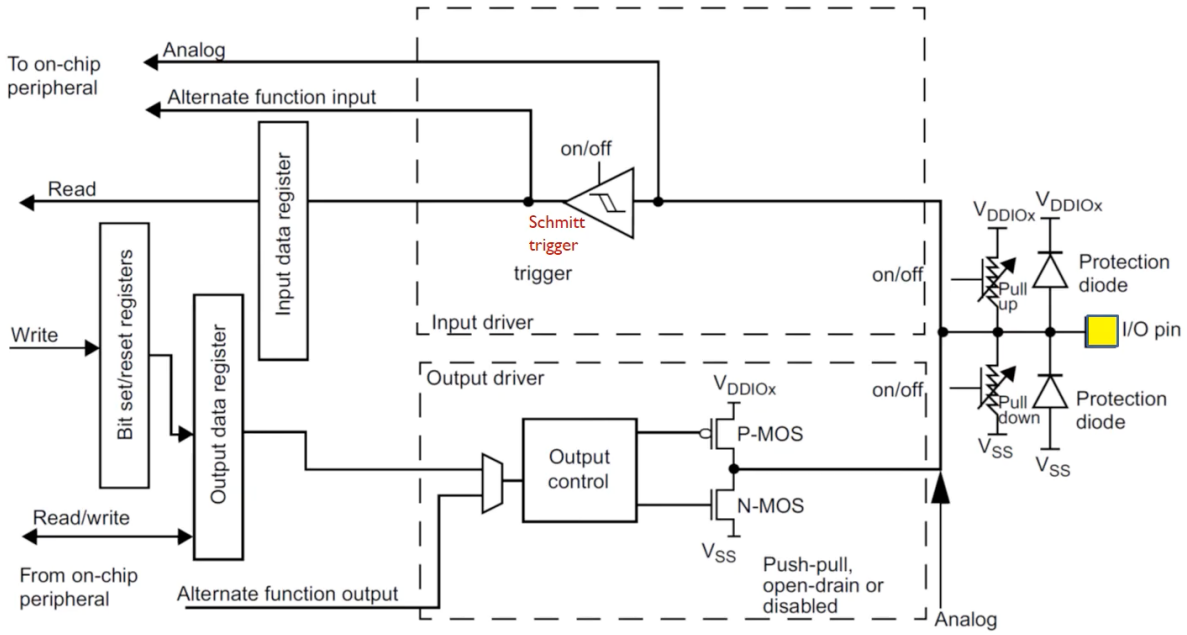
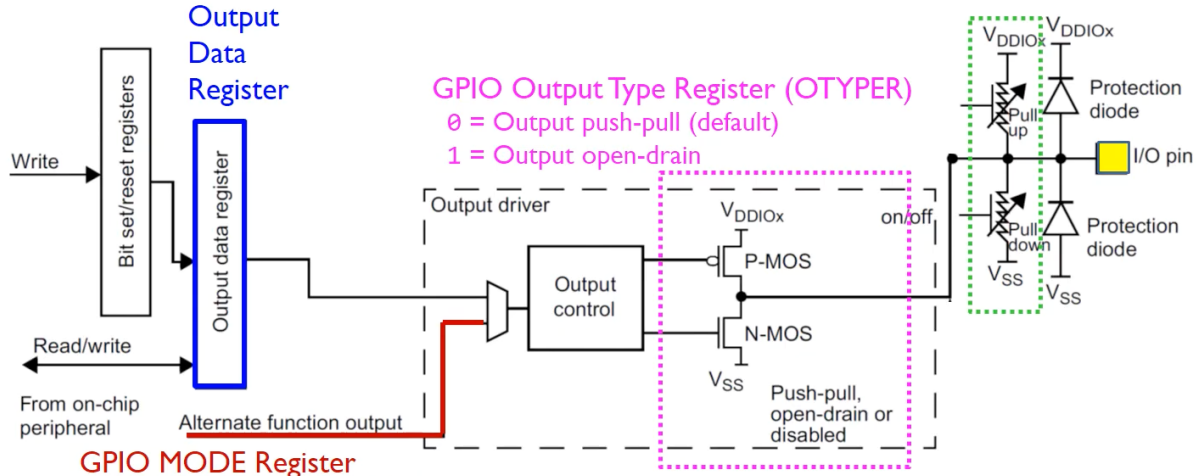
Basic Structure of an I/O Port Bit
Input and Output



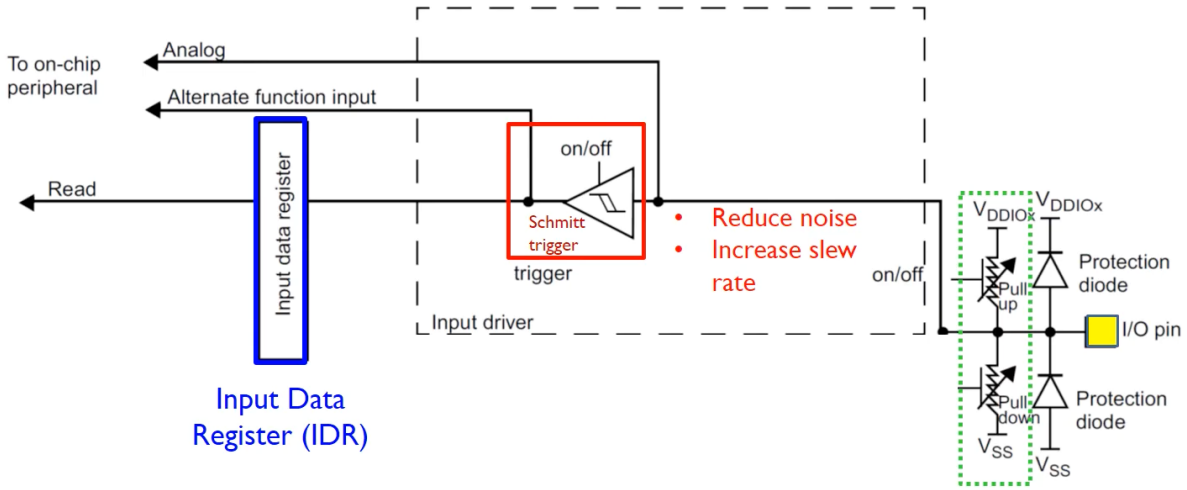
Basic Structure of an I/O Port Bit:
Output

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



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

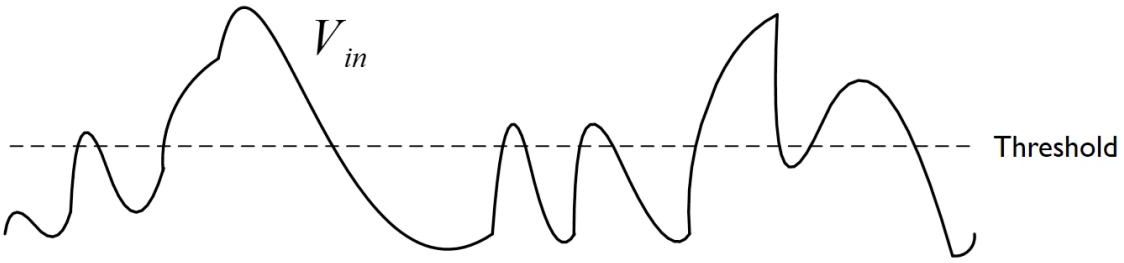
Basic Structure of an I/O Port Bit: Input



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

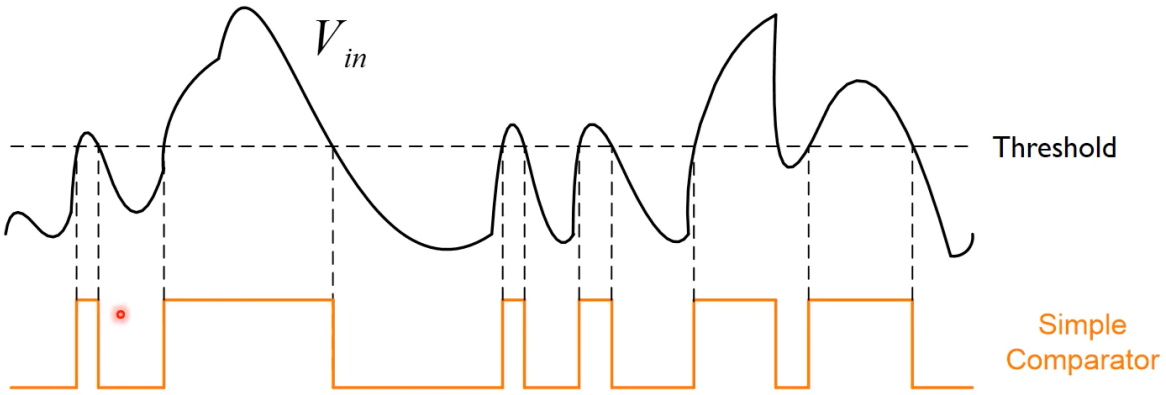
Schmitt Trigger



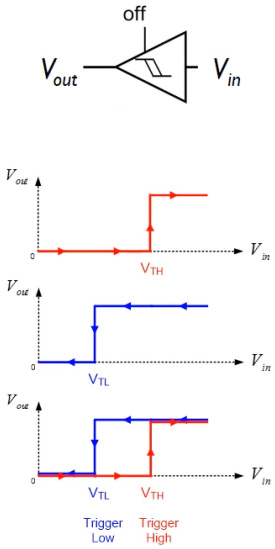
Analog signals

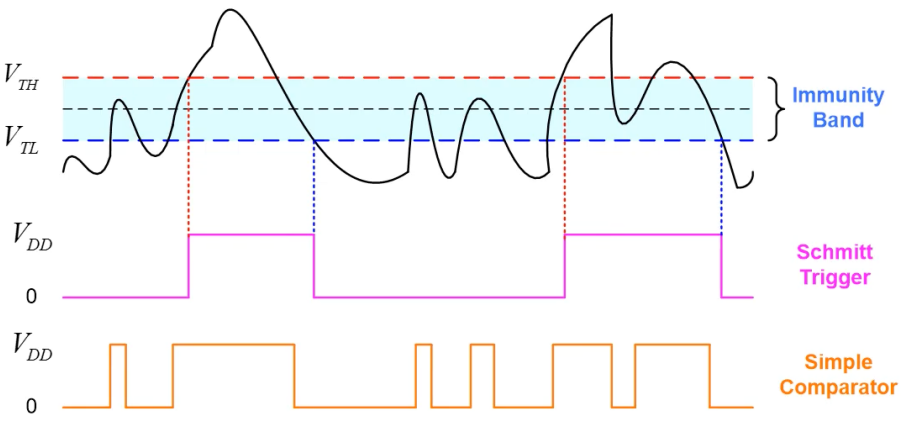
- Noisy
- Rise and fall slowly (small slew rate)

Schmitt Trigger

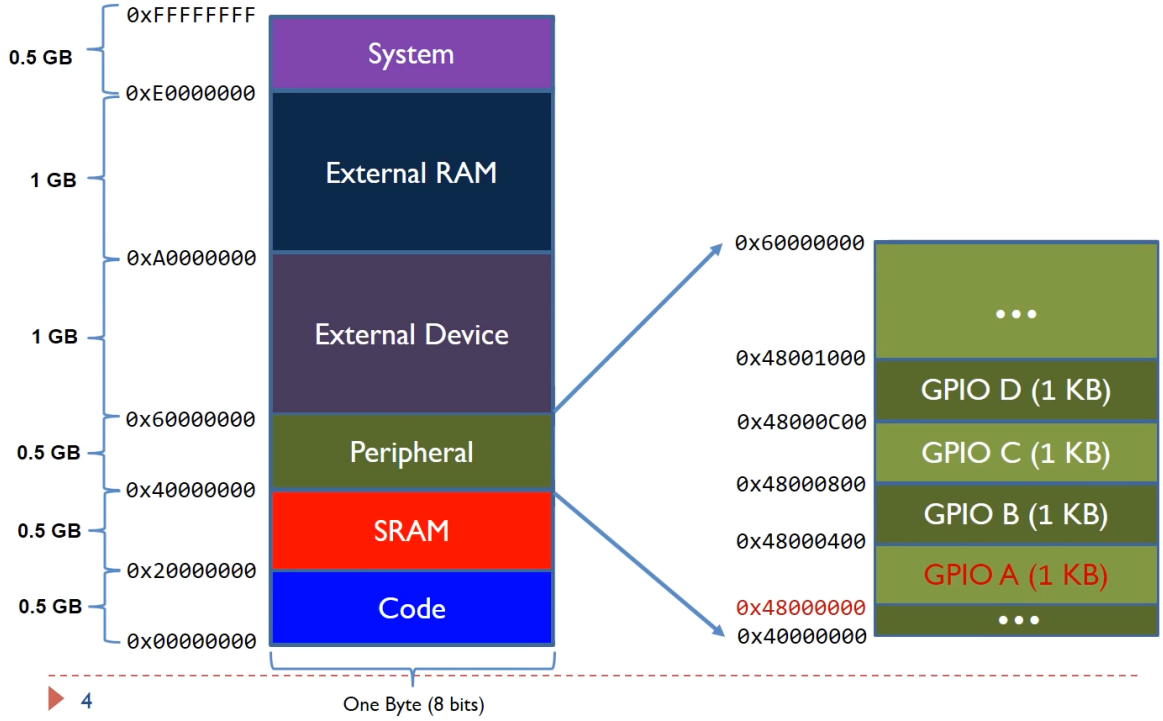


Schmitt Trigger

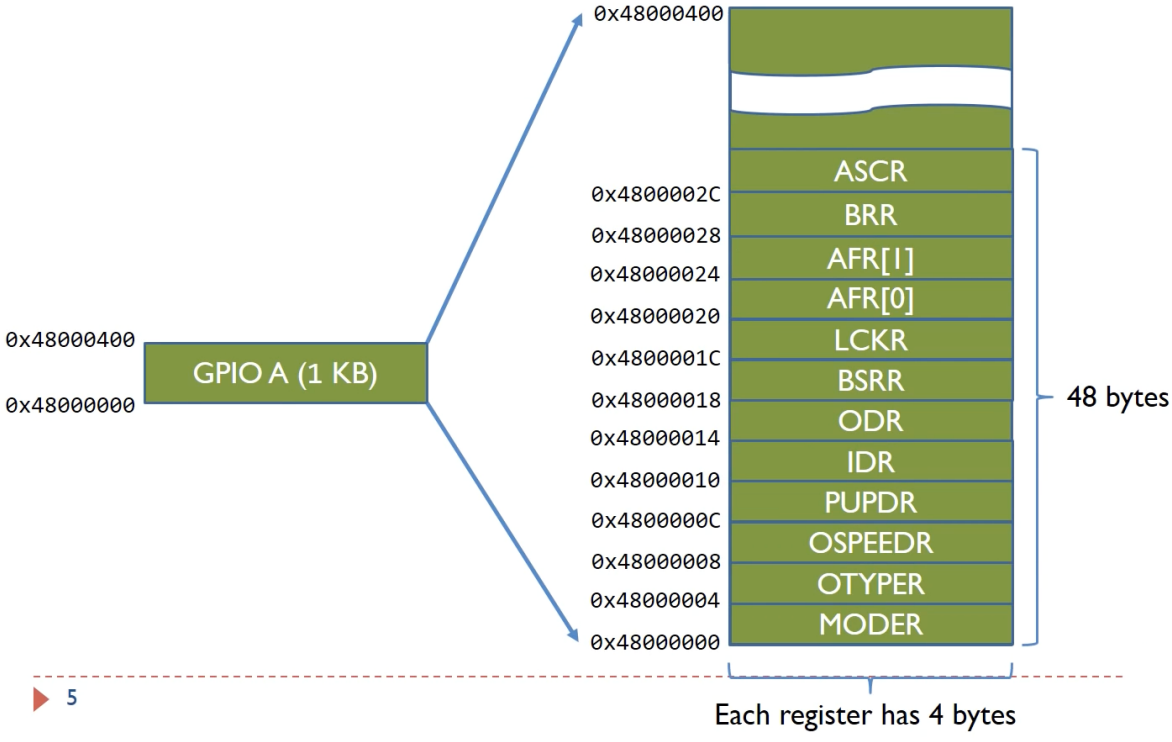




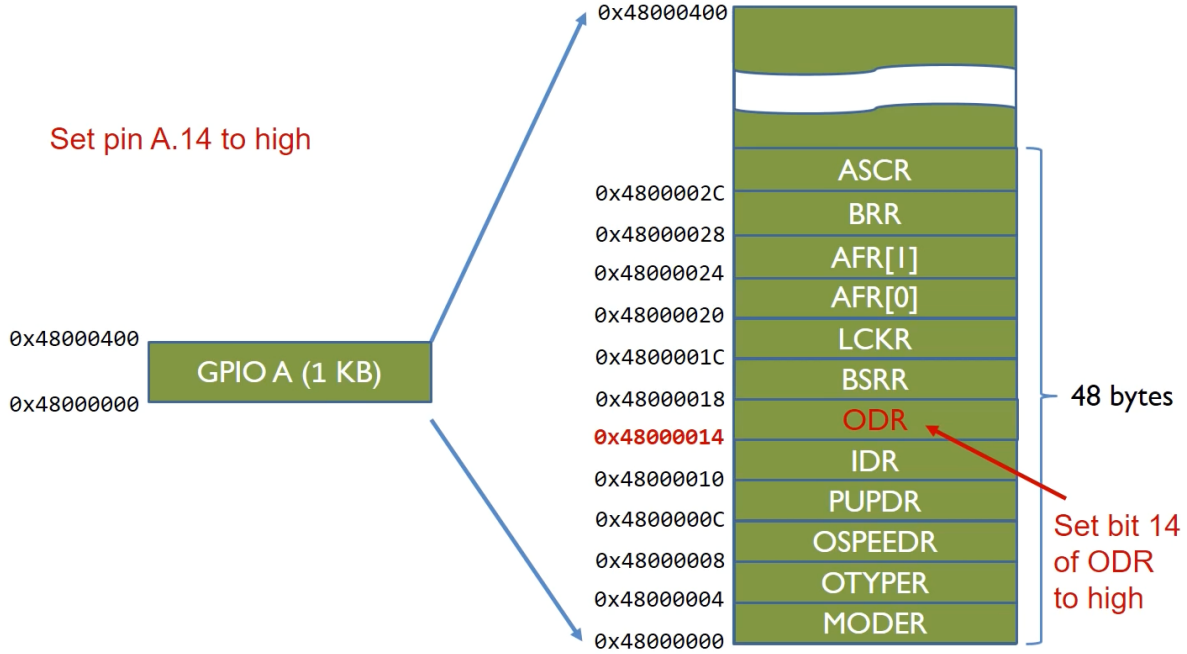
Memory Map of STM32L4



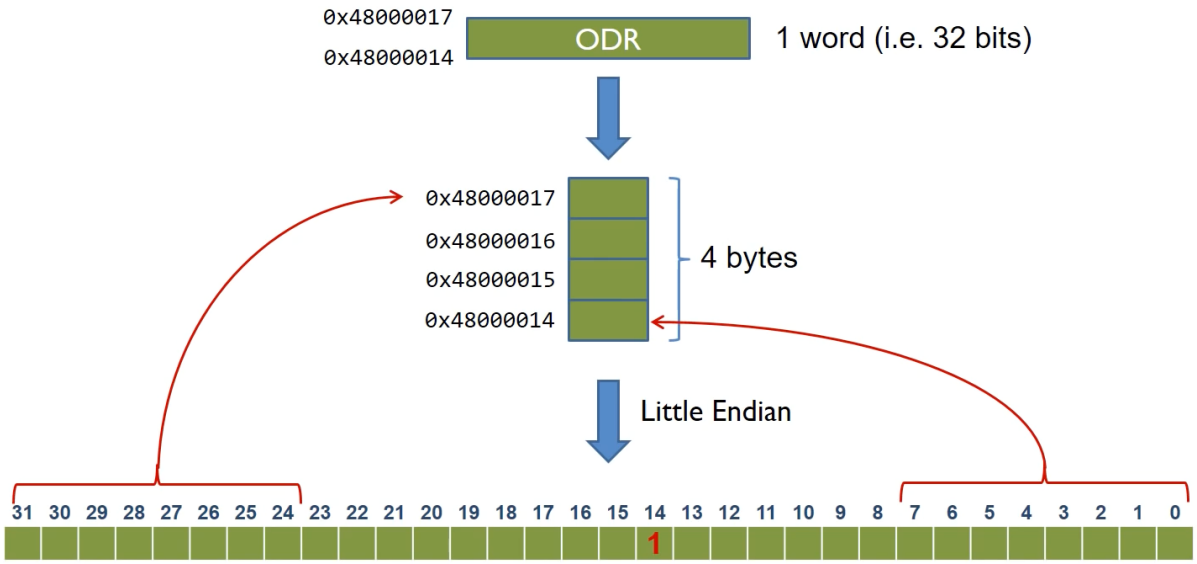
GPIO Memory Map



GPIO Memory Map



Output Data Register (ODR)

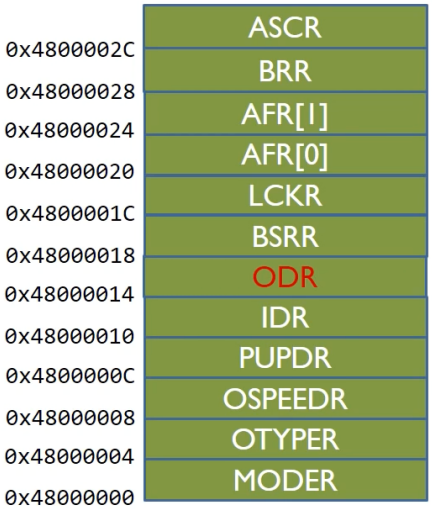



```
*((uint32_t *) 0x48000014) |= 1UL<<14;
```

Dereferencing a pointer

Bitwise OR

Dereferencing a Memory Address




```
typedef struct {
  volatile uint32_t MODER;    // Mode register
  volatile uint32_t OTYPER;   // Output type register
  volatile uint32_t OSPEEDR;  // Output speed register
  volatile uint32_t PUPDR;    // Pull-up/pull-down register
  volatile uint32_t IDR;      // Input data register
  volatile uint32_t ODR;      // Output data register
  volatile uint32_t BSRR;     // Bit set/reset register
  volatile uint32_t LCKR;     // Configuration lock register
  volatile uint32_t AFR[2];   // Alternate function registers
  volatile uint32_t BRR;      // Bit Reset register
  volatile uint32_t ASCR;     // Analog switch control register
} GPIO_TypeDef;

// Casting memory address to a pointer
#define GPIOA ((GPIO_TypeDef *) 0x48000000)
```

```
GPIOA->ODR |= 1UL<<14;
```
or
```
(*GPIOA).ODR |= 1UL<<14;
```

Programming in Assembly

In C

```
GPIOA->ODR |= 1UL<<14;    // Set bit 14 to 1
```

In Assembly

```
GPIOA_BASE EQU   0x48000000
GPIO_ODR   EQU   0x14

LDR r7, =GPIOA_BASE       ; Load GPIO port A base address
LDR r1, [r7, #GPIO_ODR]   ; r1 = GPIOA->ODR
ORR r1, r1, #(1<<14)      ; Set output of pin 14 to high
STR r1, [r7, #GPIO_ODR]   ; Write the output data register
```

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

- 16 pins per port, 2 bits per pin

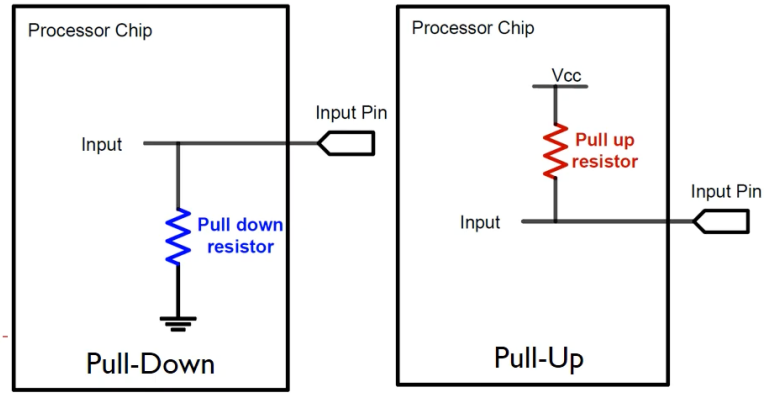
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		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;
```



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

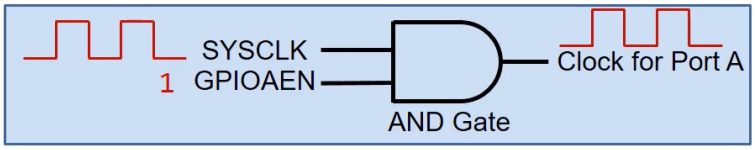
```
// Demo of reading pin 7
uint32_t mask = 1UL<<7;
uint32_t input = (GPIOA->IDR & mask) == mask;

or

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

Enable Clock

- AHB2 peripheral clock enable register (RCC_AHB2ENR)

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.	RNG EN	Res.	AESEN
													rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADCEN	OTGFS EN	Res.	Res.	Res.	Res.	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bit 0 **GPIOAEN:** IO port A clock enable
Set and cleared by software.
0: IO port A clock disabled
1: IO port A clock enabled




```
#define  RCC_AHB2ENR_GPIOAEN  ((uint32_t)0x00000001U)

RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN; // Enable clock of Port A
```

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
```