

嵌入式系统原理及实验

顾 震

信息科学与工程学院自动化系

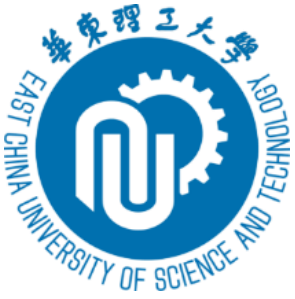
华东理工大学

[Email: guzhen@ecust.edu.cn](mailto:guzhen@ecust.edu.cn)

课程大纲

1. 嵌入式系统导论
2. Cortex-M3微处理器
3. STM32最小系统及开发环境
4. 嵌入式C语言
5. 通用输入输出GPIO模块
6. 中断
7. 定时器原理与应用
8. USART通信原理及实现
9. DMA控制器
10. SPI与I2C通信原理及实现
11. 模数转换原理及实现
12. 人工智能辅助的嵌入式项目开发
13. 嵌入式应用前沿





2. Cortex-M3微处理器

本章知识与能力要求

- ◆ 理解和掌握Cortex-M3的内核结构、寄存器、存储结构及工作模态；
- ◆ 理解微处理器内核流水线工作机制；
- ◆ 理解中断与异常的概念，以及Cortex-M3的中断处理机制；
- ◆ 了解Cortex-M3调试基本原理；
- ◆ 了解Cortex-M3内核特性。

2. Cortex-M3微处理器

2.1 内核结构

2.2 流水线

2.3 系统总线

2.4 寄存器

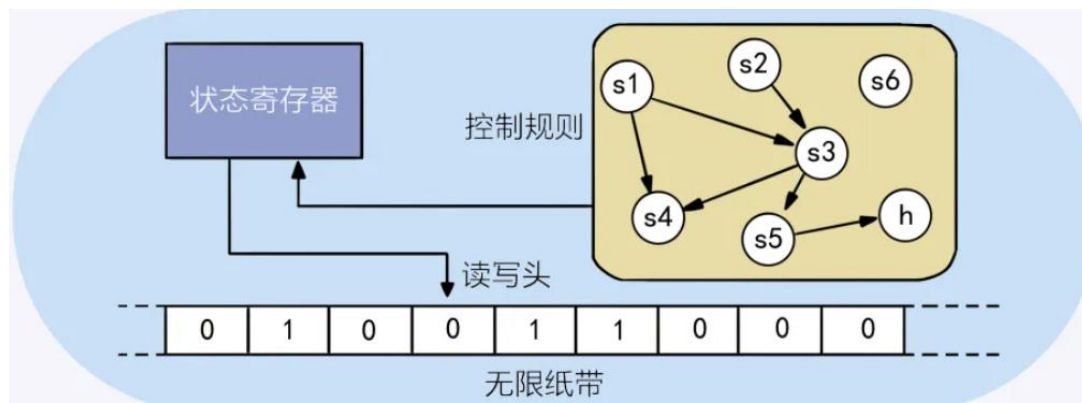
2.5 存储结构

2.6 工作模态

2.7 中断与异常

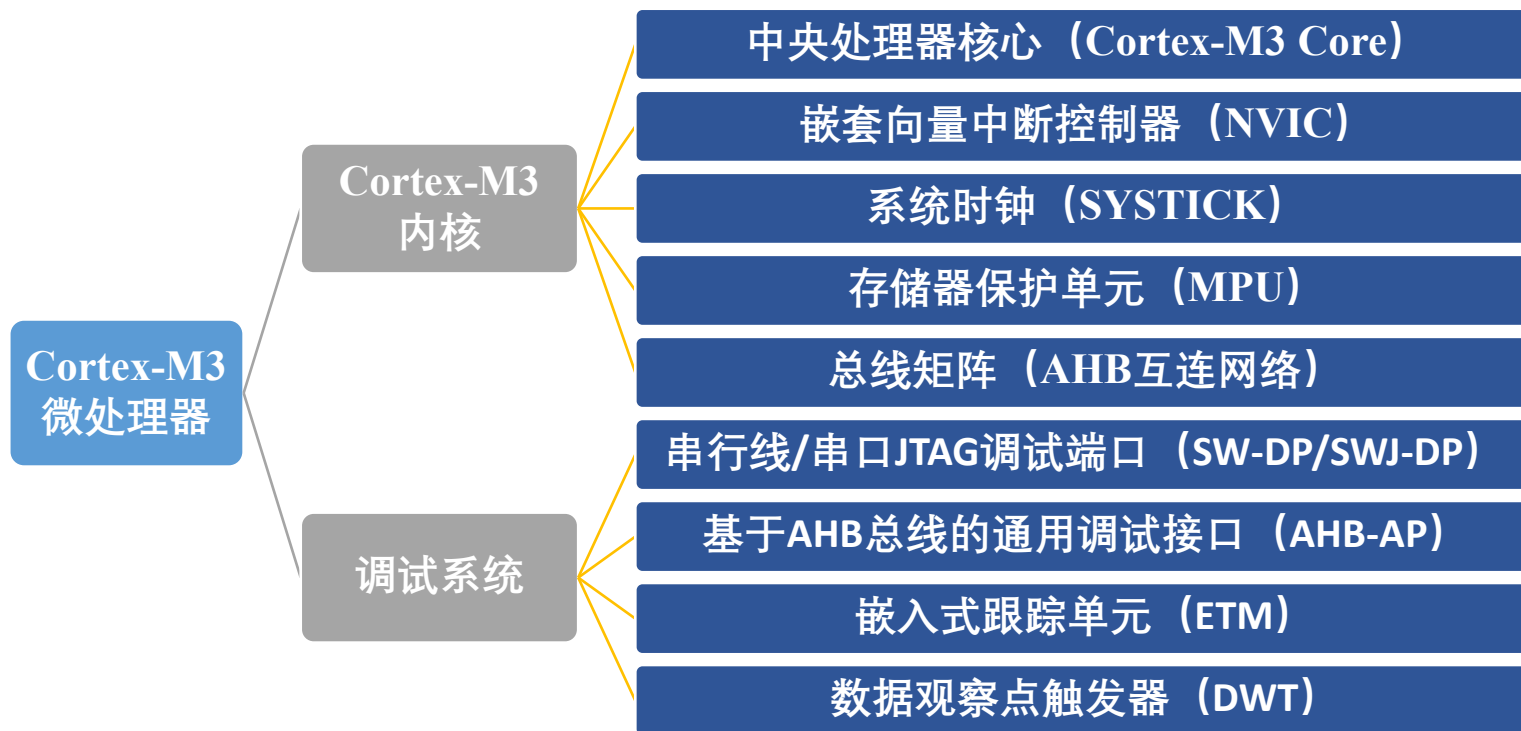
2.8 调试与跟踪

2.9 内核的其他特性



2.1 内核结构

- Cortex-M3处理器内核是建立在一个高性能**哈佛结构**的**三级流水线**技术上的ARMv7架构，可满足事件驱动的应用需求。
- 32位处理器内核(内部数据路径、寄存器、存储器接口均32位)。
- 指令总线 and 数据总线共享同一个存储器空间，寻址能力为4G。
- Thumb-2指令集(16位、32位混合指令集)。



2.1 内核结构

Cortex-M3内核

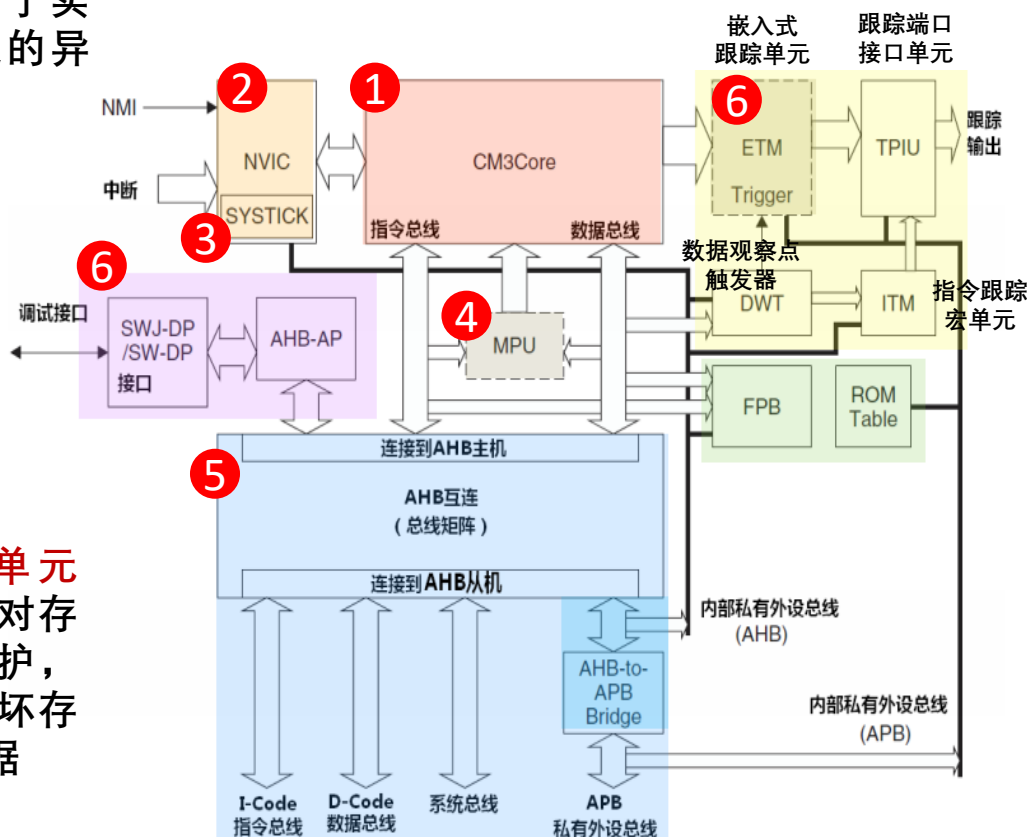
1. Cortex-M3 Core: 包括指令提取单元、译码单元、寄存器组和算术逻辑单元等

2. 嵌套向量中断控制器 (NVIC) : 用于实现快速、低延迟的异常中断处理

3. 系统时钟 (SYSTICK) : 24位倒计时计数器, 可作为系统定时器, 用于产生定时中断

4. 存储器保护单元 (MPU) : 用于对存储区域的访问保护, 防止用户程序破坏存储区域的关键数据

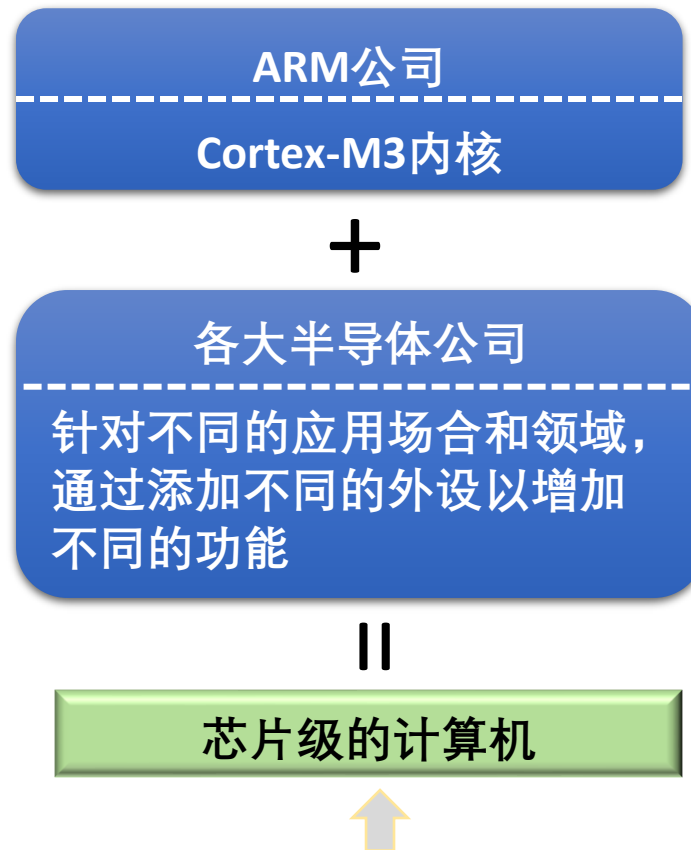
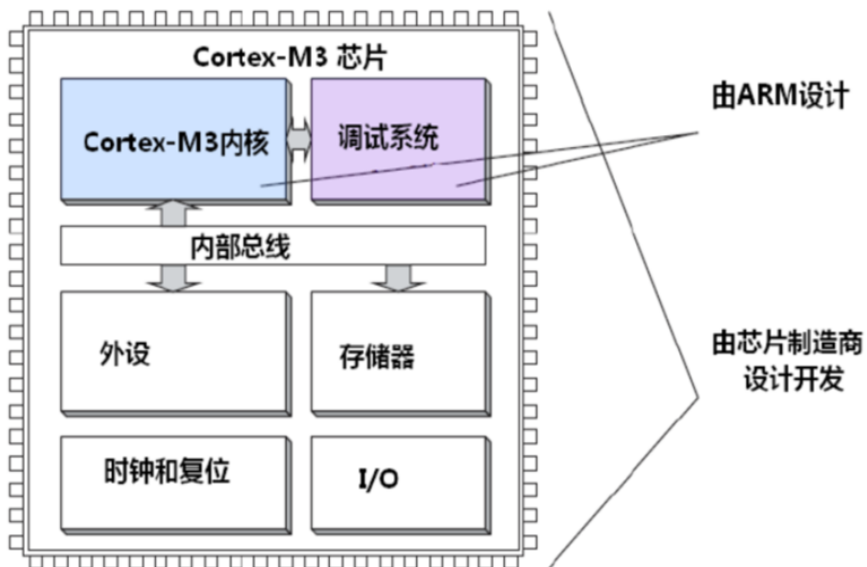
6. 调试系统: 支持对 M3 内核及外设的程序运行状态监控、断点 / 单步执行、代码下载、寄存器与存储器访问等操作



5. 总线矩阵 (AHB互连网络) : 用来将处理器内核和调试接口连接到不同功能的外部总线上, 提供并行数据传输功能, 还提供附加数据传送功能

Cortex-M3内核架构框图

2.1 内核结构



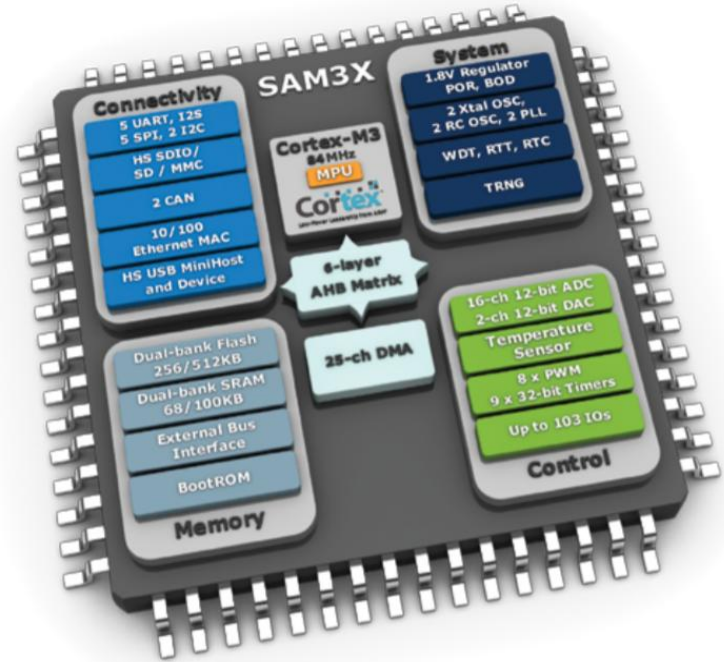
将CPU（Cortex-M3内核）、存储器、定时器/计数器、I/O等集成在一个芯片上

2.1 内核结构

ATMEL公司的SAM3X系列32位微控制器

Features:

- ARM Cortex-M3 revision 2.0 running at up to 84 MHz
- Memory Protection Unit (MPU)
- 24-bit SysTick Counter
- Thumb®-2 instruction set
- Nested Vector Interrupt Controller
- 2 x 256Kbytes embedded Flash, 128-bit wide access, memory accelerator, dual bank



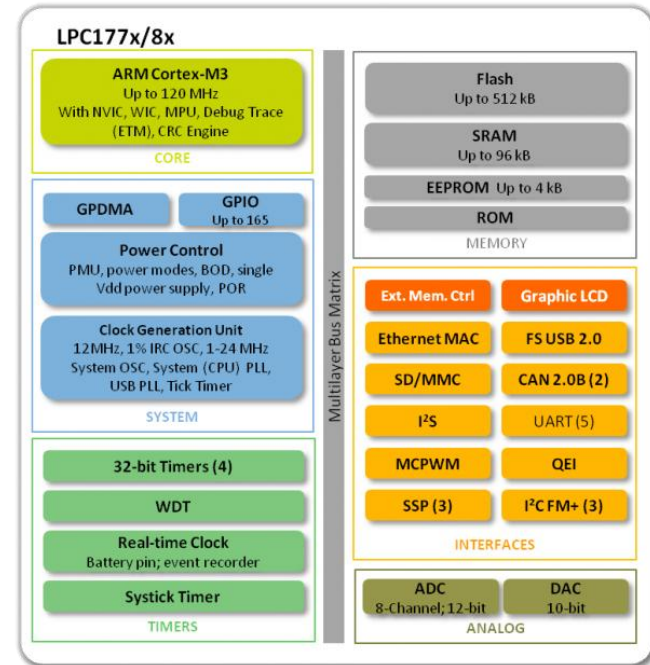
ATMEL公司的SAM3X系列MCU功能结构图

2.1 内核结构

NXP公司的LPC1700系列芯片

Key Features:

- ☀ - ARM Cortex-M3 core
- ☀ - Up to 120 MHz operation
- ☀ - Nested Vectored Interrupt Controller for fast deterministic interrupts
- ☀ - Wakeup Interrupt Controller allows automatic wake from any priority interrupt
- ☀ - Memory Protection Unit
- ☀ - Four reduced power modes: Sleep, Deep-Sleep, Power-down and Deep power-down



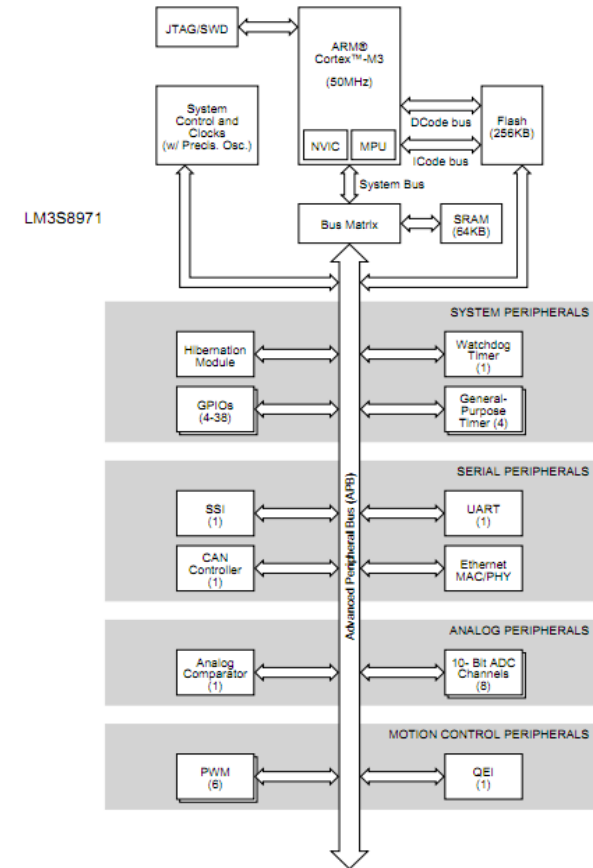
NXP公司LPC1700系列MCU功能结构图

2.1 内核结构

TI公司的Stellaris LM3S系列 微控制器LM3S8971

Features

- ✿ -32-bit ARM® Cortex™-M3 50-MHz processor core with System Timer (SysTick), integrated Nested Vectored Interrupt Controller (NVIC), Memory Protection Unit (MPU), and Thumb-2 instruction set
- ✿ - Full-featured debug solution with debug access via JTAG and Serial Wire interfaces, and IEEE 1149.1-1990 compliant Test Access Port (TAP) controller
- ✿ - 256 KB single-cycle flash and 64 KB single-cycle SRAM
- ✿ - Lower-power battery-backed Hibernation module with Real-Time Clock
- ✿ - 4-38 GPIOs (depending on configuration) with programmable control for GPIO interrupts and pad configuration



TI公司32位微控制器LM3S8971功能结构框图

2.1 内核结构

- 上述三个不同半导体公司的不同系列的芯片

相同之处	不同之处
基于ARM Cortex-M3内核的32位微控制器	SRAM和FLASH的大小
具有Cortex-M3内核的基本架构，如NVIC、JTAG/SWD调试接口	外设的不同
	I/O、时钟的不同

针对不同应用场合和各自优势领域，生产出各具特色的微控制器

2.2 流水线

- 每条指令分解为多步，并让各步操作重叠，从而实现几条指令并行处理的技术，称为**流水线技术**。

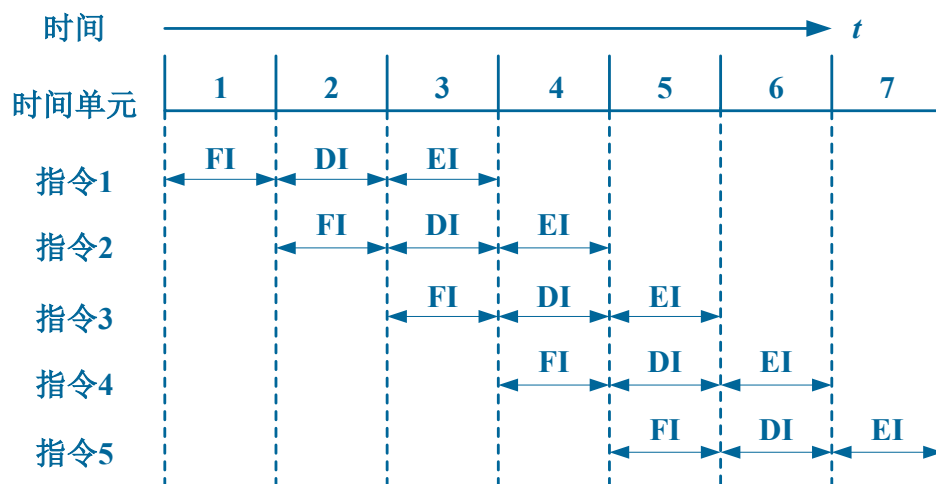
ARM Cortex-M3微控制器采用的
三级流水线：

取指—>译码—>执行

取指（FI）：将指令从存储器中取出

译码（DI）：对所取指令进行翻译

执行（EI）：执行指令



三级流水时序图

2.2 流水线



洗菜

1分钟



切菜

1分钟



炒菜

1分钟



装盘

1分钟

	流水线方式		非流水线方式
单独做一道菜	4+分钟	≈	4分钟
做四道菜	7+分钟	<	16分钟
连续工作	每1+分钟上一道菜	<	每4分钟上一道菜

≈4倍

2.2 流水线

流水线的技术指标通常用吞吐率、加速比和效率三项指标来衡量

吞吐率

单位时间内流水线所完成的指令数或输出结果的数量

流水线执行n条指令所需时间：

所需时间 = 完成一条指令所需时间 +
(指令条数n-1) × 时间最长的指令段

吞吐率表示为：

吞吐率 = 指令条数 / 指令流水线所需时间

单位：指令 / 秒 (IPS, Instructions Per Second)

百万指令 / 秒 (MIPS)、十亿指令 / 秒 (GIPS)

2.2 流水线

练习：现有一个三级指令流水线分为取指、译码和执行三个部分，其中，取指周期为1ns，译码指令周期为3ns，执行指令周期为2ns，则完成100条指令其吞吐率为多少？

解：由吞吐率的计算公式可知，需先计算出执行100条指令流水所花费的时间。

完成第一条指令需要 $1+3+2=6\text{ns}$ ，即若不采用指令流水操作，完成一条指令需6ns。由于译码指令所用时间最长为3ns，那么从第二条指令开始，每隔3ns就能完成一条指令，完成100条指令所需要的流水时间为：

$$6\text{ns} + (100-1) \times 3\text{ns} = 303\text{ns},$$

则吞吐率为：

$$100/303 \approx 0.33 \text{ 指令/ns} = 3.3 \times 10^8 \text{ IPS} = 330 \text{ MIPS}$$

2.2 流水线

加速比

完成一批任务不使用流水线时间和使用流水线所用时间之比。

若流水线各段时间均为 Δt ，流水线级数 m ，采用流水线方式完成 n 条指令共需 $m\Delta t + (n-1)\Delta t$ 时间，而不采用流水线方式完成 n 条指令所需时间为 $nm\Delta t$ ，则加速比为：

$$S_p = \frac{nm\Delta t}{m\Delta t + (n-1)\Delta t}$$

2.2 流水线

不平衡的流水线



洗菜

1分钟



切菜

2分钟



炒菜

1分钟



装盘

1分钟

	流水线方式		非流水线方式
单独做一道菜	8+分钟	>	5分钟
做四道菜	14+分钟		20分钟
连续工作	每2+分钟上一道菜	<	每5分钟上一道菜

≈2.5倍

2.2 流水线

平衡的流水线

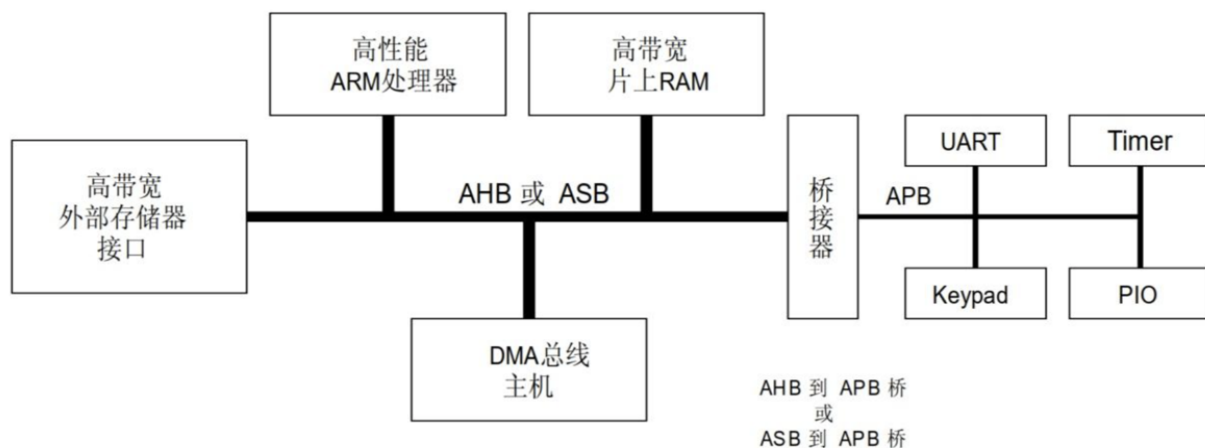


	流水线方式	非流水线方式
单独做一道菜	5+分钟	5分钟
做四道菜	8+分钟	20分钟
连续工作	每1+分钟上一道菜	每5分钟上一道菜

≈5倍

2.3 系统总线

- 计算机系统中，各个部件之间传送信息的公共通路叫**总线** (Bus)。
- 总线分类：**数据总线、地址总线、控制总线**
- AMBA是1996年由ARM公司研发推出的一种**高级微控制器总线架构**(Advanced Microcontroller Bus Architecture)。
- AMBA包含了四种不同的总线标准，分别是：
 - **AHB (Advanced High-performance Bus)** 高级高性能总线
 - **APB (Advanced Peripheral Bus)** 高级外设总线
 - ASB (Advanced System Bus) 高级系统总线 (用于低性能系统，已过时)
 - AXI (Advanced eXtensible Interface) 高级可扩展接口 (AMBA3)

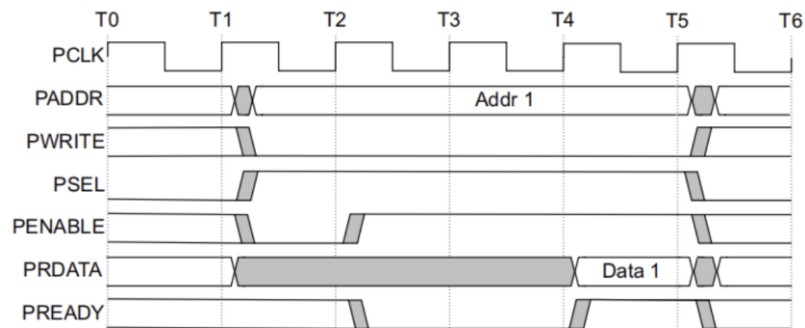
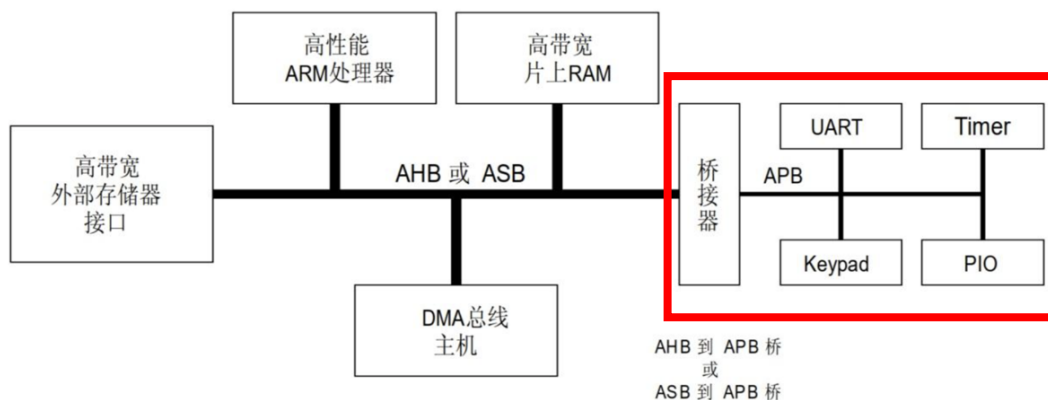


2.3 系统总线

APB高级外设总线

是一种**低成本、低功耗、结构简单**的低速通信总线。

- APB总线通信有主从设备之分，APB主设备通常是APB桥,从设备通常是一些低速的外设。

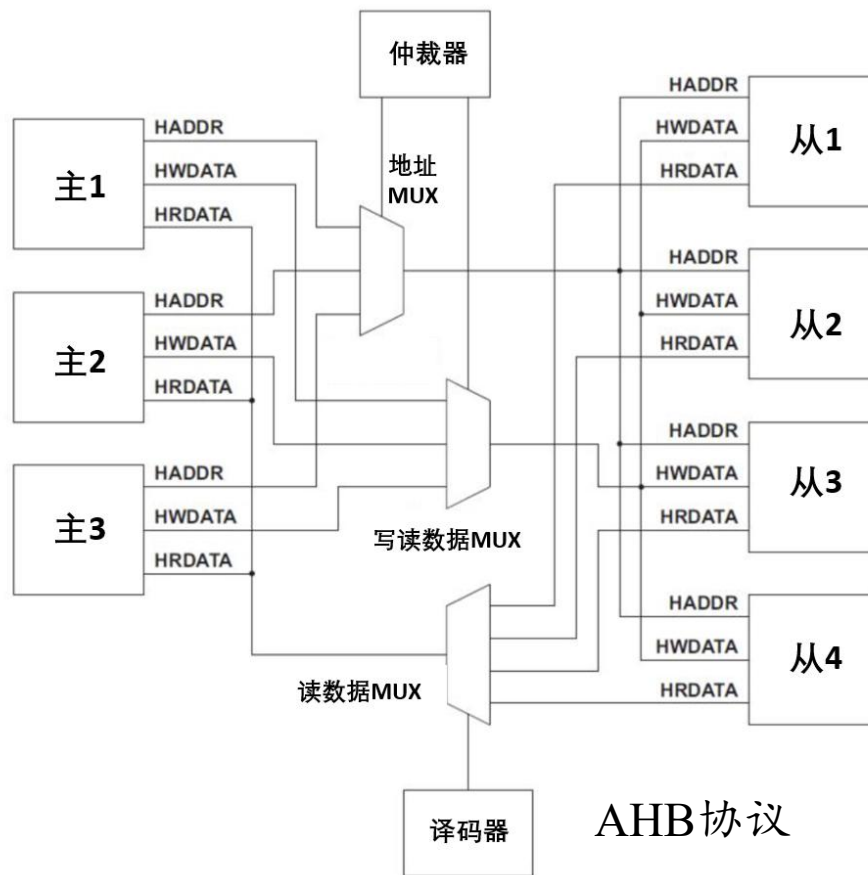


APB时序图

2.3 系统总线

AHB高级高性能总线

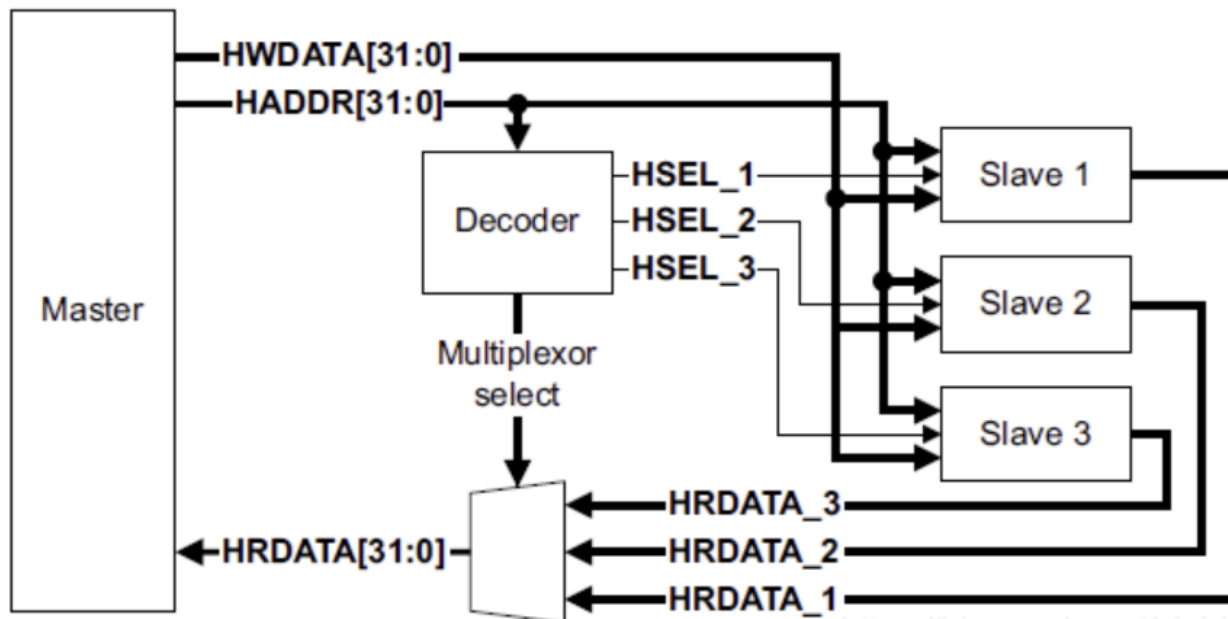
➤ AHB协议用于高性能模块之间的连接，可以连接**多个主机和多个从机**。



2.3 系统总线

AHB-lite协议

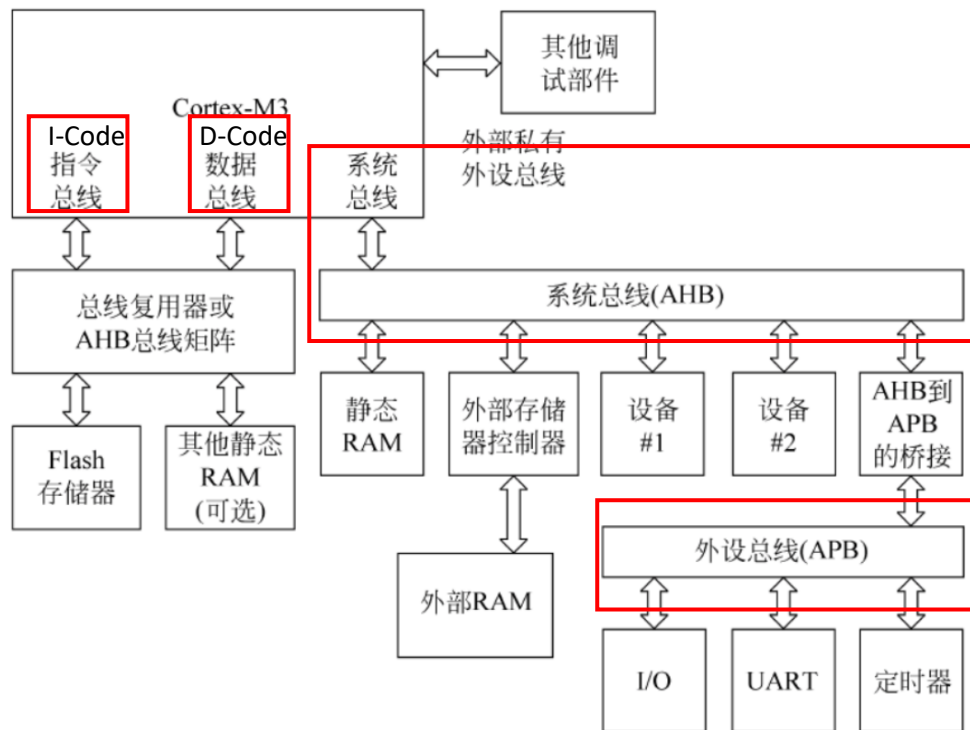
- ARM公司在2006年推出了AMBA3，其中包括了AHB-lite协议，其针对的是单个主机和多个从机之间的连接。



AHB-lite协议

2.3 系统总线

➤ Cortex-M3总线结构



I-Code 指令总线

- 基于AHB-Lite总线协议的32位总线
- 用于取指操作

D-Code 数据总线

- 基于AHB-Lite总线协议的32位总线
- 用于数据访问操作

系统总线

- 基于AHB-Lite总线协议的32位总线
- 用于访问内存和外设

外设总线

- 基于APB总线协议的32位总线
- 用于访问私有外设

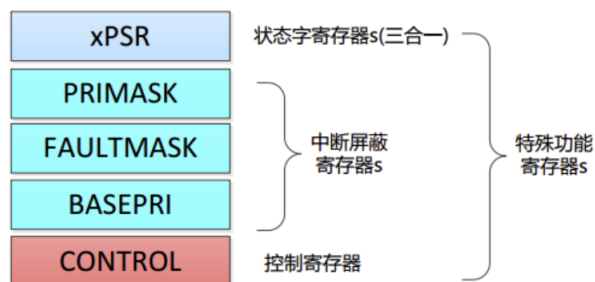
2.4 寄存器

寄存器

有限存储容量的高速存储部件，它们可用来暂存指令、数据和位址。



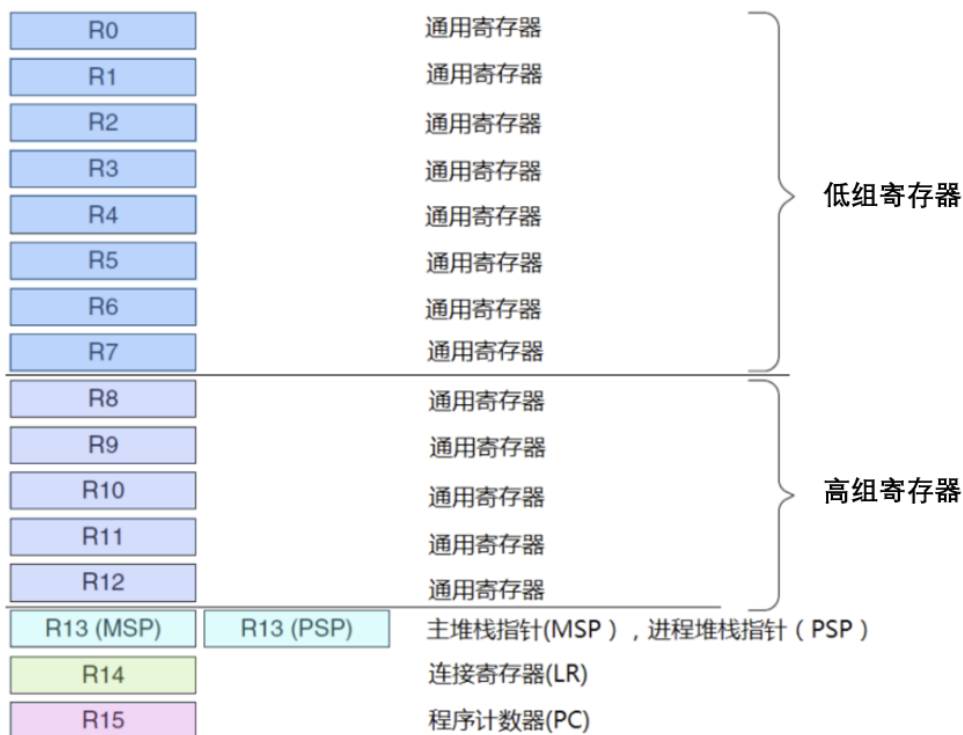
通用寄存器：
保存指令执行过程中临时存放的寄存器操作数和中间结果



特殊功能寄存器：
用于外设控制相关的操作

2.4 寄存器

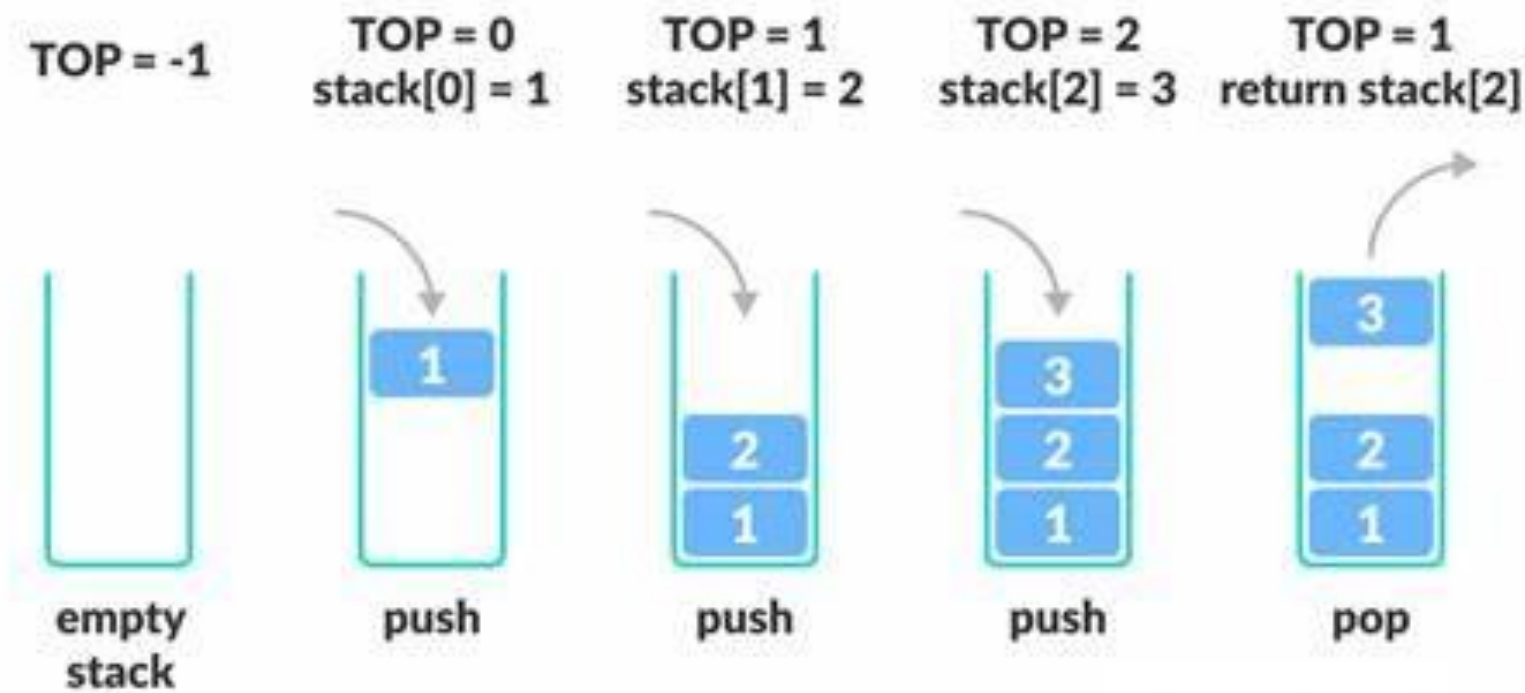
通用寄存器



- 13个32位的通用寄存器R0~R12
- R0-R7 也被称为低组寄存器。所有指令都能访问它们。
- R8-R12 也被称为高组寄存器。只有很少 16 位 Thumb指令能访问它们，32位的指令则不受限制
- R13、R14、R15三个寄存器被指定了专门的用途

2.4 寄存器

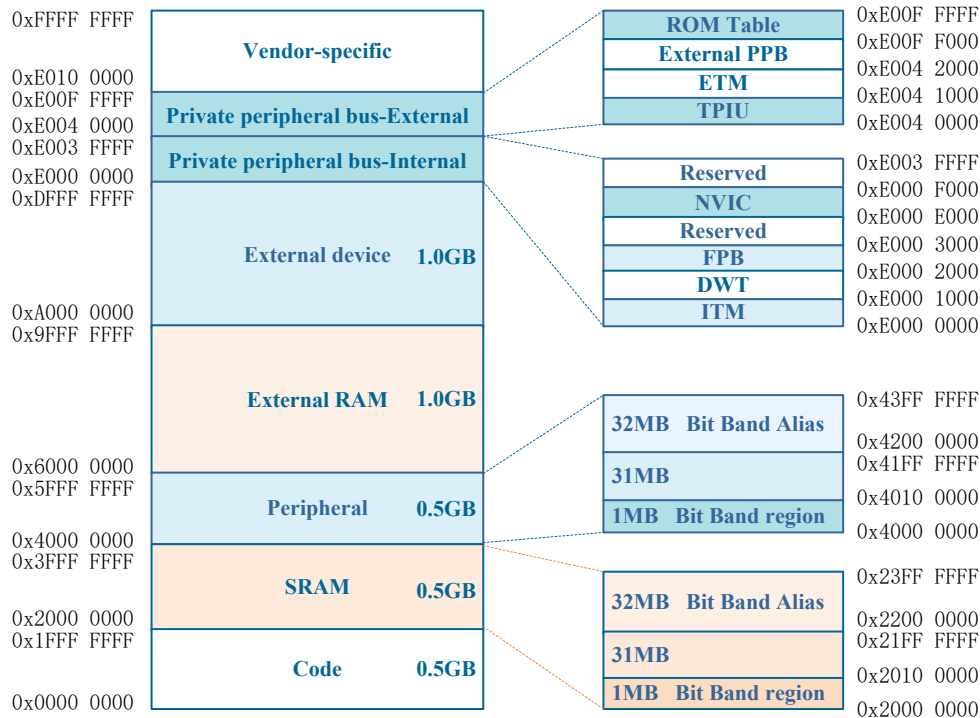
堆栈(Stack)是一种存储器的使用模型，由一块连续的内存，以及一个栈顶指针组成。



先进后出(FILO, First In Last Out)



2.5 存储结构

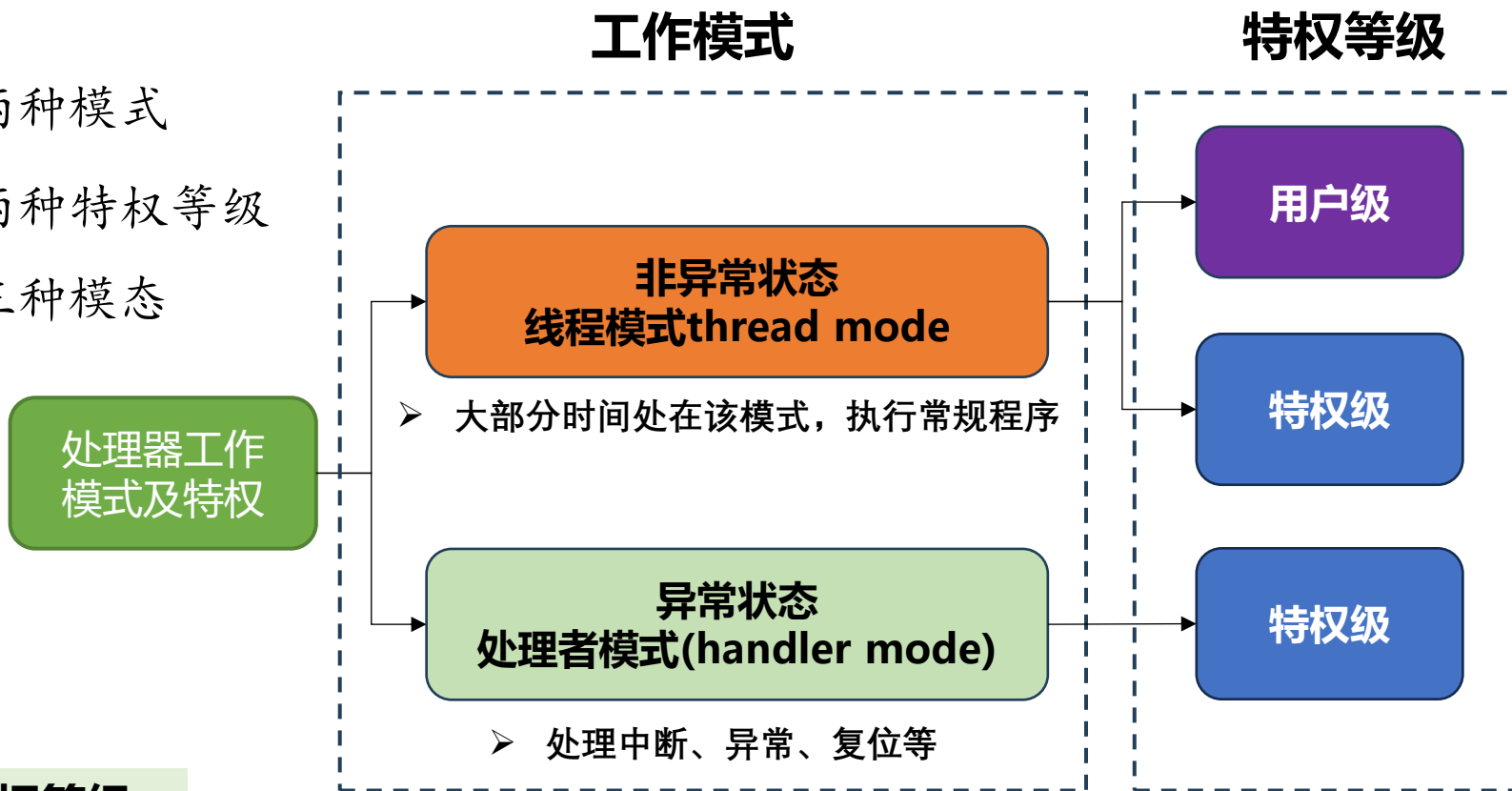


Cortex-M3存储器映射

- ARM Cortex-M3系列微处理器采用存储器与I/O设备（外设）**统一编址**的方式，设置部分存储器地址范围用于外设的访问，将这种通过存储器地址访问外设的方式，称为**存储器地址映射**。
- 存储器映射将芯片中或芯片外的**FLASH、RAM**和外设接口等进行**统一编址**，用于程序存储、数据存储、寄存器和外设。

2.6 工作模式

- 两种模式
- 两种特权等级
- 三种模态

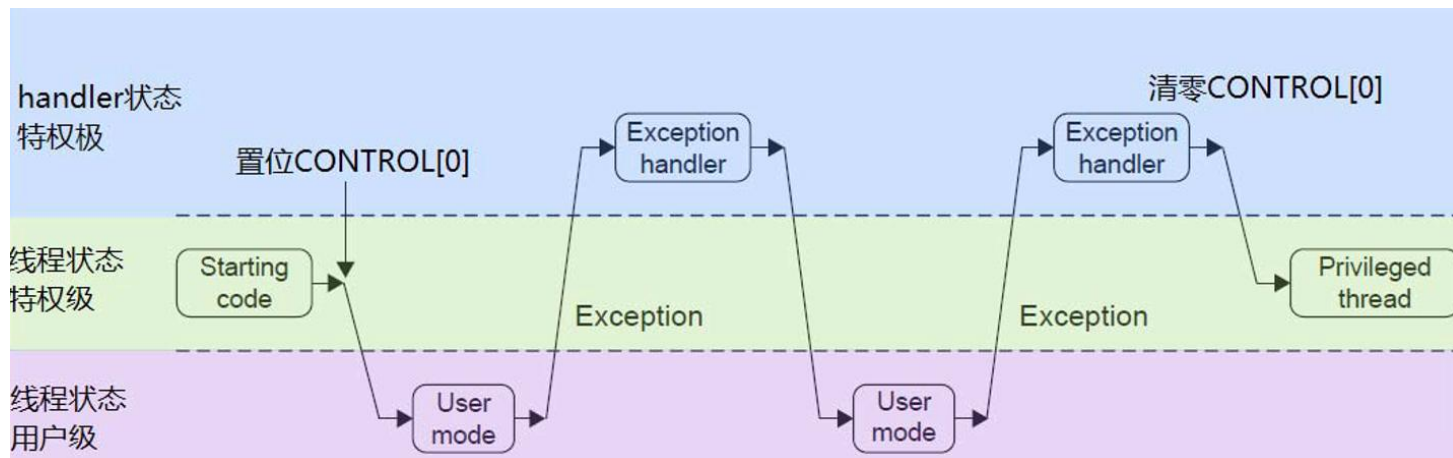
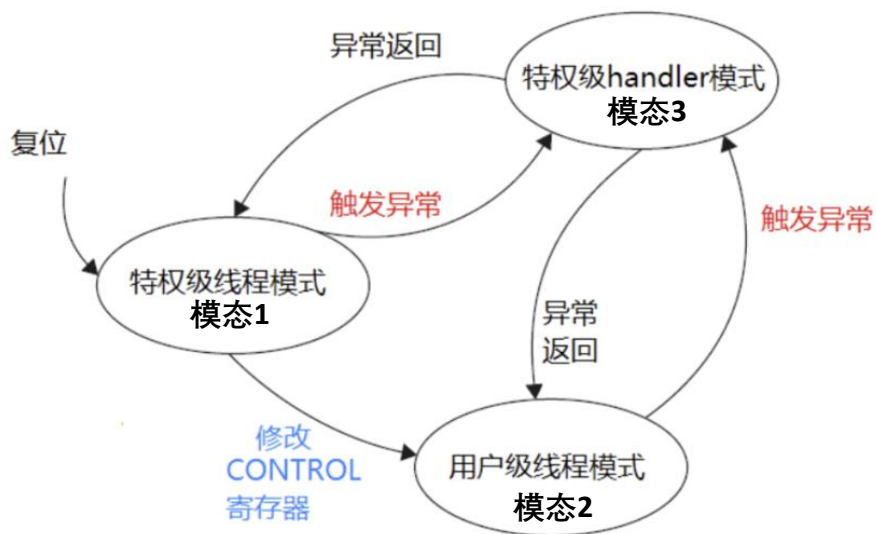


特权等级

- 对存储器访问提供了一种保护机制
 - 在特权级下, 程序可以**访问所有范围的存储器**, 并且能够执行所有指令。
 - 用户级下, **不能访问系统控制空间**(包含配置寄存器及调试组件的寄存器), 且禁止使用MSR访问特殊功能寄存器(APSR除外), 如果访问, 则产生fault。

2.6 工作模态

Cortex-m3微处理器各工作模态的互动关系



用户级下无法通过修改CONTROL[0]来回到特权级

2.7 中断与异常

中断响应过程

1. 保存现场

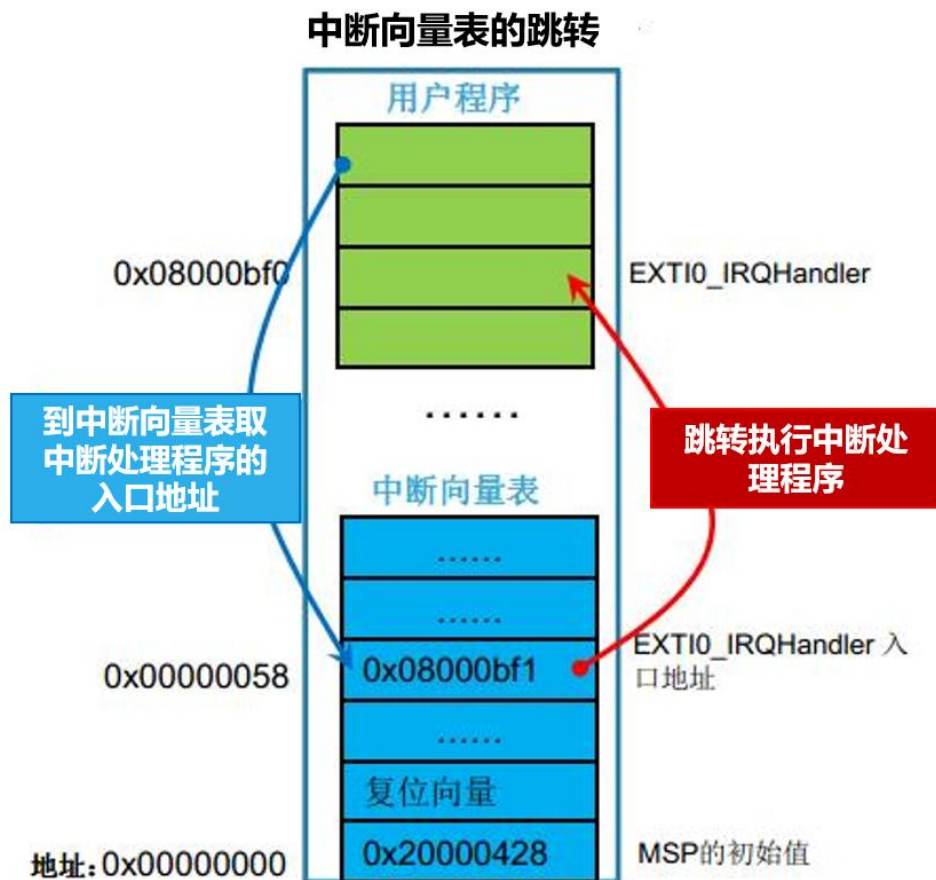
- 处理器将相关的寄存器xPSR、PC、LR、R12及R0~R3压入堆栈中。如果当前使用的是PSP，则压入进程堆栈，否则压入主堆栈。

2. 取向量

- 从中断向量表中找到相应的中断服务程序的入口地址。

3. 更新寄存器

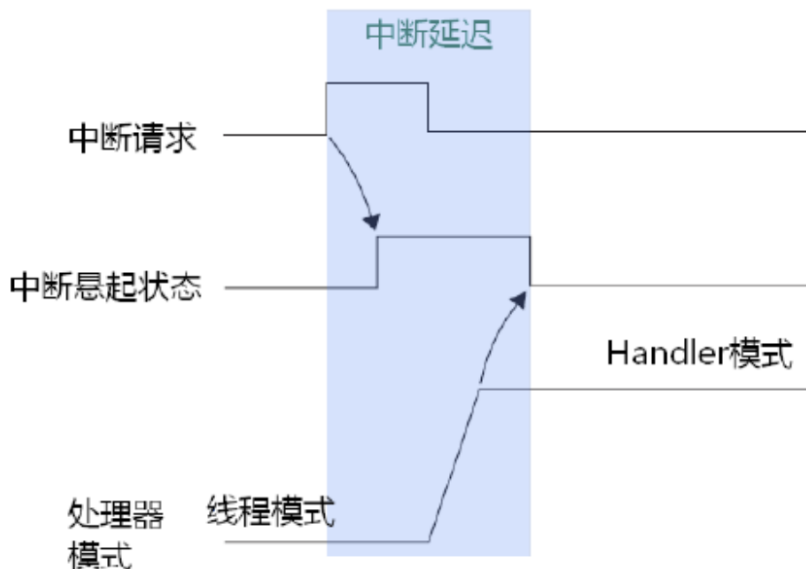
- 将堆栈指针进行更新，在向量取出后，程序寄存器指针（PC）指向中断服务程序的入口地址，更新程序寄存器指针（PC）以及更新链接寄存器（LR）。



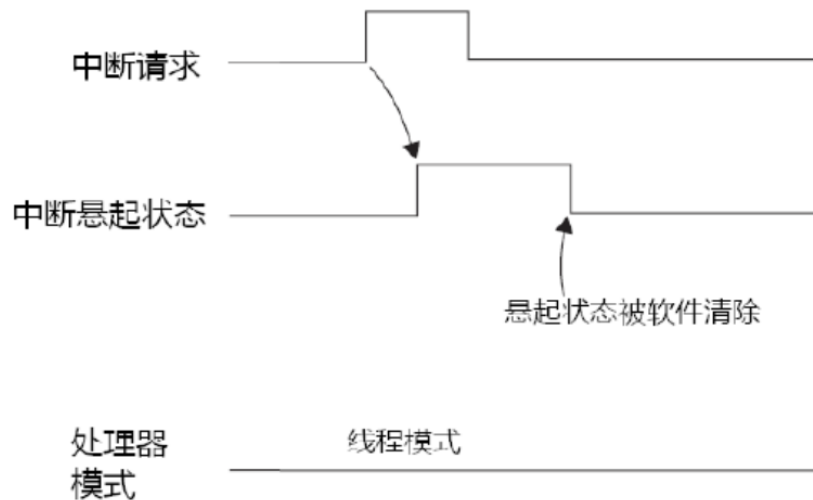
2.7 中断与异常

中断悬起

如果中断发生时，正在处理同级或高优先级异常，或者被掩蔽(primask、faultmask、basepri中断屏蔽寄存器)，则中断不能立即得到响应，此时**中断被悬起**。



中断悬起

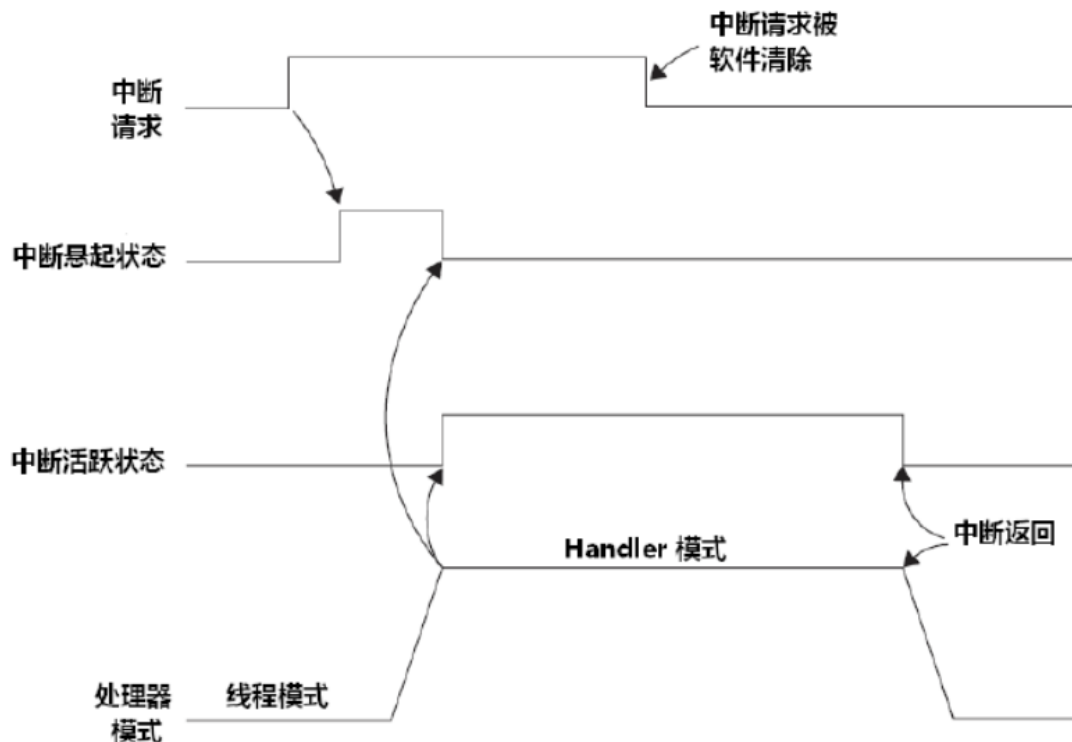


中断在处理器响应前被清除悬起状态

2.7 中断与异常

中断活跃

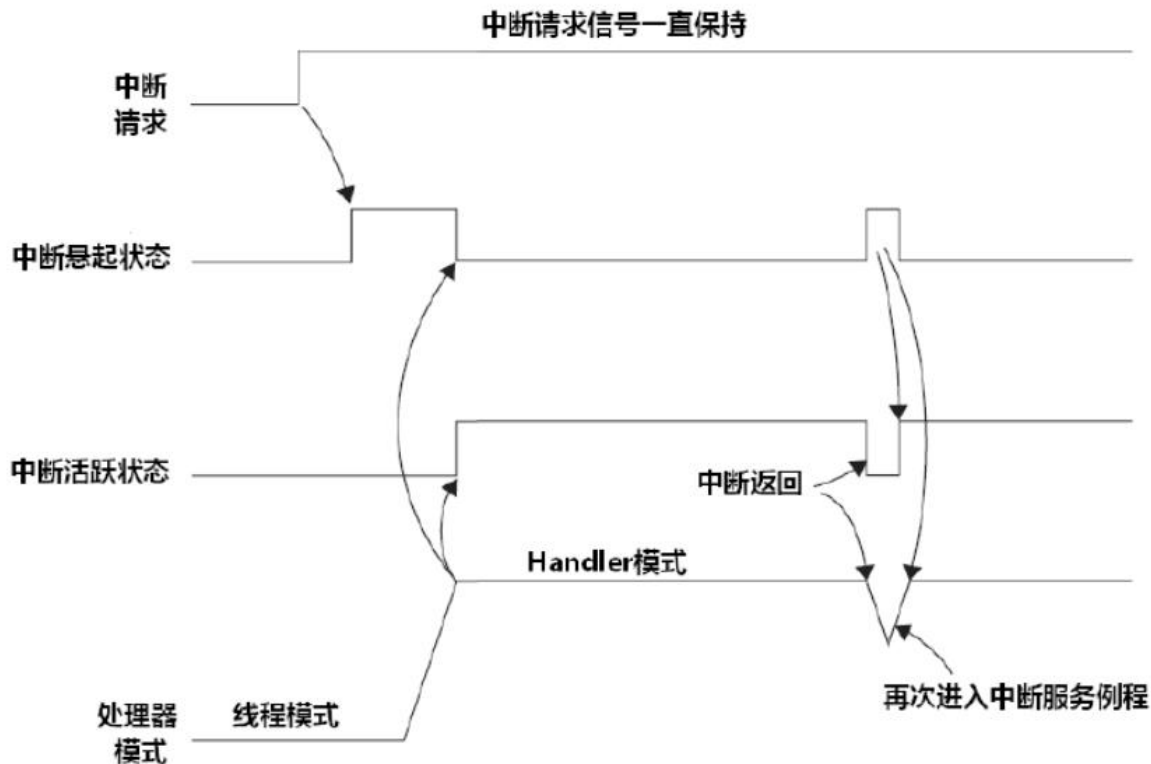
当某中断的服务例程开始执行时，中断进入活跃状态，其悬起位被硬件清除，中断退出后才能对该中断的新请求予以响应，同时硬件清除活跃标志位。



2.7 中断与异常

中断请求信号保持

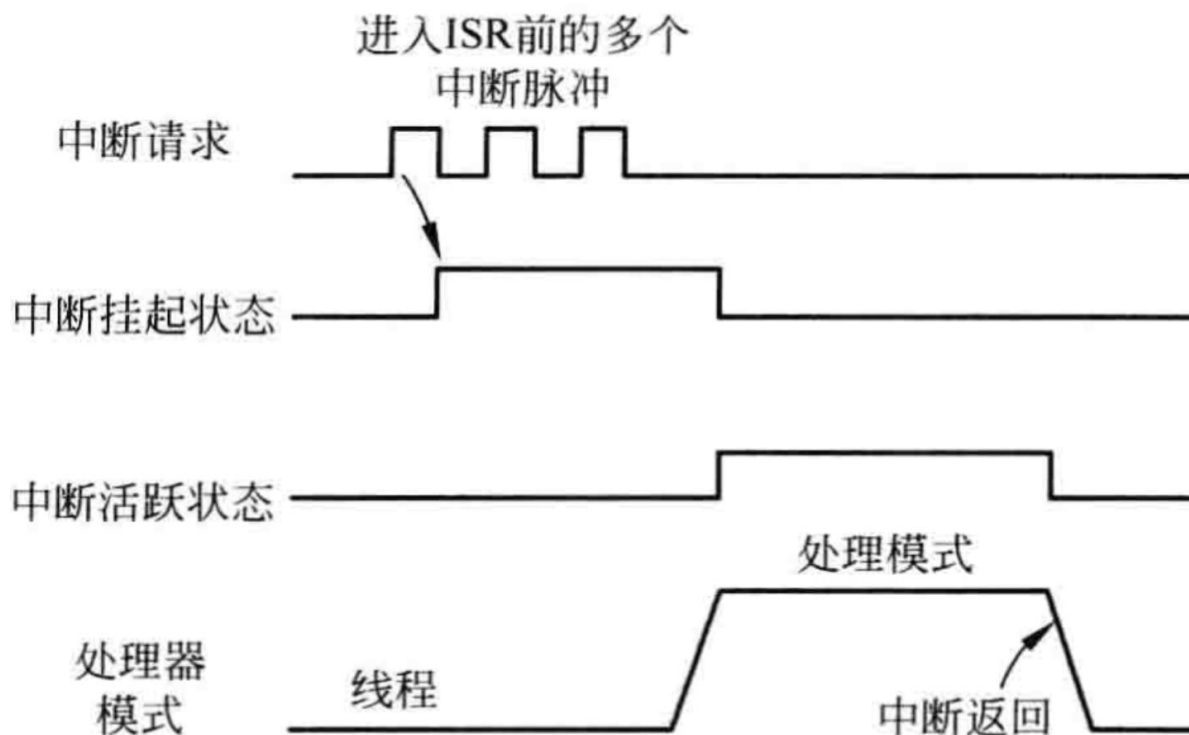
如果中断源咬住请求信号不放，该中断就会在其上次服务例程返回后再次被置为悬起状态。



2.7 中断与异常

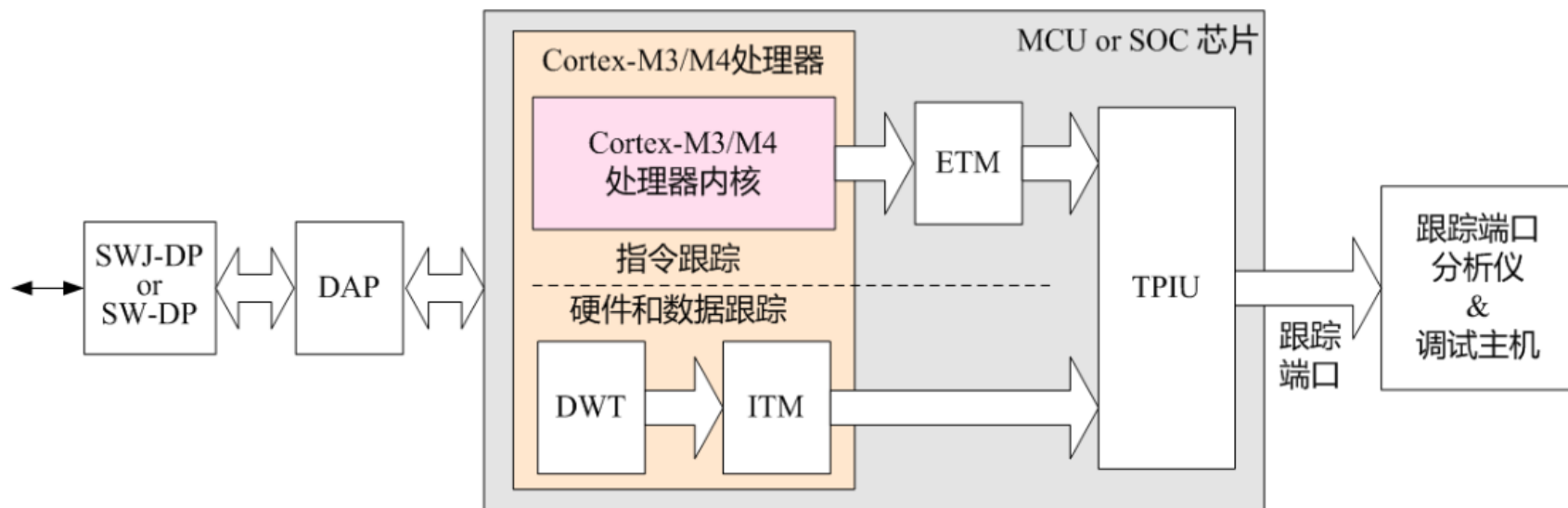
中断请求信号过快

中断在得到响应之前，其请求信号以**多个脉冲**形式呈现，则被视为只有一次中断请求，多出的请求脉冲全部错失



2.8 调试与跟踪

- CoreSight调试架构包括调试接口协议、调试总线协议、对调试组件的控制、安全特性、跟踪接口等。



2.8 调试与跟踪

调试访问端口(Debug Adapter Protocol , DAP)

- 用于Cortex-M3调试系统对处理器上总线逻辑的控制。
- 通过把ITAG或串行线协议都转换成DAP总线接口协议,再控制DAP来执行调试动作。DAP总线与APB总线很相似,很容易地连接其他调试组件,把调试接口和调试硬件分开,使得调试系统规模可调整,伸缩性很强。

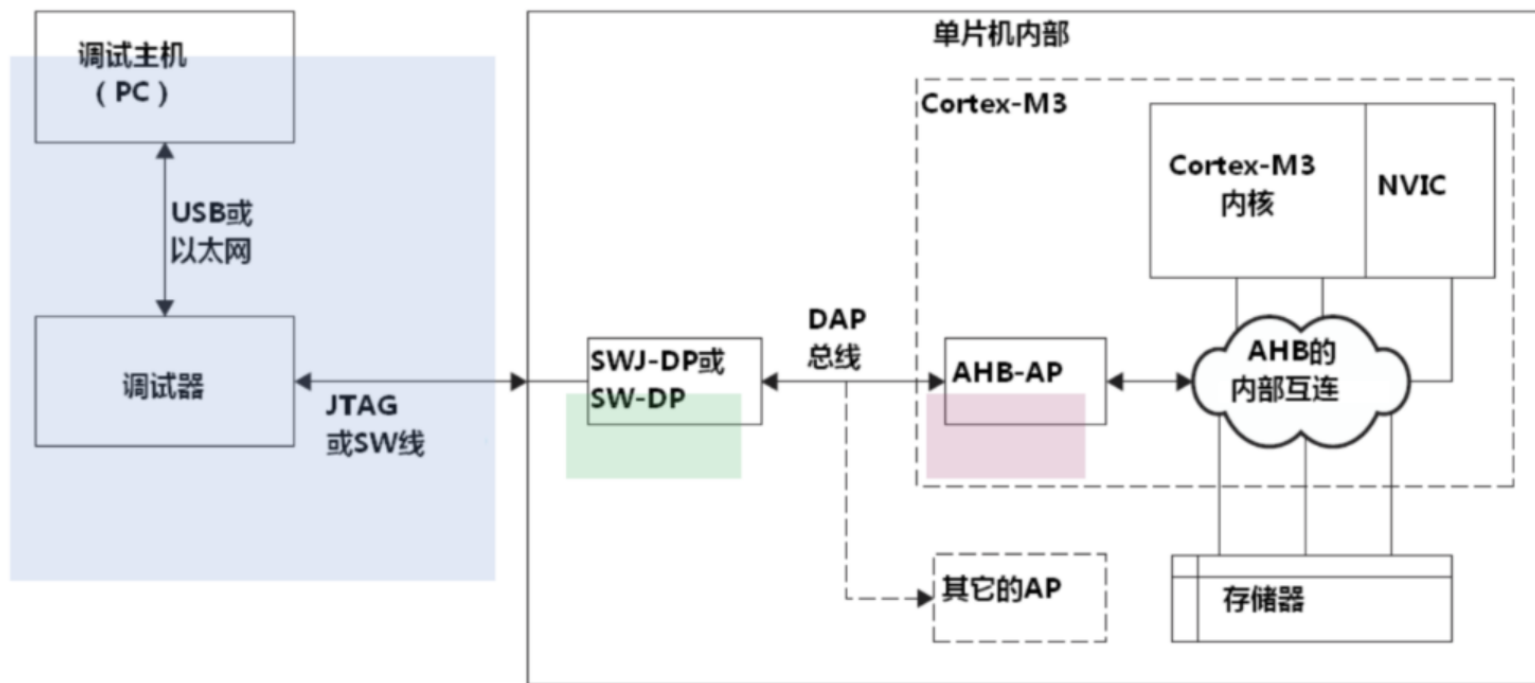
两种调试主机接口

1. JTAG 接口
2. 串行线(Serial Wire, SW)调试接口

2.8 调试与跟踪

调试接口

- DP (Debug Adapter) 充当处理器与调试器的中介,它的一端连接到调试器上,另一端则连接到 Cortex-M3 的 DAP 接口上。



2.8 调试与跟踪

- Cortex-M3基于CoreSight构建的调试与跟踪系统提供了多种多样的调试组件，能实现丰富的调试功能

侵入式调试

- a、停机以及单步执行程序
- b、硬件断点
- c、断点指令BKPT
- d、数据观察点作用于单一地址、一个范围的地址以及数据的值。
- e、访问寄存器的值既包括读也包括写
- f、调试监视器异常
- g、基于ROM的调试闪存地址重载 (flash patching)

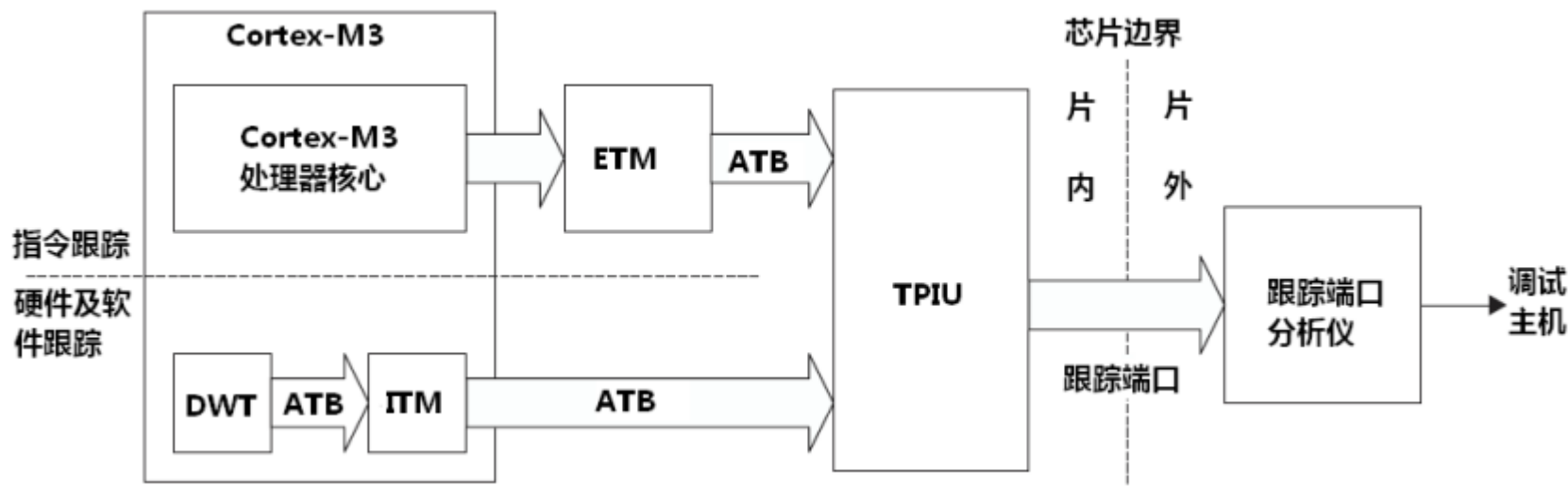
非侵入式调试

- a、在内核运行的时候访问存储器
- b、指令跟踪需要通过可选的嵌入式跟踪宏单元ETM
- c、数据跟踪
- d、软件跟踪通过ITM指令跟踪单元
- e、性能速写profiling通过数据观察点以及跟踪模块

2.8 调试与跟踪

跟踪接口

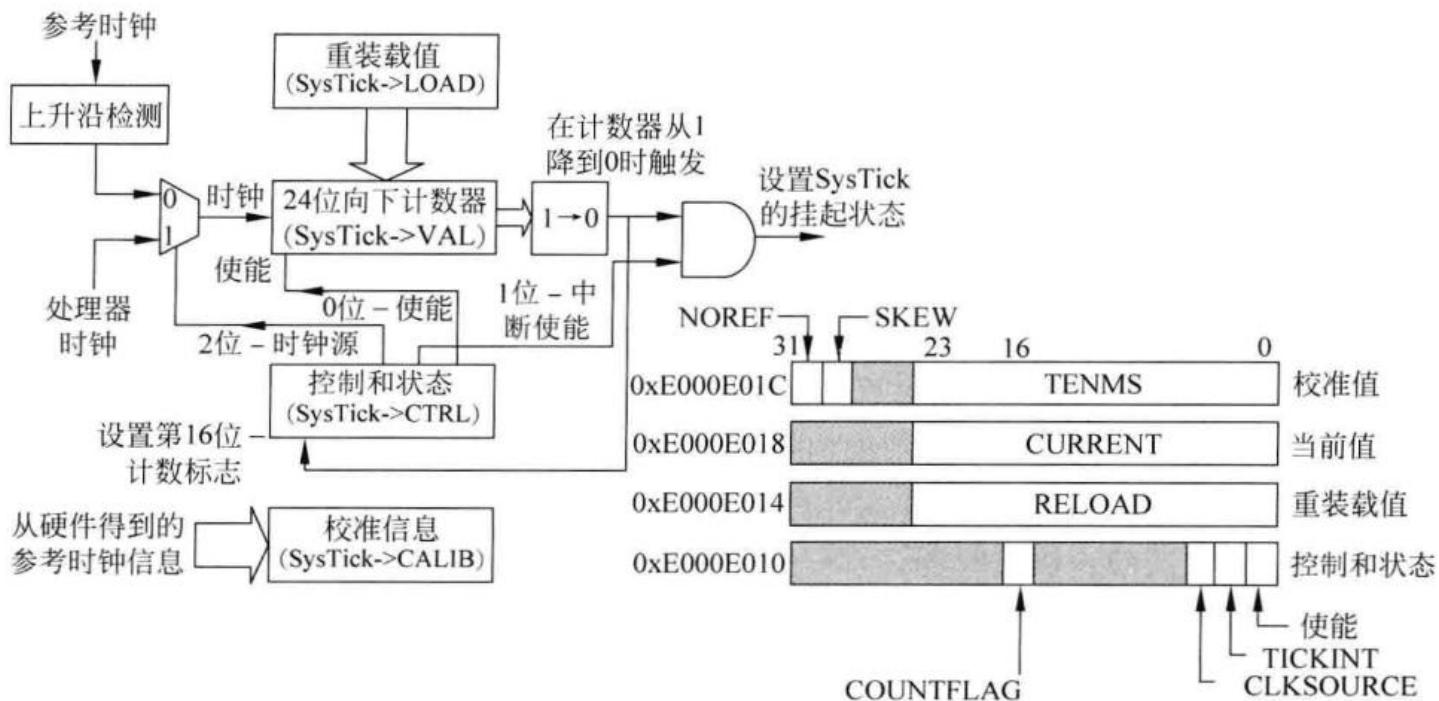
- CoreSight架构还可以用于**数据跟踪**。跟踪源产生的数据被封装成数据包,数据包长度可变。这些包通过“**高级跟踪总线**”(ATB)送往TPIU(跟踪端口接口单元),由TPIU 把它们格式化,转换成符合“跟踪总线接口协议”的数据包,再把数据导出到片外的跟踪硬件设备。



2.9 内核的其他特性

SysTick 定时器

- SysTick是一个24位的倒计时定时器,当计到0时,将从RELOAD寄存器中自动重载定时初值。只要不把它在SysTick控制及状态寄存器中的使能位清除,就永不停息。



2.9 内核的其他特性

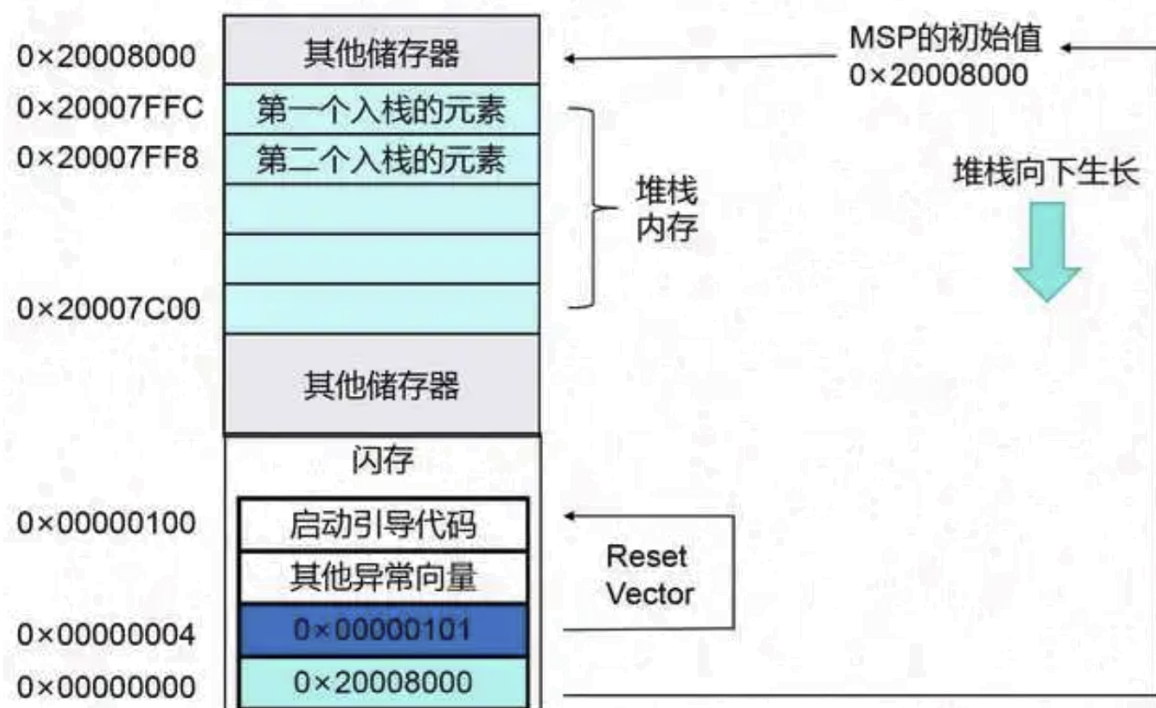
电源管理

- Cortex-M3对内核级上提供了电源管理, 有两种睡眠模式。在睡眠时, 可以停止系统时钟, 但可以让FCLK继续走, 以允许处理器能被SysTick异常唤醒。这两种睡眠模式为:
 - (1)睡眠: 由Cortex-M3处理器的 SLEEPING 信号指示。
 - (2)深度睡眠: 由Cortex-M3处理器的SLEEPDEEP 信号指示。
- 可以通过读取NVC的相关系统控制寄存器判定当前使用的睡眠模式及睡眠时的上下文。

2.9 内核的其他特性

复位

- 在复位时，CortexM3会首先读取下列两个32位整数的值。
 1. 0x00000000处取MSP（主栈指针）初始值（定栈）
 2. 0x00000004处取PC（程序计数器）初始值（定代码入口）



本章小结

- 2.1 内核结构
- 2.2 流水线
- 2.3 系统总线
- 2.4 寄存器
- 2.5 存储结构
- 2.6 工作模态
- 2.7 中断与异常
- 2.8 调试与跟踪
- 2.9 内核的其他特性