11. ARM926EJ-S 处理器概论

11.1 概论

ARM926EJ-S 处理器是通用微处理器 ARM9 家族中的一员。
ARM926EJ-S 属于 5TEJ 版 ARM 架构，针对的是多任务应用，包括全储存器管理，高性能，小核心尺寸和低功耗都是其重要的特点。

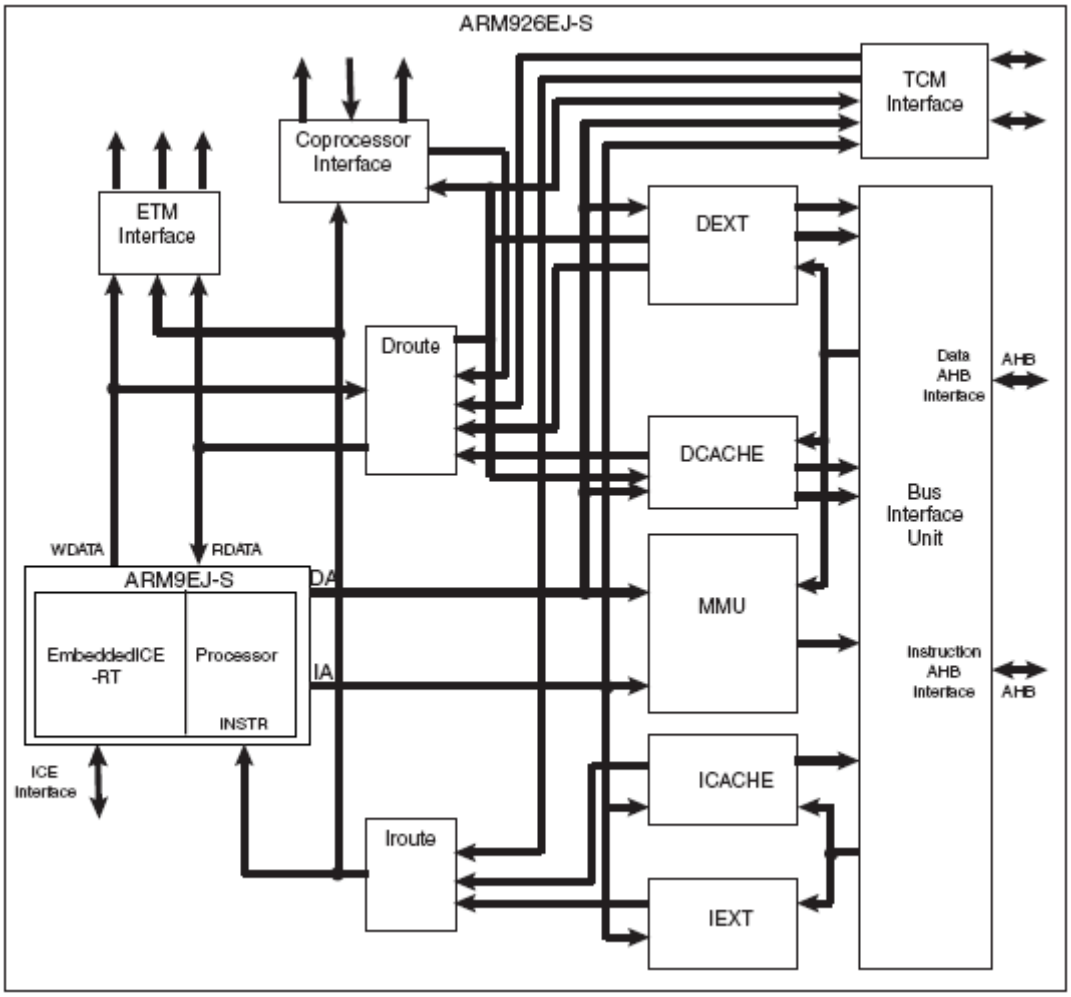
ARM926EJ-S 处理器支持 32 位 ARM 和 16 位 THUMB 指令集，使得用户能在高性能和高代码密度上取得平衡。支持 8 位 Java 指令集并且包括 Java 字节代码有效执行的功能部件，提供和 JIT（Just-In-Time 编译器）相似的 Java 性能，这些性能为下一代 Java 无线和嵌入式的设备提供了有力支持。为了提高 DSP 性能，还包含了一个增强的乘法器设计。

ARM926EJ-S 处理器支持 ARM 调试架构，包括对辅助硬件和软件调试的逻辑。
ARM926EJ-S 提供了一个完全高性能的处理器子系统，包括：

- ARM9EJ-S 整核
- 一个存储器管理部件（MMU）
- 独立的指令和数据 AMBA™ AHB 总线接口
- 独立的指令和数据 TCM 接口

11.2 方块图

图 11-1　　ARM926EJ-S 内部功能方块图



11.3 ARM9EJ-S 处理器

11.3.1 ARM9EJ-S 运行状态

ARM9EJ-S 处理器能用三种不同状态操作，每种状态带一个特殊指令集：

- ARM 状态：32 位，字对齐 ARM 指令
- THUMB 状态：16 位，半字对齐的 THUMB 指令
- Jazelle 状态：可变长度，字节对齐的 Jazelle 指令

在 Jazelle 状态，按字（words）取所有的指令。

11.3.2 状态切换

ARM9EJ-S 核的操作状态在以下状态间切换：

- 用 BX 和 BLX 指令切换 ARM 状态和 THUMB 状态，并加载到 PC
- 用 BXJ 指令切换 ARM 状态和 Jazelle 状态

所有的异常都在 ARM 状态下进入，处理和退出。如果一个异常发生在 Thumb 状态或 Jazelle 状态，处理器还原到 ARM 状态。从异常模式处理程序退出时自动切换回 Thumb 或 Jazelle 状态。

11.3.3 指令流水线

ARM9EJ-S 核使用两种流水线去增加传输至处理器的指令流的速度。

一个五级(五个时钟周期)流水线被用于 ARM 和 Thumb 状态。由取指，译码，执行，存储和回写阶段组成。

一个六级（六个时钟周期）流水线被用于 Jazelle 状态。由取指，Jazelle/译码(两时钟周期)，执行，存储和回写阶段。

11.3.4 存储器访问

ARM9EJ-S 核支持字节(8 位)，半字(16 位)和字(32 位)访问。字必须对齐到四字节边界，半字必须对齐到两字节边界而字节可以被放置在任何字节边界。

因为流水线的性质，一个当前操作需要的值在被放置到寄存器前可能被一个先前的指令操作(使得当前流水线预取的值失效，译者注)。ARM9EJ-S 的控制逻辑自动检测这些情况并停止内核或传输数据。

11.3.5 Jazelle 技术

在 ARM 处理器上，Jazelle 技术直接且有效的执行 Java 字节代码，为下一代 Java 无线应用和嵌入式设备提供高性能。

ARM9EJ-S 新的 Java 特性可以被描述为 JVM(Java 虚拟机)的一个硬件模拟。

Java 模式将表现为另外一种状态：取代执行 ARM 或 Thumb 指令，而执行 Java 字节代码。实现于 ARM9EJ-S 上 Java 字节代码解码器逻辑实现了 95%可执行字节代码的解码，并且在无任何额外开销的情况下把它们转换为 ARM 指令，同时，较小频率使用的字节解码被分解成顺序优化的 ARM 指令。硬件/软件拆分对编程者，应用程序，以及操作系统都是不可见的。所有现存的 ARM 寄存器在 Jazelle 状态被重复使用，并且在此模式下所有的寄存器都有特殊的功能。

最小的中断延迟被维护于 ARM 状态和 Java 状态切换。所以字节代码执行可以被重新开启，以方便中断处理程序执行，一个中断自动触发其内核从 Java 状态到 ARM 状态改变。这意味着当执行字节代码时没有特殊的为控制中断而准备的规定，无论是硬件还是软件。

11.3.6 AEM9EJ-S 操作模式

在所有的状态下，有七种操作模式：

- 用户模式是一般的 ARM 程序执行状态。被用于执行大多数应用程序
- 快速中断（FIQ）模式被用于控制快速中断。适用于高速数据传输或通道进程
- 中断(IRQ)模式被用于一般目的中断的处理
- 管理模式是一种操作系统的保护模式
- 中止(abort)模式在一个数据或指令预取中止后进入
- 系统模式是一个操作系统的特权用户模式
- 未定义模式当一个未定义指令异常发生后进入

模式转换可在软件控制下发生，或可能被外部中断或异常处理所引发。

大多数的应用程序执行是在用户模式下进行的。非用户模式，又称特权模式，用于处理中断或异常，以及访问被保护的资源。

11.3.7 AEM9EJ-S 寄存器

AEM9EJ-S 内核共有 37 个寄存器

- 31 个通用的 32 位寄存器
- 6 个 32 位的状态寄存器

表 11-1　显示了所有模式下所有寄存器

表 11-1　ARM9TDMI 模式和寄存器层

用户和系统模式	管理模式	异常模式	未定义模式	中断模式	快速中断模式
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	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVX	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR _ABORT	SPSR _UNDEF	SPSR _IRQ	SPSR _FIQ

灰色框为特定模式下的备份寄存器

ARM 状态寄存器集包括 16 个直接存取寄存器，r0 到 r15，和一个附加寄存器：当前程序状态寄存器(CPSR)。寄存器 r0 到 r13 是用于保存数据或地址值的通用寄存器。寄存器 r14 被用作一个连接寄存器(Link Register)，当 BL 或 BLX 被执行的时候，用来保存 r15 的值(返回地址)。寄存器 r15 被用作程序计数器(PC)，当前程序状态寄存器(CPSR)包含条件代码标志位和当前模式位。

在特权模式(FIQ，管理模式，中止模式，IRQ，未定义模式)，特定模式备份寄存器(FIQ 模式的 r8 到 r14 或其他模式的 r13 到 r14)可用。当中断或异常发生，或当 BL 或 BLX 指令在中断或异常例程的情况被执行，对应的备份寄存器 r14_FIQ，r14_SVC，r14_ABT，r14_IRQ，r14_UND 相似的被用于保存 r15(PC)的值(为每个模式的返回地址)。还有另外的一个叫保存程序状态寄存器（SPSR）的寄存器，在特权模式下代替 CPSR 变成可用。此寄存器保存进入到当前模式(特权模式)之前模式的代码标志位及模式位。

出于软件一致性，在所有的模式下，r13 被用作堆栈指针。

所有的上面描述的寄存器的功能和用途都应该服从 ARM Procedure Call Standard（APCS），包括：

- 寄存器用途的约束
- 堆栈约定

- 参数传递并返回结果的约定

Thumb 状态寄存器集是一个 ARM 状态集的一个子集。编程者可直接访问：

- 八个通用寄存器 R0-R7
- 堆栈指针,SP
- 连接寄存器，LR(ARM R14)
- PC
- CPSR

每个特权模式都有备份寄存器 SPs，LRs 和 SPSRs（更多的细节见 ARM9EJ-S 技术参考手册，ref.DDI0222B,revision r1p2 2-12 页）。

11.3.7.1 状态寄存器

ARM9EJ-S 内核包括一个 CPSR，和五个异常模式处理程序使用的 SPSRs。程序状态寄存器：

- 保存最近 ALU 操作运行的信息
- 使能和禁用中断
- 设置处理器操作模式

图 11-2 状态寄存器格式

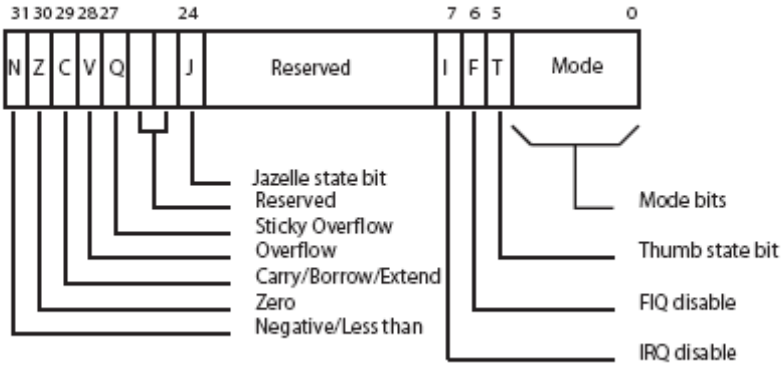


图 11-2 展示了状态寄存器格式：

- 四个 ALU 标志位 N：Negative(负)，Z：Zero（零），C：Carry（进位），V：Overflow (溢出)
- 黏着（sticky）溢出标志位(Q)可以被特定的乘法和小数运算指令像 QADD，QDADD，QSUB，QDSUB，SMLAxy，以及 AMLAWy 置位，完成需要的 DSP 操作。

Q 标志位是黏着的，意味着当其被一指令置位后，此标志位将一直保持置位，除非通过一个 MSR 指令写 CPSR 明确地清零。指令并不能依 Q 标志位的状态按条件执行。

- CPSR 中的 J 位表示 ARM9EJ-S 内核在 Jazelle 状态：
  -J=0：处理器在 ARM 或 Thumb 状态，取决于 T 位
  -J=1：处理器在 Jazelle 状态
- 模式：五位编码用于确定当前处理器模式

11.3.7.2 异常

异常类型和优先级

ARM9EJ-S 支持五种异常。每种类型在一个特权模式下驱动 ARM9EJ-S。

异常的类型有：

- 快速中断（FIQ）
- 一般中断（IRQ）
- 数据和预取指中止（Abort）
- 未定义指令（未定义）
- 软件中断和复位（管理模式）

当一个异常发生，R14 的备份寄存器和异常模式下的 SPSR 用来保存状态。

在同一时间可以发生多于一个的异常。因此 ARM9EJ-S 根据以下的优先级运行引发的异常：

- 复位（最高优先级）
- 数据中止
- FIQ
- IRQ
- 预取指中止
- BKPT，未定义的指令，和软件中断(SWI)(最低优先级)

BKPT，或未定义指令，和 SWI 异常是互斥的。

在优先级序列中有一个例外，当 FIQ 使能，并且发生数据中止异常时一个 FIQ 也同时发生，ARM9EJ-S 内核进入数据中断（Data Aborts）处理程序，然后立即进入 FIQ 向量。一个 FIQ 的一般中断返回导致(先前的)数据中断（Data Aborts）处理继续执行。数据中断（Data Aborts）必须是比 FIQ 有较高优先级以确保传输错误不会漏过检测。

异常模式和控制

无论何时，当异常发生，一个程序的正常流程必须暂时停止，例如，处理一个外设中断。

当处理一个 ARM 的异常时，ARM9EJ-S 内核执行以下操作：

1 保存下一条指令的地址到进入的新模式所对应的 Link 寄存器

当异常从以下状态进入：

- ARM 和 Jazelle 状态，ARM9EJ-S 把下一个指令的地址复制到 LR(根据异常，当前 PC(r15)+4 或 PC+8)
- THUMB 状态，ARM9EJ-S 写 PC 的值到 LR，偏移一个值（根据异常，当前 PC+2,PC+4 或 PC+8），使得程序能从正确地址返回重开始的值。

2 复制 CPSR 到相应的 SPSR

3 根据异常状态强制设置 CPSR 相关模式位

4 强制 PC 从相关的异常向量获取下一个指令

寄存器 r13 也是备份的，通过异常模式向每个异常处理提供专用堆栈指针。

ARM9EJ-S 也可以置位中断禁止标志位，以阻止不可管理的异常嵌套。

当一个异常处理完成，异常处理必须把备份 LR 的值减去一个偏移量的后移到 PC，把 SPSR 复制到 CPSR。偏移量根据异常类型变化。此操作恢复 PC 和 CPSR。

快速中断模式有七个专用寄存器 r8-r14(备份寄存器)，可以减少甚至不需要对寄存器的保存操作，以降低上下文切换的额外消耗。

预取指中止是中止模式之一，表示当前存储器访问不能完成。当一个预取指中止发生，ARM9EJ-S 标记预取指令为无效，但指令在流水线中未到达执行阶段则不执行例外异常模式。如果指令没有执行，例如由于一个分支在流水线中发生，则中止不发生。

断点指令（BKPT）是 ARM9EJ-S 的一个新的特性，被设计用于解决预取指中止的问题。一个断点指令像导致一个预取指中止的指令一样执行。
断点指令不会使 ARM9EJ-S 去执行预取指中止异常模式，直到指令到达流水线的执行阶段。如果指令未执行，例如因为一个分支在流水线中发生，断点不再发生。

11.3.8 ARM 指令集一览

ARM 指令集被分成：

- 分支（Branch）指令
- 数据处理指令
- 状态寄存器传输指令
- 装载和存储指令
- 协处理器指令
- 异常产生（Exception-generating）指令

ARM 指令可以被有条件的执行。每个指令包含一个 4 位条件代码字段(位[31：38])。

表 11-2 给出了 ARM 指令助记符列表。

表 11-2. ARM 指令助记符列表

助记符	操作
MOV	Move 传送
ADD	Add 加
SUB	Subtract 减
RSB	Reverse Subtract 逆减
CMP	Compare 比较
TST	Test 测试
AND	逻辑与
EOR	逻辑异或
MUL	乘
SMULL	带符号长乘
SMLAL	带符号长乘累加
MSR	传送到状态寄存器
B	分支
BX	分支并交换
LDR	装载字
LDRSH	装载带符号半字

LDRSB	装载带符号字节
LDRH	装载半字
LDRB	装载字节
LDRBT	装载带转换的寄存器字节
LDRT	装载带转换的寄存器
LDM	批量装载
SWP	字交换
MCR	传送至协处理器
LDC	装载至协处理器
CDP	协处理器数据处理

助记符	操作
MVN	取反传送
ADC	带进位加
SBC	带借位减
RSC	带借位逆减
CMN	取反比较
TEQ	测试等价
BIC	位清零
ORR	逻辑或
MLA	乘累加
UMULL	无符号长乘
UMULAL	无符号长乘累加
MRS	从状态寄存器传送
BL	分支并链接
SWI	软件中断
STR	存储字
STRH	存储半字
STRB	存储字节
STRBT	存储带转换的寄存器字节
STRT	存储带转换的寄存器
STM	批量存储
SWPB	交换字节
MRC	从协处理器传送
STC	从协处理器存储

11.3.9 新 ARM 指令集

表 11-3 新 ARM 指令助记符列表

助记符	操作
BXJ	分支并切换至 Java

BLX	分支，连接并切换
SMLAxy	带符号乘累加 16*16 位
SMLAL	带符号长型乘累加
SMLAWy	带符号乘累加 32*16 位
SMULxy	带符号乘 16*16 位
SMULWy	带符号乘 32*16 位
QADD	饱和加
QDADD	饱和双精度加
QSUB	饱和减
QDSUB	饱和双精度减

助记符	操作
MRRC	从协处理器传送双字
MCR2	ARM 寄存器到协处理器的选择传送
MCRR	传送双字到协处理器
CDP2	选择协处理器数据处理
BKPT	断点
PLD	软预装载，存储器从地址装载
STRD	存储双字
STC2	从协处理器选择存储
LDRD	装载双字
LDC2	选择装载到协处理器
CLZ	前导零计数

注意：一个 Thumb BLX 包含两个连续的 Thumb 指令，并需四个时钟周期。

11.3.10 Thumb 指令集概论

Thumb 指令集是 ARM 指令集的一个重编码子集。

Thumb 指令集被分成：

- 分支指令
- 数据处理指令
- 装载和存储指令
- 批量装载和批量存储指令
- 异常产生指令

表 11-4 给出了 Thumb 指令助记符列表。

表 11-4 Thumb 指令助记符列表

助记符	操作
MOV	传送
ADD	加
SUB	减
CMP	比较
TST	测试
AND	逻辑与
EOR	逻辑异或
LSL	逻辑左移

ASR	算数右移
MUL	乘
B	分支
BX	分支并切换
LDR	装载字
LDRH	装载半字
LDRB	装载字节
LDRSH	装载带符号半字
LDMIA	批量装载
PUSH	寄存器入栈
BCC	条件分支
MVN	取反传送
ADC	带进位加
SBC	带借位减
CMN	取反比较
NEG	取反
BIC	位清零
ORR	逻辑或
LSR	逻辑右移
ROR	循环右移
BLX	分支，连接，切换
BL	分支并连接
SWI	软件中断
STR	存储字
STRH	存储半字
STRB	存储字节
LDRSB	装载带符号字节
STMIA	批量存储
POP	出栈到寄存器
BKPT	断点

11.4 CP15 协处理器

协处理器 15，或系统控制协处理器 CP15，被用于配置和控制下表中所有项：

- ARM9EJ-S
- 高速缓冲（ICache,DCache 和写缓冲器）
- TCM
- MMU
- 其他系统选项

为了控制这些功能部件，CP15 提供了 16 个附加寄存器。见表 11-5

表 11-5　　CP15 寄存器

寄存器	名称	读/写
0	ID 代码*	读/不可预测

0	Cache 类型*	读/不可预测
0	TCM 状态*	读/不可预测
1	控制	读/写
2	转换表基址	读/写
3	域访问控制	读/写
4	保留	无
5	数据错误状态*	读/写
5	错误地址*	读/写
7	Cache 操作	读/写
8	TLB 操作	不可预测/写
9	Cache 锁定+	读/写
9	TCM 区域	读/写
10	TLB 锁定	读/写
11	保留	无
12	保留	无
13	FCSE PID*	读/写
13	上下文 ID*	读/写
14	保留	无
15	测试配置	读/写

注意：*寄存器位置 0，5 和 13 每个提供访问多个寄存器的能力。
被访问的寄存器取决于操作码 2 字段的值。
+寄存器位置 9 提供访问多个寄存器的能力。访问的寄存器取决于 CRm 字段的值。

11.4.1 CP15 寄存器访问

CP15 寄存器仅可以在特权模式以下列方式访问：

• MCR(从 ARM 寄存器传送到协处理器)指令被用于写一个 ARM 寄存器到 CP15.

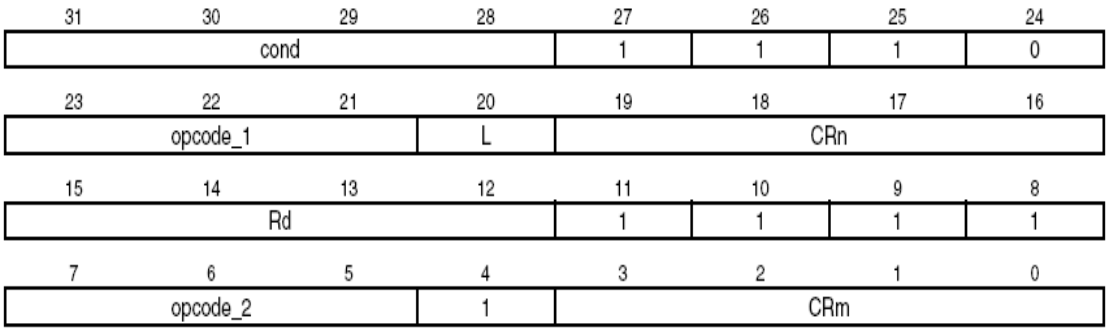
• MRC(从协处理器传送到 ARM 寄存器)指令被用于读 CP15 的值到一个 ARM 寄存器。

其他指令像 CDP,LDC,STC 可能导致一个未定义指令异常。

这些指令的汇编程序代码是：

MCR/MRC{cond} p15,opcode_1,Rd,CRn,CRm,opcode_2.

MCR,MRC 指令位模式如下所示：



• CRm[3：0]：指定协处理器操作

决定特定的协处理器操作。它的值取决于使用的 CP15 寄存器。详情请参考 CP15 特定寄存器行为。

- 操作码 2 [7：5]

决定特定协处理器操作码。缺省设置为零。

- RD[15：12]：ARM 寄存器

定义值将被传送至协处理器的 ARM 寄存器。如果选择 R15，结果是不可预测的。

- CRn[19：16]：协处理器

决定目标协处理器寄存器。

- L：指令位

0=MCR 指令

1=MRC 指令

- 操作码段 1[23：20]：协处理器代码

定义协处理器特定代码。其值对 CP15 是 c15。

- Cond[31: 28]: 条件

更多的细节，见 ARM926EJ-S　TRM 第二章，ref.DDI0198B.

11.5　存储器管理部件（MMU）

ARM926EJ-S 处理器实现了一个增强的 ARM V5 架构的 MMU，以向操作系统 Symbian OS,Windows CE，和 Linux 等提供其所需的虚拟存储特性。这些虚拟存储功能特性是存储器访问权限控制和虚拟地址到物理地址转换。

由 CPU 内核产生的虚拟地址被 FCSE(快速上下文变换扩展)用 CP15 寄存器 13 的值变换到一个修改了的虚拟地址（MVA）。MMU 用一个存储于物理存储器的单一，两级页表集转换 MVA 到物理地址。页表集中的每个入口包含访问权限和与虚拟地址对应的物理地址。

一级转换表包含 4096 个入口，与 MVA 的位[31：20]对应。这些入口包含一个指向含有相同属性信息（访问权限，域等）的 1MB 物理存储扇区的指针，或二级转换表一个入口的指针：粗页表或者细页表。

二级转换表包含两个子表，粗页表和细页表。粗页表里的一个入口包含一个指针，指向相同访问权限的大页和小页。细页表中的一个入口包含一个到大，小和极小页的指针。

表 11-6 展示了物理内存中每页的不同属性。

表 11-6 映射详述

映射名称	映射大小	访问允许	子页大小
段	1MB	段	-
大页	64KB	4 个分离的子页	16KB
小页	4KB	4 个分离的子页	1KB
极小页	1KB	极小页	-

MMU 包括：

- 访问控制逻辑
- 转换旁视缓冲区（TLB，Translation Look-aside Buffer）

- 转换表遍历硬件

11.5.1 访问控制逻辑

访问控制逻辑控制转换表中的每个入口的访问(权限)信息。

访问控制逻辑检查两种访问信息：域和访问许可。域（domain）是用于一个存储区域的访问控制的主要机制；共有 16 个域。它定义了操作能够进行的必要的条件。域决定了访问是否被允许还是被忽略。

二级访问控制机制是被为段，大页，小页和极小页定义的访问控制。段和极小页面有一个单一访问允许集，但大页和小页可以和 4 个访问允许集关联，每个子页(四分之一页)一个访问允许集。

11.5.2 转换旁视缓冲区(TLB)

转换旁视缓冲区(TLB)缓存已转换的表项从而避免每次都转换。当 TLB 包含一个 MVA(修改了的虚拟地址)入口，访问控制逻辑决定访问是否被允许并且输出对应于 MVA 的物理地址。如果访问被禁止，MMU 发信号给 CPU 内核来中止。

如果 TLB 不包含 MVA 的一个入口，转换表遍历硬件被引入，从物理内存中的转换表中检索转换信息。

11.5.3 转换表遍历硬件

转换表遍历硬件是一个逻辑，此逻辑遍历存储于物理内存中的转换表，得到物理地址和访问许可并且更新 TLB。

硬件遍历中的阶段数可以是一或二，取决于地址是否被标记为一个段映射访问或一个页面映射访问。

有三种容量的页面映射访问，一种容量的段映射访问。页面映射访问用于大页，小页和极小页。转换过程总是以一级读取启动。一个段映射访问仅需要一级读取，但一个页面映射访问需要一个额外的二级读取。MMU 的更多细节，请参考 ARM926EJ-S 参考手册第三章，ref.DDI0198B。

11.5.4 MMU 错误

MMU 在以下几种错误时产生一个中止：

- 对齐错误（仅对数据访问）
- 转换错误
- 域错误
- 权限错误

MMU 的访问控制机制检测发生这些错误的条件。如果错误是存储器访问引起的，MMU 中止访问并发错误信号给 CPU 内核。MMU 保存数据访问错误所产生的状态和错误地址信息，并将这些信息保存到数据错误状态寄存器和错误地址寄存器中。MMU 也保存由取指产生的错误信息到指令错误状态寄存器中。

当错误发生，错误状态寄存器（CP15 中的寄存器 r5）指示数据或预取指中止的原因，和中止的访问域数。错误地址寄存器（CP15 中寄存器 r6）保存与导致数据中止的访问相关联的 MVA。MMU 的更多细节，请参考 ARM926EJ-S 参考手册

第三章，ref.DDI0198B。

11.6 高速缓存和写缓冲

ARM926EJ-S 包含一个 16KB 的指令缓存(ICache)，一个 16KB 数据缓存（DCache），和一个写缓冲。虽然 ICache 和 DCache 拥有共同的特性，每种仍然有若干特定的机制。

高速缓存（ICache和DCache）是四路组相联的(four-way set associative), addressed, indexed and tagged using the Modified Virtual Address (MVA),带有一个8字长的cache line，DCache有两个dirty位。ICache和DCache提供了缓存锁定，缓存污染控制和line替换机制。

ARM926EJ-S 缓存支持一种新特性，叫做 allocate on read-miss(读取未命中), 也就是大家所共知的 wrapping。此功能特性使缓存能首先执行关键字缓存再充填。这就是说当对一个字的请求导致一个读取未命中(read-miss)时，则缓存执行一个 AHB 访问。缓存不是装载整个 line（8 words），而是首先装载关键字，所以处理器可以快速的读取，然后是剩余的字，而不管字被分配在 cache line 的哪个位置。

高速缓存和写缓冲被 CP15 寄存器 r1（控制），CP15 寄存器 r7（缓存操作）和 CP15 寄存器 r9(缓存锁定) 所控制。

11.6.1 指令缓存（ICache）

ICcahe 缓存取出来的指令，等待处理器执行。ICcache 可以通过写 1 到 CP15 寄存器 r1 的 I 位来使能，写 0 到该位来禁用。

当 MMU 被使能，所有的取指都取决于转换和权限检查。如果 MMU 被禁用，所有的取指是可缓存的，不做任何保护检查并且物理地址被平面映射(flat-ampped)到 MVA(即一一对应，不存在转换，译者注)。随着 MVA 被禁用，上下文切换将引发 ICache 清除和/或使无效。

当 ICache 被禁用，所有的取指直接出现在外部存储器（AHB）(见 ARM926EJ-S TRM 中 4-4 页表 4-1 和 4-2，ref.DDI0198B)。

复位时，ICache 条目是无效的并且 ICache 是禁用的。为了获得最佳性能，ICache 应该在复位后尽可能快地使能。

11.6.2 数据缓存（DCache）和写缓冲

ARM926EJ-S 包含一个 DCache 和一个写缓冲，以降低主内存带宽和延迟对数据访问性能上的影响。DCache 和写缓冲的操作是紧密相连的。

11.6.2.1 DCache

DCache 需要由 MMU 来使能。所有的数据访问取决于 MMU 权限和转换检查。被 MMU 中止的数据访问不会导致 line 填充或数据访问出现在 AMBA AHB 接口。如果 MMU 被禁用，所有的数据访问是不可缓存的，不可缓冲的，没有保护检查，并出现在 AHB 总线。所有的地址是平面映射(flat-ampped)的，VA=MVA=PA,每次上下文切换发生将引起 DCache 清除和/或使无效。

DCache 保存每条 line 被装载时的源物理地址标记(PA Tag)，并使用该 Tag 将修改后的 line 回写到外部存储器。这就是说回写操作无需 MMU 干预。

DCache 中的每条 line（8 words）有两个 dirty 标志位，一个用于首 4 words，另一个用于次 4 words。这些位，如果置位，标记关联的半条 line 为 dirty。如果 cache

line 由于一个 line 填充或一个缓存清除操作被代替，则 dirty 标志位被用于决定是否所有的，一半或无需将 cache line 回写到存储器。

DCache 可以通过写 1 或 0 到 CP15 寄存器 r1 中的位 C（见 ARM926EJ-S TRM 中 4-5 页表 4-3 和 4-4，ref.DDI0222B）来使能或禁用.

DCache 支持写通(write-though)或者写回(write-back)缓存操作，具体操作模式由存储器区域的 MMU 转换表中的 C 位和 B 位所选择。

DCache包含一个8 data word入口，单地址入口写回缓冲，用于在cache line eviction或dirty line清除时保存写回的数据。

写缓冲可以保存 16 字数据和 4 个独立的地址。DCache 和写缓冲操作是紧密相连的，它们的配置被 MMU 转换表中的页面描述符在每个段设置。

11.6.2.2 写缓冲

ARM926EJ-S 包含一个写缓冲，此写缓冲有一个 16 字数据缓冲和一个四地址缓冲。写缓冲适用于对某个可缓冲区域进行的所有写操作，包括写通区域和写回区域。写缓冲还可以避免处理器写到外部存储器时发生 stall。当一个存储产生，数据以内核速度（高速）被写到写缓冲。写缓冲接着以总线速度(比内核速度慢)完成数据存储到外部存储器的过程。在此期间，ARM9EJ-S 处理器可以执行其他任务。

DCache 和写缓冲支持写回和写通存储器区域，由 MMU 转换表的中的段和页面描述符中的 C 位，B 位控制。

写通操作

当一个缓存写命中(hit)，Dcache 被更新。更新后的数据被写入写缓冲，后者将其写入外部存储器。

当一个缓存写未命中(miss)，被轮换(round robin)或其他算法所选的一条 line 被存储在写缓冲器，写缓冲器传输其内容到外部存储器。

写回操作

当一个缓存写命中，缓存 line 或半条 line 被标记为 dirty，就是说缓存的内容没有被更新到对应的外部存储器中。

当一个缓存写未命中，被轮换(round robin)或其他算法所选的一条 line 被存储在写缓冲器，写缓冲器传输其内容到外部存储器。

11.7 紧耦合存储器接口(TCM)

11.7.1 TCM 描述

ARM926EJ-S 处理器有一个紧耦合存储器接口，该接口使得处理器能直接访问独立的指令和数据 TCM(ITCM 和 DTCM)。TCM 被用于存储实时性和性能要求极高的代码，它还提供一个 DMA 支持机制。不像 AHB 访问外部存储器，访问 TCM 是快速，确定的，不造成总线的负荷。

用户可以在以下范围内独立地配置每个 TCM 的容量：[0KB 到 64KB]的 ITCM 和 [0KB 到 64KB]的 DTCM。

TCM 可以使用两种方法配置： HMATRIX TCM 寄存器和 CP15 中的 TCM 区域(region)寄存器（寄存器 r9），两步都应该被执行。HMATRIX TCM 寄存器设置 TCM 容量而 CP15 中的 TCM 区域寄存器（寄存器 r9）映射 TCM 并使能 TCM。

ARM9EJ-S 内核的数据域(data side)可以访问 ITCM。为了能将代码载入 ITCM，这是必要的，比如 SWI 和仿真指令处理程序，以及访问基于 PC 的文字池。

11.7.2 使能和禁用 TCM

在任何使能操作之前，用户应该配置 HMATRIX TCM 寄存器中的 TCM 容量。

接着由 CP15 中的 TCM 区域寄存器（寄存器 r9）使能 TCM。用户应该使用由 HMATRIX TCM 寄存器所确定的容量。更多的细节和编程技巧，请参考 ARM926EJ-S 中的第 2.3 章，ref.DDI0222B.

11.7.3 TCM 映射

TCM 可以被分配到存储器映射中的任何地方，使用一个单独的区域作为 ITCM 而使用另一个独立的区域作为 DTCM。TCM 被设置于物理地址(TCM is physically addressed)并能被放置在物理地址空间的任何地方。然而，一个 TCM 的基地址必须和其容量对齐(align)，并且 DTCM 和 ITCM 区必须无重叠。TCM 映射通过 CP15 中的 TCM 区寄存器(r9)来执行。用户应该给 TCM 输入正确的映射地址。

11.8 总线接口部件

ARM926EJ-S 以一个总线接口部件（BIU）来仲裁和调度 AHB 请求。BIU 实现一个多层 AHB，基于 AHB-Lite 协议，此协议使能一个系统中多种 AHB 主控和从设备之间的并行访问路径。这通过用一个更复杂的互联矩阵来实现，它增加了总线带宽，提供了一个更灵活的系统架构。

多主控总线架构有许多好处：

- 它使得开发一个增加了总线带宽，拥有灵活架构的多主控系统得以实现。
- 每个 AHB 层变得简单，因为只有一个主控，无需仲裁或主控到从设置的多路选择。实现 AHB-Lite 协议的 AHB 层，不必支持请求和应答，也不必支持重试和拆分事务(split transactions)。
- 当多个主控同时想要访问相同的从设备时，仲裁变得有效。

11.8.1 支持的传送

ARM926EJ-S 处理器把所有的 AHB 访问当作单字，4 字突发或 8 字突发执行，任何大小不是 1，4，8 字的 ARM9EJ-S 内核请求被分成 1，4，8 字大小的包。**注意**：Atmel 总线和 AHB-Lite 协议兼容，因此，不支持拆分和重试请求。

表 8 给出了支持的传输方式和使用到他们的不同事务。

表 11-7 支持传送

HBurst[2:0]	描述	
SINGLE	单传送	字，半字，或字节的单传送 • 数据写（NCNB,NCB,WT,或 DCache 中失败的 WB） • 数据读（NCNB,或 NCB） • NC 取指（预取指和非预取指） • 页表遍历读
INCR4	4 字增长突发	若使能，half-line 缓存写回，预取指。4 字突发 NCNB，NCB,WT,或 WB 写
INCR8	8 字增长突发	Full-line 缓存写返回，8 字突发 NCNB，NCB,WT,或 WB 写

WRAP8	8 字 wrapping 突发	Cache 填充

11.8.2 Thumb 取指

所有取指，无论 ARM9EJ-S 内核的状态如何，都被处理为在 AHB 上的 32 位访问。如果 ARM9EJ-S 运行在 Thumb 状态，则两条指令可以一次被读取。

11.8.3 地址对齐

ARM926EJ-S BIU 执行地址对齐检查并对齐 AHB 地址到必要的边界。16 位访问对齐到半字边界，32 位访问对齐到字边界。

12. AT91SAM9261 调试和测试

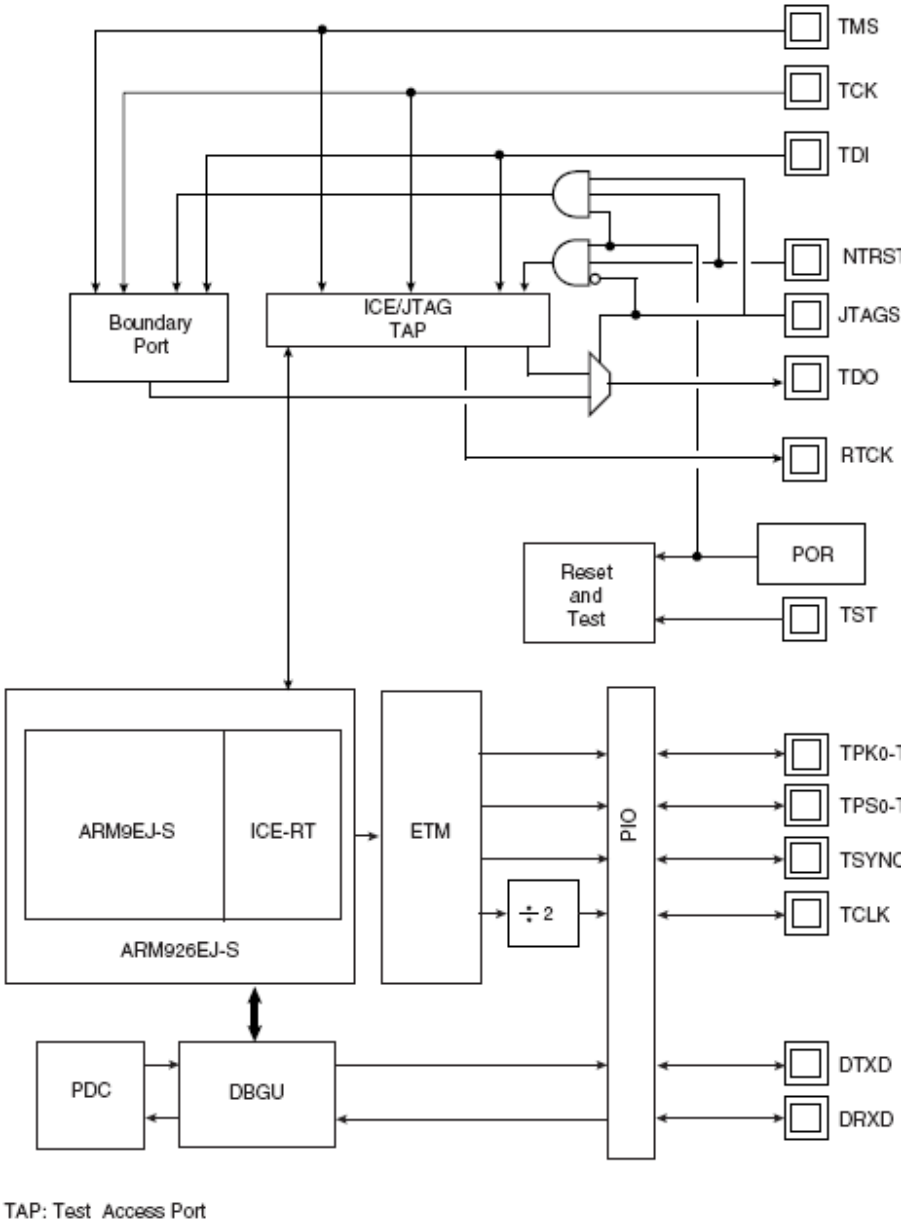
12.1 概述

AT91SAM9261 有许多补充的调试和测试能力。一个通用的 JTAG/ICE (In-Circuit Emulator)端口被用来进行标准的调试，像下载代码和单步执行程序。ETM(Embedded Trace Macrocell:嵌入式跟踪宏单元)提供更多的成熟的调试特性，象地址和数据比较器，半速率时钟模式，计数器，sequencer 和 FIFO。调试部件提供一个双引脚 UART，可以用于上传一个应用程序到内部 SRAM。它管理内部 COMMTX 和 COMMRX 信号的中断处理，这些信号跟踪调试通信通道的行为。

一套调试和测试专用的输入/输出引脚使得基于 PC 的测试环境可以直接访问以上这些功能。

12.2 方块图

图 12-1 调试和测试 方块图



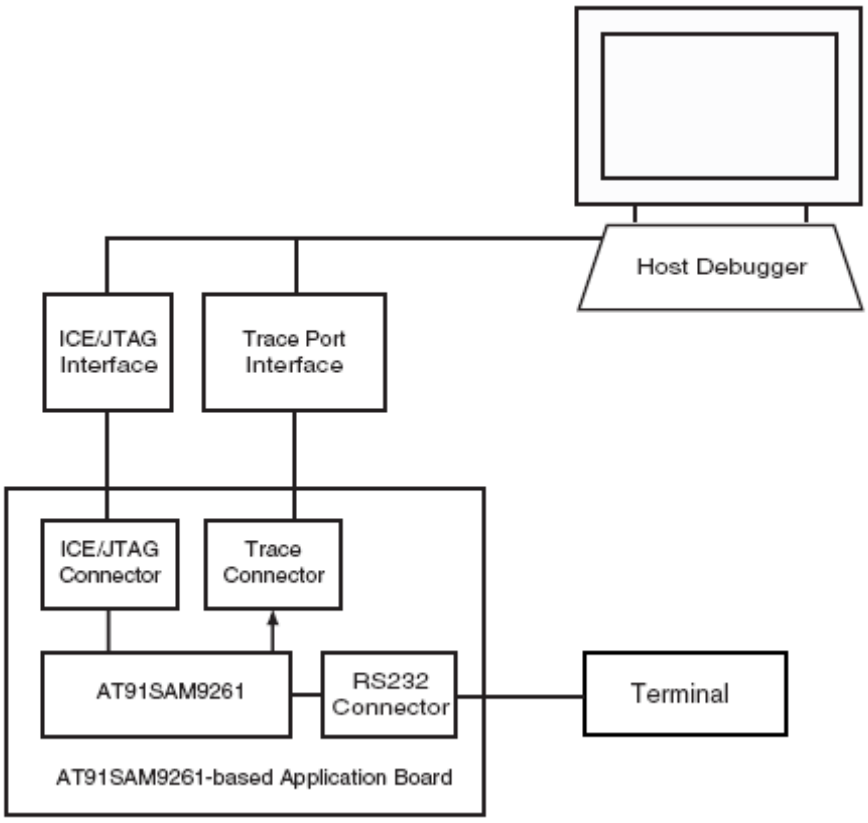
TAP: Test Access Port

12.3 应用举例

12.3.1 调试环境

62 页图 12-3 展示了一个完全的调试环境的例子。ICE/JTAG 接口被用作标准调试功能，下载代码和单步执行程序。跟踪端口(Trace Port)接口被用作跟踪信息。一个运行在个人计算机上的软件调试器利用 ICE/JTAG 接口提供一个配置跟踪端口接口的用户接口。

图 12-2 应用程序调试和测试环境举例

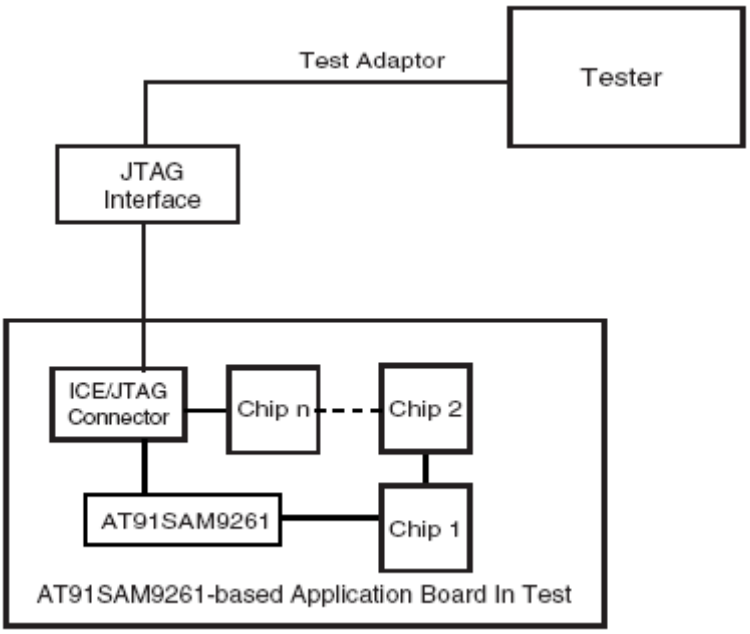


12.3.2 测试环境

图 12-3 展示了一个测试环境例子。测试向量被测试器发送并解释。

在此例中，‘测试板’使用许多 JTAG 兼容设备设计。这些设备可以被连接形成一个扫描链。

图 12-3 应用程序测试环境举例



12.4 调试和测试引脚描述

表 12-1 调试和测试引脚列表

引脚名称	功能	类型	有效电平
复位/测试			
NRST	微控制器复位	输入/输出	低
TST	测试模式选择	输入	高
ICE 和 JTAG			
TCK	测试时钟	输入	
TDI	测试数据输入	输入	
TDO	测试数据输出	输出	
TMS	测试模式选择	输入	
NTRST	测试模式复位信号	输入	低
RTCK	返回测试时钟	输出	
JTAGSEL	JTAG 选择	输入	
ETM			
TSYNC	跟踪同步信号	输出	
TCLK	跟踪时钟	输出	
TPS0-TPS2	跟踪 ARM 管道状态	输出	
TPK0-TPK15	跟踪包端口	输出	
调试部件			
DRXD	调试接收数据	输入	
DTXD	调试传输数据	输出	

12.5 功能描述

12.5.1 测试引脚

一个专用引脚，TST，被用于定义设备操作模式。用户必须确保此引脚拉至低电平以确保正常操作。和此引脚关联的其他值为生产测试预留。

12.5.2 嵌入式电路内部仿真器(Embedded In-circuit Emulator)

ARM9EJ-S 嵌入式 ICE-RT™ 通过 ICE/JTAG 端口支持。它由一个 ICE 接口连接到一个计算机主机。调试支持由一个内嵌在 ARM926EJ-S 中的 ARM9EJ-S 内核实现。ARM926EJ-S 的内部状态由一个 ICE/JTAG 端口检查，此端口允许指令在不通过外部数据总线的情况下被串行插入内核的流水线。因此，当在调试状态，一个批量存储（STM）可以被插入到指令流水线，这将可以导出 ARM9EJ-S 寄存器的内容。此数据可以在不影响系统其余部分的情况下被串行移出。

ARM9EJ-S 处理器内部有两条扫描链，扫描链支持测试，调试，和 EmbeddedICE-RT 的编程。此扫描链被 ICE/JTAG 控制。

当 JTAGSEL 为低电平时，EmbeddedICE 模式被选择。直接在 ICE 与 JTAG 操作之间转换是不能实现的。在 JTAGSEL 电平改变后必须进行芯片复位。

关于 EmbeddedICE-RT 更多的细节，参见 ARM 文件 ARM9EJ-S 技术参考手册（DDI0222A）。

12.5.3 JTAG 信号描述

TMS 是测试模式输入，此输入控制测试接口状态机器的转换。

TDI 是测试数据输入，此输入提供数据到 JTAG 寄存器（边界扫描寄存器，指令寄存器，或其他数据寄存器）。

TDO 是测试数据输出，此输出被用作从 JTAG 寄存器到控制测试设备串行的输出数据。它从边界扫描链（或其他 JTAG 寄存器）传送采样值并传送采样值到串行测试电路中的下一个芯片。

NTRST（在 IEEE 标准 1149.1 中为可选信号）是一个测试复位输入，此输入在 ARM 内核是强制的并且用于复位调试逻辑。在 Atmel 基于 ARM926EJ-S 的内核中，NTRST 是一个上电复位输出。上电时有效。如果必要，用户还可以保持 NTRST 有效 2.5 个 MCK 周期来复位调试逻辑。

TCK 是测试时钟输入，此输入使能测试接口。TCK 由控制测试的设备而不是被测试设备产生。它可以使用任何频率脉冲。**注意，**ARM926EJ-S 内核上最大的 JTAG 时钟速率是 CPU 时钟的六分之一。在一个从 32.768kHz 慢时钟运行的 ARM9E 上，最大的初始 JTAG 时钟速率是 5.45kHz。

RTCK 是一个返回测试时钟。不是一个 IEEE 标准 1149.1 中的信号，这个信号被引入是为了使仿真器能更好的处理时钟。从一些 ICE 接口探测，此返回信号可以被用于同步 TCK 时钟而不用担心 ICE 接口时钟和系统时钟之间需要遵守给定的六分之一比率。此信号仅在 JTAG ICE 模式下可用，在边界扫描模式下不可用。

12.5.4 调试部件

调试部件提供一个双引脚(DXRD 和 TXRD) USART。此 USART 可以被用于一些调试和跟踪目的，并提供一个现场(in-site)编程和调试监视通信的理想方案。此外，两个与通道关联的外设 DMA 控制器使得这些任务包的处理时间减到最小。

调试部件还管理 COMMTX 和 COMMRX 信号的中断处理，这些信号来于 ICE 并用于跟踪调试通信通道的活动。调试部件可以阻止通过 ICE 接口来访问系统。

一个特定的寄存器，调试部件芯片 ID 寄存器，给出了产品版本和其内部结构的信息。

AT91SAM9261 调试部件芯片 ID 是一个 32 位宽度的值：0x0197 03A0。

12.5.5 嵌入式跟踪宏单元

AT91SAM926 有一个嵌入式跟踪宏单元(ETM)功能部件，ETM 紧密地连接于 ARM926EJ-S 处理器。嵌入式跟踪是一个标准的中等以上规模的实现，并包含以下资源：

- 四对地址比较器
- 两个数据比较器
- 八个存储器映射解码输入
- 两个 16 位计数器
- 一个 3-state sequencer
- 四个外部输入
- 一个外部输出
- 一个 45 字节 FIFO

AT91SAM9261 的嵌入式跟踪宏单元工作在半速率时钟模式因而集成了一个时钟分频器。因此，所有跟踪端口信号的最大频率不超过 ARM926EJ-S 时钟速率的一半。

ETM 输入和输出资源在 AT91SAM9261 中不被使用。

关于 ETM 的更多细节，参见 ARM 文档：

1．ETM（Rev2p2）技术参考手册（DDI0157F）

2．ETM 规格说明(IHI 0014J)

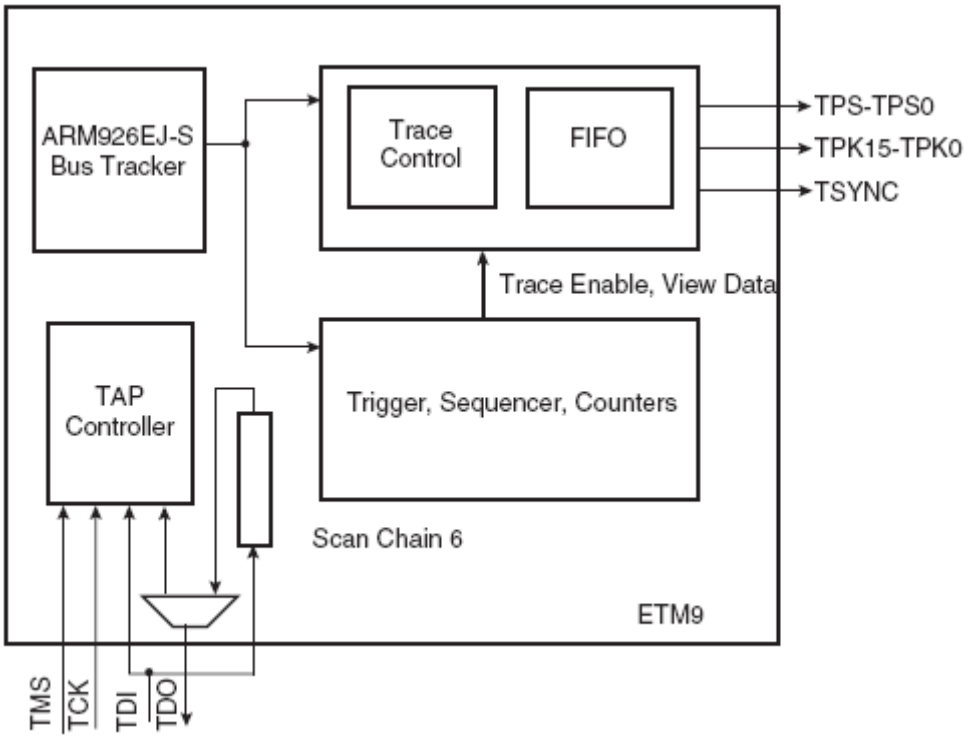
12.5.5.1 跟踪端口

跟踪端口由以下引脚构成：

- TSYNC-同步信号（表示在跟踪包端口上一个转移顺序的开始。）
- TCLK-跟踪端口时钟， ARM926EJ-S 处理器时钟频率的二分之一。
- TPS0 到 TPS2-表示在每个跟踪时钟边沿的处理器状态。
- TPK0 到 TPK15-跟踪包数据值。

跟踪包信息（地址，数据）和 TPS 表示的处理器状态相关联。一些处理器状态在跟踪包端口没有关联的附加数据（比如说，一个失败了的指令条件代码）。此包 8 位宽度，每周期可以输出多达两个包。

图 12-4 ETM9 方块图



12.5.5.2 实现细节

此段给出了嵌入式跟踪资源的概述。

Three-state Sequencer

Sequencer 有三可能的下一状态（一个专用于自己，两个用于其他），sequencer 在每个时钟周期都可以改变状态。状态转换由内部事件控制。如果用户需要 multi-stage trigger schemes，触发事件将基于一个 sequencer 状态。

地址比较器

在单一模式中，地址比较器将指令地址或数据地址和用户编程的地址进行比较。

在范围模式中，地址比较器被成对安排来形成一个虚拟地址范围资源。

地址比较器编程的细节是：

- 第一个比较器用范围开始地址编程。
- 第二个比较器用范围结束地址编程。
- 如果地址在以下范围内，则资源匹配：
  -（地址>=范围开始地址）AND（地址<范围结束地址）
- 如果两个地址比较器没有用同一种方法配置，则会产生不可预料的行为。

数据比较器

每个全地址比较器和一个特定数据比较器关联。仅当装载和存储操作发生时，一个数据比较器才被用于监测数据总线。

一个数据比较器有一个数值寄存器和一个屏蔽寄存器，因此能仅对一个预编程值的某些特定位和数据总线进行比较。

存储器解码输入

八个存储器映射解码输入被连接于客户地址解码器。地址解码器把存储器分成片上 SRAM 区，片上 ROM 区，和外设区。地址解码器还优化

ETM9 跟踪触发。

表 12—2 ETM 存储器映射输入层

产品资源	区域	访问类型	开始地址	结束地址
SRAM	内部	数据	0x0000 0000	0x002F FFFF
SRAM	内部	取指	0x0000 0000	0x002F FFFF
ROM	内部	数据	0x0040 0000	0x004F FFFF
ROM	内部	取指	0x0040 0000	0x004F FFFF
外部总线接口	外部	数据	0x1000 0000	0x8FFF FFFF
外部总线接口	外部	取指	0x1000 0000	0x8FFF FFFF
用户外设	内部	数据	0xF000 0000	0xFFFF BFFF
系统外设	内部	数据	0xFFFF C000	0xFFFF FFFF

FIFO

一个 45 字节 FIFO 被用于存储跟踪数据。FIFO 用于从跟踪包中分离流水线状态，因此，FIFO 可以被用于缓冲跟踪包。

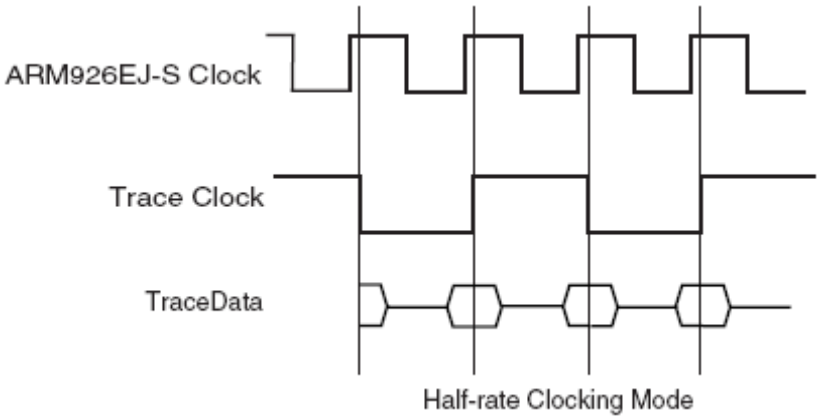
当 FIFO 溢出或当 FIFO 字节比用户编程的字节数少，则嵌入式跟踪宏单元检测到 FIFO 溢出。

半速率时钟模式

ETM9 在半速率模式下被实现，此模式允许在跟踪时钟的上升沿和下降沿进行数据跟踪。

半速率模式可以保证高速系统（达 100Mhz）的信号完整性。

图 12-5 半速率时钟模式



必须注意跟踪捕捉系统的选择，此系统必须要支持半速率时钟功能。

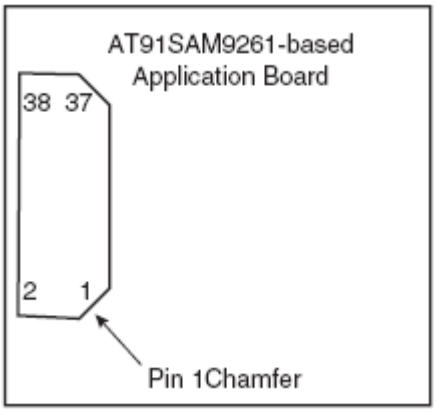
12.5.5.3 应用板限制

TCLK 信号需要小心设置，一些定时参数是必需的。

特定目标系统连接头采用 AMP Mictor 连接器。

连接器必须如图 12-6 所示定位于应用板。以下的 PCB 视图是从顶部俯视的，跟踪连接器安装在靠近板子边沿。这就使得跟踪端口分析器对互连目标的影响最小化。

图 12-6 AMP Mictor 连接器



12.5.6 IEEE 1149.1JTAG 边界扫描

IEEE 1149.1 JTAG 边界扫描允许独立于设备封装技术的引脚级访问。

IEEE 1149.1JTAG 边界扫描当 JTAGSEL 为高电平时被使能。SAMPLE,EXTEST 和 BYPASS 功能被实现。在 ICE 调试模式，ARM 处理器回应一个非 JTAG 芯片 ID 响应到 ICE 系统用于识别处理器。这不适用于 IEEE 1149.1 JTAG。

直接转换 JTAG 和 ICE 操作是不可能的。JTAGSEL 被改变后必须执行一个复位操作。

提供一个边界扫描描述符语言（BSDL）文件用于建立测试。

12.5.6.1 JTAG 边界扫描寄存器

边界扫描寄存器（BSR）包含 484 位，这些位对应于主动引脚和关联的控制信号。

每个 AT91SAM9261 输入/输出引脚对应到 BSR 中的一个 3 位寄存器。OUTPUT 位包含能被强制输出在引脚上的数据。INPUT 位提供加到引脚上的数据以供观测。CONTROL 位选择引脚的方向。

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器

位号	引脚名称	引脚类型	关联的 BSR 单元
483	A18	OUT	OUTPUT
482	A[22:16]		CONTROL
481	A18	OUT	OUTPUT
480	A20	OUT	OUTPUT
479	A21	OUT	OUTPUT
478	A22	OUT	OUTPUT
477	NCS0	OUT	OUTPUT
476	A[7:0]		CONTROL
475	NCS1	OUT	OUTPUT
474	NCS0/NCS1/NCS2/NCS3 NRD/NWR0/NWR1/NWR3		CONTROL
473	NCS2	OUT	OUTPUT
472	NCS3	OUT	OUTPUT
471	NRD	OUT	OUTPUT
470	NWR0	IN/OUT	INPUT
469			OUTPUT
468	NWR1	IN/OUT	INPUT
467			OUTPUT
466		内部	I
465	NWR3	OUT	OUTPUT
464	SDRAMCKE	OUT	OUTPUT
463	SDRAMCKE/RAS/CAS SDA10/SDWE		CONTROL
462	SDRAMCLK	IN/OUT	INPUT
461			OUTPUT
460			CONTROL
459	RAS	OUT	OUTPUT
458	CAS	OUT	OUTPUT
457	SDWE	OUT	OUTPUT

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
456	D0	IN/OUT	INPUT
455			OUTPUT
454			CONTROL
453		内部	
452	D1	IN/OUT	INPUT
451			OUTPUT
450			CONTROL
449	D2	IN/OUT	INPUT
448			OUTPUT
447			CONTROL
446	D3	IN/OUT	INPUT
445			OUTPUT
444			CONTROL
443	D4	IN/OUT	INPUT
442			OUTPUT
441			CONTROL
440		内部	
439	D5	IN/OUT	INPUT
438			OUTPUT
437			CONTROL
436	D6	IN/OUT	INPUT
435			OUTPUT
434			CONTROL
433	D7	IN/OUT	INPUT
432			OUTPUT
431			CONTROL
430	D8	IN/OUT	INPUT
429			OUTPUT
428			CONTROL
427		内部	
426	D9	IN/OUT	INPUT
425			OUTPUT
424			CONTROL
423	D10	IN/OUT	INPUT
422			OUTPUT
421			CONTROL

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
420	D11	IN/OUT	INPUT
419			OUTPUT
418			CONTROL
417	D12	IN/OUT	INPUT
416			OUTPUT
415			CONTROL
414		内部	
413	D13	IN/OUT	INPUT
412			OUTPUT
411			CONTROL
410	D14	IN/OUT	INPUT
409			OUTPUT
408			CONTROL
407	D15	IN/OUT	INPUT
406			OUTPUT
405			CONTROL
404	PC16	IN/OUT	INPUT
403			OUTPUT
402			CONTROL
401		内部	
400	PC17	IN/OUT	INPUT
399			OUTPUT
398			CONTROL
397		内部	
396	PC18	IN/OUT	INPUT
395			OUTPUT
394			CONTROL
393		内部	
392	PC19	IN/OUT	INPUT
391			OUTPUT
390			CONTROL
389		内部	
388	PC30	IN/OUT	INPUT
387			OUTPUT
386			CONTROL
385		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
384	PC31	IN/OUT	INPUT
383			OUTPUT
382			CONTROL
381		内部	
380	PC20	IN/OUT	INPUT
379			OUTPUT
378			CONTROL
377		内部	
376	PC21	IN/OUT	INPUT
375			OUTPUT
374			CONTROL
373		内部	
372	PC22	IN/OUT	INPUT
371			OUTPUT
370			CONTROL
369		内部	
368	PC23	IN/OUT	INPUT
367			OUTPUT
366			CONTROL
365		内部	
364	PC24	IN/OUT	INPUT
363			OUTPUT
362			CONTROL
361		内部	
360	PC25	IN/OUT	INPUT
359			OUTPUT
358			CONTROL
357		内部	
356	PC26	IN/OUT	INPUT
355			OUTPUT
354			CONTROL
353		内部	
352	PC27	IN/OUT	INPUT
351			OUTPUT
350			CONTROL
349		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
348	PC28	IN/OUT	INPUT
347			OUTPUT
346			CONTROL
345		内部	
344	PC29	IN/OUT	INPUT
343			OUTPUT
342			CONTROL
341		内部	
340	PC0	IN/OUT	INPUT
339			OUTPUT
338			CONTROL
337		内部	
336	PC1	IN/OUT	INPUT
335			OUTPUT
334			CONTROL
333		内部	
332	PC2	IN/OUT	INPUT
331			OUTPUT
330			CONTROL
329		内部	
328	PC3	IN/OUT	INPUT
327			OUTPUT
326			CONTROL
325		内部	
324	PC4	IN/OUT	INPUT
323			OUTPUT
322			CONTROL
321		内部	
320	PC5	IN/OUT	INPUT
319			OUTPUT
318			CONTROL
317		内部	
316	PC6	IN/OUT	INPUT
315			OUTPUT
314			CONTROL
313		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
312	PC7	IN/OUT	INPUT
311			OUTPUT
310			CONTROL
309		内部	
308	PC8	IN/OUT	INPUT
307			OUTPUT
306			CONTROL
305		内部	
304	PC9	IN/OUT	INPUT
303			OUTPUT
302			CONTROL
301		内部	
300	PC10	IN/OUT	INPUT
299			OUTPUT
298			CONTROL
297		内部	
296	PC11	IN/OUT	INPUT
295			OUTPUT
294			CONTROL
293		内部	
292	PC12	IN/OUT	INPUT
291			OUTPUT
290			CONTROL
289		内部	
288	PC13	IN/OUT	INPUT
287			OUTPUT
286			CONTROL
285		内部	
284	PC14	IN/OUT	INPUT
283			OUTPUT
282			CONTROL
281		内部	
280	PC15	IN/OUT	INPUT
279			OUTPUT
278			CONTROL
277		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
276	PA0	IN/OUT	INPUT
275			OUTPUT
274			CONTROL
273		内部	
272	PA1	IN/OUT	INPUT
271			OUTPUT
270			CONTROL
269		内部	
268	PA2	IN/OUT	INPUT
267			OUTPUT
266			CONTROL
265		内部	
264	PA3	IN/OUT	INPUT
263			OUTPUT
262			CONTROL
261		内部	
260	PA4	IN/OUT	INPUT
259			OUTPUT
258			CONTROL
257		内部	
256	PA5	IN/OUT	INPUT
255			OUTPUT
254			CONTROL
253		内部	
252	PA6	IN/OUT	INPUT
251			OUTPUT
250			CONTROL
249		内部	
248	PA7	IN/OUT	INPUT
247			OUTPUT
246			CONTROL
245		内部	
244	PA8	IN/OUT	INPUT
243			OUTPUT
242			CONTROL
241		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
240	PA9	IN/OUT	INPUT
239			OUTPUT
238			CONTROL
237		内部	
236	PA10	IN/OUT	INPUT
235			OUTPUT
234			CONTROL
233		内部	
232	PA11	IN/OUT	INPUT
231			OUTPUT
230			CONTROL
229		内部	
228	PA12	IN/OUT	INPUT
227			OUTPUT
226			CONTROL
225		内部	
224	PA13	IN/OUT	INPUT
223			OUTPUT
222			CONTROL
221		内部	
220	PA14	IN/OUT	INPUT
219			OUTPUT
218			CONTROL
217		内部	
216	PA15	IN/OUT	INPUT
215			OUTPUT
214			CONTROL
213		内部	
212	PA16	IN/OUT	INPUT
211			OUTPUT
210			CONTROL
209		内部	
208	PA17	IN/OUT	INPUT
207			OUTPUT
206			CONTROL
205		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
204	PA18	IN/OUT	INPUT
203			OUTPUT
202			CONTROL
201		内部	
200	PA19	IN/OUT	INPUT
199			OUTPUT
198			CONTROL
197		内部	
196	PA20	IN/OUT	INPUT
195			OUTPUT
194			CONTROL
193		内部	
192	PA21	IN/OUT	INPUT
191			OUTPUT
190			CONTROL
189		内部	
188	PA22	IN/OUT	INPUT
187			OUTPUT
186			CONTROL
185		内部	
184	PA23	IN/OUT	INPUT
183			OUTPUT
182			CONTROL
181		内部	
180	PA24	IN/OUT	INPUT
179			OUTPUT
178			CONTROL
177		内部	
176	PA25	IN/OUT	INPUT
175			OUTPUT
174			CONTROL
173		内部	
172	PA26	IN/OUT	INPUT
171			OUTPUT
170			CONTROL
169		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
168	PA27	IN/OUT	INPUT
167			OUTPUT
166			CONTROL
165		内部	
164	PA28	IN/OUT	INPUT
163			OUTPUT
162			CONTROL
161		内部	
160	PA29	IN/OUT	INPUT
159			OUTPUT
158			CONTROL
157		内部	
156	PA30	IN/OUT	INPUT
155			OUTPUT
154			CONTROL
153		内部	
152	PA31	IN/OUT	INPUT
151			OUTPUT
150			CONTROL
149		内部	
148	PB0	IN/OUT	INPUT
147			OUTPUT
146			CONTROL
145		内部	
144	PB1	IN/OUT	INPUT
143			OUTPUT
142			CONTROL
141		内部	
140	PB2	IN/OUT	INPUT
139			OUTPUT
138			CONTROL
137		内部	
136	PB3	IN/OUT	INPUT
135			OUTPUT
134			CONTROL
133		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
132	PB4	IN/OUT	INPUT
131			OUTPUT
130			CONTROL
129		内部	
128	PB5	IN/OUT	INPUT
127			OUTPUT
126			CONTROL
125		内部	
124	PB6	IN/OUT	INPUT
123			OUTPUT
122			CONTROL
121		内部	
120	PB7	IN/OUT	INPUT
119			OUTPUT
118			CONTROL
117		内部	
116	PB8	IN/OUT	INPUT
115			OUTPUT
114			CONTROL
113		内部	
112	PB9	IN/OUT	INPUT
111			OUTPUT
110			CONTROL
109		内部	
108	PB10	IN/OUT	INPUT
107			OUTPUT
106			CONTROL
105		内部	
104	PB11	IN/OUT	INPUT
103			OUTPUT
102			CONTROL
101		内部	
100	PB12	IN/OUT	INPUT
99			OUTPUT
98			CONTROL
97		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
96	PB13	IN/OUT	INPUT
95			OUTPUT
94			CONTROL
93		内部	
92	PB14	IN/OUT	INPUT
91			OUTPUT
90			CONTROL
89		内部	
88	PB15	IN/OUT	INPUT
87			OUTPUT
86			CONTROL
85		内部	
84	PB16	IN/OUT	INPUT
86			OUTPUT
85			CONTROL
84		内部	
83	PB17	IN/OUT	INPUT
82			OUTPUT
81			CONTROL
80		内部	
79	PB18	IN/OUT	INPUT
78			OUTPUT
77			CONTROL
76		内部	
75	PB19	IN/OUT	INPUT
74			OUTPUT
73			CONTROL
72		内部	
71	PB20	IN/OUT	INPUT
70			OUTPUT
69			CONTROL
68		内部	
67	PB21	IN/OUT	INPUT
66			OUTPUT
65			CONTROL
64		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

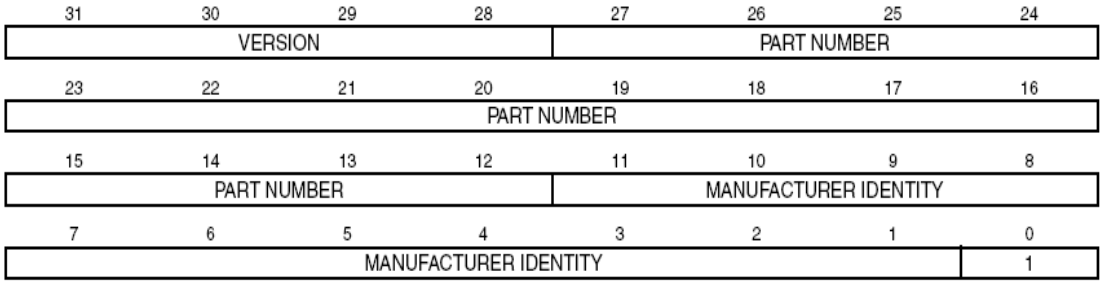
位号	引脚名称	引脚类型	关联的 BSR 单元
60	PB22	IN/OUT	INPUT
59			OUTPUT
58			CONTROL
57		内部	
56	PB23	IN/OUT	INPUT
55			OUTPUT
54			CONTROL
53		内部	
52	PB24	IN/OUT	INPUT
51			OUTPUT
50			CONTROL
49		内部	
48	PB25	IN/OUT	INPUT
47			OUTPUT
46			CONTROL
45		内部	
44	PB26	IN/OUT	INPUT
43			OUTPUT
42			CONTROL
41		内部	
40	PB27	IN/OUT	INPUT
39			OUTPUT
38			CONTROL
37		内部	
36	PB28	IN/OUT	INPUT
35			OUTPUT
34			CONTROL
33		内部	
32	PB29	IN/OUT	INPUT
31			OUTPUT
30			CONTROL
29		内部	
28	PB30	IN/OUT	INPUT
27			OUTPUT
26			CONTROL
25		内部	

表 12-3 AT91SAM9261 JTAG 边界扫描寄存器(续)

位号	引脚名称	引脚类型	关联的 BSR 单元
24	PB31	IN/OUT	INPUT
23			OUTPUT
22			CONTROL
21		内部	
20	A0	OUT	OUTPUT
19		内部	
18	A1	OUT	OUTPUT
17	A2	OUT	OUTPUT
16	A3	OUT	OUTPUT
15	A4	OUT	OUTPUT
14	A5	OUT	OUTPUT
13	A6	OUT	OUTPUT
12	A7	OUT	OUTPUT
11	A8	OUT	OUTPUT
10	A[15:8]		CONTROL
09	A9	OUT	OUTPUT
08	A10	OUT	OUTPUT
07	SDA10	OUT	OUTPUT
06	A11	OUT	OUTPUT
05	A12	OUT	OUTPUT
04	A13	OUT	OUTPUT
03	A14	OUT	OUTPUT
02	A15	OUT	OUTPUT
01	A16	OUT	OUTPUT
00	A17	OUT	OUTPUT

12.5.7 ID 代码寄存器

访问：只读



- VERSION[31：28]：产品版本号

设置为 0x0.

- PART NUMBER[27：12]：产品部件号

产品部件号是 0x5B08

- MANUFACTURER IDENTITY[11：1]

设置为 0x01F

Bit[0]根据 IEEE Std.1149.1 要求

设置为 0x1.

JTAG ID 代码值是 0x05B0_803F.

13. AT91SAM9261 启动程序

13.1 描述

启动程序集成了不同的程序，这些程序管理下载和/或上传产品的不同存储器。

首先，它初始化调试部件串行端口（DBGU）和 USB 设备端口。

接着执行 DataFlash 启动程序。此程序在连接于 SPI 的 DataFlash 中寻找七个有效的 ARM 异常向量的序列。所有这些向量必须是 B-branch 或 LDR 装载寄存器指令，除了第六个向量。第六个向量被用于存储下载的镜像大小。

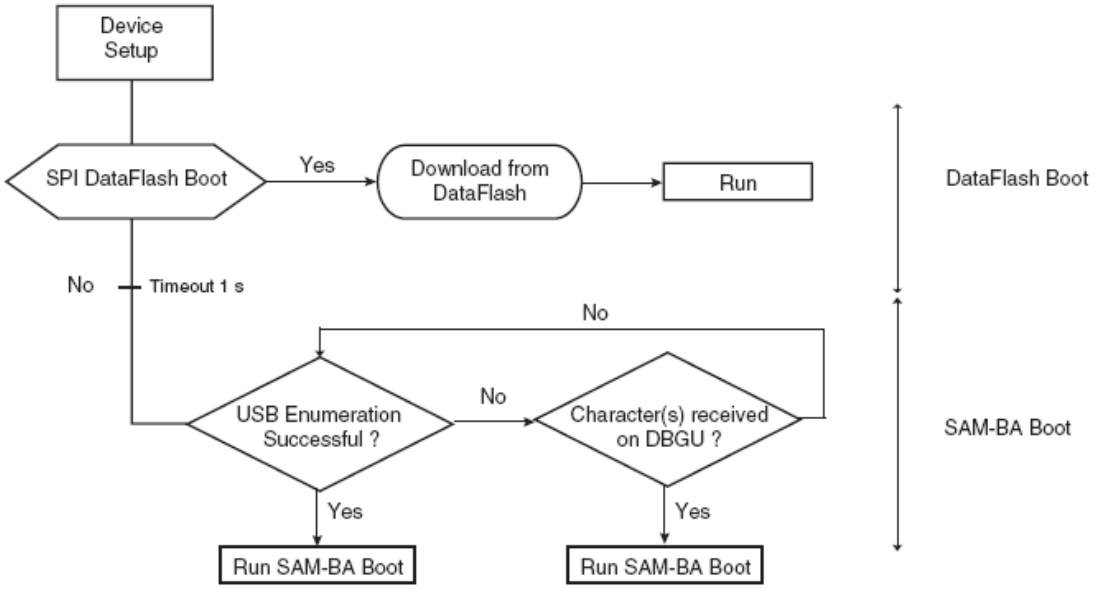
如果一个有效的序列被找到，代码将被下载到内部 SRAM。接着重映射，并跳到 SRAM 的首地址。

如果没有有效的 ARM 向量序列，则接着执行 SAM-BA。它等待 USB 设备或 DBGU 串行端口上的事件。

13.2 流程图

启动程序实现算法见图 13-1。

图 13-1 启动程序算法流程图



13.3 设备初始化

初始化按照如下描述的步骤进行：

1. 为 ARM 管理模式(SVC)设置堆栈
2. 主振荡器频率检测
3. C 变量初始化
4. PLL 设置：PLLB 被初始化并用来为 USB 设备产生一个 48MHz 时钟。一个置于电源管理控制器（PMC）中的寄存器决定主振荡器的频率，以及 PLLB 的正确参数。

表 13-1 定义了启动程序支持的晶振

表 13-1 由软件自动检测支持的晶振（MHz）

3.0	3.2768	3.6864	3.84	4.0
4.433619	4.608	4.9152	5.0	5.24288
6.0	6.144	6.4	6.5536	7.159090
7.3728	7.864320	8.0	9.8304	10.0
11.05920	12.0	12.288	13.56	14.31818
14.7456	16.0	17.734470	18.432	20.0

5． DBGU 串行端口的初始化（115200bps，8，N,1）

6． 禁用看门狗并使能用户复位

7． USB 设备端口的初始化

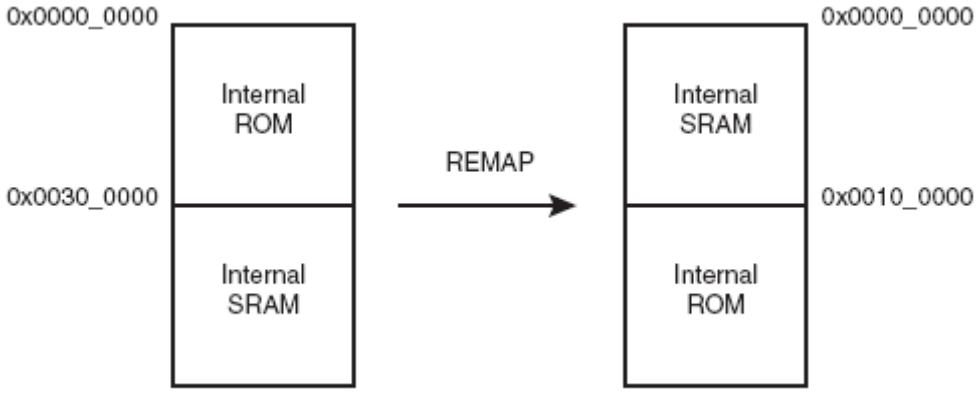
8． 跳至 DataFlash 启动序列

如果 DataFlash 启动失败：

9． 使能指令缓存

10． 跳至 SAM-BA 启动序列

图 13-2 下载完成后重映射方式



13.4 DataFlash 启动

DataFlash 启动程序在 SPI DataFlash 存储器中寻找一个有效的应用程序。有效的应用程序一旦被找到，该程序将被加载到内部 SRAM，在 remap 之后通过一个跳转到 0x00000000 的指令开始运行。此应用程序可能是应用程序代码或一个二级 bootloader。

所有的程序调用均基于 PC，不要使用绝对地址。

复位后，内部 ROM 的代码被映射到地址 0x0000_0000 和 0x0010_0000：

```
400000  ea000006  B       0x20            00  ea000006  B       0x20
400004  eafffffe  B       0x04            04  eafffffe  B       0x04
400008  ea00002f  B       _main           08  ea00002f  B       _main
40000c  eafffffe  B       0x0c            0c  eafffffe  B       0x0c
400010  eafffffe  B       0x10            10  eafffffe  B       0x10
400014  eafffffe  B       0x14            14  eafffffe  B       0x14
400018  eafffffe  B       0x18            18  eafffffe  B       0x18
```

13.4.1 有效镜像检测

DataFlash 启动软件通过分析开始的 28 位字节，对应 ARM 异常向量，来寻

找一个有效的应用程序。这些字节必须以 ARM 指令跳转或使用基于 PC 的相对地址装载 PC 来实现。

第六个向量，偏移量 0x14，包含下载镜像的大小。用户必须用自己的向量代替此向量。（见 87 页“ARM 第六个向量的结构”）。

图 13-3 LDR 操作码

31 28	27 24	23 20	19 16	15 12	11 0
1 1 1 0	1 1 I P	U 1 W 0	Rn	Rd	

图 13-4 分支(B)操作

31 28	27 24	23 0
1 1 1 0	1 0 1 0	偏移量（24 位）

无条件指令：0xE 用于 31 位到 28 位

用基于 PC 的相对地址装载 PC：

-Rn=Rd=PC=0xF

-I=1

-P=1

-U offset added(U==1) or subtracted(U==0)

-W=1

13.4.2 ARM 第六个向量的结构

ARM 第六个例外向量被用作存储 DataFlash 启动程序所需的信息。

此信息描述如下。

图 13-5 ARM 第六个向量的结构

31 0
以字节计算的下载的代码大小

13.4.2.1 举例

以下是有效向量的一个例子：

```
00    ea000006  B       0x20
04    eafffffe  B       0x04
08    ea00002f  B       _main
0c    eafffffe  B       0x0c
10    eafffffe  B       0x10
14    00001234  B       0x14      <- Code size = 4660 bytes
18    eafffffe  B       0x18
```

装载到 SRAM 的镜像大小包含在第六个 ARM 向量的位置。

因此用户必须用自己的应用程序提供正确的值代替此向量。

13.4.3 DataFlash 启动顺序

DataFlash 启动程序执行设备初始化，接着下载程序。

DataFlash 启动程序支持所有的 Atmel DataFlash 设备。表 13-2 总结了包括所有设备的用于 ARM 第六个向量的参数。

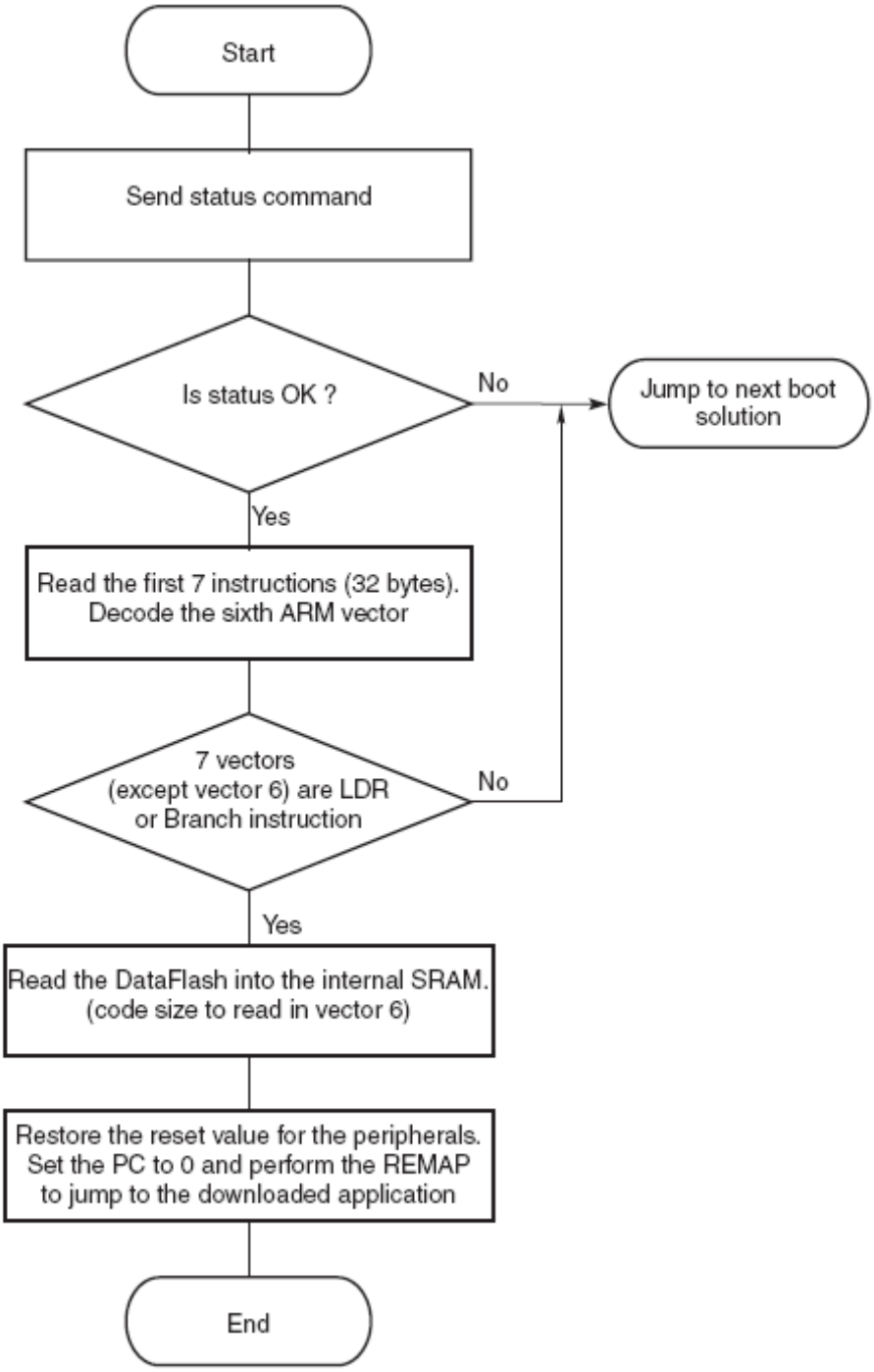
表 13-2 DataFlash 设备

设备	密度	页面大小（字节）	页数
AT45DB011B	1M 位	264	512
AT45DB021B	2 M 位	264	1024
AT45DB041B	4 M 位	264	2048
AT45DB081B	8 M 位	264	4096
AT45DB161B	16 M 位	528	4096
AT45DB321B	32 M 位	528	8192
AT45DB642B	64 M 位	1056	8192

DataFlash 有一状态寄存器，此寄存器决定访问设备时需要的所有参数。

DataFlash 启动程序被配置成和以后设计更新的 DataFlash 相兼容。

图 13-6　串行 DataFlash 下载



13.5　SAM-BA 启动

如果在 DataFlash 启动序列中未找到有效的 DataFlash 设备，SAM-BA 启动程序被执行。

SAM-BA 启动规则：

-检查 USB 设备的枚举事件

-检查 DBGU 上是否接收到字符

-一旦通信接口被识别，应用程序运行在一个等待不同命令的无限循环，这些命令见表 13-3

表 13-3 SAM-BA 启动可用的命令

命令	方式	自变量（s）	举例
O	写一个字节	Address,Value#	O200001,CA#
o	读一个字节	Address,#	o200001,#
H	写一个半字	Address, Value #	H200002,CAFÉ#
H	读一个半字	Address,#	h200002,#
W	写一个字	Address, Value #	W200000,CAFEDECA#
w	读一个字	Address,#	w200000,#
S	发送一个文件	Address,#	S200000,#
R	接收一个文件	Address,NbOfBytes#	R200000,1234#
G	到	Address,#	G200200#
V	显示版本	No argument	V#

- 写命令：写一个字节（O），一个半字（H）或一个字（W）到目标。
  -地址：十六进制地址。
  -值：字节，半字或字，用十六进制写。
  -输出：‘>’。
- 读命令：从目标读一个字节（o），一个半字（h）或一个字（w）。
  -地址：十六进制地址。
  -输出：十六进制读字节，半字或字，其后紧跟‘>’
- 发送一个文件（S）：发送一个文件到特定地址
  - 地址： 十六进制地址
  - NbOfBytes：用十六进制接收的字节数
  - 输出：‘>’
- Go（G）：跳至一个特定的地址并执行代码
  - 地址：十六进制地址跳转
  - 输出：‘>’
- 得到版本信息：返回 SAM-BA 启动版本
  - 输出：‘> ’

13.5.1 DBGU 串行端口

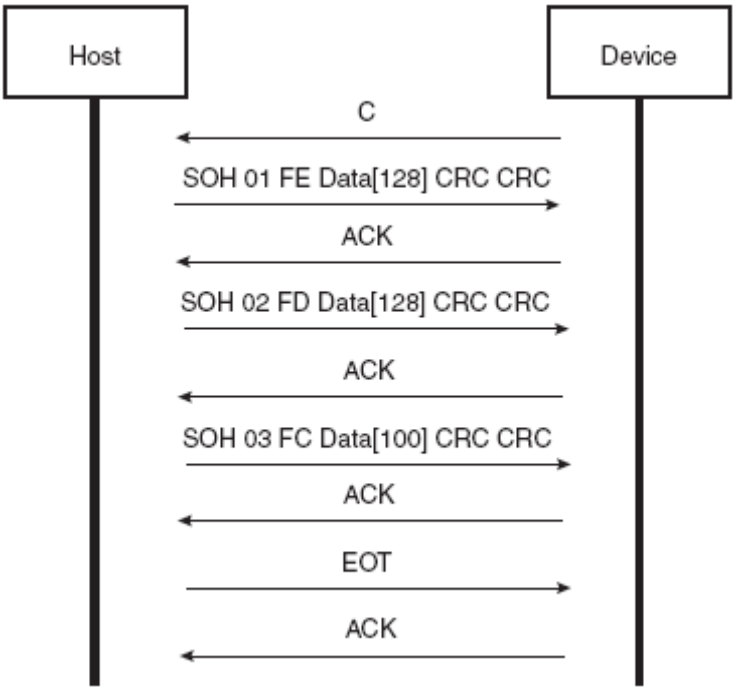
通过初始化为 115200，8，n，1 的 DBGU 串行端口来实现通信。
发送和接收文件命令采用 Xmodem 协议。任何执行此协议的终端可以被用于发送应用程序文件到目标。发送二进制文件的大小依赖于产品中的 SRAM 的大小。在所有的情形下，二进制文件的大小必须比 SRAM 的容量小，因为 Xmodem 协议需要一些 SRAM 存储空间才能工作。

13.5.2 Xmodem 协议

支持 Xmodem 协议的是 128 字节长的块。此协议用一个双字符 CRC-16 保证检测到一个最大位错误(maximum bit error)。
带 CRC 的 Xmodem 协议精确的提供发送器和接收器传送成功的报告。
每个传送块如:
<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

– <SOH> = 01 hex
– <blk #> = binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01)
- <255-blk #> = 1's complement of the blk#.
- <checksum> = 2 bytes CRC16\
图 13-7 展示了用此协议的一个传送。

图 13-7 Xmodem 传送举例



13.5.3 USB 设备端口

使用 USB 设备端口时必须提供一个 48MHz USB 时钟。在设备初始化过程早期已经通过对 PLLB 的配置完成。

设备采用 USB 通信设备类(CDC)，以利于已安装驱动程序的 PC 与 USB 设备交互。CDC 类在所有的 Window 版本中都被实现，从 Windows 98SE 到 Windows XP，CDC 文档可以在 www.usb.org 上得到。它描述了一种实现像 ISDN modem 和虚拟 COM 端口设备的方法。

USB 设备的厂商 ID 是 Atmel 的 ID：0x03EB。 产品 ID 是 0x6124。主机操作系统通过这些 ID 来正确安装合适的驱动。在 Windows 系统 ，INF 文件包含厂商 ID 和产品 ID 间的信息。

Atmel 提供一个 INF 例子可将设备看作一个新的串行端口，也提供另一个 SAM-BA 应用程序使用的客户驱动：atm6124.sys。更多细节，参考文档"USB 基本应用"(USB Basic Application)，文献号 6123。

13.5.3.1 枚举过程

USB 协议是一个主/从协议。主机通过控制端点发送请求到设备来开始枚举。设备处理在 USB 规范说明中定义的标准请求。

表 13-4 处理标准请求

请求	定义
GET_DESCRIPTOR	返回当前设备描述符
SET_ADDRESS	设置设备地址，并用于访问设备功能
SET_CONFIGURATION	设置设备配置
GET_CONFIGURATION	返回当前设备配置值
GET_STATUS	返回指定接受者的状态
SET_FEATURE	用于置位或使能特定的功能部件
CLEAR_FEATURE	用于清零或禁用一个特定的功能部件

设备还处理一些在 CDC 类中定义的类请求。

表 13-5 处理类请求

表

请求 1	定义
SET_LINE_CODING 3	配置 DTE 速率，停止位，奇偶校验位和字符位数。
GET_LINE_CODING 5	获得当前 DTE 速率，停止位，奇偶校验位和字符位数
SET_CONTROL_LINE_STATE 处	RS-232 信号用于告知 DCE 设备，DTE 设备现在存在。

不被处理的请求将被 STALL。

13.5.3.2 通信端点（endpoint）

有两个通信端点，而端点 0 被用作枚举。端网点 1 是一个 64 字节 Bulk(批量)输出端点，端点 2 是一个 64 字节 Bulk 输入端点。SAM-BA 启动命令被主机通过端点 1 发送。如果需要，信息被主机通过主机驱动分为若干有效载荷。如果命令需要一个响应，主机可以发出 IN 事务去获得响应。

13.6 硬件和软件限制

- 下载到 DataFlash 的代码必须低于 156K 字节。
- 代码总是从设备地址 0x0000_0000 下载到内部 SRAM 的地址 0x0000_0000（重映射后）.
- 下载的代码必须位置无关或连接在地址 0x0000_0000.
- DataFlash 必须被连接到 SPI 的 NPCS0。

SPI 驱动器用具有若干功能的 PIO 去和设备通信。当这些 PIO 被应用程序使用时必须谨慎。连接的设备可以在启动时被无意的驱动，在 SPI 输出引脚和连接的设备之间的电平冲突也可能出现。

为了确保正确的功能，推荐插入关键设备到其它的引脚。

表 13-6 包含引脚列表，这些引脚在启动程序执行期间被驱动。如果没有找到合适的启动程序，这些引脚将在小于 1 秒的启动序列期间被驱动。

当 SPCK 信号为 8MHz 时驱动的 DataFlash，下载 156K 字节代码所需的时间可被缩短到 200ms.

执行跳转至内部 SRAM 中的应用程序之前，所有在启动程序中使用到的 PIO 和设备被设置为它们在复位时的状态。

表 13-6 在启动程序执行期间被驱动的引脚

外设	引脚	PIO 口线
SPI0	MOSI	PIOA1
SPI0	MISO	PIOA0
SPI0	SPCK	PIOA2
SPI0	NPCS0	PIOA3
DBGU	DRXD	PIOA9
DBGU	DTXD	PIOA10