Cortex-M3的操作模式和特权级别

操作模式：（1）处理者模式即handler模式（异常和中断） （2）线程模式（主应用程序）

运行级别：（1）特权级 （2）用户级

在CM3的主应用程序即可以在特权级中运行也可以在用户级中运行，但是处理者模式必须是在特权级中运行。

复位后处理器进入默认进入线程模式中以特权级运行。

在特权级下的程序可以访问除MPU禁止的存储区域以外的任何地址，以及执行所有的指令。

用户级的程序要回到特权级中不能通过改写CONTROL寄存器而达到目的。只能通过触发SVC异常而进入异常服务程序，在异常服务程序中修改CONTROL寄存器而进入特权级。

Cortex-M3寄存器

1. 通用寄存器（R0——R12）
2. 低组寄存器（R0——R7）：所有指令都能访问
3. 高组寄存器（R8——R12）：所有32位是Thumb-2指令都能访问。但只有少部分16位的Thumb指令能访问高组寄存器。
4. 堆栈指针寄存器R13（SP\_Main和SP\_Process）
5. 主堆栈指针MSP（SP\_Main）：这是默认的堆栈指针，它由操作系统内核、异常服务例程以及所有需要特权访问的应用程序代码来使用。
6. 进程堆材指针PSP（SP\_Process）：用于线程模式中不需要特权访问的应用程序代码。

handler模式和线程模式都可以使用MSP但是只有线程模式才能使用PSP。

任何时侯MSP和PSP都只有一个可见，此时R13指向它。

1. 连接寄存器R14

在汇编中可以把连接寄存器写成R14或者LR。连接寄存器的使用是在调用子程序时保存返回地址，也用于异常返回。

1. 程序计数器R15

在汇编中可以把程序计数器写成R15或者PC（Program Counter）。由于CM3使用3级指令流水线，所以读PC寄存器的内容时返回的是当前指令的地址加4。

如果向PC寄存器写入数据会引起一次程序的分支，但不更新LR寄存器。

1. 程序状态寄存器PSR

程序状态寄存器分为3个子状态寄存器：应用程序状态寄存器APSR、中断号寄存器IPSR、执行状态寄存器EPSR。它们可以通过MRS/MSR指令单独访问，也可以通过使用“PSR”或者“xPSR”这两个名字来同时访问这三个子状态寄存器。

程序状态寄存器用来表示大于、小于、不等于、进位、溢出等程序状态。

1. PRIMASK寄存器：中断屏蔽位寄存器。
2. BASEPRI寄存器：可屏蔽等于于或者低于某个优先级的中断。
3. FAULTMASK寄存器：错误屏蔽寄存器。
4. 控制寄存器CONTROL

作用一是用来定义特权级别，使用二是用于选择当前使用哪个堆栈指针。

Cortex-M3总线接口

Cortex-M3处理器采用了哈佛结构，拥有独立的指令总线和数据总线，可以让取指令与数据访问同时进行，数据访问不再占用指令总线，从而提升了性能。

Cortex-M3内部有4个总线接口，通过总线矩阵把处理器和调试接口连接到外部总线，以使Cortex-M3能同时取指令和访问内存或者外设，它们分别是：

1. I-Code接口：32位的AHB-Lite总线接口

从Code存储空间（0x0000 0000 —— 0x1FFF FFFF）的取指令和取向量都在这条总线上进行。只有CM3Core取指令总线可以访问I-Code接口，以获取最佳的代码读取性能。所有取指令都是按字来操作的。

1. D-Code接口：32位AHB-Lite总线接口

对Code空间（0x0000 0000 —— 0x1FFF FFFF）进行数据和调试访问都是在这条总线上进行。内核数据访问的优先级比调试访问的优先级要高，因而总线上同时出现内核和调试访问时，必须在内核数据访问结束后才能进行调试访问。

1. 系统接口：32位AHB-Lite总线接口

对系统空间（0x2000 0000 —— 0xE00F FFFF）进行取指令、取数据和调试访问都在这条总线上进行。系统总线用于访问内存和外设，覆盖的区域包括SRAM、片上外设、片外SRAM、片外RAM、片外扩展设备及系统存储区域的部分空间。系统总线接口包含处理不对齐访问、FPB重映射访问、位带访问及流水线取指令的控制逻辑。

1. APB接口：外部专用外设总线桥接口

对外部PPB空间（0xE004 0000 —— 0xE00F FFFF）进行数据访问和调试都在这条总线进行。跟踪端口接口单元TPIU和厂商特定的外围器件都在这条总线上。内核数据访问的优先级比调试访问的优先级要高，因而总线上同时出现内核和调试访问时，必须在内核数据访问结束后才能进行调试访问。APB1 最大速度为36MHz，而APB2最大速度为72MHz。

※. 另外，Cortex-M3处理器还包含一条内部专用外设总线，用来访问嵌套向量中断控制器NVIC、数据观察点以及触发单元DWT、Flash修补和断点单元FPB以及存储器保护单元MPU。

Cortex-M3位绑定

位绑定是指把一个地址单元的32位变量中的每一位，通过一个简单的地址换算法，映射到另一个地址空间，每一位占用一个地址，对地址的操作，只有数据的最低位有效。位绑定可以把代码缩小，使程序速度更快、效率更高、更安全。

支持位绑定的两个内存区域：

1. SRAM中的最低1M空间：0x2000 0000 —— 0x200F FFFF
2. 片上外设的最低1M空间：0x4000 0000 —— 0x400F FFFF

Cortex-M3的异常和中断

1. CM3有15个异常，编号分别为1到15，没有0号异常。

复位异常、NMI异常、硬件失败异常，这三个异常的优先级为负数，即它们有最高优先级。其它异常为：存储管理异常、总线错误、错误应用、SVCall、调试监控、PendSV、SysTick，另外没用到的异常号是保留的。

1. CM3有240个中断源，但大多数芯片都不支持240个中断。
2. 为了提高时实性CM3的异常和中断支持以下功能：
3. 抢占：

当在进行异常处理时，如果一个新的异常发生，且新异常的优先级比当前正在处理的异常的优先级要高，则新异常会抢占当前的异常。这时就产生了中断嵌套。

1. 末尾连锁（tail-chainning）:

当新的中断比正在处理的中断优先级低时，新中断被挂起，直到当前中断处理完任务后，再去处理挂起的中断。从当前中断切换到新中断的传统方法是先弹栈，然后又把弹出去的内容压回栈中。为加快对被挂起中断的响应，CM3去掉了出栈再入栈的动作，前一个中断完成以后直接执行被挂起的中断。这被称为末尾连锁。

1. 迟来（late-arriving）:

如果前一个中断正在压栈，还未进入中断函数，此时一个更高优先级的中断到来，则会优先处理更高优先级的新中断，然后再末尾连锁地执行先前的中断。

STM32主要时钟源

1. 高速外部时钟源HSE

高速外部时钟源频率要求4到16MHz，常用8MHz的晶振作为高速外部时钟源。HSE经过分频或倍频（PLL）后，得到系统时钟。系统时钟通过分频，产生各种外设所需要的时钟。

1. 内部振荡时钟HIS

内部振荡时钟频率为8MHz，可直接作为系统时钟，或在2分频后作为PLL输入。

1. 外部32.768KHz时钟源LSE

LSE是实时时钟RTC的时钟源。

1. 内部40KHz时钟源LSI

LSI是一个低功耗的时钟源，它可以在停机和待机模式下保持运行，为独立看＇门狗和自动咉醒单元提供时钟。

1. AHB和APB总线桥

APB1和APB2是片上外设的时钟。APB1最大速度为36MHz，主要负责DA、USB、SPI2、SPI3、I2C、CAN、串口2345、普通TIM。APB2最大速度为72MHz主要负责AD、I/O、高级TIM、SPI1、串口1。

STM32复位模式

1. 系统复位

系统复位后，除了时钟控制器CSR中的复位标志和备用域的寄存器外，其它所有的寄存器都将被复位。以下事件将产生系统复位：

1. NRST引脚出现低电平（外部复位）
2. 窗口看门狗计数终止（WWDG复位）
3. 独立看门狗计数终止（IWDG复位）
4. 软件复位（SW复位）
5. 低功耗管理复位
6. 电源复位

电源复位分为上电复位POR和掉电复位PDR。电源复位能复位除了备份寄存器以外的所有寄存器。以下事件将产生电源复位：

1. 上电或掉电
2. 从待机模式中返回
3. 备份区域复位

以下事件将产生备份区域复位：

1. 软件置位备份区域控制寄存器RCC\_BDCR中的BDSRT位。
2. 在Udd和Ubat两者同时掉电时将导致备份区域复位。

STM32的启动设置

1. 系统启动时，内部的8MHz时钟源被先为默认的CPU时钟。随后可以选择外部具有失效监控的4到16MHz时钟，当外部时钟失效时，它将会被隔离，同时产生一个中断。
2. STM32启动时，CPU从0x0000 0000处取栈顶地址并设置栈顶，然后从0x0000 0004处开始执行代码。
3. STM32F10x可以通过BOOT[1:0]引脚来选择3种不同的启动模式：
4. BOOT[X:0] ：从主闪存存储器启动（NORFlash）。

主闪存存储器地址空间被映射到启动空间0x0000 0000，但仍然能够在它原来的地址0x0800 0000处访问。

1. BOOT[0:1] ：从系统存储器启动。

系统存储器地址空间被映射到启动空间0x0000 0000，但仍然能够在它原来的地址处访问，互联型产品系统存储器的地址为0x1FFF B000，其它产品系统存储器的地址为0x1FFF F000

1. BOOT[1:1] ：从内部SRAM启动。

从内置SRAM启动只能从0x2000 0000地址处访问SRAM。从SRAM启动一般只作为调试来使用，或者用来解除内部Flash的读/写保护。

STM32F10x的GPIO

|  |  |
| --- | --- |
| 通用输出 | 推挽（push\_pull）：上拉输出。 |
| 开漏（open\_drain）：如果需要上拉，则必须外接。 |
| 复用输出 | 推挽复用：片上外设控制输出。 |
| 开漏复用：片上外设控制输出，如果需要上拉则必须外接。 |
| 输入 | 上拉输入：带内部上拉的数字输入。 |
| 下拉输入：带内部下拉的数字输入。 |
| 模拟输入：在模拟输入下，数字输入总为1。 |
| 浮空输入：内部没有上拉或下拉的数字输入。 |

GPIO的使用方式：

1. 作为普通GPIO输出：配置为推挽输出或者开漏输出，同时不要使能该引脚的复用功能。如果配置成开漏输出，则应外接上拉电阻。
2. 作为普通GPIO输入：配置成上拉输入、下拉输入或者浮空输入。如果配置成浮空输入则应该根据需求外拉上拉电阻或者下拉电阴。
3. 作为内置外设的输出：根据需要配置成推挽复用或者开漏复用输出，并使能该外设的复用功能，如果是开漏复用输出，则应外接上拉电阻。
4. 作为内置外设的输入：配置成上拉输入、下拉输入或者浮空输入，同进使该外设的复用功能。如果配置成浮空输入则应根据需求外接上拉电阻或者下拉电阻。
5. 作为模拟输入：ADC或者DAC都应该配置成模拟输入。虽然DAC是一种输出模拟量的工作方式，但它却要配置成模拟输入。

GPIO的速度选择：

在输出模式下，GPIP有3种速度可以选择：2MHz、10MHz、50MHz。这里的速度是指IO口驱动电路的响应速度，而不是输出信号的速度，输出信号的速度与程序有关。由于同频电路的噪声较高，所以当不需要太高的输出频率时，应选择较低的GPIO速度，这有利于提高系统的EMI性能。比如：UART串口使用的最大波特率为115200Hz，那么使用2MHz的GPIO速度就足够了。

GPIO的控制寄存器：

1. 两个32位配置寄存器CRL和CRH
2. 端口配置低寄存器GPIOx\_CRL：用于设置端口低8位的工作模式。
3. 端口配置高寄存器GPIOx\_CRH：用于设置端口高8位的工作模式。
4. 两个32位数据寄存器IDR和ODR
5. 输入数据寄存器GPIOx\_IDR：如果端口被配置成输入端口，则可以从IDR寄存器中读取输出数据。
6. 输出数据寄存器GPIOx\_ODR：如果端口被配置成输出端口，则对寄存器某一位写1则对应管脚输出高电平。

* IDR和ODR都是低16位使用，高16位保留。IDR和ODR这两个寄存器都是可读的，但是IDR反映IO口的真实电平状态。比如GPIOA的ODR寄存器设置bit0为1，此时读ODR可以得出bit0为1。如果此时读IDR得出bit0为1则说明GPIOA Pin0正常输出了高电平，但如果读IDR得出bit0为0则说明GPIOA Pin0接到了低电平上，或者被内置外设置0了。

1. 复位/置位寄存器BSRR

对GPIOx\_BSRR写0无动作，对GPIOx\_BSRR低16位写1则对应管脚输出高电平，对GPIOx\_BSRR高16位写1则对应管脚输出低电平。

1. 复位寄存器BRR（低16位有效高16位保留）

对低16位写1则对应管脚被复位，输出低电平。写0无动作。

1. 端口配置锁定寄存器LCKR

端口配置锁定寄存器GPIOx\_LCKR用来锁定管脚的配置，锁定后不允许修改配置。

STM32中断系统

stm32中断优先级的概念是针对“中断通道”而言的，当中断通道的优先级确定以后，中断通道中所有的中断源都使用中断通道的优先级。

stm32支持84个中断源，分别为15个内核异常和68个外部中断。68个中断通道已经固定地分配给外部设备。每个中断通道都有自己的中断优先级控制8位字节PRI\_n，stm32只使用该字节的高4位。每4通道的8位中断优先级控制字节PRI\_n组成一个32位的优先级寄存器（Priority Register）。68个通道的优先级控制寄存器构成17个32位的优先级寄存器，它们是NVIC寄存器的重要组成部分。

某些中断源会打断当前的中断程序的运行，这称为中断嵌套。能打断其它中断的中断源必须具有占先式优先级。中断抢占和嵌套的一些情况：

1. 高占式先优先级的中断可以抢占正在运行的低占先式优先级的中断。
2. 如果两个中断同时到来，在占先式优先级相同的情况下，副优先级高的中断会优先被响应。
3. 如果一个中断正在执行，此时一个新的中断到来。新中断的占先式优先级和当前中断的占先式优先级相同，但新中断的副优先级比当前中断的副优先级要高。这种情况下新中断无法抢占前当中断，只能等当前中断执行完退出后才能执行新中断。
4. 如果两个中断同时到来，这两个中断的占先式优先级和副优先级都相同，则根据中断向量表中中断函数的顺序来执行。

STM32中断优先级分组

中断优先级分组库函数：

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_x)

参数可先项：

1. NVIC\_PriorityGroup\_0：无占先式什么级，16个副优先级。
2. NVIC\_PriorityGroup\_1：2个占先式什么级，8个副优先级。
3. NVIC\_PriorityGroup\_2：4个占先式什么级，4个副优先级。
4. NVIC\_PriorityGroup\_3：8个占先式什么级，2个副优先级。
5. NVIC\_PriorityGroup\_0：16个占先式什么级，无副优先级。

STM32外部中断/事件硬件结构

Stm32的外部中断/事件系统由两部分组成：一个标准的CM3嵌套向量中断控制器NVIC；一个外部中断/事件控制器EXIT。所有来自外部的中断都通过EXTI汇集到NVIC。

Stm32有19根外部中断/事件线（EXTI），所有IO口的0号IO（如PA0、PB0、PC0等）都接在EXTI0上，以此类推所有IO口的15号IO都接到EXTI15上。所以PA0、PB0、PC0…之中只以有一个管脚为外部中断。剩下的EXTI16连接到PVD输出；EXTI7连接到RTC时钟事件；EXTI18连接到USB唤醒事件。

每一个EXTI都有一个边沿检测器，可以独立地配置输入类型（脉冲或挂起）和对应的触发事件（上升沿触发、下降沿触发或者双边沿触发）。

通用同步/异步串行收发器

通用异步收发器UART：

UART需要固定的波持率进行数据的接收和发送，即每两个位之间的时间间隔是相等的。UART最少只需要3根线：Tx、Rx、GND。

通用同步/异步收发器USART：

当使用异步模式收发数据时USAET和UART没有区别。当使用同步模式时，需要使用同步时钟信号进行数据的接收和发送。使用同步模式时最少需要3根线：Tx、Rx、SCKL。

※ 异步通信的传输方式为：空闲时Tx和Rx都为高电平，当要发送数据时Tx拉低电平一个传输位的时间，以此作为起始位。然后开始传输数据位，每隔一个位的时间根据要传输的数据拉高或拉低电平来传输数据。传输完数据位后，根据配置来传输或不传输奇偶校验位。如果还有下一个字节要传输，则Tx拉低电平一个传输位的时间来作为下一个字节的起始位。如果所有数据都已经传输完，则拉高Tx电平。

※ 同步通信的传输方式为：在时钟信号的上升沿或下降沿或者上下边沿来发送或接收一位数据。USART使用同步模式收发数据时可以和SPI连接通信。

※ 硬件流控制RTS/CTS：

RTS流控制：如果RTS流被使能，则只需要UART准备好了接收数据，则接收端的RTS引脚输出低电平，使发送端的CTS收到低电平信号，则发送端就可以发送数据了。如果接收端的接收缓冲已满，则接收端的RTS引脚输出高电平，使发送端的CTS收到高电平信号，发送商量则停止发送，直到收到低电平信号才会继续发送。

CTS流控制：如果CTS流被使能，则发送端在发送前都会检测CTS引脚，如果CTS引脚处于高电平则暂停发送，直到收到低电平信号才断续发送。

STM32定时器

1. 定时器的定时计算

定时计算公式：Tout = ((arr+1)\*(psc+1))/Tclk

Tclk：TIM3 的输入时钟频率（单位为 Mhz）。

Tout：TIM3 溢出时间（单位为 us）。

1. 定时器时钟频率

系统初始化的时候在默认的系统初始化函数 SystemInit 函数里面已经初始化 APB1 的时钟为 2 分频，所以 APB1 的时钟为 36M，而从 STM32 的内部时钟树图得知：当 APB1 的时钟分频数为 1 的时候，TIM2~7 的时钟为 APB1 的时钟，而如果 APB1 的时钟分频数不为 1，那么 TIM2~7 的时钟频率将为 APB1 时钟的两倍。所以默认情况下不管是挂在APB2上的TIM1、TIM8还是挂在APB1上的其它定时器，时钟频率都为72MHz。

|  |
| --- |
| 高级定时器：TIM1、TIM8 |
| 通用定时器：TIM2、TIME3、TIME4、TIME5 |
| 普通定时器：TIM6、TIM7（主要作用是触发DAC） |
| 实时时钟RTC |
| SysTick定时器：24位递减计数器。主要用来作为延时或者提供RTOS的时钟节拍 |
| 独立看门狗定时器、窗口看门狗定时器 |

|  |  |
| --- | --- |
| 定时器时基单元 | |
| TIMx\_CNT | 计数值寄存器 |
| TIMx\_PSC | 预分频寄存器 |
| TIMx\_ARR | 自动重载寄存器 |
| TIMx\_CRC | 循环计数寄存器，作用是设定n个更新事件后才产生一个更新中断（仅高级定时器有） |

PSC、ARR、CCR1、CCR2、CCR3、CCR4这几个寄存器都对应两个物理寄存器。分别是：可由程序员读写的说装载寄存器；程序员无法直接读写的影子寄存器。影子寄存器的作用是为了能够同步更新PSC、ARR、CCRx这几个寄存器。

|  |  |
| --- | --- |
| TIMx寄存器 | |
| TIMx\_CR1 | 控制寄存器1 |
| TIMx\_CR2 | 控制寄存器2 |
| TIMx\_SMCR | 模式控制寄存器，用于从模式控制 |
| TIMx\_SR | 定时器状态寄存器 |
| TIMx\_EGR | 事件产生寄存器 |
| TIMx\_CCMR1 | 捕获/比较模式寄存器1 |
| TIMx\_CCMR2 | 捕获/比较模式寄存器2 |
| TIMx\_DCR | 捕获/比较使能寄存器 |
| TIMx\_CNT | 定时器DMA控制寄存器 |
| TIMx\_PSC | 预分频寄存器 |
| TIMx\_ARR | 自动重装载寄存器 |
| TIMx\_CCR1 | 捕获/比较通道1的计数值 |
| TIMx\_CCR2 | 捕获/比较通道2的计数值 |
| TIMx\_CCR3 | 捕获/比较通道3的计数值 |
| TIMx\_CCR4 | 捕获/比较通道4的计数值 |
| TIMx\_DMAR | 定时器DMA传输连续模式的起始地址 |

|  |  |
| --- | --- |
| TIMx\_CR1寄存器常用位 | |
| ARPE（ARR预装载允许位） | 0：ARR寄存器不使用预装载，写入ARR的值直接进入影子寄存器。 |
| 1：ARR允许预装载，写入ARR的值在下一个更新事件写入影子寄存器。 |
| CEN | 0：禁止计数器工作 |
| 1：使能计数器 |
| UDIS | 0：允许UEV更新事件。UEV更新事件由以下产生：  （1）计数器溢出 （2）软件设置UG位 （3）从模式控制器产生的更新 |
| 1：禁止UEV更新事件产生。影子寄存器（ARR、PSC、CCRx）保持它们原有的值。如果软件设置UG位或者从模式控制器触发了一个定时器复位，则计数器和预分频器的计数清0（预分频的系数也就是PSC寄存器的值保持不变）。 |
| URS | 0：允许计数器溢出、设置UG位以及从模式控制器这三种情况产生更新事件并触发更新中断和DMA传输。 |
| 1：只允许计数器溢出产生这一种情况产生更新事件并触发更新中断和DMA传输。 |
| DIR | 0：计数器向上计数 |
| 1：计数器向下计数 |
| CMS【1:0】 | 00：边沿对齐模式计数。根据DIR寄存器的值决定计数方向。 |
| 01：中央对齐模式1。当向上计数的值等于ARR时产生更新事件。 |
| 10：中央对齐模式2。当向下计数的值等于0时产生更新事件。 |
| 11：中央对齐模式3。向上计数值等于ARR时以及向下计数值等于0时都会产生更新事件。 |

SPI

1. SPI占用4根线：

MISO：Master in slave out

MOSI：Master out slave in

SCLK：时钟线

CS：片选线

1. SPI工作原理

SPI的主机和从机都有一个串行移位寄存器，主机通过向它的移位寄存器写入一个字节来发起一次传输。寄存器通过 MOSI 信号线将字节传送给从机，从机也将自己的移位寄存器中的内容通过 MISO 信号线返回给主机。这样，两个移位寄存器中的内容就被交换。

SPI的写操作和读操作是同步完成的。如果只进行写操作，主机只需忽略接收到的字节；反之，若主机要读取从机的一个字节，就必须发送一个空字节来引发从机的传输。

1. SPI 模块为了和外设进行数据交换，根据外设工作要求，其输出串行同步时钟极性和相位可以进行配置，时钟极性（CPOL）对传输协议没有重大的影响。如果CPOL=0，串行同步时钟的空闲状态为低电平；如果 CPOL=1，串行同步时钟的空闲状态为高电平。时钟相位（CPHA）能够配置用于选择两种不同的传输协议之一进行数据传输。如果CPHA=0，在串行同步时钟的第一个跳变沿（上升或下降）数据被采样；如果 CPHA=1，在串行同步时钟的第二个跳变沿（上升或下降）数据被采样。SPI 主模块和与之通信的外设备时钟相位和极性应该一致。
2. SPI可以设置从低位到高位依次发送数据（MSB），也可以设置从高位到低位依次发送数据（LSB）。
3. STM32的SPI最大工作频率为18MHz。由于SPI1是挂在APB2上的，而APB2默认72MHz频率，所以SPI1的时钟分频数必须大于等于4。SPI2、SPI3挂在APB1上，APB1默认36MHz，所以SPI2、SPI3的时钟分频数必须大于等于2。

STM32使用SPI控制74HC595进行IO扩展

1. 74HC695主要管脚：
2. Q0——Q7：并行IO输出
3. Q7＇：串行数据溢出线
4. MR（master reset）：复位线
5. SHCP（shift register clock input）：移位寄存器时钟线
6. STCP（store clock input）：串行数据锁存到并行端口时钟线
7. DS（serial data input）：串行数据线
8. 74HC595串行数据转并行数据原理

当移位时钟SHCP从低电平跳变成高电平时（也就是上升沿），DS上的1个位被读取到74HC595的移位寄存器里的第1个位上，原来的数据后移1位，最高位从Q7＇溢出。如果Q7＇连接到另一块74HC595的DS引脚上，则被移出的数据就进入下一块74HC595中。当STCP管脚上出现上升沿时，移位寄存器中的数据会被锁存到Q0到Q7管脚上，这样就完成了串行数据转并行数据。

数据从移位寄存器锁存到并行端口后可以接着继续移位，而移位寄存器中的数据不会影响并行端口的数据，只有再次进行锁存操作才会从移位寄存器中刷新数据到并行端口。

1. 移位寄存器快速清0

使用MR线可以对移位寄存器进行快速清0，但是并不影响并行端口的数据。必须对移位寄存器清0后产即进行锁存操作，才能将并行端口的数据清0。

1. Stm32使用SPI控制74HC595的一些设置
2. 由于SPI1挂在APB2上，APB2默认情况下工作在最大频率72MHz下，而stm32的SPI允许的最大工作频率为18MHz，所以SPI1的时钟分频数要大于等于4。SPI2、SPI3挂在APB1上，APB1默认情况下工作在最大频率36MHz下，所以SPI2、SPI3的时钟分频数要大于等于2。
3. 设置CPOL为High，使用空闲时时钟线为高电平
4. 设置CPHA为2Edge，使用数据的传送发生在每个时钟周期的第2个跳变沿。
5. 设置数据传送的位顺序为MSB，即从高位到低位依次传送。