# **Start.S笔记**

* What is the interrupt vector? How the interrupted vector table is indexed?

An **interrupt vector** is the [memory address](http://en.wikipedia.org/wiki/Memory_address) of an [interrupt handler](http://en.wikipedia.org/wiki/Interrupt_handler), or an [index](http://en.wikipedia.org/wiki/Array_index) into an [array](http://en.wikipedia.org/wiki/Array_data_structure) called an [interrupt vector table](http://en.wikipedia.org/wiki/Interrupt_vector_table) that contains the memory addresses of interrupt handlers. When an **interrupt** is generated, the [Operating System](http://en.wikipedia.org/wiki/Operating_System) saves its [execution state](http://en.wikipedia.org/wiki/State_(computer_science)) via a [context switch](http://en.wikipedia.org/wiki/Context_switch), and begins execution of the interrupt handler at the interrupt vector.

* **ARM中断向量表里有几种异常？每一种异常在什么情况下产生？**

**中断触发后，中断向量表是如何被使用的？**

<http://blog.csdn.net/gooogleman/article/details/3597175>

When an exception or interrupt occurs, the processor sets the pc to a speciﬁc memory  
address. The address is within a special address range called the vector table. The entries  
in the vector table are instructions that branch to speciﬁc routines designed to handle a  
particular exception or interrupt.

The memory map address 0x00000000 is reserved for the vector table, a set of 32-bit  
words. On some processors the vector table can be optionally located at a higher address  
in memory (starting at the offset 0xffff0000). Operating systems such as Linux and  
Microsoft’s embedded products can take advantage of this feature.

When an exception or interrupt occurs, the processor suspends normal execution and  
starts loading instructions from the exception vector table (see Table 2.6). Each vector table  
entry contains a form of branch instruction pointing to the start of a speciﬁc routine:

**Different interrupt vectors:**

Reset vector is the location of the ﬁrst instruction executed by the processor when power  
is applied. This instruction branches to the initialization code.

 Undeﬁned instruction vector is used when the processor cannot decode an instruction.

Software interrupt vector is called when you execute a SWI instruction. The SWI  
instruction is frequently used as themechanismto invoke an operating systemroutine.

Prefetch abort vector occurs when the processor attempts to fetch an instruction froman  
address without the correct access permissions. The actual abort occurs in the decode  
stage.

Data abort vector is similar to a prefetch abort but is raised when an instruction attempts  
to access data memory without the correct access permissions.

Interrupt request vector is used by external hardware to interrupt the normal execution  
ﬂow of the processor. It can only be raised if IRQs are not masked in the cpsr.

# **【Questions】**

* Why for every mode(svc,fiq), it should have its own registers?
* What is the difference between the svc mode and sys mode,
* Except for the mode which is automatically entered by the generation of interrupt, how to enter into svc mode and sys mode and user mode? (Can we enter by configuring cpsr register?)
* CPU 异常与CPU模式之间的对应关系？
* u-boot.lds如何修改？
* Arm2440中断控制器的机制？ 以及中断控制器又是如何与arm内核配合工作来处理中断的？
* 时钟模块的原理？

PLL原理？

RISC架构的特点？

中断源的仲裁（IRQ0~IRQ5）是如何做的？

如果有两个中断源被设置成FIQ, cpu如何实现优先级别的仲裁？

如果MPLL被配置之前都是使用外部时钟或者外部晶振提供的时钟，那么对于外部时钟和外部晶振有什么要求？

如果有两个FIQ同时产生，如何处理？

如果arm920t有32根地址线，那么为什么存储控制只能访问一共1G的空间？

使用nand flash与使用nor flash引导cpu的boot-up有什么区别？

SDRAM的相关问题：**DQM信号**

       如前所述，32位的SDRAM存储单元以4Byte为单位进行数据访问的时候，内存会忽略CPU的A1和A0地址线（事实上，2410的A1和A0地址线也根本没接到SDRAM的芯片上）。 但是处理器如果需要访问地址偏移量为01的单个字节，就需要DQM(Data Mask)信号进行帮助，这个信号接在2410的nWEB线上。DQM信号由处理器根据当前的访问情况发出，如果当前的访问只需要低16字节，那么nWEB0和nWEB1线就会有效。

DQM,nBE,nWBE如何被使用？

Start.S分析：

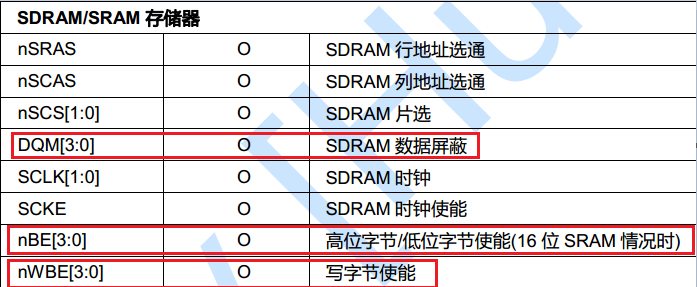
* \_TEXT\_BASE在代码relocate之前是什么值？

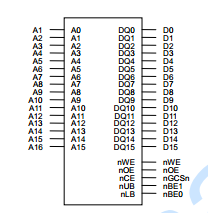
adr r0, \_start /\* r0 <- current position of code \*/

ldr r1, \_TEXT\_BASE /\* test if we run from flash or RAM \*/

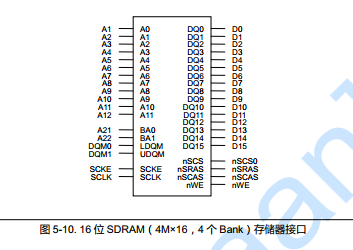
cmp r0, r1 /\* don't reloc during debug \*/

beq stack\_setup



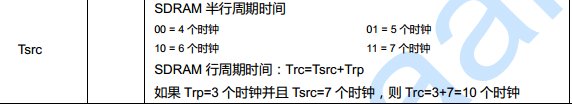


对于SDRAM, LDQM,UDQM用作什么？

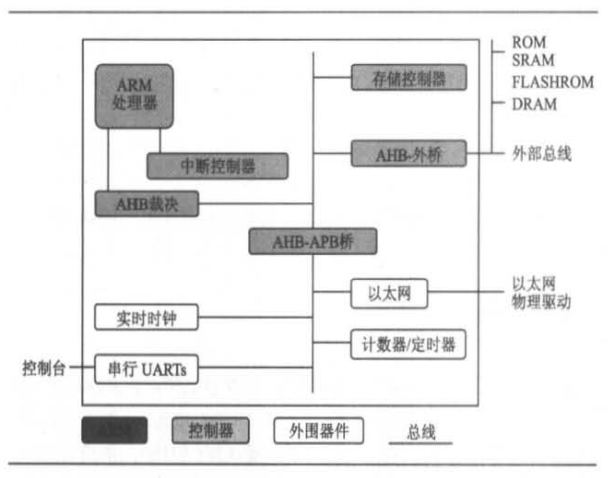


* SDRAM半行周期时间参数是指什么？

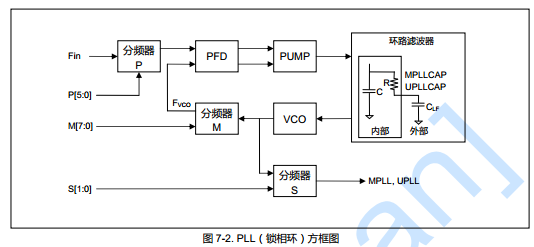
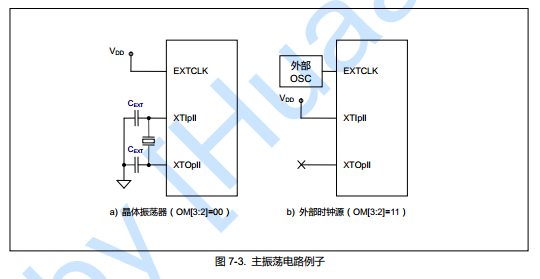
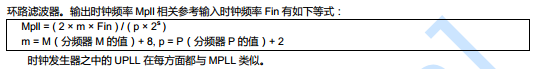
Tsrc：SDRAM的半行周期时间 ，而SDRAM行周期时间(Row Cycle Time)Trc=Tsrc+Trp。Trc包括行单元开启和行单元刷新在内的整个过程所需要的时钟周期数。

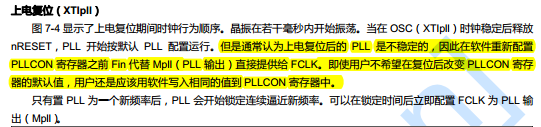


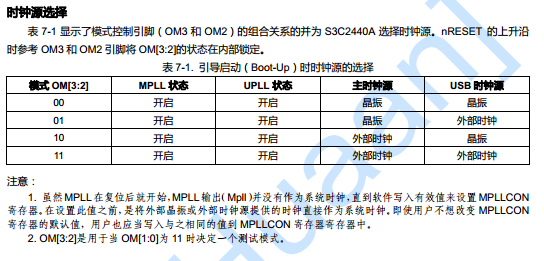
* **Arm 架构**

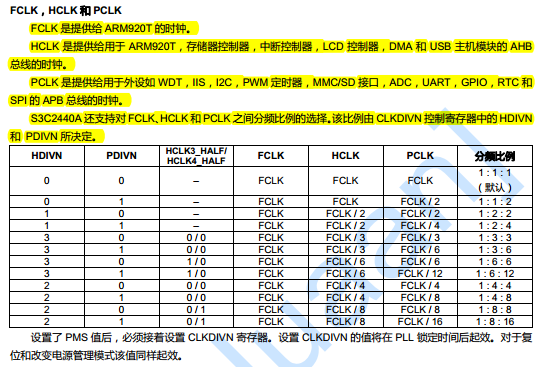


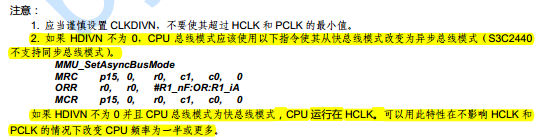
# **【时钟模块总结】**



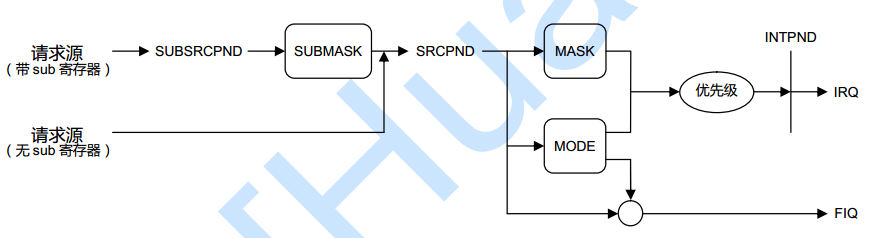






# **【中断控制器总结】**

当从内部外设和外部中断请求引脚收到多个中断请求时，中断控制器在仲裁步骤后请求ARM920T内核的FIQ或IRQ。



**中断配置步骤：**

1. CPSR寄存器内配置使能IRQ或FIQ.

**源挂起（SRCPND）寄存器**

SRCPND寄存器由32位组成，其每一位都涉及一个中断源。如果中断源产生了中断则相应的位被设置为1并且等待中断服务。因此此寄存器指示出是哪个中断源正在等待请求服务。注意SRCPND寄存器的每一位都是由中断源自动置位，其不顾INTMASK寄存器中的屏蔽位。另外SRCPND寄存器不受中断控制器的优先级逻辑的影响。

**换句话说，如果SRCPND寄存器的指定位被设置为1，其通常被认作一个有效中断请求正在等待服务。**

**可以通过写入一个数据到此寄存器来清除SRCPND寄存器的指定位。其只清除那些数据中被设置为1的相应位置的SRCPND位。那些数据中被设置为0的相应位置的位保持不变。**



**中断模式（INTMOD）寄存器**

此寄存器由32位组成，其每一位都都涉及一个中断源。如果某个指定为被设置为1，则在FIQ（快中断）模式中处理相应中断。否则则在IRQ模式中处理。



注意：如果中断模式在INTMOD寄存器中设置为FIQ模式，则FIQ中断将不会影响INTPND和INTOFFSET寄存器。这种情况下，这2个寄存器只对IRQ中断源有效。

**中断屏蔽（INTMSK）寄存器**

此寄存器由32位组成，其每一位都都涉及一个中断源。如果某个指定为被设置为1，则CPU不会去服务来自相应中断源（请注意即使在这种情况中，SRCPND寄存器的相应位也设置为1）的中断请求。如果屏蔽位为0，则可以服务中断请求。

