The lwIP library

Create a socket server on the STM32F746G-DISCOVERY board

Objectives

- learn how to create and manage threads with CMSIS-OS
- create a socket server on an embedded platform

Materials & Resources

Environment

- Make sure that System Workbench for STM32 is installed on your machine
- Make sure that STM32CubeF7 library package is installed on your machine

Training

Material	Duration
What is an Operating System as Fast As Possible	5:15
Types of Operating Systems as Fast As Possible	5:42
freeRTOS fundamentals - Multitasking	-
freeRTOS fundamentals - Scheduling	-
freeRTOS fundamentals - Real Time Applications	-
freeRTOS fundamentals - Real Time Scheduling	-
What is the difference between real time operating system and non real time operating system?	-
CMSIS-RTOS - Generic RTOS Interface	-

The following material could be useful during the workshop.

Material	Duration
CMSIS-RTOS - Function Overview	-
lwIP Socket API	-

Material Review

OS basics

- Why do we need them?
  - multitasking
  - memory management
- Real time vs Non real time
  - Scheduling determinacy
- Process vs Task vs Thread
  - these are just playing with words, but:
    - processes are running on different memory spaces
      - a process can have multiple threads
    - threads have the same shared memory
  - in the freeRTOS operating system terminology there are only tasks
    - and they use the thread expression as a synonym to task
    - in freeRTOS usually the memory is shared, so there is no memory protection
    - hence all running programs are using the same shared memory -> they are threads

RTOS basics

- The scheduler
  - Fixed scheduling tick
  - Preemptive
  - Task priorities
- Tasks
  - A function which can do something
  - Priority can be assigned
  - The higher priority task runs on the CPU until it starts to wait for something
- Communication between Tasks
  - It is possible
  - We will learn about it later

CMSIS-OS

- Based on freeRTOS
- It is a higher level abstraction on top of freeRTOS
- Creating tasks
  - osThreadDef()
  - osThreadCreate()
  - this will start the thread if the kernel is running
- Starting the OS
  - always create an initializer thread
    - create it before the kernel is started
    - always terminate the initializer thread at the end of initializing
      - osThreadTerminate()
  - osKernelStart() will start the scheduler
- Running tasks
  - the higher priority thread will run always, except if it
    - waits for another task with osWait()
    - waits for a specified time with osSleep()
  - always use osSleep() or osWait() in every task, or it will block other tasks

lwIP

- The light weight IP library
- Used in low resource systems for networking
- Lot's of APIs
- Berkley sockets

Workshop

Loading the template onto the board

- Open the template project in System Workbench for STM32
- You have to set up the board's MAC address to a unique one
  - Open the `stm32f7xx_hal_conf.h` header file
  - Search for the following part:

```
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
```

  - Change the `#define MAC_ADDR5 0U` to a unique value
    - Open this file and search for your name
    - Use the row number of your name as the value of `#define MAC_ADDR5`
    - For example for `Veréb Szabolcs` the code would look like

```
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   21U
```

- After that clean and build the project
- Program the board
- Connect the board to the network with an UTP cable
- Push the reset button on the board
- After a few seconds the board should print out the IP of the the board
  - If not ask a mentor to help find the problem!

Trying out the `LCD_Log` utility

- Try to use the following functions
  - LCD_UsrLog()
  - LCD_ErrLog()
  - LCD_DbgLog()

Socket server

The lwIP library has the same sockets as the winsock2 sockets. It can be used the same way as we used the sockets in a previous workshop.

The task is to implement a socket server on the STM32F746G-DISCOVERY board, which can accept TCP connections and receive messages, then it sends back the received message to the sender.

There are `//TODO:` comments in the template where you may have to write code.

Run the TCP client on your PC. You can use your or our solution for this purpose.

Advanced - Find the board with your PC program

Implement a broadcast server on the STM32F746G-DISCOVERY board which listens on the network for specific message, which contains a port number, just like you did in this project.

Solution

Solution