

# 第二章

## 处理器工作模式

### 2.1 概述

S3C2440 采用了非常先进的 ARM920T 内核，它是由 ARM(Advanced RISC Machines) 公司研制的。

### 2.2 处理工作状态

从程序员的角度上看，ARM920T 可以工作在下面两种工作状态下的一种：

- ARM 状态：执行 32 位字对齐的 ARM 指令
- THUMB 状态：执行 16 位半字对齐的 THUMB 指令。在这种状态下，PC 寄存器的第一位来选择一个字中的哪个半字

注意：这两种状态的转换不影响处理模式和寄存器的内容。

### 2.3 切换状态

#### 进入 THUMB 状态

进入 THUMB 状态，可以通过执行 BX 指令，同时将操作数寄存器的状态位（0 位）置 1 来实现。

当从异常（IRQ, FIQ, UNDEF, ABORT, SWI 等）返回时，只要进入异常处理前处理器处于 THUMB 状态，也会自动进入 THUMB 状态。

#### 进入 ARM 状态

进入 ARM 状态，可以通过执行 BX 指令，并且操作数寄存器的状态位（0 位）清零来实现。

当处理进入异常（IRQ, FIQ, RESET, UNDEF, ABORT, SWI 等）。这时，PC 值保持在异常模式下的 link 寄存器中，并从异常向量地址处开始执行处理程序。

#### 存储空间的格式

ARM920T 将存储器空间视为从 0 开始由字节组成的线性集合，字节 0 到 3 中保存了第一个字节，字节 4 到 7 中保存第二个字，以此类推，ARM920T 对存储的字，可以按照小端或大端的方式对待。

### 大端格式:

在这种格式中, 字数据的高字节存储在低地址中, 而字数据的低字节则存放在高地址中, 因此字节 0 存储在数据的 24 到 31 行里

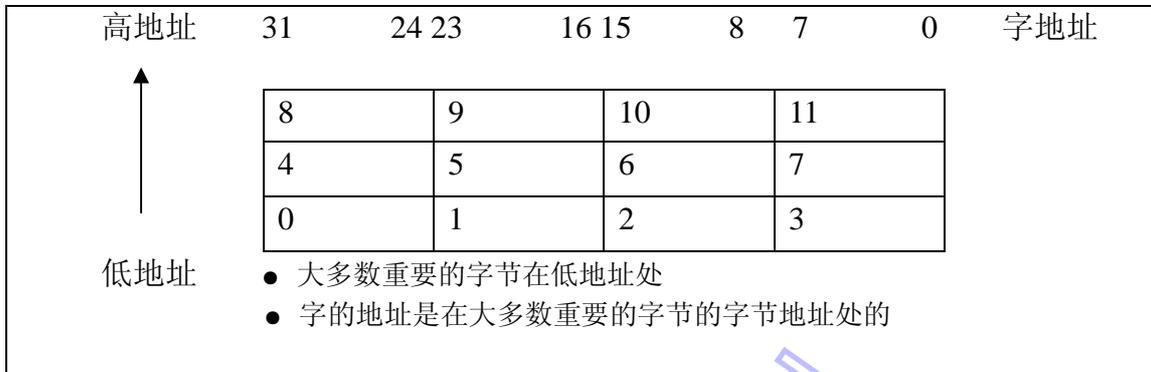


图 2-1 以大端格式存储字数据

### 小端格式:

与大端格式相反, 在小端存储格式中, 低地址中存放的是字数据的低字节, 高地址存放的是字数据的高字节。

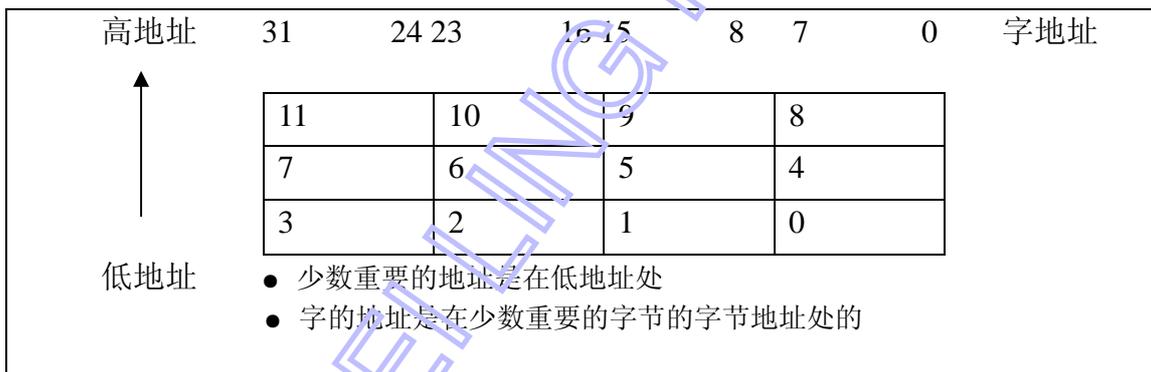


图 2-2 以小端格式存储字数据

## 2.4 指令长度

指令可以是 32 位长度(在 ARM 状态下) 或 16 位长度 (在 THUMB 状态)。

### 数据类型

ARM920T 支持字节(8 位), 半字(16 位) 和字(32 位) 数据类型。字必须按照 4 字节对齐, 半字必须是 2 字节对齐。

## 2.5 操作模式

ARM920T 支持 7 种操作模式:

- 用户模式(user 模式), 运行应用的普通模式

- 快速中断模式(fiq 模式), 用于支持数据传输或通道处理
- 中断模式(irq 模式), 用于普通中断处理
- 超级用户模式(svc 模式), 操作系统的保护模式
- 异常中断模式(abt 模式), 输入数据后登入或预取异常中断指令
- 系统模式(sys 模式), 使操作系统使用的一个有特权的用户模式
- 未定义模式(und 模式), 执行了未定义指令时进入该模式]

外部中断, 异常操作或软件控制都可以改变中断模式。大多数应用程序都是在用户模式下进行, 进入特权模式是为了处理中断或异常请求或操作保护资源服务的。

## 2.6 寄存器

ARM 共有 37 个 32 位的寄存器, 其中 31 个是通用寄存器, 6 个是状态寄存器。但在同一时间, 对程序员来说并不是多有的寄存器都可见。在某一时刻存储器是否可见, 是由处理器当前的工作状态和工作模式决定的。

### ARM 状态寄存器

在ARM状态下, 任何时刻都可以看到16个通用寄存器, 1个或2个状态寄存器。在特权模式(非用户模式)下会切换到具体模式下的寄存器组, 其中包括模式专用的私有(banked)寄存器。图2-3显示了在每个模式下哪种寄存器是可见的; 私有寄存器上都有一个黑三角标记。

ARM状态寄存器系列中含有16个直接操作寄存器: R0到R15。除了R15外其他的都是通用寄存器, 用来存放地址或数据值。除此之外, 实际上有17个寄存器用来存放状态信息。

寄存器14	专职持有返回点的地址, 在系统执行一条“跳转并链接”(BL)指令的时候, R14将收到一个R15的拷贝。其他时候, 它可以用作一个通用寄存器。相应的私有寄存器R14_svc, R14_irq, R14_fiq, R14_abt和R14_und都同样用来保存在中断或异常发生时, 或在中断和异常时执行了BL指令时, R15的返回值。
寄存器15	这个寄存器是程序计数器(PC)。在ARM状态下, R15的bits[1:0]为0, bits[31:2]保存了PC的值。在Thumb 状态下, bits[0]为0同时bits[31:1]保存了PC值。
寄存器16	这个寄存器是CPSR(当前程序状态寄存器), 用来保存当前代码标志和当前模式位。

FIQ模式拥有7个私有寄存器R8-14(R8\_fiq-R14\_fiq)。在ARM状态下, 多数FIQ处理都不需要保存任何寄存器。用户、中断、异常中止, 超级用户和未定义模式都拥有2个私有寄存器, R13和R14。允许这些模式都可拥有1个私有堆栈指针和

链接寄存器。

**ARM 状态下寄存器集**

系统和用户	FIQ	超级用户	异常中止	IRQ	未定义
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)					



图 2-3 ARM 状态下的寄存器结构

## THUMB 状态寄存器

THUMB 状态寄存器是 ARM 状态寄存器的一个子集。程序员可以直接操作 8 个通用寄存器 R0-R7，同样可以这样操作程序计数器(PC)，堆栈指针寄存器(SP)，链接寄存器(LR)，和 CPSR。它们都是各个特权模式下的私有寄存器，链接寄存器和程序状态寄存器(SPSRs)。如图 2-4



图 2-4Thumb 状态下寄存器结构

ARM 和 THUMB 状态寄存器间的关系:

- THUMB 状态下 R0-R7 和 ARM 状态下 R0-R7 是等同的
- THUMB 状态下 CPSRs 和 SPSRs 跟 ARM 状态的 CPSR 和 SPSRs 是等同的
- THUMB 状态下的 SP 映射在 ARM 状态下得 R13 上
- THUMB 状态下的 LR 映射在 ARM 状态下得 R14 上
- THUMB 状态下程序计数器映射在 ARM 状态下的程序计数器上(R15)

图 2-5 显示了它们的关系:

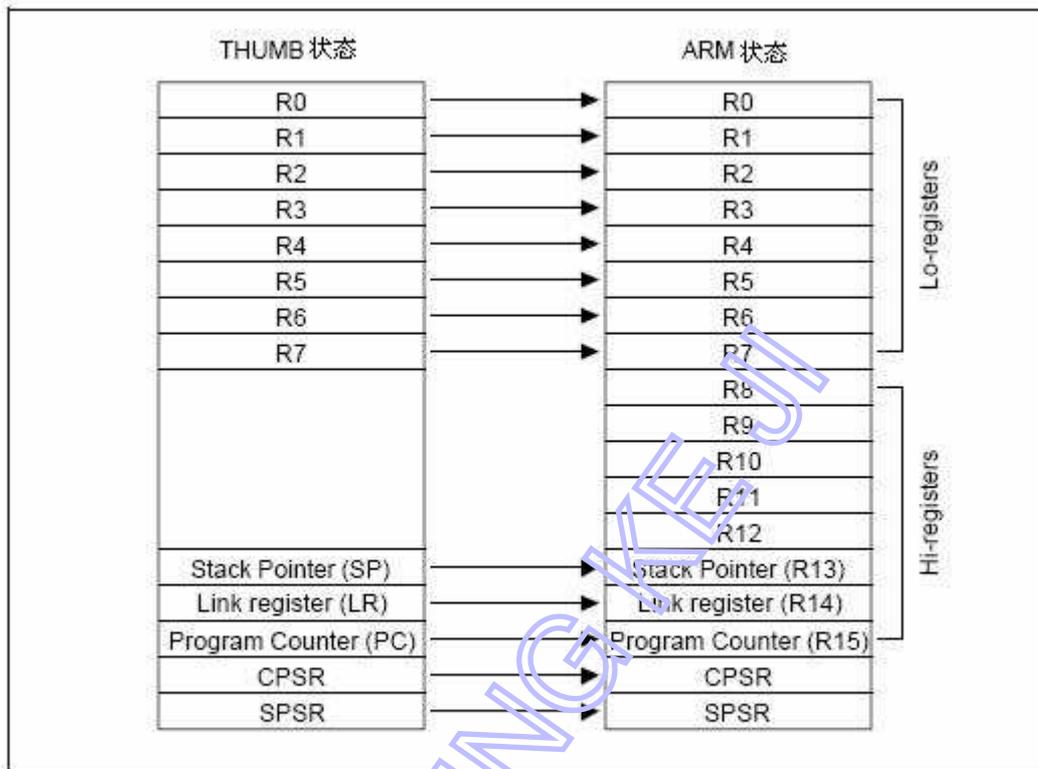


图2-5 THUMB状态下和ARM状态下寄存器之间的映射关系

## 在 THUMB 状态下访问高地址寄存器

在 THUMB 状态下寄存器 R8-15(高地址寄存器)不是标准寄存器集。但是，汇编语言的程序员可以访问它们并用它们作快速暂存。

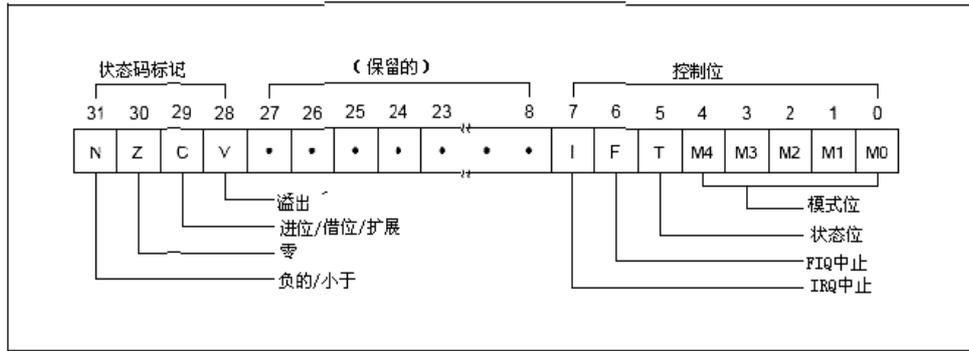
采用 MOV 指令的某个变型，从 R0-R7（低地址寄存器）的某个寄存器传送数据到达高地址寄存器，或者从高地址寄存器传送到低地址寄存器。还可以采用 CMP 和 ADD 指令，将高地址寄存器的值与低地址寄存器的值进行比较或相加。想获得这方面更多的信息，请参考图 3-34。

## 2.7 程序寄存器状态

ARM920T 具有一个当前程序状态寄存器 (CPSR)，另外还有 5 个保存程序状态寄存器 (SPSRs) 用于异常中断处理。这些寄存器的功能有：

- 保留最近完成的 ALU 操作的信息
- 控制中断的使能和禁止
- 设置处理器的操作模式

下图 2-6 显示了程序状态寄存器的位定义：



### 2.7.1 条件码标志

N、Z、C、V 均为条件码标志位。它们的内容根据算术或逻辑运算的结果所改变，并且来作为一些指令是否运行的检测条件。

在 ARM 状态下，绝大多数指令都是有条件执行的，详情请看表 3-2；

在 Thumb 状态下，仅有分支指令是有条件执行的，详情请看图 3-46。

### 2.7.2 控制位

PSR 的低 8 位(包括 I、F、T、M[4:0])称为控制位，当发生异常时这些位将被改变。如果处理器工作在特权模式下，这些位也可以由软件操纵。

T 标记位	该位反映处理器的运行状态。该位被设置为 1 时，处理器执行在 THUMB 状态，否则执行在 ARM 状态，这些由外部信号 TBIT 反映出来。注意：软件决不能改变 CPSR 的 TBIT 状态。如果这样做，处理器将会进入一种不可预知的状态。
中断禁止位	I、F 位为中断禁止位，当它们被设置成 1 时可以相应的禁止 IRQ 和 FIQ 中断。
模式位	M4, M3, M2, M1 和 M0 位(M[4:0])是模式位，它们决定了处理器的操作模式，如下表 2-7 所示。并不是所有的模块组合位都决定一个有效的处理器模式，只有那些明确地描述值才能被使用，用户必须意识到任何一种非法的值写入模块位，处理器都会进入到一种不可重获的状态，如果这种情况发生，就要进行复位。
保留位	PSR 中的其余位为保留位，当改变 PSR 中的位或者控制位时，必须确保保留位不被改变，在程序中也不要使用保留位来存储数据值。因此在将来的处理器中它们也许作为 1 或 0 来读。

图2-7 PSR模式位的值

M[4:0]	模式	可视的THUMB状态寄存器	可视的ARM状态寄存器
10000	用户模式	R7..R0, LR, SP, PC, CPSR	R14..R0, PC, CPSR
10001	FIQ 模式	R7..R0, LR_fiq, SP_fiq, PC, CPSR, SPSR_fiq	R7..R0, R14_fiq..R8_fiq, PC, CPSR, SPSR_fiq
10010	IRQ 模式	R7..R0, LR_irq, SP_irq, PC, CPSR, SPSR_irq	R12..R0, R14_irq, R13_irq, PC, CPSR, SPSR_irq
10011	超级用户模式	R7..R0, LR_svc, SP_svc, PC, CPSR, SPSR_svc	R12..R0, R14_svc, R13_svc, PC, CPSR, SPSR_svc
10111	中止	R7..R0, LR_abt, SP_abt, PC, CPSR, SPSR_abt	R12..R0, R14_abt, R13_abt, PC, CPSR, SPSR_abt
11011	未定义模式	R7..R0, LR_und, SP_und, PC, CPSR, SPSR_und	R12..R0, R14_und, R13_und, PC, CPSR
11111	系统模式	R7..R0, LR, SP, PC, CPSR	R14..R0, PC, CPSR

保留位 PSR 中的其余位为保留位，当改变 PSR 中的位或者控制位时，必须确保保留位不被改变，在程序中也不要使用保留位来存储数据值。因此在将来的处理器中它们也许作为 1 或 0 来读。

## 2.8 异常

当正常的程序执行流程被临时中断时，称为产生了异常。例如程序执行转向一个外设的中断请求。在异常能被处理前，当前处理器的状态必须被保留，这样按处理程序完成时就能恢复原始的程序。

有可能同时产生好几个异常，如果出现这种情况，就应该按固定的顺序处理。详情请看后面对异常优先级的说明。

### 2.8.1 进入异常时的行为

当一个异常发生时，ARM920T 将进行以下步骤：

1. 将下一条指令的地址保存到相应的 Link 寄存器中。如果异常是从 ARM

状态进入的，下一条指令的地址拷贝到 Link 寄存器（根据异常的类型，数值为当前 PC+4 或 PC+8，具体请看表 2-8）。如果异常是从 THUMB 态进入，那么写入到 link 寄存器的值是当前的 PC 偏移一个值。这表示异常处理程序不需要关心是从哪种状态进入异常的。例如，在 SWI 情况下，无论是来自 ARM 或 THUMB 状态，处理程序只要采用 MOVSPC, R14-svc 语句，总可以返回到原始程序的下一条语句。

2. 拷贝 CPSR 到相应的 SPSR；
3. 根据异常类型强制改变 CPRS 模式位的值；
4. 令 PC 的值指向异常处理向量所指的下一条指令。

这时也可能设置中断禁止标志，以防止不可估计的异常嵌套发生。

当处理器处于 Thumb 状态时发生了异常，当 PC 载入异常矢量所在地址时，它将自动的切换到 ARM 状态。

## 2.82 离开异常处理时的行为

当完成异常处理时：

1. 将 Link 寄存器，减去相应的偏移量，赋给 PC（偏移量的值由异常的类型决定）；
2. 拷贝回 SPSR 到 CPSR；
3. 如果在进入中断时设置了中断禁止标志，清除它。

注意：你不需要特别指明切换回 THUMB 状态。因为原来的 CPSR 被自动的保存到了 SPSR。

## 2.83 异常进入/退出的总结

表 2-8 总结了在进入异常时，保留到相应的 R14 中的 PC 的值，和推荐使用的退出异常处理时采用的语句。

	返回指令	进入异常时R14的值		Notes
		ARM R14_x	THUMB R14_x	
BL	MOV PC, R14	PC + 4	PC + 2	1
SWI	MOVS PC, R14_svc	PC + 4	PC + 2	1
UDEF	MOVS PC, R14_und	PC + 4	PC + 2	1
FIQ	SUBS PC, R14_fiq, #4	PC + 4	PC + 4	2
IRQ	SUBS PC, R14_irq, #4	PC + 4	PC + 4	2
PABT	SUBS PC, R14_abt, #4	PC + 4	PC + 4	1
DABT	SUBS PC, R14_abt, #8	PC + 8	PC + 8	3
RESET	NA	-	-	4

注意:

- 1.这里 PC 所赋的是 BL/SWI/未定义模式等指令所取的地址,它们在预取的阶段就被中断了。
- 2.这里 PC 所赋的是由于 FIQ 或 IRQ 取得了优先权,而没有来得及得到执行的指令地址。
- 3.这里 PC 所赋的地址是 Load 或 Store 指令的地址,它们在执行时产生了数据的异常中断。
- 4.在 R14\_svc 复位之前保存的数值是不可预知的。

#### 2.8.4 FIQ 中断

FIQ (快速中断请求) 异常通常是用来支持数据传输和通道操作的,在 ARM 状态下,它具有充分的私有寄存器,用来减少寄存器存取的需要(从而减少进入中断前的“上下文切换”的工作)。

FIQ 中断是由外部设备通过拉低 nFIQ 引脚触发的。通过对 ISYNC 输入引脚的控制 nFIQ 可以区别同步或异步的传输情况,当 ISYNC 为低电平 nFIQ 和 nIRQ 将被认为是异步的,中断之前产生同步周期延长的话会影响处理器的流程。

不管是 ARM 还是 Thumb 状态下的异常,FIQ 处理程序都可以通过执行以下的语句来退出中断处理:

```
SUBS    PC, R14-fiq, #4
```

通过设置 CPSR 的 F 标记位可以禁止 FIQ 中断(但是要注意到在用户模式下是不可行的)。如果 F 标记位已经清除,ARM920T 在每个指令的最后检测来自 FIQ 中断同步器的低电输出。

#### 2.8.5 IRQ 中断

IRQ (中断请求) 异常是由 nIRQ 输入低电平引发的普通中断。IRQ 中断相对 FIQ 中断来说是优先级低,当一个 FIQ 中断序列进入时它将被屏蔽。IRQ 也可以通过设置 CPRS 中的“I”标志来禁止,只能在特权(非用户)模式下这样做。

无论 IRQ 发生在 ARM 或者 Thumb 状态下,都可以采用以下语句来退出中断处理:

```
SUBS    PC, R14-fiq, #4
```

#### 2.8.6 Abort 异常中止

异常中止表示当前存储访问不能完成。通过外部的 ABORT 输入信号来告知内核。ARM920T 在每次的存储操作中检测该异常是否发生。

有两种类型的异常中止:

- 预取指异常中断: 指令预取时产生

- 数据异常中断；数据访问时产生

如果产生预取指中止，所取得的指令将会被标志为无效的，但是异常不会立即发生，要直到取指到达了管道的头部才会发生。/如果这些指令不执行-例如在管道内发生了分支跳转，那么异常就不会发生了。

如果产生数据异常中止，根据指令类型进行操作：

- 简单数据传输指令 (LDM, STR) 写回改变的基址寄存器：异常中断处理器必须清楚这些。
- 取消交换指令尽管它还没执行
- 数据块传输指令 (LDM, STM) 完成。如果设置为写回，基址已经矫正。如果指令超出了数据的写基址（传输目录中有它的基址），就应该防止写超出。在中止异常将会发生时，所有寄存器的覆盖写入都是禁止的。这意味着特别是 R15（经常是最后一个改变的寄存器）的值将在中止的 LDM 指令中保留下来。

Abort 机制使得页面虚拟存储器机制得以实现。在采用虚拟存储器的系统中，处理器可以产生任意的地址。当某个地址的数据无效，MMU（存储器管理单元）将产生一个 abort 中止。这样 abort 的处理程序也就不需要了解实际可用存储空间的大小，也不需要了解异常中断对他的影响。

在完成 了异常中断的处理后，通过以下语句推出中断处理（与 ARM 状态还是 Thumb 状态无关）：

```
SUBS PC, R14-abt,#4; 预取指 abort
```

```
SUBS PC, R14-abt,#8; 数据 abort
```

通过执行该语句，就恢复了 PC 和 CPSR，并重试被中断的指令。

## 2.8.7 软件中断

SWI（软件中断指令）用来进入超级用户模式，通常用于请求特殊的超级用户功能。SWI 的处理程序通过执行以下状态（ARM 或 Thumb）的语句，退出异常处理：

```
MOV PC, R14-svc
```

通过执行该语句，就恢复了 PC 和 CPSR，并返回到 SWI 后面的指令上。

注意：

前面提到的 nFIQ, nIRQ, ISYNC, LOCK, BIGEND,和 ABORT 引脚只存在于 ARM920TCPU 的内核里。

## 2.8.8 未定义指令

当 ARM920T 遇到一个它不能执行的指令，它将产生一个未定义指令陷阱。这个机制是软件仿真器用来扩展 Thumb 和 ARM 指令集用的。

在完成对未知指令的处理后，陷阱处理程序应该执行以下的语句退出异常处

理（无论是 ARM 或 Thumb 状态）：

MOVS PC, R14-und

通过执行该语句，恢复了 CPSR，并返回执行未定义指令的下一条指令。

### 2.8.9 异常中断向量

异常中断向量的地址如下图所示：

表格 2-3 异常中断向量

地址	异常中断类型	进入时处理器模式
0x00000000	Reset	Supervisor
0x00000004	Undefined instruction	Undefined
0x00000008	Software interrupt	Supervisor
0x0000000C	Abort (prefetch)	Abort
0x00000010	Abort (data)	Abort
0x00000014	Reserved	Reserved
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

### 2.8.10 异常中断优先级

当多个异常中断同时发生时，处理器根据一个固定的优先级系统来决定处理他们的顺序。

最高优先级：

1. 复位
2. 数据 abort
3. FIQ
4. IRQ
5. 预取指 abort

最低优先级：

6. 未定义指令，软件中断。

并非所有的异常中断都可能同时发生：

未定义指令和软件中断是相互排斥的，因为他们都对应于当前指令的唯一的（非重叠的）解码结果。

如果一个数据 abort 和 FIQ 中断同时发生了，并且此时的 FIQ 中断是使能的，ARM920T 先进入到数据 abort 处理程序，然后立即进入 FIQ 向量。从 FIQ 正常的返回后，数据 abort 的处理程序才恢复执行。将数据 abort 设计为比 FIQ 拥有更高的优先级，可以确保传输错误不能逃避检测。这种情况下进入 FIQ 异常处理时间延长了，这一时间必须考虑到 FIQ 中断最长反映时间的计算中去。

### 2.8.11 中断反应时间

最坏情况下的 FIQ 中断的反映时间，假设它是使能的，包括通过同步器最长请求时间（如果是异步则是  $T_{syncmax}$ ），加上最长指令的完成时间( $T_{ldm}$ ，最长指令是 LDM，它装载了所有的寄存器包括 PC)，加上数据 abort 进入时间 ( $T_{exc}$ )，加上进入 FIQ 所需要的时间 ( $T_{fiq}$ )。在这些时间的最后，ARM920T 会执行位于 0x1C 的指令。

$T_{syncmax}$  是 3 个处理器周期， $T_{ldm}$  是 20 个， $T_{exc}$  是 3 个， $T_{fiq}$  是 2 个周期，因此总共是 28 个处理器周期。在一个连续的 20MHz 的处理器时钟系统里，它的使用时间超过了 1.4 微妙。最长 IRQ 的反应时间计算是类似的，但是必须考虑到更高优先级的 FIQ 中断可以推迟任意长时间进入 IRQ 中断处理。最小的 FIQ 或 IRQ 的反应时间包括通过同步器( $T_{syncmin}$ )的最短时间加上  $T_{fiq}$ ，它是 4 个处理器周期。

### 2.8.12 复位

当 nRESET 信号为低，ARM920T 放弃任何指令的执行，并从增加的地址处取指令。

当 nRESET 信号为高时 ARM920T 进行如下操作：

- 1 将当前的 PC 值和 CPSR 值写入 R14\_svc 和 SPSR\_svc，已保存的 PC 和 CPSR 的值是未知的。
- 2 强制 M[4:0]为 10011(超级用户模式)，将 CPSR 中的“I”和“F”位设为 1，并将 T 位清零。
- 3 强制 PC 从 0x00 地址处取得下一条指令。
- 4 恢复为 ARM 状态并开始执行。