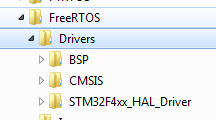
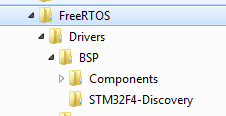
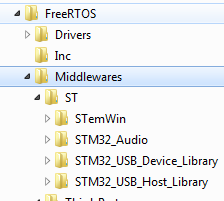
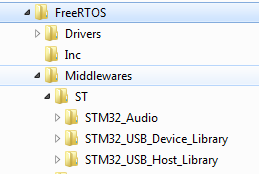
Step by Step

1. STM32CubeMX New Project.
2. Board Selector:
   1. Vendor:
      1. STMicroelectronics
   2. Type of Board:
      1. Discovery
   3. MCU Series:
      1. STM32F4
   4. Board List:
      1. STM32F4Discovery
3. OK
4. Pinout:
   1. Middlewares:
      1. FreeRTOS
   2. Peripherals:
      1. ADC1
      2. ADC2
      3. ADC3
      4. CAN1
      5. CAN2
      6. CRC
      7. I2C1 – I2C
      8. I2S3 – Full-Duplex Master
      9. RCC – High Speed Clock (HSE) Crystal/Ceramic  
         RCC – Low Speed Clock (LSE) Disable
      10. RNG
      11. SPI1 – Full-Duplex Master
      12. TIM1 – Internal Clock
      13. TIM2 – Internal Clock
      14. USB\_OTG\_FS – OTG/Dual\_Role\_Device
5. Configuration:
   1. IS23 – Multimedia – 49 kHz
6. Project:
   1. Settings
      1. Project Name:
         1. FreeRTOS
      2. Toolchain/IDE
         1. SW4STM32
      3. Generated Files – Generate peripheral initialization as a pair …
   2. OK
7. Project:
   1. Generate Code
8. Close STM32CubeMX
9. In Windows Explorer
   1. Delete Configuration from the folder name as the space will cause a failure  
      “FreeRTOS\SW4STM32\FreeRTOS Configuration”
10. Edit .cproject
    1. Remove Configuration from both occurrences of artifactName
       1. <configuration artifactExtension="elf" artifactName="FreeRTOS Configuration"
       2. <configuration artifactExtension="elf" artifactName="FreeRTOS"
11. Edit .project
    1. Remove Configuration from name
       1. <name>FreeRTOS Configuration</name>
       2. <name>FreeRTOS</name>
12. Copy STM32Cube\Repository\STM32Cube\_FW\_F4\_V1.5.0\Drivers\BSP to FreeRTOS\Drivers  
    
13. In FreeRTOS\Drivers\BSP, delete all content except for, Components and STM32F4-Discovery  
    
14. Copy STM32Cube\Repository\STM32Cube\_FW\_F4\_V1.5.0\Middlewares\ST to  
    FreeRTOS\Middlewares  
    
15. Delete STemWin  
    
16. The on disk directory tree should look like this  
    
17. Start System Workbench
    1. Import Existing Projects into Workspace
    2. Choose the workspace used by System Workbench
    3. Select FreeRTOS and click Finish
    4. Project Properties (ALT + Enter)
       1. Resource:
          1. Text File Encoding – UTF-8
       2. C/C++ Build
          1. Settings
             1. Tool Settings

Floating point hardware – fpv4-sp-d16

Floating point ABI – hard

* + - * 1. Build Steps (add this to generate bin and hex files)

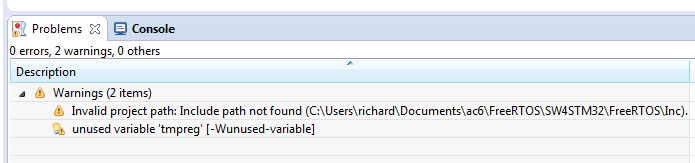
Post Build Steps

Command:  
arm-none-eabi-objcopy -O binary "${BuildArtifactFileBaseName}.elf" "${BuildArtifactFileBaseName}.bin"  
&  
arm-none-eabi-objcopy -O ihex "${BuildArtifactFileBaseName}.elf" "${BuildArtifactFileBaseName}.hex"

* + - * 1. Target:

Board: STM32F4Discovery

* + 1. Click on OK
  1. Select the Project and choose Build All, result is two warnings



* 1. Edit main.c, compare my file to your main.c, mine works
  2. Add the BSP folder to the Project
     1. Click on Drivers, then Import from File System, Navigate to BSP and select Components and STM32F4-Discovery and then click Finish
  3. Clean the Project and then Build All
  4. Click on Run, Debug Configurations, click on AC6 STM32 Debugging and click on New
  5. Click on the Debug button
  6. After the code is loaded into the STM32F4 Discovery Board, click on the Resume ICON.

At this point, you should have a working FreeRTOS.

Main.c

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* File Name : main.c

\* Date : 03/05/2015 21:49:06

\* Description : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* COPYRIGHT(c) 2015 STMicroelectronics

\*

\* Redistribution and use in source and binary forms, with or without modification,

\* are permitted provided that the following conditions are met:

\* 1. Redistributions of source code must retain the above copyright notice,

\* this list of conditions and the following disclaimer.

\* 2. Redistributions in binary form must reproduce the above copyright notice,

\* this list of conditions and the following disclaimer in the documentation

\* and/or other materials provided with the distribution.

\* 3. Neither the name of STMicroelectronics nor the names of its contributors

\* may be used to endorse or promote products derived from this software

\* without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

\* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE

\* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

\* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

\* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

\* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

\* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

\* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* Includes ------------------------------------------------------------------\*/

#include "../Drivers/BSP/STM32F4-Discovery/stm32f4\_discovery.h"

#include "stm32f4xx\_hal.h"

#include "cmsis\_os.h"

#include "adc.h"

#include "can.h"

#include "crc.h"

#include "i2c.h"

#include "i2s.h"

#include "rng.h"

#include "spi.h"

#include "usb\_otg.h"

#include "gpio.h"

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* LED Private Variables \*/

osThreadId LEDThread1Handle, LEDThread2Handle, LEDThread3Handle,

LEDThread4Handle;

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

void MX\_FREERTOS\_Init(void);

/\* USER CODE BEGIN PFP \*/

/\*LED Function Prototypes \*/

static void LED\_Thread1(void const \*argument);

static void LED\_Thread2(void const \*argument);

static void LED\_Thread3(void const \*argument);

static void LED\_Thread4(void const \*argument);

/\* USER CODE END PFP \*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

int main(void) {

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration----------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* Configure the system clock \*/

SystemClock\_Config();

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_ADC1\_Init();

MX\_ADC2\_Init();

MX\_ADC3\_Init();

MX\_CAN1\_Init();

MX\_CAN2\_Init();

MX\_CRC\_Init();

MX\_I2C1\_Init();

MX\_I2S3\_Init();

MX\_RNG\_Init();

MX\_SPI1\_Init();

MX\_USB\_OTG\_FS\_USB\_Init();

/\* USER CODE BEGIN 2 \*/

/\* Configure all four LEDs\*/

BSP\_LED\_Init(LED3);

BSP\_LED\_Init(LED4);

BSP\_LED\_Init(LED5);

BSP\_LED\_Init(LED6);

/\* Thread 1 definition and assign to LED \*/

osThreadDef(LED3, LED\_Thread1, osPriorityNormal, 0,

configMINIMAL\_STACK\_SIZE);

/\* Thread 2 definition and assign to LED \*/

osThreadDef(LED4, LED\_Thread2, osPriorityNormal, 0,

configMINIMAL\_STACK\_SIZE);

/\* Thread 3 definition and assign to LED \*/

osThreadDef(LED5, LED\_Thread3, osPriorityNormal, 0,

configMINIMAL\_STACK\_SIZE);

/\* Thread 4 definition and assign to LED \*/

osThreadDef(LED6, LED\_Thread4, osPriorityNormal, 0,

configMINIMAL\_STACK\_SIZE);

/\* Start Thread 1 \*/

LEDThread1Handle = osThreadCreate(osThread(LED3), NULL);

/\* Start Thread 2 \*/

LEDThread2Handle = osThreadCreate(osThread(LED4), NULL);

/\* Start Thread 3 \*/

LEDThread3Handle = osThreadCreate(osThread(LED5), NULL);

/\* Start Thread 4 \*/

LEDThread4Handle = osThreadCreate(osThread(LED6), NULL);

/\* USER CODE END 2 \*/

/\* Call init function for freertos objects (in freertos.c) \*/

MX\_FREERTOS\_Init();

/\* Start scheduler \*/

osKernelStart();

/\* We should never get here as control is now taken by the scheduler \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1) {

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\* Toggle LED3 \*/

static void LED\_Thread1(void const \*argument) {

uint32\_t count = 0;

(void) argument;

for (;;) {

count = osKernelSysTick() + 5000;

/\* Toggle LED3 every 200 ms for 5 s \*/

while (count >= osKernelSysTick()) {

BSP\_LED\_Toggle(LED3);

osDelay(200);

}

/\* Turn off LED3 \*/

BSP\_LED\_Off(LED3);

/\* Suspend Thread 1 \*/

osThreadSuspend(NULL);

count = osKernelSysTick() + 5000;

/\* Toggle LED3 every 400 ms for 5 s \*/

while (count >= osKernelSysTick()) {

BSP\_LED\_Toggle(LED3);

osDelay(400);

}

/\* Resume Thread 2 \*/

osThreadResume(LEDThread2Handle);

}

}

/\* Toggle LED4 \*/

static void LED\_Thread2(void const \*argument) {

uint32\_t count;

(void) argument;

for (;;) {

count = osKernelSysTick() + 10000;

/\* Toggle LED4 every 500 ms for 10 s \*/

while (count >= osKernelSysTick()) {

BSP\_LED\_Toggle(LED4);

osDelay(500);

}

/\* Turn off LED4 \*/

BSP\_LED\_Off(LED4);

/\* Resume Thread 1 \*/

osThreadResume(LEDThread1Handle);

/\* Suspend Thread 2 \*/

osThreadSuspend(NULL);

}

}

/\* Toggle LED5 \*/

static void LED\_Thread3(void const \*argument) {

uint32\_t count;

(void) argument;

for (;;) {

count = osKernelSysTick() + 10000;

/\* Toggle LED5 every 500 ms for 10 s \*/

while (count >= osKernelSysTick()) {

BSP\_LED\_Toggle(LED5);

osDelay(500);

}

/\* Turn off LED5 \*/

BSP\_LED\_Off(LED5);

/\* Resume Thread 4 \*/

osThreadResume(LEDThread4Handle);

/\* Suspend Thread 3 \*/

osThreadSuspend(NULL);

}

}

/\* Toggle LED6 \*/

static void LED\_Thread4(void const \*argument) {

uint32\_t count;

(void) argument;

for (;;) {

count = osKernelSysTick() + 10000;

/\* Toggle LED6 every 500 ms for 10 s \*/

while (count >= osKernelSysTick()) {

BSP\_LED\_Toggle(LED6);

osDelay(500);

}

/\* Turn off LED6 \*/

BSP\_LED\_Off(LED6);

/\* Resume Thread 3 \*/

osThreadResume(LEDThread3Handle);

/\* Suspend Thread 4 \*/

osThreadSuspend(NULL);

}

}

/\*\* System Clock Configuration

\*/

void SystemClock\_Config(void) {

RCC\_OscInitTypeDef RCC\_OscInitStruct;

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_PeriphCLKInitTypeDef PeriphClkInitStruct;

\_\_PWR\_CLK\_ENABLE()

;

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 336;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 7;

HAL\_RCC\_OscConfig(&RCC\_OscInitStruct);

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_SYSCLK | RCC\_CLOCKTYPE\_PCLK1

| RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5);

PeriphClkInitStruct.PeriphClockSelection = RCC\_PERIPHCLK\_I2S;

PeriphClkInitStruct.PLLI2S.PLLI2SN = 192;

PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;

HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t\* file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif

/\*\*

\* @}

\*/

/\*\*

\* @}

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/