual, which covers the procedure that I adapted for the TM4C123. Videos of this tutorial are posted on the YouTube channel, jspicer-ltu.