University of Victoria

Department of Electrical and Computer Engineering

CENG 455
REAL TIME COMPUTER SYSTEMS
DESIGN PROJECTS

Introduction and Project 1

Acknowledgement

The collection of 455 Real Time Computer Systems Design Projects is the result of the hard work of numerous individuals. Without their effort, time, and ideas over the years, it would not have been possible to provide our students the best possible learning experience.

The Department of Electrical and Computer Engineering would like to express its gratitude to these people who have contributed to the design projects in a major way: A. Jooya, S. Khoshbakht, N. Agarwal, S. Campbell, N.J. Dimopoulos, J. Dorocicz, K. Jones, T. Kroeker, E. Laxdal, K.F. Li, L. Palmer, N. Rebenich, T. Torres-Bonet, B. Zakeri, and D. Zhao.

In addition, the Department acknowledges the generous financial support from the Wighton Engineering Product Development Fund and the Curriculum Development Fund from the UVic Design Engineering Office.

1. Introduction

The CENG 455 Laboratory component consists of one simple introduction project and two medium-sized design projects. The purpose of the projects is to introduce you to designing and implementing hardware and software for applications in a real time, multitasking environment.

1.1 Marking

The mark distribution for the laboratory component is:

Project 1	5
Project 2	45
Project 3	50

Total	100

1.2 Time Table

The timeline and milestones for the projects are given in Table 1.

Table 1. Time table

Date	Description
First scheduled lab	Project 1 must be demonstrated by 5:30 PM; no late Project 1 will be accepted. No report is required for Project 1. Project 2 will be assigned.
Fourth scheduled lab	Project 2 must be demonstrated by 5:30 PM; any Project 2 demonstrated after this time will be subjected to an initial penalty of -1 mark, and an additional -1 mark for every 24 hours. Project 3 will be announced.
72 hours after the Fourth scheduled lab	The last day for demonstrating a late Project 2. After 5:30 PM, no demonstrations of Project 2 will be accepted, and a failing grade for the COURSE will be given.
One week after Fourth scheduled lab	Reports for Project 2 must be handed in by 5:30 PM. Late reports will be penalized the same way as late demonstration.
72 hours after above deadline	The last day for submitting reports for Project 2.
Eighth scheduled lab	'Demo days' for Project 3. (Late penalties as Project 2).
One week after the Demo	Reports for Project 3 due. (Late penalties as Project 2).

1.3 Notes

- All three projects must be demonstrated to the lab instructor, and for Project 3, to the course instructor as well, in order to have a mark assigned. A project report without a demonstration will result in a 0 mark, and thus a **failing grade for the COURSE**. An acceptable demonstration is one that presents to the lab/course instructors how much of your project works according to specification, and shows the project source code. You can demonstrate the project even if it is not working completely or correctly. Be prepared to answer questions.
    - The stated deadlines are firm and will not be extended.
    - The projects must be performed in groups of no more than three.
    - The lab instructor will in general not be available outside of the lab session hours. If you want to see the lab instructor outside of the lab session you must arrange it via e-mail. The lab instructor will not be available on a drop-in basis.

2. Overview

This section gives a general overview of the hardware and software development platform used in the lab.

2.1 The Development Environment

No setup of the development environment should be required. The PCs in the lab run Windows. You login to Windows using your UNIX username and password, the home directory of your UNIX account will be mounted automatically as *"drive M"*. It is a good practice to save your work to your *"M drive"* regularly. Atollic TrueSTUDIO® for STM32 is available on your workstation.

2.2 Hardware

The hardware platform includes the STM32F4 Discovery board. Relevant manuals and references can be found on the course (www.ece.uvic.ca/~ceng455) and the lab (www.ece.uvic.ca/~ceng455l) web pages.

2.3 Software

The software development platform Atollic TrueSTUDIO® for STM32 is used to build, run, and debug your FreeRTOS applications. Relevant manuals and references can be found on the course and the lab web pages.

Project 1

Introduction to TrueSTUDIO and FreeRTOS

The purpose of this project is to introduce you to Atollic TrueSTUDIO IDE and FreeRTOS.

P1.1 Marking

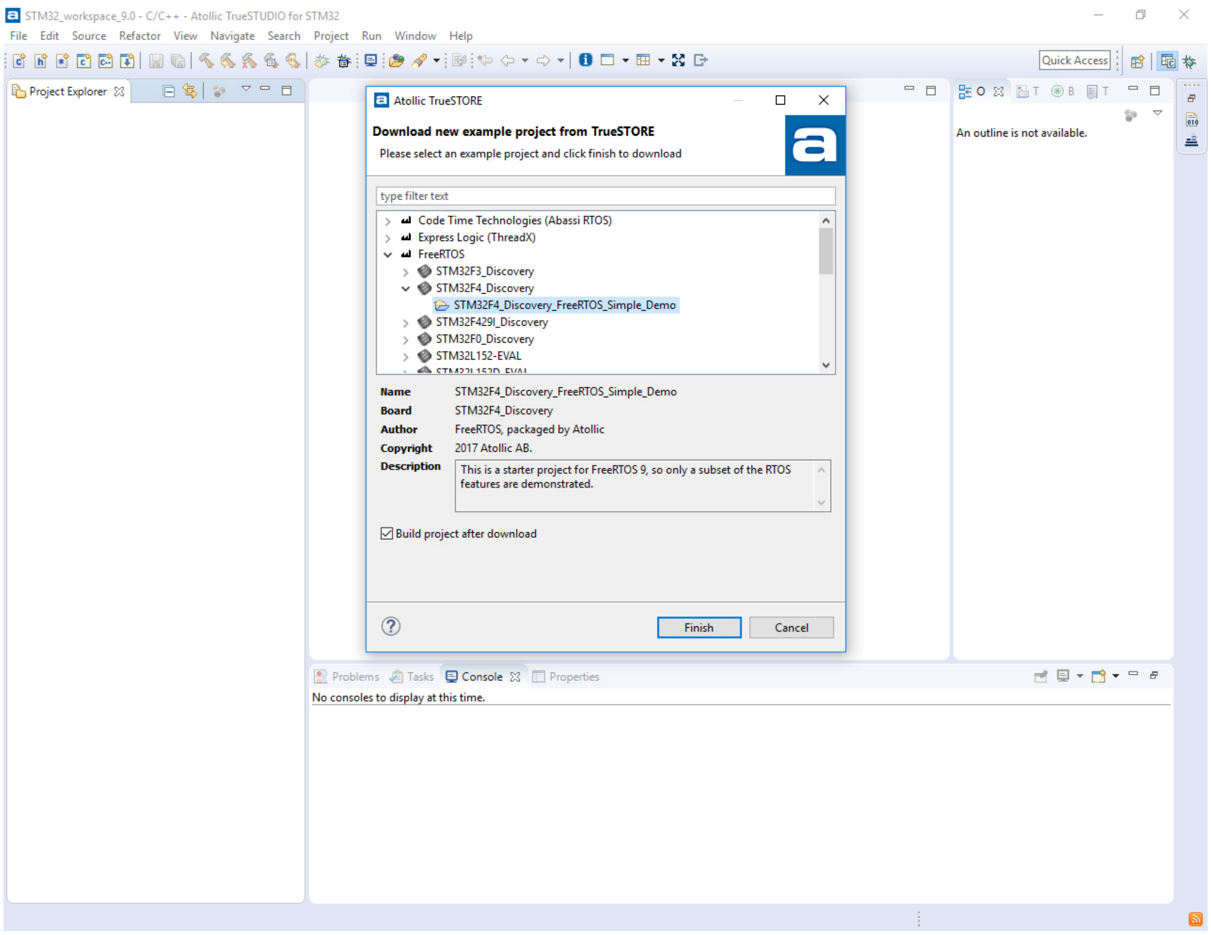
This project is worth 5% of your Lab mark. This mark will be given if you demonstrate a successful implementation of the project before the end of the first lab session. There is no write up required for this project.

P1.2 Introduction to TrueSTUDIO and FreeRTOS

In this section you will learn how to use TrueSTUDIO IDE to create a new project, compile, build and debug it. To do this, you will develop a code based on FreeRTOS and practice some of the basic concepts of real time operating system.

TrueSTUDIO is built on Eclipse and the interface and menus is the same as of Eclipse which you are already familiar with.

1. Run Atollic TrueSTUDIO for STM32 using the icon shortcut on the desktop. Note the workspace path and click OK.
2. Go to ***File*** menu, select ***new***, then select ***Download new example project from TrueSTORE*** and wait for Atollic TrueSTORE window to appear on the screen.
3. Select ***FreeRTOS***, select ***STM32F4_Discovery*** and select the ***STM32F4_Discovery_FreeRTOS_Simple_Demo***. Make sure that the option "build project after download" is checked (see Figure1). Click on Finish button.


Atollic TrueSTORE

Download new example project from TrueSTORE

Please select an example project and click finish to download

- Code Time Technologies (Abassi RTOS)
- Express Logic (ThreadX)
- FreeRTOS
  - STM32F3_Discovery
  - STM32F4_Discovery
    - STM32F4_Discovery_FreeRTOS_Simple_Demo
  - STM32F429I_Discovery
  - STM32F0_Discovery
  - STM32L152-EVAL

Name	STM32F4_Discovery_FreeRTOS_Simple_Demo
Board	STM32F4_Discovery
Author	FreeRTOS, packaged by Atollic
Copyright	2017 Atollic AB.
Description	This is a starter project for FreeRTOS 9, so only a subset of the RTOS features are demonstrated.

☑ Build project after download

An outline is not available.

No consoles to display at this time.

STM32_workspace_9.0 - C/C++ - Atollic TrueSTUDIO for STM32
File Edit Source Refactor View Navigate Search Project Run Window Help
Quick Access
Project Explorer
type filter text
Finish
Cancel
Problems Tasks Console Properties

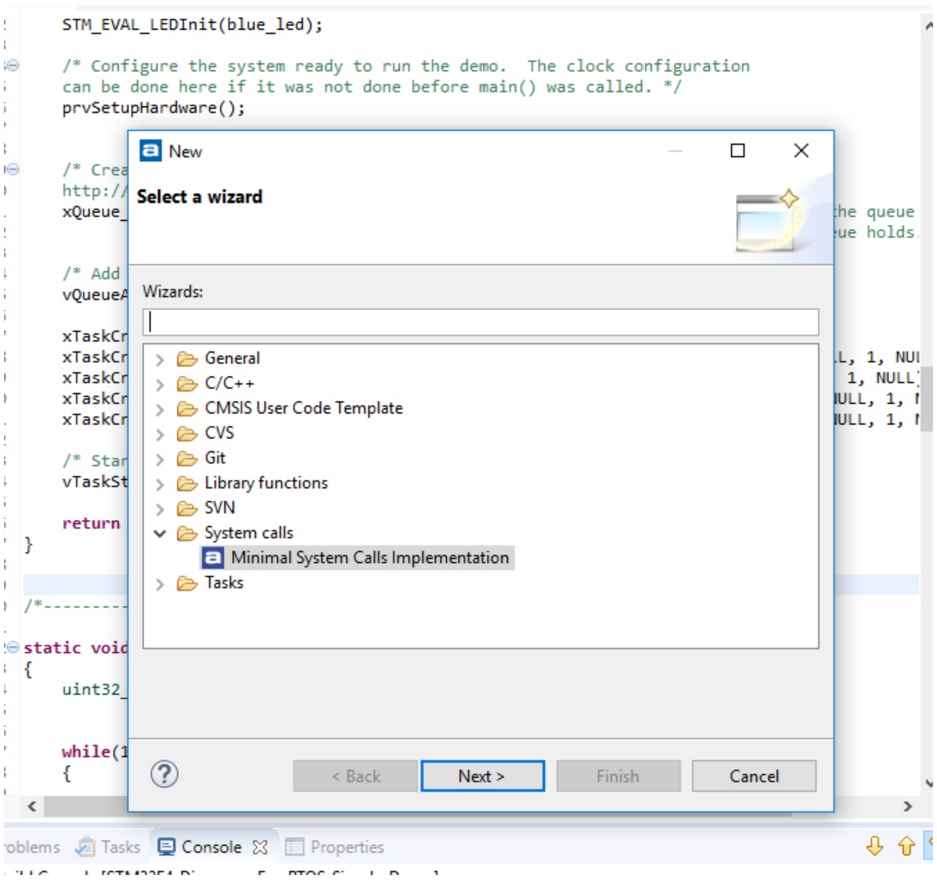
Figure 1. Download a sample FreeRTOS project from TrueSTORE.

4. TrueSTUDIO will download and build the demo project. Make sure the build process was successful.
5. Download the “`Lab1.c`” from coursespaces and replace the content of the `main.c` file with the code in “`Lab1.c`”.
6. Open the “`FreeRTOSConfig.h`“ file and make the following change:

```
#define configUSE_TICK_HOOK      1
```

7. Build the project and make sure that the built process was successful.
8. Spend 15 minutes to read the code and discuss it with your teammate. Ask your TA if you have any question.
9. Inform your TA that you have read the code and ready to move to the next step. Your TA may ask you question related to the code to make sure that you have fully understood the code.
10. On the left panel, right click on the ***src*** folder, from the pop-up menu, select *new* and then select *other*…. On the new screen, open *System calls* and choose *Minimum System*

Calls Implementation as in Figure2.



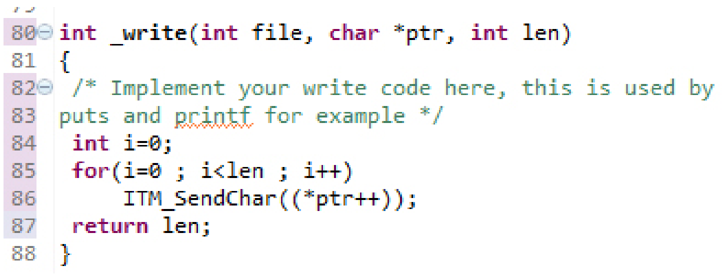
```
STM_EVAL_LEDInit(blue_led);

/* Configure the system ready to run the demo.  The clock configuration
can be done here if it was not done before main() was called. */
prvSetupHardware();
```

Figure 2. Adding a system call file to the project.

11. Click *next* and then *Finish*. The syscall.c file has been added to the project under ***src*** folder.
12. Open the syscalls.c file and add the following code to the top of the file.

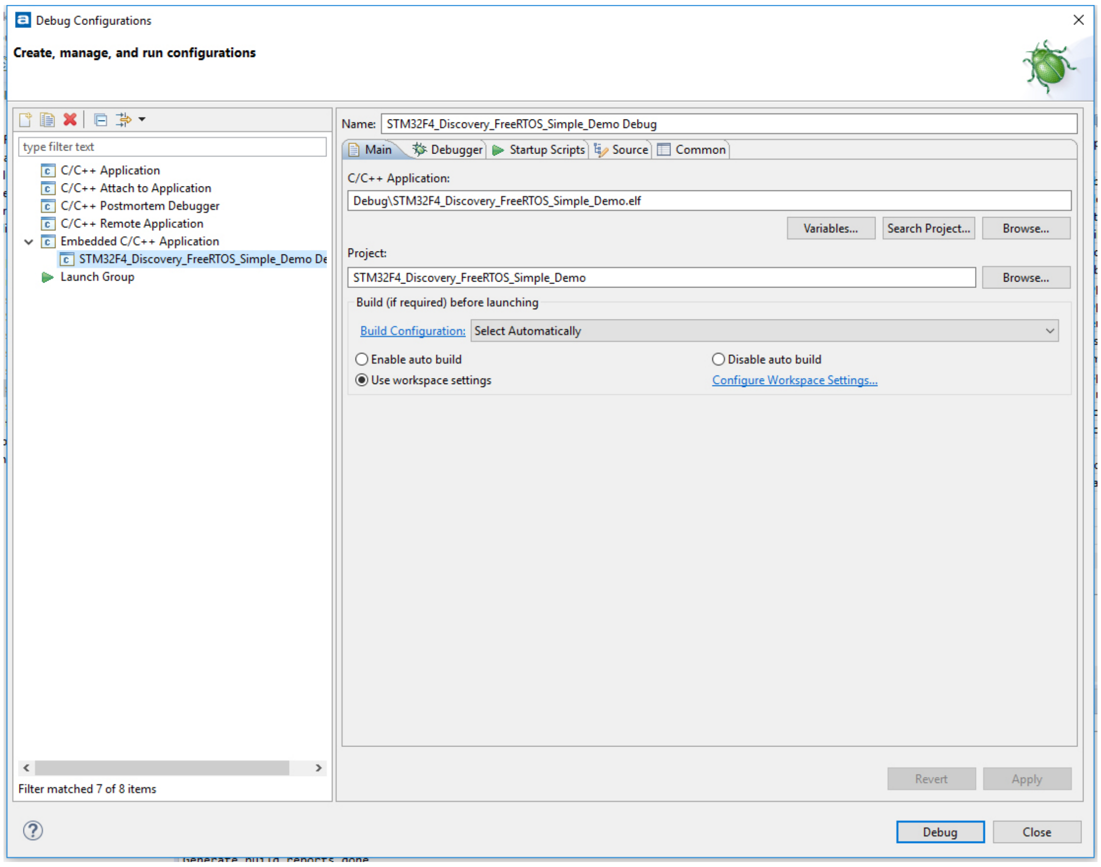
    #include "stm32f4xx.h"

13. Replace the **`_write function with the following code.`**



```
int _write(int file, char *ptr, int len)
{
 /* Implement your write code here, this is used by
puts and printf for example */
 int i=0;
 for(i=0 ; i<len ; i++)
     ITM_SendChar((*ptr++));
 return len;
}
```

14. Build the project. It should be built without any error or warning.
15. Connect the STM32F4_Discovery board to the PC using a USB cable. The red power LED should light up.
16. Now you are all set to start debugging the code. From the ***Run menu*** select ***debug configurations***. On the left panel, under ***Embedded C/C++ Application***, there is a debug configuration with the same name as the project's. select the debug configuration as in Figure3.


Debug Configurations

Create, manage, and run configurations

Name: STM32F4_Discovery_FreeRTOS_Simple_Demo Debug

C/C++ Application: Debug\STM32F4_Discovery_FreeRTOS_Simple_Demo.elf

Project: STM32F4_Discovery_FreeRTOS_Simple_Demo

Build (if required) before launching

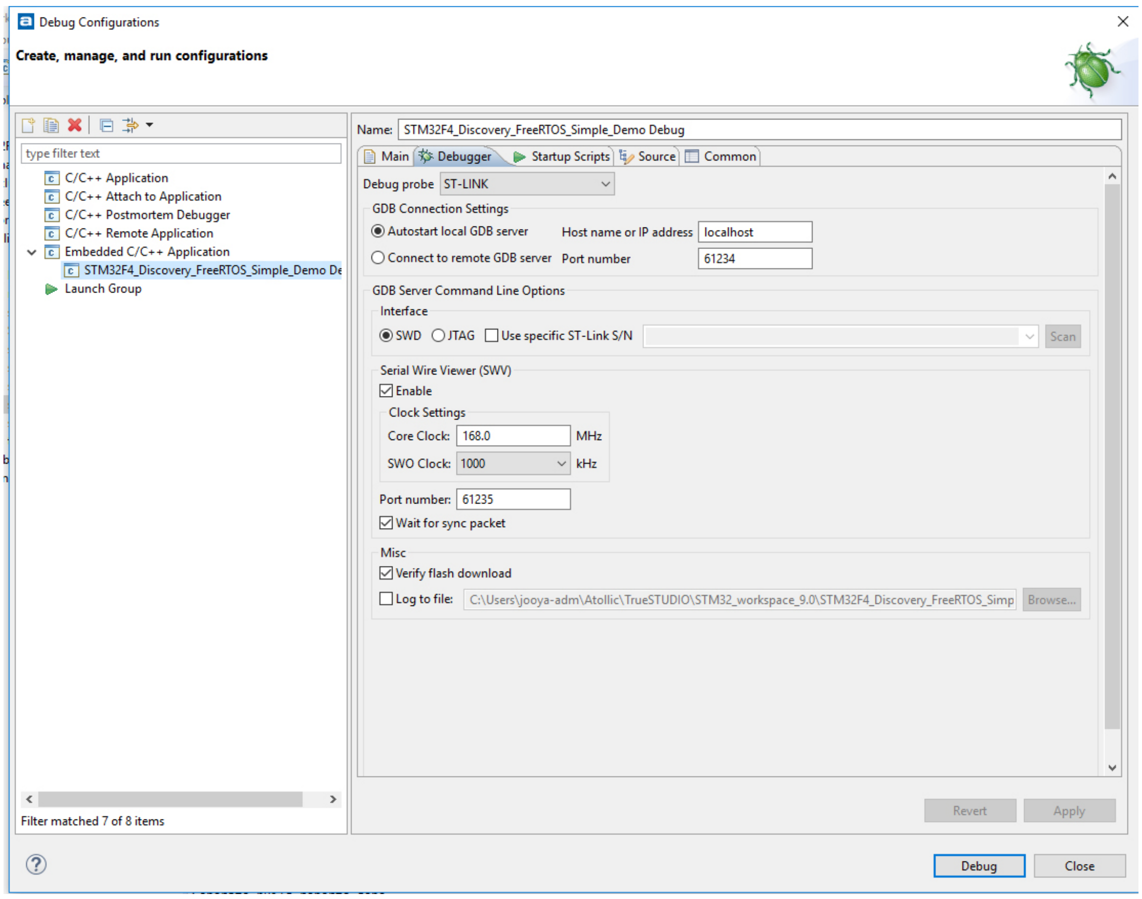
Build Configuration: Select Automatically

( ) Enable auto build
( ) Disable auto build
(●) Use workspace settings

Filter matched 7 of 8 items

Figure 3. Debug configuration window.

17. Go to Debugger tab and check Enable under ***Serial Wire Viewer*** (***SWV***) as in Figure4.


Debug Configurations

Create, manage, and run configurations

Name: STM32F4_Discovery_FreeRTOS_Simple_Demo Debug

Debug probe: ST-LINK

GDB Connection Settings
- (●) Autostart local GDB server — Host name or IP address: localhost
- ( ) Connect to remote GDB server — Port number: 61234

GDB Server Command Line Options
- Interface: (●) SWD ( ) JTAG ☐ Use specific ST-Link S/N

Serial Wire Viewer (SWV)
- ☑ Enable
- Clock Settings: Core Clock: 168.0 MHz; SWO Clock: 1000 kHz
- Port number: 61235
- ☑ Wait for sync packet

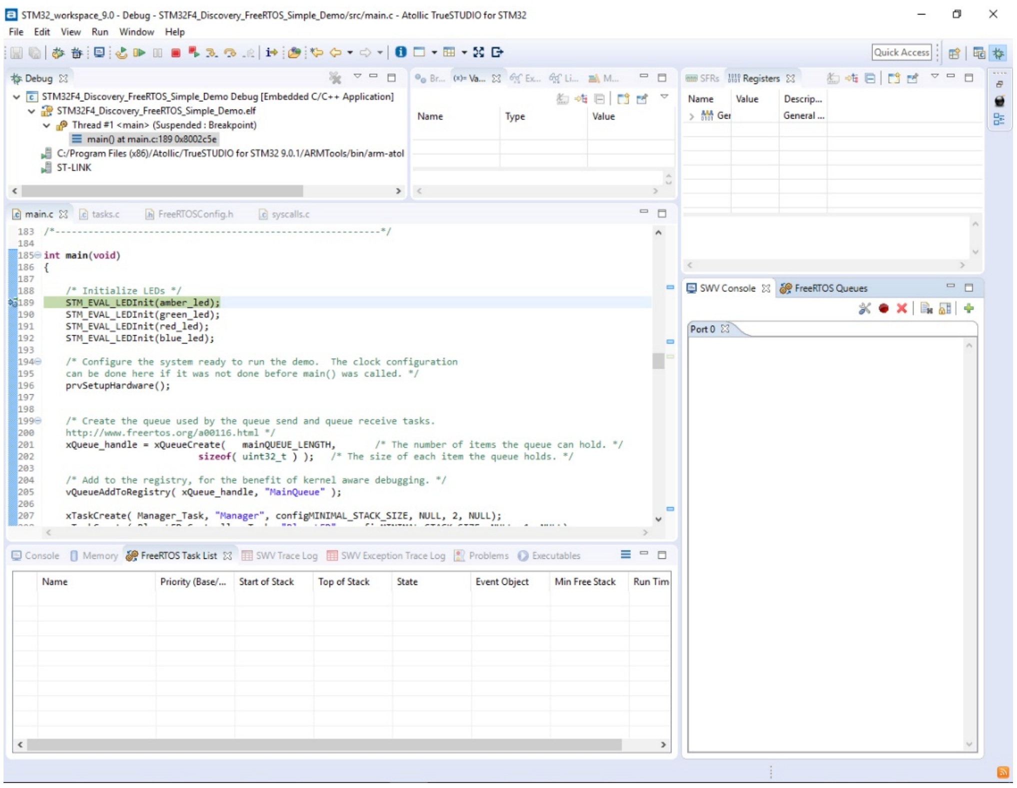
Misc
- ☑ Verify flash download
- ☐ Log to file: C:\Users\jooya-adm\Atollic\TrueSTUDIO\STM32_workspace_9.0\STM32F4_Discovery_FreeRTOS_Simp

Filter matched 7 of 8 items

Figure 4. Enabling SWV.

18. Check the other tabs and make yourself familiar with the debugger options. Do not change other settings and click ***Debug***. TrueSTUDIO switches into debug view.
19. Under ***View*** menu, enable the following options:
    a. select ***FreeRTOS*** and enable ***FreeRTOS Queues.***
    b. select ***FreeRTOS*** and enable ***FreeRTOS Task List***.
    c. Select ***SWV*** and enable ***SWV Console***.

    Your debug interface should look like Figure5.



STM32_workspace_9.0 - Debug - STM32F4_Discovery_FreeRTOS_Simple_Demo/src/main.c - Atollic TrueSTUDIO for STM32

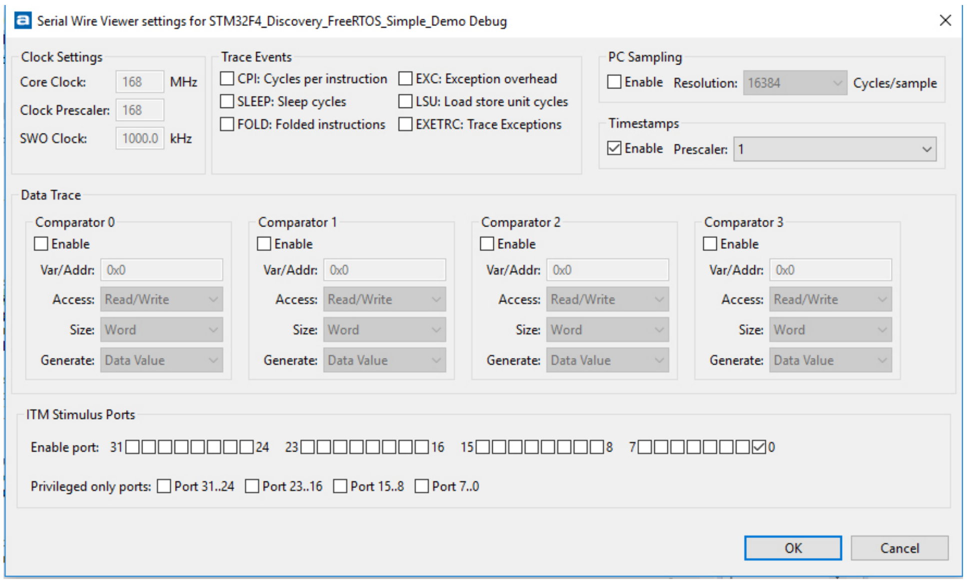
STM32F4_Discovery_FreeRTOS_Simple_Demo Debug [Embedded C/C++ Application]
STM32F4_Discovery_FreeRTOS_Simple_Demo.elf
Thread #1 <main> (Suspended : Breakpoint)
main() at main.c:189 0x8002c5e
C:/Program Files (x86)/Atollic/TrueSTUDIO for STM32 9.0.1/ARMTools/bin/arm-atol
ST-LINK

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

int main(void)
{

    /* Initialize LEDs */
    STM_EVAL_LEDInit(amber_led);
    STM_EVAL_LEDInit(green_led);
    STM_EVAL_LEDInit(red_led);
    STM_EVAL_LEDInit(blue_led);

    /* Configure the system ready to run the demo.  The clock configuration
    can be done here if it was not done before main() was called. */
    prvSetupHardware();


    /* Create the queue used by the queue send and queue receive tasks.
    http://www.freertos.org/a00116.html */
    xQueue_handle = xQueueCreate(   mainQUEUE_LENGTH,       /* The number of items the queue can hold. */
                            sizeof( uint32_t ) );   /* The size of each item the queue holds. */

    /* Add to the registry, for the benefit of kernel aware debugging. */
    vQueueAddToRegistry( xQueue_handle, "MainQueue" );

    xTaskCreate( Manager_Task, "Manager", configMINIMAL_STACK_SIZE, NULL, 2, NULL);
```

Name	Priority (Base/...	Start of Stack	Top of Stack	State	Event Object	Min Free Stack	Run Tim

Figure 5. Debugger view with SWV console and FreeRTOS queue and task list enabled.

20. Click on the gear icon on the ***SWV Console***.
21. Check port0 box under ***ITM Stimulus Ports*** as shown in Figure6. Click ***OK*** to close the window.



Serial Wire Viewer settings for STM32F4_Discovery_FreeRTOS_Simple_Demo Debug

Clock Settings
Core Clock: 168 MHz
Clock Prescaler: 168
SWO Clock: 1000.0 kHz

Trace Events
☐ CPI: Cycles per instruction ☐ EXC: Exception overhead
☐ SLEEP: Sleep cycles ☐ LSU: Load store unit cycles
☐ FOLD: Folded instructions ☐ EXETRC: Trace Exceptions

PC Sampling
☐ Enable Resolution: 16384 Cycles/sample

Timestamps
☑ Enable Prescaler: 1

Data Trace

	Comparator 0	Comparator 1	Comparator 2	Comparator 3
Enable	☐	☐	☐	☐
Var/Addr:	0x0	0x0	0x0	0x0
Access:	Read/Write	Read/Write	Read/Write	Read/Write
Size:	Word	Word	Word	Word
Generate:	Data Value	Data Value	Data Value	Data Value

ITM Stimulus Ports
Enable port: 31☐☐☐☐☐☐☐☐24 23☐☐☐☐☐☐☐☐16 15☐☐☐☐☐☐☐☐8 7☐☐☐☐☐☐☐☑0

Privileged only ports: ☐ Port 31..24 ☐ Port 23..16 ☐ Port 15..8 ☐ Port 7..0

Figure 6. Enabling Port0 of ITM.

22. To validate your understanding of the code behavior, set breakpoints in different tasks before or after the print statements of queue access instructions.
23. Click on the red button ● on the ***SWV Console*** panel and then run the debugger.
24. As you step through the code, check the print messages on ***SWV Console***, the number of elements in the Queue on ***FreeRTOS Queues*** panel and how the program switched between tasks on ***FreeRTOS Task List*** panel.