# STG - STM32 + FreeRTOS + LwIP

*This course covers the STM32 ARM-based MCU family, the FreeRTOS Real Time OS, the LWIP TCP/IP Stack and/or the EmWin GUI Stack*

## Objectives

- Get an overview on the Cortex-M architecture
- Understand the Cortex-M software implementation and debug
- Learn how to deal with interrupts
- Get an overview on STM32F4 architecture
- Describing the units which are interconnected to other modules, such as clocking, interrupt controller and DMA controller
- Describing some independent I/O modules like the ADC and GPIOs
- Getting started with the ST Drivers to program STM32 peripherals (The STM32Cube Library or ST Standard Peripheral Library)
- Understand the FreeRTOS architecture
- Discover the various FreeRTOS services and APIs
- Learn how to develop and debug FreeRTOS applications
- Getting started with the LwIP TCP/IP stack (Describing the STM32 Ethernet Controller, having a look on porting, describing the parameterizing, and developing application based on UDP and TCP protocols) (not available for STM32F0 family)
- The peripherals overview presented in this course can be detailed upon request (STR9 - STM32 Peripheralscourse)

This course can be based on STM32F4xx, STM32F2xx, STM32F1xx, or STM32F0xx families
On request TouchGFX and EmWin can be added in a specific training

## Course environment

- Convenient course material
- Example code, labs and solutions
- A STM32-Eval Board (like the STM3240G-Eval Board (Cortex-M4)) with the SW4STM32 IDE (Keil, IAR or Trace32 (Lauterbach) can also be used)

## Prerequisites

- Familiarity with C concepts and programming targeting the embedded world
- Basic knowledge of embedded processors
- Basic knowledge of multi-task scheduling
- The following courses could be of interest:
  - AAM - ARM Cortex-M Architecture (v7/v8)course
  - STR7 - STM32 F4-Series implementationcourse
  - L2 - C language for Embedded MCUscourse
  - L3 - Embedded C++course
  - STR9 - STM32 Peripheralscourse

## Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

# Course Outline

## First Day

### Cortex-M Architecture Overview

- V7-M Architecture Overview
- Core Architecture
  - Harvard Architecture, I-Code, D-Code and System Bus
  - Write Buffer
  - Bit-Banding
  - Registers (Two stacks pointers)
  - States
  - Different Running-modes and Privileged Levels
  - System Control Block
  - Systick Timer
  - MPU Overview
- Programming
  - Alignment and Endianness
  - CMSIS Library
- Exception / Interrupt Mechanism Overview
  - Vector Table
  - Interrut entry and return Overview
  - Tail-Chaining
  - Pre-emption (Nesting)
  - NVIC Integrated Interrupt Controller
  - Exception Priority Management
  - Fault escalation
- Debug Interface

**Exercise:** Becoming familiar with the IDE and clarifying the boot sequence
**Exercise:** Cortex-M4 Mode Privilege (with CMSIS library)
**Exercise:** Cortex-M4 Exception Management (put in evidence tail-chaining/nesting)
**Exercise:** Cortex-M4 MPU

### STM32F4 MCUs Architecture Overview

- ARM core based architecture
- Description of STM32Fx SoC architecture
- Clarifying the internal data and instruction paths: Bus Matrix, AHB-lite interconnect, peripheral buses, AHB-to-APB bridges, DMAs
- Memory Organization
  - Flash memory read interface
  - Adaptive Real-Time memory accelerator, instruction prefetch queue and branch cache
  - Sector and mass erase
  - Concurrent access to RAM blocks
- SoC mapping
- Flash Programming methods
- Boot Configuration

# Second Day

## Reset, Power and Clocking

- Reset
  - Reset sources
  - Boot configuration, physical remap
  - Embedded boot loader
- Clocking
  - Clock sources, HSI, HSE, LSI, LSE
  - Integrated PLLs
  - Clock outputs
  - Clock security system
- Power control
  - Power supplies, integrated regulator
  - Battery backup domain, backup SRAM
  - Independent A/D converter supply and reference voltage
  - Power supply supervisor
  - Brownout reset
  - Programmable voltage detector
- Low power modes
  - Entering a low power mode, WFI vs WFE
  - Sleep mode
  - Stop mode
  - Standby mode

**Exercise:** Configure the system to measure the current consumption in different low-power modes
**Exercise:** How to configure the programmable BOR thresholds using the FLASH option bytes
**Exercise:** How to enter the Standby mode and wake up from this mode by using an external reset/WKUP pin
**Exercise:** How to enter the Stop mode and wake up from this mode by using the RTC wakeup timer event or an interrupt

## ST Firmware Library Description

## DMA

- Dual AHB master bus architecture, one dedicated to memory accesses and one dedicated to peripheral accesses
- 8 streams for each DMA controller, up to 8 channels (requests) per stream
- Priorities between DMA stream requests
- FIFO structure
- Independent source and destination transfer width
- Circular buffer management
- Double buffer mode
- DMA1 and DMA2 request mapping

**Exercise:** DMA FIFO mode
**Exercise:** Flash To RAM using DMA

## Hardware implementation

- Power pins
- Pinout
  - Pin Muxing, alternate functions
- GPIO module
  - Configuring a GPIO
  - Speed selection
  - Locking mechanism
  - Analog function

    - Integrated pull-up / pull-down
    - I/O pin multiplexer and mapping
- System configuration controller
    - I/O compensation cell
    - External Interrupts / Wakeup lines selection
    - Ethernet PHY interface selection
- External Interrupts

**Exercise:** Configure an external Interrupt

## 12-bit Analog-to-Digital Converter

- 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
- Regular channel group vs Injected channel group
- Single and continuous conversion modes
- Scan mode
- External trigger option with configurable polarity for both regular and injected conversions
- Discontinuous mode
- Analog watchdog
- Dual/Triple mode (on devices with 2 ADCs or more)
- Configurable delay between conversions in Dual/Triple interleaved mode
- DMA request generation during regular channel conversion

**Exercise:** Get voltage from the potentiometer using, DMA transfer generation, display the result on LCD screen

## Optional: Timers Overview

- Advanced-control timers TIM1 and TIM8
    - 16-bit up, down, up/down auto-reload counter; 16-bit programmable prescaler
    - Input Capture, Output Compare, PWM generation, One-pulse mode
    - Synchronization circuit/ Controlling Timers external signals / Interconnecting several timers
    - Interrupt/DMA generation
- Real Time Clock
    - Independent BCD timer/counter; 16-bit programmable prescaler
    - Daylight saving compensation programmable by software
    - Two programmable alarms with interrupt function
    - Automatic wakeup unit
    - Reference clock detection / Digital calibration circuit
    - Tamper detection

**Exercise:** How to use DMA with TIM1 Update request to transfer Data from memory to TIM1
**Exercise:** Configuring the RTC

# Third Day

## The FreeRTOS source code

- Introduction to FreeRTOS
    - The FreeRTOS architecture and features
- Getting FreeRTOS source code
    - Files and directories structure
- Data types and coding style
    - Naming conventions
- FreeRTOS on the Cortex/M processors

## Task Management

- Prioritized Pre-emptive Scheduling / Co-operative scheduling
- The Task life-cycle
    - Task Functions

    - Creating tasks
    - Using the task parameter
    - The Task State Machine
    - Deleting tasks
- Task Priorities
    - Assigning task priorities
    - Changing task priorities
- The idle task
    - Idle task hook
- Blocking a task for a specific delay
- Editing the FreeRTOSConfig.h header file
- Suspending a task
- The Kernel Structures Overview
- FreeRTOS Debug Capabilities (Hook, Trace)
- Visual trace diagnostics using Tracealyzer

**Exercise:** Understand the notion of task context and the context switch mechanism
**Exercise:** Create a debug configuration to debug your program using a FreeRTOS-aware debugging mode
**Exercise:** Periodic Tasks
**Exercise:** Task Statistics

### Memory Management

- FreeRTOS-provided memory allocation schemes
    - Choosing the heap_x.c file depending on the application
- Adding an application-specific memory allocator
- Checking remaining free memory
- Stack monitoring
- Dimensioning Stack and Heap

**Exercise:** Direct Context Switch measurement and Stack Overflow Detection
**Exercise:** Debugging memory

## Fourth Day

### Queue Management

- Blocking on queue Reads
- Blocking on queue Writes
- Queue Creation
- Sending on a queue
- Receiving from a queue
- Sending compound types
- Transfering large data
- Queue Set Overview/ Blocking on multiple objects
- Semaphores and Events Introduction

**Exercise:** Synchronizing and communicating between tasks through queues to send datas to a bus communication

### Resource Management

- Conflict examples
- Mutual exclusion
- Critical sections
    - Disabling the interrupts
    - Suspending (locking) the scheduler
- Mutexes
    - Mutual exclusion scenario
    - API functions for Mutexes
    - Recursive Mutexes

    - Priority inversion
    - Priority inheritance
    - Deadlock
- Gatekeeper tasks

**Exercise:** Readers / Writers Problem
**Exercise:** Producer / Consumer Problem
**Exercise:** Understand deadlock and starvation

### Interrupt Management

- Binary semaphore used for interrupt synchronization
    - API function for binary semaphore
- Counting semaphores
- Using queues within an ISR
- Interrupt Nesting
    - Interrupts on Cortex-M
- Low Power Support

**Exercise:** Synchronize Interrupts with tasks
**Exercise:** Low-Power FreeRTOS Support (Tickless Mode)

## Fifth Day

### Software Timer

- The Timer Daemon Task
- Timer Configuration
- One-shot / Auto-reload Timer
- Software Timer API

**Exercise:** Understand the use of software timers

### FreeRTOS MPU

- User Mode and Privilege Mode
    - Access Permission Attributes
- Defining an MPU region
- Creating a non-privileged task
- Linker configuration
- Practical Usage Tips

### STM32 Fast Ethernet Controller Overview

- Architecture of the MAC
- Connection to PHY, RMII / MII
- Transmit and receive FIFO threshold setting
- Multicast and unicast address filtering
- Management interface
- Buffer and Buffer Descriptor organization
- Low level Drivers for STM32

### LwIP TCP/IP Stack Presentation

- Overview
- Buffer and memory management
- LwIP configuration options
- Network interfaces
- MAC and IP address settings
- IP processing

- UDP processing
- TCP processing
- Interfacing the stack
- Application Program Interface (API)
  - Standalone
  - Netconn and BSD socket library
- STM32/FreeRTOS Port Overview

**Exercise:** Run an http server application based on Netconn API of LwIP TCP/IP stack
**Exercise:** http server application based on Socket API of LwIP TCP/IP stack
**Exercise:** TCP Echo Client/Server
**Exercise:** In-Application Programming (IAP) over Ethernet using TFTP or HTTP

## Optional : EmWin GUI Stack Presentation

- Library and package description
- How to use the library
  - Configuration
  - Initialization
  - Core functions
  - Developing a multi-task application with EmWin
  - Working with some widgets (as the Windows, Buttons, Multipage, Image, ListBox, CheckBox)
  - Using the EmWinGuiBuilder software

**Exercise:** Getting started with the emWin stack, create a GUI to control input/output from the touch screen

## Optional: TouchGFX

- Basic Application Development
- Advanced Application Development
- Application Configuration
- Widgets
- Integration
- Getting Started with CubeMX and TouchGFX
- Deploying your application using ST-Link

**Exercise:** How to configure and use TouchGFX under FreeRTOS (Demo)