配置寄存器

- 目标：通过直接操作 寄存器 完成 GPIO 的配置

- 问题：
  - 不同 寄存器 的缩写对应什么意思？
  - 直接操作 寄存器 的本质是什么？

通过直接操作 寄存器 完成 GPIO 的配置

我们以使能GPIOC为例; APB2ENR是APB2 外设时钟使能寄存器 RCC_APB2ENR_IOPCEN是一个宏，定义为0x00000010，表示 GPIOC 时钟使能位，位于 APB2ENR 寄存器的第 4 位。 直接操作寄存器的写法如下：

```
RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;
```

上面的代码经过宏展开后是：

```
((RCC_TypeDef *) ((0x40000000 + 0x20000) + 0x1000))->APB2ENR |=
0x00000010;
```

RCC是一个指向结构体RCC_TypeDef的指针

```
typedef struct
{
    volatile uint32_t CR;          // 偏移量: 0x00
    volatile uint32_t CFGR;        // 偏移量: 0x04
    volatile uint32_t CIR;         // 偏移量: 0x08
    volatile uint32_t APB2RSTR;    // 偏移量: 0x0C
    volatile uint32_t APB1RSTR;    // 偏移量: 0x10
    volatile uint32_t AHBENR;      // 偏移量: 0x14
    volatile uint32_t APB2ENR;     // 偏移量: 0x18
    volatile uint32_t APB1ENR;     // 偏移量: 0x1C
    volatile uint32_t BDCR;        // 偏移量: 0x20
    volatile uint32_t CSR;         // 偏移量: 0x24
} RCC_TypeDef;
```

成员APB2ENR是一个32bit的寄存器，具体结构体开头的偏移量是0x18. 所以 RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;还可以写成这样：

```
*( (uint32_t*)0x40021018 ) |= 0x00000010;
```

Q1:寄存器缩写以及对应的含义

STM32 中有许多常用的寄存器和缩写，这些寄存器用于控制和配置 STM32 的外设和核心功能。以下是一些常用寄存器和缩写的列表，并对其进行简单说明：

1. GPIO（通用输入/输出）相关寄存器

- **GPIOx_MODER**（GPIO 模式寄存器）
  配置 GPIO 引脚的工作模式，例如输入、输出、复用功能、模拟模式等。每个引脚占用 2 位。
- **GPIOx_OTYPER**（GPIO 输出类型寄存器）
  设置 GPIO 引脚的输出类型是推挽输出还是开漏输出。
- **GPIOx_OSPEEDR**（GPIO 输出速度寄存器）
  配置 GPIO 引脚的输出速度，包括低速、中速、高速和极高速。
- **GPIOx_PUPDR**（GPIO 上拉/下拉寄存器）
  配置 GPIO 引脚的上拉或下拉电阻。
- **GPIOx_IDR**（GPIO 输入数据寄存器）
  读取 GPIO 引脚的输入电平。
- **GPIOx_ODR**（GPIO 输出数据寄存器）
  用于设置 GPIO 引脚的输出电平。
- **GPIOx_BSRR**（GPIO 位设置/复位寄存器）
  用于设置或复位 GPIO 引脚的输出电平，单个操作既可以设置高电平，也可以设置低电平。
- **GPIOx_AFRL / GPIOx_AFRH**（GPIO 复用功能低/高寄存器）
  配置 GPIO 引脚的复用功能，例如将引脚配置为 UART、I2C、SPI 等外设的引脚。

2. RCC（时钟控制）相关寄存器

- **RCC_CR**（时钟控制寄存器）
  用于启用或禁用内部或外部时钟源，管理 PLL（相位锁定环）以及外部振荡器 HSE、内部振荡器 HSI 等。
- **RCC_CFGR**（时钟配置寄存器）
  配置系统时钟源以及总线时钟（AHB、APB）的分频系数等。
- **RCC_AHBENR**（AHB 外设时钟使能寄存器）
  用于启用 AHB 总线上外设的时钟，如 DMA、CRC、FSMC、SDIO 等外设的时钟。
- **RCC_APB1ENR / RCC_APB2ENR**（APB1/APB2 外设时钟使能寄存器）
  用于启用 APB1 和 APB2 总线上的外设时钟，例如 USART、SPI、I2C、ADC 等。

3. USART（通用异步收发器）相关寄存器

- **USART_SR**（状态寄存器）
  用于监控 USART 的状态，如传输完成、接收数据是否就绪、是否发生错误等。

- **USART_DR**（数据寄存器）
  用于存储要发送或接收的数据。写入该寄存器将发送数据，读取该寄存器将获取接收到的数据。

- **USART_BRR**（波特率寄存器）
  用于配置 USART 的波特率。

- **USART_CR1 / CR2 / CR3**（控制寄存器）
  控制 USART 的工作模式和功能。例如启用发送、接收、中断，配置数据位、停止位、校验位等。

4. SPI（串行外设接口）相关寄存器

- **SPI_CR1**（控制寄存器 1）
  用于配置 SPI 模式、时钟极性、时钟相位、数据帧格式以及 SPI 的启用和禁用。

- **SPI_CR2**（控制寄存器 2）
  配置 SPI 的数据管理、DMA 功能、片选管理等。

- **SPI_SR**（状态寄存器）
  监控 SPI 的状态，如发送和接收缓冲区是否为空、传输完成、总线是否繁忙等。

- **SPI_DR**（数据寄存器）
  存储要发送或接收的数据。

5. I2C（集成电路互连）相关寄存器

- **I2C_CR1**（控制寄存器 1）
  控制 I2C 外设的启用、禁用、应答使能、时钟伸展等功能。

- **I2C_CR2**（控制寄存器 2）
  配置 I2C 的时钟频率、中断、DMA 请求等。

- **I2C_SR1 / SR2**（状态寄存器）
  监控 I2C 外设的状态，如启动/停止条件的生成、传输完成、接收缓冲区是否已满、错误标志等。

- **I2C_DR**（数据寄存器）
  存储要发送或接收的数据。

6. NVIC（嵌套向量中断控制器）相关寄存器

- **NVIC_ISER**（中断使能寄存器）
  用于使能特定中断。

- **NVIC_ICER**（中断禁用寄存器）
  用于禁用特定中断。

- **NVIC_ISPR**（中断挂起寄存器）
  手动挂起一个中断。

- **NVIC_ICPR**（中断清除挂起寄存器）
  清除挂起的中断标志。

- **NVIC_IPR**（中断优先级寄存器）
  设置中断的优先级。

7. SysTick（系统定时器）相关寄存器

- **SYST_CSR**（控制状态寄存器）
  控制系统定时器的启用、禁用、时钟源以及是否使能中断。
- **SYST_RVR**（重载值寄存器）
  配置系统定时器的重载值，用于设定定时器的计数周期。
- **SYST_CVR**（当前值寄存器）
  记录定时器当前的计数值。
- **SYST_CALIB**（校准值寄存器）
  用于定时器校准，存储工厂校准的时钟值。

Q2:直接操作 寄存器 的本质是什么?

直接操作寄存器的本质是**通过修改和读取特定硬件寄存器的值来控制和配置微控制器（如 STM32）的外设或核心功能**。寄存器是微控制器内部用于控制外设的内存位置，通过向这些特定的寄存器写入数据，开发者可以直接影响微控制器的行为。

直接操作寄存器的本质：

1. **寄存器是硬件资源**：
   - 寄存器是微控制器中用于存储数据或控制信息的存储单元，每个寄存器都对应一个特定的地址。不同的寄存器负责控制不同的功能模块，如 GPIO、USART、SPI、定时器等。
   - 每个外设都有一组寄存器，通过这些寄存器，开发者可以配置外设的工作模式、启用/禁用功能、传递数据等。
2. **寄存器是特定的内存地址**：
   - 在微控制器的内存映射（Memory Map）中，外设的寄存器被映射到特定的内存地址。通过访问这些内存地址，开发者可以读写寄存器。
   - 每个寄存器对应一个唯一的内存地址，当开发者操作这个地址时，实际上是控制硬件的某一部分行为。例如，设置 GPIO 引脚为输出模式或控制 UART 的波特率。
3. **读写寄存器以控制硬件**：
   - **写寄存器**：通过写入数据到某个寄存器，开发者可以改变外设的配置，例如设置 GPIO 引脚为输入或输出，启动定时器，启用或禁用中断等。
   - **读寄存器**：通过读取某个寄存器的值，开发者可以获取外设的状态信息，例如读取 GPIO 引脚的电平，检查 USART 的发送缓冲区是否空闲，判断是否有中断发生等。
4. **寄存器操作的位级控制**：
   - 每个寄存器通常包含多个位，每个位或几个位控制硬件的某个特定功能。开发者通过对这些位进行位操作（置位、清位、掩码等）来精确控制硬件。

- 例如，某个寄存器的某一位可能用于控制一个 LED，开发者可以通过设置这一位为 1 或 0 来打开或关闭 LED。