# ARM Cortex Core Microcontrollers

## 1st Laboratory: CMSIS

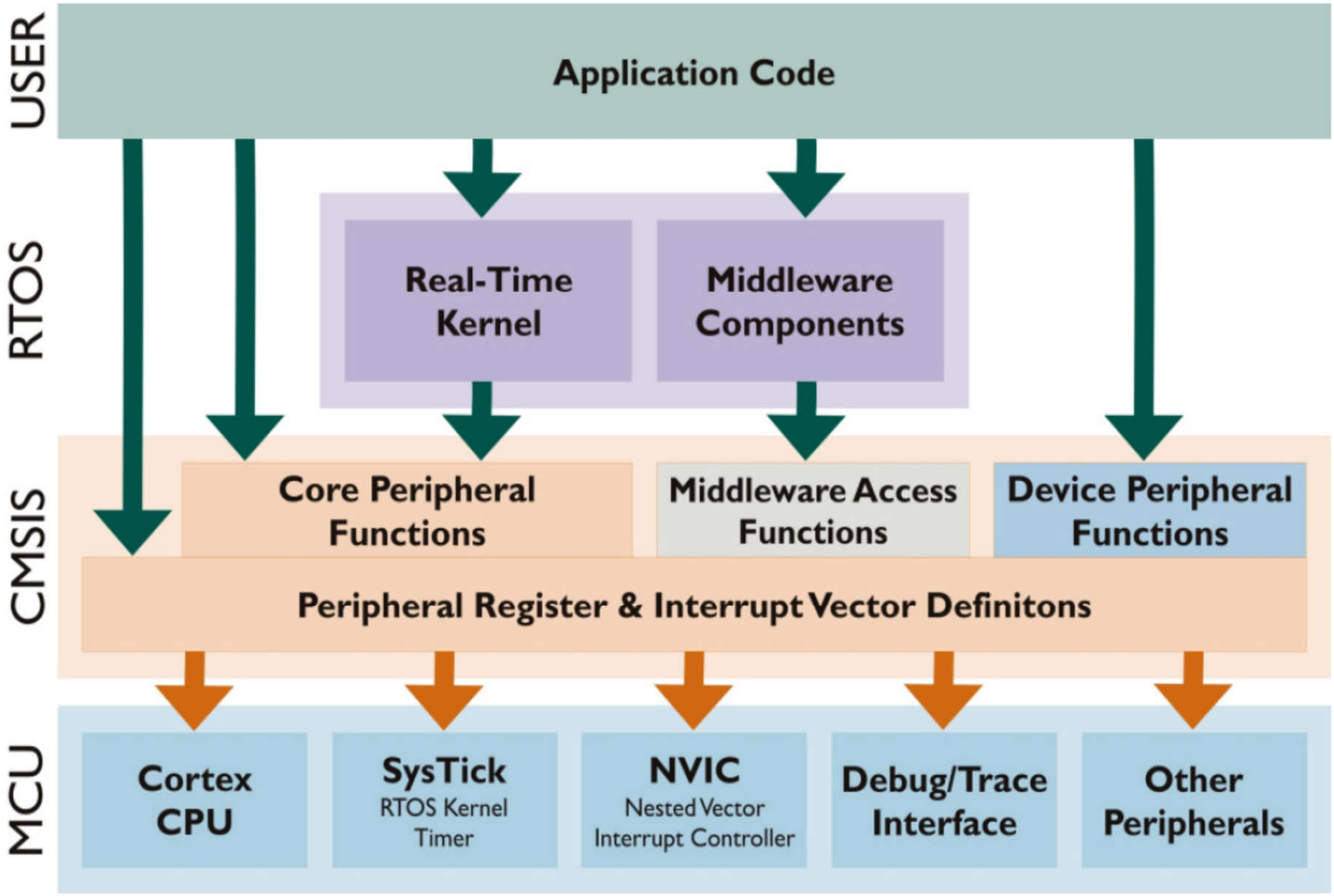
Scherer Balázs

Méréstechnika és
Információs Rendszerek
Tanszék

**Budapest University of Technology and Economics**
**Department of Measurement and Information Systems**



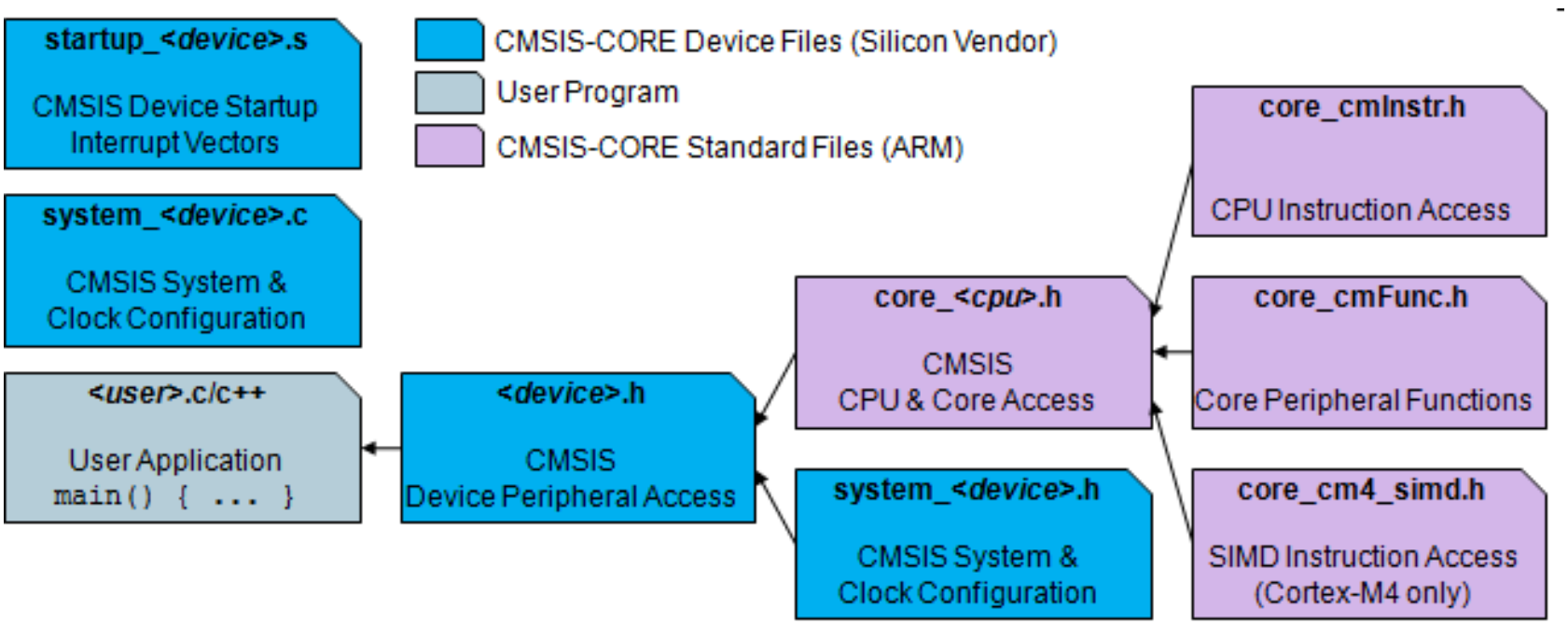
# CMSIS architecture (v1.3)



# CMSIS Core

- **Hardware Abstraction Layer (HAL):** Standardized peripheral handling for al Cortex M core variant. Standard for register access and internal peripheral functions like SysTick, NVIC, MPU and FPU.
- **Exception handling:** Standardized names and function interfaces
- **Header file organization:** Naming conventions
- **System start:** Standardized SystemInit() function to cover microcontroller specific clock startups
- **Support for special instructions**
- Global variable for **system clock** frequncy

# CMSIS core files



# The *startup_device* file

- The interrupt table

```
* The minimal vector table for a Cortex M3. Note that the proper constructs
* must be placed on this to ensure that it ends up at physical address
* 0x0000.0000.
*
*******************************************************************************/
    .section  .isr_vector,"a",%progbits
  .type  g_pfnVectors, %object
  .size  g_pfnVectors, .-g_pfnVectors


g_pfnVectors:
  .word  _estack
  .word  Reset_Handler
  .word  NMI_Handler
  .word  HardFault_Handler
  .word  MemManage_Handler
  .word  BusFault_Handler
  .word  UsageFault_Handler
  .word  0
  .word  0
  .word  0
  .word  0
  .word  SVC_Handler
  .word  DebugMon_Handler
  .word  0
  .word  PendSV_Handler
  .word  SysTick_Handler

  /* External Interrupts */
  .word     WWDG_IRQHandler                   /* Window WatchDog              */
  .word     PVD_IRQHandler                    /* PVD through EXTI Line detection */
  .word     TAMP_STAMP_IRQHandler             /* Tamper and TimeStamps through the EXTI line */
```

Allocating this code part at the start of the Flash

The vector table, including empty spaces

# The Reset_Handler

```
70   .weak  Reset_Handler
71   .type  Reset_Handler, %function
72 Reset_Handler:
73   ldr   sp, =_estack    /* Atollic update: set stack pointer */
74
75 /* Copy the data segment initializers from flash to SRAM */
76   movs  r1, #0
77   b  LoopCopyDataInit
78
79 CopyDataInit:
80   ldr  r3, =_sidata
81   ldr  r3, [r3, r1]
82   str  r3, [r0, r1]
83   adds  r1, r1, #4
84
85 LoopCopyDataInit:
86   ldr  r0, =_sdata
87   ldr  r3, =_edata
88   adds  r2, r0, r1
89   cmp  r2, r3
90   bcc  CopyDataInit
91   ldr  r2, =_sbss
92   b  LoopFillZerobss
93 /* Zero fill the bss segment. */
94 FillZerobss:
95   movs  r3, #0
96   str  r3, [r2], #4
97
98 LoopFillZerobss:
99   ldr  r3, = _ebss
100  cmp  r2, r3
101  bcc  FillZerobss
102
103 /* Call the clock system intitialization function.*/
104  bl  SystemInit
105 /* Call static constructors */
106    bl __libc_init_array
107 /* Call the application's entry point.*/
108  bl  main
109  bx  lr
110 .size  Reset Handler, .-Reset Handler
```

Setting stack pointer

Copy the start values of the initialized variable from the Flash-to the SRAM

Clearing the values of uninicialized variables

Calling the SystemInit CMSIS function to setup clocks

Calling the main function

# The system_device.c

- Minimum services to start the microcontroller

| Function | description |
|---|---|
| void SystemInit (void) | Function to set up the system clocks |
| void SystemCoreClockUpdate(void) | Upgrading the system clock. |

| Variable | description |
|---|---|
| uint32_t SystemCoreClock | The current value of the system clock. |

# Controlling LEDS on the STM32F429 Discovery

- User LD3:  The green LED connected to the I/O PG13
- User LD4:  The red LED connected to the I/O PG14

# Enabling Clock

### 6.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reser-ved | OTGHS ULPIEN | OTGHSEN | ETHMACPTPEN | ETHMACRXEN | ETHMACTXEN | ETHMACEN | Res. | **DMA2DEN** | DMA2EN | DMA1EN | CCMDATARAMEN | Res. | BKPSRAMEN | Reserved | |
| | rw | rw | rw | rw | rw | rw | | rw | rw | rw | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | CRCEN | Res. | GPIOKEN | GPIOJEN | GPIOIEN | GPIOHEN | GPIOGEN | GPIOFEN | GPIOEEN | GPIODEN | GPIOCEN | GPIOBEN | GPIOAEN |
| | | | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 6 **GPIOGEN:** IO port G clock enable

This bit is set and cleared by software.

0: IO port G clock disabled

1: IO port G clock enabled

# GPIO configuration

**Figure 25. Basic structure of a five-volt tolerant I/O port bit**

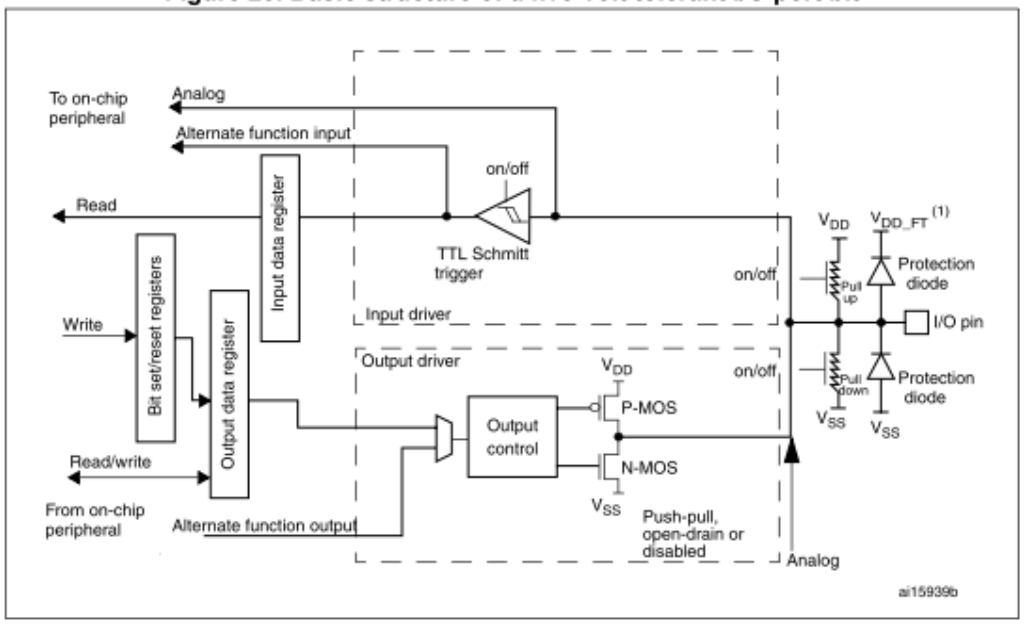



**Table 35. Port bit configuration table(1)**

| MODER(i) [1:0] | OTYPER(i) | OSPEEDR(i) [B:A] | PUPDR(i) [1:0] | | I/O configuration | |
|---|---|---|---|---|---|---|
| 01 | 0 | SPEED [B:A] | 0 | 0 | GP output | PP |
| | 0 | | 0 | 1 | GP output | PP + PU |
| | 0 | | 1 | 0 | GP output | PP + PD |
| | 0 | | 1 | 1 | Reserved | |
| | 1 | | 0 | 0 | GP output | OD |
| | 1 | | 0 | 1 | GP output | OD + PU |
| | 1 | | 1 | 0 | GP output | OD + PD |
| | 1 | | 1 | 1 | Reserved (GP output OD) | |

# GPIO configuration

- Setting port pins to output

### 8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODER15[1:0] | | MODER14[1:0] | | MODER13[1:0] | | MODER12[1:0] | | MODER11[1:0] | | MODER10[1:0] | | MODER9[1:0] | | MODER8[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MODER7[1:0] | | MODER6[1:0] | | MODER5[1:0] | | MODER4[1:0] | | MODER3[1:0] | | MODER2[1:0] | | MODER1[1:0] | | MODER0[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

# GPIO pin state manipulation

- Setting pins value to 1 or 0

### 8.4.6 GPIO port output data register (GPIOx_ODR) (x = A..I/J/K)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ODR15 | ODR14 | ODR13 | ODR12 | ODR11 | ODR10 | ODR9 | ODR8 | ODR7 | ODR6 | ODR5 | ODR4 | ODR3 | ODR2 | ODR1 | ODR0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

*Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).*

# GPIO pin state manipulation

- Setting pins value to 1 or 0

### 8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I/J/K)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BS15 | BS14 | BS13 | BS12 | BS11 | BS10 | BS9 | BS8 | BS7 | BS6 | BS5 | BS4 | BS3 | BS2 | BS1 | BS0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

*Note: If both BSx and BRx are set, BSx has priority.*

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

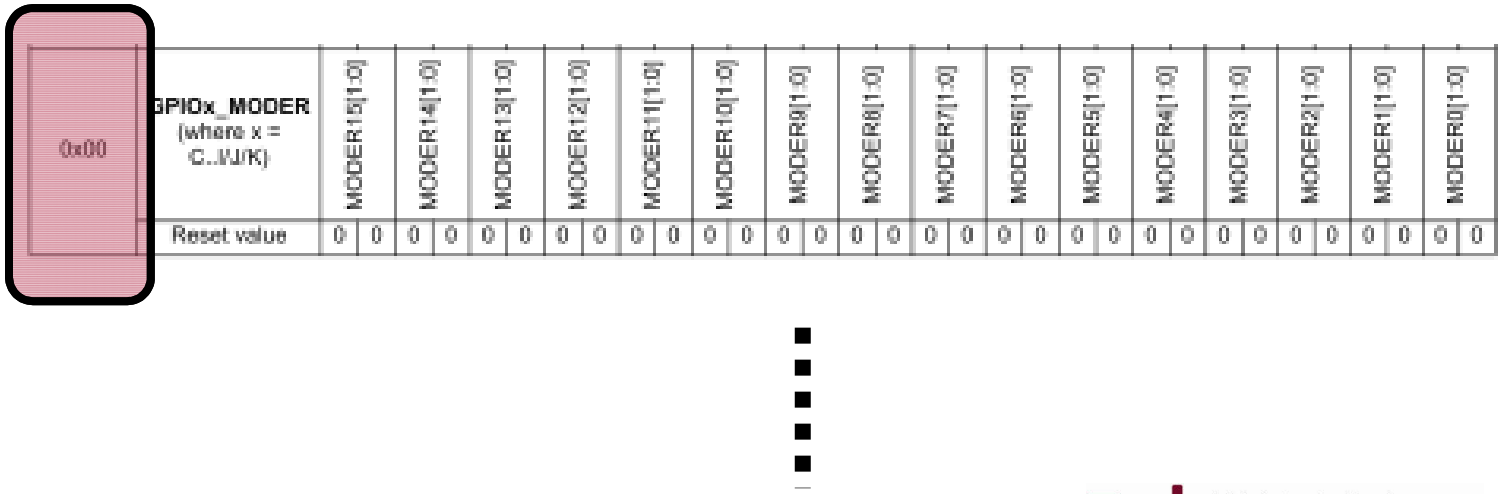
1: Sets the corresponding ODRx bit

# Register Configurations

- RCC_AHB1ENR: bit 6 -> 1
- GPIOG_MODER: bit 26,28 -> 1
- GPIOG_BSRR: bit 13,14 for setting pin to logic 1, bit 29, 30 to clearing the this pins.

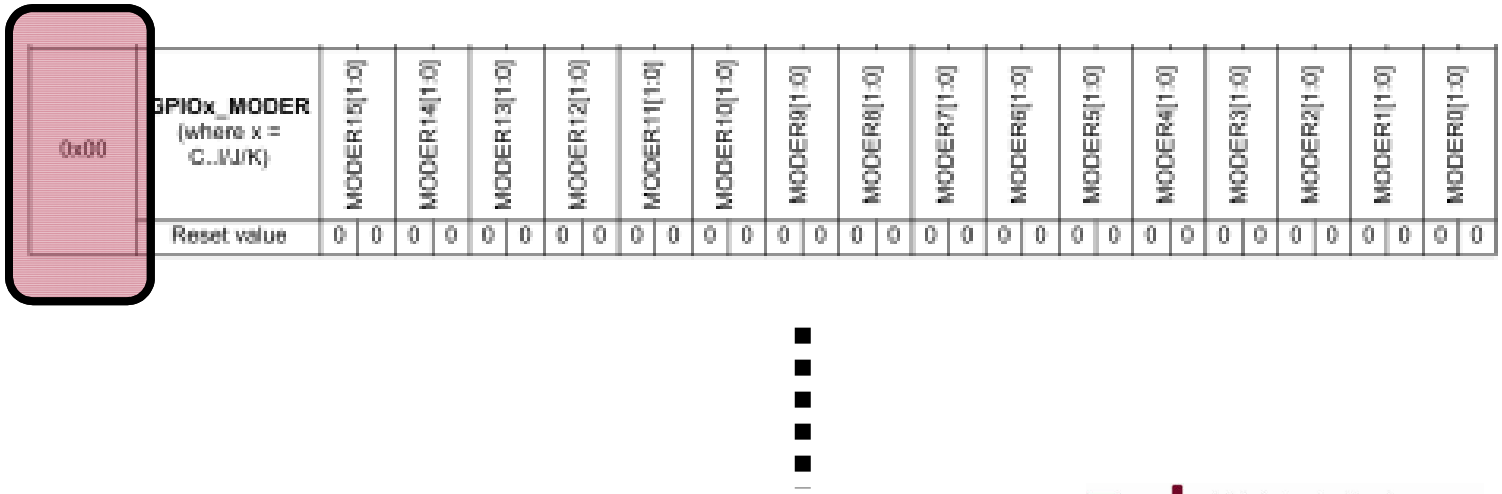
# GPIO Register Map

- Every GPIO block has many registers. But every block has the same registers
  - GPIOG starts at: 0x4002 1800



# GPIO Register Map

- Every GPIO block has many registers. But every block has the same registers
  - GPIOG starts at: 0x4002 1800
  - Old fashioned acces of GPIOG_MODER:
    - #define GPIOG_MODER  *((uint32_t*)(0x400218000))



# CMSIS definitions, and rules

- Using a structure for register blocks

```
typedef struct
{
  __IO uint32_t MODER;    /*!< GPIO port mode register,                Address offset: 0x00      */
  __IO uint32_t OTYPER;   /*!< GPIO port output type register,         Address offset: 0x04      */
  __IO uint32_t OSPEEDR;  /*!< GPIO port output speed register,        Address offset: 0x08      */
  __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register,   Address offset: 0x0C      */
  __IO uint32_t IDR;      /*!< GPIO port input data register,          Address offset: 0x10      */
  __IO uint32_t ODR;      /*!< GPIO port output data register,         Address offset: 0x14      */
  __IO uint16_t BSRRL;    /*!< GPIO port bit set/reset low register,   Address offset: 0x18      */
  __IO uint16_t BSRRH;    /*!< GPIO port bit set/reset high register, Address offset: 0x1A      */
  __IO uint32_t LCKR;     /*!< GPIO port configuration lock register, Address offset: 0x1C      */
  __IO uint32_t AFR[2];   /*!< GPIO alternate function registers,      Address offset: 0x20-0x24 */
} GPIO_TypeDef;
```

# CMSIS definitions, and rules

- Pre defined memory aliases

```
#define GPIOA                ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOF                ((GPIO_TypeDef *) GPIOF_BASE)
#define GPIOG                ((GPIO_TypeDef *) GPIOG_BASE)
#define GPIOH                ((GPIO_TypeDef *) GPIOH_BASE)
#define GPIOI                ((GPIO_TypeDef *) GPIOI_BASE)
#define GPIOJ                ((GPIO_TypeDef *) GPIOJ_BASE)
#define GPIOK                ((GPIO_TypeDef *) GPIOK_BASE)
```

- Therefore accessing the GPIOG_MODER
  - GPIOG->MODER