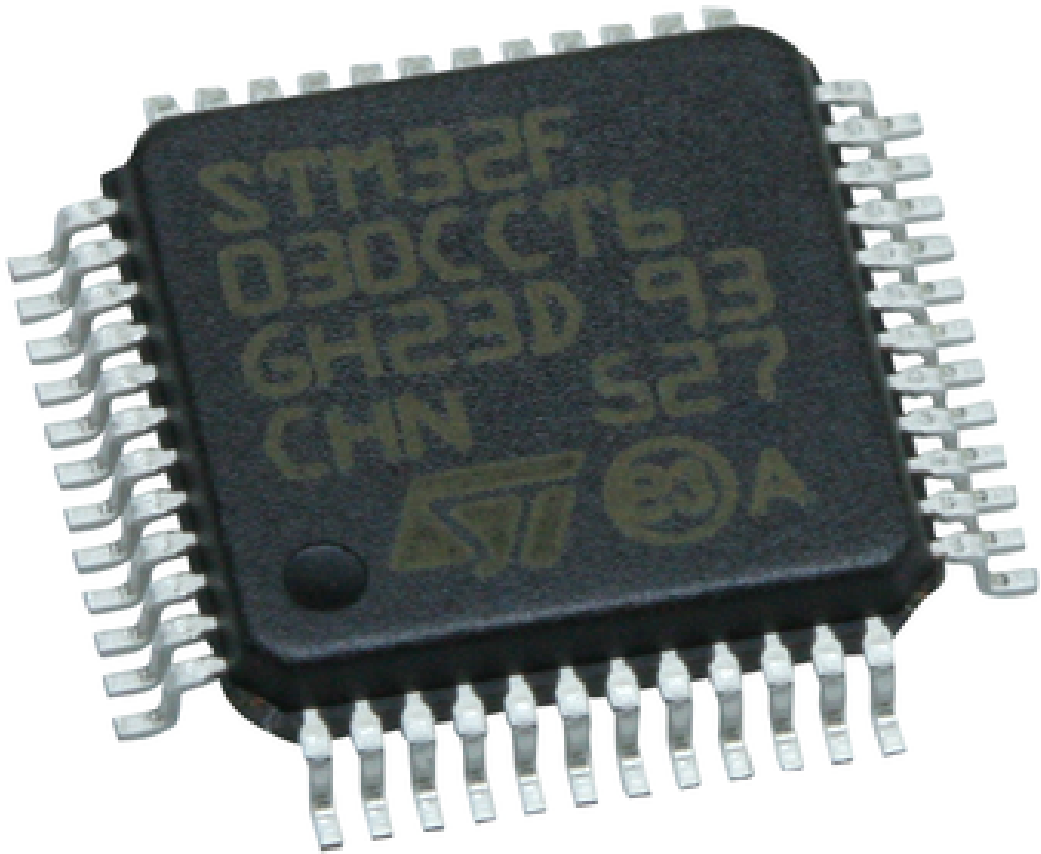
# Embedded Systems Hands-On 1: Design and Implementation of Hardware/Software Systems

## Task 3: Cortex-M0 Bare-metal Programming

TECHNISCHE UNIVERSITÄT DARMSTADT



# Questions so far

TECHNISCHE UNIVERSITÄT DARMSTADT

- Any problems with group work?
- Any questions regarding Task 2?
- Two weeks appropriate for Task 2?

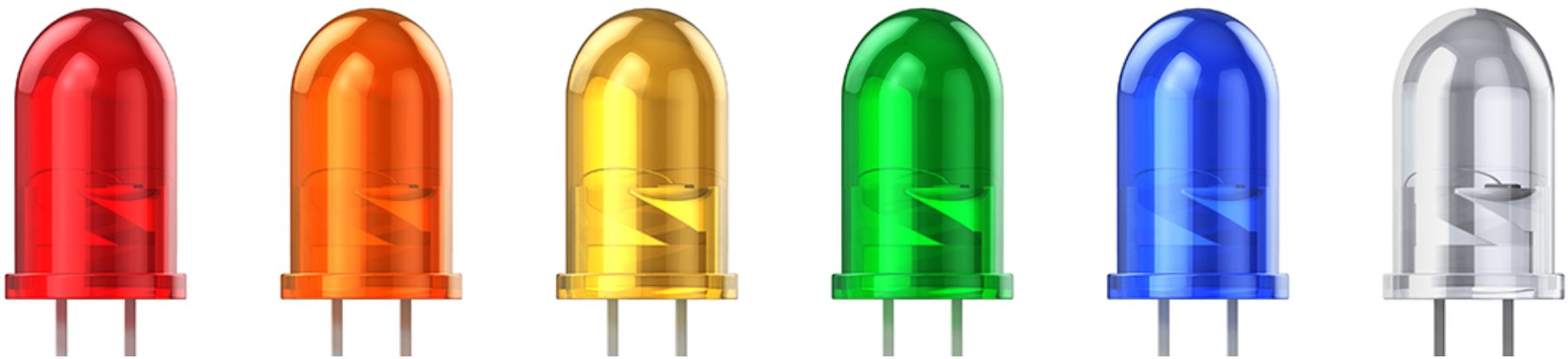
- Consultation Hour: Thursdays 11am

# Task 3:
# Cortex-M0 Bare-metal Programming

TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduction to basic microcontroller programming
- Read and understand datasheets
- Controlling GPIO and timers via CMSIS
- Event-based control with polling and interrupts

$\Rightarrow$ Blinking LED = "Hello World" of embedded systems



# Subtasks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Blinky: Toggle LED with Timer
- Better Blinky: Use Interrupts

# Development Environment

TECHNISCHE UNIVERSITÄT DARMSTADT

- Provided via GitLab
- Includes CMSIS library (used in this task)
- Includes processor startup code

# Startup-Code

TECHNISCHE UNIVERSITÄT DARMSTADT

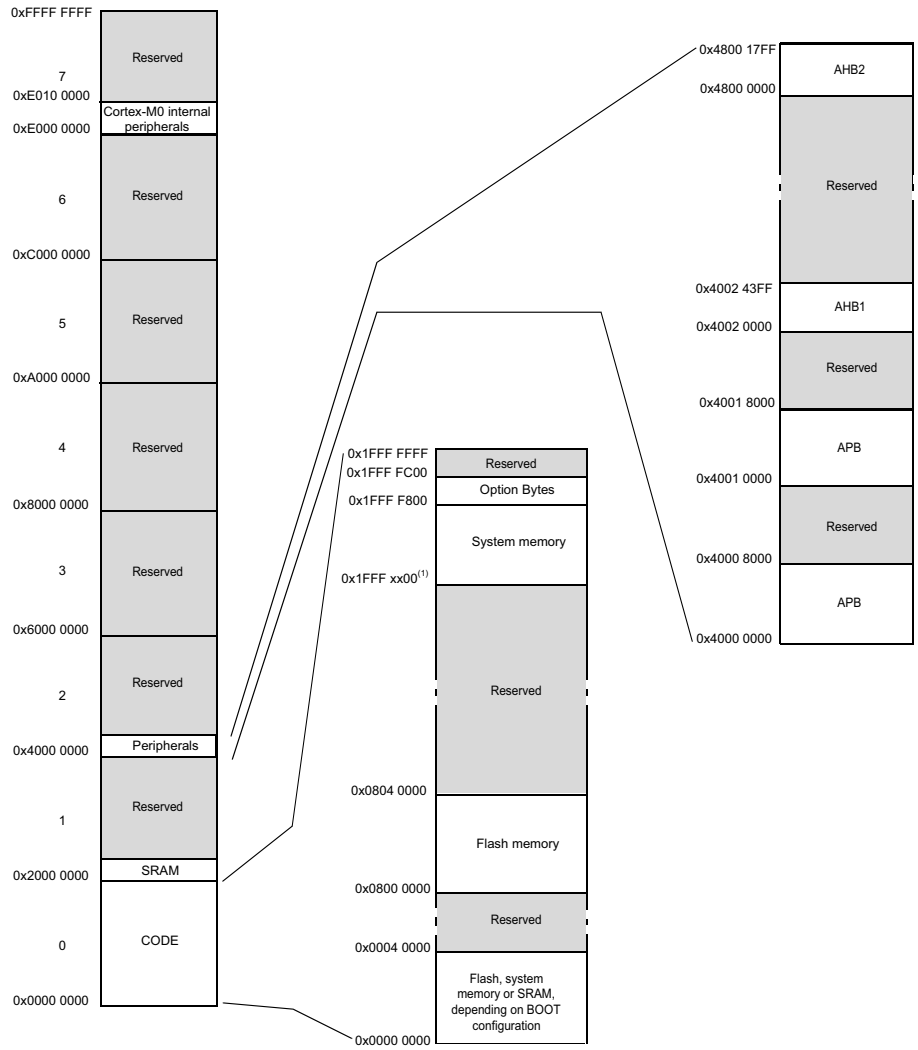
- Initializes the microcontroller
- Prepares basic settings (e.g., clock frequency)
- Typically provided by MCU vendor

```
...
/* Reset vector :
   Set up environment to
   call C main() */

.thumb_func
Reset_Handler:

/* Copy initialized data
   from flash to RAM */

copy_data:
ldr r1, DATA_BEG
ldr r2, TEXT_END
ldr r3, DATA_END
...
```

# Memory Mapped IO

TECHNISCHE UNIVERSITÄT DARMSTADT

- All peripherals are accessed via special memory addresses
- Write/Read special registers (e.g., to toggle GPIO pin)
- Otherwise, special instructions would be required



# Cortex-M0 Reference Manual: RM0360

TECHNISCHE UNIVERSITÄT DARMSTADT

- 779 pages documentation of peripherals
- Uniformly structured:
  - Feature, e.g.,
    - GPIO,
    - Reset and clock control,
    - Interrupts and events
  - Subfeature (if available), e.g.,
    - Reset,
    - Clocks
    - Nested vectored interrupt controller
  - Introduction: what is it used for
  - Block diagrams and implementation details
  - *Register mapping and description for memory mapped IO*


**Independent watchdog (IWDG) . . . . . .**

19.1 Introduction . . . . . .

19.2 IWDG main features . . . . . .

19.3 IWDG functional description . . . . . .

19.3.1 IWDG block diagram . . . . . .

19.3.2 Window option . . . . . .

19.3.3 Hardware watchdog . . . . . .

19.3.4 Behavior in Stop and Standby modes

19.3.5 Register access protection . . . . . .

19.3.6 Debug mode . . . . . .

19.4 IWDG registers . . . . . .

19.4.1 Key register (IWDG_KR) . . . . . .

19.4.2 Prescaler register (IWDG_PR) . . . . .

19.4.3 Reload register (IWDG_RLR) . . . . . .

19.4.4 Status register (IWDG_SR) . . . . . .

19.4.5 Window register (IWDG_WINR) . . . .

19.4.6 IWDG register map . . . . . .

# Programming Language

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use C or C++
- Dynamic memory allocation may cause problems in memory constrained embedded devices

$\Rightarrow$ Should be avoided

- Avoid polymorphism in C++
- Templates are well supported
- Use abstraction to organize your code, but keep in mind how things are realized

```
#include <ch.hpp>

int yes[1024];
int main(void) {
  halInit();
  System::init();
  int *nope = new[1024];
  awesomeFun(&yes);
  return 0;
}
```

# Alternative Programming Languages

TECHNISCHE UNIVERSITÄT DARMSTADT

- More powerful microcontrollers also support scripting languages
- Suitable for simple control applications and prototyping
- Not targeted in this lab
- You may have a look at
  - `https://github.com/micropython`
  - `http://www.eluaproject.net`

# Embedded Systems Hands-On 1: Design and Implementation of Hardware/Software Systems

heinz@esa.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT