第一章 绪论

1.嵌入式系统的定义
嵌入式系统是嵌入到对象体系中的，用于执行独立功能的专用计算机系统
以应用为中心，以计算机、通信、控制等技术为基础采用可裁剪软硬件，适用于对功能、可靠性、成本、体积、功耗等有严格要求的专用计算机系统。

2.嵌入式系统三要素
（1）嵌入性　　->满足环境要求
（2）专用性　　->满足配置要求
（3）计算机系统　　->满足控制要求

3.嵌入式系统与桌面通用系统的区别
（1）嵌入式系统中运行的任务是专用而确定的
（2）桌面通用系统需要支持大量的、需求多样的应用程序
（3）嵌入式系统往往对实时性提出较高的要求
（4）嵌入式系统对可靠性要求高
（5）嵌入式系统有功耗约束
（6）嵌入式系统内核小，可用资源少
（7）嵌入式系统的开发需要专用工具和特殊方法
开发：交叉编译、交叉链接

4.嵌入式系统的通用结构
嵌入式系统一般由嵌入式处理器、外围硬件设备、嵌入式操作系统（可选），以及用户的应用软件系统等四个部分组成。

第二章 ARM 微处理器与编程模式

1.计算机体系结构

（1）冯*诺伊曼结构

核心思想：将程序和数据存放在同一存储器的不同地址；顺序执行指令；执行过程：取指令（or 数据）->分析指令->执行指令。

（2）哈佛结构

核心思想：将程序和数据存放在不同的存储器中；并行执行指令；执行过程：取指令（and 数据）->分析指令->执行指令

2.计算机指令系统：

CISC:复杂指令集计算机

RISC：精简指令集计算机

类别	CISC	RISC
指令系统	指令数量很多	较少，通常少于100
执行时间	有些指令执行时间很长，如整块的存储器内容拷贝；或将多个寄存器的内容拷贝到存贮器	没有较长执行时间的指令
编码长度	编码长度可变，1-15字节	编码长度固定，通常为4个字节
寻址方式	寻址方式多样	简单寻址
操作	可以对存储器和寄存器进行算术和逻辑操作	只能对寄存器进行算术和逻辑操作，Load/Store体系结构
编译	难以用优化编译器生成高效的目标代码程序	采用优化编译技术，生成高效的目标代码程序

3.流水线技术

几条指令可以并行执行

4.ARM 的含义

三种含义

（1）一个公司的名字：英国知识产权核（IP）设计公司

（2）一类微处理器的通称

（3）一种技术的名字（ARM 微处理器核）

命名规则（选择题 有印象即可）

（1） T：支持 Thumb 指令集

（2） D:片上调试

（3） M：支持长乘法
（4） I：提供片上断点和调试点
（5） E：增加了 DSP 指令
（6） J：Java 加速器
（7） S：可综合，提供 VHDL 或 Verilog 语言设计文件。

5.ARM 微处理器结构

（1）ARM 芯片内部结构

ARMCPU(内核)+外部设备（部件）=ARM 芯片

（2）ARM 内核结构

=运算部件+控制部件+寄存器组+总线

详细：包括 ALU、桶形移位器、乘法器、浮点部件（可选）、指令译码及控制逻辑、指令流水线、数据/地址寄存器、状态寄存器、总线等。

6.ARM 微处理器运行模式

7 种运行模式

（1） USR（用户模式） ARM 处理器正常的程序执行状态
（2） FIQ（快速中断模式） 用于高速数据传输或通道处理
（3） IRQ（外部中断模式）用户通用的中断处理
（4） SVC（管理模式） 操作系统使用的保护模式，处理软件中断。（可以通过 SWI 指令进入）
（5） ABT（数据访问中止模式）当数据或指令预取中止时进入该模式，可用于虚拟存储器和存储器保护。
（6） UND（未定义指令模式） 当未执行的指令执行时进入该模式，可用于支持硬件协处理器的软件仿真
（7） SYS（系统模式）运行具有特权的操作系统任务

用户模式与系统模式的区别：

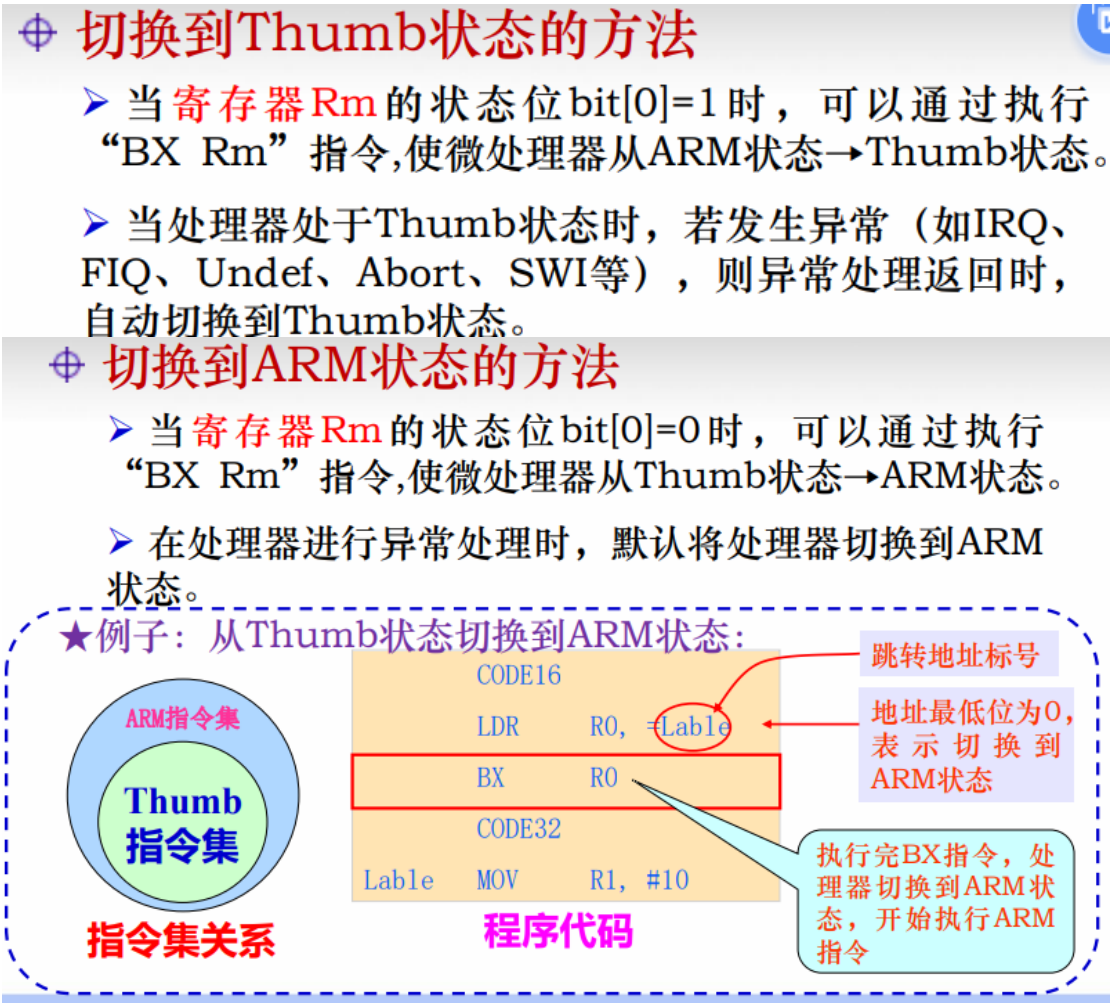
（1） 都不能由异常直接进入，且使用完全相同的寄存器组
（2） 系统模式：特权模式，没有用户模式的限制。OS 在该模式下可以访问用户模式的寄存器；特权任务可以使用该模式访问一些受控的资源。

7 种模式的切换方法：

处理器模式	备 注
USR（用户模式）	不能直接切换到其它模式
SYS（系统模式）	与用户模式类似，但具有可以直接切换到其它模式等特权
FIQ（快速中断模式）	FIQ异常响应时进入此模式
IRQ（外部中断模式）	IRQ异常响应时进入此模式
SVC（管理模式）	系统复位和软件中断响应时进入此模式
ABT（数据访问中止模式）	在ARM7TDMI中几乎没有用处
UND（未定义指令模式）	未定义指令异常响应时进入此模式

7.ARM 指令集与 Thumb 指令集

ARM 指令 4 字节对齐，Thumb 指令 2 字节对齐
切换工作状态方法：

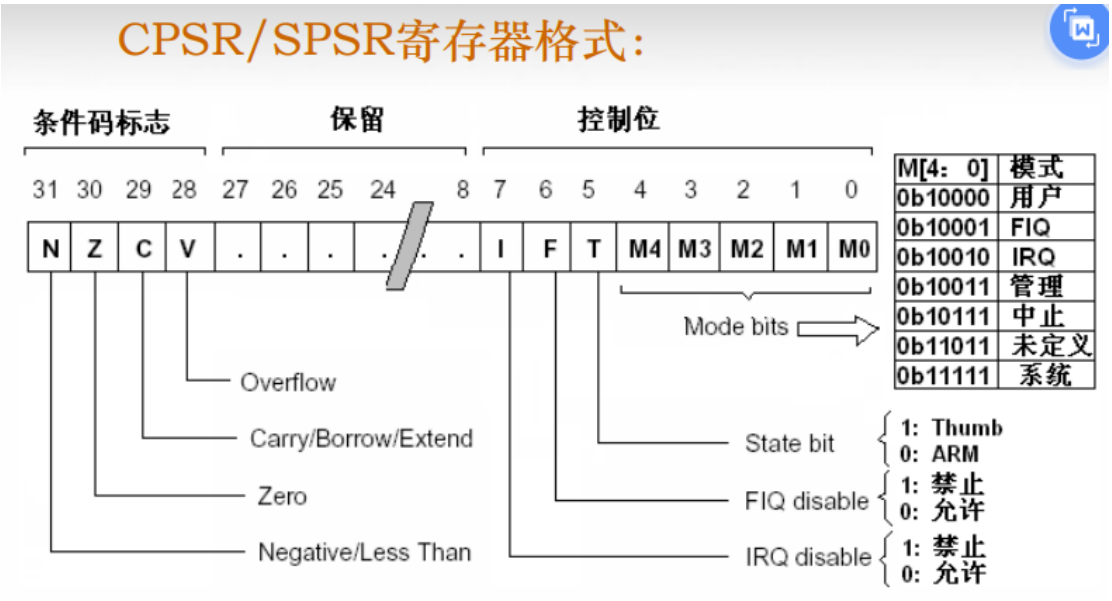


8.ARM 微处理器寄存器组织
ARM 状态下：37 个寄存器（31 个通用寄存器+6 个状态寄存器）
Thumb 状态下：27 个寄存器（21 个通用寄存器+6 个状态寄存器）

31 个通用寄存器：
R0~R7：所有 7 种运行模式下，指向相同的物理寄存器
R8~R12：除了 FIQ 模式下单独使用不同的 R8_fiq~R12_fiq,其他模式下指向相同的物理寄存器（Thumb 状态下不能使用这组寄存器，这就是 Thumb 为什么少了 10 个寄存器的原因）
R13:常用作堆栈指针，除用户模式与系统模式共用同一个物理寄存器，其他模式下都对应各自独立的物理寄存器
R14：子程序链接寄存器（链接寄存器 LR）。可以保持子程序或者异常处理后的返回地址。除用户模式与系统模式共用同一个物理寄存器，其他模式下都对应各自独立的物理寄存器
R15：程序计数器 PC。所有 7 种运行模式下，都指向同一个物理寄存器。

6 个状态寄存器：
CPSR：当前程序状态寄存器。所有 7 种运行模式下都指向相同的物理寄存器。可在任何运行模式下被访问。
SPSR：备份程序状态寄存器。5 种异常模式（即 7 种运行模式种除了用户模式与系统模式

的）都由自己的专用物理寄存器。异常发生时，保持 CPSR;异常退出时用于恢复 CPSR。



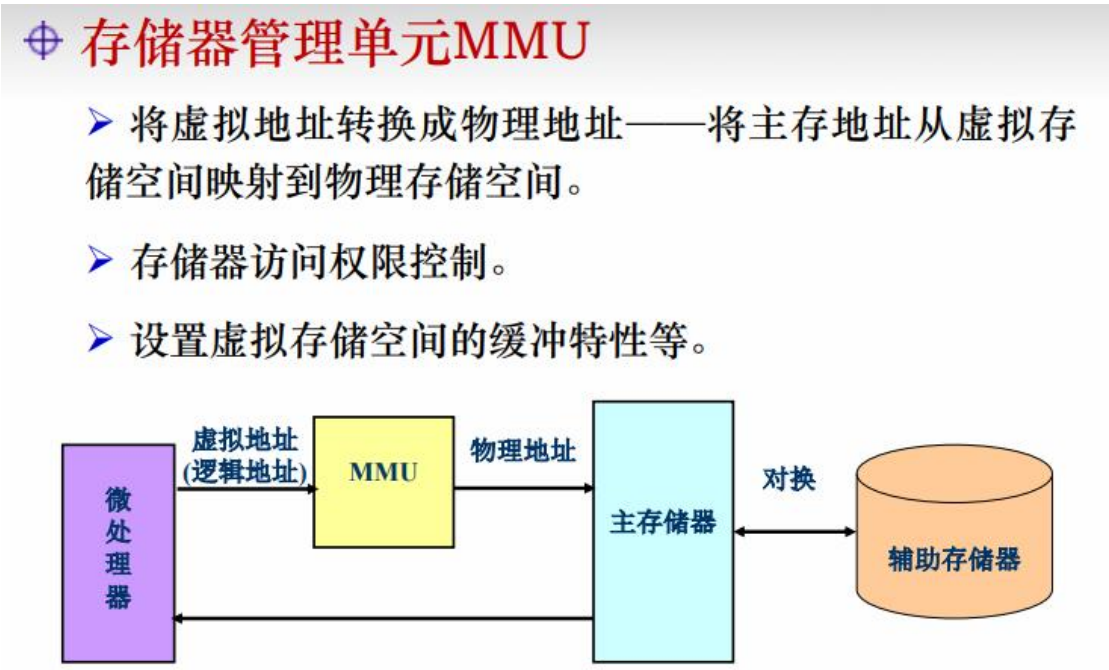
注意 I、F、T 和 M[4:0]发生异常时会自动修改。当运行于特权模式下，可以由程序修改。

9.ARM 微处理器存储器格式

大端格式：字数据的高字节存储在低地址中，而字数据的低字节存放在高地址中。

小端格式：字数据的高字节存储在高地址中，而字数据的低字节存放在低地址中。

10.存储器管理单元 MMU



11.虚拟存储管理

虚拟存储管理的三种方式:

（1） 分段式存储管理 （2）分页式存储管理（3）段页式存储管理

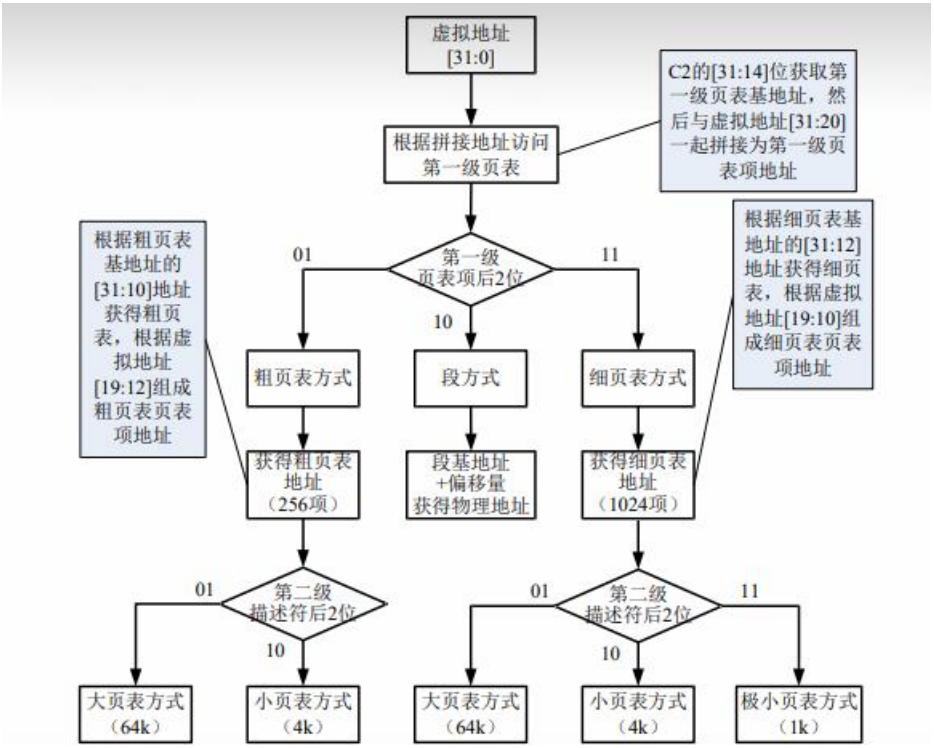
着重介绍分页式存储管理:

虚拟存储空间到物理存储空间的映射是以存储块为单位进行的。

存储块类型：
由地址转换条目[1:0]位的描述符确定如下四种虚拟地址对应的存储块类型
（1） 段（1MB） （2）大页（64KB） （3）小页（4KB） （4）极小页（1KB）
两级页表结构：一级页表 L1+二级页表 L2
二级页表 L2：分别由粗页表和细页表构成
一级页表：4096 条地址转换条目（段），占用内存 16KB
粗页表：256 条地址转换条目（大页和小页），占用内存 1KB
细页表：1024 条地址转换条目（大页、小页和极小页），占用内存 4KB
地址转换过程：认真看 ppt 上的例子，简答题 80%会考

地址转换过程

- (1) 根据给定的虚拟地址查找一级页表，获得一个条目。
- (2) 若该条目是段描述符，则返回物理地址，转换结束。——单步搜索
- (3) 若该条目是粗页表目录项，则继续利用虚拟地址查找二级粗页表，获得大页描述符或小页描述符，返回物理地址，转换结束。——两步搜索
- (4) 若该条目是细页表目录项，则继续利用虚拟地址查找二级细页表，获得大页、小页或极小页描述符，返回物理地址，转换结束。——两步搜索
- (5) 其它情况出错。



12.ARM 微处理器异常处理

7 个异常类型

异常类型	进入模式	具体含义	异常向量地址
复位	管理	复位电平有效时，产生复位异常，程序跳转到复位处理程序处执行。	0x00000000
未定义指令	未定义	遇到不能处理的指令时，产生未定义指令异常。	0x00000004
软件中断（SWI）	管理	执行SWI指令产生，用于用户模式下的程序调用特权操作指令。	0x00000008
指令预取中止	中止	处理器预取指令的地址不存在，或该地址不允许当前指令访问，产生指令预取中止异常。	0x0000000C
数据访问中止	中止	处理器数据访问指令的地址不存在，或该地址不允许当前指令访问时，产生数据中止异常。	0x00000010
IRQ（中断）	IRQ	外部中断请求有效，且CPSR中的I位为0时产生IRQ异常。	0x00000018
FIQ（快速中断）	FIQ	快速中断请求有效，且CPSR中的F位为0时产生FIQ异常。	0x0000001C

ARM 处理异常的具体步骤

（1） 将下一条的地址存入相应链接寄存器 LR，以便程序在处理异常返回时能从正确的位置重新开始执行。

（2） 将 CPSR 复制到相应的 SPSR 中

（3） 根据异常类型，强制设置 CPSR 的运行模式位

（4） 强制 PC 从相关的异常向量地址取下一条指令执行，从而跳转到相应的异常处理程序处。同时设置中断禁止位。

ARM 处理器异常返回的具体步骤

（1） 将链接寄存器 LR 的值减去相应的偏移量后送到 PC 中

（2） 将 SPSR 复制回 CPSR 中。

（3） 若再进入异常处理时设置了中断禁止位，要再次清除。

（4） 可以认为应用程序总是从复位异常处理程序开始执行的，因此复位异常处理程序不需要返回。

举例：

程序在系统模式下运行用户程序，假定当前处理器状态为 Thumb 状态，允许 IRQ 中断；

ARM 处理器进入异常过程：

（1） 将 CPSR 寄存器内容存入 IRQ 模式的 SPSR 寄存器

（2） 置位 I 位（禁止 IRQ 中断），清零 T 位（进入 ARM 状态），设置 MOD 位，切换处理器模式至 IRQ 模式

（3） 将第三条指令的地址（PC）存入 IRQ 模式的 LR 寄存器，将跳转地址放入 PC，实现跳转

ARM 处理器返回异常过程：

（1） 将 SPSR 寄存器的值复制回 CPSR 寄存器

（2） 将 LR 寄存器的值减去一个常量（IRQ 是 4）后放入 PC，从而返回到被中断的用户程

序继续执行。

异常类型对应的返回指令

异常类型	异常返回的位置	需设置的 PC 值	对应的返回指令
复位	无返回	无	无
未定义指令	未定义指令后第 1 条指令地址	LR	MOVS PC, LR
软中断 SWI	SWI 指令后第 1 条指令地址	LR	MOVS PC, LR
指令预取中止	本预取中止指令地址	LR-4	SUBS PC, LR, #4
IRQ	断点后第 1 条指令地址	LR-4	SUBS PC, LR, #4
FIQ	断点后第 1 条指令地址	LR-4	SUBS PC, LR, #4
数据预取中止	本数据中止指令地址	LR-8	SUBS PC, LR, #8

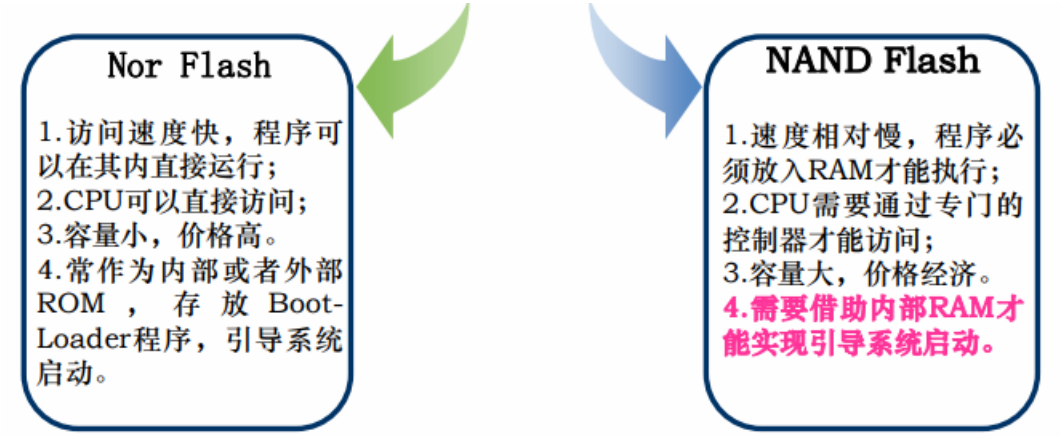
第三章

认真看看 PPT，寻址方式一定要会。占考试>=10 分

第四章 嵌入式存储器系统

1.嵌入式存储器的分类
按用途（所在位置）分类:
（1） 主存储器 （2）外部存储器
按存储介质分类:
（1）半导体存储器 （2）光盘存储器 （3）磁表面存储器
按存取方式分类:
（1） 随机存取存储器 RAM （2）顺序存取存储器 SAM （3）直接存取存储器 DAM
（4）只读存储器 ROM (5)闪速存储器 Flash Memory

Nor Flash 与 NAND Flash 的对比:
（1） NAND Flash 的写入速度比 Nor Flash 快很多, 而 Nor Flash 的读取速度比 NAND Flash 稍快一些。
（2） NAND Flash 最大擦写次数是 100 万次，而 Nor Flash 是 10 万次
（3） Nor Flash 主要用于程序存储，而 NAND Flash 同时也适用于数据存储



2.存储器系统空间分布

- 可寻址外部存储空间为1GB。
  - 被分成8个Bank(Bank0～Bank7)，每个Bank为128MB。(8×128MB = 1024MB = 1GB)
    - ✓ Bank0的数据位宽只能是16或者32位(由引脚电平决定)，其它Bank可以编程设定为8、16或者32位。
    - ✓ 前6个Bank(Bank0～Bank5)可以外接ROM、SRAM类型的存储器；Bank6和Bank7可以外接ROM、SRAM、SDRAM类型的存储器。
    - ✓ Bank6和Bank7的大小可以编程设定。
    - ✓ 前7个Bank(Bank0～Bank6)有固定的起始地址；最后1个Bank(Bank7)起始地址 = Bank6的末地址 + 1。
    - ✓ 所有Bank的访问周期可编程设定。

3.外部存储器芯片连接

注意地址总线的连接方式:

- 与系统数据总线的宽度(位数)有关。

以S3C2440为例

存储器芯片的地址引脚	S3C2440的地址引脚（8 位数据总线）	S3C2440的地址引脚（16 位数据总线）	S3C2440的地址引脚（32 位数据总线）
A0	A0	A1	A2
A1	A1	A2	A3
……	…	…	…

看看 ppt 上的连接示例

4.NAND Flash 存储器配置列表

1. NAND Flash存储器选择位 NCON
2. NAND Flash存储器页容量选择位GPG13
3. NAND Flash存储器地址周期选择位GPG14
4. NAND Flash存储器总线宽度选择位GPG15

NCON0	GPG13	GPG14	GPG15
0：普通 NAND	0：256 字	0：3 个地址周期	0：8 位总线宽度
	1：512 字节	1：4 个地址周期	
1：先进 NAND	0：1K 字	0：4 个地址周期	1：16 位总线宽度
	1：2K 字节	1：5 个地址周期	

5. NAND Flash 自动启动模式下引导系统程序启动的流程：

首先将位于 NAND Flash 中的 Boot Loader 处的 4KB 程序放入微处理器内部 SRAM（Stepping Stone）处，执行代码，启动引导系统，进而将 NAND Flash 中的剩余程序复制到主存储器 SDRAM 中运行。

第五章 常用外围设备与接口

5.1 通用输入输出端口(GPIO)

GPIO：
定义：通用输入输出端口，是嵌入式系统的重要组成部分。
功能：用于连接各种类型的输入输出设备，以实现它们与微处理器之间的数据传输。
构成：130 个 GPIO 引脚,分为 9 组，分别是 GPA~GPJ
每个端口都有与之对应的控制寄存器。
控制寄存器组成：

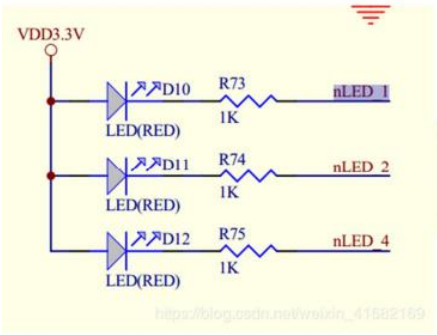
关于引脚功能，重点掌握 GPF 与 GPG 两组引脚。
P105-p107

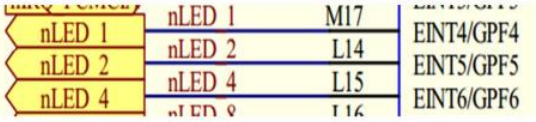
掌握 GPIO 端口点亮发光二极管的程序实现

举例

➢使用GPIO端口点亮发光二极管。

1.硬件原理图





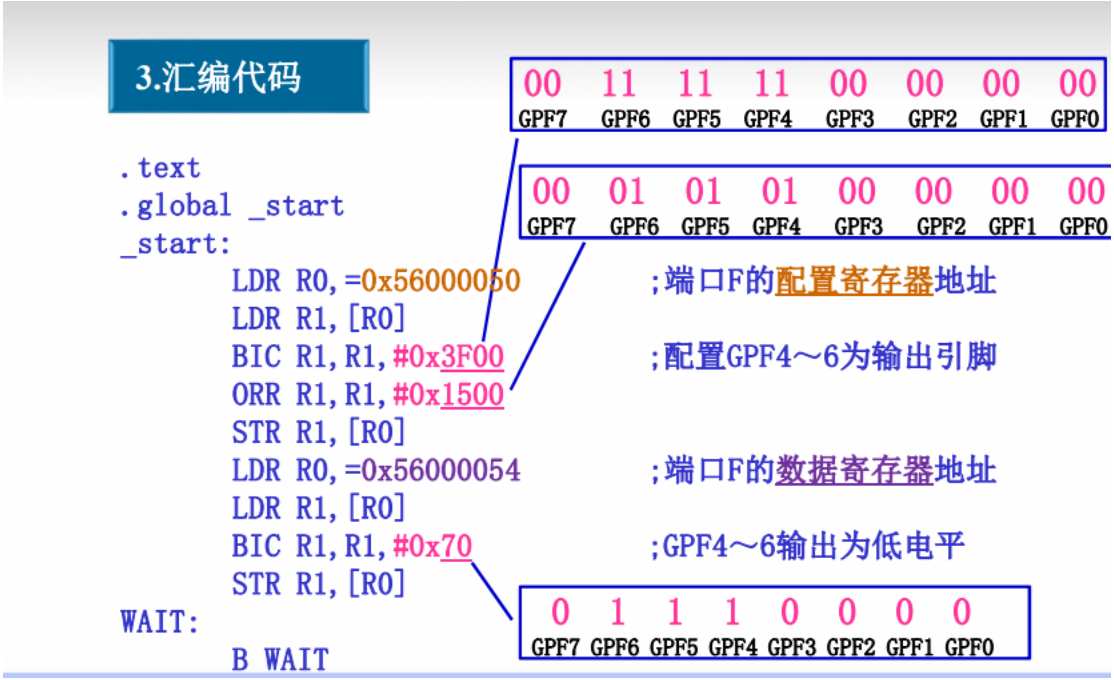
2.端口F寄存器

端口 F 控制寄存器（GPFCON，GPFDAT，GPFUP）

如果 GPF0 至 GPF7 在掉电模式中用于唤醒信号，端口将被设置为中断模式。

寄存器	地址	R/W	描述	复位值
GPFCON	0x56000050	R/W	配置端口 F 的引脚	0x0
GPFDAT	0x56000054	R/W	端口 F 的数据寄存器	–
GPFUP	0x56000058	R/W	端口 F 的上拉使能寄存器	0x00
保留	0x5600005C	–	保留	–

GPFCON	Bit	Description	
GPF7	[15:14]	00 = Input 10 = EINT[7]	01 = Output 11 = Reserved
GPF6	[13:12]	00 = Input 10 = EINT[6]	01 = Output 11 = Reserved
GPF5	[11:10]	00 = Input 10 = EINT[5]	01 = Output 11 = Reserved
GPF4	[9:8]	00 = Input 10 = EINT[4]	01 = Output 11 = Reserved



5.2 中断系统

中断的概念:

微处理器再执行正常程序的过程中，因某事件发生，收到来自外围部件的请求信号。若能够响应该信号，则暂停当前程序的正常执行，转去执行针对请求事件的处理操作，待结束后再返回被暂时中断的程序继续执行。

中断的作用:

（1） 并行处理

在外围设备需要传输数据时才产生“中断”，使得微处理器可以与多个外围设备同时工作，提高了微处理器的工作效率。

（2） 实时处理

在实时控制系统中，外围设备提出服务请求的时间是随机的。只有通过中断系统，才能对它们进行快速响应。

（3） 故障处理

系统在运行过程中，常常会出现一些突发性故障，利用中断功能就可以对它们进行实时处理。

中断系统的构成:

ARM 的 7 种异常类型中，由芯片外部中断请求引脚和内部外设触发的是外部中断请求（IRQ）和快速中断请求（FIQ）。

借助中断控制器处理 60 个中断源。

中断控制器:

功能:

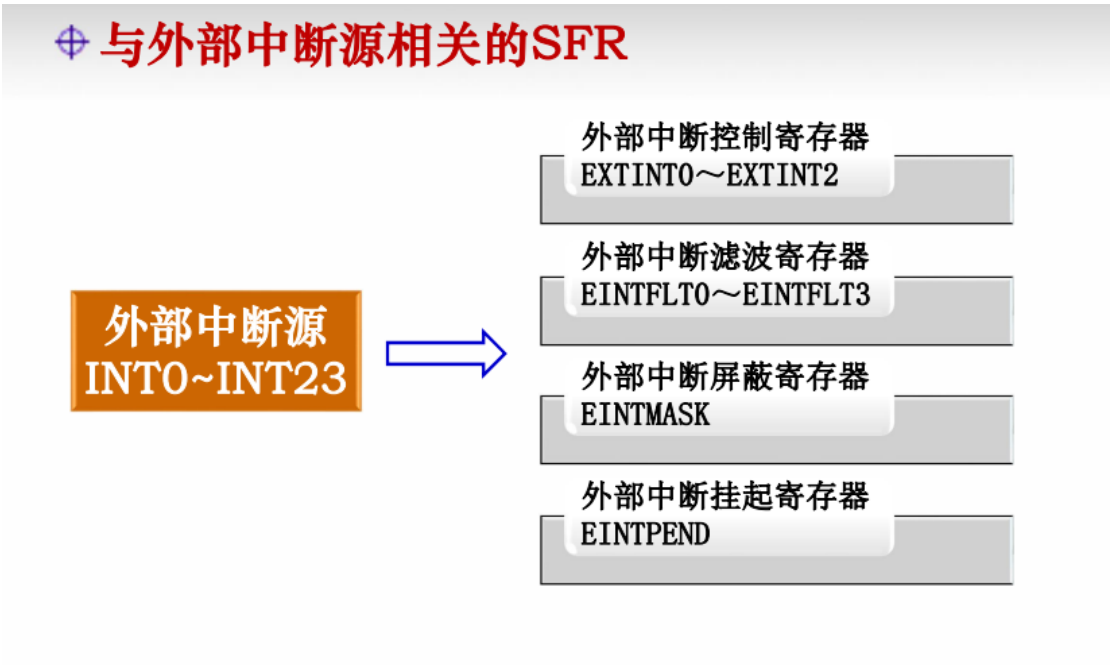
（1） 外部中断请求信号管理

（2） 中断模式设定
（3） 中断请求信号标记
（4） 中断屏蔽设计
（5） 中断优先级管理
（6） 中断服务标记

EINT8_23	外部中断8_23	ARB1
EINT4_7	外部中断4_7	ARB1
EINT3	外部中断3	ARB0
EINT2	外部中断2	ARB0
EINT1	外部中断1	ARB0
EINT0	外部中断0	ARB0

32 个一级中断源包含了 0~23 个 EINT 外部中断源

与外部中断源相关的寄存器组成:



下面了解:
外部中断控制寄存器
EXTINT0:
引脚名称: EINTn(0~7),位索引[4*n+2:4*n],000=低电平有效, 001=高电平有效
EXTINT1:
引脚名称: EINTn(8~15)
EXTINT2:
引脚名称: EINTn(16~23)
外部中断屏蔽寄存器(EINTMASK)

■外部中断屏蔽寄存器(EINTMASK)

名称	位索引	描述	初始值
EINTn (n=20~23)	[n]	设置EINTn的中断允许 0=使能中断；1=禁止中断。	0
EINTn (n=4~19)	[n]	设置EINTn的中断允许 0=使能中断；1=禁止中断。	1
保留	[3:0]	保留	111

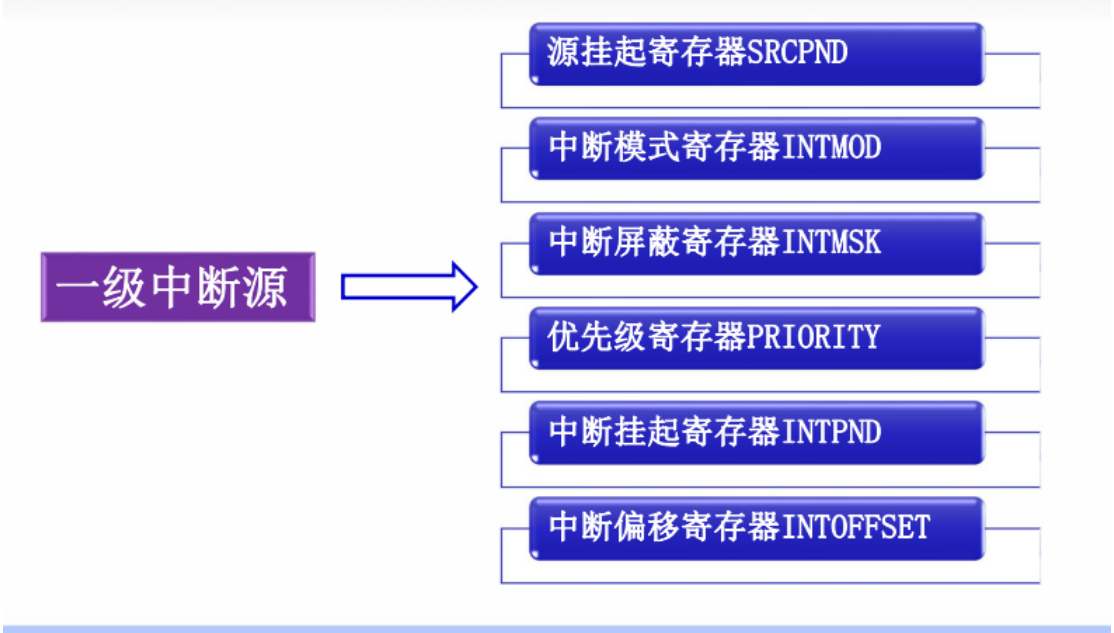
外部中断挂起寄存器（EINTPEND）:

■外部中断挂起寄存器(EINTPEND)

名称	位索引	描述	初始值
EINTn (n=4~23)	[n]	表明EINTn引脚中断请求信号是否有效 0=无中断请求；1=有中断请求。	0
保留	[3:0]	保留	000

△对某位写入“1”，可实现对其清零。

与一级中断源相关的寄存器



（1） 源挂起寄存器 SRCPND:

功能：用于标记 32 个一级中断源的中断请求信号是否触发，即是否有中断等待处理，供中断服务程序读取和判断。

0=中断未被请求 1=中断源声明了中断请求

需要人工清楚置位，写入 1 表示清除，0 表示保持不变

与外部中断源挂起寄存器一同确定 EINT4~EINT23 中具体哪些信号有效

(2) 中断模式寄存器 INTMOD:

功能：用于设定 32 个一级中断源的中断模式

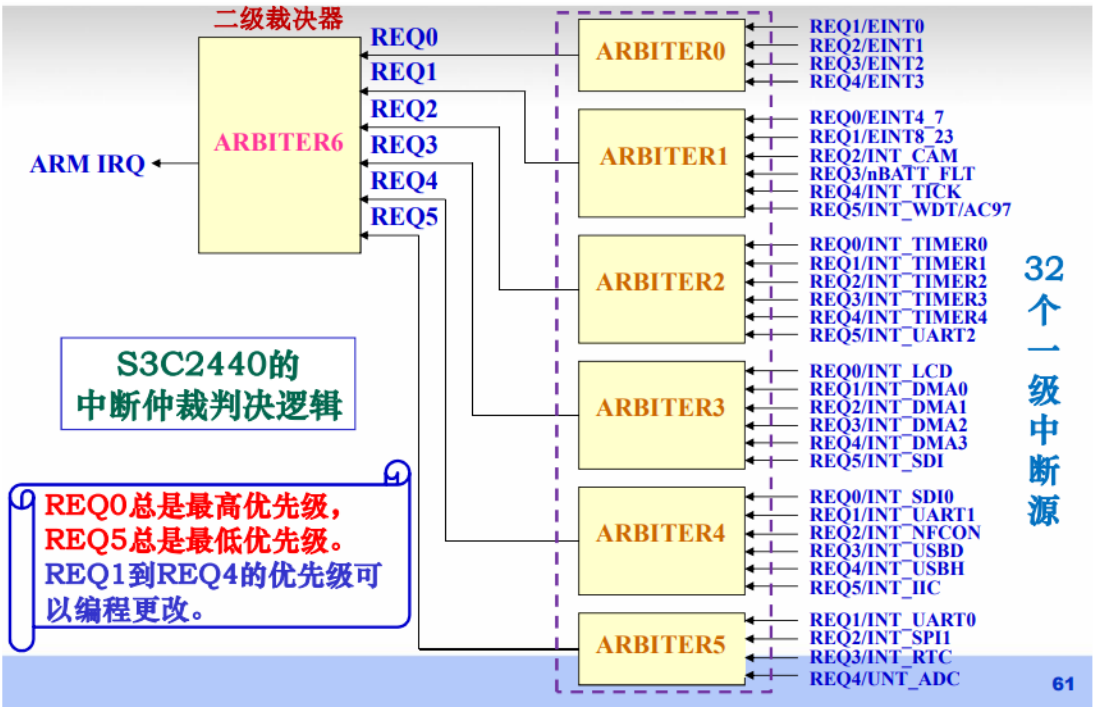
0=IRQ 模式 1=FIQ 模式

(3) 中断屏蔽寄存器 INTMSK:

功能：用于设定 32 个一级中断源是否被允许中断

0=能接收中断源中断 1=中断源发出中断被屏蔽

(4) 优先级寄存器 PRIORITY:



功能：用于设定 32 个一级中断源再 IRQ 模式下的优先级顺序

当多个中断源的中断请求信号同时有效时，需要通过中断优先级来确定对这些中断请求的服务顺序。

中断控制器借助优先级裁决器实现对 32 个一级中断源的优先级判决。

中断优先级的类型

静态优先级：REQ1~REQ4 优先级一旦通过程序设定好，则固定不再变化。

动态优先级：REQ1~REQ4 优先级通过程序设定好后，只要被中断处理，则优先级自动降为最低。

(5) 中断挂起寄存器 INTPND:

功能：用于标记 32 个一级中断源的中断请求是否即将或者正在被微处理器服务。

=1 表明所对应的中断在所有已触发的中断中优先级最高且没有被屏蔽。

同一时刻只能有 1 位被置 1。

需要人工清除置位

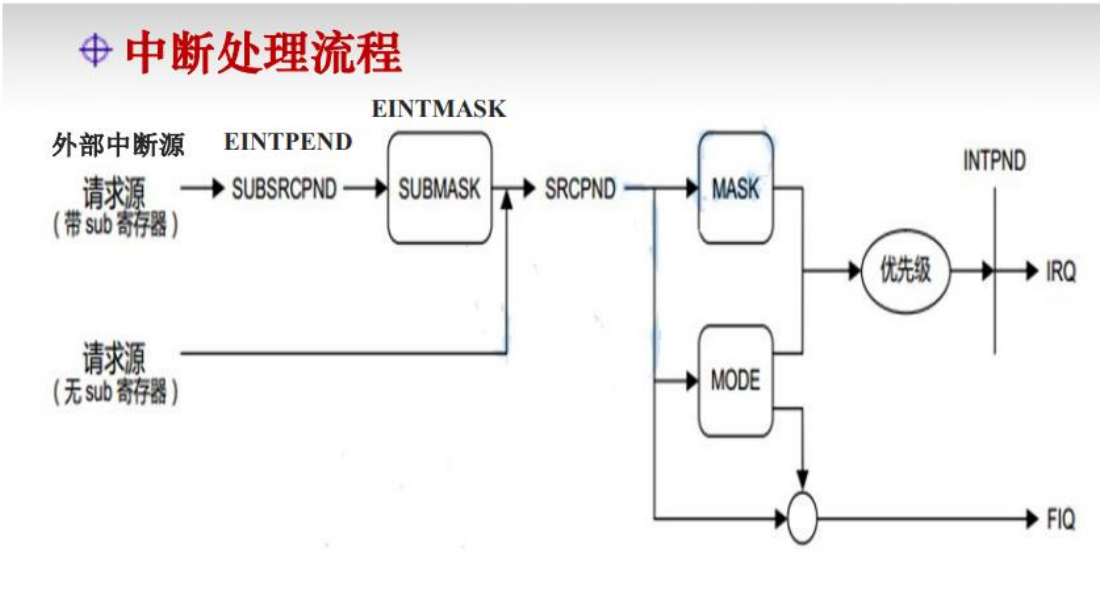
(6) 中断偏移寄存器 INTOFFSET

用于标记 INTPND 中置“1”位对应中断源的偏移量，代表了即将或者正在被微处理器服务的中断源号。

人工清除 SRCPND 和 INTPND 中的置 1 位，该寄存器的值会被自动清楚。

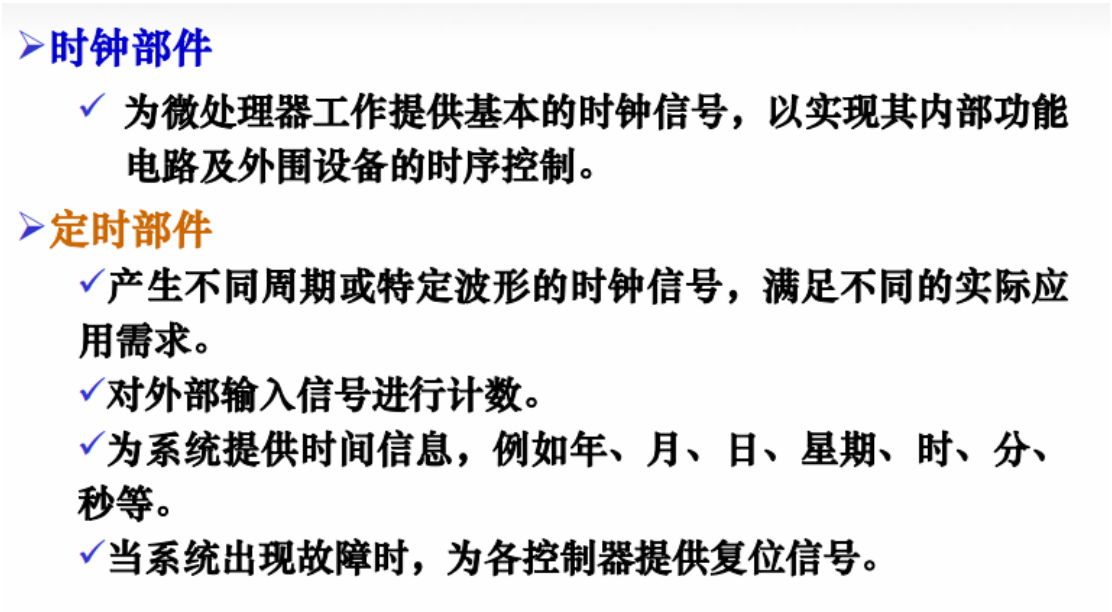
与 INTPND 一样，仅对 IRQ 模式的中断有效。

中断处理流程：



5.3 定时部件

功能：



时钟部件：

时钟控制模块

三种时钟信号：

FCLK(内核时钟):供微处理器（内核）使用的时钟信号

HCLK(总线时钟)：供高性能总线 AHB 使用的时钟信号

PCLK(I/O 接口时钟)：供外围总线 APB 使用的时钟信号

时钟频率的计算：

△时钟频率的计算公式

FCLK= $F_{OUT} = 2 * m * Fin / (p*2^S)$

●Fin：输入时钟源的频率。

●m、p、s：分频控制参数，分别由锁相环配置寄存器MPLLCON中MDIV、PDIV和SDIV的值确定。

MPLLCON（锁相环配置寄存器）：

MPLLCON 寄存器用于设置 FCLK 与 Fin 的倍数。MPLLCON 的位[19:12]称为 MDIV，位[9:4]称为 PDIV，位[1:0]称为 SDIV。

对于 S3C2440，FCLK 与 Fin 的关系如下面公式：

MPLL(FCLK) = (2×m×Fin)/(p×2^s)

其中： m=MDIC+8，p=PDIV+2，s=SDIV

CLKDIVN（时钟分频控制寄存器）：设定三种时钟信号的比例关系

时钟控制模块(续)

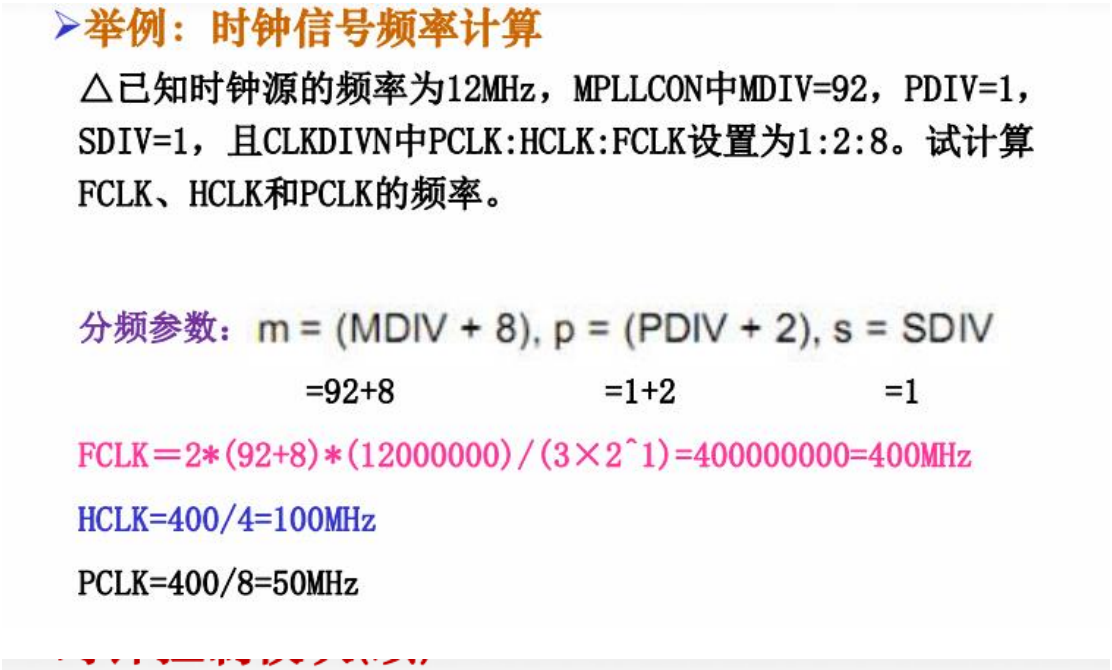
分频比例的选择 ——设定三种时钟信号的比例关系

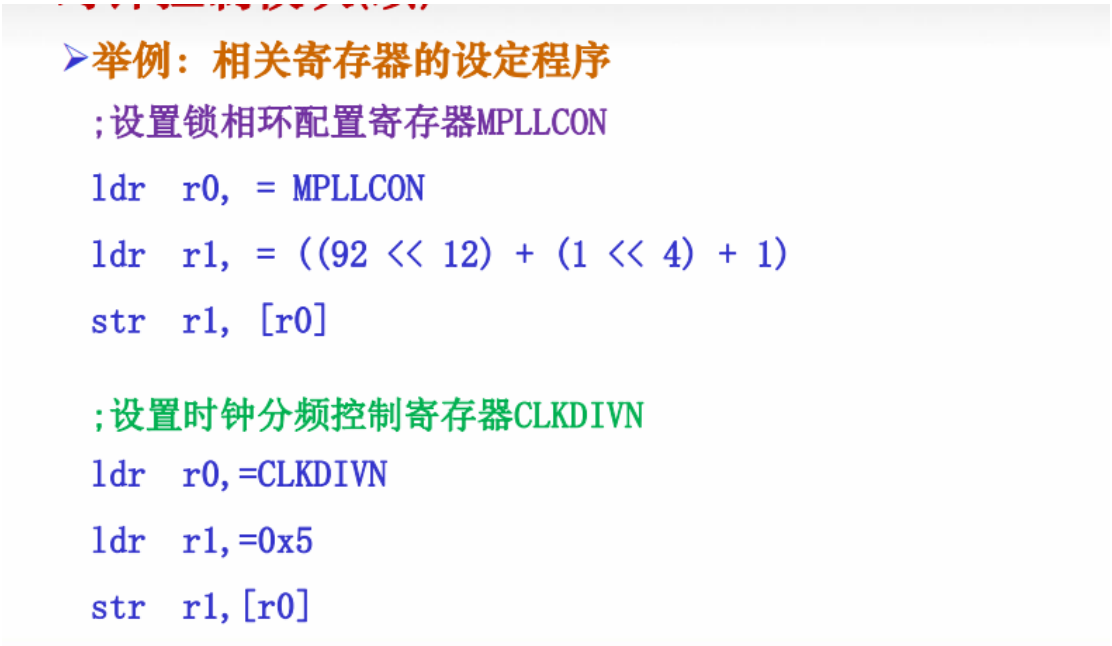
时钟分频控制（CLKDIVN）寄存器

寄存器	地址	R/W	描述	复位值
CLKDIVN	0x4C000014	R/W	时钟分频控制寄存器	0x00000004

CLKDIVN	位	描述	初始状态
DIVN_UPLL	[3]	UCLK 选择寄存器（UCLK 必须为 48MHz 给 USB） 0：UCLK = UPLL 时钟　　1：UCLK = UPLL 时钟 / 2 当 UPLL 时钟被设置为 48MHz 时，设置为 0 当 UPLL 时钟被设置为 96MHz 时，设置为 1	0
HDIVN	[2:1]	00：HCLK = FCLK/1 01：HCLK = FCLK/2 10：HCLK = FCLK/4　当 CAMDIVN[9] = 0 时 HCLK = FCLK/8　当 CAMDIVN[9] = 1 时 11：HCLK = FCLK/3　当 CAMDIVN[8] = 0 时 HCLK = FCLK/6　当 CAMDIVN[8] = 1 时	00
PDIVN	[0]	0：PCLK 是和 HCLK/1 相同的时钟 1：PCLK 是和 HCLK/2 相同的时钟	0

重要例题：





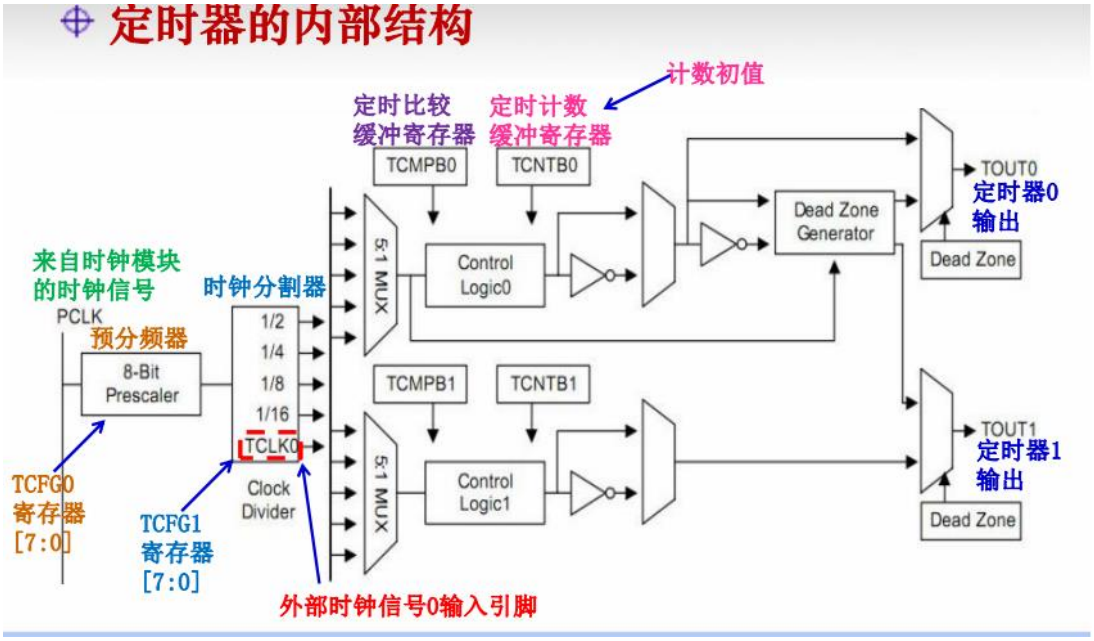
定时部件：

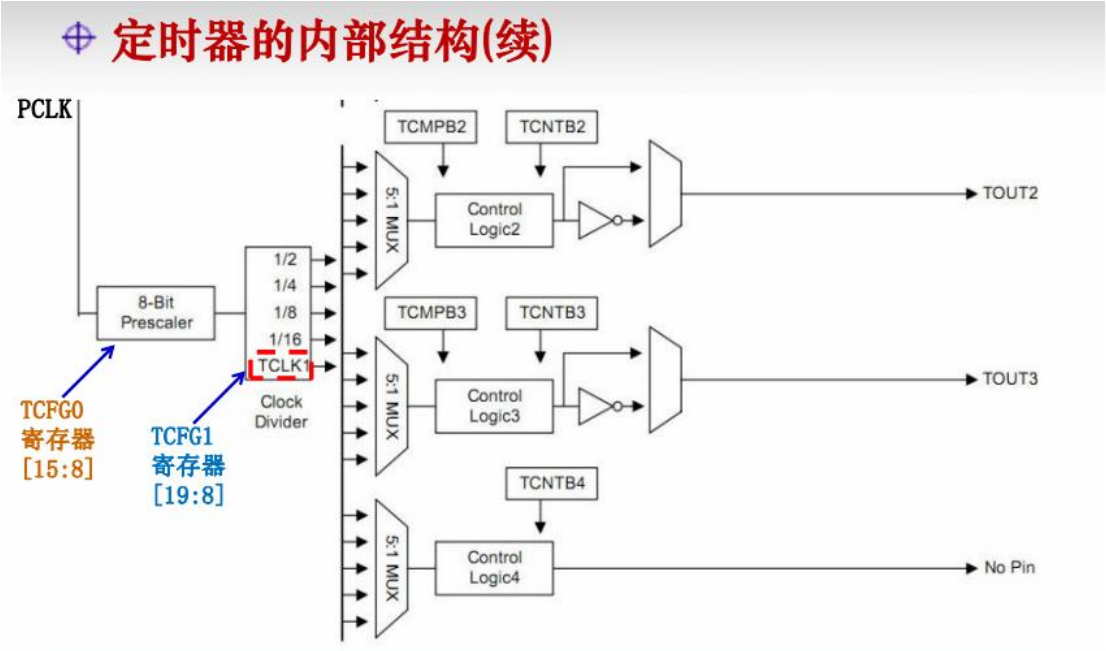
（1） 定时器：

功能：定时，计数，脉宽调制（PWM）

5 个 16 位定时器（递减计数器）

定时器 0 和 1 共用一个 8 位预分频器，定时器 2，3 和 4 共用另一个 8 位预分频器





定时器相关寄存器:

定时器配置寄存器 0 （TCFG0）

功能：配置两个 8 位预分频器

预分频1	[15:8]	决定了定时器 2、3 和 4 的预分频值	0x00
预分频0	[7:0]	决定了定时器 0 和 1 的预分频值	0x00

定时器配置寄存器 1 （TCFG1）

功能：5 路多路选择器和 DMA 模式选择寄存器

MUX 4	[19:16]	选择 PWM 定时器 4 的选通输入 0000 = ½；0001 = ¼；0010 = 1/8；0011 = 1/16； 01xx = 外部 TCLK1	0x0
MUX 3	[15:12]	选择 PWM 定时器 3的选通输入(同上)	0x0
MUX 2	[11:8]	选择 PWM 定时器 2的选通输入(同上)	0x0
MUX 1	[7:4]	选择 PWM 定时器 1的选通输入(同上)	0x0
MUX0	[3:0]	选择 PWM 定时器 0的选通输入(同上)	0x0

定时器控制寄存器（TCON)：
史

■ 定时器0计数缓冲寄存器(TCNTB0)

名称	位索引	描述	初始值
定时器 0 计数缓冲寄存器	[15:0]	设置定时器 0 的计数缓冲器的值	0x0000

■ 定时器0比较缓冲寄存器(TCMPB0)

名称	位索引	描述	初始值
定时器 0 比较缓冲寄存器	[15:0]	设置定时器 0 的比较缓冲器的值	0x0000

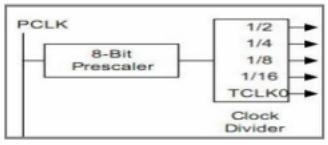
■ 定时器0计数监视寄存器(TCNTO0)

名称	位索引	描述	初始值
定时器 0 计数监视寄存器	[15:0]	定时器 0 的计数监视值	0x0000

参数计算：TCNTB0 的计数初值

计数初值 = 定时器工作频率÷(PWM)定时信号频率
= (PWM)定时信号周期÷定时器工作周期

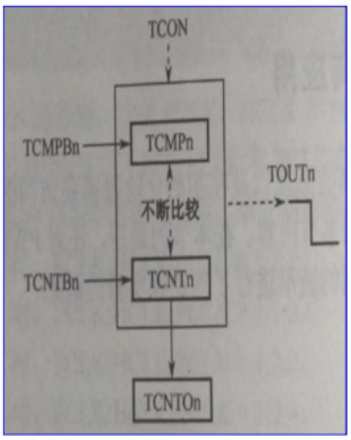
定时器工作频率= PCLK频率÷(预分频值+1)÷分频值
TCLK频率



定时器工作特点：

定时器的特点

- 实际执行减1操作的是TCNTn。为了防止对其读写影响计数，不允许程序对其直接访问。
  - ✓通过程序将计数初值写入TCNTBn。每次定时开始时，其值会被复制到TCNTn中。
  - ✓通过程序读取TCNTOn，间接获取TCNTn的动态计数值。
  - ✓通过程序读取TCNTBn，读出的是下次定时操作的重载值。
- TCNTn减到0时，可根据TCON的“自动重载”标志，决定是否复制TCNTBn到TCNTn中，以开启下一次定时操作。



定时器的特点(续)

- 内部设有双缓冲功能。因此，在当前定时操作中，若设置新的定时器值，仅当当前定时操作执行完毕后才有效。
  ——非立即有效

(2) 实时部件：
史，不考

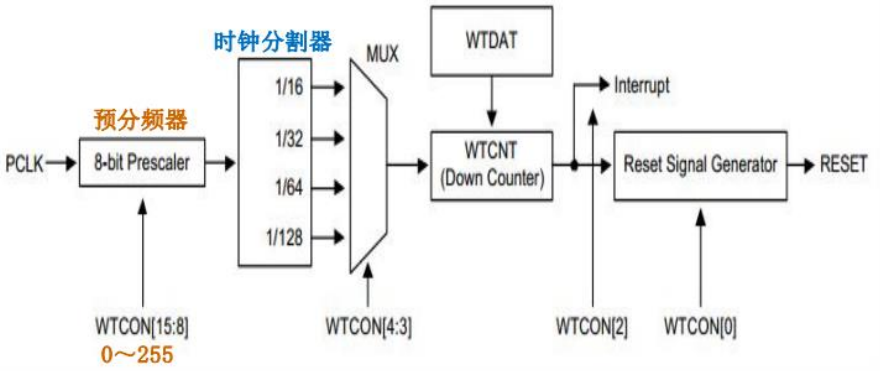
(3) 看门狗定时器

看门狗定时器

以S3C2440为例

- WDT(WatchDog Timer)是一种用于当噪音或系统错误引起故障时，恢复控制器操作的定时器。
- 本质上是一个16位内部定时器，当计数器值为0(即超时)时，通过触发中断服务，激活128个PCLK时钟周期的内部复位信号。
- 可用作一个普通的带中断请求的16位定时器。
- 只能使用PCLK时钟信号作为定时器的源输入时钟信号，且没有输出信号引脚。
- 系统处于嵌入式ICE调试模式时，WDT必须无操作(自动关闭)。

看门狗定时器的内部结构



F_watchdog = PCLK / (Prescaler value + 1) / Division_factor

相关寄存器:

WTCON（看门狗定时器控制寄存器）

注意：1=使能，0=禁止：复位同理

后面两个直接粘 ppt

■看门狗定时器数据寄存器(WTDAT)

名称	位索引	描述	初始值
计数重载值	[15:0]	看门狗定时器重载的计数值(0～65535)	0x8000

△计数值=所需时间间隔/T_watchdog = 所需时间间隔*F_watchdog

■看门狗定时器计数寄存器(WTCNT)

名称	位索引	描述	初始值
计数值	[15:0]	看门狗定时器的当前计数值	0x8000

注意事项看一下，可能有判断题

看门狗定时器的注意事项

- 看门狗定时器中断与看门狗定时器复位
  - ✓看门狗定时器中断是看门狗定时器作为普通定时器应用时，向微处理器发出内部中断请求；
  - ✓看门狗定时器复位是让微处理器复位，相当于重新启动所有程序。
- WTDAT存放WDT的计数初值，相当于普通定时器的TCNTBn。
- WTCNT是WDT的减1计数器，相当于普通定时器的TCNTn(程序不可直接访问)。
- 不同于普通定时器，WTDAT的值在WDT初始使能时，不能自动装载到WTCNT中，所以必须要给WTCNT设定一个初始值。

第六章 LINUX

1.BOOTLOADER 的定义:
答: BootLoader 是 系统加电后、操作系统内核或用户应用程序运行之前，首先必须运行的一段程序代码。通过这段程序，为最终调用操作系统内核、运行用户应用程序准备好正确的环境。
2.常用的 Linux 指令

3.Linux 中文件的访问权限

4.交叉编译定义
在一种平台上能够编译出在另一种平台（体系结构不同）上运行的程序。

5