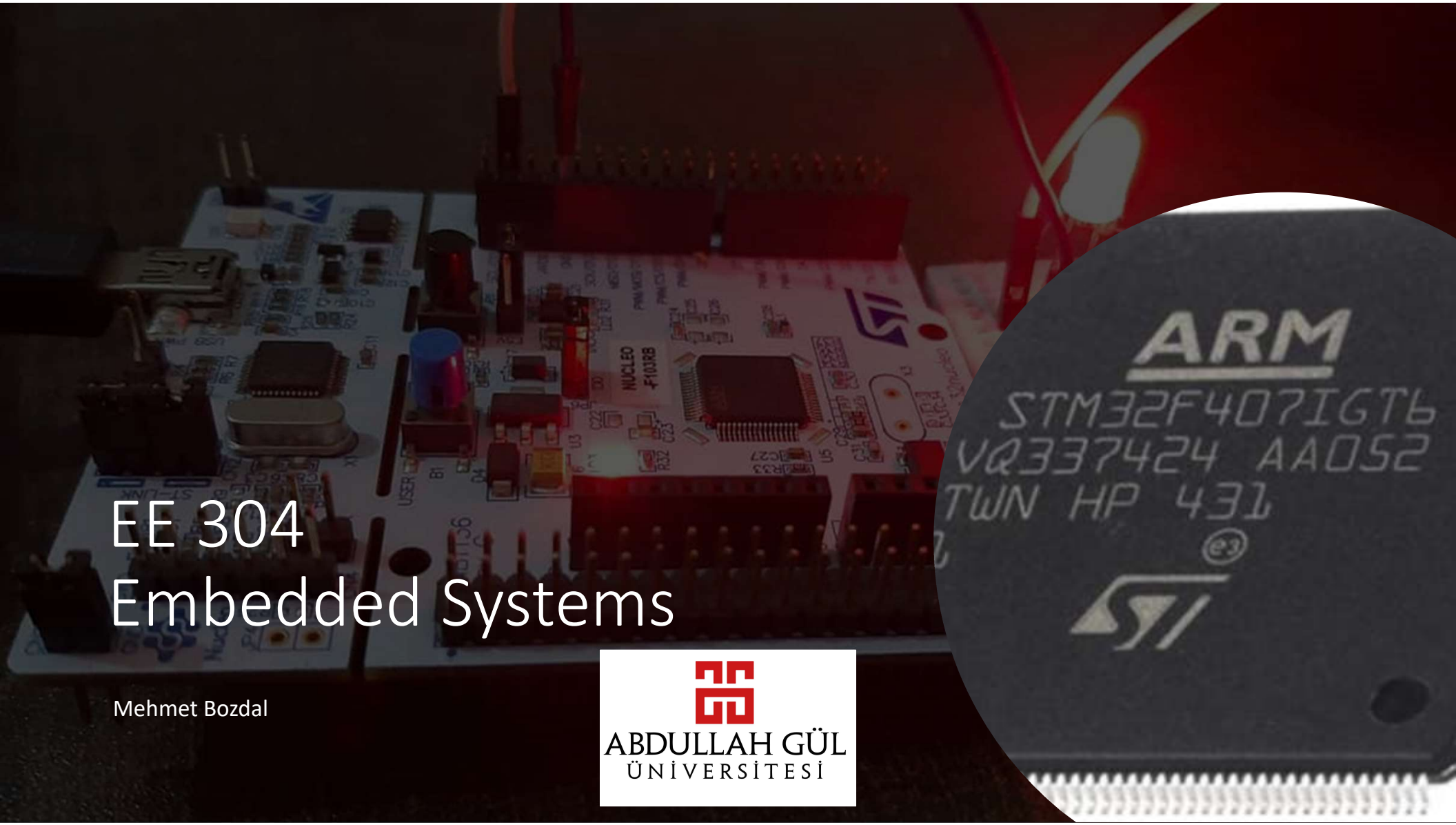


EE 304
Embedded Systems

Mehmet Bozdal

ABDULLAH GÜL
ÜNİVERSİTESİ

Agenda

- CMSIS: Cortex Microcontroller Software Interface Standard

Introduction to CMSIS

- The Cortex Microcontroller Software Interface Standard (CMSIS) is a **standardized** software interface for simplifying microcontroller software development.
  - Code reusability
  - Portability
  - Reduced development time.
- CMSIS abstracts the underlying hardware complexities, enabling end users to focus on high-level software development.

Advantages

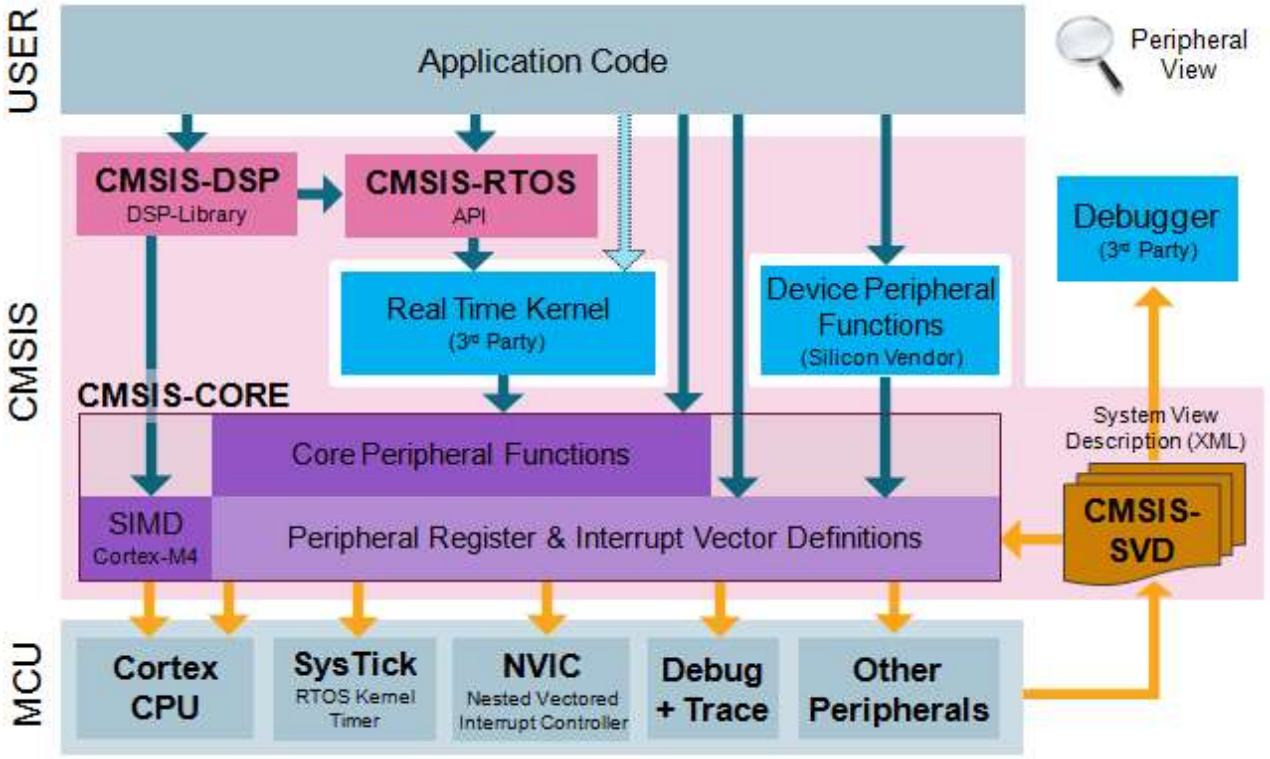
- Consistent software interfaces improve the software portability and re-usability. Generic software libraries can interface with device libraries from various silicon vendors.
- Reduces the learning curve, development costs, and time-to-market. Developers can write software quicker through an easy to use and standardized software interface.
- Provides a compiler independent layer that allows using different compilers. CMSIS is supported by all mainstream compilers
- Enhances program debugging with peripheral information for debuggers and ITM channels for printf-style output and RTOS kernel awareness.

Why Standardization?

- Despite hardware similarities, the lack of a common standard leads to diversity in HAL libraries. This diversity increases software complexity and cost.
- Reusing software and adopting common standards become crucial for cost-effective development.
- CMSIS leverages a common framework for core-specific components, streamlining the development of code for ARM Cortex-M-based devices.
- CMSIS supports a wide range of microcontroller devices and users can switch between different devices without rewriting substantial portions of their code, promoting flexibility in development.
- CMSIS integrates seamlessly with common microcontroller development environments.

CMSIS Overview

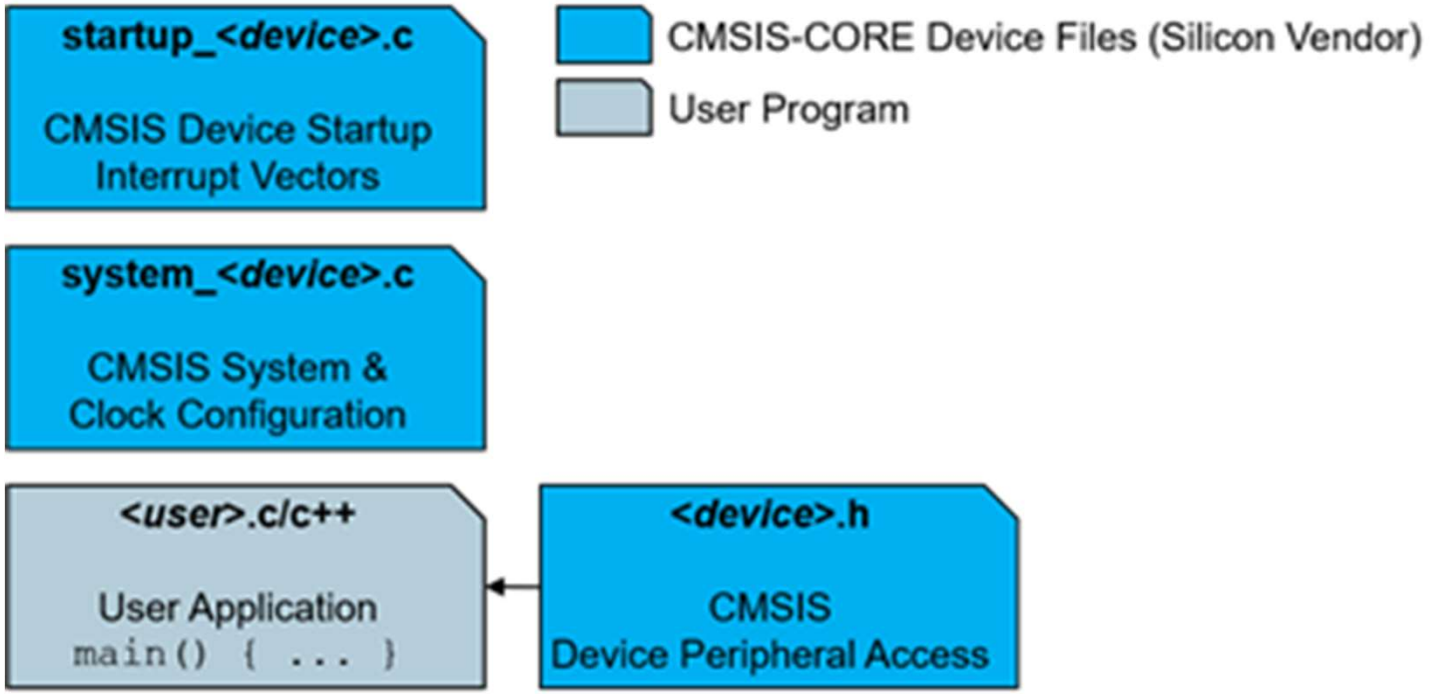
- CMSIS-CORE
- CMSIS-DSP
- CMSIS-RTOS API
- CMSIS-SVD



Core Access Functions

- **Hardware Abstraction Layer (HAL)** for Cortex-M processor registers with standardized definitions for the SysTick, NVIC, System Control Block registers, MPU registers, FPU registers, and core access functions.
- **System exception names** to interface to system exceptions without having compatibility issues.
- **Methods to organize header files** that makes it easy to learn new Cortex-M microcontroller products and improve software portability. This includes naming conventions for device-specific interrupts.
- **Methods for system initialization** to be used by each MCU vendor. For example, the standardized SystemInit() function is essential for configuring the clock system of the device.
- **Intrinsic functions** used to generate CPU instructions that are not supported by standard C functions.
- A variable to determine the **system clock frequency** which simplifies the setup the SysTick timer.

Using CMSIS in Embedded Applications



Startup File startup_<device>.c

- The reset handler which is executed after CPU reset and typically calls the SystemInit function.
- The setup values for the Main Stack Pointer (MSP).
- Exception vectors of the Cortex-M Processor with weak functions that implement default routines.
- Interrupt vectors that are device specific with weak functions that implement default routines.

System Configuration Files

- The System Configuration Files system_<device>.c and system_<device>.h provides as a minimum the functions described under System and Clock Configuration.
- These functions are device specific and need adaptations. In addition, the file might have configuration settings for the device such as XTAL frequency or PLL prescaler settings.
- As a minimum requirement, this file must provide:
  - A device-specific system configuration function, SystemInit().
  - A global variable that contains the system frequency, SystemCoreClock.

Device Header File <device.h>

- **Peripheral Access** provides a standardized register layout for all peripherals. Optionally functions for device-specific peripherals may be available.
- **Interrupts and Exceptions (NVIC)** can be accessed with standardized symbols and functions for the Nested Interrupt Vector Controller (NVIC) are provided.
- **Intrinsic Functions for CPU Instructions** allow to access special instructions, for example for activating sleep mode or the NOP instruction.
- **Intrinsic Functions for SIMD** Instructions provide access to the DSP-oriented instructions.
- **Systick Timer (SYSTICK)** function to configure and start a periodic timer interrupt.
- **Debug Access** are functions that allow printf-style I/O via the CoreSight Debug Unit and ITM communication.

Keil_create_project

Further Readings

- https://www.arm.com/technologies/cmsis

Q&A

Any questions?