Timer and GPIO

Lecture 11

Yeongpil Cho

Hanynag University

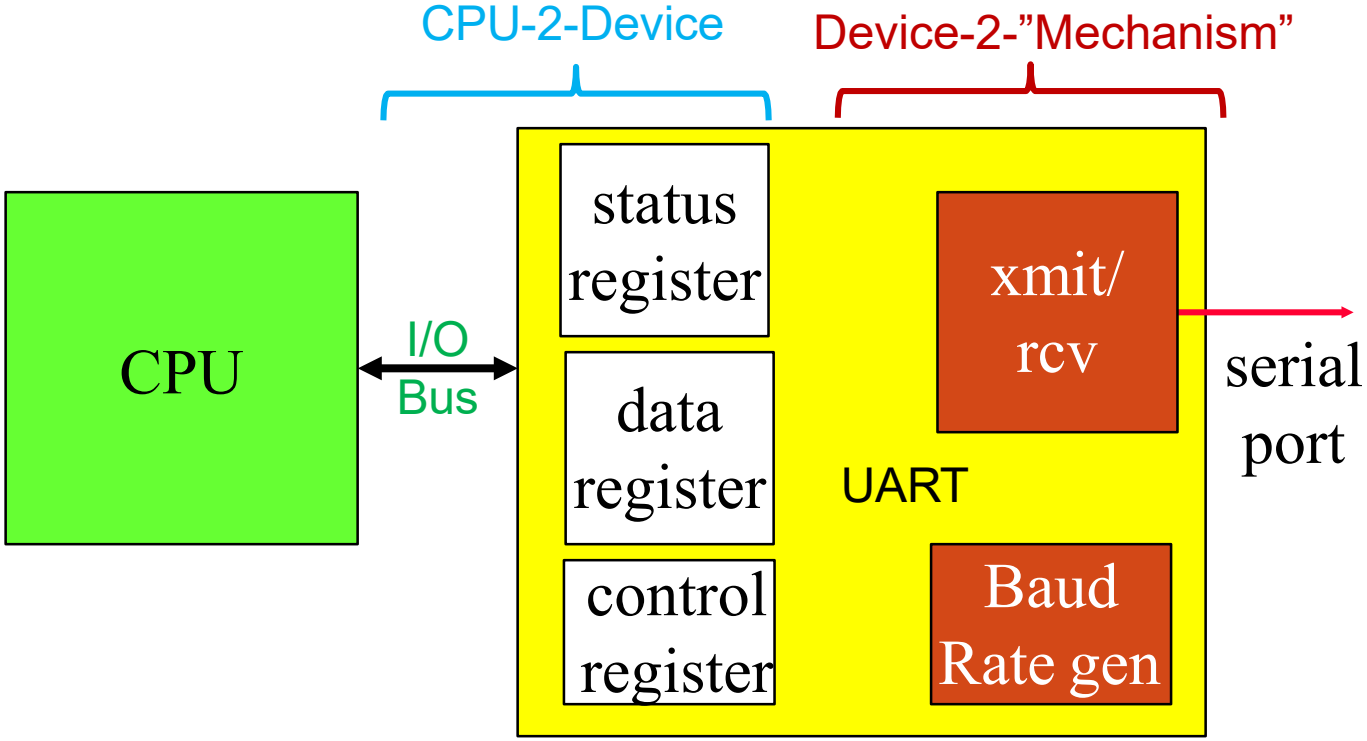
Topics

- Interfacing Peripherals
- Timer
- GPIO

Interfacing Peripherals

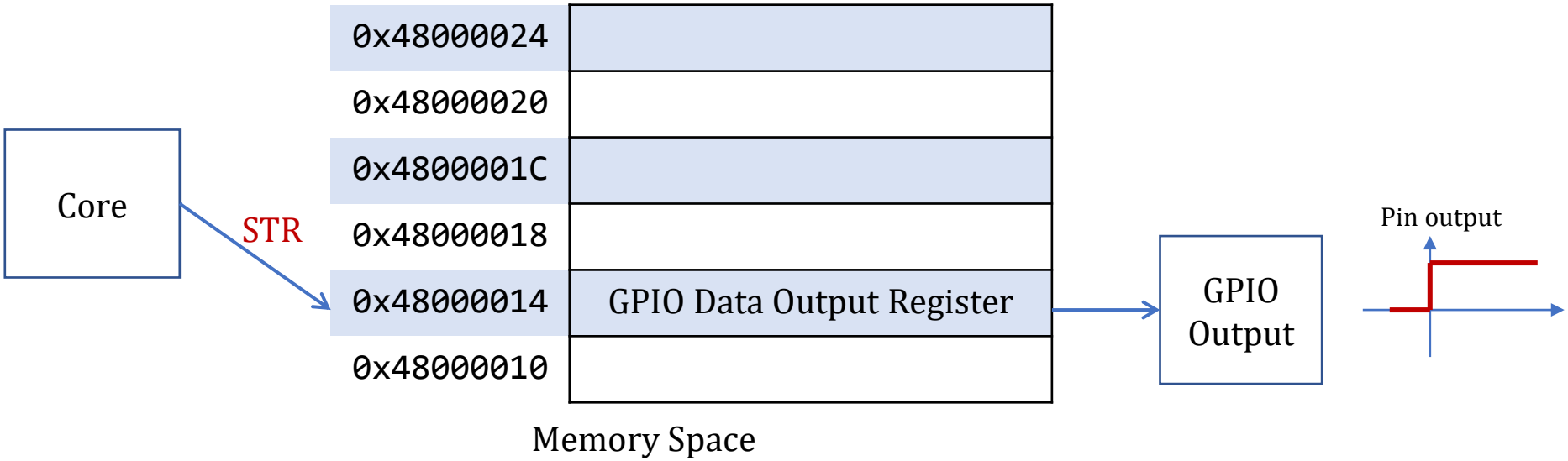
I/O devices

- "Devices" consist of two parts
  - CPU - to - Device, Device – to - I/O Mechanism
- Example : UART device
  - CPU to/from device via register read/write
  - I/O "mechanism" effectively transparent to CPU



Interfacing Peripherals

- Port-mapped I/O
  - Use special CPU instructions: `Special_instruction Reg, Port`
- Memory-mapped I/O
  - A simpler and more convenient way to interface I/O devices
  - Each device registers is assigned to a memory address in the address sp ace of the microprocessor
  - Use native CPU load/store instructions: `LDR/STR Reg, [Reg, #imm]`



ARM memory-mapped I/O

- Define location(address) for device:

```
.equ DEV1, 0x40010000
```

- Read/write assembly code:

```
LDR   r1,=DEV1   ; set up device address
LDRB  r0,[r1]    ; read byte from DEV1
MOV   r0,#8      ; set up value to write
STRB  r0,[r1]    ; write value to device
```

- Equivalent C code:

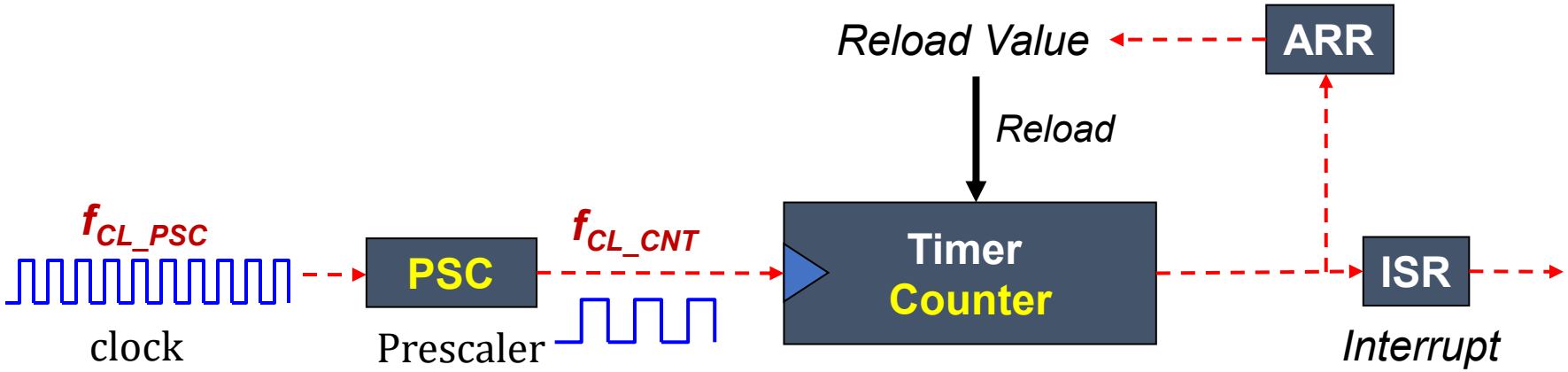
```
Var1 = *(char*)DEV1;     // read from DEV1 to variable
*(char*)DEV1 = Var1;     // write variable to DEV1
```

Timer

Timer

- Free-run counter (independently of processor)
- Functions
  - Input capture
  - Output compare
  - Pulse-width modulation (PWM) generation

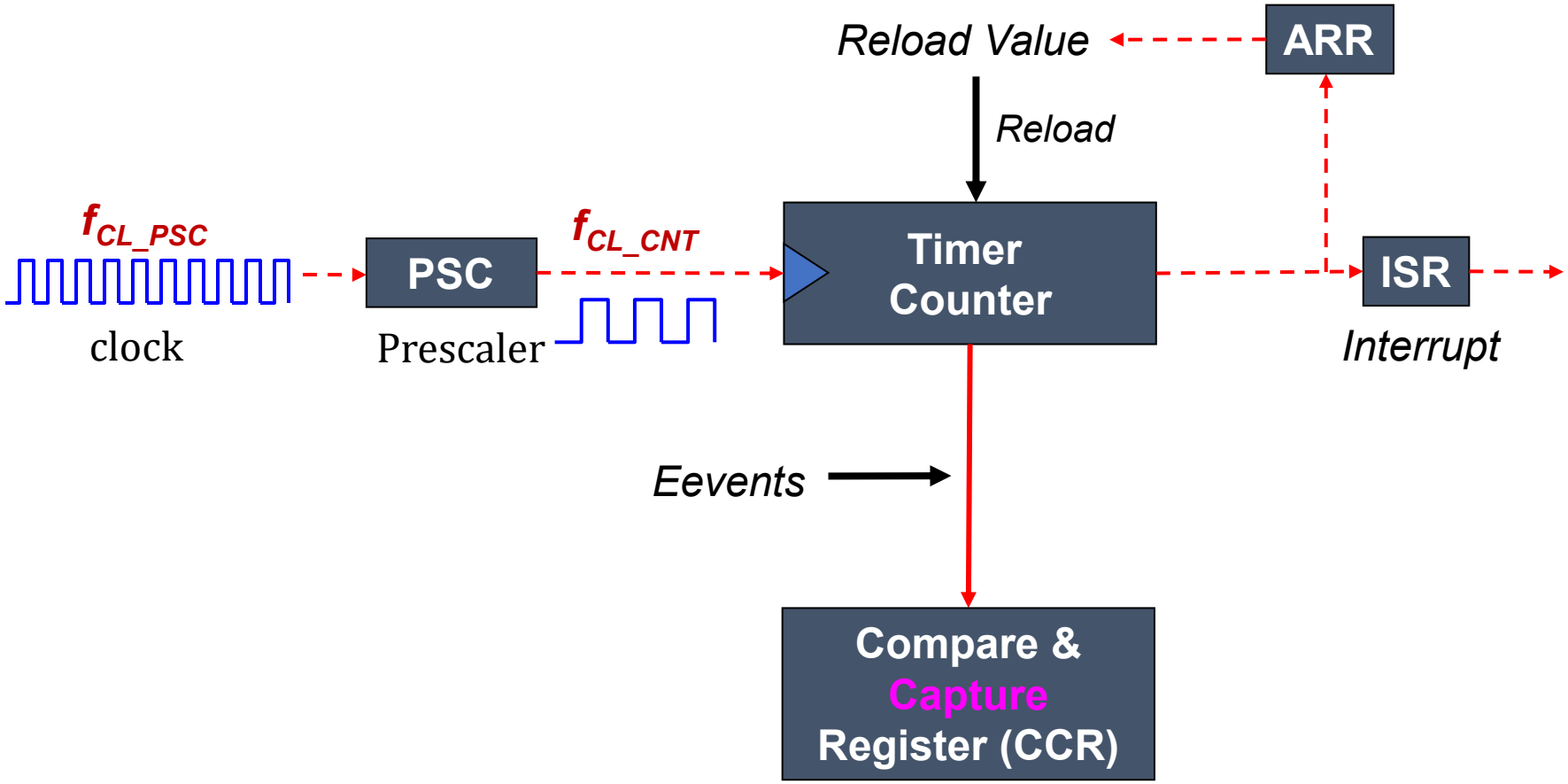
Timer: Clock



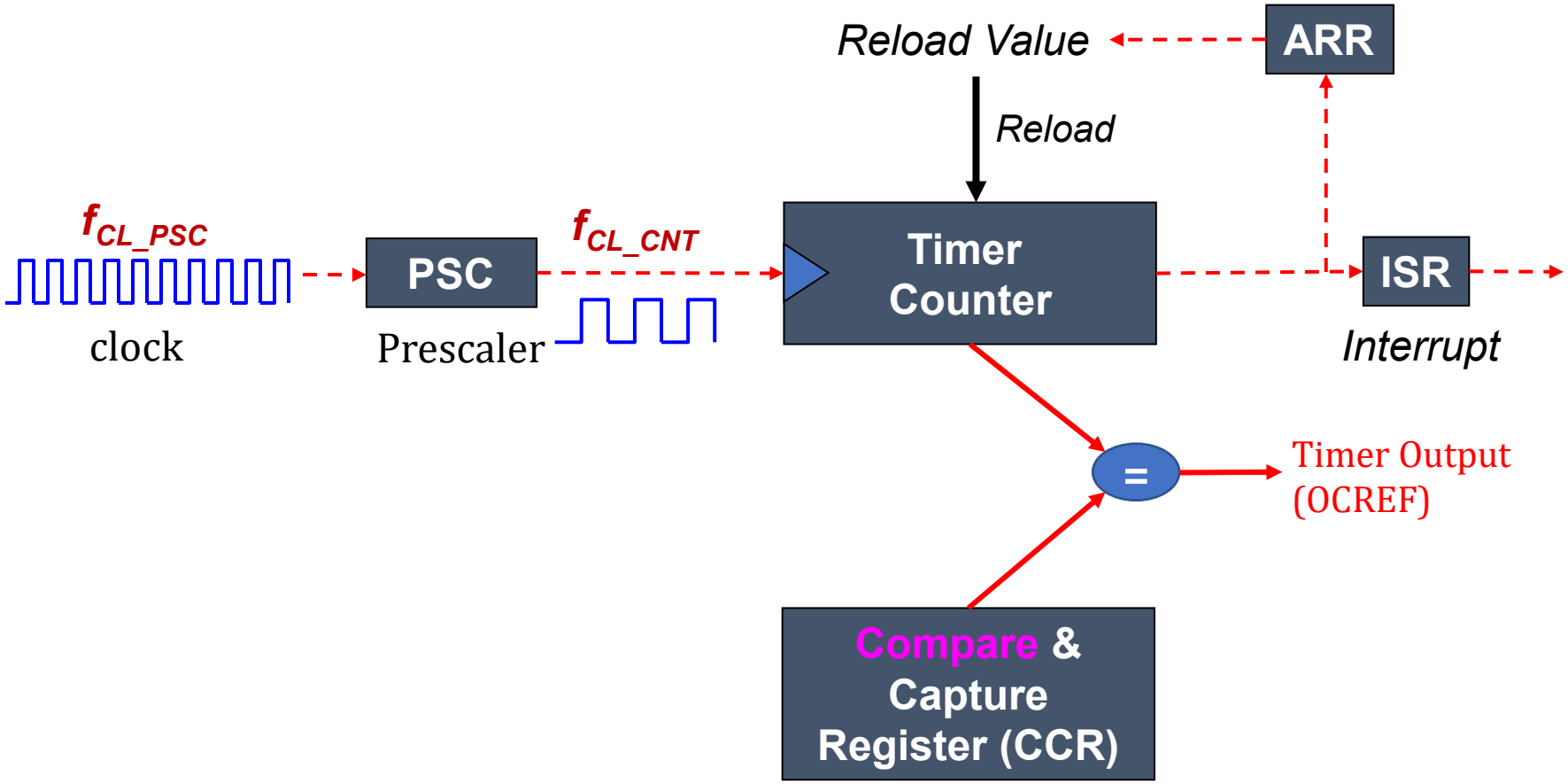
$$f_{CK\_CNT} = \frac{f_{CL\_PSC}}{PSC + 1}$$

PSC: Prescaler
ARR: Auto-Reload Register

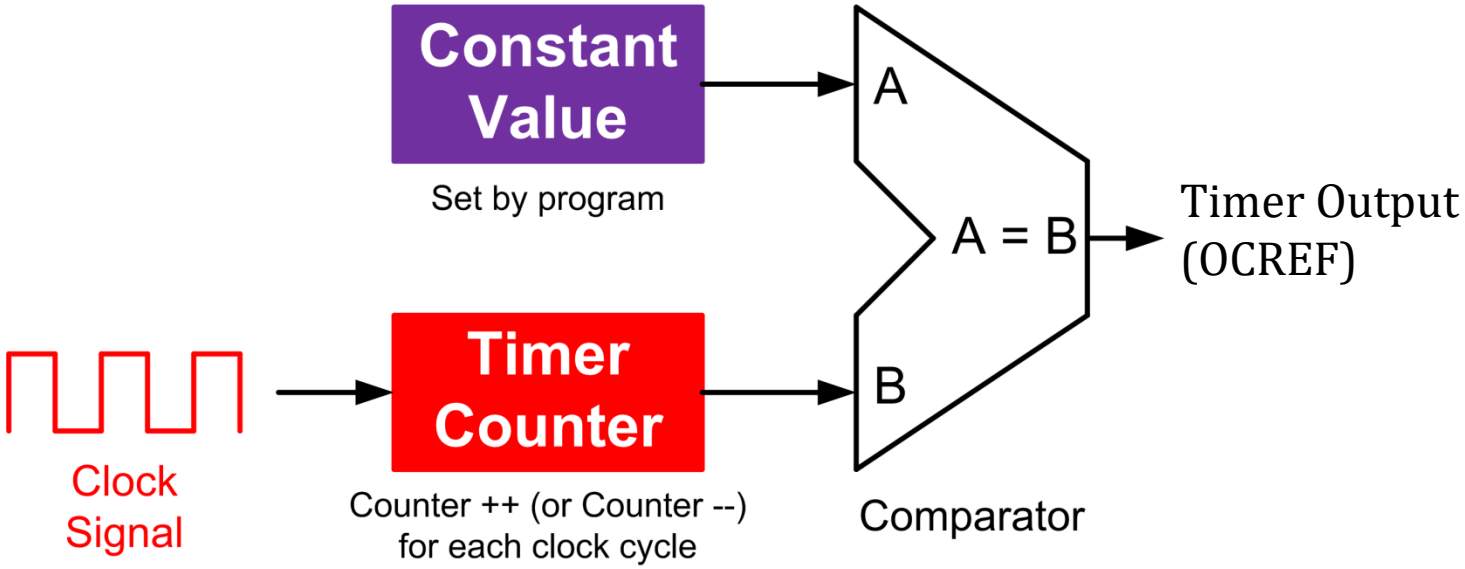
Input Capture



Output Compare

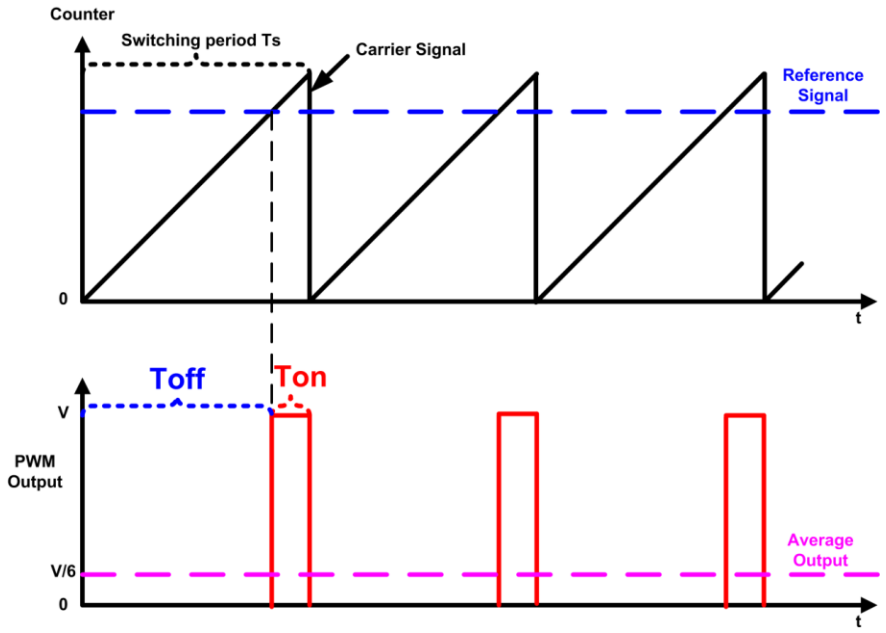


Output Compare



Output Compare Mode (OCM)	Timer Output (OCREF)
000	Frozen
001	High if CNT == CCR
010	Low if CNT == CCR
011	Toggle if CNT == CCR
100	Forced low (always low)
101	Forced high (always high)

PWM (Pulse-Width Modulation) Generation



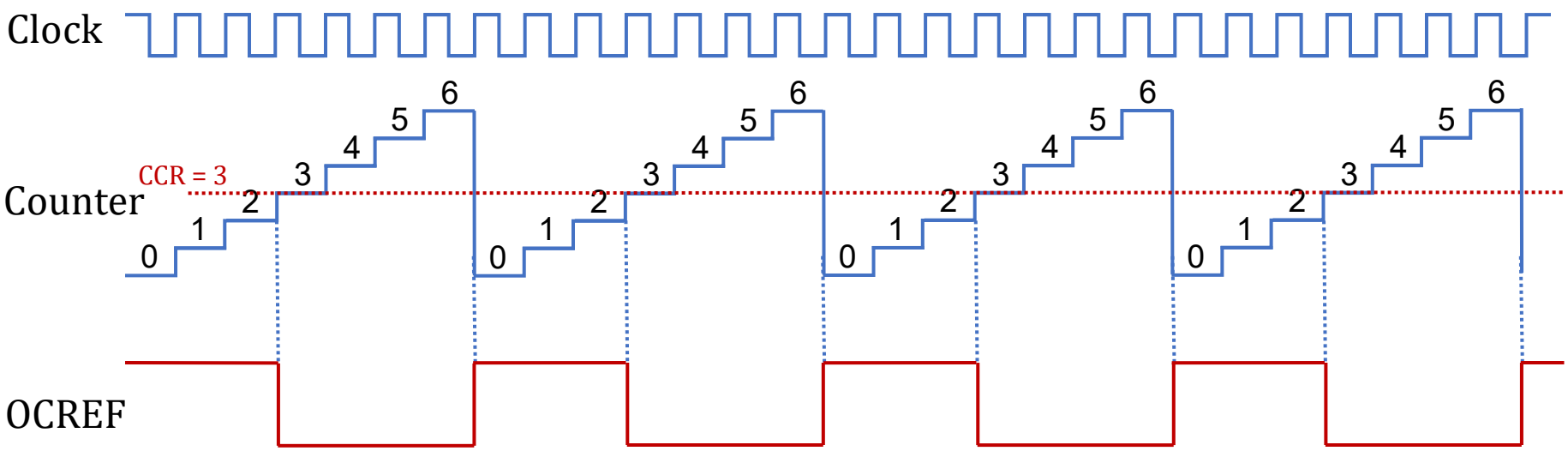
Mode	Counter < Reference	Counter ≥ Reference
PWM mode 1 (Low True)	Active	Inactive
PWM mode 2 (High True)	**Inactive**	**Active**

PWM Mode 1 (Low-True)

Mode 1

Timer Output = { High if counter < CCR ; Low if counter ≥ CCR }

Upcounting, ARR = 6, CCR = 3



$$\text{Duty Cycle} = \frac{\text{CCR}}{\text{ARR} + 1} = \frac{3}{7}$$

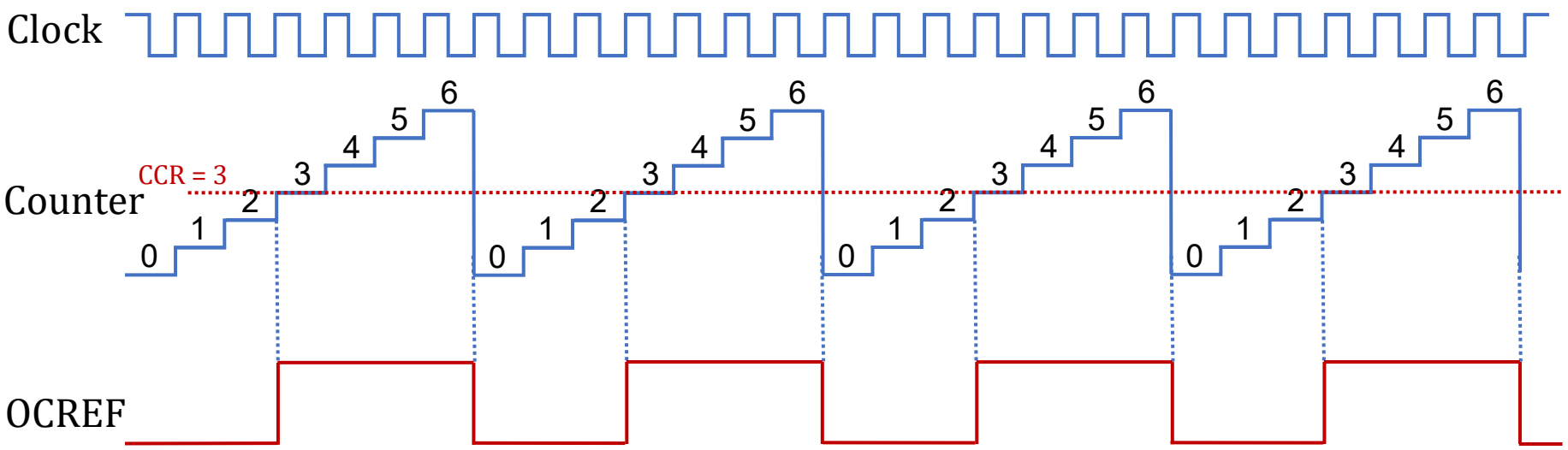
$$\text{Period} = (1 + \text{ARR}) * \text{Clock Period} = 7 * \text{Clock Period}$$

PWM Mode 2 (High-True)

Mode 2

Timer Output =
- Low if counter < CCR
- High if counter ≥ CCR

Upcounting, ARR = 6, CCR = 3



$$\text{Duty Cycle} = 1 - \frac{CCR}{ARR + 1} = \frac{4}{7}$$

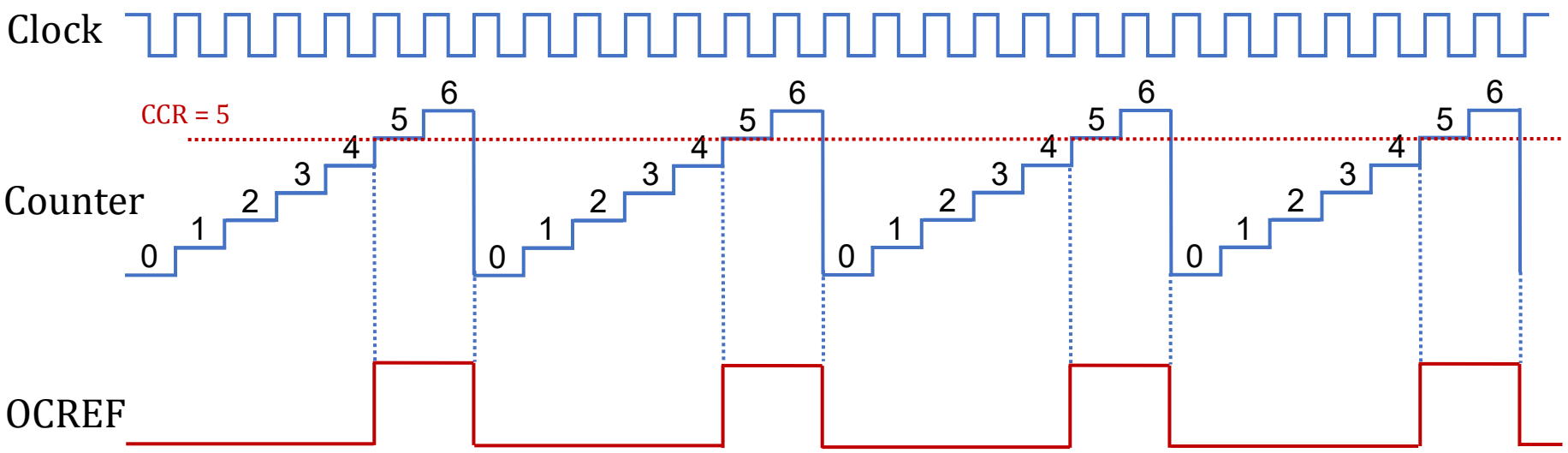
$$\text{Period} = (1 + ARR) * \text{Clock Period} = 7 * \text{Clock Period}$$

PWM Mode 2 (High-True)

Mode 2

Timer Output =
- Low if counter < CCR
- High if counter ≥ CCR

Upcounting, ARR = 6, CCR = 5

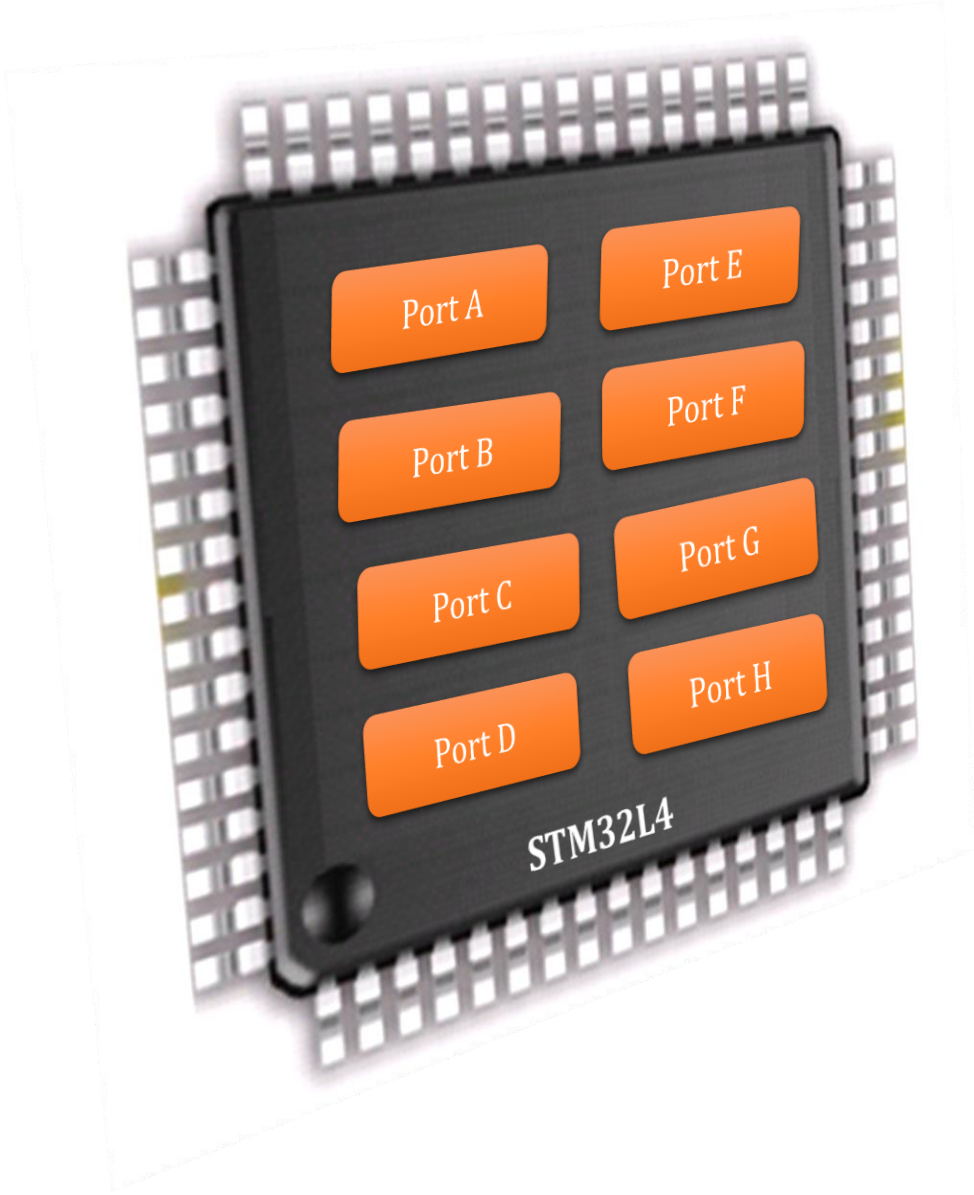


$$\text{Duty Cycle} = 1 - \frac{CCR}{ARR + 1} = \frac{2}{7}$$

$$\text{Period} = (1 + ARR) * \text{Clock Period} = 7 * \text{Clock Period}$$

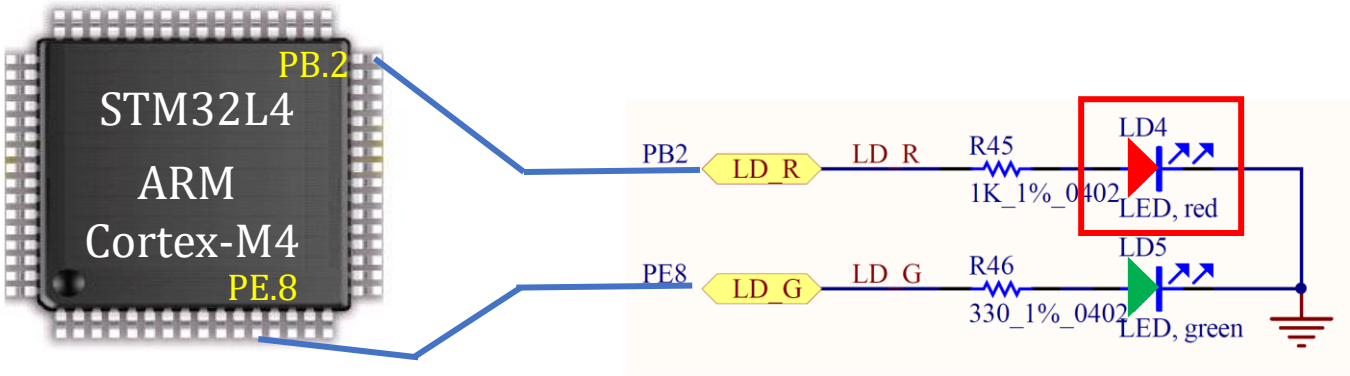
GPIO

General Purpose Input/Output (GPIO)



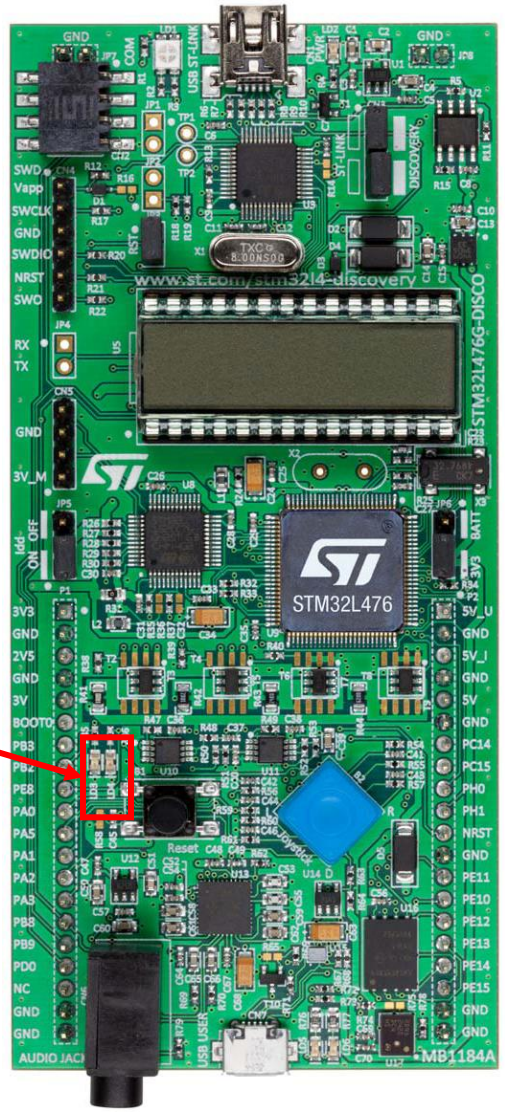
- 8 GPIO Ports:
  A, B, C, D, E, F, G, H
- Up to 16 pins in each port

Red LED (PB.2)

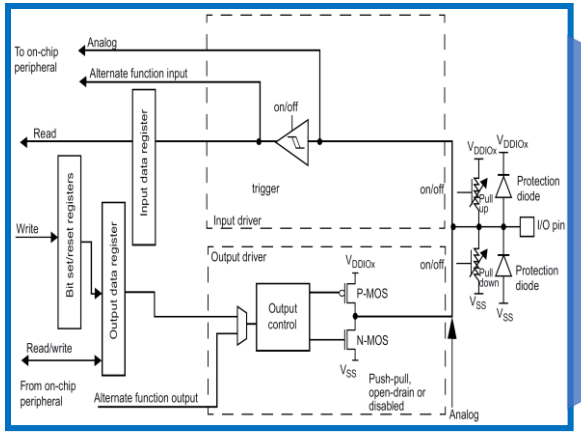


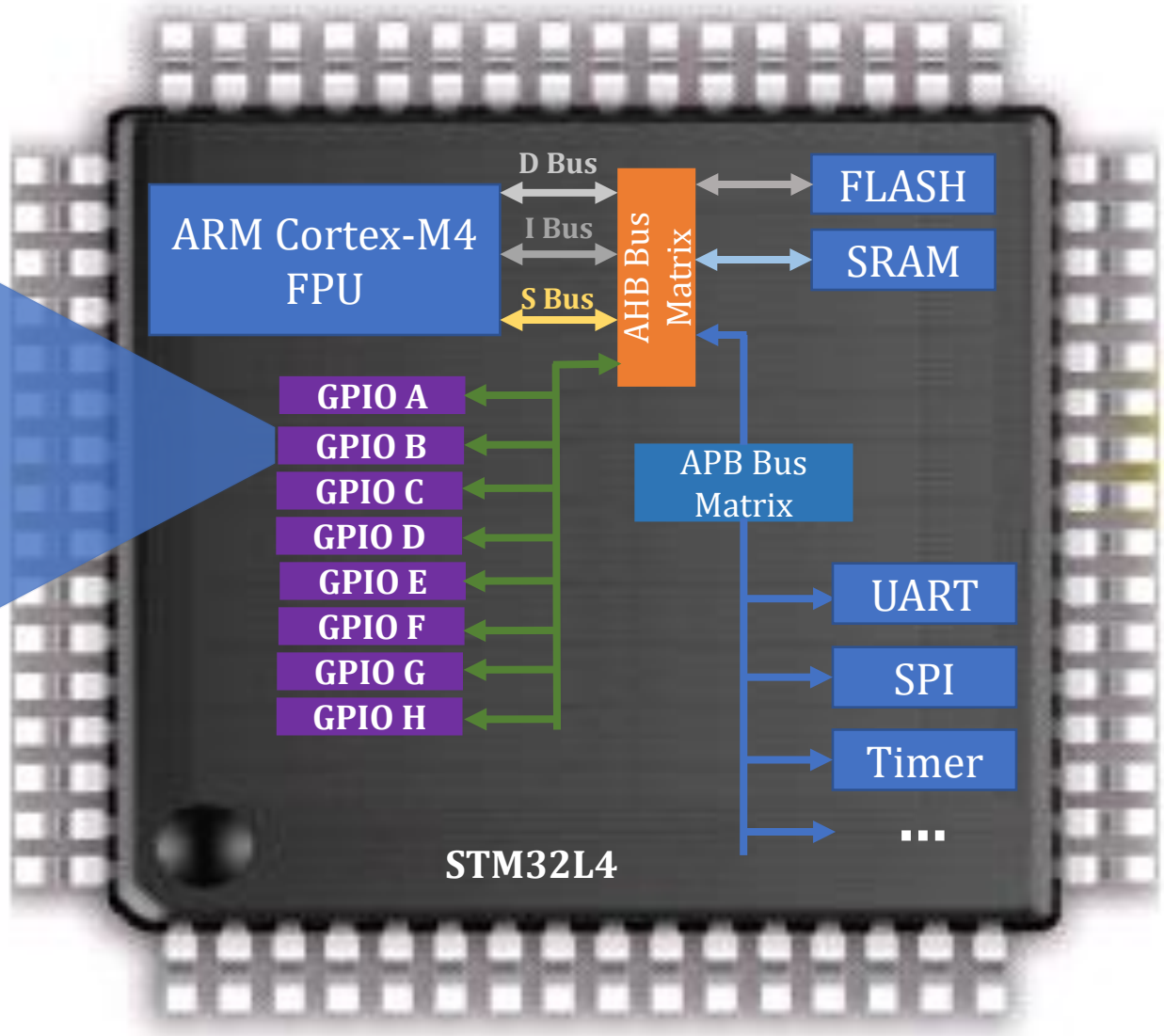
PB.2	Red LED
High	On
Low	Off

STM32L4 Discovery Kit

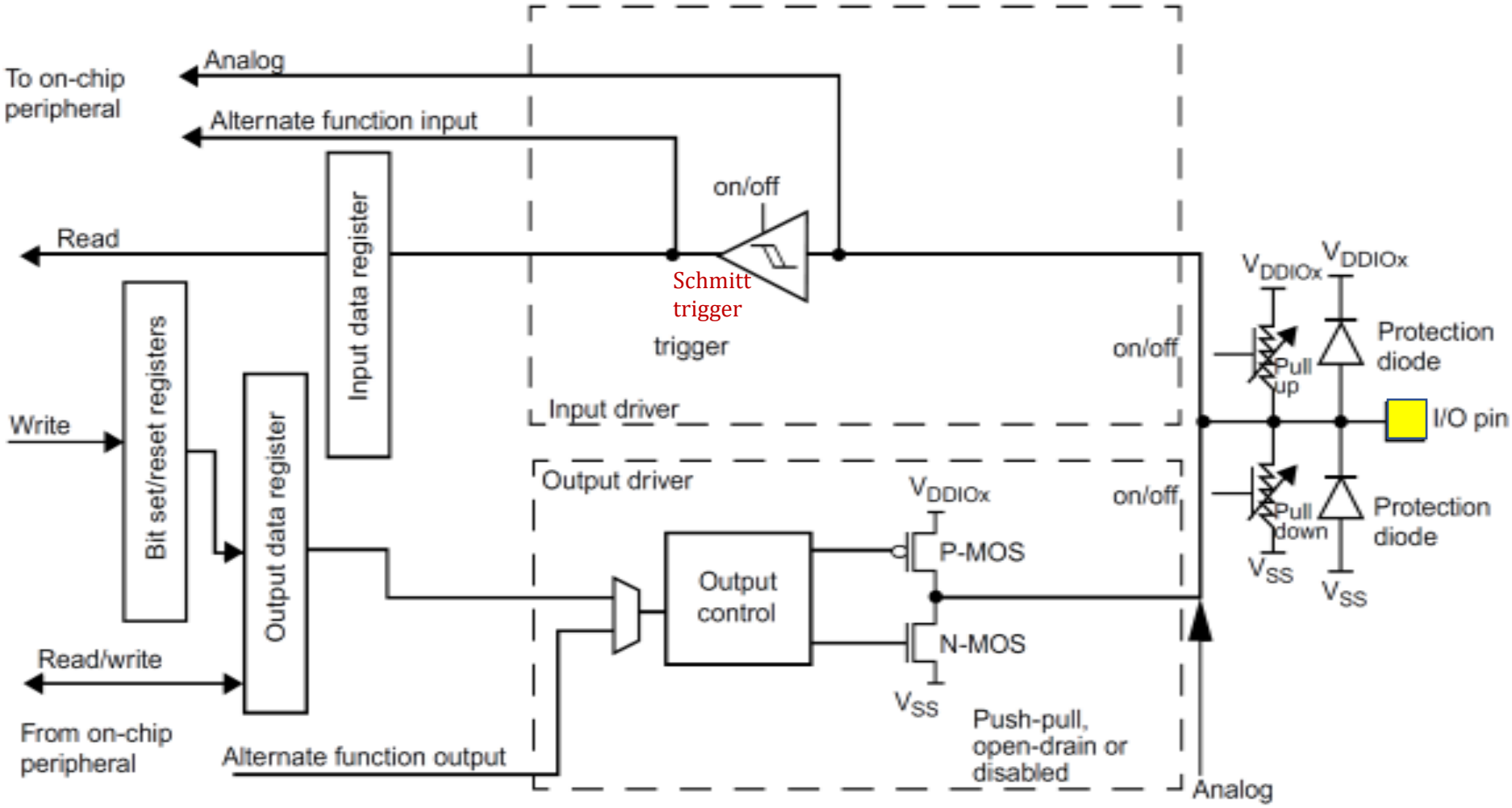


GPIO Ports

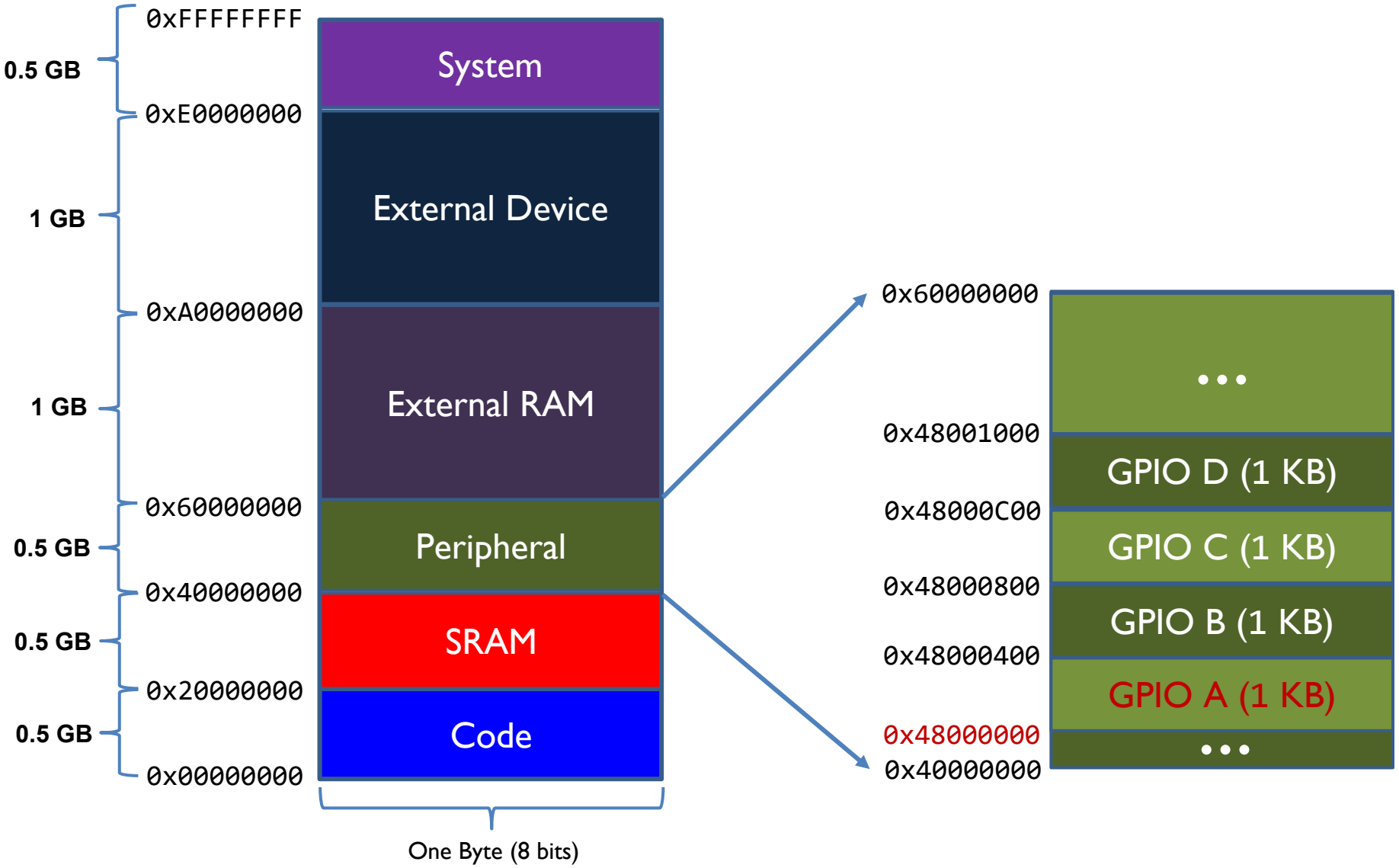

To on-chip peripheral
Analog
Alternate function input
on/off
Read
Input data register
trigger
Input driver
Write
Bit set/reset registers
Output data register
Output driver
Output control
P-MOS
N-MOS
Read/write
From on-chip peripheral
Alternate function output
Push-pull, open-drain or disabled
Protection diode
I/O pin
Analog


ARM Cortex-M4
FPU
D Bus
I Bus
S Bus
AHB Bus Matrix
FLASH
SRAM
GPIO A
GPIO B
GPIO C
GPIO D
GPIO E
GPIO F
GPIO G
GPIO H
APB Bus Matrix
UART
SPI
Timer
...
STM32L4

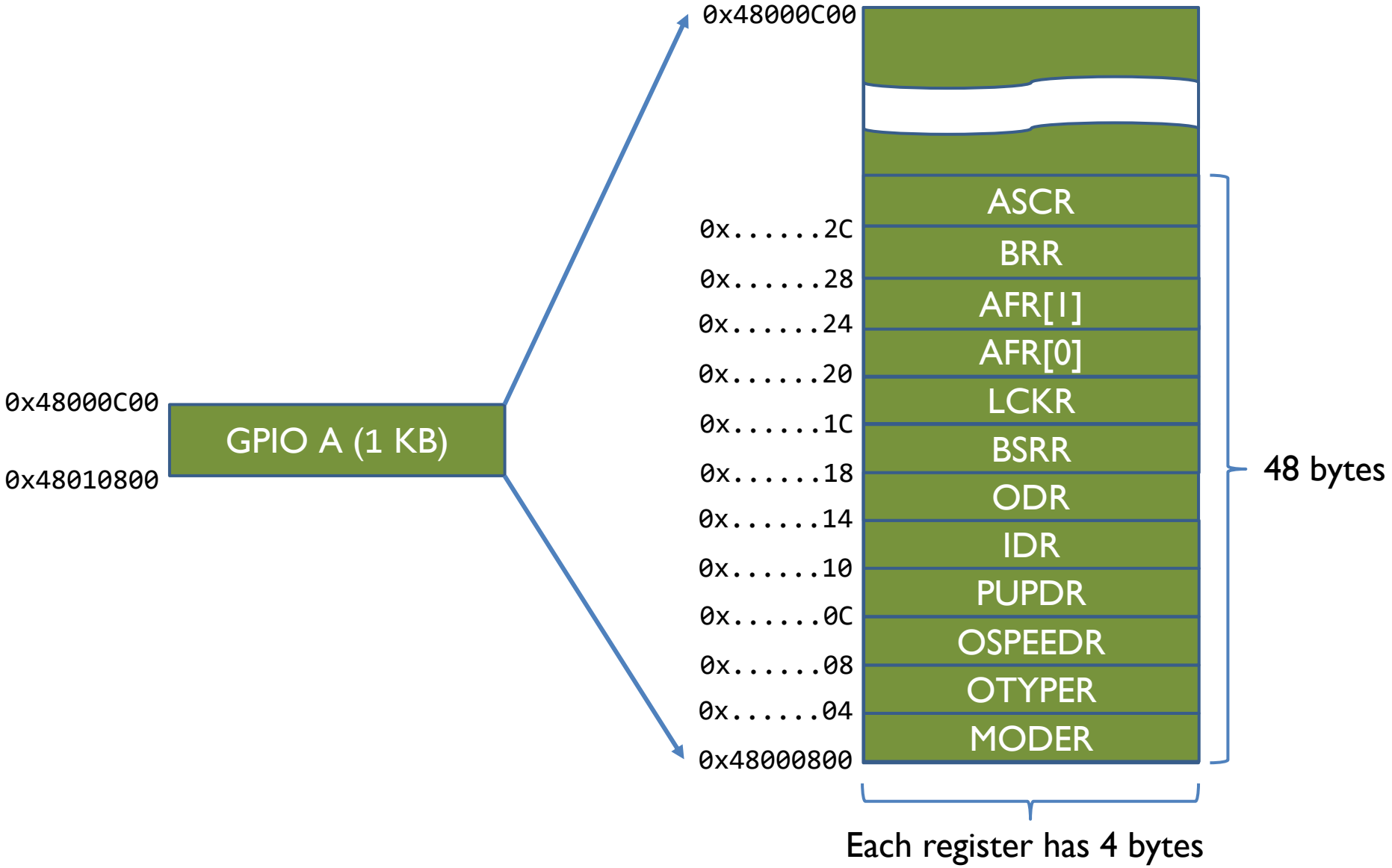
Basic Structure of a GPIO Port Pin
Input and Output



GPIO Memory Map (STM32L4)



GPIO Memory Map (STM32L4)

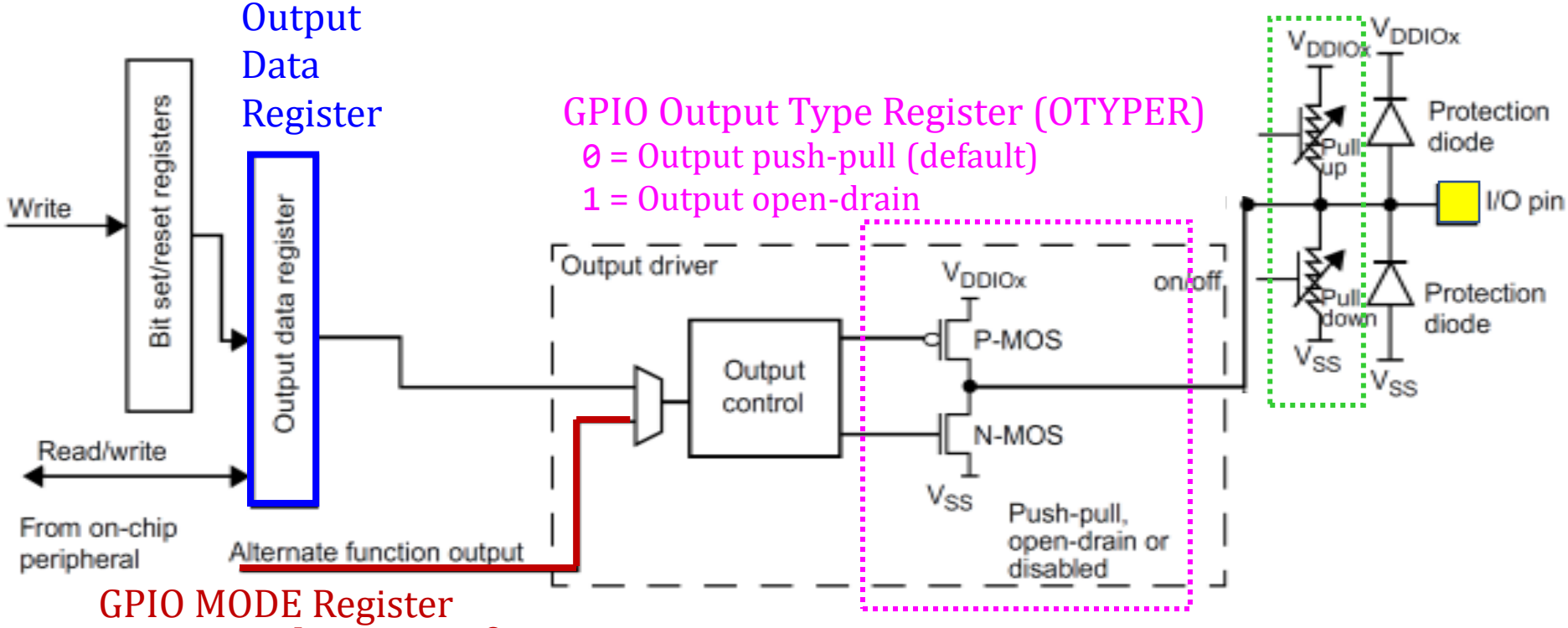


Basic Structure of an GPIO Port Pin:
Output

GPIO Pull-up/Pull-down Register (PUPDR)
`00` = No pull-up, pull-down `01` = Pull-up
`10` = Pull-down `11` = Reserved

Output
Data
Register

GPIO Output Type Register (OTYPER)
`0` = Output push-pull (default)
`1` = Output open-drain



GPIO MODE Register
`00` = Input, `01` = Output,
`10` = AF, `11` = Analog (default)

GPIO Mode Register (MODER)

- 32 bits (16 pins, 2 bits per pin)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[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
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Pin 2 Pin 1 Pin 0

Bits 2y+1:2y **MODEy[1:0]:** Port x configuration bits (y = 0..15)

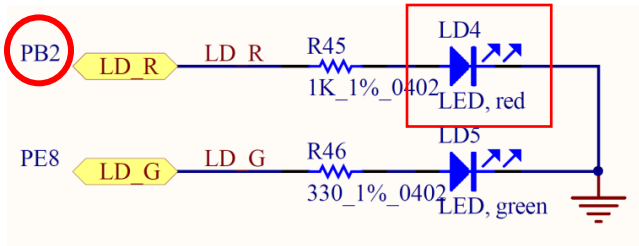
These bits are written by software to configure the I/O mode.

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)



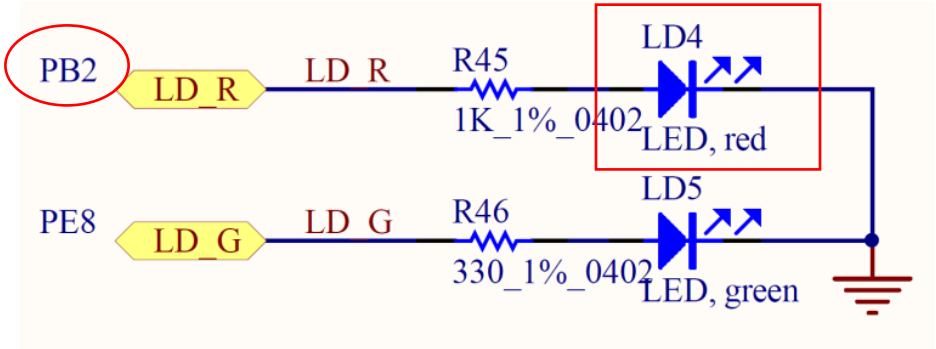

```
GPIOB->MODER &= ~(3UL<<4);  // Clear bits 4 and 5 for Pin 2
GPIOB->MODER |=   1UL<<4;   // Set bit 4, set Pin 2 as output
```

GPIO Output Data Register (ODR)

- 16 bits reserved, 16 data bits, 1 bit for each pin

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Pin 2




```
GPIOB->ODR |= 1UL << 2;     // Set bit 2
```

GPIO Output Type Register (OTYPE)

- 16 bits reserved, 16 data bits, 1 bit for each pin

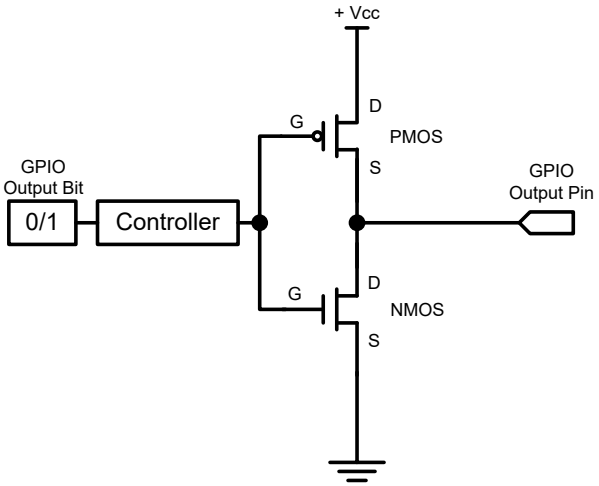
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **OTy:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output type.

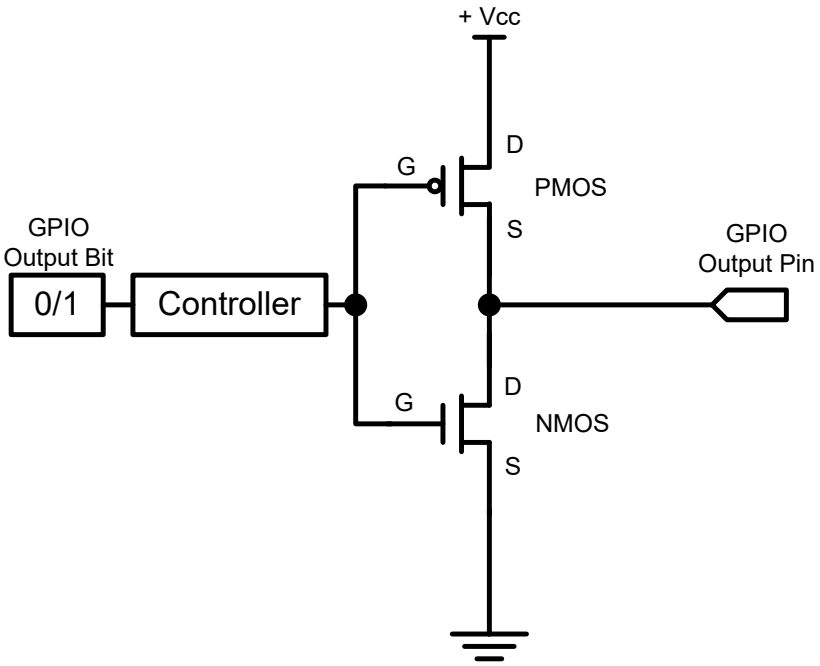
0: Output push-pull (reset state)

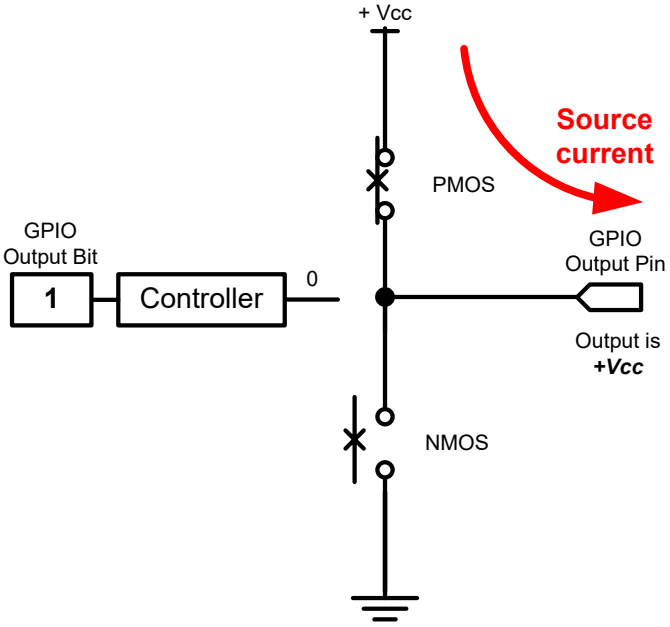
1: Output open-drain




```
GPIOB->OTYPE &= ~(1UL<<2);  // Clear bit 2
```

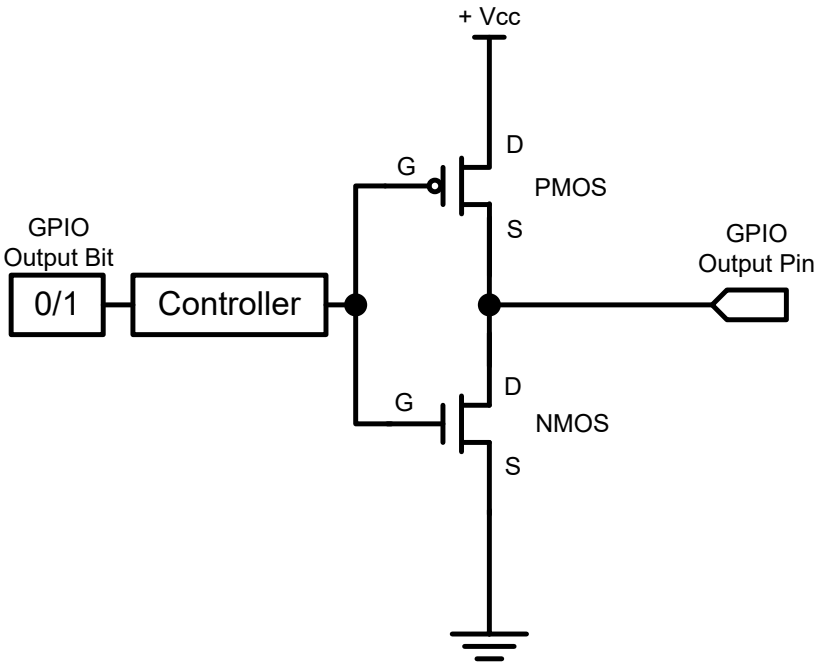
GPIO Output: Push-Pull

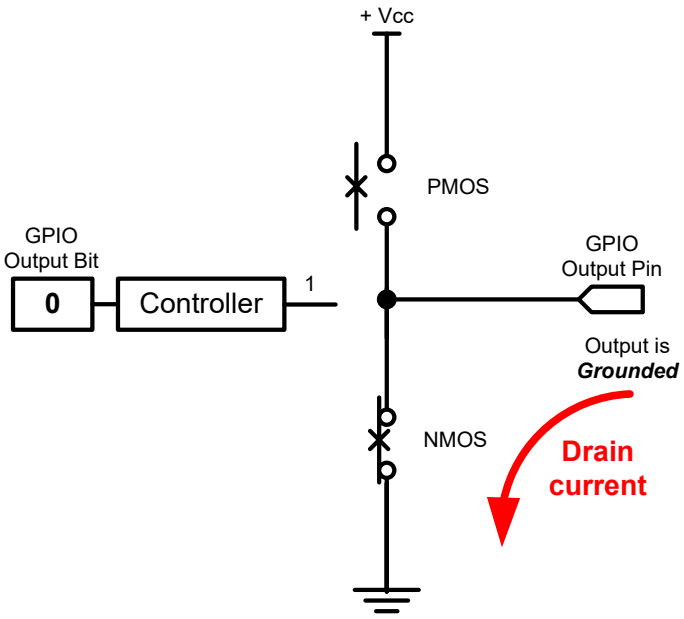




GPIO Output = 1
Source current to external circuit

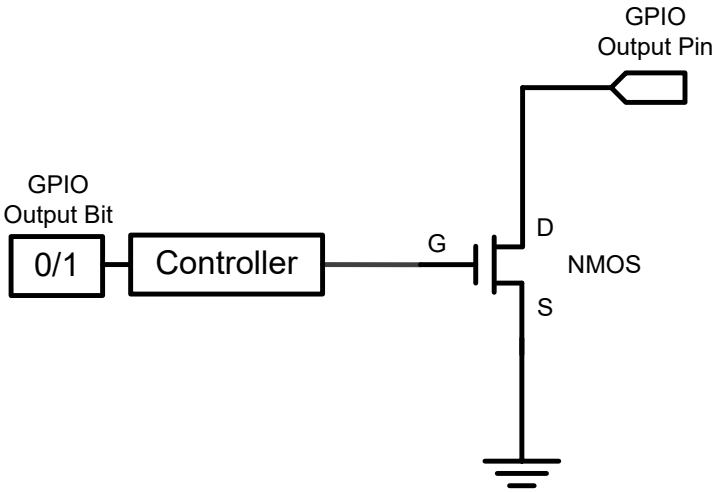
GPIO Output: Push-Pull

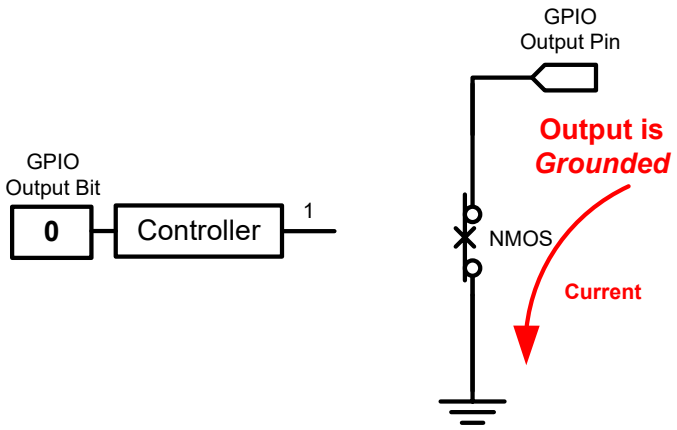




GPIO Output = 0
Drain current from external circuit

GPIO Output: Open-Drain

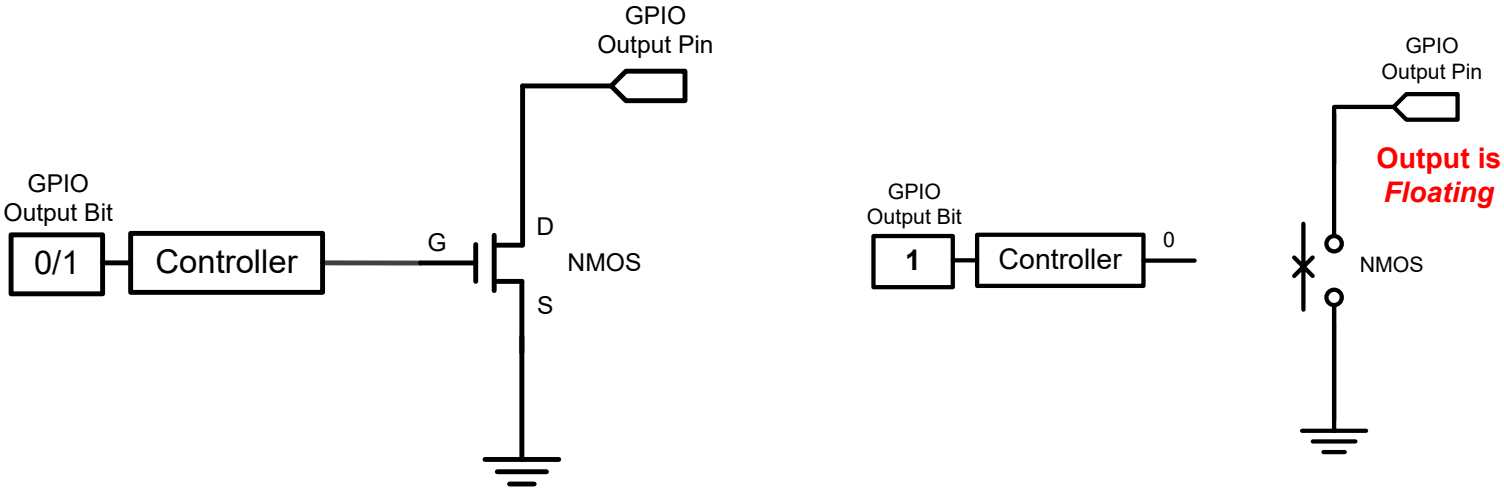




GPIO Output = 0
Drain current from external circuit

GPIO Output:
Open-Drain

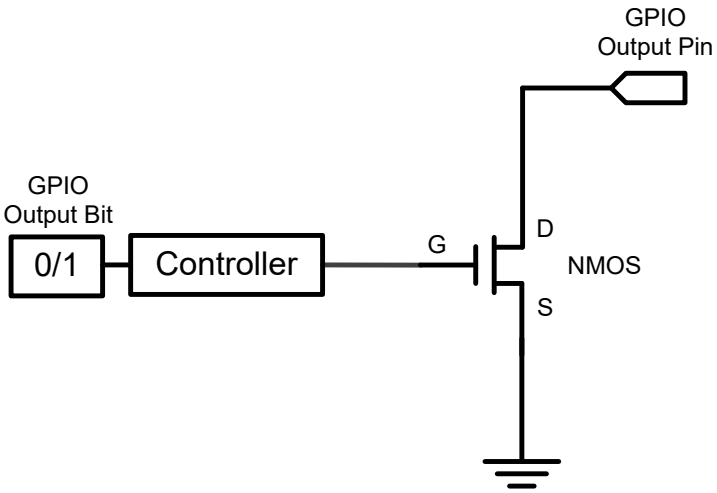
- Output is floating (also called High-Impedance, tri-stated, HiZ)
- An external pull-up resister is needed for a high signal

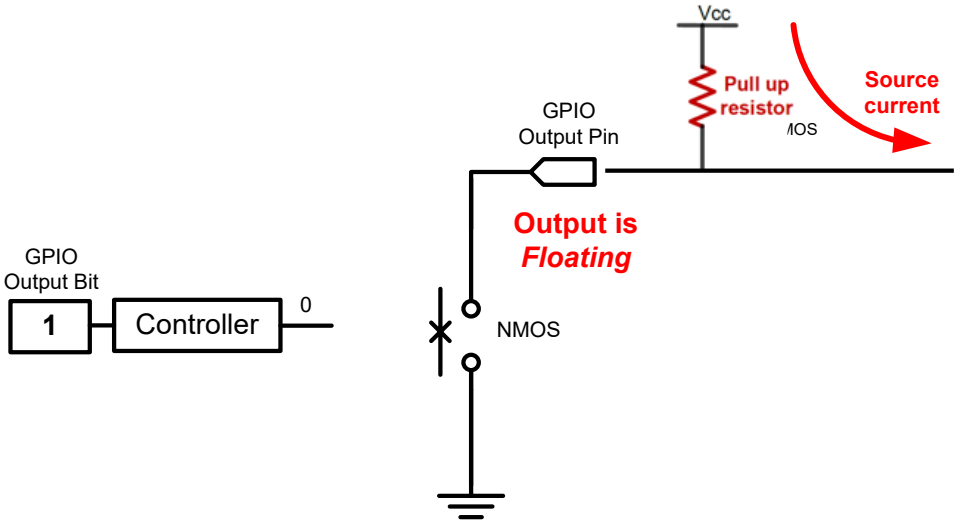


Output = 1
GPIO Pin has high-impedance to external circuit

GPIO Output: Open-Drain

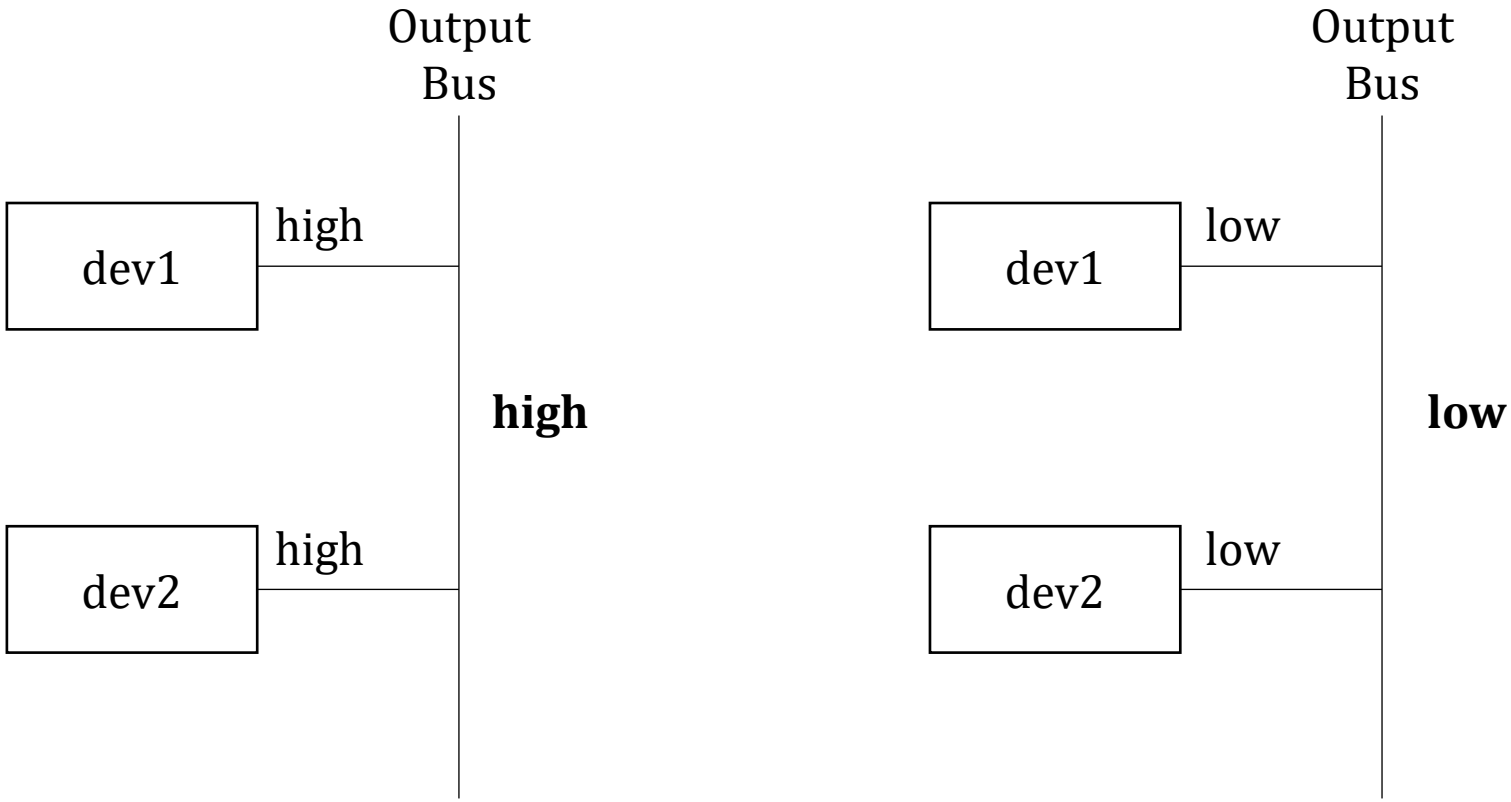
- Output is floating (also called High-Impedance, tri-stated, HiZ)
- An external pull-up resister is needed for a high signal



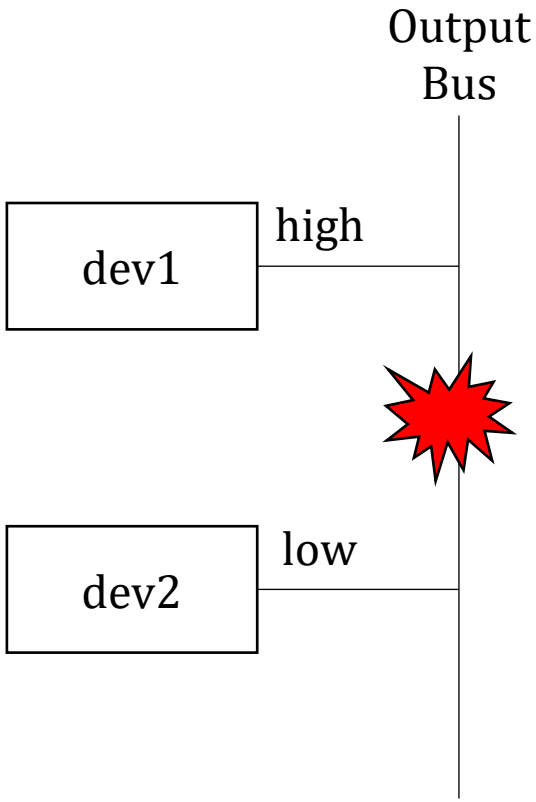


Output = 1
GPIO Pin has high-impedance to external circuit

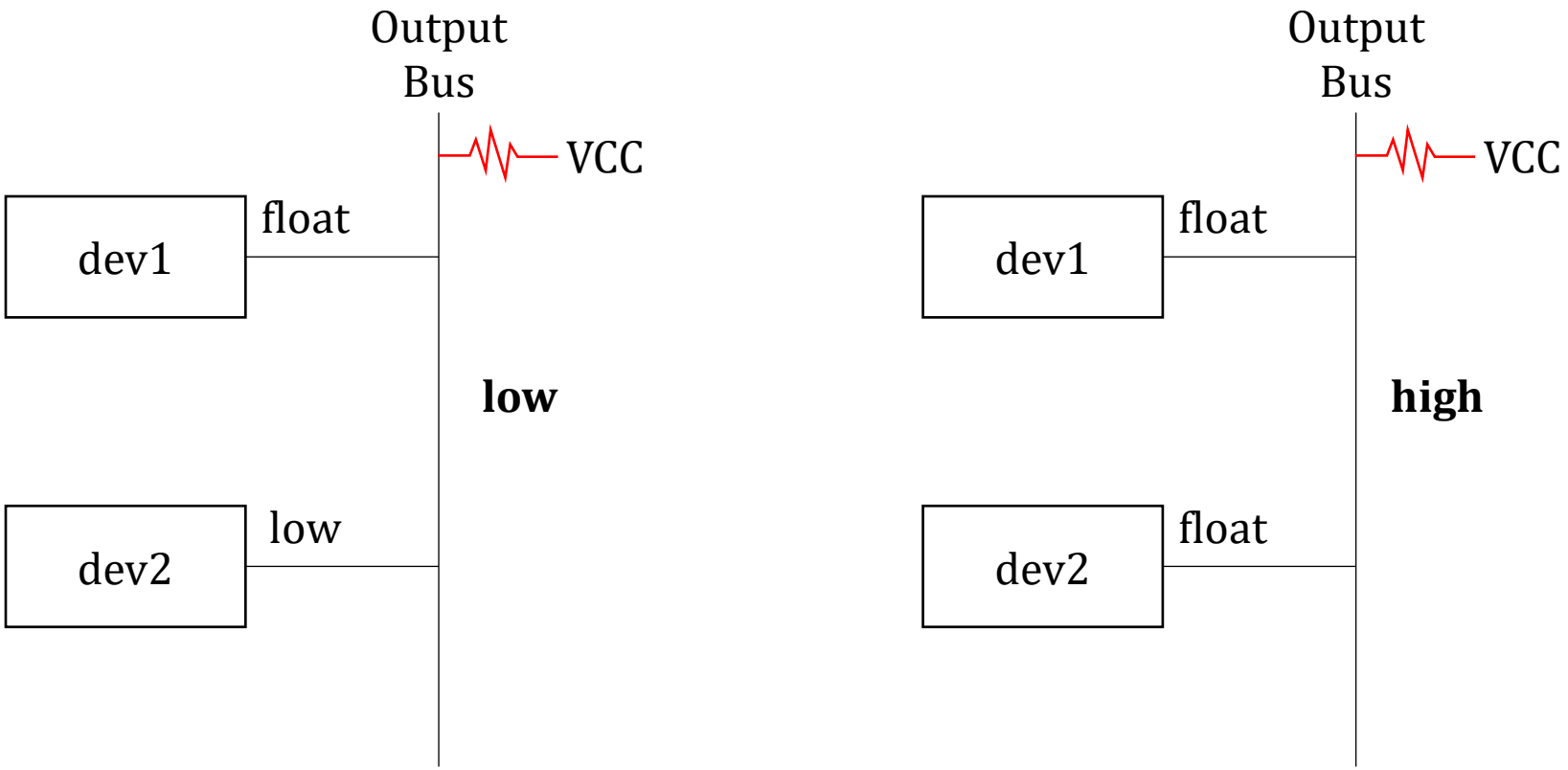
GPIO Output: Open-Drain



GPIO Output: Open-Drain



GPIO Output: Open-Drain



GPIO Pull-up/Pull-down Register (PUPDR)

- 16 pins per port, 2 bits per pin

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[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
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

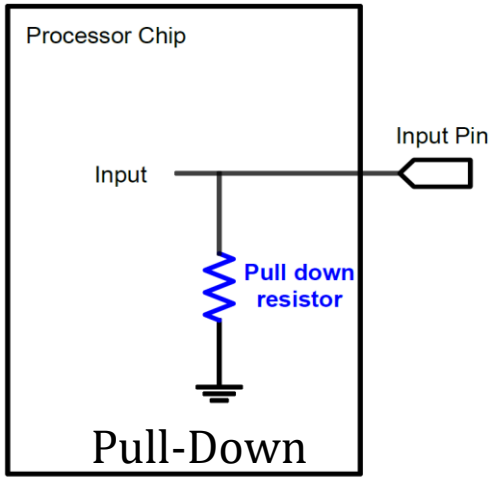
Pin 2 Pin 1 Pin 0

Bits 2y+1:2y **PUPDy[1:0]:** Port x configuration bits (y = 0..15)

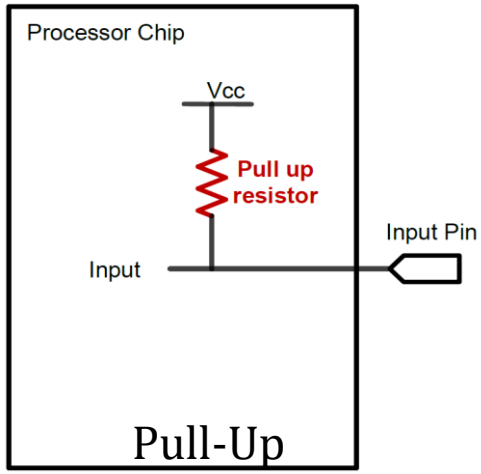
These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

```
// No pull-up, pull-down
GPIOA->PUPDR &= ~3UL;
```

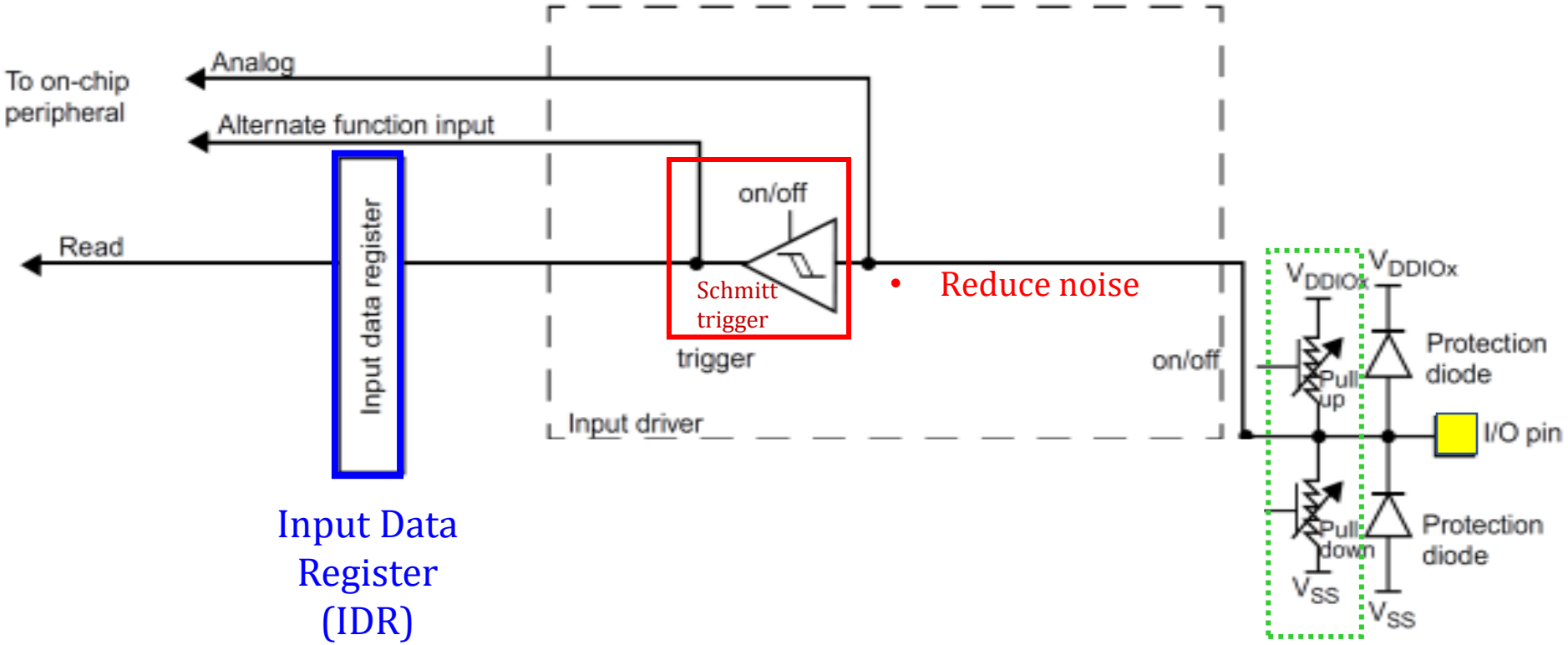


If external input is HiZ, read as a valid HIGH.



If external input is HiZ, read as a valid LOW.

Basic Structure of an GPIO Port Pin: Input



- Reduce noise

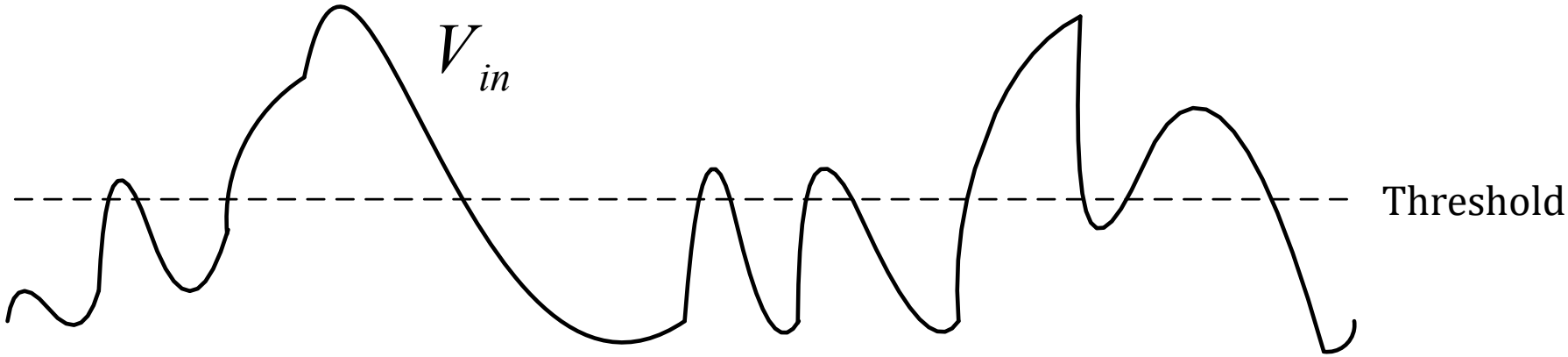
Input Data Register (IDR)

GPIO Pull-up/Pull-down Register (PUPDR)

`00` = No pull-up, pull-down `01` = Pull-up

`10` = Pull-down `11` = Reserved

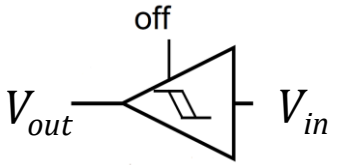
Schmitt Trigger

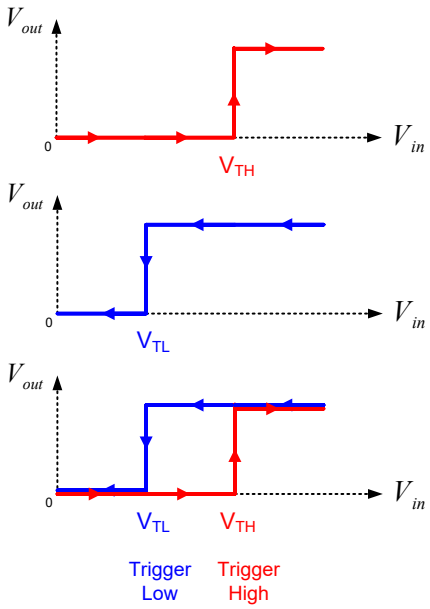


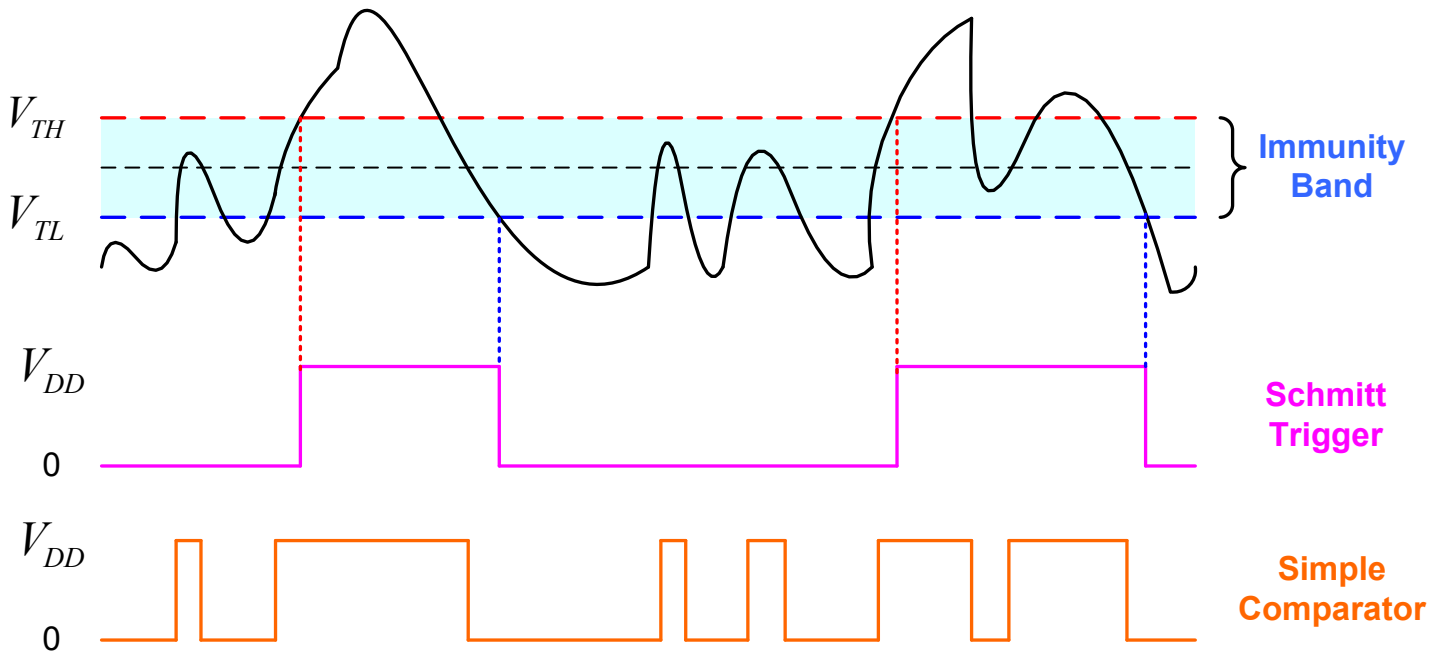
Analog signals

- Noisy
- Unstable

Schmitt Trigger







GPIO Input Data Register (IDR)

- 16 bits reserved, 16 data bits (1 bit per pin)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

```
// Demo of reading pin 7
uint32_t mask = 1UL<<7;
uint32_t input = (GPIOA->IDR & mask) == mask;

or

uint32_t input = (GPIOA->IDR & mask) >> 7;
```

GPIO Mode Register (MODER)

- 32 bits (16 pins, 2 bits per pin)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[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
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Pin 2 Pin 1 Pin 0

Bits 2y+1:2y **MODEy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)

```
// Set Pin 0 as input
GPIOA->MODER &= ~3UL;  // Clear bits 1 and 2 for Pin 0
```

Summary

- Timer
- GPIO