Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 6-2: Ethernet – LwIP HTTP Server Socket RTOS

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Hands-On LwIP HTTP Server Socket RTOS

Objectives

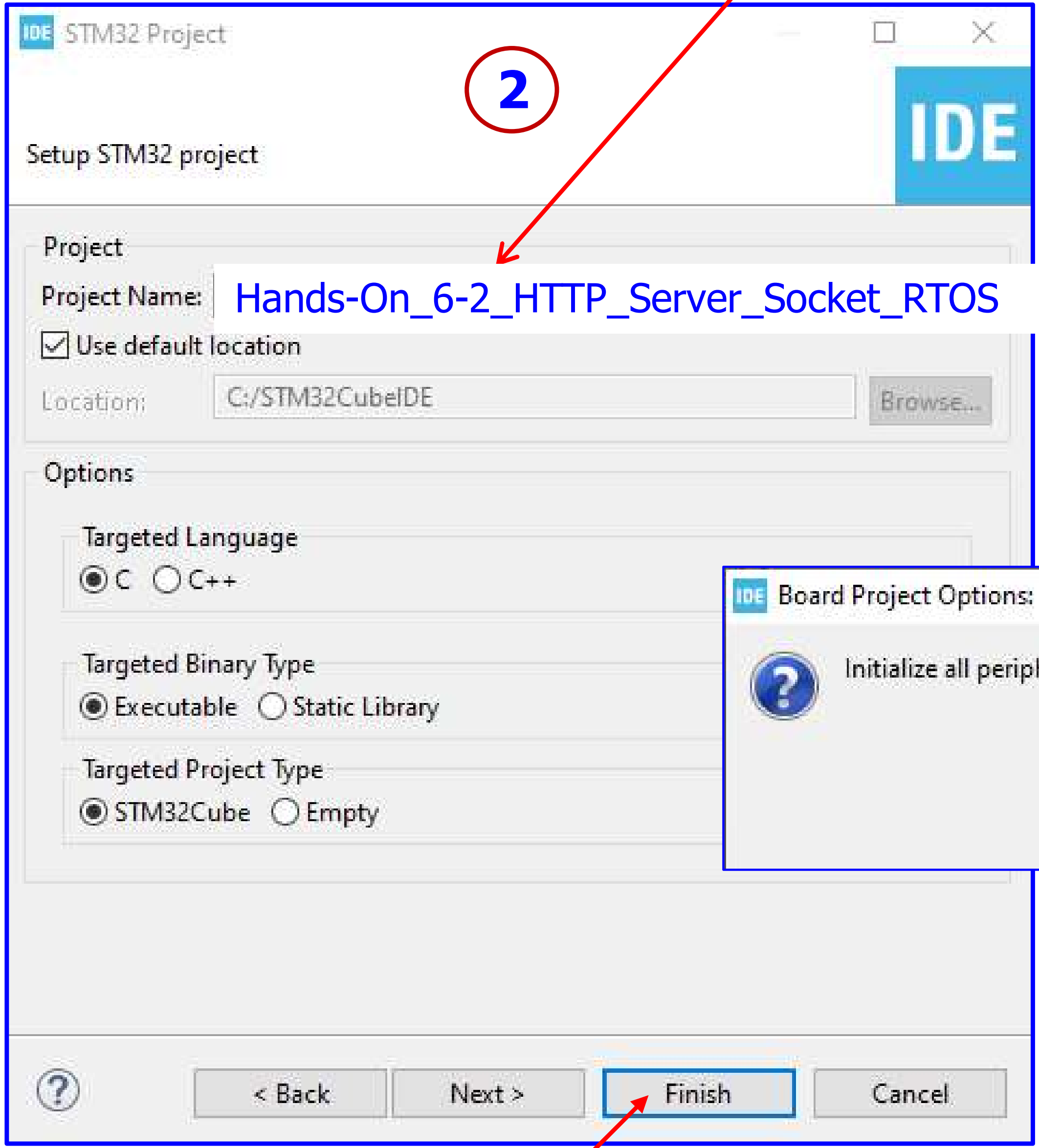
The aims of this hands-on session are to

- develop a STM32 (STM32CubeIDE) project
- Implement a web (HTTP) server application based on Socket RTOS using STM32F767 microcontroller
- configure and program the Ethernet peripheral to make the microcontroller operating as a HTTP server and connecting web clients for loading of HTML pages
- develop program using the htmlgen.exe software to generate the web pages
- test the developed application by opening a web client on a remote PC to interact with the web server
- build up the knowledge of Ethernet application development

Note that, this web server contains two HTML pages. The first one gives general information about STM32F7xx microcontrollers and the LwIP stack. The second one lists the running tasks and their status. This page is automatically updated every second.

Hands-On LwIP HTTP Server Socket RTOS

Create the STM32 Project: **Hands-On_6-2_HTTP_Server_Socket_RTOS**



STM32 Project

Setup STM32 project

Project

Project Name: Hands-On_6-2_HTTP_Server_Socket_RTOS

[✓] Use default location

Location: C:/STM32CubeIDE

Options

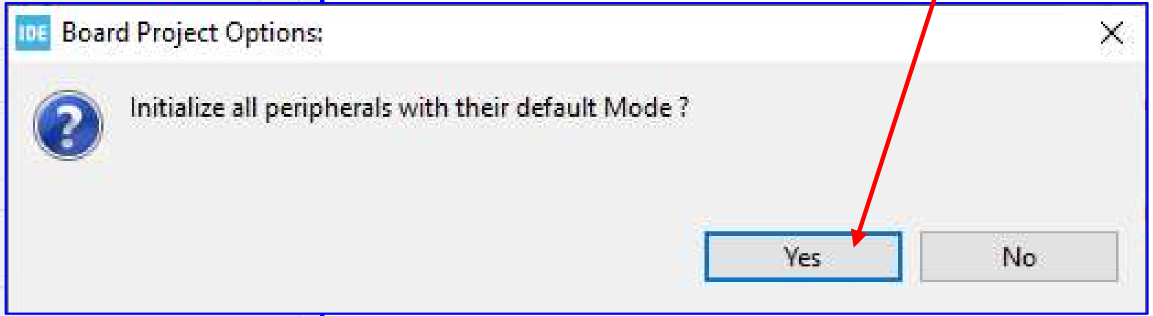
Targeted Language: (●) C ( ) C++

Targeted Binary Type: (●) Executable ( ) Static Library

Targeted Project Type: (●) STM32Cube ( ) Empty

- Run STM32CubeIDE 1
- Select workspace: C:\STM32_CS397
- File -> Close All Editors
- Start a New STM32 Project
- Select the Nucleo-F767ZI Board

4



Board Project Options:

Initialize all peripherals with their default Mode ?

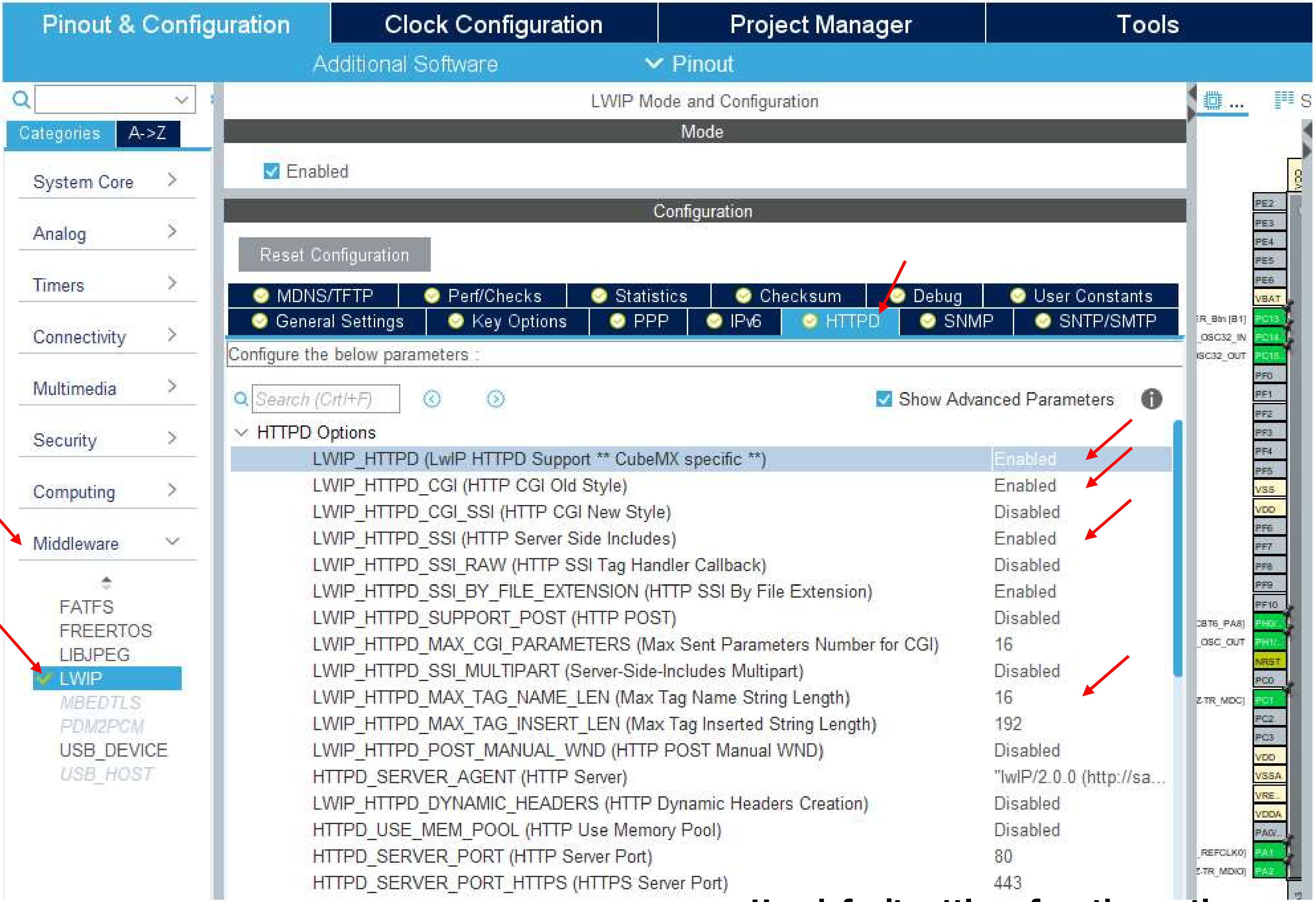
Follow all the setup steps in 5

Hands-on_4-1_TCP_Echo_Client

(Pages 4-18)

Hands-On LwIP HTTP Server Socket RTOS

Configure LwIP – HTTPD:



[✓] Enabled

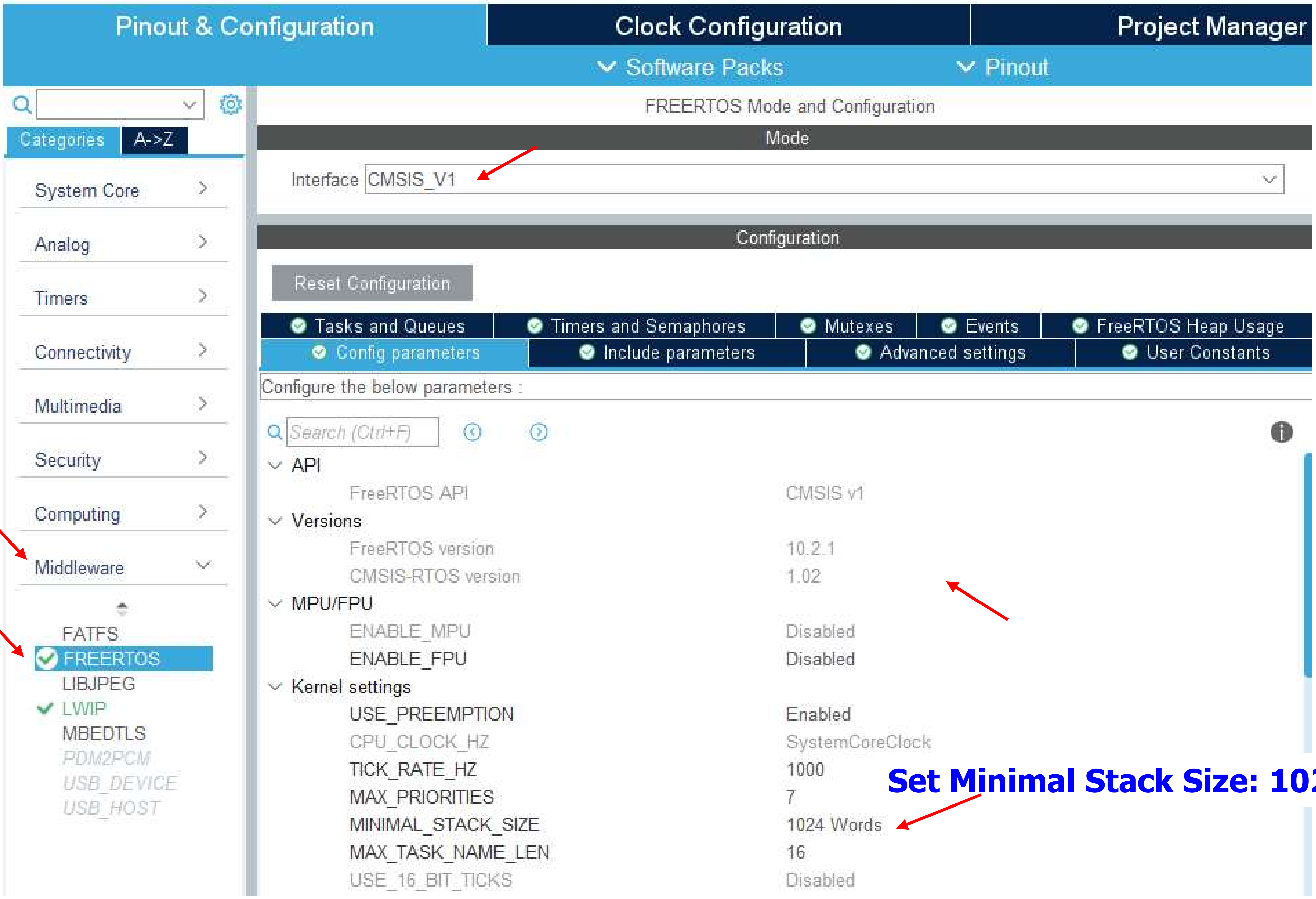
[✓] Show Advanced Parameters

LWIP_HTTPD (LwIP HTTPD Support ** CubeMX specific **)	Enabled
LWIP_HTTPD_CGI (HTTP CGI Old Style)	Enabled
LWIP_HTTPD_CGI_SSI (HTTP CGI New Style)	Disabled
LWIP_HTTPD_SSI (HTTP Server Side Includes)	Enabled
LWIP_HTTPD_SSI_RAW (HTTP SSI Tag Handler Callback)	Disabled
LWIP_HTTPD_SSI_BY_FILE_EXTENSION (HTTP SSI By File Extension)	Enabled
LWIP_HTTPD_SUPPORT_POST (HTTP POST)	Disabled
LWIP_HTTPD_MAX_CGI_PARAMETERS (Max Sent Parameters Number for CGI)	16
LWIP_HTTPD_SSI_MULTIPART (Server-Side-Includes Multipart)	Disabled
LWIP_HTTPD_MAX_TAG_NAME_LEN (Max Tag Name String Length)	16
LWIP_HTTPD_MAX_TAG_INSERT_LEN (Max Tag Inserted String Length)	192
LWIP_HTTPD_POST_MANUAL_WND (HTTP POST Manual WND)	Disabled
HTTPD_SERVER_AGENT (HTTP Server)	"lwIP/2.0.0 (http://sa...
LWIP_HTTPD_DYNAMIC_HEADERS (HTTP Dynamic Headers Creation)	Disabled
HTTPD_USE_MEM_POOL (HTTP Use Memory Pool)	Disabled
HTTPD_SERVER_PORT (HTTP Server Port)	80
HTTPD_SERVER_PORT_HTTPS (HTTPS Server Port)	443

Use default settings for other options

Hands-On LwIP HTTP Server Socket RTOS

Enable **FREERTOS** by selecting the interface “**CMSIS_V1**”.

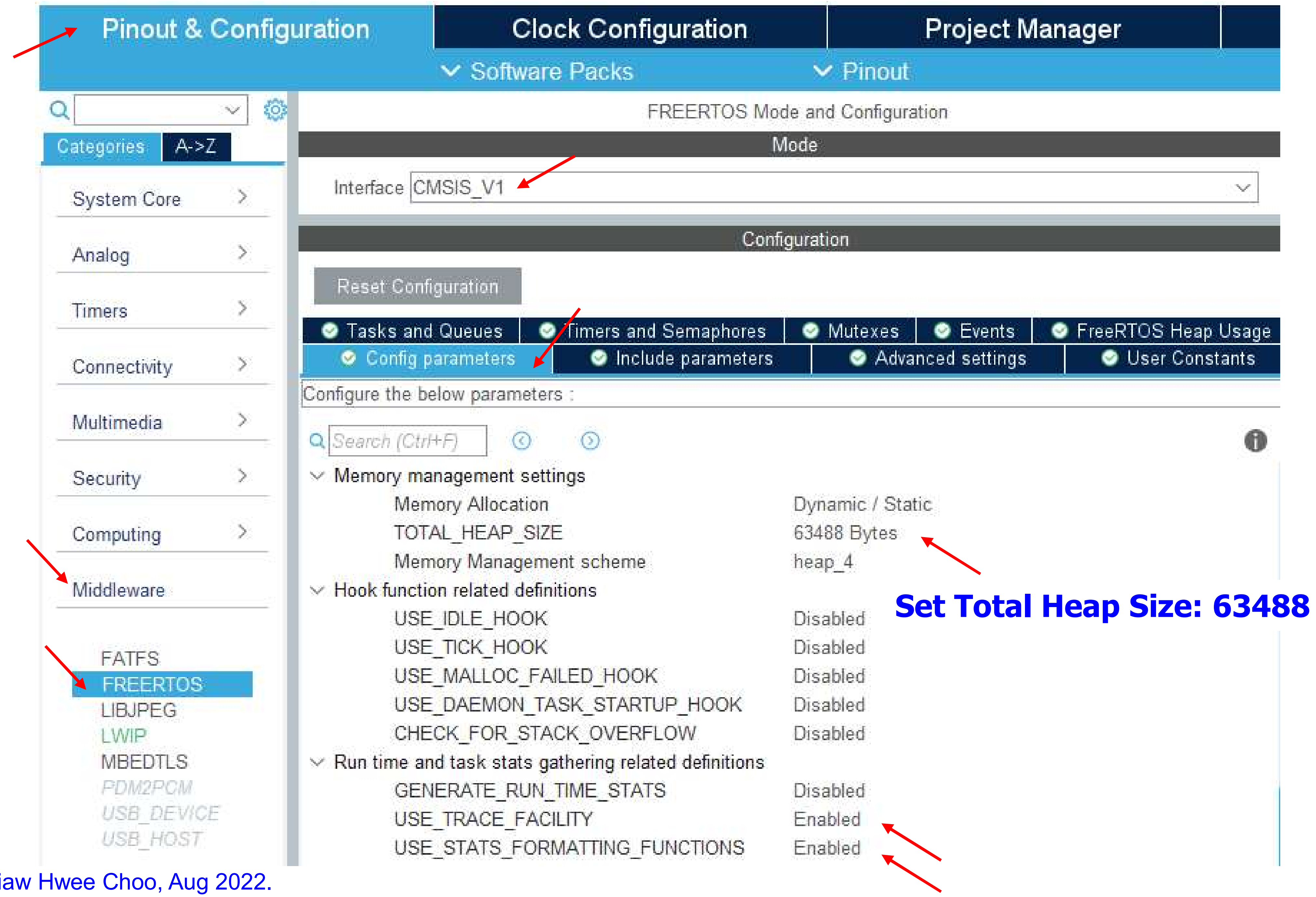


Interface: CMSIS_V1

API	
FreeRTOS API	CMSIS v1
Versions	
FreeRTOS version	10.2.1
CMSIS-RTOS version	1.02
MPU/FPU	
ENABLE_MPU	Disabled
ENABLE_FPU	Disabled
Kernel settings	
USE_PREEMPTION	Enabled
CPU_CLOCK_HZ	SystemCoreClock
TICK_RATE_HZ	1000
MAX_PRIORITIES	7
MINIMAL_STACK_SIZE	1024 Words
MAX_TASK_NAME_LEN	16
USE_16_BIT_TICKS	Disabled

Set Minimal Stack Size: 1024

Increase **TOTAL_HEAP_SIZE**, enable **USE_TRACE_FACILTY** and **USE_STATS_FORMATTING_FUNCTIONS**



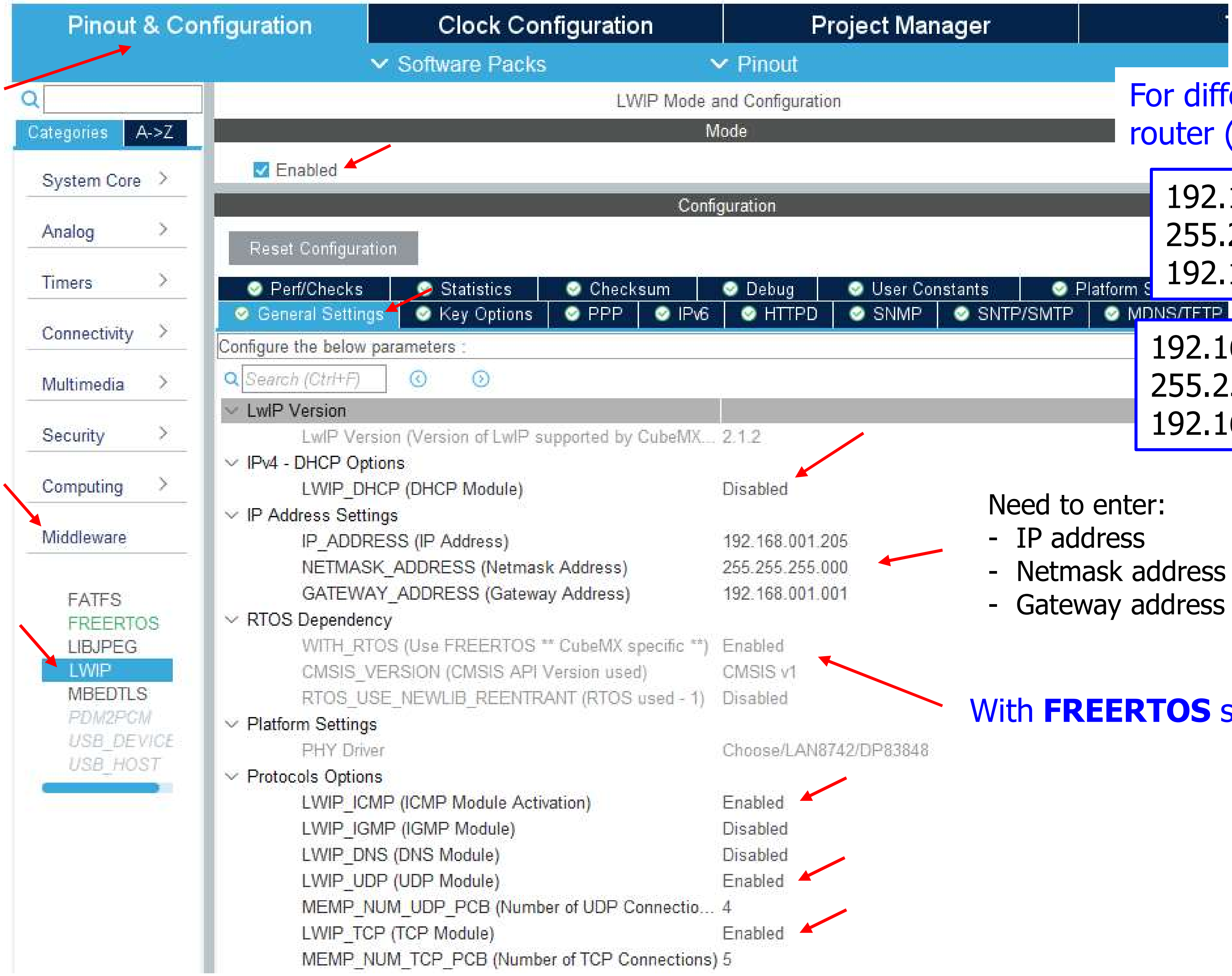
Interface CMSIS_V1

Memory management settings	
Memory Allocation	Dynamic / Static
TOTAL_HEAP_SIZE	63488 Bytes
Memory Management scheme	heap_4
Hook function related definitions	
USE_IDLE_HOOK	Disabled
USE_TICK_HOOK	Disabled
USE_MALLOC_FAILED_HOOK	Disabled
USE_DAEMON_TASK_STARTUP_HOOK	Disabled
CHECK_FOR_STACK_OVERFLOW	Disabled
Run time and task stats gathering related definitions	
GENERATE_RUN_TIME_STATS	Disabled
USE_TRACE_FACILITY	Enabled
USE_STATS_FORMATTING_FUNCTIONS	Enabled

Set Total Heap Size: 63488

LwIP Settings

Hands-On LwIP HTTP Server Socket RTOS



☑ Enabled

Configure the below parameters :

```
LwIP Version
    LwIP Version (Version of LwIP supported by CubeMX...  2.1.2
IPv4 - DHCP Options
    LWIP_DHCP (DHCP Module)                            Disabled
IP Address Settings
    IP_ADDRESS (IP Address)                            192.168.001.205
    NETMASK_ADDRESS (Netmask Address)                  255.255.255.000
    GATEWAY_ADDRESS (Gateway Address)                  192.168.001.001
RTOS Dependency
    WITH_RTOS (Use FREERTOS ** CubeMX specific **)     Enabled
    CMSIS_VERSION (CMSIS API Version used)             CMSIS v1
    RTOS_USE_NEWLIB_REENTRANT (RTOS used - 1)          Disabled
Platform Settings
    PHY Driver                                         Choose/LAN8742/DP83848
Protocols Options
    LWIP_ICMP (ICMP Module Activation)                 Enabled
    LWIP_IGMP (IGMP Module)                            Disabled
    LWIP_DNS (DNS Module)                              Disabled
    LWIP_UDP (UDP Module)                              Enabled
    MEMP_NUM_UDP_PCB (Number of UDP Connectio...       4
    LWIP_TCP (TCP Module)                              Enabled
    MEMP_NUM_TCP_PCB (Number of TCP Connections)       5
```

For different router (gateway):

192.168.1.205
255.255.255.0
192.168.1.1

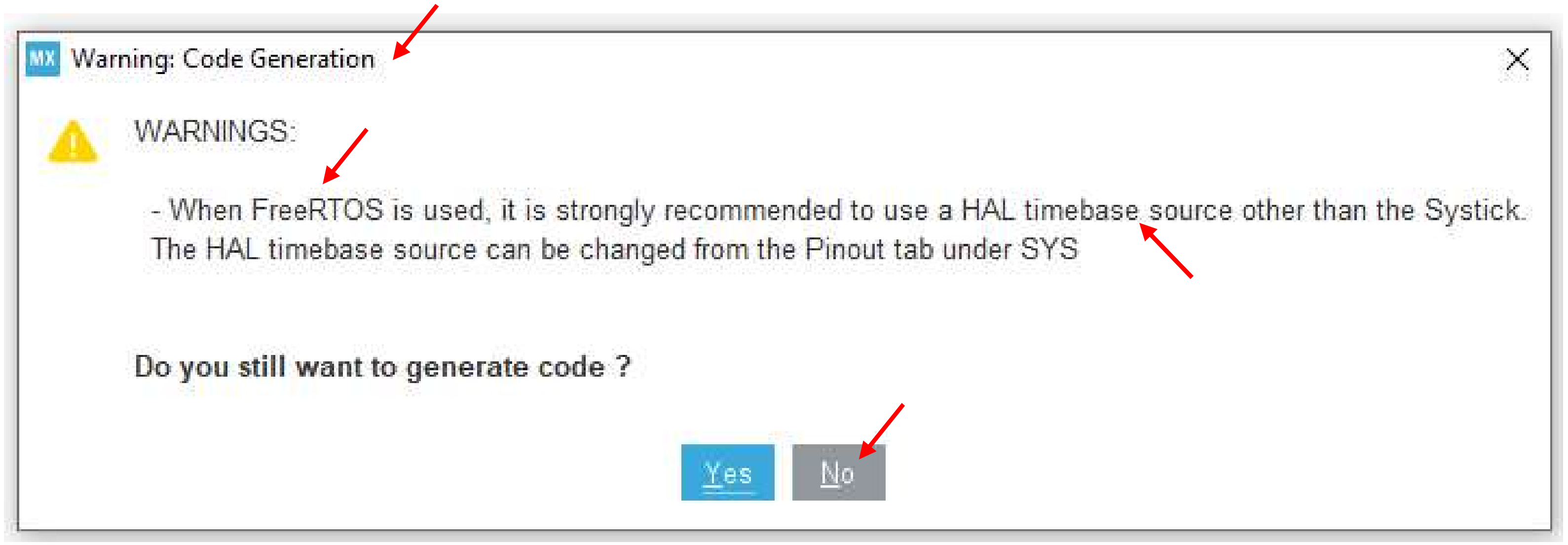
192.168.50.205
255.255.255.0
192.168.50.1

Need to enter:
- IP address
- Netmask address
- Gateway address

With **FREERTOS** selected

Hands-On LwIP HTTP Server Socket RTOS

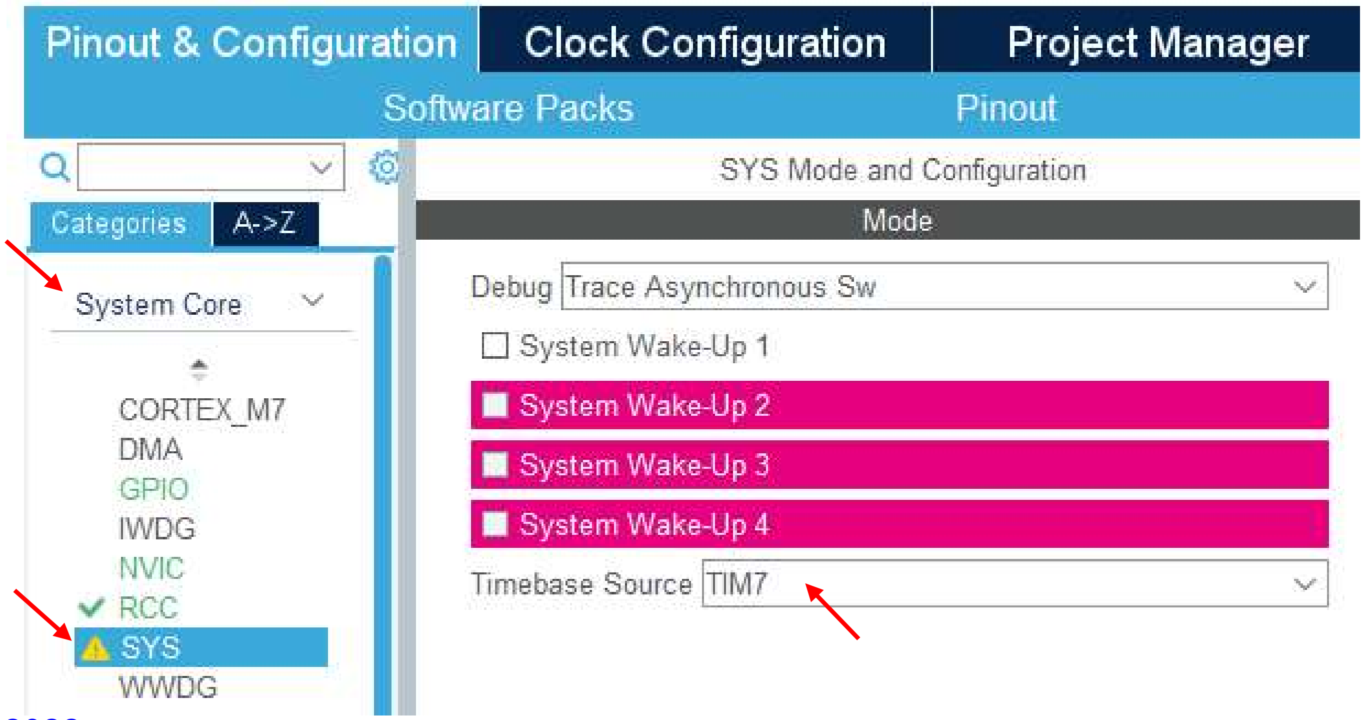
With **FREERTOS** selected, the **Timebase Source** is changed to **TIM7** manually.


Warning: Code Generation

WARNINGS:

- When FreeRTOS is used, it is strongly recommended to use a HAL timebase source other than the Systick. The HAL timebase source can be changed from the Pinout tab under SYS

Do you still want to generate code ?

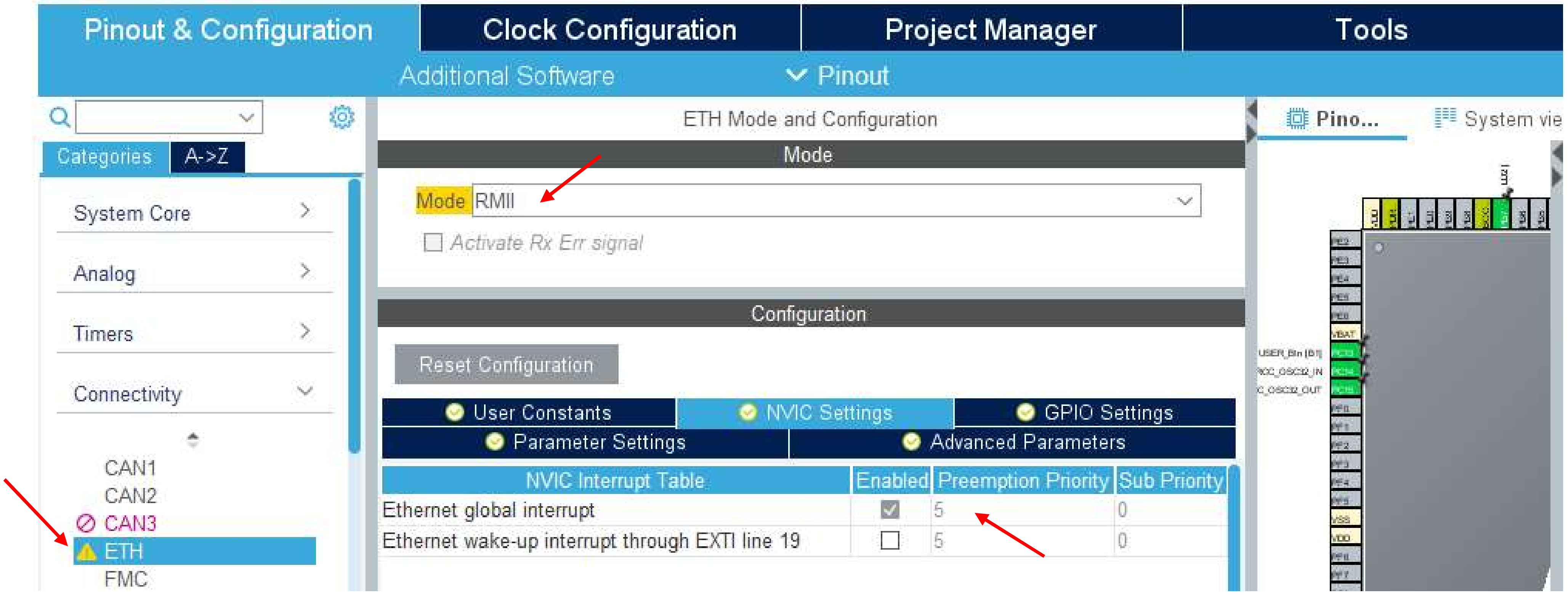


Debug: Trace Asynchronous Sw

☐ System Wake-Up 1
☐ System Wake-Up 2
☐ System Wake-Up 3
☐ System Wake-Up 4

Timebase Source: TIM7

With **FREERTOS** selected, **Ethernet Global Interrupt** is enabled and assigned with Preemption Priority.



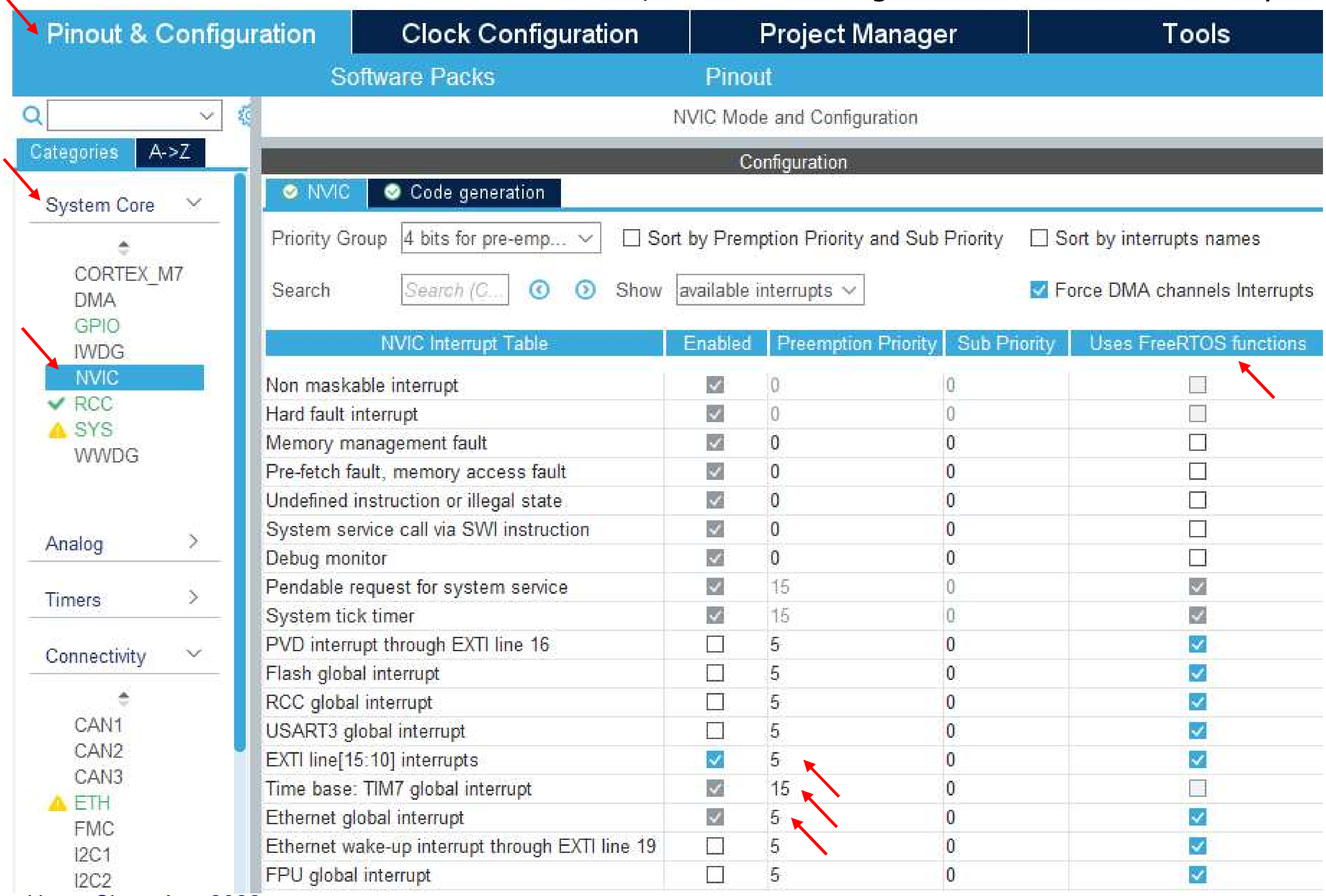
Mode: RMII

☐ Activate Rx Err signal

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Ethernet global interrupt	✓	5	0
Ethernet wake-up interrupt through EXTI line 19	☐	5	0

Hands-On LwIP HTTP Server Socket RTOS

With **FREERTOS** and **Time Base** selections, the NVIC settings are modified automatically



Priority Group: 4 bits for pre-emp...

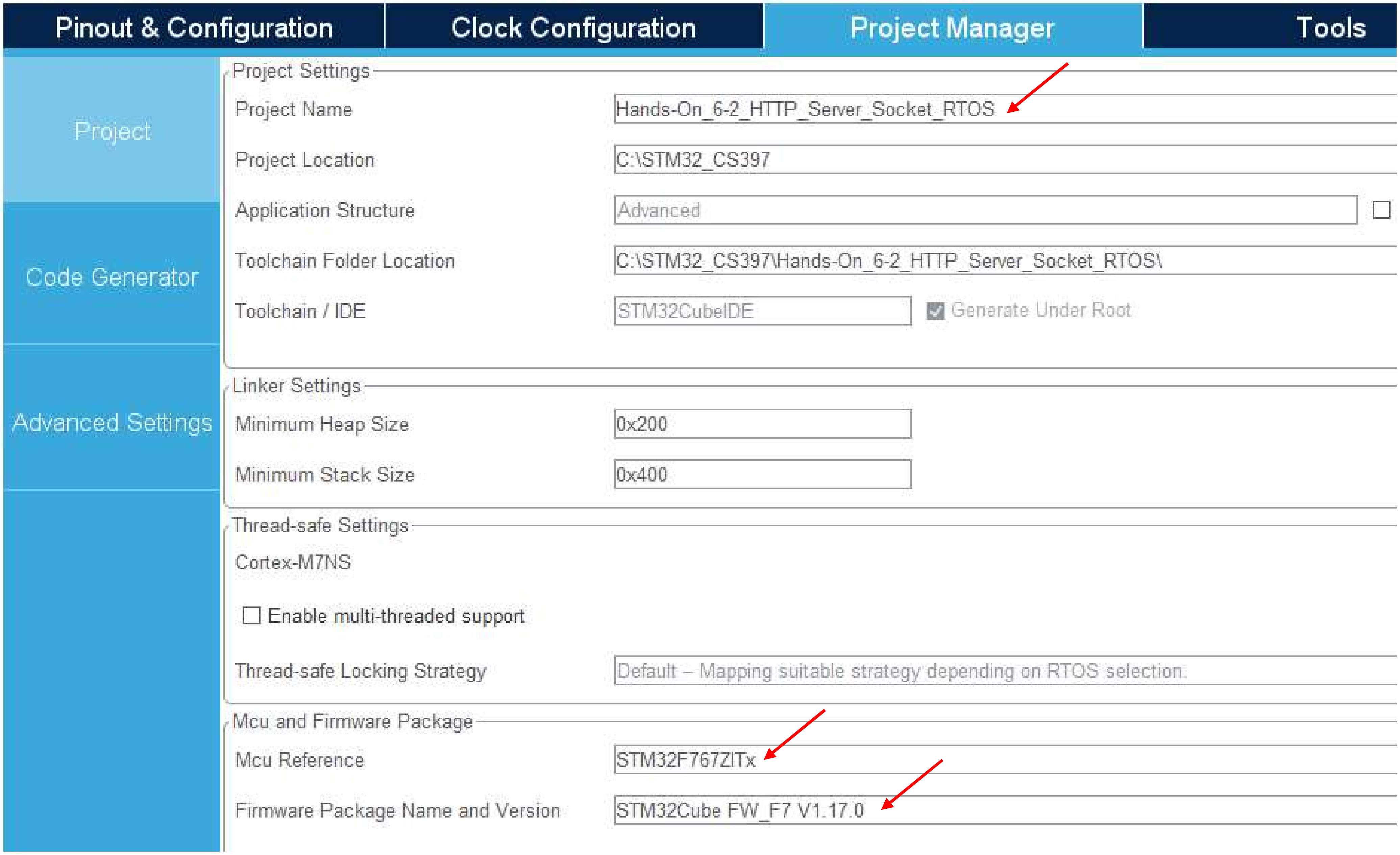
☐ Sort by Premption Priority and Sub Priority ☐ Sort by interrupts names

Search: Search (C... Show: available interrupts ✓ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority	Uses FreeRTOS functions
Non maskable interrupt	✓	0	0	☐
Hard fault interrupt	✓	0	0	☐
Memory management fault	✓	0	0	☐
Pre-fetch fault, memory access fault	✓	0	0	☐
Undefined instruction or illegal state	✓	0	0	☐
System service call via SWI instruction	✓	0	0	☐
Debug monitor	✓	0	0	☐
Pendable request for system service	✓	15	0	✓
System tick timer	✓	15	0	✓
PVD interrupt through EXTI line 16	☐	5	0	✓
Flash global interrupt	☐	5	0	✓
RCC global interrupt	☐	5	0	✓
USART3 global interrupt	☐	5	0	✓
EXTI line[15:10] interrupts	✓	5	0	✓
Time base: TIM7 global interrupt	✓	15	0	☐
Ethernet global interrupt	✓	5	0	✓
Ethernet wake-up interrupt through EXTI line 19	☐	5	0	✓
FPU global interrupt	☐	5	0	✓

Hands-On LwIP HTTP Server Socket RTOS

Information: Firmware Package Name and Version



Project Settings

Project Name	Hands-On_6-2_HTTP_Server_Socket_RTOS
Project Location	C:\STM32_CS397
Application Structure	Advanced
Toolchain Folder Location	C:\STM32_CS397\Hands-On_6-2_HTTP_Server_Socket_RTOS\
Toolchain / IDE	STM32CubeIDE ☑ Generate Under Root

Linker Settings

Minimum Heap Size	0x200
Minimum Stack Size	0x400

Thread-safe Settings

Cortex-M7NS

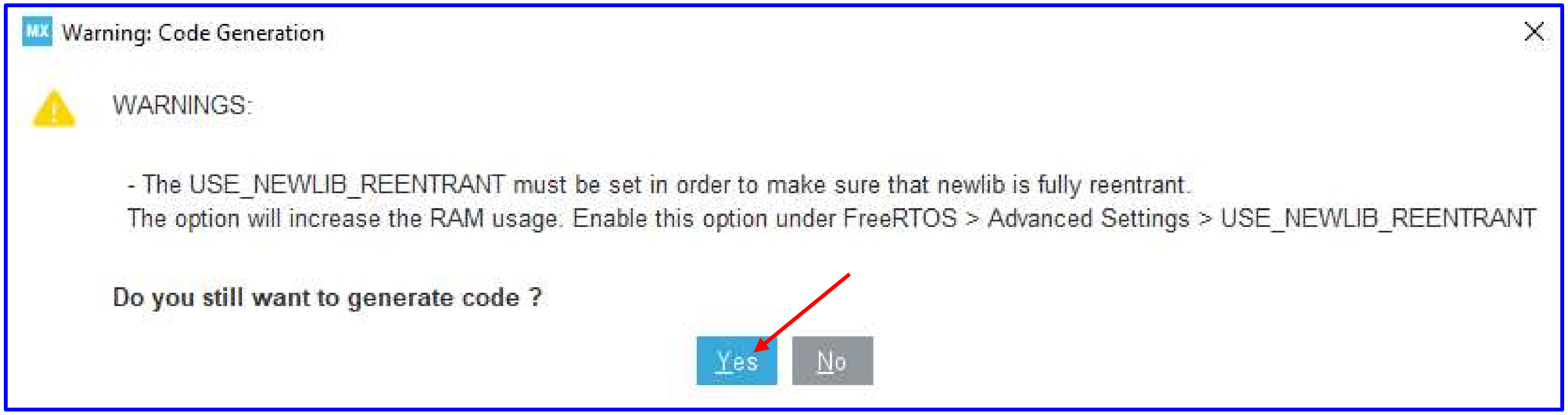
☐ Enable multi-threaded support

Thread-safe Locking Strategy	Default – Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

Mcu Reference	STM32F767ZITx
Firmware Package Name and Version	STM32Cube FW_F7 V1.17.0

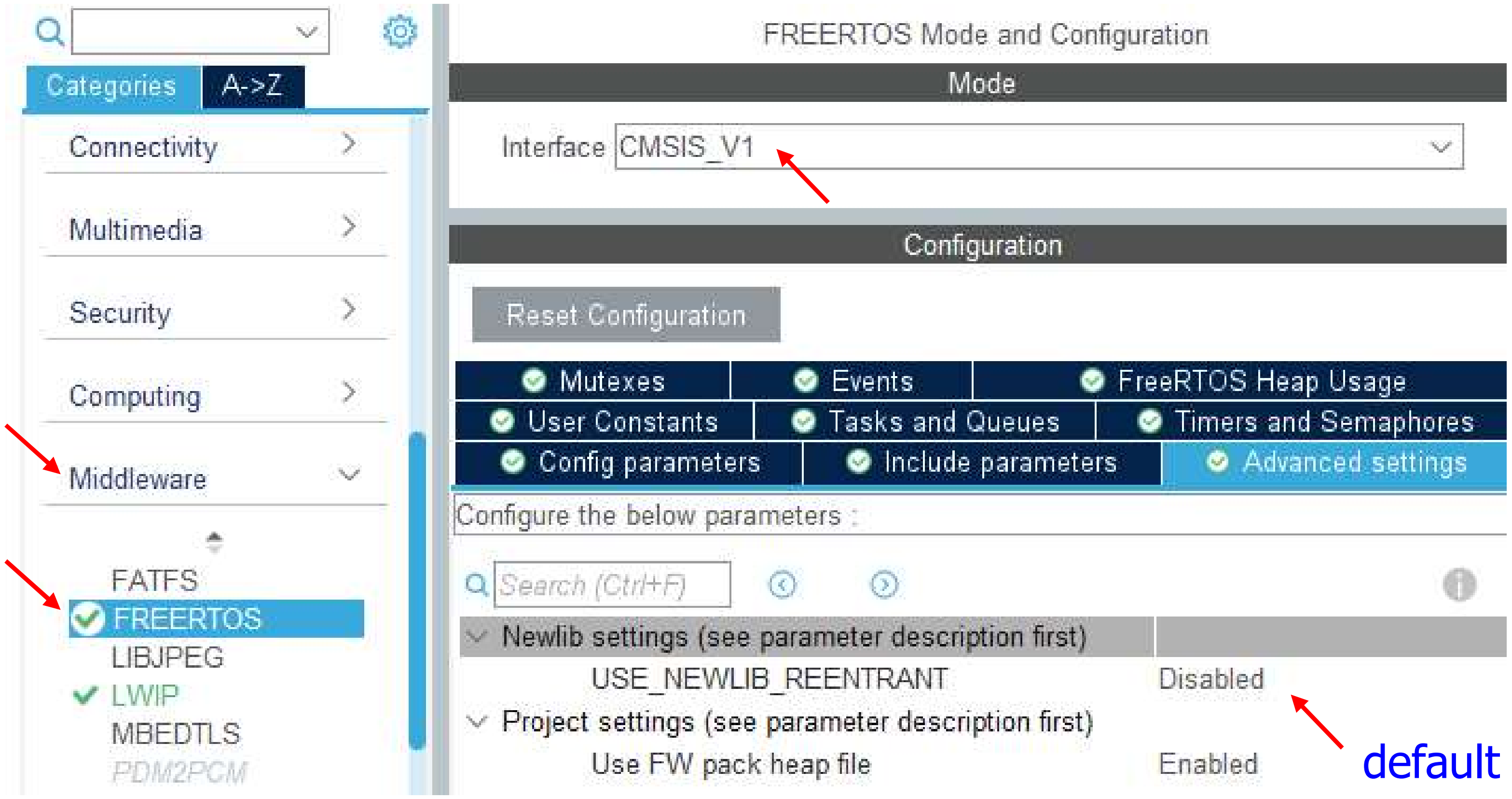
Code Generation: Do not enable USE_NEWLIB_REENTRANT



WARNINGS:

- The USE_NEWLIB_REENTRANT must be set in order to make sure that newlib is fully reentrant.
The option will increase the RAM usage. Enable this option under FreeRTOS > Advanced Settings > USE_NEWLIB_REENTRANT

Do you still want to generate code ?



Interface CMSIS_V1

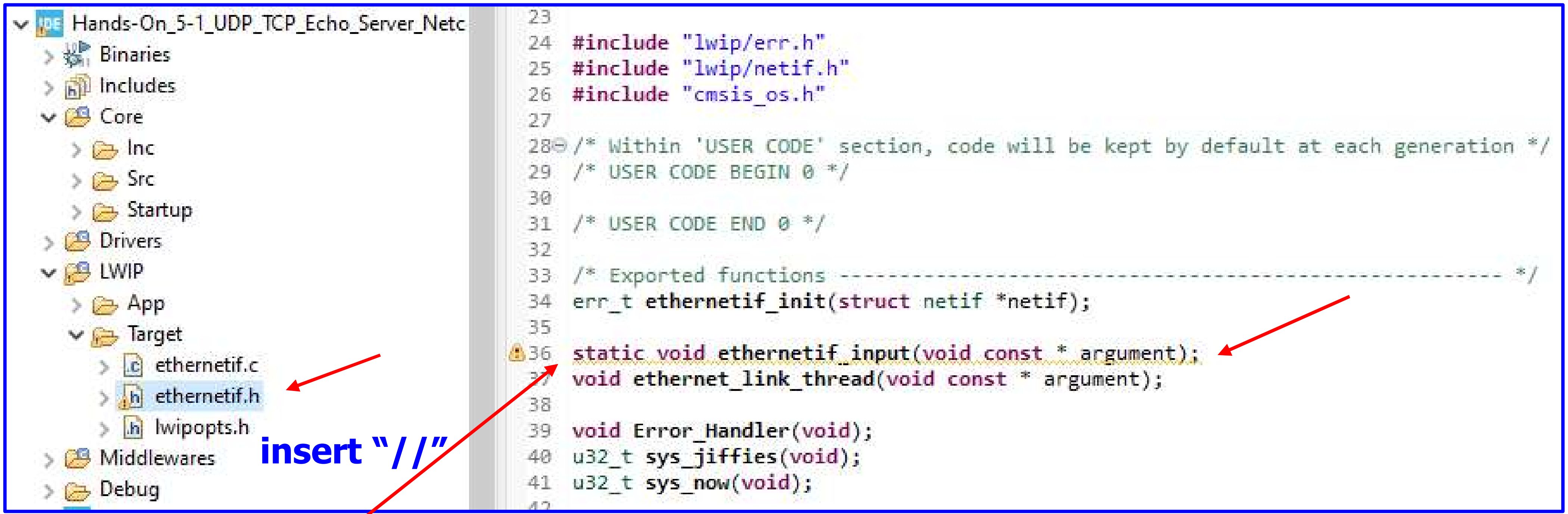
Newlib settings (see parameter description first)	
USE_NEWLIB_REENTRANT	Disabled
Project settings (see parameter description first)	
Use FW pack heap file	Enabled

default setting

Build warning:

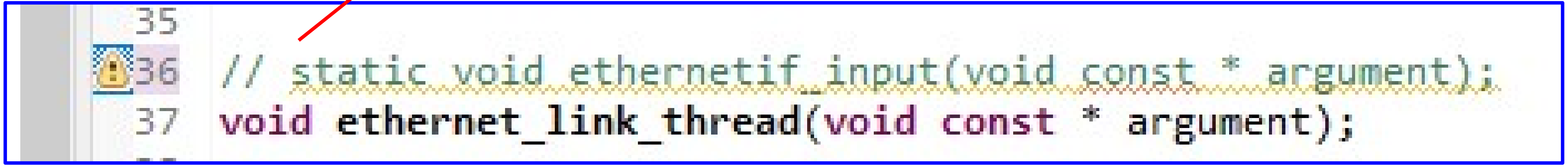
Hands-On LwIP HTTP Server Socket RTOS

```
../LWIP/Target/ethernetif.h:36:13: warning: 'ethernetif_input' declared
'static' but never defined [-Wunused-function]
   36 | static void ethernetif_input(void const * argument);
```

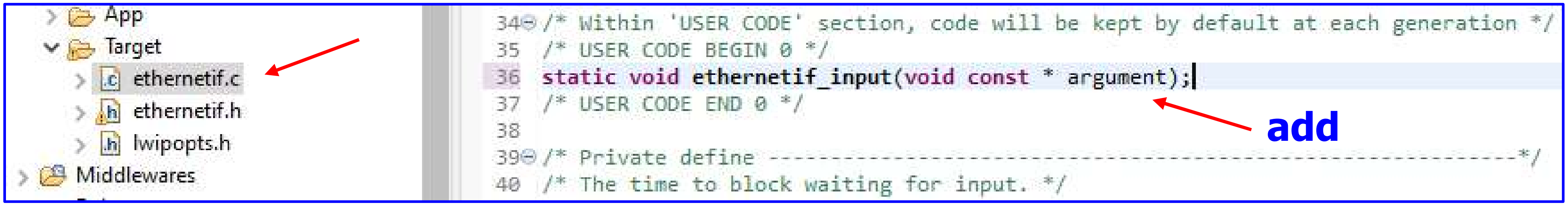


```
#include "lwip/err.h"
#include "lwip/netif.h"
#include "cmsis_os.h"

/* Within 'USER CODE' section, code will be kept by default at each generation */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/* Exported functions ------------------------------------------------------- */
err_t ethernetif_init(struct netif *netif);

static void ethernetif_input(void const * argument);
void ethernet_link_thread(void const * argument);

void Error_Handler(void);
u32_t sys_jiffies(void);
u32_t sys_now(void);
```

insert "//"



```
// static void ethernetif_input(void const * argument);
void ethernet_link_thread(void const * argument);
```

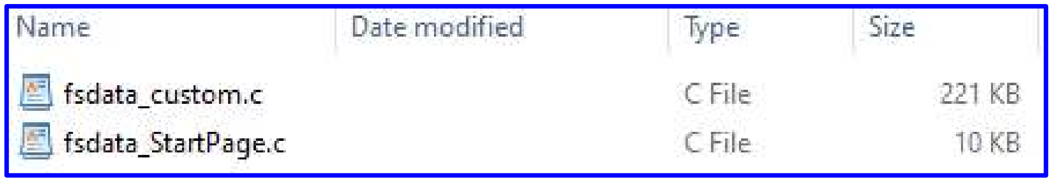


```
/* Within 'USER CODE' section, code will be kept by default at each generation */
/* USER CODE BEGIN 0 */
static void ethernetif_input(void const * argument);
/* USER CODE END 0 */

/* Private define ------------------------------------------------------------*/
/* The time to block waiting for input. */
```

add

Hands-On LwIP HTTP Server Socket RTOS

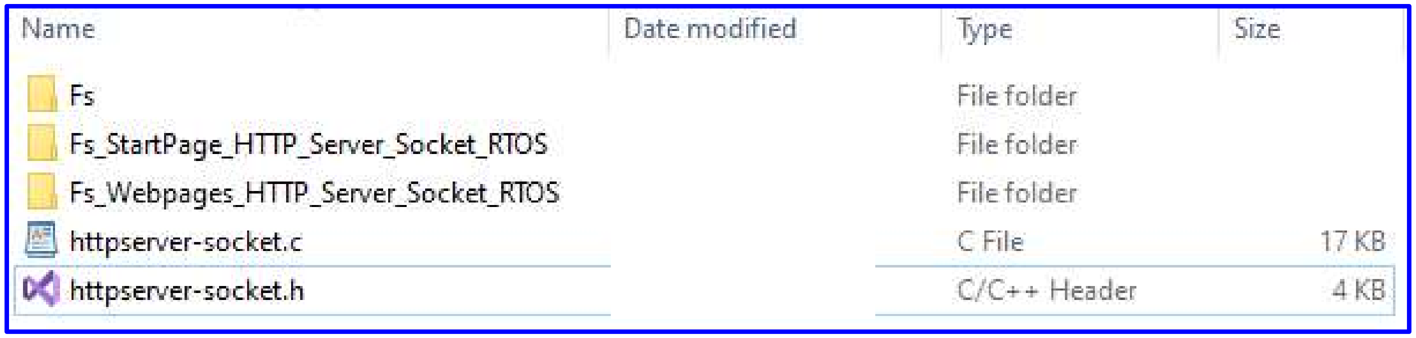
Generate the **`fsdata_custom.c`** and **`fsdata_StartPage.c`**



Name	Date modified	Type	Size
fsdata_custom.c		C File	221 KB
fsdata_StartPage.c		C File	10 KB

4 Copy generated files to folder "Fs"

1 **Unzip 13_CS397_Hands-On_6-2_LwIP_HTTP_Server_Socket_RTOS.zip**

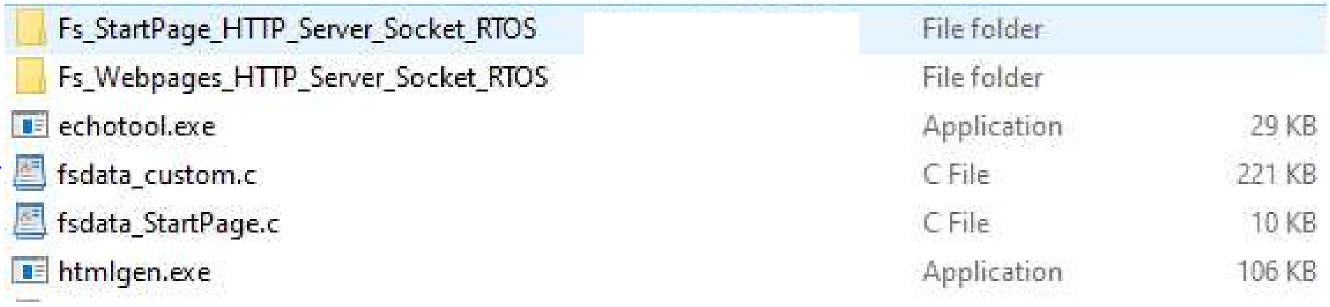


Name	Date modified	Type	Size
Fs		File folder	
Fs_StartPage_HTTP_Server_Socket_RTOS		File folder	
Fs_Webpages_HTTP_Server_Socket_RTOS		File folder	
httpserver-socket.c		C File	17 KB
httpserver-socket.h		C/C++ Header	4 KB

2 Copy these folders to

`C:\CS397>`

5 Copy above folders and files to STM32 project

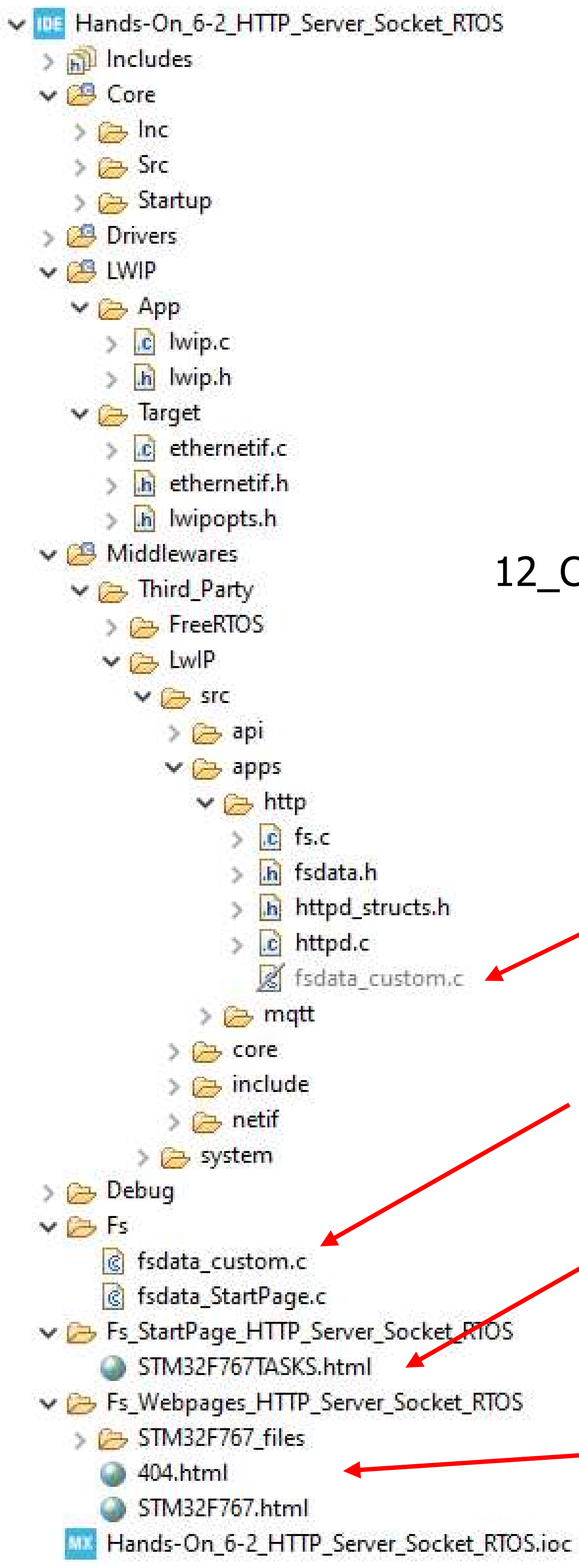


Fs_StartPage_HTTP_Server_Socket_RTOS	File folder	
Fs_Webpages_HTTP_Server_Socket_RTOS	File folder	
echotool.exe	Application	29 KB
fsdata_custom.c	C File	221 KB
fsdata_StartPage.c	C File	10 KB
htmlgen.exe	Application	106 KB

3 Run

```
C:\CS397>htmlgen Fs_StartPage_HTTP_Server_Socket_RTOS  -f:fsdata_StartPage.c
C:\CS397>htmlgen Fs_Webpages_HTTP_Server_Socket_RTOS   -f:fsdata_custom.c
```

Hands-On LwIP HTTP Server Socket RTOS



```
Hands-On_6-2_HTTP_Server_Socket_RTOS
  Includes
  Core
    Inc
    Src
    Startup
  Drivers
  LWIP
    App
      lwip.c
      lwip.h
    Target
      ethernetif.c
      ethernetif.h
      lwipopts.h
  Middlewares
    Third_Party
      FreeRTOS
      LwIP
        src
          api
          apps
            http
              fs.c
              fsdata.h
              httpd_structs.h
              httpd.c
              fsdata_custom.c
            mqtt
          core
          include
          netif
        system
  Debug
  Fs
    fsdata_custom.c
    fsdata_StartPage.c
  Fs_StartPage_HTTP_Server_Socket_RTOS
    STM32F767TASKS.html
  Fs_Webpages_HTTP_Server_Socket_RTOS
    STM32F767_files
    404.html
    STM32F767.html
  Hands-On_6-2_HTTP_Server_Socket_RTOS.ioc
```

12_CS397_Hands-On_5-2_LwIP_HTTP_Server_Raw_25Jul2022.pptx

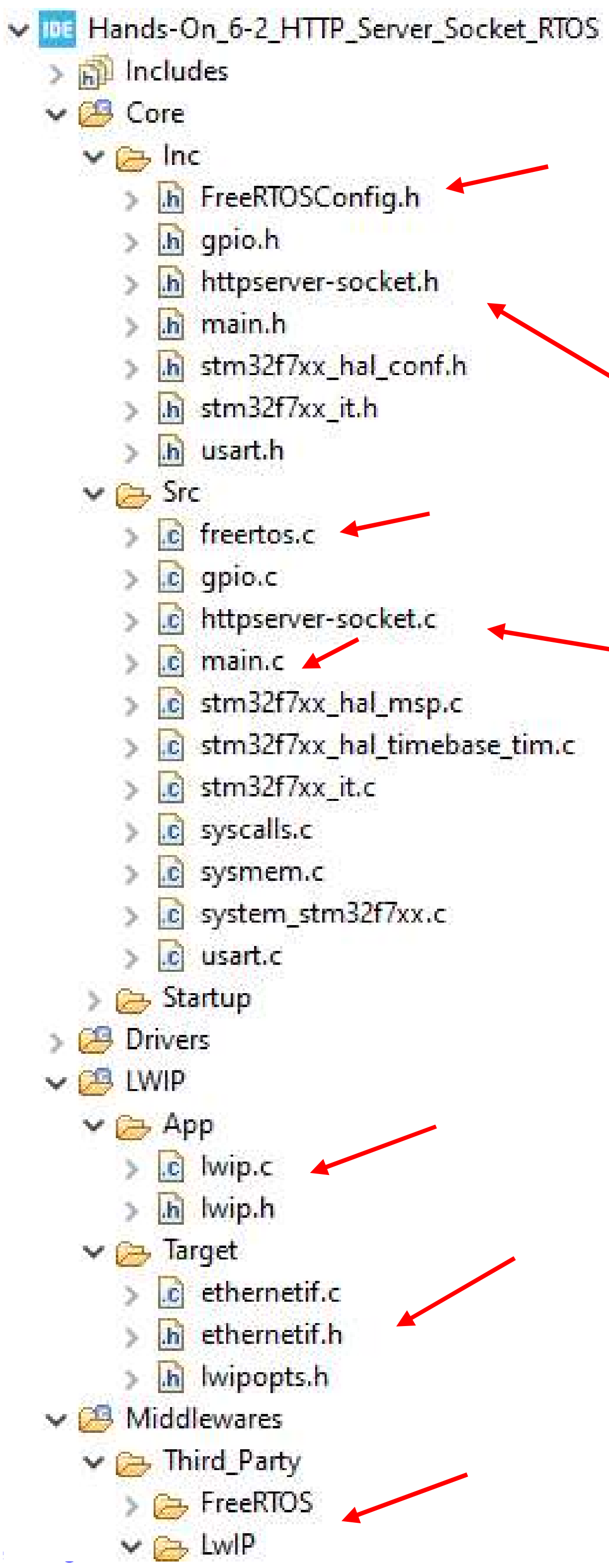
Refer to the previous example for setting up these files, pages 10 – 12, and pages 15 – 16.

standby for copying

This **`STM32F767TASKS.html`** file is converted to **`fsdata_StartPage.c`**

These files (webpages) are converted to **`fsdata_custom.c`**

Hands-On LwIP HTTP Server Socket RTOS



```
Hands-On_6-2_HTTP_Server_Socket_RTOS
  Includes
  Core
    Inc
      FreeRTOSConfig.h
      gpio.h
      httpserver-socket.h
      main.h
      stm32f7xx_hal_conf.h
      stm32f7xx_it.h
      usart.h
    Src
      freertos.c
      gpio.c
      httpserver-socket.c
      main.c
      stm32f7xx_hal_msp.c
      stm32f7xx_hal_timebase_tim.c
      stm32f7xx_it.c
      syscalls.c
      sysmem.c
      system_stm32f7xx.c
      usart.c
    Startup
  Drivers
  LWIP
    App
      lwip.c
      lwip.h
    Target
      ethernetif.c
      ethernetif.h
      lwipopts.h
  Middlewares
    Third_Party
      FreeRTOS
      LwIP
```

Copy two files to this project:

httpserver-socket.h

httpserver-socket.c

Hands-On LwIP HTTP Server Socket RTOS

Part of the **main.c**

UM1713 User manual

Developing applications on STM32Cube with LwIP TCP/IP stack

Section 6 Using the LwIP applications

6.2.3 Web Server based on Socket RTOS

```
/* Part of the main.c */
/* Includes */
#include "main.h"
#include "cmsis_os.h"
#include "lwip.h"
#include "usart.h"
#include "gpio.h"

/* Private function prototypes */
void SystemClock_Config(void);
void MX_FREERTOS_Init(void);
int main(void)
{
  /* 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_USART3_UART_Init();
  /* 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 */
  while (1) { }
}
```

Hands-On LwIP HTTP Server Socket RTOS

Add to **main.c**

```
/* USER CODE BEGIN 4 */

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD1_Pin);
    }
}

int __io_putchar(int ch)
{
    uint8_t c[1];
    c[0] = ch & 0x00FF;
    HAL_UART_Transmit(&huart3, &*c, 1, 10);
    return ch;
}

int _write(int file, char *ptr, int len)
{
    int DataIdx;
    for(DataIdx= 0; DataIdx< len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}

 /* USER CODE END 4 */
```

Add code
(optional)

The freertos.c (1/2)

Hands-On LwIP HTTP Server Socket RTOS

```
/* freertos.c */
/* Includes */
#include "FreeRTOS.h"
#include "task.h"
#include "main.h"
#include "cmsis_os.h"

/* Private includes */
/* USER CODE BEGIN Includes */
#include "httpserver-socket.h"
/* USER CODE END Includes */

osThreadId defaultTaskHandle;
void StartDefaultTask(void const * argument);

extern void MX_LWIP_Init(void);
void MX_FREERTOS_Init(void); /* (MISRA C 2004 rule 8.1) */

/* GetIdleTaskMemory prototype (linked to static allocation support) */
void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t
**ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize );

/* USER CODE BEGIN GET_IDLE_TASK_MEMORY */
static StaticTask_t xIdleTaskTCBBuffer;
static StackType_t xIdleStack[configMINIMAL_STACK_SIZE];

void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t
**ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize )
{
    *ppxIdleTaskTCBBuffer = &xIdleTaskTCBBuffer;
    *ppxIdleTaskStackBuffer = &xIdleStack[0];
    *pulIdleTaskStackSize = configMINIMAL_STACK_SIZE;
}
/* USER CODE END GET_IDLE_TASK_MEMORY */
```

Add code

The freertos.c (2/2)

Hands-On LwIP HTTP Server Socket RTOS

```c
/* @brief  FreeRTOS initialization */
void MX_FREERTOS_Init(void)
{
  /* Create the thread(s) */
  /* definition and creation of defaultTask */
  osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 1024);
  defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);
}

/* USER CODE BEGIN Header_StartDefaultTask */
/* @brief  Function implementing the defaultTask thread */
/* USER CODE END Header_StartDefaultTask */
void StartDefaultTask(void const * argument)
{
  /* init code for LWIP */
  MX_LWIP_Init();
  /* USER CODE BEGIN StartDefaultTask */

  /* Initialize webserver demo */
  http_server_socket_init();

  /* Infinite loop */
  for(;;)
  {
      osDelay(500);
      HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
  }
  /* USER CODE END StartDefaultTask */
}
```

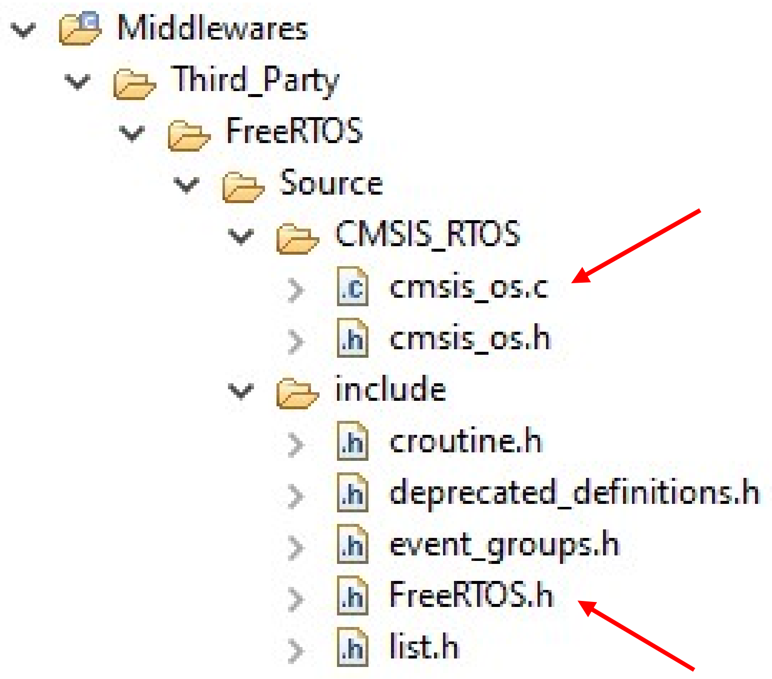
Add code

Hands-On LwIP HTTP Server Socket RTOS

Why Enabled USE_TRACE_FACILTY and USE_STATS_FORMATTING_FUNCTIONS ?

```
/* cmsis_os.c */ // Line 1535

/* Lists all the current threads, along with their current state and stack usage high water mark. */
osStatus osThreadList (uint8_t *buffer)
{
#if ( ( configUSE_TRACE_FACILITY == 1 ) && ( configUSE_STATS_FORMATTING_FUNCTIONS == 1 ) )
  vTaskList((char *)buffer);
#endif
  return osOK;
}
```

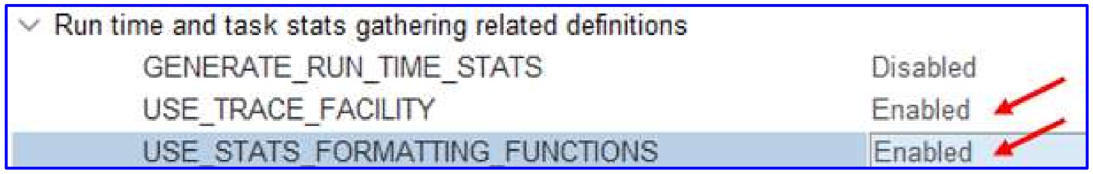
```
// Need to enable the below two settings defined in

/* FreeRTOS.h */

#ifndef configUSE_STATS_FORMATTING_FUNCTIONS     // line 7
#define configUSE_STATS_FORMATTING_FUNCTIONS  0
#endif

#ifndef configUSE_TRACE_FACILITY      // line 793
#define configUSE_TRACE_FACILITY  0
#endif
```



Hands-On LwIP HTTP Server Socket RTOS

Part of the **FreeRTOSConfig.h**

STM32CubeMX



Run time and task stats gathering related definitions	
GENERATE_RUN_TIME_STATS	Disabled
USE_TRACE_FACILITY	Enabled
USE_STATS_FORMATTING_FUNCTIONS	Enabled

```
/* Application specific definitions */

/* USER CODE BEGIN Includes */
/* Section where include file can be added */
/* USER CODE END Includes */

/* Ensure definitions are only used by the compiler, and not by the assembler. */
#if defined(__ICCARM__) || defined(__CC_ARM) || defined(__GNUC__)
    #include <stdint.h>
    extern uint32_t SystemCoreClock;
#endif
#define configENABLE_FPU                         0
#define configENABLE_MPU                         0

#define configUSE_PREEMPTION                     1
#define configSUPPORT_STATIC_ALLOCATION          1
#define configSUPPORT_DYNAMIC_ALLOCATION         1
#define configUSE_IDLE_HOOK                      0
#define configUSE_TICK_HOOK                      0
#define configCPU_CLOCK_HZ                       ( SystemCoreClock )
#define configTICK_RATE_HZ                       ((TickType_t)1000)
#define configMAX_PRIORITIES                     ( 7 )
#define configMINIMAL_STACK_SIZE                 ((uint16_t)128)
#define configTOTAL_HEAP_SIZE                    ((size_t)15360)
#define configMAX_TASK_NAME_LEN                  ( 16 )
#define configUSE_TRACE_FACILITY                 1
#define configUSE_STATS_FORMATTING_FUNCTIONS     1
#define configUSE_16_BIT_TICKS                   0
#define configUSE_MUTEXES                        1
#define configQUEUE_REGISTRY_SIZE                8
#define configUSE_PORT_OPTIMISED_TASK_SELECTION  1
```

Hands-On LwIP HTTP Server Socket RTOS

Part of the **httpserver-socket.c**

```
/* httpserver-socket.c */
/* Includes */
#include "lwip/opt.h"
#include "lwip/api.h"
#include "lwip/inet.h"
#include "lwip/sockets.h"
#include "lwip/apps/fs.h"
#include "string.h"
#include "httpserver-socket.h"
#include "cmsis_os.h"

#include <stdio.h>

/* Private typedef */
/* Private define */
#define WEBSERVER_THREAD_PRIO    ( osPriorityAboveNormal )

/* Private macro */
/* Private variables */
u32_t nPageHits = 0;
portCHAR PAGE_BODY[512];

/* Format of dynamic web page: the page header */
/* Copy from fsdata_StartPage.c after the line: */
/* raw file data (1581 bytes) */
static const unsigned char PAGE_START[] = {
0x3c,0x21,0x44,0x4f,0x43,0x54,0x59,0x50,0x45,0x20,0x68,0x74,0x6d,0x6c,0x20,0x50,
0x55,0x42,0x4c,0x49,0x43,0x20,0x22,0x2d,0x2f,0x2f,0x57,0x33,0x43,0x2f,0x2f,0x44,
0x54,0x44,0x20,0x48,0x54,0x4d,0x4c,0x20,0x34,0x2e,0x30,0x31,0x2f,0x2f,0x45,0x4e,
0x22,0x20,0x22,0x68,0x74,0x74,0x70,0x3a,0x2f,0x2f,0x77,0x77,0x77,0x2e,0x77,0x33,
. . .
```

httpserver-socket.h

```
/* Define to prevent recursive inclusion */
#ifndef __HTTPSERVER_SOCKET_H__
#define __HTTPSERVER_SOCKET_H__

void http_server_socket_init(void);

#endif /* __HTTPSERVER_SOCKET_H__ */
```

Hands-On LwIP HTTP Server Socket RTOS

Part of the httpserver-socket.c

```
0x6e,0x20,0x73,0x74,0x79,0x6c,0x65,0x3d,0x22,0x66,0x6f,0x6e,0x74,0x2d,0x66,0x61,
0x6d,0x69,0x6c,0x79,0x3a,0x20,0x56,0x65,0x72,0x64,0x61,0x6e,0x61,0x3b,0x22,0x3e,
0x4e,0x75,0x6d,0x62,0x65,0x72,0x20,0x6f,0x66,0x20,0x68,0x69,0x74,0x73,0x3a,0x20,
0x3c,0x2f,0x73,0x70,0x61,0x6e,0x3e,0x3c,0x2f,0x73,0x6d,0x61,0x6c,0x6c,0x3e,0x3c,
0x2f,0x62,0x6f,0x64,0x79,0x3e,0x3c,0x2f,0x68,0x74,0x6d,0x6c,0x3e,0x00};
// add 0x00 at the end

/* Private function prototypes */
void http_server_serve(int conn);
static void http_server_socket_thread(void const *arg);
void DynWebPage(int conn);

/* Private functions */

/**
  * @brief serve tcp connection
  * @param conn: connection socket
  * @retval None
  */
void http_server_serve(int conn)
{
  int buflen = 1500;
  int ret;
  struct fs_file file;
  unsigned char recv_buffer[1500];

  /* Read in the request */
  ret = read(conn, recv_buffer, buflen);
  if(ret < 0) return;
```

Part of the **httpserver-socket.c**

```
/* Check if request to get ST.gif */
if (strncmp((char *)recv_buffer,"GET /STM32F767_files/ST_DigiPen.jpg",35)==0) //ST.gif
{
  fs_open(&file, "/STM32F767_files/ST_DigiPen.jpg"); //ST.gif
  write(conn, (const unsigned char*)(file.data), (size_t)file.len);
  fs_close(&file);
}
/* Check if request to get stm32.jpeg */
else if (strncmp((char *)recv_buffer,"GET /STM32F767_files/stm32.jpg",30)==0)
{
  fs_open(&file, "/STM32F767_files/stm32.jpg");
  write(conn, (const unsigned char*)(file.data), (size_t)file.len);
  fs_close(&file);
}
/* Check if request to get ST logo.jpeg */
else if (strncmp((char *)recv_buffer,"GET /STM32F767_files/logo.jpg", 29) == 0)
{
  fs_open(&file, "/STM32F767_files/logo.jpg");
  write(conn, (const unsigned char*)(file.data), (size_t)file.len);
  fs_close(&file);
}
/* Check if request to get DigiPen logo.jpg */
else if (strncmp((char *)recv_buffer,"GET /STM32F767_files/digipen.gif", 32) == 0)
{
  fs_open(&file, "/STM32F767_files/digipen.gif");
  write(conn, (const unsigned char*)(file.data), (size_t)file.len);
  fs_close(&file);
}
```

Part of the `httpserver-socket.c`

```
 else if(strncmp((char *)recv_buffer, "GET /STM32F767TASKS.html", 24) == 0)
  {
    /* Load dynamic page */
    DynWebPage(conn);
  }
  else if((strncmp((char *)recv_buffer, "GET /STM32F767.html", 19) == 0)||(strncmp((char 
*)recv_buffer, "GET / ", 6) == 0))
  {
    /* Load STM32F767page */
    fs_open(&file, "/STM32F767.html");
    write(conn, (const unsigned char*)(file.data), (size_t)file.len);
    fs_close(&file);
  }
  else
  {
    /* Load 404 page */
    fs_open(&file, "/404.html");
    write(conn, (const unsigned char*)(file.data), (size_t)file.len);
    fs_close(&file);
  }
  /* Close connection socket */
  close(conn);
}
```

Hands-On LwIP HTTP Server Socket RTOS

Part of the **httpserver-socket.c**

```
/* @brief  http server thread */
static void http_server_socket_thread(void const *arg)
{
    int sock, newconn, size;
    struct sockaddr_in address, remotehost;

    /* create a TCP socket */
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)    // Domain, type, Protocol
    {
        return;
    }

    /* bind to port 80 at any interface */
    address.sin_family = AF_INET;            // sin = socket_in
    address.sin_port = htons(80);
    address.sin_addr.s_addr = INADDR_ANY;

    if (bind(sock, (struct sockaddr *)&address, sizeof (address)) < 0)
    {
        return;
    }

    /* listen for incoming connections (TCP listen backlog = 5) */
    listen(sock, 5);

    size = sizeof(remotehost);

    while (1)
    {
        newconn = accept(sock, (struct sockaddr *)&remotehost, (socklen_t *)&size);
        http_server_serve(newconn);
    }
}
```

Hands-On LwIP HTTP Server Socket RTOS

Part of the `httpserver-socket.c`

```
/**
  * @brief  Initialize the HTTP server (start its thread)
  * @param  none
  * @retval None
  */
void http_server_socket_init()
{
    // sys_thread_new("HTTP", http_server_socket_thread, NULL, DEFAULT_THREAD_STACKSIZE * 2,
                                                                      WEBSERVER_THREAD_PRIO);

    osThreadDef(HTTP, http_server_socket_thread, WEBSERVER_THREAD_PRIO, 0,
                                                             DEFAULT_THREAD_STACKSIZE * 2);
    osThreadCreate(osThread(HTTP), NULL);

    // note: 1. Heap size must be large enough (63488 bytes) to have (configMINIMAL_STACK_SIZE*2),
    //          else (configMINIMAL_STACK_SIZE) is working too for this program
    // note: 2. If sys_thread_new() is used, http_server_socket_thread(void const *arg) must be
    //          reduced to http_server_socket_thread(), i.e., no passing of argument in
    //          http_server_socket_thread().
    // note: 3. Set configMINIMAL_STACK_SIZE = DEFAULT_THREAD_STACKSIZE = 2014 words

}
```

Part of the **httpserver-socket.c**

```
/**
  * @brief  Create and send a dynamic Web Page. This page contains the list of
  *         running tasks and the number of page hits.
  * @param  conn connection socket
  * @retval None
  */
void DynWebPage(int conn)
{
  portCHAR pagehits[10];

  memset(PAGE_BODY, 0, 512);

  /* Update the hit count */
  nPageHits++;
  sprintf( pagehits, "%d", (int)nPageHits );
  strcat(PAGE_BODY, pagehits);
  strcat((char *) PAGE_BODY, "<pre><br>Name          State  Priority  Stack   Num" );
  strcat((char *) PAGE_BODY, "<br>---------------------------------------------<br>");

  /* The list of tasks and their status */
  osThreadList((unsigned char *)(PAGE_BODY + strlen(PAGE_BODY)));
  strcat((char *) PAGE_BODY, "<br><br>---------------------------------------------");
  strcat((char *) PAGE_BODY, "<br>B : Blocked, R : Ready, D : Deleted, S : Suspended<br>");

  /* Send the dynamically generated page */
  write(conn, PAGE_START, strlen((char*)PAGE_START));
  write(conn, PAGE_BODY, strlen(PAGE_BODY));
}
```

Hands-On LwIP HTTP Server Socket RTOS

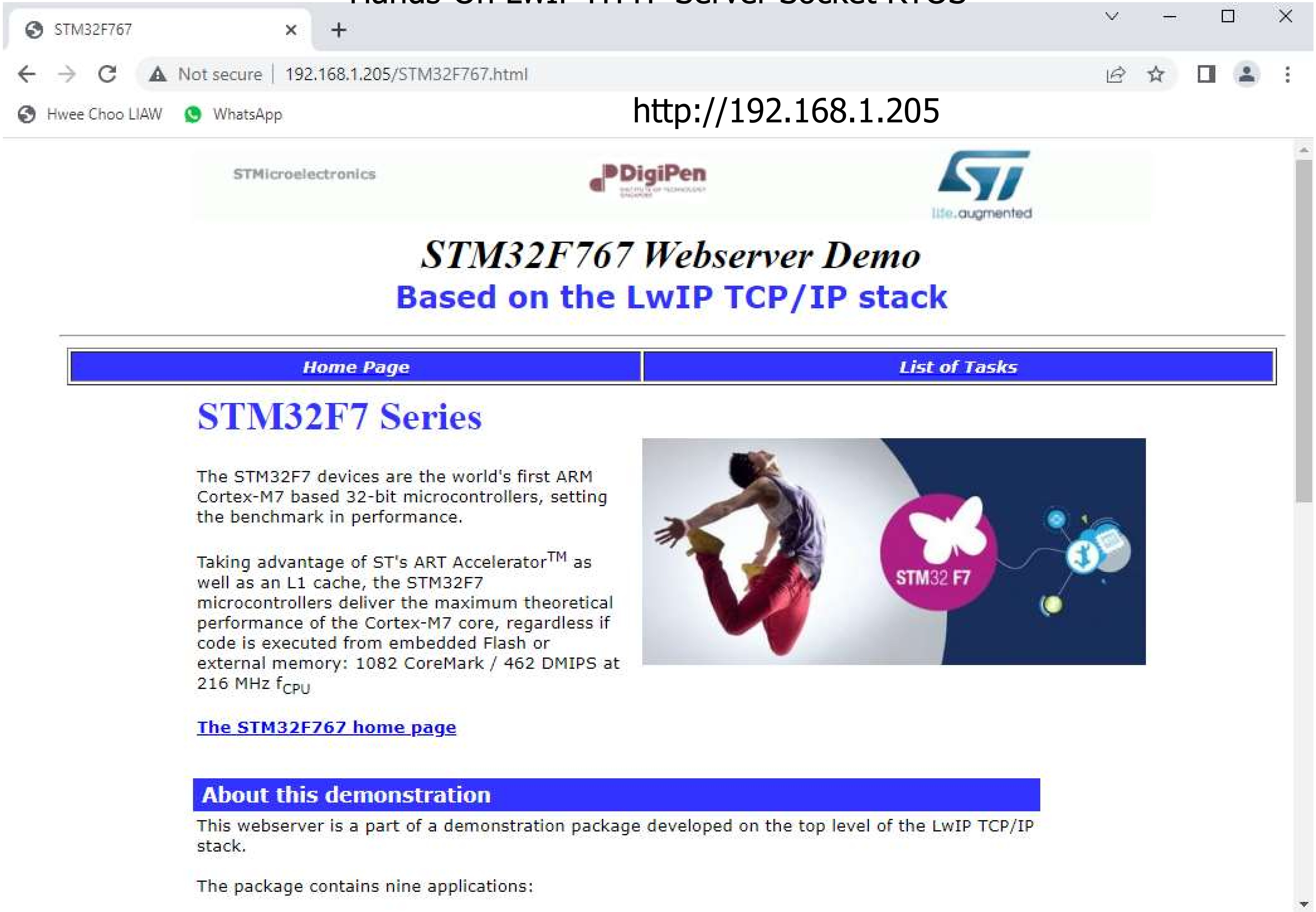
Generated Code in **`Lwip.c`**

```
/* LwIP initialization function  */
void MX_LWIP_Init(void)
{
  /* IP addresses initialization */
  IP_ADDRESS[0] = 192;
  IP_ADDRESS[1] = 168;
  IP_ADDRESS[2] = 1;
  IP_ADDRESS[3] = 205;
  NETMASK_ADDRESS[0] = 255;
  NETMASK_ADDRESS[1] = 255;
  NETMASK_ADDRESS[2] = 255;
  NETMASK_ADDRESS[3] = 0;
  GATEWAY_ADDRESS[0] = 192;
  GATEWAY_ADDRESS[1] = 168;
  GATEWAY_ADDRESS[2] = 1;
  GATEWAY_ADDRESS[3] = 1;

/* USER CODE BEGIN IP_ADDRESSES */
/* USER CODE END IP_ADDRESSES */

  /* Initilialize the LwIP stack without RTOS */
  lwip_init();

  /* IP addresses initialization without DHCP (IPv4) */
  IP4_ADDR(&ipaddr, IP_ADDRESS[0], IP_ADDRESS[1], IP_ADDRESS[2], IP_ADDRESS[3]);
  IP4_ADDR(&netmask, NETMASK_ADDRESS[0], NETMASK_ADDRESS[1] , NETMASK_ADDRESS[2], NETMASK_ADDRESS[3]);
  IP4_ADDR(&gw, GATEWAY_ADDRESS[0], GATEWAY_ADDRESS[1], GATEWAY_ADDRESS[2], GATEWAY_ADDRESS[3]);

  /* add the network interface (IPv4/IPv6) without RTOS */
  netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &ethernetif_init, &ethernet_input);
```

For a different router (gateway):

```
IP_ADDRESS[0] = 192;
IP_ADDRESS[1] = 168;
IP_ADDRESS[2] = 50;
IP_ADDRESS[3] = 205;
NETMASK_ADDRESS[0] = 255;
NETMASK_ADDRESS[1] = 255;
NETMASK_ADDRESS[2] = 255;
NETMASK_ADDRESS[3] = 0;
GATEWAY_ADDRESS[0] = 192;
GATEWAY_ADDRESS[1] = 168;
GATEWAY_ADDRESS[2] = 50;
GATEWAY_ADDRESS[3] = 1;
```

Hands-On LwIP HTTP Server Socket RTOS



http://192.168.1.205

STM32F767 Webserver Demo

Based on the LwIP TCP/IP stack

Home Page | ***List of Tasks***

STM32F7 Series

The STM32F7 devices are the world's first ARM Cortex-M7 based 32-bit microcontrollers, setting the benchmark in performance.

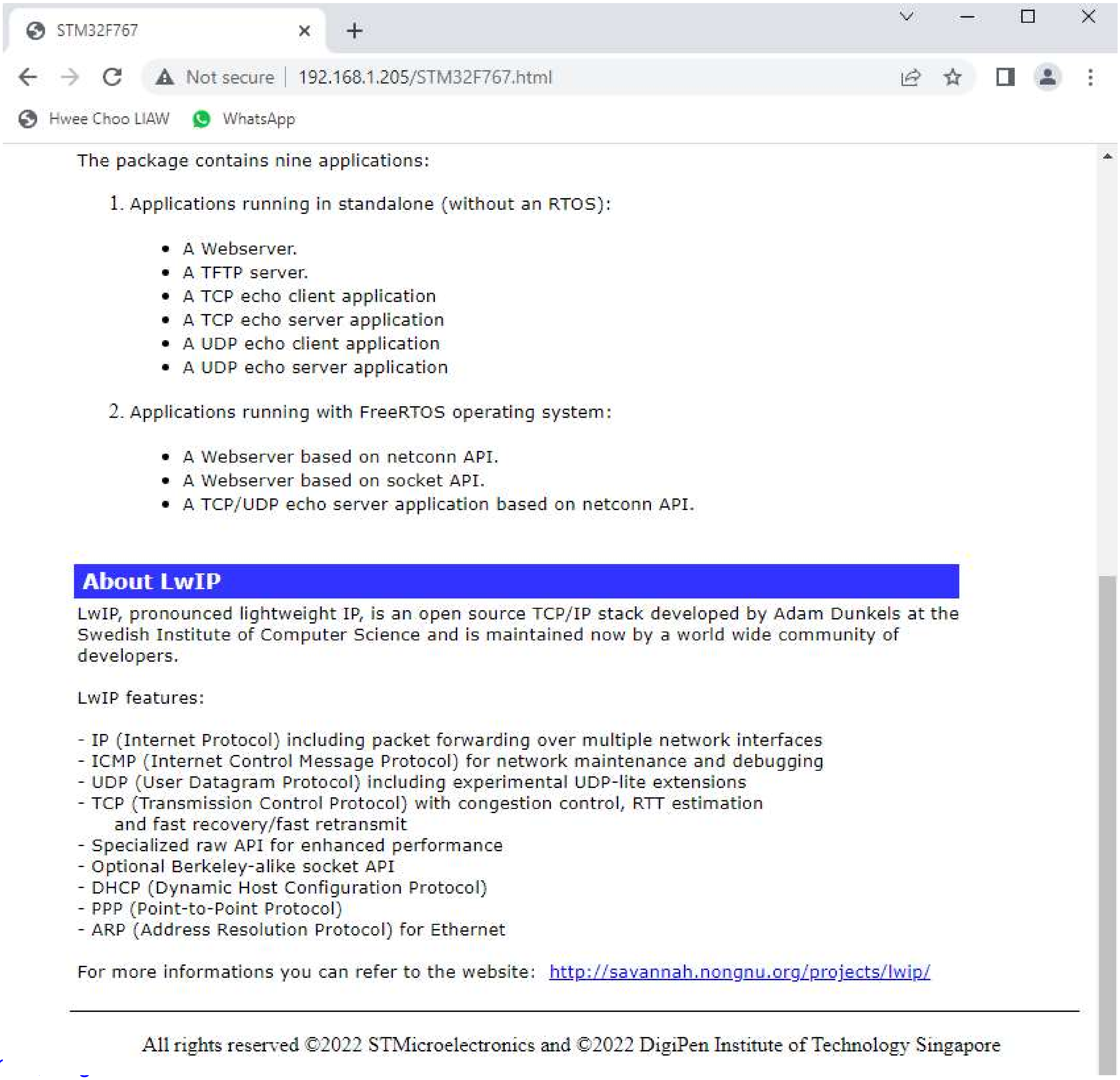
Taking advantage of ST's ART Accelerator^{TM} as well as an L1 cache, the STM32F7 microcontrollers deliver the maximum theoretical performance of the Cortex-M7 core, regardless if code is executed from embedded Flash or external memory: 1082 CoreMark / 462 DMIPS at 216 MHz f_{CPU}

The STM32F767 home page

About this demonstration

This webserver is a part of a demonstration package developed on the top level of the LwIP TCP/IP stack.

The package contains nine applications:



The package contains nine applications:

1. Applications running in standalone (without an RTOS):
   - A Webserver.
   - A TFTP server.
   - A TCP echo client application
   - A TCP echo server application
   - A UDP echo client application
   - A UDP echo server application
2. Applications running with FreeRTOS operating system:
   - A Webserver based on netconn API.
   - A Webserver based on socket API.
   - A TCP/UDP echo server application based on netconn API.

About LwIP

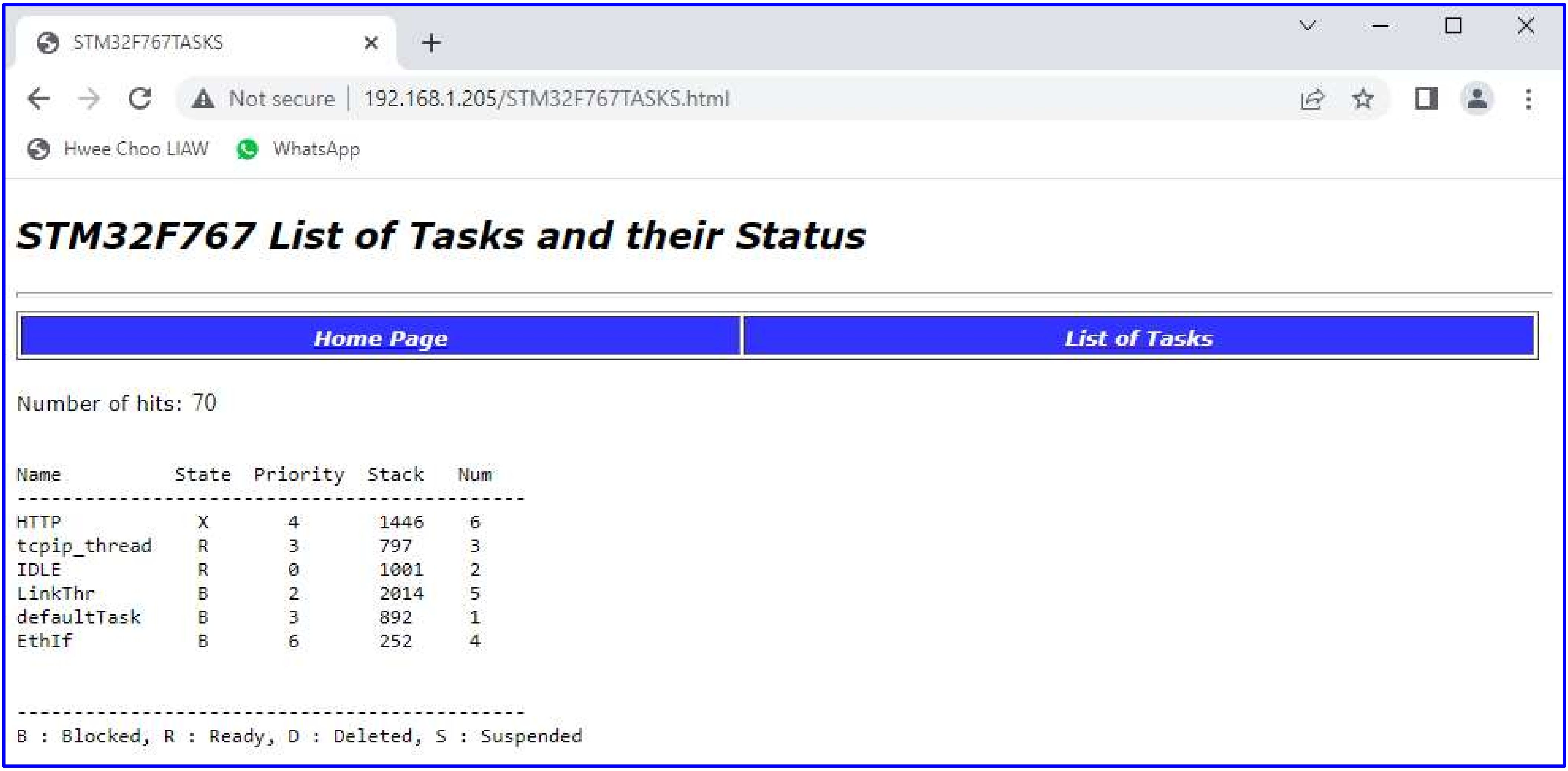
LwIP, pronounced lightweight IP, is an open source TCP/IP stack developed by Adam Dunkels at the Swedish Institute of Computer Science and is maintained now by a world wide community of developers.

LwIP features:

- IP (Internet Protocol) including packet forwarding over multiple network interfaces
- ICMP (Internet Control Message Protocol) for network maintenance and debugging
- UDP (User Datagram Protocol) including experimental UDP-lite extensions
- TCP (Transmission Control Protocol) with congestion control, RTT estimation and fast recovery/fast retransmit
- Specialized raw API for enhanced performance
- Optional Berkeley-alike socket API
- DHCP (Dynamic Host Configuration Protocol)
- PPP (Point-to-Point Protocol)
- ARP (Address Resolution Protocol) for Ethernet

For more informations you can refer to the website: http://savannah.nongnu.org/projects/lwip/

Web server lists of task page



STM32F767 List of Tasks and their Status

Number of hits: 70

```
Name          State  Priority  Stack   Num
--------------------------------------------
HTTP            X       4       1446    6
tcpip_thread    R       3       797     3
IDLE            R       0       1001    2
LinkThr         B       2       2014    5
defaultTask     B       3       892     1
EthIf           B       6       252     4


--------------------------------------------
B : Blocked, R : Ready, D : Deleted, S : Suspended
```