目录

1

2

3

4

ARM 架构

ARM（Advanced RISC Machine）架构是一种精简指令集（RISC）架构，由ARM Holdings公司开发。ARM架构的特点是指令集简单、执行速度快、功耗低、可扩展性强等。ARM架构广泛应用于嵌入式系统、移动设备、智能家居、物联网等领域，是当前最流行的嵌入式处理器架构之一。ARM架构的优点包括低功耗、高性能、易于移植和开发，因此受到了广泛的关注和应用。

5

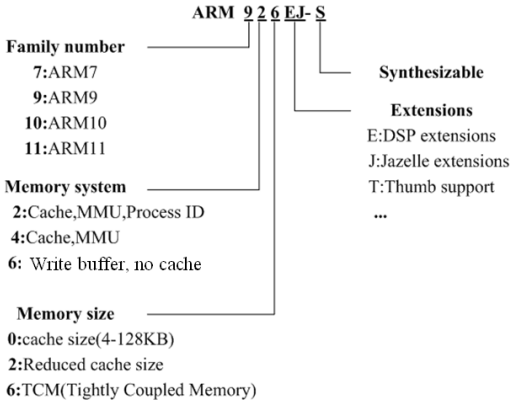
ARM介绍

1、ARM 体系结构
2、ARM 存储器
3、ARM 时钟及电源管理
4、ARM 定时技术
5、ARM 中断
6、ARM DMA

6

1 ARM嵌入式微处理器

- ARM嵌入式微处理器命名规则



7

1 ARM嵌入式微处理器(续)

- ARM系列微处理器扩展命名符号的含义

标志	含义	补充说明
T	16位Thumb指令集	Thumb指令集版本1：ARMv4T Thumb指令集版本2：ARMv5T Thumb-2：ARMv6T (支持16位Thumb压缩指令集)
D	片上调试	支持片上Debug，允许处理器响应调试请求暂停
M	支持增强型乘法器	32位乘32位得到64位
I	嵌入式ICE部件	提供片上断点和调试点的支持
E	增强型DSP指令	增加了DSP算法处理器指令：16位乘加指令，饱和的带符号数的加减法，双字数据操作，Cache预取指令
J	Java加速器Jazelle	提高Java代码的运行速度；与无加速器相比，最高可达到8倍
S	可综合	提供VHDL或Verilog HDL硬件描述语言设计文件

8

1 ARM嵌入式微处理器(续)

- ARM 变种：T变种、M变种，E变种、J变种和SIMD*变种
- T变种(Thumb指令集)
  - Thumb指令集（16位）是ARM指令集（32位）的子集。
  - 当系统的数据总线宽度小于32位时，使用Thumb指令集性能好
  - 代码尺寸：用Thumb指令编译其长度只占ARM指令的65%左右
  - 紧凑的代码和窄带宽的存储器系统，还会带来功耗上的优势
  - 与ARM指令集相比，Thumb指令集有以下局限
    - 完成相同的操作，Thumb指令通常需要更多的指令。ARM指令集更为适合对系统运行时间要求苛刻的应用场合
    - Thumb指令集没有包含进行异常处理时需要的一些指令，因此在异常中断的低级处理时，还是需要使用ARM指令
  - 使用ARM指令集还是使用Thumb指令集，需要从存储器开销和性能要求两方面加以权衡考虑

9

1 ARM嵌入式微处理器(续)

- ARM 变种 --- M变种(长乘法指令)
  - M变种增加了两条用于进行长乘法操作的AKM指令
    - 一条指令实现32位整数乘以32位整数，生成64位整数的长乘法操作
    - 另一条指令实现32位整数乘以32位整数，然后再加上32位整数，生成64位整数的长乘加操作
  - 在场合中，乘法操作的性能并不重要，但对于尺寸要求很苛刻，在系统实现时就不适合增加M变种的功能
  - M变种首先在ARMV3版本中引入
  - 如果没有上述的设计方面的限制，在ARMV4及其以后的版本中，M变种是系统中的标准部分

10

1 ARM嵌入式微处理器(续)

- ARM 变种 --- E变种(增强型DSP指令)
  - 包含一些附加指令，用于增强处理器对一些典型的DSP算法的处理性能
    - (1)几条新的实现16位数据乘法和乘加操作的指令
    - (2)实现饱和的带符号数的加减法操作的指令。所谓饱和的带符号数的加减法操作是在加减法操作溢出时，结果并不进行卷绕(Wrapping around)，而是使用最大的整数或最小的负数来表示
    - (3)进行双字数据操作的指令，包括双字读取指令LDRD，双字写入指令STRD和协处理器的寄存器传输指令MCRR / MKRC

11

1 ARM嵌入式微处理器(续)

- ARM 变种 --- J变种(Java加速器Jazelle)
  - 将Java的优势和先进的32位RISC芯片结合在一起
  - Jazelle技术提供了Java加速功能，可以得到比普通Java虚拟机高得多的性能。与普通的Java虚拟机相比，Jazelle使代码运行速度提高了8倍，而且功耗降低了80%
  - Jazelle技术使得程序员可以在一个单独的处理器上同时运行Java应用程序、已经建立好的操作系统、中间件以及其他的应用程序
  - 与使用协处理器和双处理器相比，使用单独处理器可以在提供高性能的同时，保证低功耗和低成本

12

1 ARM嵌入式微处理器(续)

- ARM 变种 --- SIMD*变种(ARM媒体功能扩展)
  - SIMD (Single Instruction Multiple Data, 单指令多数据流): 是能够复制多个操作，并把它们打包在大型寄存器的一组指令集；SIMD能力使得软件更有效地完成高性能的多媒体应用等数据密集型运算，如语音和图像编码器
  - SIMD在性能上的优势
    - 以加法指令为例，单指令单数据(SISD)的MPU对加法指令译码后，执行部件先访问内存，取得第一个操作数；之后再一次访问内存，取得第二个操作数；随后才能进行求和运算
    - 在SIMD型的MPU中，指令译码后几个执行部件同时访问内存，一次性获得所有操作数进行运算。这个特点使SIMD特别适合于多媒体应用等数据密集型运算

13

1 ARM嵌入式微处理器(续)

- ARM架构

Architecture	THUMB™	DSP	Jazelle™	Media
	特性集			
v4T	✓			
v5TE	✓	✓		
v5TEJ	✓	✓	✓	
v6	✓	✓	✓	✓

  - 不断创新以提升性能
  - 向下兼容以保护软件投入

14

1

- ARMv4T结构
  - ARM9采用,五级流水处理以及分离的Cache结构,平均功耗为0.7mW/MHz。时钟速度为120MHz-200MHz，每条指令平均执行1.5个时钟周期
  - ARM920T、ARM940T和ARM9E为含Cache的CPU核。性能为132MIPS（120MHz时钟，3.3V供电）或220MIPS(200MHz时钟)
  - ARM9系列嵌入式微处理器主要有ARM9 TDMI、ARM9 ES等系列
- ARMv6架构
  - ARM1136J(F)-S，ARM1156T2(F)-S，以及ARM1176JZ(F)-S
  - 许多新技术被引进，存储器系统加入了很多的崭新的特性，单指令多数据流（SIMD）指令开始首次引入的
  - 经过优化的Thumb-2指令集，适应低成本的单片机及汽车电子

15

1 ARM嵌入式微处理器(续)

- ARMv7架构, 内核架构首次从单一款式变成3种款式
  - 款式A(ARMv7-A):需要运行复杂应用程序的“应用处理器”。支持大型嵌入式操作系统，比如Symbian、Linux、Windows CE和3G智能手机操作系统(微软Windows Mobile、GOOGLE Android等)。这些应用需要劲爆的处理性能，并且需要硬件MMU实现的完整而强大的虚拟内存机制，还基本上会配有Java支持，有时还要求一个安全程序执行环境。典型的产品包括高端手机和手持仪器，电子钱包以及金融事务处理机
  - 款式R(ARMv7-R):硬实时且高性能的处理器。高端实时市场,像高档轿车的组件、大型发电机控制器、机器手臂控制器等，它们使用的处理器不但要很好、很强大，还要极其可靠，对事件的反应也要极其敏捷
  - 款式M(ARMv7-M):为旧世代单片机/MCU的应用而量身定制。在这些应用中，尤其是对于实时控制系统，低成本、低功耗、极速中断反应以及高处理效率，至关重要

16

1 ARM嵌入式微处理器(续)

- ARMv7架构
  - ARM Cortex系列 采用v7架构
  - Cortex-A8
  - Cortex-R4
  - Cortex-M3 , 包括所有的16位Thumb指令集和基本的32位Thumb-2指令集架构，但不能执行ARM指令集。Thumb-2在Thumb指令集架构(ISA)上进行了大量的改进，它与Thumb相比，具有更高的代码密度并提供16/32位指令的更高性能

17

1 ARM嵌入式微处理器(续)

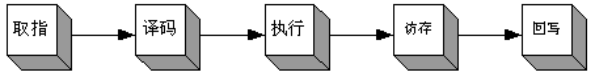
- ARM7的三级流水线
  - 一条指令有3个时钟周期的执行时间，但吞吐量是每个周期1条指令

    - 取指：从程序存储器中取指令,放入流水线.(占用存储器访问操作)
    - 译码：指令译码。(占用译码逻辑)
    - 执行：执行指令/读写REG。(占用ALU及数据路径)
  - PC值如何计算
    - PC 指向处于读取级的指令地址，而不是处于执行级的指令地址
    - PC=当前执行指令地址+8

18

1 ARM嵌入式微处理器(续)

- ARM9 五级流水线
  - 取指：从存储器中取出指令(fetch)，并将其放入指令流水线
  - 译码：对指令进行译码(dec)
  - 执行：执行运算ALU(exe)
  - 访存(缓冲/数据)：如果需要，则访问数据存储器(accmem)；否则ALU的结果只是简单地缓冲1个时钟周期，以便所有的指令具有同样的流水线流程
  - 回写：将指令产生的结果回写到寄存器(wtbk res)，包括任何从存储器中读取的数据



19

1 ARM嵌入式微处理器(续)

- ARM使用的是AMBA(Advanced Microcontroller Bus Architecture)总线体系结构
  - ARM公司颁布的总线标准
  - 通过AMBA可以方便地扩充各种处理器及I/O,可以把DSP、其他处理器和I/O(如UART、定时器和接口等)都集成在一块芯片中.该标准定义了如下三种总线:
    - AHB(Advanced High-performance Bus):连接高性能系统模块。它支持突发数据传输方式及单个数据传输方式；另外，它还支持分离式总线事务处理
    - ASB(Advanced System Bus):连接高性能系统模块，它支持突发数据传输模式
    - APB(Advanced Peripheral Bus) 简单接口，支持低性能外围接口

20

1 ARM嵌入式微处理器(续)

- ARM9 E-S内核
  - 32位定点RISC处理器，改进型ARM/Thumb代码，增强型乘法器设计，支持实时调试；
  - 片内指令和数据SRAM，而且指令和数据的存储器容量可调；
  - 片内指令和数据高速缓冲器（cache）容量从4K字节到1M字节；
  - 设置保护单元（protection unit），非常适合嵌入式应用中对存储器进行分段和保护；
  - 采用AMBA AHB总线接口，为外设提供统一的地址和数据总线；
  - 支持外部协处理器，指令和数据总线有简单的握手信令支持；
  - 支持标准基本逻辑单元扫描测试方法，并且支持内建自测试技术(BIST: Built-In-Self-Test)
  - 支持嵌入式跟踪宏单元(ETM, Embedded Trace Macrocell),支持实时跟踪指令和数据

21

1 ARM嵌入式微处理器(续)

- Thumb指令集
  - ARM指令集为32位指令集，可以实现ARM架构下所有功能
  - Thumb指令集是对32位ARM指令集的扩充，实现更高的代码密度。Thumb指令集实现的功能只是ARM指令集的子集，把常用的ARM指令压缩成16位的指令编码方式
  - Thumb指令集的16位指令代码长度大约是标准ARM指令代码密度的两倍，可以在16位存储系统上运行
  - 在执行时，16位的Thumb指令透明地实时解压缩成32位的ARM指令，并没有明显的性能损失，节省存储空间和硬件成本
  - Thumb指令的操作是在标准的ARM寄存器下进行的，在ARM指令代码和Thumb指令代码之间可以方便地进行切换

22

1 ARM嵌入式微处理器(续)

- Thumb指令集(续)
  - Thumb指令集为实现16位指令长度，舍弃了ARM指令集的一些特性，如大多数Thumb指令是无条件执行的，而几乎所有的ARM指令都是有条件执行的。大多数的Thumb数据处理指令的目的寄存器与其中一个源寄存器相同
  - 与全部采用ARM指令集的方式相比， 使用Thumb指令可以在代码密度方面改善大约30%，但这种改进是以代码的效率为代价的。尽管每个Thumb指令都有相对应的ARM指令，但是，执行相同的功能，需要更多的Thumb指令才能完成。因此，当指令预取需要的时间没有区别时，ARM指令相对Thumb指令具有更好的性能
  - 开发者在进行系统设计的时候需要综合考虑成本、性能和功耗等因素。如果在一个系统中综合使用ARM指令和Thumb指令，充分发挥各自的优点，就能在成本、性能和功耗等因素上取得较好的平衡

23

1 ARM嵌入式微处理器(续)

- ARM工作模式
  - 用户模式(usr)：ARM处理器正常执行程序时的处理。
  - 快速中断模式(fiq)：用于高速数据传输或通道处理。
  - 外部中断模式(irq)：用于通用的中断处理。
  - 管理模式(svc)：操作系统使用的保护模式。
  - 指令/数据访问终止模式(abt)：当数据或指令预取终止时进入该模式，可用于虚拟存储及存储保护。
  - 系统模式(sys)：运行具有特权的操作系统任务时的模式。
  - 未定义指令中止模式(und)：当未定义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

24

1 ARM嵌入式微处理器(续)

- ARM工作模式(续)
  - 工作模式可通过软件控制改变，也可通过外部中断或异常处理改变。大多数的应用程序运行在用户模式下，某些被保护的系统资源不能被访问。除用户模式以外，其余的所有6种模式称之为非用户模式或特权模式；除去用户模式和系统模式以外的5种又称为异常模式，用于处理中断或异常，以及需要访问受保护的系统资源等情况
  - 当某种异常发生时，处理器核即进入相应的工作模式。
    - 若发生了IRQ中断并响应IRQ中断，则ARM9 TDMI核将进入IRQ模式
    - 每种工作模式下均有其附加的某些寄存器，即使有异常情况发生，异常模式下的处理程序也不至于破坏用户模式的数据及状态

25

1 ARM嵌入式微处理器(续)

- ARM工作模式(续)
  - 从程序代码的角度来看，有两种工作状态：ARM状态和Thumb状态
    - 在ARM状态下，处理器核执行32位的、字对齐的ARM指令
    - 在Thumb状态下，处理器核执行16位的、半字对齐的Thumb指令
    - 在程序的执行过程中，ARM9处理器核可以随时在两种工作状态之间切换，不影响处理器的工作模式和相应寄存器中的内容
    - ARM指令集和Thumb指令集中均有切换ARM处理器工作状态的指令，使ARM处理器可在两种工作状态之间切换。
    - ARM处理器核在上电或复位并开始执行程序代码时，处于ARM状态
    - 当操作数寄存器的状态位(位0)为1时，可采用执行BX指令的方法，使ARM处理器从ARM状态切换到Thumb状态；当操作数寄存器的状态位为0时，执行BX指令可以使ARM处理器从Thumb状态切换到ARM状态
    - 在处理器进行异常处理时，将PC指针放入异常模式链接寄存器中，并从异常向量地址开始执行程序，也可以使处理器切换到ARM状态

26

1 ARM嵌入式微处理器(续)

- ARM工作模式(续)
  - 从代码角度来看，有两种工作状态：ARM状态和Thumb状态(续)

```
        ……
        ADR R0,THUMBCODE+1 ; 将R0的bit[0]置1
        BX R0              ; 跳转,并根据R0的bit[0]实现状态切换
        CODE16             ;16位Thumb代码
THUMBCODE MOV R2,#2
        ……
        ADR R0,ARMCODE     ;加载ARMCODE地址到R0中
        BX R0
        CODE32             ;32位ARM代码
        ARMCODE MOV R4,#4
        ……
```

注释: BX{<cond>} Rm ； 带状态切换的分支指令

功能：PC=Rm&0xfffffffe, T=Rm[0]&1

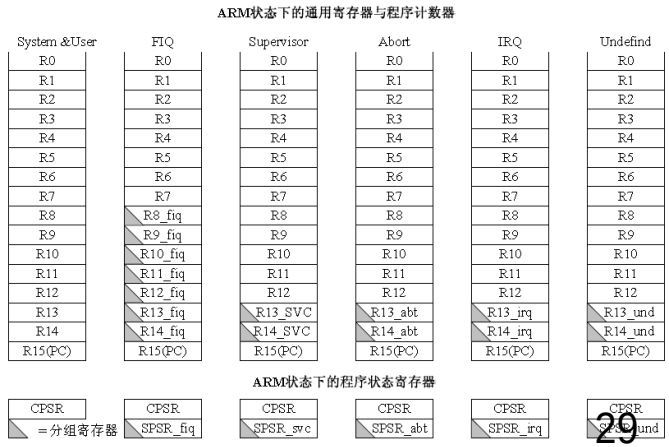
27

1 ARM嵌入式微处理器(续) --- 内部寄存器

- ARM9共有37个32位寄存器，可分成通用寄存器和状态寄存器
  - 31个通通用寄存器,保存数据或地址
  - 6个状态寄存器, 标识或设置存储器的工作模式或工作状态等,每个状态寄存器只使用了其中的12位
- 这37个寄存器根据处理器的工作状态及工作模式的不同而被分成不同的组。程序代码运行时涉及的工作寄存器组是由ARM9处理器的工作模式确定的

28

1 ARM嵌入式微处理器 --- 内部寄存器(续)



29

1 ARM嵌入式微处理器 --- 内部寄存器(续)

- 通用寄存器
  - 用字母R前缀加该寄存器的序号来标识，包括R0～R15寄存器，可分成未分组寄存器、分组寄存器及程序计数器三种
  - 未分组寄存器R0～R7
    - 在所有工作模式下，它们在物理上是同一个寄存器。也就是说，不管在哪种工作模式下，若访问R0寄存器，访问到的是同一个32位的物理寄存器R0；若访问R1寄存器，访问到的是同一个32位的物理寄存器R1；依次类推。由于不同的处理器工作模式均使用相同的未分组寄存器，可能会造成寄存器中数据的破坏
  - 分组寄存器R8～R14
    - 它们每一次所访问的物理寄存器与处理器当前的工作模式有关，对于R8～R12寄存器，每个寄存器对应两个不同的物理寄存器。当使用fiq模式时，访问寄存器R8_fiq～R12_fiq；当使用fiq模式以外的其他模式时，访问寄存器R8_usr～R12_usr

30

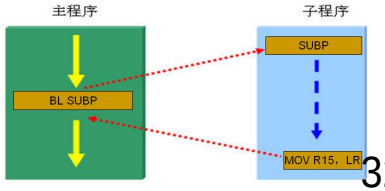
1 ARM嵌入式微处理器 --- 内部寄存器(续)

- 通用寄存器(续)
  - 分组寄存器R8～R14(续)
    - 对R13、R14，每个寄存器对应6个不同的物理寄存器，其中的1个是用户模式与系统模式共用；另外5个物理寄存器对应于其他5种不同的工作模式，采用R13_<mode>或R14_<mode>来区分不同的物理寄存器，<mode>为usr、fiq、irq、svc、abt、und这6种模式之一
    - R13寄存器在ARM指令中常用作堆栈指针，SP(Stack Pointer)，但这只是一种习惯用法，用户也可使用其他寄存器作为堆栈指针。而在Thumb指令集中，某些指令强制性地要求使用R13作为堆栈指针

31

1 ARM嵌入式微处理器 --- 内部寄存器(续)

- 通用寄存器(续)
  - 分组寄存器R8～R14(续)
    - R14寄存器可用作子程序链接寄存器（Subroutine Link Register）或链接寄存器LR（Link Register）。当ARM处理器执行带链接的分支指令BL时，R14中保存R15(程序计数器PC)的备份。当发生中断或异常时，对应的分组寄存器R14_fiq、R14_irq、R14_svc、R14_abt、R14_und用于保存R15的返回值。其他情况下，R14用作通用寄存器。即，R14有两种特殊功能：一是每种工作模式下所对应的那个R14可用于保存子程序的返回地址；二是当异常发生时，该异常模式下的那个R14被设置成异常返回地址



32

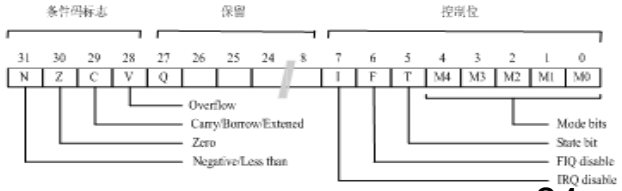
1 ARM嵌入式微处理器 --- 内部寄存器(续)

- 通用寄存器(续)
  - 程序计数器R15：用于控制程序中指令的执行顺序
    - 在ARM指令工作状态下，R15的位［1:0］是0，位[31:2]保存PC的值
    - 在Thumb指令工作状态下，位[0]为0，位[31:1]保存PC的值
    - 读R15寄存器的结果是读到的值为该指令地址加8(ARM状态)或加4(Thumb状态)
    - R15虽然也可用作通用寄存器，但一般不这么使用，因为对R15的使用有一些特殊的限制，当违反了这些限制时，程序的执行结果是未知的。
    - ARM指令始终是字对齐的，所以读出R15的结果值的位[1:0]总是0。读R15的主要作用是快速地对临近的指令和数据进行位置无关寻址，包括程序中的位置无关转移
    - 写R15的通常结果是将写到R15中的值作为指令地址，并以此地址发生转移。由于ARM指令要求字对齐，通常希望写到R15中值的位[1:0]=0b00

33

1 ARM嵌入式微处理器 --- 内部寄存器(续)

- 程序状态寄存器
  - 1个当前程序状态寄存器CPSR(Current Program Status Register)和5个备份的程序状态寄存器SPSR(Saved Program Status Register)，CPSR又称为R16
  - 在任何工作模式下，CPSR都是同一个物理寄存器，它保存了程序运行的当前状态，如条件码标志、控制允许和禁止中断、设置处理器的工作模式以及其他状态和控制信息等。每种异常模式都有一个备份的程序状态寄存器SPSR；当异常发生时，SPSR用于保存CPSR的状态



34

1 ARM嵌入式微处理器 --- 内部寄存器(续)

- 程序状态寄存器(续)
  - 条件码标志
    - CPSR寄存器的高4位是N、Z、C、V(Negative、Zero、Carry、oVerflow)，称为条件码标志位
    - 它们的内容可被算术或逻辑运算的结果所改变，并且可以决定某条指令是否被执行
    - CPSR中的条件码标志可由大多数指令检测以决定指令是否执行。在ARM状态下，绝大多数的指令都是有条件执行的。在Thumb状态下，仅有分支指令是有条件执行的
    - 条件码标志可以通过执行比较指令(CMN、CMP、TEQ、TST)、一些算术运算、逻辑运算和传送指令进行修改

35

1 ARM嵌入式微处理器 --- 内部寄存器(续)

- 程序状态寄存器(续)
  - 控制位,CPSR寄存器的低8位是I、F、T和M[4:0]
    - 当发生异常时，这些位可以被改变；当处理器运行在特权模式时，这些位也可以由程序修改
    - 中断禁止位：包括I和F，用来禁止或允许IRQ和FIQ两类中断。当I=1时，表示禁止IRQ中断；I=0时，表示允许IRQ中断。当F=1时，表示禁止FIQ中断；F=0时，表示允许FIQ中断
    - T标志位：T标志位用于标识/设置处理器的工作状态。对于ARM体系结构v4及以上版本的T系列处理器，当T=1时，表示程序运行于Thumb状态；当T=0时，表示程序运行于ARM状态。ARM指令集和Thumb指令集均有切换处理器状态的指令，这些指令通过修改T位的值为1或0来实现两种工作状态之间的切换
    - 工作模式位：工作模式位（M[4:0]）用于标识或设置处理器的工作模式。M4、M3、M2、M1、M0决定了处理器的工作模式

36

1 ARM嵌入式微处理器 --- 内部寄存器(续)

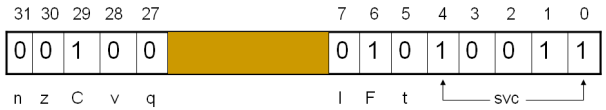
- 程序状态寄存器(续)
  - 控制位(续)

M[4:0]	模式	可访问的寄存器
10000	用户(usr)	PC、R14~R0、CPSR
10001	FIQ(fiq)	PC、R14_fiq~R8_fiq、R7~R0、CPSR、SPSR_fiq
10010	IRQ(irq)	PC、R14_irq、R13_irq、R12~R0、CPSR、SPSR_irq
10011	管理(svc)	PC、R14_svc、R13_svc、R12~R0、CPSR、SPSR_svc
10111	中止(abt)	PC、R14_abt、R13_abt、R12~R0、CPSR、SPSR_abt
11011	未定义(und)	PC、R14_und、R13_und、R12~R0、CPSR、SPSR_und
11111	系统(sys)	PC、R14~R0、CPSR

37

1 ARM嵌入式微处理器 --- 内部寄存器(续)

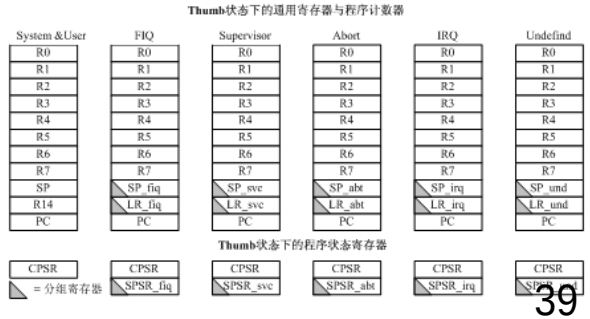
- 程序状态寄存器(续)
  - 控制位(续)
    - 例：设在程序运行某时刻， CPSR寄存器的值如图所示。试说明处理器的条件标志、中断允许情况、工作状态以及工作模式
    - 上述条件标志用符号可表示为nzCvq， 即C标志位置1，其他标志位为0。因为bit[7-6]为I和F，所以IRQ中断被使能，即允许CPU响应IRQ中断，FIQ中断被禁止。因为bit[5]为t，所以处理器工作在ARM状态。因为bit[4-0]为10011，系统工作于管理模式(svc)



38

1 ARM嵌入式微处理器 --- 内部寄存器(续)

- Thumb寄存器
  - Thumb状态下的寄存器集是ARM状态下寄存器集的一个子集，程序可以直接访问通用寄存器R0～R7、程序计数器PC、堆栈指针SP、连接寄存器LR和CPSR。在每种特权模式下都有对应的SP、LR和SPSR



39

1 ARM嵌入式微处理器 --- 内部寄存器(续)

- Thumb寄存器(续)
  - Thumb状态下寄存器的组织与ARM状态下寄存器的组织之间的关系：
    - Thumb状态和ARM状态的R0～R7是相同的；
    - Thumb状态和ARM状态的CPSR以及所有的SPSR是相同的；
    - Thumb状态的SP对应于ARM状态的R13；
    - Thumb状态的LR对应于ARM状态的R14；
    - Thumb状态的PC对应于ARM状态的R15
  - 在Thumb状态下， R8～R15寄存器并不是标准寄存器集的一部分，但可以使用汇编语言受限制地访问这些寄存器，将其用作快速的暂存器

40

1 ARM嵌入式微处理器 --- 异常

- 异常(Exception)是指处理器由于内部或外部的原因，停止执行当前的程序，转而处理特定的事件，处理完毕返回原来的程序继续执行
  - 在处理异常之前，处理器状态必须保留，以便在异常处理程序完成后，原来的程序能够重新执行
  - 同一时刻可能会出现多个异常，处理器按优先级对多个异常进行处理
  - 异常与中断的概念并不完全等同

41

1 ARM嵌入式微处理器 --- 异常(续)

- 异常的类型及向量地址

异常名称	对应模式	正常向量	高地址向量
复位	管理(svc)	0x00000000	0xFFFF0000
未定义指令	未定义(und)	0x00000004	0xFFFF0004
软件中断(SWI)	管理(svc)	0x00000008	0xFFFF0008
指令预取中止（取指令存储器中止）	中止(abt)	0x0000000C	0xFFFF000C
数据中止	中止(abt)	0x00000010	0xFFFF0010
IRQ（中断）	IRQ(irq)	0x00000018	0xFFFF0018
FIQ（快速中断）	FIQ(fiq)	0x0000001C	0xFFFF001C

42

1 ARM嵌入式微处理器 --- 异常(续)

- 异常的类型及向量地址(续)
  - 复位：处理器上一旦有复位信号输入，ARM处理器立刻停止执行当前指令，复位后，ARM处理器在禁止中断的管理模式下，从地址0x00000000或0xFFFF0000开始执行程序
  - 未定义指令异常：有两种情况:①当ARM处理器执行协处理器指令时，它必须等待任一外部协处理器应答后，才能真正执行这条指令。若协处理器没有响应，会出现未定义指令异常。②试图执行未定义的指令，也会出现未定义指令异常
  - 软件中断异常：是由软件中断指令SWI引起的。软件中断异常指令SWI进入管理模式，以请求执行特定的管理功能

1 ARM嵌入式微处理器 --- 异常(续)

- 异常的类型及向量地址(续)
  - 指令预取中止(prefetch abort):指令预取访问存储器失败时产生的异常。存储器系统向ARM处理器发出存储器中止(abort)信号，响应取指激活的中止，预取的指令被标记为无效，若处理器试图执行无效指令，则产生预取中止异常；若指令未执行，则不发生预取中止
  - 数据中止(data abort):ARM处理器访问数据存储器失败时产生的异常。存储器系统向ARM处理器发出存储器中止(Abort)信号，响应数据访问(加载/存储)激活的中止，数据被标记为无效
  - IRQ(中断请求):通过处理器上的nIRQ引脚输入低电平产生。IRQ异常的优先级比FIQ异常的低。当进入FIQ处理时，会屏蔽掉IRQ异常
  - FIQ(快速中断请求):通过处理器上的nFIQ引脚输入产生

1 ARM嵌入式微处理器 --- 异常(续)

- 异常的类型及向量地址(续)
  - 异常向量是异常服务程序的入口，在某些ARM应用中，允许异常向量的位置由32位地址空间低端的正常位置(0x00000000~0x0000001C)，移到地址空间高端的另一地址范围(0xFFFF0000~0xFFFF001C)。这些改变后的地址位置称为高端向量
  - 由Implementation Defined决定是否支持高端向量。如果支持，则在输入硬件配置时，选择是使用正常向量(normal vectors)还是高端向量(high vectors)
  - 应用程序中的异常处理
    - 1) 在异常向量表中的特定位置放置跳转指令，跳转到异常处理程序
    - 2) 当ARM处理器发生异常时，程序计数器PC会被强制设置为对应的异常向量，从而跳转到异常处理程序
    - 3) 当异常处理完成以后，返回到主程序继续执行

1 ARM嵌入式微处理器 --- 异常(续)

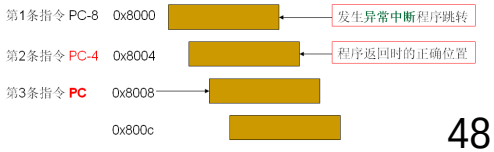
- 进入和退出异常(续)
  - 进入异常
    - 1) 将下一条指令的地址保存在相应的LR寄存器中。如果异常是从ARM状态进入，则保存在LR中的是下一条指令的地址(当前PC+4或PC+8，与异常的类型有关)。如果异常是从Thumb状态进入，则保存在LR中的是当前PC的偏移量。这样异常处理程序就不需要确定异常是从何种状态进入的(如：在软件中断异常SWI产生时，指令MOV PC,R14_svc总是返回到下一条指令，不管SWI是在ARM状态下执行还是在Thumb下执行)
    - 2) 将CPSR复制到相应的SPSR中
    - 3) 迫使CPSR模式位M[4：0]的值设置成对应的异常模式值
    - 4) 迫使PC从相关的异常向量取下一条指令
    - 5) 也可以设置中断禁止位来阻止其他无法处理的异常嵌套。如果异常发生时，处理器处于Thumb状态，当用中断向量地址加载PC时，自动切换到ARM状态

1 ARM嵌入式微处理器 --- 异常(续)

- 进入和退出异常(续)
  - 退出异常
    - 1) 将LR寄存器的值减去相应的偏移量,偏移量根据异常的不同而不同，送到PC中
    - 2) 将SPSR复制回CPSR中
    - 3) 清除中断禁止位标志

1 ARM嵌入式微处理器 --- 异常(续)

- 进入和退出异常(续)
  - 退出异常(续)
    - ARM9，程序执行完第1条指令(地址0x8000)发生跳转时，PC正指向第3条指令(地址0x8008)。在执行完异常中断服务程序返回时，PC应指向第2条指令(地址0x8004)
    - 第1条指令执行时，ARM处理器硬件自动把PC(=0x8008)保存至LR寄存器中，但接下去ARM处理器会马上对LR自动进行调整：LR=LR-0x4(即LR寄存器的值为0x8004)。这样，最终保存在LR中的是第2条指令的地址，因此从异常中断返回时，LR中正好是正确的返回地址
    - 由于各种异常中断响应的过程不同，因此，保存在LR中的地址是不同的。大多数情况下，保存在LR中的地址值是LR保存的值=PC值-4



1 ARM嵌入式微处理器 --- 异常(续)

- 进入和退出异常(续)
  - 退出异常(续)
    - 因为保存在LR中的地址值是不同的，因而，不同的异常中断返回时的指令也不尽相同。下表总结了进入异常处理时保存在相应的LR(R14)寄存器中的PC值，以及在退出异常处理时推荐使用的返回操作指令，以实现处理器返回断点处

发生转移条件	返回操作	R14当前内容	
		ARM状态	Thumb状态
子程序调用	MOVS PC，R14	PC+4	PC+2
软件中断异常	MOVS PC,R14_svc	PC+4	PC+2
未定义异常	MOVS PC,R14_und	PC+4	PC+2
FIQ异常	SUBS PC,R14_fiq,#4	PC+4	PC+4
IRQ异常	SUBS PC,R14_irq,#4	PC+4	PC+4
指令预取中止异常	SUBS PC,R14_abt,#4	PC+4	PC+4
数据中止异常	SUBS PC,R14_abt,#8	PC+8	PC+8
复位	_	_	_

49

1 ARM嵌入式微处理器 --- 存储器组织结构

- 存储器的地址空间可被看成是从0地址单元开始的字节的线性组合，一个地址对应于一个存储字节
- 32位长度地址
  - 通常字节地址是无符号整数，则字节地址范围是0~2^32(16进制:x00000000~0xffffffff)，存储容量：2^32=4GB字节
- 大端存储和小端存储
  - 大端格式（big-endian）：字的地址对应的是该字中最高有效字节所对应的地址；半字的地址对应的是该半字中最高有效字节所对应的地址。32位字数据的最高字节存储在低字节地址中，而其最低字节则存储在高字节地址中
  - 小端格式（low-endian）：字的地址对应的是该字中最低有效字节所对应的地址；半字的地址对应的是该半字中最低有效字节所对应的地址。32位字数据的最高字节存储在高字节地址中，而其最低字节则存储在低字节地址中

50

1 ARM嵌入式微处理器 --- 存储器组织结构(续)

- 大端存储和小端存储(续)

51

1 ARM嵌入式微处理器 --- 存储器组织结构(续)

- 大端存储和小端存储(续)
  - 小端存储格式是ARM默认的格式。ARM汇编指令集中，没有相应的指令来选择是采用大端存储格式还是小端存储格式，但可以通过硬件输入引脚来配置它。若要求ARM目标系统支持小端存储格式，则将引脚BIGEND接低电平，否则接高电平
  - ARM体系结构对于存储器单元的访问需要适当地对齐，即访问字存储单元时，字地址应该是字对齐（地址能被4整除）；访问半字存储单元时，半字地址应该半字对齐（地址能被2整除）。如果不按对齐的方式访问存储单元，称作非对齐的存储器访问。非对齐的存储器访问可能会导致不可预知的状态

52

1 ARM嵌入式微处理器 --- 存储器组织结构(续)

- I/O端口和存储器统一编址
  - ARM9体系结构使用存储器映射方式实现I/O端口的访问。为每个I/O端口分配特定的存储器地址，当从这些地址读出或向这些地址写入时，实际上就完成了I/O功能

53

2、ARM 存储器

54

5.1 存储器概述

55

1 存储器概述

存储器按存取速度和在计算机系统中的地位存储器分为两大类：

(1) 主存储器：速度较快，容量较小，价格较高，用于存储当前计算机 运行所需要的程序和数据，可与CPU直接交换信息，习惯上称为主存， 又称内存(内部存储器)。

(2) 辅存储器：速度较慢，容量较大，价格较低，用于存放计算机当前 暂时不用的程序、数据或需要永久保持的信息,辅存又称外存(外部存储器)或海量存储器。

外存要配备专门的设备才能完成对外存的读写。通常将外存归入到计算机外设类。

56

1 存储器概述(续)

存储器分类

内存：RAM、ROM
外存：软盘、硬盘、光盘、磁光盘MO、U盘

57

1 存储器概述(续)

主存储器的分类

- 按存储功能分
  - 只读存储器（ROM）
  - 随机存储器（RAM）
- 按制造工艺分
  - 双极性
  - MOS型

58

1 存储器概述(续)

主存储器的分类

- 半导体存储器
  - ROM
    - 掩膜型ROM 可编程
    - ROM (PROM)
    - 电可擦除可编程ROM (EEPROM)
    - 紫外线可擦除可编程ROM (EPROM)
    - 快闪存储器 (Flash Memory)
  - RAM
    - 静态RAM (SRAM) —— 通常用于计算机的Cache
    - 动态RAM (DRAM) —— 主要用于计算机的内存条
    - 组合RAM (IRAM) —— 将刷新电路与DRAM集成在一起

59

1 存储器概述(续)

只读存储器分类：

掩膜ROM：出厂后内部存储的数据不能改动，只能读出。

PROM：可编程，只能写一次。

EPROM：用紫外线擦除，擦除和编程时间较慢，次数也不宜多。

E^2PROM：电擦除，擦除和写入时需高电压脉冲，擦、写时间较长。

快闪存储器(Flash Memory)：吸收了EPROM结构简单，编程可靠的优点，又保留了E^2PROM用隧道效应擦除的快捷特性，集成度很高。可靠性一般不如E^2PROM。

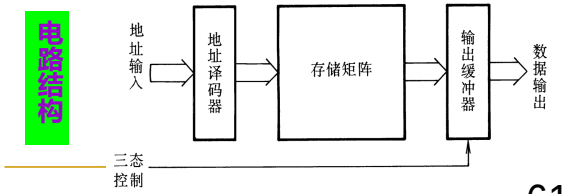
60

1 存储器概述(续)

Read Only Memory

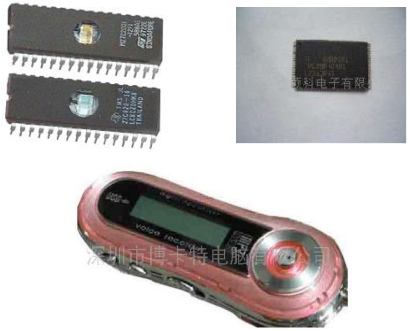
优点：电路简单，断电后数据不丢失，具有非易失性。

缺点：只适用于存储固定数据的场合。



61

1 存储器概述(续)



只读存储器芯片图

62

1 存储器概述(续)

- Compact Flash, CF卡

CF卡是利用Flash技术的存储卡，接口具有PCMCIA- ATA功能，可以工作在IDE接口模式，也可以工作在PC Card模式。衍生出来的CF+卡物理规格和CF完全相同，在手持设备上应用，如CF串口卡、CF ModemCF蓝牙、CF USB卡、CF网卡、CF GPS卡、CF GPRS卡等。按照CF+卡标 准，它不一定要支持ATA接口。通常建议CF+卡工作在 PCMCIA模式。CF卡可以看作是PCMCIA卡的一个子集，可以 通过物理上的转换器，直接转换成PCMCIA卡使用。

63

1 存储器概述(续)

- Secure Digital Card, SD卡
  - 由日本Panasonic、TOSHIBA和美国 SanDisk公司共同开发研制的SD卡是一种全新的存储卡 产品，在MP3、数码摄像机、数码相机、电子图书及AV 器材等中广泛应用。
  - SD存储卡采用一个完全开放的标准（系统），比MMC卡略厚，具有更大的容量，兼容MMC卡 接口规范。
  - SD卡具有加密功能，可以保证数据资料的安全保密。
  - SD卡具有版权保护技术，所采用的版权保护技 术是DVD中使用的CPRM技术（可刻录介质内容保护）。

64

1 存储器概述(续)

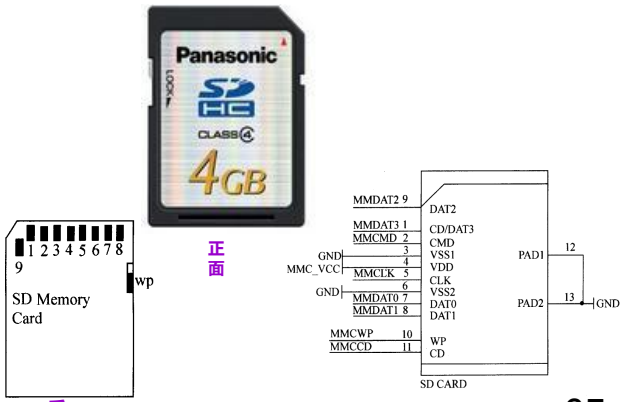


图 SD卡的外形和接口原理图

65

1 存储器概述(续)

随机存储器（RAM)

Random Access Memory

优点：读、写方便，使用灵活。

缺点：一旦停电所存储的数据将随之丢失（易失性）

基本结构：地址译码器、存储矩阵和读\写控 制电路构成

66

1 存储器概述(续)

随机存储器芯片图

1 存储器概述(续)

存储系统的层次结构

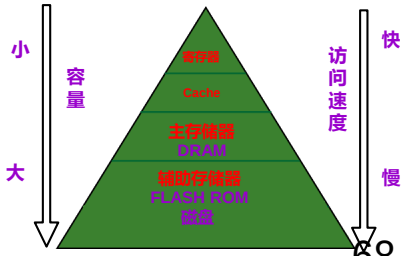
存储器是用来存储信息的部件，是嵌入式系统硬件中的 重要组成部分。在复杂的嵌入式系统中，存储器系统的 组织结构按作用可以划分为4级：

寄存器

Cache

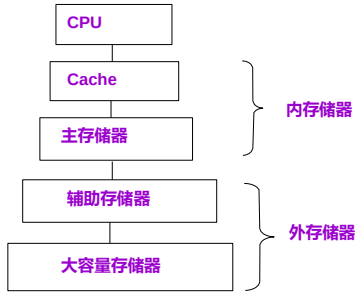
主存储器

辅助存储器



1 存储器概述(续)

越靠近CPU的存 储器速度越快而 容量越小



1 存储器概述(续)

SRAM和DRAM

RAM：随机存取存储器

SRAM：静态随机存储器

DRAM：动态随机存储器

DRAM是常用的RAM。和其他RAM相比，它每MB的价格最低。不过 DRAM需要动态地刷新，因此在使用DRAM之前要先设置好DRAM控制器。

SDRAM (Synchronous DRAM:同步动态随机存储器)是众多DRAM中的 一种，它能够工作在比普通DRAM更高的时钟频率下。因为SDRAM使用时钟，所以它和处理器总线是同步的。数据从存储器中被流水化地取出， 最后突发(burst)地传输到总线，因而传输效率高。

SRAM读/写速度比DRAM读/写速度快；

SRAM比DRAM功耗大；

DRAM的集成度可以做得更高，则其存储器容量更大；

DRAM需要周期性的刷新，而SRAM不需要。

1 存储器概述(续)

NAND Flash简介

以页为单位进行读和编程操作，以块为单位进行擦除操作。

数据、地址采用同一总线。实现串行读取。随机读取速度慢且不能按字节随机编程。

芯片尺寸小、引脚少，是位成本最低的固态存储器。

1 存储器概述(续)

NAND Flash操作

K9F1208芯片有4096个Block，每个Block有32个Page， 每个Page有528个Byte，Block是NAND Flash中最大的操作单元，擦除是以Block为单位完成的，而编程和读取是以Page为单位完成的。因此，对NAND Flash的操作要形 成以下三类地址：

1)块地址(Block Address)；

2)页地址(Page Address)；

3)页内地址/列地址(Column Address)；

由于NAND Flash的数据线和地址线是复用的，因此，在传送地址时要用3/4个时钟周期来完成。

内容提要

1	存储器概述
2	*存储系统机制*

2.1 存储器接口方式

SRAM型的全地址/数据总线接口：这种类型的地址线数目和片内存储单元数一一对应，接口比较简单。拥有此类接口的存储器有SRAM、EPROM、EEPROM、NOR Flash等。

DRAM型动态存储器接口：存储单元需要定期地刷新。CPU与其接口的信号线除了有与SRAM相同的信号线外，还有RAS（行地址选择）信号线和CAS（列地址选择）信号线。一般和具有动态存储器控制器的CPU相连接。拥有此类接口的存储器有 DRAM、SDRAM、DDR SDRAM等。

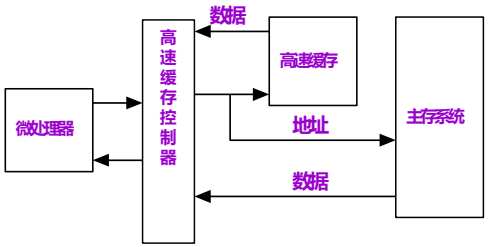
串行存储器接口：与CPU以串行的方式传送地址和数据，传送 速度相对较慢，多用于嵌入式系统的辅助存储器。拥有此类接口的存储器有NOR Flash、串行EEPROM、串行SRAM等。

2.2 高速缓存机制

高速缓存控制器是微处理器用于控制访问高速缓存及主存系统的桥梁，它处于微处理器和高速缓存及主存系统之间用于解决主存访问速度与CPU处理速度不相匹配的一种部件（由集成于CPU芯片中的专门的高速存取电路实现）。

或用于解决辅存访问速度与CPU处理速度不相匹配的一种部件（由主存的一部分实现）。

2.2 高速缓存机制(续)



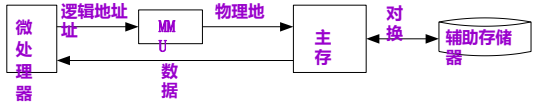
2.3 存储管理单元MMU

存储管理单元（MMU）是集成在微处理器芯片内部、专 门管理外部存储器总线的一部分硬件。主要用来完成虚实 地址和物理地址之间的转换。目前，越来越多的微处理器 芯片均带有存储管理单元（MMU）。MMU完成的主要功 能有：

将主存地址从虚拟存储空间映射到物理存储空间。

存储器访问权限控制。

设置虚拟存储空间的缓冲特性等。



2.3 存储管理单元MMU(续)

MMU管理方式

分段方式：分段方式支持较大的、任意大小的内存区域

分页方式：分页方式支持较小的、固定大小的内存区域

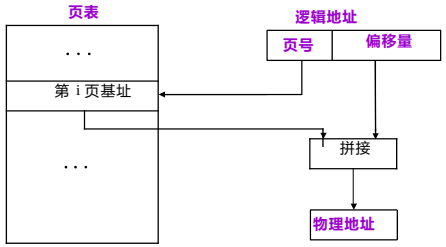
段页方式：段页方式介于分段方式和分页方式之间

每种方式都有其特点

2.3 存储管理单元MMU(续)

分页虚拟存储管理

虚拟存储空间分成一个个固定大小的页，把物理主存储的空间也分成同样大小的一个个的页。通过查询存放在主存中的页表，来实现虚拟地址到物理地址的变换。



79

ARM 时钟及电源管理（以S3C2410为例）

80

1 S3C2410时钟结构

- **S3C2410时钟与电源管理模块包括**
  - **◆时钟控制**
  - **◆USB控制**
  - **◆电源控制**

81

1 S3C2410时钟结构(续)

- 时钟控制逻辑可以产生系统所需要的时钟信号，包括提供给CPU的FCLK，提供给AHB总线设备的 HCLK 和提供给APB总线设备的PCLK
- S3C2410 有 2 个 锁 相 环 (PLLs)
  - ◆一 个 提 供 FCLK、HCLK和PCLK
  - ◆另一个提供USB时钟(48MHz)
- 时钟控制逻辑可以产生不带锁相环的低速时钟，并可由软件控制是否提供给某个设备模块，这样有利于降低功耗

82

1 S3C2410时钟结构(续)

主时钟来源于外部晶振(XTIpll)或外部时钟(EXTCLK)。

时钟发生器包含一个连接外部晶振的振荡器，两个产生高频时钟的锁相环(PLLs)。

两个时钟源依据操作模式控制引脚(OM3和OM2)的不同组合来选择。

OM[3:2]	MPLL 状态	UPLL状态	主时钟源	USB时钟源
00	On	On	晶振	晶振
01	On	On	晶振	外部时钟
10	On	On	外部时钟	晶振
11	On	On	外部时钟	外部时钟

83

84

2 S3C2410电源管理模式

S3C2410有4种电源管理模式

1. 空闲模式(IDLE Mode)

FCLK被关断，主要用于CPU core节电。可以任何通过外 部中断唤醒 。

2. 正常模式(NORMAL Mode)

耗电最大、可以通过关闭具体控制器的时钟来节电 。

3. 低速模式(SLOW Mode)

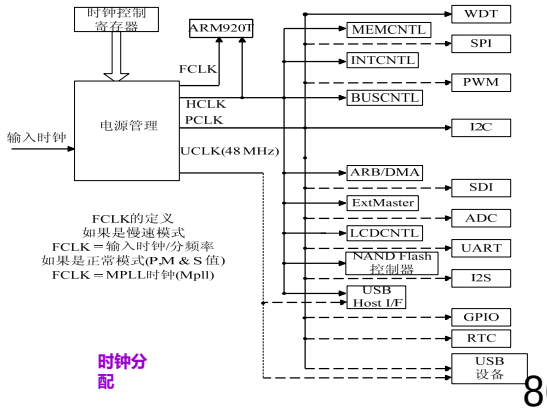
在此模式下可以没有内部PLL，耗电情况依赖于外部时钟 的频率 。

4. 休眠/掉电模式(POWER-OFF Mode)

除了处理器唤醒逻辑单元外，处理器不损耗任何电量。可以通过EINT[15:0]或 RTC 报警唤醒系统 。

85

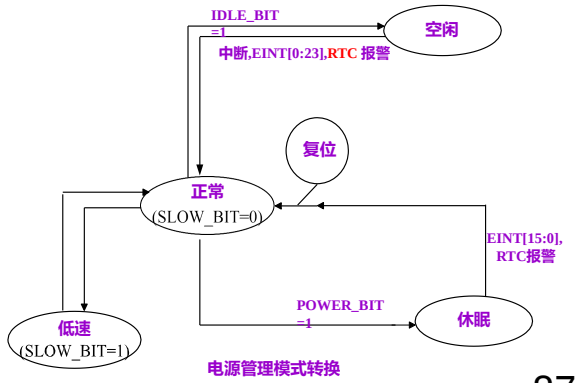
2 S3C2410电源管理模式(续)



时钟分配

86

2 S3C2410电源管理模式(续)



电源管理模式转换

87

2 S3C2410电源管理模式(续)

各种模式下时钟和电源状态

模式	ARM920T	AHB模块(1)/WDT	电源管理模块	GPIO	32.768KHz RTC时钟	APB模块(2)和USB主控/LCD/NAND
正常	工作	工作	工作	可选	工作	可选
低速	工作	工作	工作	可选	工作	可选
空闲	停止	工作	工作	可选	工作	可选
休眠	断电	断电	等待唤醒事件	前一个状态	工作	断电

88

2 S3C2410电源管理模式(续)

■正常模式

◆所有外围设备和基本模块包括电源管理模块、CPU核、总线控制器、存储控制器、中断控制器、 DMA 和外部控制单元都在运行，但每一个外围设备的时钟 (不含基本模块),都可以通过软件控制运行或停止,以便降低功耗

■空闲模式

◆停止供给CPU核时钟，但总线控制器、存储控制器、中断控制器和电源管理模块仍然有时钟供给。若退出空闲模式，需要激活EINT[23:0]，或者RTC中断，或其它中断

■低速模式

◆即无PLL模式(Non-PLL Mode)，通过低速时钟频率来达到降低功耗的目的。此时PLL不参与时钟电路，FCLK是外部输入时钟(XTIpll或EXTCLK)的一个n分频。 分频比率由两个控制寄存器CLKSLOW和 CLKDIVN的SLOW_VAL值来决定的

■休眠模式

◆模块断开内部电源连接(唤醒逻辑除外)。 休眠模式有效的前提是系统需要两套独立的电源，其中一套给唤醒逻辑供电，另一套则给包括CPU的其他设备供电，并且电源上电可控制。在休眠模式，给CPU和内部逻辑供电的第二套电源被关闭。可以由EINT[15:0]或通过预设系统启动 时间的中断将系统从休眠模式下唤醒

89

3 常用单元电路设计

■ 电源电路设计

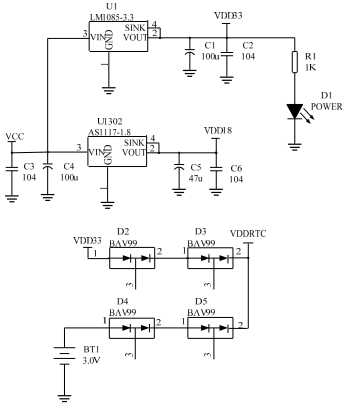
◆ S3C2410电源引脚分析

- VDDalive引脚给处理器复位模块和端口状态寄存器提供1.8V电压，无论是在正常模式还是在休眠模式，VDDalive都应该供电；
- VDDi和VDDiarm为处理器内核提供1.8V电压；
- VDDi_MPLL为MPLL提供1.8V模拟电源和数字电源；
- VDDi_UPLL为UPLL提供1.8V模拟电源和数字电源；
- VDDOP和VDDMOP分别为处理器端口和存储端口提供3.3V电压；
- VDDA_ADC为处理器内的ADC系统提供3.3V电压；
- VDDRTC为时钟电路提供1.8V电压，该电压在系统掉电后仍需要维持

90

3 常用单元电路设计(续)

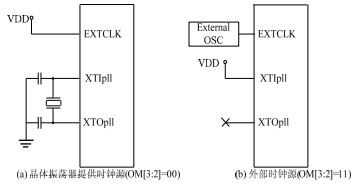
电源电路设计



91

3常用单元电路设计(续)

晶振电路设计



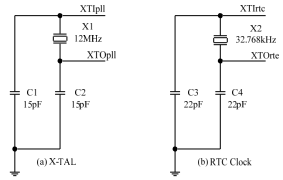
(a) 晶体振荡器提供时钟源(OM[3:2]=00) (b) 外部时钟源(OM[3:2]=11)

92

3 常用单元电路设计(续)

晶振电路设计

以OM[3:2]均接地的方式为例，即采用外部振荡器提供系统时钟。外部振荡 器由10-20MHz晶振和2个15-20PF的电容组成。振荡电路输出接到S32410X微处 理器的XTIPll脚，输入由XTOPll脚提供。如果是15MHz的晶振，经过S3C2410X 片内的PLL电路**倍频**后，最高可达203MHz。由于片内的PLL电路兼有倍频和时钟信号整形的功能，因此，系统可以以较低的外部时钟信号获得较高的工作频率， 从而降低外部振荡电路因高速开关所造成的高频噪声。产生RTC时钟的振荡电路 与系统时钟振荡电路采用相同的方式。如下图。



(a) X-TAL (b) RTC Clock

93

ARM 定时技术

94

1 定时器工作原理

- 定时器是嵌入式系统的常用部件，其主要用作定时功能或计数功能。不同的定时部件在使用上有所差异，但它们的逻辑原理是相同的。
- S3C2410芯片中的定时部件有多个
  - ◆看门狗定时器
  - ◆RTC部件
  - ◆Timer部件

95

2 S3C2410看门狗定时器

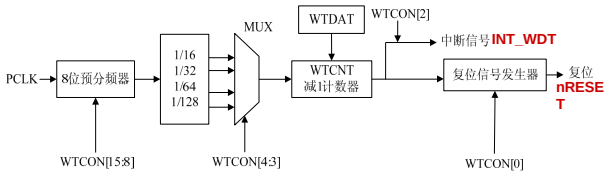
1) 看门狗概念

当嵌入式系统运行时受到外部干扰或者系统 错误，程序有时会出现“跑飞”，导致整个系统瘫 痪。在对系统稳定性要求较高的场合，为了防 止这一现象的发生，需要一种叫“看门 狗”(Watchdog)的电路。看门狗的作用就是当系统“跑飞”而进入死循环时，恢复系统的运行。

看门狗：是一种电路，具有监视并恢复程序 正常运行的功能。它是一定时器电路。

96

2 S3C2410看门狗定时器(续)



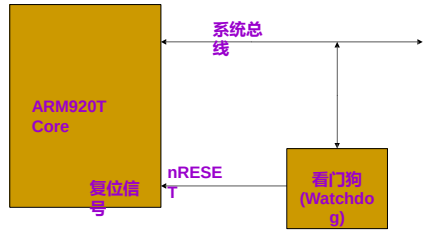
2 S3C2410看门狗定时器(续)

S3C2410 的看门狗定时器有两个功能：

（1）定时器功能：可以作为常规定时器使用，它是一个十六位的定时器，并且可以产生中断，中断名为INT_WDT，中断号(中断偏移量) 是0x09。

（2）复位功能：作为看门狗定时器使用，当时钟计数减为0(超时)时，它将产生一个128个时钟周期的复位信号nReset (用于ARM920T核的复位)。

2 S3C2410看门狗定时器(续)

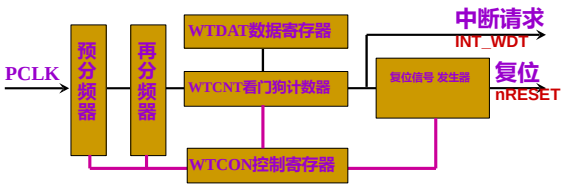


看门狗作用示意图

2 S3C2410看门狗定时器(续)

S3C2410 ARM9的看门狗主要由五部分构成： 时钟、看门狗计数器、看门狗数据寄存器、复位 信号发生器、控制逻辑等。

S3C2410 ARM9的看门狗工作原理：



2 S3C2410看门狗定时器(续)

注意：

一旦看门狗的定时器启动工作，其数据存器寄WTDAT）中的值将不会自动读到时间寄 存器 (WTCNT)中去。

由于这个原因，程序员必须在看门狗定时 器启动之前，应该将一个初始值写入到看门狗 的时间计数器(WTCNT)中间去 。

即先对时间计数器(WTCNT)赋初值，再启 动看门狗工作。

2 S3C2410看门狗定时器(续)

看门狗的基本工作原理可叙述为：

设一系统程序完整运行一周期的时间是Tp，
看门狗的定时周期为Ti，要求Ti>Tp。

在程序运行一周期后，修改定时器的计数值，只要程序正常运行，定时器就不会溢出。

若由于干扰等原因使系统不能在Tp时刻修改定时器的计数值，定时器将在Ti 时刻溢出，引发系统复位，使系统得以重新运行，从而起到监控作用

3 S3C2410 RTC部件

1) 概述

RTC（Real Time Clock: 实时时钟）

功能： 通常采用RTC 来提供可靠的系统时间，包括时、分、秒 和年、月、日等；而且要求在系统处于关机状态下它也 能够正常工作（通常在系统电源关闭后，由后备电池供 电），它的外围也不需要太多的辅助电路，典型的就是 只需要一个高精度的32.768KHz晶振和电阻电容等。

RTC应用：现在很多电子产品都有RTC功能，如 电子日历（台式、壁式等）、手持数码产品（手机、电 子词典、各种学习机、照相机、摄像机等）、电子计量 仪表（电度表、燃气表、水表等）、家用电器（电视机、 机顶盒、DVD等）等，应用非常广泛。

3 S3C2410 RTC部件(续)

S3C2410的RTC的特点：

时钟数据采用BCD 编码

时钟数据有： 时、分、秒、年、月、日、星期

能够对闰年的年月日进行自动处理

具有告警功能，当系统处于关机状态时，能产生告警中断

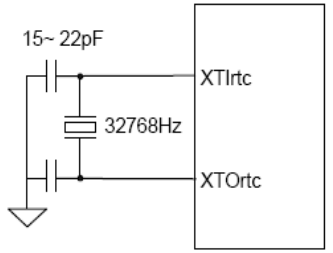
具有独立电源输入

提供毫秒级时钟中断，该中断可用作嵌入式操作系统的内核时钟

3 S3C2410 RTC部件(续)

S3C2410 RTC的振荡电路

S3C2410 RTC的只需外接2个20pf左右的小电容、32.768KHz的晶振即可。如下图所示



3 S3C2410 RTC部件(续)

S3C2410 RTC的振荡电路(Cont.)

RTC的时间片计数器用于产生一个中断请求，TICNT寄存器有一个中断使能位，和计数器中的值一起用来控制中断。当计数器的值变 为0时，引起时间片计时中断(INT_TICK)。中 断信号周期用下列公式计算：

周期=(n+1)/128 (s)

n=1～127(n**为时间片计数器中的值**)。

RTC **时间片计数器可用来产生实时操作系统内核所 需的时间片。**

3 S3C2410 RTC部件(续)

可能会引起的显示错误

例如，假设用户在2006 年12 月31 日23 点59 分59秒读取寄存器BCDYEAR 到BCDSEC， 在用户读取BCDSEC 寄存器时，如果结果是0， 那么很有可能年、月、日、时、分已经变成了 2006 年1 月1 日0 时0 分了，数据组合在一起可 能是错的。读取的数据可能是：

1） 2006 年12 月1 日0 时0分

2） 2006 年1 月1 日0 时0 分等

解决的方法：当读取到的BCDSEC 等于0 时，用户应该**再读取一次**BCDYEAR到BCDSEC的值。

3 S3C2410 RTC部件(续)

报警功能

RTC的报警寄存器(RTCALM)决定了报警的使能、禁止以及报警时间设定的条件。

在RTC报警使能情况下：

(1) 在正常工作模式下，报警中断(ALMINT) 是激活状态的。

(2)在掉电模式下(PWDN信号有效)，电源 管理唤醒信号(PMWKUP)与报警中断（ALMINT)都是激活状态。

3 S3C2410 RTC部件(续)

时间片中断 (节拍中断: INT_TICK)

RTC时间片用于中断请求。

TICNT 寄存器：有中断使能位、时间片计数位。

当时间片计数器中的值到达0 时，就会触发时间片中断INT_TICK。时间片中断的间隔时间计算如下：

Period=(n+1)/128 秒

n: 时间片计数值(1~127)

说明：RTC时间片中断可以作为RTOS实时操作系统)内核的时间节拍。

109

4 S3C2410 Timer部件及PWM

Timer部件主要是用于提供定时功能、脉宽调制（PWM: Pulse Width Modulation）功能的部件，它的应用比较灵活，对于需要一定频率的脉冲信号、一定时间间隔的定时信号的应用场合，它都能提供应用支持。

110

4 S3C2410 Timer部件及PWM(续)

- S3C2410定时器的主要特性
  - 5个16位定时器；
  - 2个8位预分频器和2个4位分频器；
  - 可编程PWM输出占空比；
  - 具有初值自动重装连续输出模式和单脉冲输出模式；
  - 具有死区(dead zone)发生器。
  - 定时器0~3具有 PWM(脉宽调制)功能。定时器4是一个内部定时器，没有输出引脚，供内部使用。定时器0有死区发生器，用于大电流设备控制。
  - 有2个8位预分频器和2个4位分频器。定时器0/1分享同一个8位的预分频器和分频器，定时器2/3/4分享另一个预分频器和分频器，分频器有1/2、1/4、1/8、1/16这4种分频值。定时器从分频器接收自己的时钟信号， 时钟分频器从相应的预分频器接收时钟信号

111

4 S3C2410 Timer部件及PWM(续)

PWM(脉宽调制)概念

PWM（脉宽调制）：就是只对一方波序列信号的占 空比按照要求进行调制，而不改变方波信号的其它参数，即不改变幅度和周期，因此脉宽调制信号的产生和传输， 都是数字式的。

用脉宽调制技术可以实现模拟信号：如果调制信号的频率远远大于信号接收者的分辨率，则接收者获得的是信号的平均效果，不能感知数字信号的0和1，其信号大小的平均值与信号的占空比有关，信号的占空比越大，平均信号越强，其平均值与占空比成正比。只要带宽足够(频率足 够高或周期足够短)，任何模拟信号都可以使用PWM 来实现。

PWM技术的应用：借助于微处理器，使用脉宽调制方 法实现模拟信号是一种非常有效的技术，广泛应用在从测 量、通信到功率控制与变换的许多领域中。

112

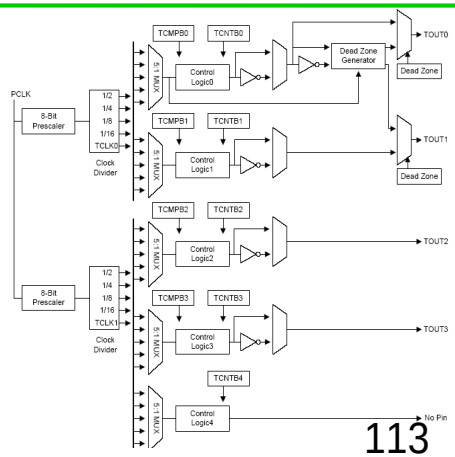
4 S3C2410 Timer部件及PWM(续)

定时器结构

(1) 时钟控制：系统为每个定时器设置有：预分频器、分 频器。

(2) 定时器组成（5 部分）：

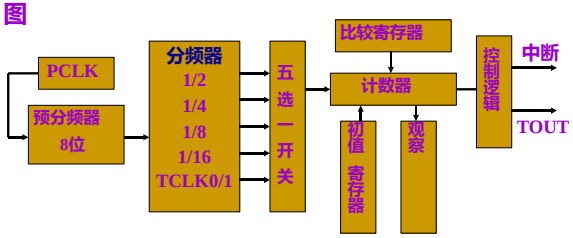
减法计数器(TNTn)、初值寄存器(TCNTBn)、比较寄存器(TCMPBn)、观察寄存器(TCNTOn)、控制逻辑等部分构成。



113

4 S3C2410 Timer部件及PWM(续)

定时器结构图



寄存器

初值寄存器(定时器计数缓冲寄存器)TCNTBn

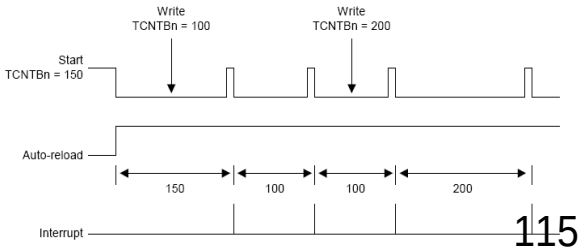
114

4 S3C2410 Timer部件及PWM(续)

工作原理

（1）定时器工作过程

装入初值、启动计数，计数结束产生中断请求，并且可以重装初 值连续计数。S3C2410的PWM定时器具有双缓冲功能，该功能可以 在不停止当前定时器操作的情况前下，重新加载为下一轮定时器操作 而改变的值，如下图所示。



115

4 S3C2410 Timer部件及PWM(续)

（2）初值自动重装、手动装载和双缓冲

初值自动重装功能：5个定时器都具有此功能。当 计数器中的值减到0后，若设置了自动重装功能，则在 下一计数周期开始前将初值装入计数器重新计数。

初值手动装载功能：在启动计数前，必须使用手动 装载功能将初值装入计数器，而初值自动重装仅是一 次计数结束后重新装入初值。

双缓冲功能：如果定时器正在工作，此时写入新 的数据到TCNTBn或者到TCMPBn，该写入的数据不 影响本次定时器的操作。当定时器到达0后下一次运行 定时器时，新写入的TCNTBn或者TCMPBn才生效。

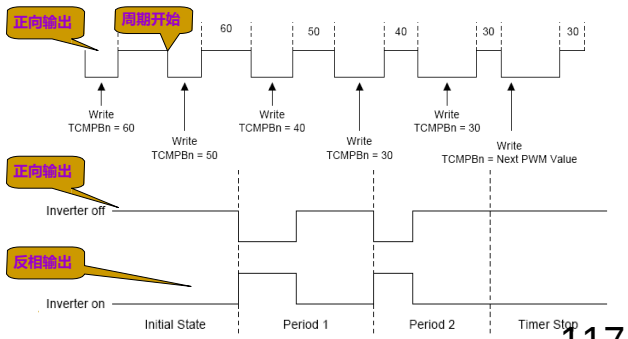
116

4 S3C2410 Timer部件及PWM(续)

（3）PWM输出

寄存器TCMPB的作用：当计数器TCNT中的值减到与TCMPB的 值相同时，TOUT的输出值取反。改变TCMPB的值，便改变了输出 方波的占空比。

TOUT的输出可以设置为反相输出，如下图所示。



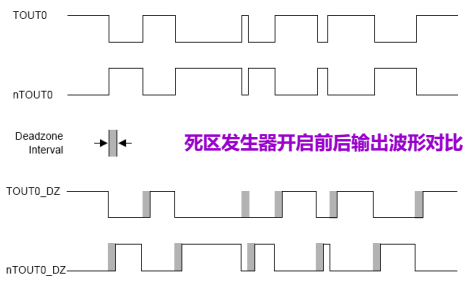
117

4 S3C2410 Timer部件及PWM(续)

（4）死区发生器

死区的概念：是一小段时间间隔，在这个时间间隔内，禁止两个开关同时处于开启状态。死区发生器是在大功率设备PWM控制中常采用的 一种技术，该功能用于在一个开关设备的断开和另一个开关设备的闭合之间插入一个时间间隔，这个时间间隔使得两个开关设备不可能同时被 打开，即使是很短的一段时间也是如此。

S3C2410的timer0具有死区发生器功能，可用于控制大功率设备。



118

4 S3C2410 Timer部件及PWM(续)

（5）DMA请求模式

S3C2410中定时器的DMA功能：系统中的5个定时器 都有DMA请求功能，但是在同一时刻只能设置一个使用 DMA功能，通过设置其DMA模式位来实现。

DMA请求过程：定时器可以在任意时间产生DMA请 求，并且保持DMA请求信号(nDMA_REQ)为低直到定时器 收到ACK信号。当定时器收到ACK信号时，它使请求信号 变得无效。

DMA请求与中断的关系：如果一个定时器被配置为 DMA模式，该定时器不会产生中断请求了；其他的定时器 会正常的产生中断。

119

4 S3C2410 Timer部件及PWM(续)

计数时钟和输出计算

1）定时器输入时钟频率f_{Tclk} (即计数时钟频率)：

$f_{Tclk}=[f_{pclk}/(Prescaler+1)]$x分频值

式中：Prescaler，预分频值，0---255；分频值为1/2、1/4、1/8、1/16。

2）PWM输出时钟频率：

PWM输出时钟频率= f_{Tclk} / TCNTBn

3）PWM输出信号占空比(即高电平持续时间所占信号周期的比例)：

PWM输出信号占空比 = TCMPBn/TCNTBn

120

4 S3C2410 Timer部件及PWM(续)

定时器最大、最小输出周期

设PCLK的频率为50MHz，经过预分频和分频器后，送给定时器 的可能计数时钟频率由下表给出。

分频值	最小输出周期 (预分频器=0、TCNTBn=1)	最大输出周期 (预分频器=255、TCNTBn=65535)	最小输出周期 (预分频器=0、TCNTBn=65535)	最小输出周期 (预分频器=0、TCNTBn=255)
1/2	25.00MHz(0.04μs)	0.6710s	381Hz	97656
1/4	12.50MHz(0.08μs)	1.3421s	191Hz	48828
1/8	6.250MHz(0.16μs)	2.6843s	95Hz	24414
1/16	3.125MHz(0.32μs)	5.3686s	48Hz	12207

121

ARM 中断

122

1 中断概述

- 微处理器与外设间数据传送方式简介
  - 嵌入式系统中，微处理器与外设之间的数据传送方式可以有3种：程序查询方式、中断方式、DMA方式；其中，程序查询方式和中断方式是常用的两种数据传送方式
  - 程序查询方式：是由微处理器周期性地执行一段查询程序来读取I/O端口或部件中状态寄存 器的内容，并判断其状态，从而使微处理器与 I/O端口或部件在进行数据、命令传送时保持同步
    - 程序查询方式特点： 程序查询方式下，微处理器的效率是非常低的，因为微处理器要花费大量的时间检测 I/O端口或部件的状态。并且，I/O端口或部件的数据也不能得到实时地处理

123

1 中断概述(续)

- 微处理器与外设间数据传送方式简介(续)
  - 中断方式：是I/O端口或部件在完成了一个I/O操作后，产生一个信号给微处理器， 这个信号叫做“中断请求”，微处理器响应这个请求信号，停止其当前的程序操作，而转向对该I/O端口或部件进行新的读/ 写操作
    - 中断方式特点：(1) 实时性能好；(2) 调试较复杂
  - DMA方式：DMA传送方式是在存储器和I/O部件之间存储器和存储器之间直接进行数据传送(如高速数据采集、内存和内存间的高速数据块传送等)，传送过程不需要微处理器的干预，这样，在传送时就不必进行保护现场等一系列额外操作，传输速度基本取决于存储器和外设的速度
    - 在S3C2410内部就有许多I/O部件可以采用DMA方式进行传送，当然也可以支持芯片外部I/O部件的DMA传送需求

124

1 中断概述(续)

- 微处理器与外设间数据传送方式简介(续)
  - DMA方式与中断方式传送数据相比较，有如下的特点
    - (1) 中断方式下，MPU需要执行多条指令，占用一定的时间；而DMA传送1个字节只占用MPU的1个总线周期，占用MPU的时间少。
    - (2) DMA的响应速度比中断快。I/O设备发出中断请求后，MPU要执行完当前指令后才给予响应并且要保护现场，而DMA请求是在总线周期执行完后即可响应。
    - (3) 对于快速的I/O设备，中断方式的传输速度已无法满足要求，必须采用DMA方式完成快速I/O设备的数据传送操作

125

1 中断概述(续)

- 中断是指计算机在执行某一程序的过程中，由于计算机系统内、外的某种原因，而必须中止原程序的执行，转去执行相应的处理程序，待处理结束之后，再返回继续 执行被中止的原程序的过程。
- 采用中断技术的计算机，可以解决CPU与外设之间速度 匹配的问题，使计算机可以及时处理系统中许多随机的参数和信息，同时，它也提高了计算机处理故障与应变 的能力。
- “中断”与“查询”相比：
  - 执行效率↑
  - 实时性 ↑

126

1 中断概述(续)

中断的优点

a .中断可以解决快速的 CPU与慢速的外设之间的矛盾,使CPU 和外设同时工作.CPU在启动外设工作后继续执行主程序,同时外设也在工作。每当外设做完一件事就发出中断申请,请求 CPU 中断它正在执行的程序，转去执行中断服务程序（一般情况是处理输入/输出数据），中断处理完之后，CPU恢复执行主程序，外设也继续工作。这样，CPU可启动多个外设同 时工作，大大地提高了 CPU 的效率。

b.在实时控制中，现场的各种参数、信息均随时间和现场而变 化。这些外界变量可根据要求随时向 CPU 发出中断申请,请求 CPU及时处理中断请求。如中断条件满足，CPU 马上就会响 应，进行相应的处理，从而实现实时处理。

c .针对难以预料的情况或故障，如掉电、存储出错、运算溢出等，可通过中断系统由故障源向CPU发出中断请求，再由CPU 转到相应的故障处理程序进行处理。

127

1 中断概述(续)

中断向量

当微处理器响应中断后，要求中断源提供一个地址信息， 该地址信息称为中断向量（或中断矢量），微处理器根据 这个中断向量转移到该中断源的中断服务程序处执行。

根据形成中断服务程序入口地址机制的不同,向量中断 可分成固定中断和可变中断。

固定中断向量

各个中断源的中断服务入口地址是固定不变的，由微处理器设计 时已经 确定，系统设计者不能改变。ARM920T核采用此种。

固定中断向量的中断源识别机制主要是通过硬件电路实现，典型 的是采用“菊花链”(daisy chain)逻辑电路。

可变中断向量

中断服务程序的入口地址不是固定不变的，系统设计者可以根据自己 的需要进行设置。优点：设计比较灵活，用户可根据需要设定中断向量 表在主存中的位置 。缺点：中断响应速度较慢。

128

1 中断概述(续)

中断响应的一般过程

(1) 在每条指令结束后，系统都自动检测中断请求信号，如果有中断请求，且CPU处于开中断状态下，则响应中断。

(2) 保护现场，在保护现场前，一般要关中断，以防止现场被破坏。保护现场是用堆栈指令将原程序中用到的寄存器推入堆栈。

(3) 中断服务，即为相应的中断源服务。

(4) 恢复现场，用堆栈指令将保护在堆栈中的数据弹出来，在恢复现场前要关中断，以防止现场被破坏。在恢复现场后应及时开中断。

(5) 返回，此时 CPU将压入到堆栈的断点地址弹回到程序计数器，从而使CPU继续执行刚才被中断的程序。

129

1 中断概述(续)

- 中断方式控制的I/O操作步骤
  - 初始化微处理器中用于中断方式的寄存器，开放中断。
  - I/O端口或部件完成数据操作后并产生中断请求信号。
  - 当中断请求信号有效时，微处理器可能处在不可中断状态。等到微处理器允许中断时，微处理器就保存当前状态，停止它现行的操作并开始进行中断源的识别。
  - 在识别出优先级最高的中断源后,微处理器转到对应的中断服务程序入口,并应答中断，I/O端口或部件收到应答信号后,撤消其中断请求。
  - 微处理器读入或写出数据，当中断服务程序结束后，回到原来被中断程序处继续执行。

130

131

2 S3C2410中断系统

S3C2410中断控制器有56个中断源，对外提供 24个外中断输入引脚，内部所有设备都有中断请求 信号，例如DMA控制器、UART、IIC等等。

S3C2410的ARM920T内核有两个中断，IRQ中断 和快速中断FIQ。

中断仲裁：当中断控制器接收到多个中断请求 时，其内部的优先级仲裁器裁决后向CPU发出优先 级最高的中断请求信号或快速中断请求信号。

132

2 S3C2410中断系统(续)

S3C2410主要由中断源和控制寄存器两大部分构成， 其寄存器主要有4种：模式、屏蔽、优先级、挂起(标志)

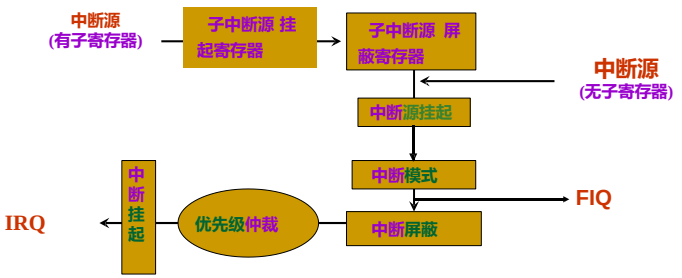


图 S3C2410中断处理流程

133

2 S3C2410中断系统(续)

ARM系统的中断处理

在 ARM系统中，支持复位、未定义指令、软中 断、指令预取中止、数据中止、IRQ和FIQ 7种 异常，每种异常对应于不同的处理器模式，有 对应的异常向量（固定的存储器地址）。

在 ARM系统中，一旦有中断发生，正在执行的程序都会停下来，通常都会执行如下的中断步骤：

(1) 保存现场。保存当前的PC值到R14，保存 当前的程序运行状态到SPSR。

(2) 模式切换。根据发生的中断类型，进入IRQ 模式或FIQ模式。

134

2 S3C2410中断系统(续)

(3) 获取中断服务程序的地址。PC指针跳到异常向量 表所保存的IRQ或FIQ地址处，IRQ或FIQ的异常向量 地址处一般保存的是中断服务子程序的地址，PC指针 跳入到中断服务子程序，进行中断处理。

(4) 多个中断请求处理。在ARM系统中，可以存在多个 中断请求源，比如串口中断、AD中断、外部中断 、定 时器中断及DMA中断等，所以可能出现多个中断源同 时请求中断的情况。为了更好地区分各个中断源，通常 为这些中断定义不同的优先级别，并为每一个中断设置 一个中断标志位。当发生中断时，通过判断中断优先级 以及访问中断标志位的状态来识别哪一个中断发生了， 进而调用相应的函数进行中断处理。

(5) 中断返回，恢复现场。当完成中断服务程序后，将 SPSR中保存的程序运行状态恢复到CPSR中，R14中 保存的被中断程序的地址恢复到PC中，继续执行被中 断的程序。

135

2 S3C2410中断系统(续)

S3C2410的中断控制器

S3C2410采用ARM920T CPU内核，ARM920T CPU的中断包含有IRQ和FIQ. IRQ是普通中断，FIQ是快速中断，FIQ的优先级高于IRQ。FIQ中断通常在进行大批量的复制、数据传输等工作时使用。

S3C2410通过对程序状态寄存器(PSR)中的F位和I位进行设置控制CPU的中断响应。如果设置 PSR的F位为1，则CPU不会响应来自中断控制 器的FIQ中断；如果设置PSR的I位为1，则CPU 不会响应来自中断控制器的IRQ中断。如果把 PSR的F位或I位设置为0，同时将中断屏蔽寄存 器（INTMSK）中的相对应位设置为0，CPU响 应来自中断控制器的IRQ或FIQ中断请求。

136

2 S3C2410中断系统(续)

中断屏蔽寄存器用于指示中断是否禁止。如果 设置中断屏蔽寄存器中的相对应屏蔽位为1，表 示相对应的中断禁止；如果设置为0，表示中断 发生时将正常执行中断服务。

S3C2410有SRCPND(中断源挂起寄存器)和INTPND(中断挂起寄存器)两个中断挂起寄存器.

SRCPND和INTPND两个挂起寄存器用于指示某个中断请求是否处于挂起状态。当多个中断源请求中断服务时，SRCPND寄存器中的相应位 设置为1，优先级仲裁过程结束后INTPND寄存 器中只有1位被自动设置为1。

137

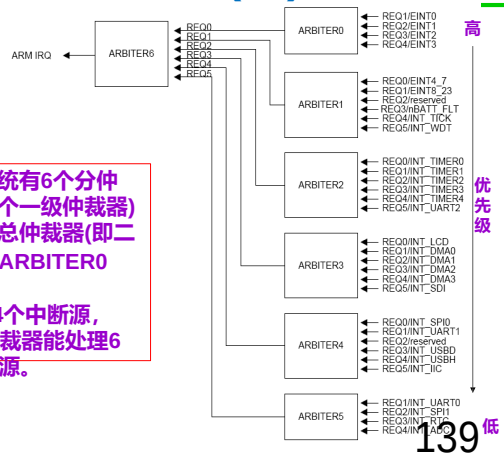
2 S3C2410中断系统(续)

S3C2410中的中断控制器能够接收来自56个中断源的请求，这些中断源来自片内外部中断如DMA控制器、UART、I2C等及外部中断引脚两大类。

S3C2410共有32个中断请求信号。 S3C2410A采用了中断共享技术，INT_UARTO、INT_UART1、INT_UART2、EINT8_23和EINT4_7为多个中断源共享使用的中断请求信号。 中断请求的优先级逻辑是由7个仲裁器组成的，其中包括6个一级仲裁器和1个二级仲裁器。每个仲裁器是否使能由寄存器PRIORITY[6:0]决定。每个仲裁器可以处理4~6个中断源，从中选出优先级最高的。优先级顺序由寄存器RIORITY[20:7]的相应位决定。

138

2 S3C2410中断系统(续)



中断系统有6个分仲裁 器(6个一级仲裁器)和1 个总仲裁器(即二级仲裁器)；ARBITER0 、5 能处理4个中断源，其 余仲裁器能处理6个中 断源。

139

内容提要

140

3 IRQ和FIQ异常中断处理(续)

■FIQ异常中断为快速异常中断，它比IRQ异常中断优先级高，这主要表现在下面两个方面：

- 1) 当FIQ和IRQ异常中断同时发生时，CPU优先处理FIQ异常中断；
- 2) 在FIQ异常中断处理程序中IRQ异常中断被禁止。

■由于FIQ异常中断通常用于处理系统中对于响应时间要求比较苛刻的任务，ARM体系结构在设计上有一些特别的安排，以尽量减少FIQ异常中断的响应时间

◆FIQ异常中断的中断向量为0x1C，位于中断向量表的最后。这样FIQ异常中断处理程序可以直接放在地址0x1C开始的存储单元，这种安排省掉了中断向量表中的跳转指令，从而也就节省了中断响应的时间。

◆当系统中存在Cache时，可以把FIQ异常中断向量以及处理程序一起锁定在Cache 中，从而大大地提高FIQ异常中断的响应时间。

◆除此之外，与其他异常模式相比，FIQ异常模式还有额外的5个物理寄存器(R8_fiq~R12_fiq)，这样在进入FIQ中断处理程序时可以不用保存这5个寄存器，从而也提高了FIQ异常中断的处理速度。

141

3 IRQ和FIQ异常中断处理(续)

多数情况下是用软件来处理异常分支，如下图所示。

软件可以通过读取中断控制器来获得中断源的详细信息。

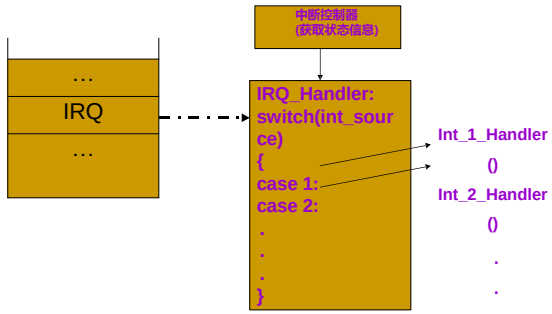
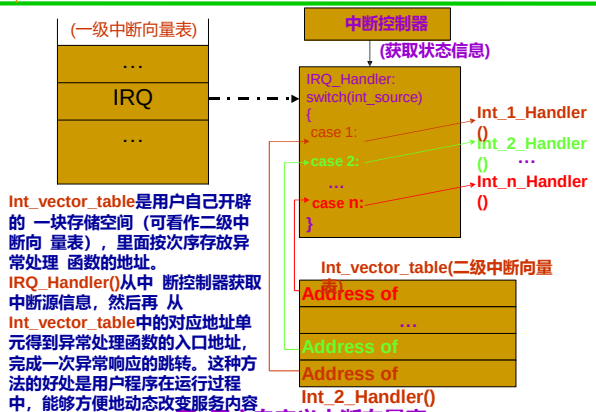


图 软件控制中断分支

142

由于软件设计的灵活性，用户还可以设计出比前面图中更好的流程控制方 法。如下图所示，用户能自行定义中断向量表来实现中断处理程序。



Int_vector_table是用户自己开辟的 一块存储空间（可看作二级中断向 量表），里面按次序存放异常处理 函数的地址。

IRQ_Handler()从中 断控制器获取中断源信息，然后再 从Int_vector_table中的对应地址单元得到异常处理函数的入口地址，完成一次异常响应的跳转。这种方法的好处是用户程序在运行过程中，能够方便地动态改变服务内容。

图 用户自定义中断向量表

143

ARM DMA

144

1 DMA 概述

DMA技术是一种高速的数据传输方式，允许在外部设备 和存储器之间、存储器与存储器之间等直接传输数据

DMA方式传输特点

中断方式下，CPU需要执行多条指令，占用一定的时间；而DMA 传送1个字节只占用CPU的1个总线周期。

DMA的响应速度比中断快。I / O设备发出中断请求后，CPU要执 行完当前指令后才给予响应并且要保护现场，而DMA请求是在总 线周期执行完后即可响应。

对于快速的I / O设备，中断方式，其传输速度已无法满足要求。 必须采用DMA方式来完成快速I / O设备的数据传送的操作。

145

1 DMA 概述(续)

- HOLD和HLDA用于DMA方式请求和响应
- DMAC（DMA控制器）是DMA传送的核心电路。
- DMA传送过程一般分为如下四个阶段:

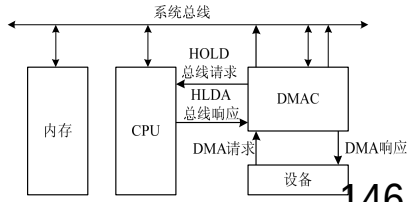
  申 请 阶 段 ;

  响 应 阶 段 ;

  数 据 传 送 阶 段 ;

  传 送 结 束 阶 段 。

DMA传送过程



146

147

2 S3C2410 DMA

S3C2410芯片的DMA系统拥有4个独立通道的DMA控制器，每个通道的DMA控制器都可以控制处理芯片内部与内部之间、芯片内部与外部之间、芯片外部与外部之 间的数据传输。也就是说，每一个DMA通道都可以处理以下4种情况的DMA操作:

源设备和目的设备都在内部系统总线上。

源设备在内部系统总线上，目的设备在外部总线上。

源设备在外部总线上，目的设备在内部系统总线上。

源设备和目的设备都在外部总线上。

148

2 S3C2410 DMA (Cont.)

DMA请求源

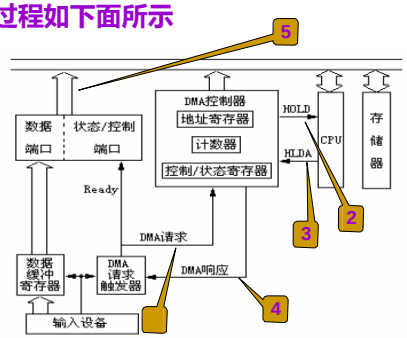
在H/W请求模式（硬件请求模式）下，S3C2410芯片中的4个DMA通道的每一个通道都可以从5个DMA源中选择一个DMA请求 源。但在S/W请求模式（软件请求模式）下，DMA请求源就没有 任何意义。

	请求源0	请求源1	请求源2	请求源3	请求源4
通道0	nXDREQ0	UART0	SDI	Timer	USB设备EP1
通道1	nXDREQ1	UART1	I2SSDI	SPI0	USB设备EP2
通道2	I2SSDO	I2SSDI	SDI	Timer	USB设备EP3
通道3	UART2	SDI	SPI1	Timer	USB设备EP4

149

2 S3C2410 DMA (Cont.)

一般DMA的工作过程如下面所示



(1) 外设向DMAC 发出请求

(2) DMAC通过HOLD 向CPU 发出总 线请求;

(3) CPU响应释放三总线，并且发应答 HLDA

(4) DMAC向外设发DMA应答

(5) DMAC发出地址、控制信号，为外设传送数据;

(6) 传送完规定的数据后，DMAC撤销HOLD信号，CPU也撤销HLDA信号，并 且恢复对三总线的控制。

150

2 S3C2410 DMA (Cont.)

DMA的工作过程

S3C2410X的DMA工作过程可以分为三个状态：

状态1：等待状态。DMA 等待一个DMA请求。如果有请 求到来，将转到状态2。在这个状态下，DMA ACK和INT REQ为0。

状态2：准备状态。DMA ACK变为1，计数器（CURR_TC）装入DCON[19:0]寄存器。

注意：DMA ACK保持为1直至它被清除。

状态3：传输状态。DMA控制器从源地址读入数据并将 它写到目的地址，每传输一次，CURR_TC数器（在DSTAT 中）减1，并且可能做以下操作：

重复传输：在全服务模式下，将重复传输，直到计数器 CURR_TC变为0；在单服务模式下，仅传输一次。

151

2 S3C2410 DMA (Cont.)

设置中断请求信号：当CURR_TC变为0时，DMAC发出INT REQ信号，而且DCON[29]即中断设定位被设为1。

清除DMA ACK信号：对单服务模式，或者全服务模式CURR_TC变为0。

注意：在单服务模式下，DMAC的3个状态被执行一遍，然后停止，等待下一个DMA REQ的到来。如果DMA REQ到来，则这些状态被重复操作， 直到CURR_TC减为0 。

说明：DMA传输分为一个单元传输和4个单元突发式传输。

152

2 S3C2410 DMA (Cont.)

外部DMA请求/响应规则

DMAC有3种类型的外部DMA请求/响应规则：

(1) single service demand，**单服务请求**（对应于需求模式）

(2) single service handshake，**单服务握手**（握手模式）

(3) whole service handshake，**全服务握手**（全服务模式） 每种类型都定义了像DMA请求和DMA响应这些信号怎样与这些规则相联系。

demand 与 handshake模式的比较：

在一次传输结束时，DMA检查xnxDREQ（DMA请求）信号的 状态：

在demand模式下：如果DMA请求（xnxDREQ）信号仍然有效，则传输马上再次开始。否则等待。

在handshake模式下：如果DMA请求信号无效，DMA在两个时钟周期后将DMA响应（xnxDACK）信号变得无效。否则，DMA 等待直到DMA请求信号变得无效。每请求一次传输一次。

153