用“芯”捍卫安全

32位USBKEY芯片

Z32U256系列

用户手册

Version 1.5

目　录

第一部分 系统概述

第1章 概述

Z32U 系列 USBKey 芯片是 ZTEIC 面向高端 USBKey 市场应用，在基于国产 32 位 RISC 处理器的多功能安全处理平台基础上开发出的，具备高处理能力、高安全性、低功耗、低成本等特点。

该系列芯片可用于大容量 USB KEY、桌面加密机、桌面型 VPN 等设备上，可以实现的功能包括：

- 片上密钥管理（密钥生成、密钥存储、密钥更新等）；
- 片上签名及身份认证（可以支持 RSA、ECC(p 或 2n 域)等公钥算法）；
- 专用算法下载执行及高速率数据加解密（支持 DES/3DES 算法及各种专用算法）；
- 通过丰富的 GPIO 接口可以实现多种附加应用；

该系列芯片的逻辑框图，如图 1 所示：

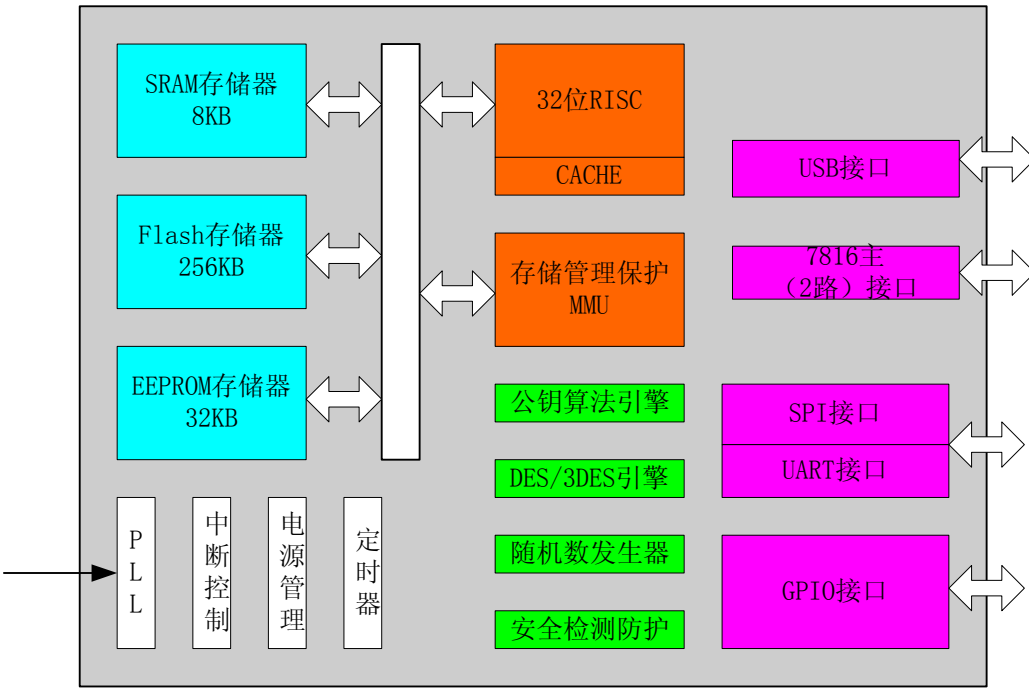


图 1 Z32U 系列芯片功能框图

（**注**：图 1 中 SPI、UART、7816 主接口请参见相应的产品型号）

产品系列：

产品型号	Feq	Flash/ PageSize	EEPROM (C&D)	1024bitRSA& ECC(p)/ 2048bitRSA &ECC(2n)	Interface	(SPI&UART &7816Master) /Flash slave &RAM	Package/ Embedded LDO
Z32L256D32U	24M	(256/1)KB	32KB	Yes/ No	USB2.0FS	Yes/No	LQFP44/Yes
Z32H256D32U	96M	(256/1)KB	32KB	Yes/ No	USB2.0FS	Yes/No	LQFP44/Yes
Z32H256D32SU	96M	(256/1)KB	32KB	Yes/ Yes	USB2.0FS	Yes/No	LQFP44/Yes

注：Z－ZTEIC; H/L － High Feq/Low Feq; D－Data Area; U－USB; S－Super Crypto; F－Nand Flash Interface

第2章 关键特性

2.1 处理器性能

- 专门定制的国产高安全核 Arca2sc
  - 32 位 RISC
  - 5 级流水线
  - 频率可变，主频最高可工作在 96MHz
- 高性能 CACHE
  - 1K 字节指令 CACHE
  - 1K 字节数据 CACHE
- 存储管理和保护单元（MMU）
  - 支持多级基于硬件的保护机制；
  - 支持可变页长管理模式，多级查找结构；
  - 可以配置为段式管理模式；
  - 面向应用的存储分区，硬件支持各个分区之间的安全隔离；
  - 支持虚拟存储空间管理；
  - 支持硬件安全访问控制，外围组件访问受控。

2.2 片上存储单元

- 32KB EEPROM 用于数据和程序的存储
  - 可进行单字节的读、擦除、写操作；
  - 可进行字/字节或多字节（最大 64 字节）的擦除、写操作；
  - 最少擦写次数 30 万次；
  - 擦写性能
    - 单字/字节擦写编程时间：1.3ms
    - 页擦除时间：4ms
  - 室温下数据保持时间最少 100 年；
- 128KB FLASH 用于程序、函数库、不常变动数据的存储
  - 页面大小为 1K 字节；
  - 最少擦写次数 2 万次；
  - 擦写性能
    - 单字节写时间：20us
    - 页擦除时间：4ms
  - 室温下数据保持时间最少 100 年。
- RAM：8KB

2.3 安全组件

- **Z32L256D32U 和 Z32H256D32U** 公钥算法引擎
  - 支持大数（1024 比特）模乘/模幂/乘法运算协处理
  - 支持多项式（511 比特）模乘/乘法运算协处理
  - 1024 比特 RSA 算法签名速度达到 6 次/秒@24MHz（不带 CRT）
  - 1024 比特 RSA 算法签名速度达到 14 次/秒@24MHz（带 CRT）
  - 1024 比特 RSA 算法签名速度达到 25 次/秒@96MHz（不带 CRT）
  - 1024 比特 RSA 算法签名速度达到 55 次/秒@96MHz（带 CRT）

  - 192 比特 ECC 算法（p 域）验证速度达到 18 次/秒@96MHz
  - 192 比特 ECC 算法（p 域） 验证速度达到 4 次/秒@24MHz
- **Z32H256D32SU** 公钥算法引擎
  - 支持大数（1024 比特）模乘/模幂/乘法运算协处理
  - 支持大数（2048 比特）模乘运算协处理
  - 支持多项式（511 比特）模乘/乘法运算协处理
  - 1024 比特 RSA 算法签名速度达到 25 次/秒@96MHz（不带 CRT）
  - 1024 比特 RSA 算法签名速度达到 55 次/秒@96MHz（带 CRT）
  - 192 比特 ECC 算法（p 域）验证速度达到 18 次/秒@96MHz
  - 361 比特 ECC 算法（2n 域）验证速度达到 4 次/秒@96MHz 以上
  - 193 比特 ECC 算法（2n 域）验证速度达到 16 次/秒@96MHz 以上
- DES/3DES 引擎
  - 支持 DES、3DES（2 KEY 和 3 KEY）算法的加密解密
  - 支持 EBC 模式和 CBC 模式的加密和解密
  - 优化的数据传送通道，3DES 加解密速度达到 3.5Mbps
- 真随机数发生器
  - 随机数发生码率为 64Kbps
  - 通过随机数测试国际标准 FIPS—140—2 测试
- 安全检测与防护单元
  - 高低电压检测
  - 高低频率检测
- 其他安全特性
  - 防止 DPA/SPA 攻击
  - 存储区域加密
  - 总线加扰
  - 时钟和复位信号脉冲过滤
  - 安全优化布线
  - 内部上电复位
  - 每一个芯片唯一序列号

2.4 通讯接口

- **Z32L256D32U、Z32H256D32U 和 Z32H256D32SU** 的 **USB** 接口
  - 支持 USB2.0 协议全速率 12Mbps 工作模式。
  - 支持四个端点，包括一个双向控制端点、一个 Bulk IN 端点、一个 Bulk Out 端点和一个中断端点。
- **Z32L256D32U、Z32H256D32U** 和 **Z32H256D32SU** 的 **ISO7816 Master**
  - 具备两路 ISO7816 主控制器，能够同时接入两路智能卡
  - 符合 ISO7816-3 标准，最高传送速率 310Kbps
- **Z32L256D32U、Z32H256D32U** 和 **Z32H256D32SU** 的 **UART** 接口
  - 支持的最高传送速率为 115.2Kbps
- **Z32L256D32U、Z32H256D32U** 和 **Z32H256D32SU** 的 **SPI** 接口
  - 作为 SPI 主设备，能够接入串行 FLASH 等设备
- **Z32L256D32U、Z32H256D32U** 和 **Z32H256D32SU** 10 多路 GPIO 接口，支持两个外部中断。

2.5 电气特性

- 内置振荡控制器和 PLL，可外部接 4MHz~12MHz 晶体

- 整个芯片的功耗小于 500mw（5V@96M 主频情况下）和 300mw（5V@24M 主频情况下）
- 支持低功耗模式
- 电源：2.7~5.5V
- ESD 保护：4000V 以上
- 符合：ISO7816-2 规范

2.6 开发环境

- 集成开发环境
  - 提供丰富的开发例程
  - 提供硬件开发板
- C 函数库（包括加密函数库等）

2.7 应用产品

- USBKey
- 桌面加密机
- 桌面型 VPN
- MMC/7816 读卡器

第二部分 主处理器单元

第3章 CPU 核

3.1 概述

Z32U256 系列芯片的 CPU 核是在方舟科技的 Arca2 CPU 核的基础上进行安全改造而成的 Arca2S CPU 核。Arca2S CPU 核是一款高性能低功耗微处理器内核，实现的是 Arca 第二版指令集。有关指令集的详细文档请参考《*Arca Instruction Set Architecture Reference Manual-V2*》。

Arca2S CPU 核采用五级流水和哈佛高速缓存结构。它集成了带 32 路全关联 TLB 和段/页式物理地址保护的存储管理单元和 1K 字节的指令和数据高速缓存，使其具有高性能、低功耗的特点，并适合复杂的多应用系统。

Arca2S CPU 核包含的 debug 模块提供强大的软硬件调试功能。硬件支持指令和数据断点。通过标准的 JTAG 接口，软件调试器可以直接与调试目标机连接而不需要其它的硬件接口。

Arca2S CPU 核还包括三类片内存储单元来满足应用的不同存储需求：FLASH、EEPROM 和 SRAM。并且对 FLASH 和 EEPROM 的内容加密以提高其安全性。

Arca2S CPU 核与系统其它功能模块的接口是标准的 AMBA AHB 总线。

3.1.1 模块图

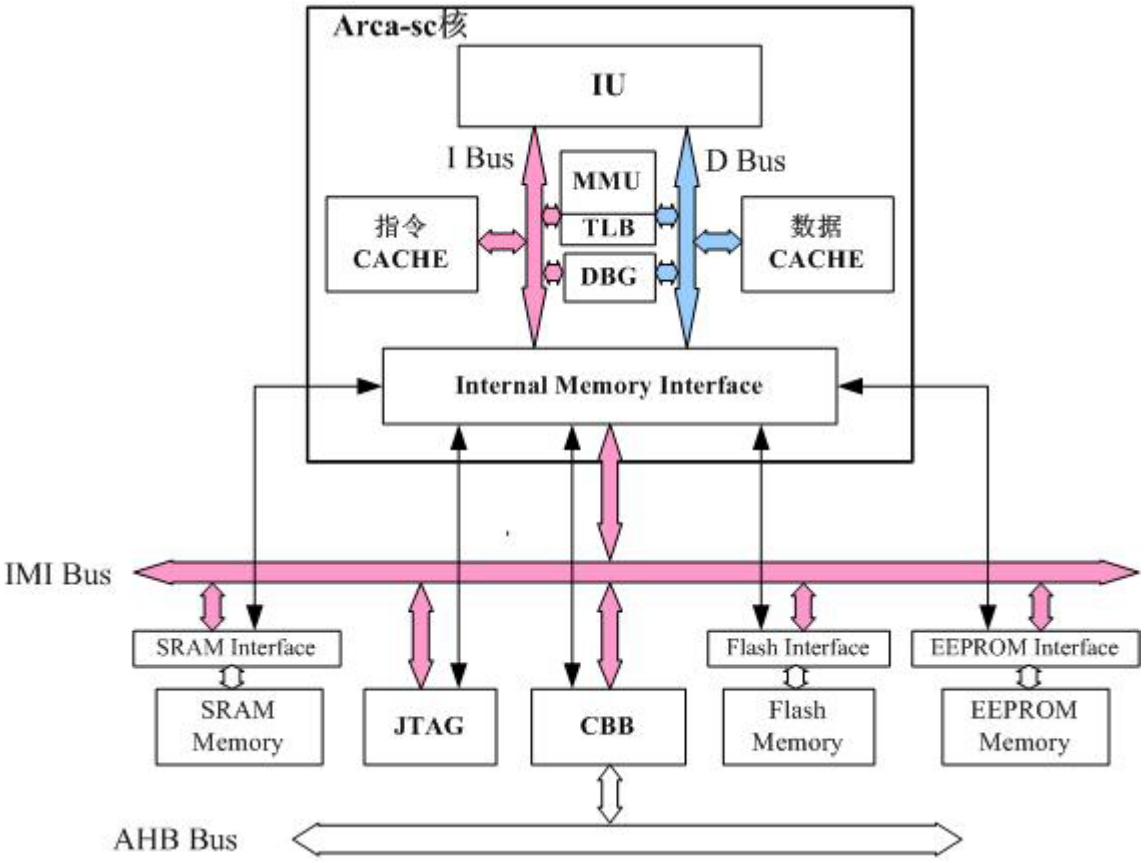


图 3-1 Arca2S 内核及存储接口部分结构图

3.1.2 特性

表 3-1 列举了 Arca2S CPU 核的部分关键特性。

表 3-1 Arca2S CPU 内核特性

项目	特性
Arca2S CPU	Arca2 体系结构 和 32 位 Arca 指令集 32 个 32 位的通用寄存器 五级流水线
存储管理单元 (MMU)	4G 字节地址空间，分成五部分来管理 32 路全相联 TLB，采用循环替换算法 同时支持四种页面大小：64 字节、256K 字节、1K 字节 和 4K 字节 支持 TLB 关键表项锁定

	把 32 位的虚地址转换成 32 位的物理地址 最多可支持 16 个进程同时运行 支持段/页式物理地址保护 支持厂商、超户和用户模式
数据高速缓存	1K 字节，虚地址查找 硬件解决别名问题 32 路全相联，采用循环替换算法 每行含 32 字节的数据 只支持写直达（write-through） 支持关键数据锁定
指令高速缓存	1K 字节，虚地址查找 32 路全相联，采用循环替换算法 每行含 32 字节，即 8 条指令 支持关键数据锁定
调试单元	通过 JTAG 接口与主机连接 支持调试指定的进程 两个或一个可部分屏蔽的指令地址断点 两个或一个可部分屏蔽的数据地址断点 一个存储数据断点 支持软件断点 支持主机异步断点 支持主机异步启动系统
FLASH 存储器	256K 字节的主存储块 256 字节的信息存储块 支持片上编程、页擦除和块擦除 内容加密
EEPROM 存储器	32K 字节的主存储块 64 字节的信息存储块 支持片上编程和擦除 内容加密
SRAM	8K 字节 工作在主频时钟，读写一拍完成

第4章 存储管理单元 MMU

4.1 概述

MMU 具有强大的管理功能，它能对访问存储器提供既严格又灵活的保护，同时更有效的利用存储空间。为了加速虚拟地址(VA)到物理地址（PA）的转换，MMU 引入了一个最新的数据指令共享的地址转换表缓存 UTLB(Union Translation Look-aside Buffer)。MMU 支持四种页面大小：64B，256B，1KB，4KB。MMU 还支持四种工作模式：基本超户，基本用户，高级超户，高级用户。

4.2 存储管理空间

固定保护区域

启动区域”：虚拟地址 0x00000000 ~ 0x00003FFF 和物理地址 0x00000000 ~ 0x00003FFF 的共计 16KB 的区域。

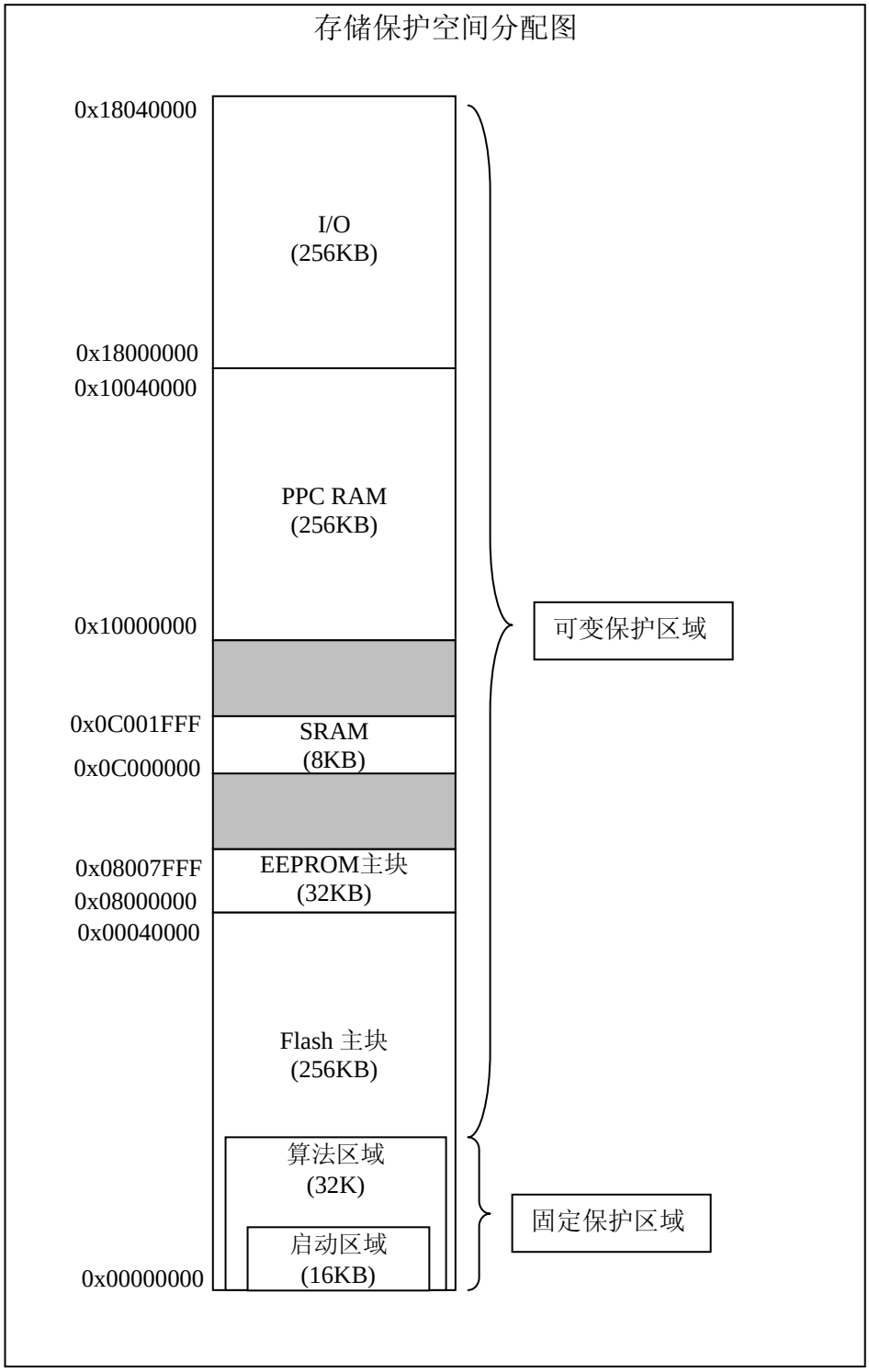
算法区域”：虚拟地址 0x00000000 ~ 0x00007FFF 和物理地址 0x00000000 ~ 0x00007FFF 的共计 32KB 的区域。注意：算法区域包含启动区域。

对启动和算法区域的区域的保护是固定的。仔细来讲，启动区域有本地可读，不能写和可执行的属性。算法区域有本地可读，可写和可执行的属性。启动区域可以读写算法区域，但是算法区域不能读写启动区域，其他区域均不能读写者两个区域，只能执行着两个区域的程序。这些属性将在任何模式下都存在。

可变保护区域

除了以上固定保护区域以外的其他区域，用户可以根据自己的需要来保护，只需要配置相应的保护寄存器就可以了。

MMU 具体使用方法请参考应用方案文档。


存储保护空间分配图
0x18040000
I/O
(256KB)
0x18000000
0x10040000
PPC RAM
(256KB)
0x10000000
可变保护区域
0x0C001FFF
SRAM
(8KB)
0x0C000000
0x08007FFF
EEPROM主块
(32KB)
0x08000000
0x00040000
Flash 主块
(256KB)
算法区域
(32K)
固定保护区域
启动区域
(16KB)
0x00000000

第5章 存储空间分配表

表 5-1 存储空间及寄存器映射基地址列表

类型	模块	虚拟地址	物理地址	注意
CPU-Core	Flash reg	0x18010800~18010FFF	0x D0800~0x D0FFF	
	Flashrom	0x00000000~0003FFFF	0x 00000~0x 3FFFF	
	EEPROM reg	0x18011200~180113FF	0x D1100~0x D12FF	
	EEPROM	0x08000000~0x08007FFF	0x 40000~0x 47FFF	
AHB-Bus	PPC RAM	0x10000000	0x 80000	256K space
	UDC	0x18000000	0x C0000	4K space
	PAE	0x18020000	0x E0000	4K space, 000E0000 for reg 000E0800 for ram
	SPI	0x1802E000	0x EE000	1K space
	UART	0x 1802E400	0x EE400	1K space
	PPC	0x 1802E800	0x EE800	1K space
	NFC	0x 1803FC00	0x FFC00	
	RNG	0x 1802F000	0x EF000	1K space
	SEC	0x 1802F400	0x EF400	1K space
	DES	0x 1802F800	0x EF800	1K space
APB-Bus	INTC	0x 1803F000	0x FF000	
	CGU/PMC	0x 1803F100	0x FF100	
	TMU	0x 1803F200	0x FF200	
	WDT	0x 1803F400	0x FF400	
	GPIO	0x 1803F500	0x FF500	
	SCC0	0x 1803FE00	0x FFE00	
	SCC1	0x 1803FF00	0x FFF00	

第6章 FLASH 存储器控制器（FCU）

6.1 概述

Flash 存储控制器（FCU）是由一个 32 为宽的嵌入式 Flash 存储阵列构成。嵌入式 Flash 是片上系统（SOC）应用中程序和数据的理想存储介质，它能够实现片上擦写而不需要额外的高电压。

FCU 支持典型的 Flash 操作而不需要软件的参与。软件可以直接使用“Load”指令来读取 Flash 存储器的内容。Flash 编程和页擦除直接使用两个“Store”指令来实现（第一次操作），或一个“Store”指令来实现（以后的操作）。块擦除可以通过三个“Store”指令外加软件定时器来实现。

在系统复位以后，CPU 能够直接读 Flash 存储器而不需要任何配置，因此系统能够从 Flash 存储器启动。另外，FCU 支持多字编程模式，既连续的对同一行的 32 个字编程。多字编程可以减少操作的时间，明显提高 Flash 的可靠性。

6.1.1 特性

Flash 存储器只支持 32 位（字）写操作。FCU 不支持半个字或字节写操作。

FCU 支持用户模式操作：读/编程/页擦除/块擦除，FCU 在一个周期只能接收一个读或写的命令。

支持两种模式的编程：单字和同一行的多字。在多字模式，FCU 能够支持对同一行的 32 个字进行编程。

读写时钟和时序可配置

支持工作频率：从 200kHz 到 400MHz。

内部 FCC 寄存器控制读和编程的时序。

在块擦除操作中使用软件延时。

当 FCU 进行编程/页擦除操作时，FCU 自动延迟访问 Flash 的读和写操作。

当 FCU 进行块擦除操作时，FCU 将忽略访问 Flash 的读和写操作。

寄存器位保护可避免无意的擦除或编程误操作。

6.2 Flash 存储器的操作特性

页擦除/块擦除将把当前页或整个存储区域的各个比特置为 1。

Flash 中的一个字在两次擦除中只能执行一次编程。

表 6-1 Flash 操作特性

操作	单位	总时间（最小）
读	字(32bit)	40ns
编程	字(32bit)	45us
	同一列的多字	20us x word_number + 25 uS
页擦除	page	20ms
块擦除	主存储块	200ms

6.3 寄存器配置

FCU 提供 32 位的控制寄存器，它们能够决定对 Flash 操作。这个寄存器映射到存储地址空间，数据能够通过 Store 或 Load 指令读写 FCU 寄存器。

Flash 写控制寄存器（Flash Write Control Register）记录写控制和状态比特，它们决定 Flash 存储器的写操作。

表 6-2 FCU 寄存器

名称	描述	R/W	上电初始值	位宽	地址
FWC	Flash 写控制寄存器	R/W	0x00000000	32 bit	0x18010840

6.3.1 Flash 写控制寄存器(FWC)

此寄存器包括 mass erase、page erase 以及 program 操作控制。

Bit:	31	30	29	28	……	17	16
Read:	ME_BUSY						
Write:							
Reset:	0	0	0	0	0	0	0

Bit:	15	……	4	3	2	1	0
Read:					MOD		WRE
Write:							
Reset:	0	0	0	0	0	0	0

- **Bit 30~3**：保留，在进行写操作时被忽略，在进行读操作时常为 0。
- **ME_BUSY**：当 Flash Memory 进行 mass erase 时此位为 1。否者，为 0。在写 FWC 寄存器时忽略。
- **WRE**：Flash Memory 写使能位。如果此比特为 1，写操作(program/page erase/mass erase)使能，如果为 0，写操作失效。
- **MOD**：写操作模式。当 WRE 为 1 时下面比特有效。

表 6-3 Flash 写控制编码

State	Bit 2 MD[1]	Bit 1 MD[0]	Bit 0 WEN
写禁止	X	X	0
单字编程使能	0	0	1
多字编程使能	0	1	1
页擦除使能	1	0	1
块擦除使能	1	1	1

6.4 地址映射

IU 通过 Store 和 Load 指令直接访问 FCU。主存储块在物理存储空间上占据 256K 字节空间。

内部的地址配置参见表 6-4。

表 6-4 FCU 内部地址配置

	虚地址	物理地址	尺寸(字，i.e.，4 字节)
主存储块	0x00000000~0x0003FFFF	0x00000000~0x0003FFFF	64K
Flash 写控制寄存器(FWC)	0x18010840	0xD0840	1

操作：

- IU 通过 Store 和 Load 指令直接访问 FCU 和 Flash memory。
- 用于访问半字和字节数据的 Store 指令将被 FCU 忽略。
- 用于访问信息块的 Store 指令将被 FCU 忽略。

6.4.1 FCU 寄存器访问

两个配置寄存器能够直接进行读写访问：

6.4.2 Flash Memory 读操作

可以使用 Load 指令直接访问 Flash Memory。

例如：从 Flash 中读一个字，

```
L32 R1, R2, 0                ; R2 contains the address of the excepte
d
                                ; word in Flash memory
```

6.4.3 编程（Program）操作

FCU 支持两种模式的编程：单字操作和同一列的多字编程。通过设置**错误！未找到引用源。**中的 FWC 寄存器来实现。

Flash Memory 中的任何一字两次擦写操作中只能编程一次。

6.4.3.1单字编程

设置完 FWC 以后，软件能够在单字模式下用 Store 指令来完成对一个字的编程。

步骤：

- 向 FWC 中的 bit[2:0]写 3b’001 来激活单字编程。
- 写 32 位数据到 Flash 中。

自动编程/擦写处理注意事项：

如果 IU 不从 FCU 地址空间取指或访问它，则 IU 能够连续地执行指令。否则，FCU 将停止 IU 地流水线，一直等到完成内部的编程。

同样的情况会在页擦除或块擦除的第一步操作中发生。

6.4.4 擦除操作

FCU 支持两种擦除操作：页擦除和块擦除，是由 FWC 寄存器控制，如表 1-4 所示。

在页擦除/块擦除时，Flash 内的所有比特都被置为 1。页擦除将擦写一个页，块擦除将只擦除主存储块。请查阅表 1-1 Flash memory 的组织结构。

页擦除处理时间大约是 20ms，而块擦除需要 200ms，硬件控制页擦除的时序，而块擦除的时序必须由软件控制。

页擦除操作过程：

- 将 FWC 的 bit[2:0]设置为 3b’101 来激活页擦除。
- 向要擦除的页的开始地址（bit[10:0]=0）写 32 位数据，这将激活页擦除处理过程。
  - 此步骤将花费 **20~40ms** 的时间。

块擦除操作过程：

- 系统复位以后设置 FCC。
- 将 FWC 的 bit[2:0]设置为 3b’111 来激活块擦除。
- 向主存储块的开始地址（bit[17:0]=0）写 32 位数据，
- FCU 在进行内部块擦除处理的第一步，在这期间 FCU 忽略任何对 Flash memory 的访问。同时软件将设置一个定时器，它不能超过 200ms，当定时器停止时，将 FWC 的 bit[2:0]设置为 3b’000 来结束内部处理块擦除，这将花 100~200us。

6.4.5 注意事项

除了比较明显的注意事项，在本文档中我们总结一些不明显的注意事项，而且必须在软件设计中遵循它们。

注意下列情况以避免错误操作：

半字或字节写命令将被忽略。

向信息块写将被忽略。

系统复位以后应当设置正确的时钟参数，否则编程/页擦除/块擦除命令将损坏 Flash 存储器内的数据。

在多字编程模式，软件要确保连续的数据的编程在同一行上。如果地址改变为新的一行，将启动新的多字编程处理过程。

当进行块擦除时，即，FWC.ME_BUSY 为 1,配置寄存器可以访问。然而，不要改变 FCC 的值。请参考 6.4.4 节，正确的配置 FWC。错误的写这些寄存器将使块擦除无法进行。

当进行块擦除时，即，FWC.ME_BUSY 为 1 Flash memory 不可以访问。写 Flash memory 的操作将被忽略，读它们的值是不确定的。

当进行块擦除时，一个异常或中断发生，向量表和处理者不应该确定发生在 FCU 的地址空间，而且处理者不应该 FCU 的地址空间。

当进行块擦除时，“Sleep” 指令不应该用。

多字编程的程序代码也不应该保存在 Flash memory 里，否则，它的性能就象单字编程一样。

第7章 EEPROM 控制单元 ECU

7.1 概述

EEPROM 控制单元(ECU)包括嵌入式 EEPROM 存储器，嵌入式 EEPROM 是单片应用程序和数据存储的理想解决方案。现场再编程不需要外部的高电压。

ECU 能够在不需要软件干预的情况下支持典型的 EEPROM 的各项操作。软件可以通过“Load/Store”指令直接访问 EEPROM。

系统复位以后，CPU 勿需配置便可以直接访问 EEPROM，但是读的速度较慢。在系统复位后，软件不能对 EEPROM 进行编程，因为时钟需要配置成 5MHZ。为了能够加速访问，软件必须要小心地配置 ETC。

7.2 特性

ECU 支持字节/半字/字读写操作。

ECU 的 EWC 控制决定是单字 ERASE_PROGRAM 还是页 ERASE_PROGRAM 操作。

支持 CPU 核工作时钟 5Mhz 到 200MHz。

当 ECU 正在执行编程操作时，ECU 将延时访问。

寄存器保护比特能够防止对 EEPROM 的无意擦除或编程。

EEPROM 编程时需要 5Mhz 的工作时钟。

在页的擦除+编程模式中，ECU 检查输入的地址是否在相同的一页中。

有中断标志可以指示发生了什么中断。

7.3 EEPROM 存储器操作特性

表 7-1 EEPROM 操作特性

操作	单位	总时间(最小)
读	单字 (32bit)	50 ns
单字擦除+编程	单字	1.2 ~1.3ms
页擦除+编程	页	3~5 ms

7.4 寄存器配置

ECU 提供两个 32 比特控制寄存器，它们能够决定 EEPROM 的操作。这些寄存器映射到存储地址空间中。

EEPROM 时钟控制寄存器（Timing Control Register）控制访问 EEPROM 时钟的配置。

EEPROM 写控制寄存器（Write Control Register）通过写控制位和状态位，来决定对于 EEPROM 的写操作。

表 7-2 ECU 寄存器

名称	全名	读/写	复位初始值	访问大小	地址
EWC	ECU Write Control Register	读/写	32'h0000_0000	32 bit	0x18011140
EIF	ECU Interrupt Flag Register	读/写	32'h0000_0000	32 bit	0x18011180

7.4.1 EEPROM 写控制寄存器(EWC)

这个寄存器有一些比特位控制编程操作。

Bit:	31	30	29	28	……	17	16
Read:	BUSY						
Write:							
Reset:	0	0	0	0	0	0	0

Bit:	15	……	4	3	2	1	0
Read:				ACIEN		MOD	WRE
Write:							
Reset:	0	0	0	0	0	0	0

- **Bit 30~4** ：保留位，写操作中忽略，读操作为全 0。
- **BUSY**：指示 EEPROM 存储器是否忙于擦除/编程。写 EWC 时操作将被忽略，当 EEPROM 忙于擦写/编程时读的值为 1，否则为 0。
- **ACIEN**：地址检查中断使能位，如果这一位置 1，页编程地址错误中断将使能，

当这一位为 0 时，页编程地址错误中断将失效。

- **WRE：** EEPROM 存储器写使能位。如果为 1，可以对 EEPROM 执行写（编程）操作。如果设置为 0，不允许对 EEPROM 执行写操作。
- **MOD：**写操作模式，当 WRE 为 1 时此比特值有效。

表 7-3 EWC 编码

状态	Bit 1 MOD	Bit 0 WRE
屏蔽写操作	X	0
使能单字擦除+编程	0	1
使能页擦除+编程	1	1

7.4.2 EEPROM 中断标志寄存器（EIF）

这个寄存器的比特位指示什么中断发生了。对这个寄存器写 0 将清掉这个寄存器和中断源，对这个寄存器写 1 将被忽略。

Bit:	31	30	29	28	……	17	16
Read:							
Write:							
Reset:	0	0	0	0	0	0	0

Bit:	15	……	4	3	2	1	0
Read:						AEIF	
Write:							
Reset:	0	0	0	0	0	0	0

- **Bit 31~2:** 保留位，写操作中忽略，读操作为全 0。
- **AEIF:** EEPROM 页编程地址错误中断标志。如果这一位为 1，表示页编程地址错误中断发生，如果这一位为 0，页编程地址错误中断没有发生。

7.5 地址映射

CPU 可以通过 Load 和 Store 指令直接访问 ECU。主存储块占据 32K 字节物理空间，ECU 控制寄存器占据 4K 字节的空间。

内部的地址配置见**表 7-4 ECU 内部地址配置**

表 7-4 ECU 内部地址配置

	虚拟地址	物理地址	大小 (Word,i.e., 4 bytes)
主存储块	0x08000000~0x08007FFF	0x40000~0x47FFF	8K
EWC	0x18011140	0xD1140	1
EIF	0x18011180	0xD1180	1

7.6 操作

CPU 通过 Load 和 Store 指令直接读/写 ECU 寄存器以及 EEPROM。

7.6.1 EEPROM 存储器读操作

使用 Load 指令直接读 EEPROM。

例如：从 EEPROM 中读出一个字。

```
L32 R1, [R2, 0]; R2 contains one address of EEPROM
```

7.6.2 为 EEPROM 产生一个 5Mhz 的时钟

EEPROM 需要一个[5Mhz +/- 10%]时钟来执行擦除/编程操作，通过配置 CGU_CFCR 寄存器来实现。参考 CGU 的说明。

// freq is the output frequency of PLL, so PLL should be configured reasonablly.

```
#define CGU_CFCR_ADDR 0x9803F100
void ecu_set_5m_clk(int freq)
{
  double period = 1000.0  / freq;
  double value = 200 / period + 0.5 - 1;
  int rate = value;
  int cfcr;
  cfcr = i_l32(CGU_CFCR_ADDR, 0x0);
```

```
  cfcr = (cfcr & (~0x3f00)) | ((rate & 0x3f) << 8);
  i_s32(cfcr, CGU_CFCR_ADDR, 0x0);
}
```

7.6.3 单字 Erase+Program 操作

设置完 ETC，EIF 以及 EWC 寄存器以后，软件能够通过 Store 指令实现字节/半字/字编程操作。

步骤：

- 为 EEPROM 产生 5Mhz 时钟。
- 基于当前的 ICLK 频率设置 ETC。
- 向 EWC 的 bit[1:0]写 2b'01 来激活单字 erase+program 操作；
- 向 EWC 的 bit[3:2] 写数据，使能/失效 EEPROM 坏死中断和 EEP ROM 页编程地址错误中断
- 向 EEPROM 内写一个字节/半字/字。
- 读 EWC 并检查 bit[31]，如果 bit[31]为 1，表明 EEPROM 正忙于编 程操作。
- 如果有中断发生，读 EIF 寄存器可以知道什么中断发生了。

自动编程/擦除的注意事项：

在编程期间，如果 CPU 不访问 EEPROM 则 CPU 能够继续执行指令，否则，ECU 将停止 CPU 的流水线，直到完成编程操作。

例如： 在 EEPROM 存储空间进行单字编程,当 ETC 设置正确后。

```
ORI R1, R0, 3; Do ERASE+PROGRAM
S32 R1, [R2, 0]; R2 contains the address of EWC
S32 R3, [R4, 0]; R4 contains one address of EEPROM, R3 contains data
```

7.6.4 多字的 Erase+Program

如果软件想编辑很多的字，最好在多字的模式下编程。

设置完 ETC，EIF 以及 EWC 寄存器以后，软件能够通过 Store 指令实现多字编程操作。

步骤：

- 为 EEPROM 产生 5Mhz 时钟。
- 基于当前的 ICLK 频率设置 ETC。
- 向 EWC 的 bit[1:0]写 2b’11 来激活多字 erase+program 操作；
- 向 EWC 的 bit[3:2] 写数据，使能/失效 EEPROM 坏死中断和 EEP ROM 页编程地址错误中断
- 向 EEPROM 内写多字。
- 向 EWC 的 bit[1:0]写 2b’00
- 读 EWC 并检查 bit[31]，如果 bit[31]为 1，表明 EEPROM 正忙于编 程操作。
- 如果有中断发生，读 EIF 寄存器可以知道什么中断发生了。

自动编程/擦除的注意事项：

在编程期间，如果 CPU 不访问 EEPROM 则 CPU 能够继续执行指令，否则，ECU 将停止 CPU 的流水线，直到完成编程操作。

第三部分 系统功能控制

第8章 中断控制器(INTC)

8.1 概述

中断控制器对每个不同的中断源分配了一个中断控制位。所有中断源可以通过 INTC 向 CPU 发送中断请求。CPU 响应中断请求，停止当前指令执行过程，跳转到中断服务例程中去。通过这种方式，设备可以使用中断方式获得 CPU 的服务。

8.1.1 特征

INTC 特征：

- 总共 23 个中断源。
- 独立的中断源屏蔽。
- 中断源寄存器和中断请求寄存器给用户足够信息。
- 所有的中断输入源基于电平触发，低有效或者上升沿有效。（外部沿触发中断通过 GPIO 模块转变为电平触发）。
- 所有寄存器通过 AHB 总线读写。
- 没有被屏蔽的中断可以将芯片从睡眠模式和暂停模式唤醒。

8.2 寄存器配置

下表列出了中断控制器的所有寄存器。所有这些寄存器都是 32 位，每一位控制表所示的中断源。

表 8-1 INTC 寄存器

名字	描述	R/W	初始值	虚拟地址	物理地址	位宽
ISR	中断源寄存器	R	0x00000000	0x 1803F000	0xFF000	32
IMR	中断屏蔽寄存器	R/W	0x00000000	0x 1803F004	0xFF004	32
IMSR	中断屏蔽置位寄存器	W	Undefined	0x 1803F008	0xFF008	32
IMCR	中断屏蔽清除寄存器	W	Undefined	0x1803F00C	0xFF00C	32
IPR	中断挂起寄存器	R	0x00000000	0x 1803F010	0xFF010	32
RISIFR	上升沿中断源标志寄存器	R/W	0x0000	0x 1803F014	0xFF014	16

注意：初始值定义为复位后寄存器值。

8.2.1 中断源寄存器(ISR)

中断源寄存器包括所有中断的状态．1 代表有响应的中断发生（在低电平），0 代表没有相应的中断(在高位)。本寄存器为只读。

Bit of ISR	描述	
0	相应的中断源未被激活	(初始值)
1	相应的中断源被激活	

8.2.2 中断屏蔽(IMR)

中断屏蔽寄存器用来屏蔽输入中断源并定义哪些活动资源可以允许生成中断请求，它的值可以通过写 IMSR 和 IMCR 来改变，也可以通过写自身改变。屏蔽的中断源对处理器不可见。

Bit of IMR	描述	
0	相应的中断未被屏蔽	(初始值)
1	相应中断被屏蔽	

8.2.3 中断屏蔽置位寄存器（IMSR）

本寄存器只写，用来设置 IMR(中断屏蔽寄存器)相应位。

Bit of IMSR	描述	
0	忽略	(初始值)
1	设置相应中断屏蔽位	

8.2.4 中断屏蔽清除寄存器（IMCR）

本寄存器只写，用来清除 IMR(中断屏蔽寄存器)相应位。

Bit of IMCR	描述	
0	忽略	(初始值)
1	将清除相应中断的屏蔽位	

8.2.5 中断挂起寄存器（IPR）

中断挂起寄存器包括屏蔽后的中断源状态。此寄存器只读。

Bit of IPR	描述	
0	相应的中断未被激活或被屏蔽	(初始值)
1	相应的中断被激活并且未被屏蔽	

8.2.6 IMR, IMSR, IMCR, ISR, IPR 位定义

表 8-2 中断源

Bit/Bits	中断源
Bit 31	---- 保留
Bit 30	---- SEC
Bit 29	---- 保留
Bit 28	---- UDC_RST
Bit 27	---- RSA
Bit 26	---- DES
Bit 25	---- RNG
Bit 24	---- TMU0
Bit 23	---- TMU1
Bit 22	----保留
Bit 21	---- 保留
Bit 20	---- 保留
Bit 19	---- IRQ0(GP18)
Bit 18	---- IRQ1(GP23)
Bit 17	---- 保留
Bit 16	---- 保留
Bit 15	---- 保留
Bit 14	---- 保留
Bit 13	---- EEPROM
Bit 12	---- UDC_INT
Bit 11	---- 保留
Bit 10	---- 保留
Bit 9	---- SCC0
Bit 8	---- SCC1
Bit 7	---- SPI
Bit 6	---- CPU-core

Bit 5	---- 保留
Bit 4	---- PPC
Bit 3	---- UART
Bit 2	---- 保留
Bit 1	---- 保留
Bit 0	---- 保留

注意：IMR、IMSR 和 IMCR 相应保留位可读可写。ISR 和 IPR 相应保留位只读，恒为 0。

8.2.7 上升沿中断源标志寄存器 (RISIFR)

RISIFR 是一个 16 位可读可写的寄存器.它的初始值是 H0000.这个寄存器包含检测到的中断源的上升沿的标志.软件可以清除位来清除相应的中断.

Bit:	15	14	13	12	11	10	9	8
读:								
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
读:		UART	PPC	SPI	RNG	DES	RSA	SEC
写:								
复位	0	0	0	0	0	0	0	0

注意:对位写 1 可以清除相应的中断源。写 0 却没有作用。位 15-7:保留，这些位读出均为 0.写数据将被忽略。

Bit n of RISIFR	描述	
0	中断源的上升沿没有被检测到.	(初始值)
1	中断源的上升沿有被检测到	

注意: n = 0~6

8.3 INTC 操作

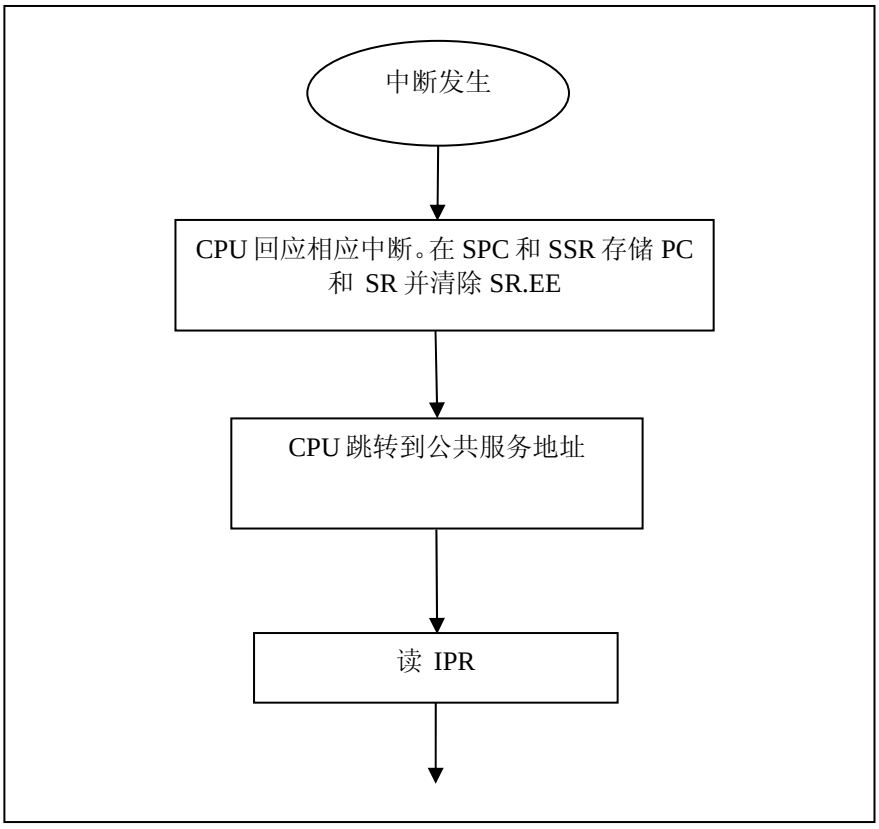
中断控制器提供了一个表示系统设备中断源的状态和软件中断屏蔽状态的接口。

当需要对多个未屏蔽中断进行优先级区分时，由软件负责根据 IPR 中中断挂起状态进行判断。在这个芯片中，未处理的中断有两级结构。软件需要根据各状态寄存器判断如何服务多个中断。

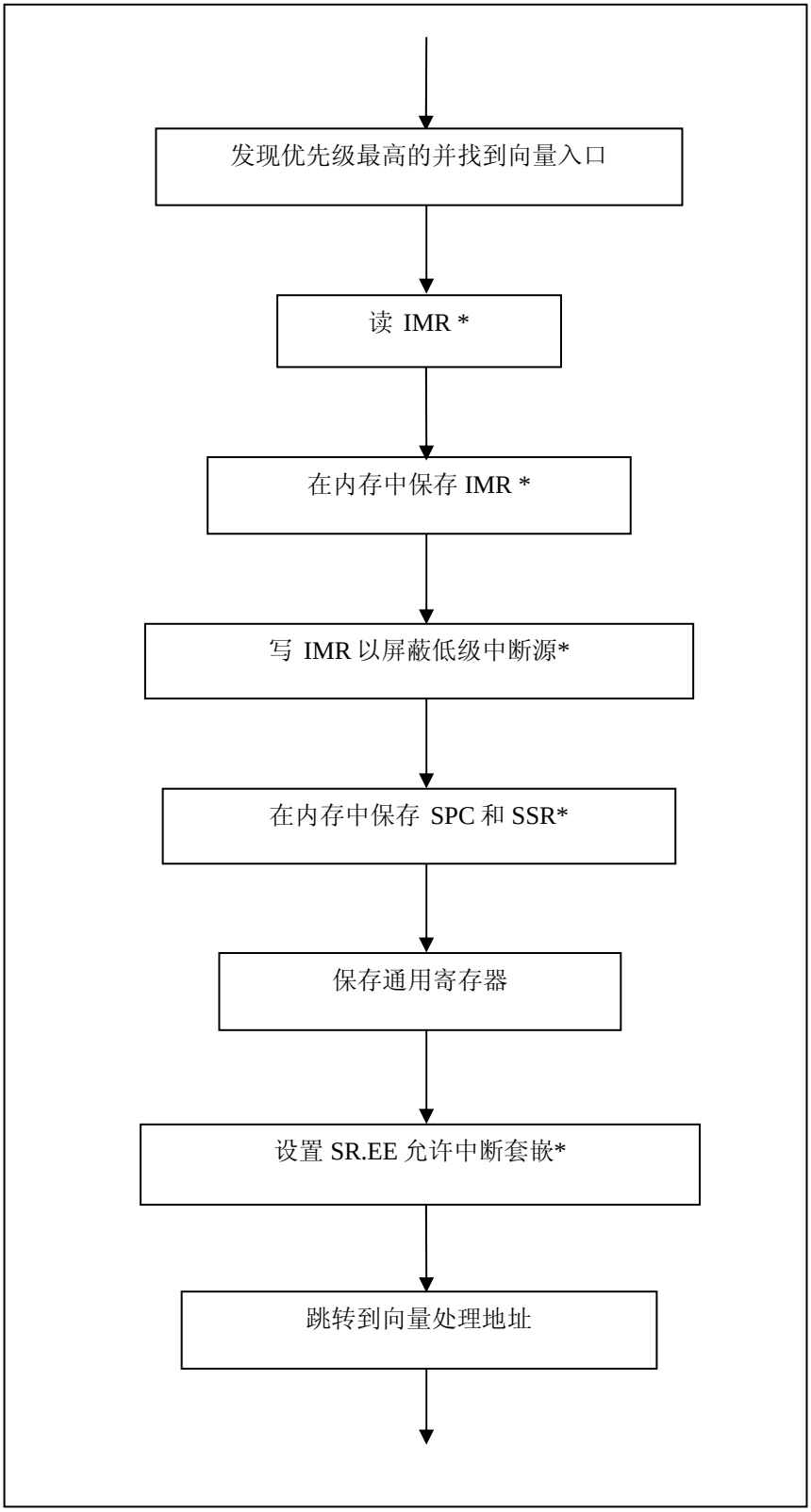
在中断处理例程中，需要首先清除被处理的中断源，可以通过查询 IPR 相应位来确定源已经被清除，同时软件需要在执行 RTE 指令前等足够的时间来保证。

所有的外部中断（连接到外部 GPIO）送到 INTC 中作为两个独立的 GPIO 中断，中断方式可以配置位电平触发或者沿触发。当执行 GPIO 中断例程时，软件可以清除 GPFR 相应位来清除沿触发中断，也可以读 GPDR 去检测挂起状态，并且通过写相应外部设备来清除中断源。

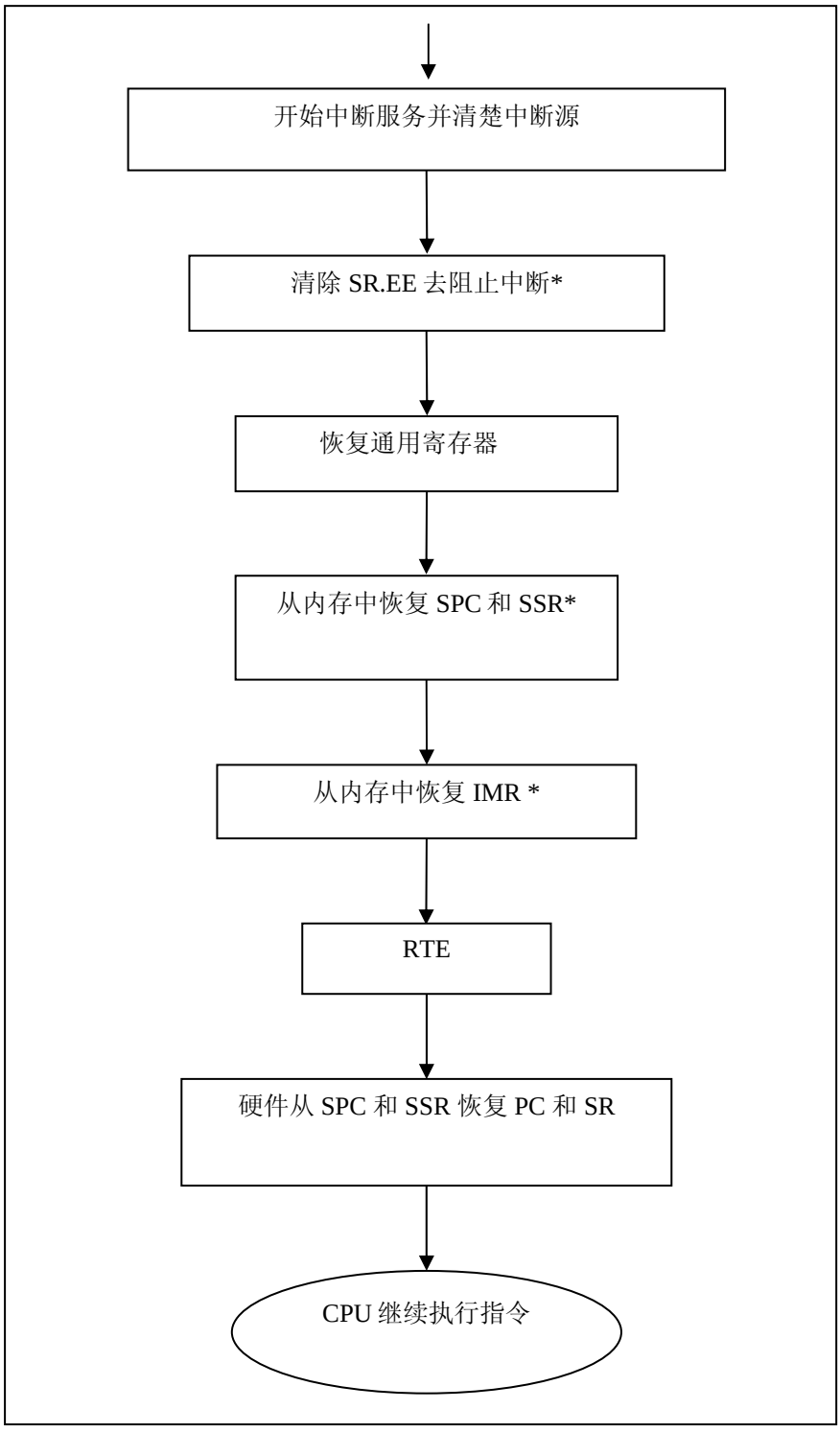
整个过程的流程图如下：



流程图：（继续）


发现优先级最高的并找到向量入口
读 IMR *
在内存中保存 IMR *
写 IMR 以屏蔽低级中断源*
在内存中保存 SPC 和 SSR*
保存通用寄存器
设置 SR.EE 允许中断套嵌*
跳转到向量处理地址

流程图：（继续）



注：当中断嵌套打开的时候这些步骤十分必要。

第9章 功率复位管理控制器（PMC）

9.1 概述

在低功耗模式，部分或全部处理器工作单元会被停止使用。这样能降低系统功耗。功率/复位管理控制器包含了低功耗模式控制和复位顺序控制功能。

9.1.1 低功耗模式和功能

芯片支持五种低功耗模式和功能：

正常模式

当处理器在复位后开始工作时，系统处于正常工作模式，所有时钟连续地运行。

睡眠模式

当 LPCR 的 LPM 位为 0 时，系统执行 SLEEP 指令后，便可进入睡眠模式。在睡眠模式中，提供给 CPU 核，DES，RSA，RNG 的时钟将被停止，直到有中断或复位发生。其他的片上模块时钟连续地运行。PLL 照常工作。

待命模式

当 LPCR 的 LPM 位为 1 时，系统执行 SLEEP 指令后，便可进入待命模式。在待命模式中，所有的时钟供给将被停止。PLL 被关闭。片上 100KHz 振荡器和外部 6MHz 振荡器或时钟输入也可以被停止。待命模式通过复位或中断而被取消。

模块停止功能

模块停止功能用来在系统某个模块不再使用时停止其工作。通过设置 LPCR 中 MSTP 一些相应位，可以停止特定功能模块。这些模块的时钟将被停止。模块停止功能通过复位或者清除 MSTP 一些相应位后被取消。

第10章 看门狗(WDT)

10.1 概述

WDT 提供跳出软件死循环或系统锁死的功能。WDT 一旦启动，软件必须周期性的对它的计数器执行清零操作，否则，WDT 将溢出产生复位信号。

10.1.1 特性

产生复位信号

16bit 计数器

计数器时钟使用分频时钟（4～20KHz），由外部输入时钟（1～5MHz）分频而来

10.2 寄存器配置

本节描述的是 WDT 的寄存器，下表列出了寄存器的定义。

表 10-1 WDT 寄存器

寄存器名	描述	R/W	初始值	地址错误！未找到引用源。	位宽
WTCNT	看门狗计数器	R/W	0x00000000	0x1803F404	读: 16 写: 16
WTCSR	看门狗控制/状态寄存器	R/W	0x00	0x1803F400	读: 8 写: 8

这里的地址是虚地址；对应的物理地址为 000FF4xx

10.2.1 WDT 控制/状态寄存器(WTCSR)

WTCSR 是 8bit 读/写寄存器，上电复位初始值为 H’00，也可由 WDT 复位成初始值。

位	7	6	5	4	3	2	1	0
读				START				
写:								
初始值	0	0	0	0	0	0	0	0

- **Bit 0~3, 5~7**: 保留位。 读出为 0， 写被忽略。
- **START:** 定时器计数启动/停止位

Bit 4: START	描述	
0	定时器停止	(初始值)
1	定时器启动运行	

10.2.2 计数器(WTCNT)

WDT 是工作在由（1～5MHz）分频时钟下的 16bit 可读/写计数器。当计数器溢出，产生上电复位。复位初始值为 H’0000。

位	16 ~ 0
读	WTCNT
写	
初始值	H’0000

10.3 使用事项

10.3.1 WDT 功能

- 设置 WTCSR 的 START 位为 1，计数器开始计数。
- 软件应当周期性地将 WTCNT 清零，防止计数器溢出。
- 如果计数器溢出，将产生复位。

第11章 定时器单元(TMU)

11.1 总述

定时器单元提供了两个独立的定时器，每个定时器可以配置成使用独立的时钟源进行计数。

11.1.1 技术指标

关键的技术指标如下：

- 提供两个独立的定时器。
- 每个定时器包括一个 32 比特递减计数器和一个 32 比特常数寄存器。
- 每个定时器提供自动重载功能。
- 当递减计数器下溢时（H‘00000000→H‘FFFFFFFF），自动触发中断。
- 递减计数器的工作时钟可以从以下 5 个输入时钟选择：EXTAL（外部时钟输入）、ϕ/4, ϕ/16, ϕ/64 和ϕ/256。（ϕ为片上设备时钟）

11.2 寄存器配置

本节描述 TMU 所涉及到的寄存器。其定义如所示。详细的功能定义将在后面描述。

表 11-1 TMU 寄存器

名称	描述	R/W	初始值	地址	位宽
TER	定时器使能寄存器	R/W	0x 00	0x1803F200	8
TRDR0	定时器重载数据寄存器 0	R/W	0x FFFFFFFF	0x 1803F204	32
TCNT0	定时器计数器 0	R/W	0x FFFFFFFF	0x 1803F208	32
TCSR0	定时器控制/状态寄存器 0	R/W	0x 0000	0x 1803F20C	16
TRDR1	定时器重载数据寄存器 1	R/W	0x FFFFFFFF	0x 1803F210	32
TCNT1	定时器计数器 1	R/W	0x FFFFFFFF	0x 1803F214	32
TCSR1	定时器控制/状态寄存器 1	R/W	0x 0000	0x1803F218	16
TCRB0	定时器计数器读缓冲 0	R	Undefined	0x1803F228	32
TCRB1	定时器计数器读缓冲 1	R	Undefined	0x1803F22C	32

11.2.1 定时器使能寄存器（TER）

TRE 是 8 比特可读写寄存器，它可用来控制定时器 0 和 1 的计数器（TCNT）运行与否。TER 在复位后初始值为 H‘00。

Bit:	7	6	5	4	3	2	1	0
Read:	—	—	—	—	—	—	TE1	TE0
Write:								
Reset:	0	0	0	0	0	0	0	0

- **Bit 2-7**: 保留比特。读值为 0，写动作将被忽略。
- **TE0 ：** 定时器 0 使能。停止或者运行定时器 0。

Bit 0: TE0	描述	
0	停止定时器 0 计数	(Initial value)
1	启动定时器 0 计数	

- **TE1 ：** 定时器 1 使能。停止或者运行定时器 1。

Bit 1: TE1	描述	
0	停止定时器 1 计数	(Initial value)
1	启动定时器 1 计数	

11.2.2 定时器控制/状态寄存器（TCSR）

定时器控制/状态寄存器（TCSR）是 16 比特可读写寄存器，用来进行时钟选择和中断控制。每个定时器有其独立控制/状态寄存器。复位后初始化值为 H‘00。

Bit:	15	14	13	12	11	10	9	8
Read:	—	—	—	—	—	—	—	—
Write:								
Reset:	0	0	0	0	0	0	0	0
Bit:	7	6	5	4	3	2	1	0
Read:	BUSY	UF	UIE	—	—	CKS2	CKS1	CKS0
Write:								
Reset:	0	0	0	0	0	0	0	0

- **Bit 3, 4, 10-15**: 保留比特。读值为 0，写动作将被忽略。
- **CKS0-CKS2:** 时钟选择。这些比特选择 TCNT 计数用时钟，在定时器运行时

不能改变。

Bit 2: CKS2	Bit 1: CKS1	Bit 0: CKS0	描述	
0	0	0	内部时钟 PCLK/4	(Initial value)
0	0	1	内部时钟 PCLK/16	
0	1	0	内部时钟 PCLK/64	
0	1	1	内部时钟 PCLK/256	
1	0	0	Reserved	
1	0	1	外部晶体 EXTAL	
1	1	0	Reserved	
1	1	1	Reserved	

- **UIE:** 下溢中断使能。当下溢指示（UF）置为 1 时控制中断发生。

Bit 5: UIE	描述	
0	下溢中断去能	(Initial value)
1	下溢中断使能	

- **UF:** 下溢指示。时钟 TCNT 是否发生下溢。对此位写 1 被忽略。

Bit 6: UF	描述	
0	TCNT 没有发生下溢 清除条件：向 UF 中写 0	(Initial value)
1	TCNT 发生下溢 置位条件：当 TCNT 下溢	

- **BUSY**: 忙指示。当定时器开始切换时钟或者同步 TCNT 读写操作时由硬件置位。当时钟切换或者同步完成后由硬件清零。

11.2.3 定时器重载数据寄存器(TRDR)

TRDR 是 32 比特读写寄存器，用来存储重载数据。计数器（TCNT）从其初始值开始递减计数，当其发生下溢时 TRDR 值被重载到 TCNT 中去，并重新开始递减计数，此过程被称为重载过程。每个定时器有其独自的 TRDR，具有相同的配置。复位后初始化值为 H'FFFFFFFF。

Bit:	31	……	4	3	2	1	0
Read:	TRDR						
Write:							
Reset:	FFFF_FFFF						

当定时器运行时禁止改变 TRDR 值。

11.2.4 定时器计数器（TCNT）

TCNT 是 32 比特读写寄存器。当 TCNT 下溢时，相应 TCSR 寄存器中下溢指示（UF）置位，TRDR 值被重载到 TCNT 中去，TCNT 重新开始计数。每个定时器有其独自的 TCNT，具有相同的配置。复位后初始化值为 H'FFFFFFFF。

Bit:	31	……	4	3	2	1	0
Read:	TCNT						
Write:							
Reset:	FFFF_FFFF						

11.2.5 定时器计数器读缓冲器（TCRB）

定时器计数器读缓冲（TCRB）是 32 比特只读寄存器，用来锁存 TCNT 读数据。当 EXTAL 选为计数器时钟时，由于异步时钟域的问题，对 TCNT 进行读操作将返回不正确的值。但是读 TCNT 操作将触发一个内部读时序，此时 BUSY 位被自动置位，当正确的 TCNT 值被锁存到 TCRB 中后，BUSY 被自动清楚。软件需要自动监视 BUSY 位，以便读出正确的 TCRB 值。

Bit:	31	……	4	3	2	1	0
Read:	TCRB						
Write:							
Reset:	—						

11.3 TMU 操作

11.3.1 基本功能

计数器操作：设置定时器使能寄存器 TER 可以启动相应的定时器计数器（TCNT）从初始值开始递减计数。当 TCNT 发生下溢时，相应的控制/状态寄存器中 UF 指示被置位。此时如果 UIE 位打开，则将引发一个中断。然后 TCNT 重载 TRDR 值，继续其递减计数。

计数器操作依照如下规则：

- 如果定时器处在运行状态，清除 TER 中相应 TE 位将停止定时器运行。

- 通过 TCSR 寄存器中 CKS2－CKS0 位选中计数器时钟。
- 使用 TCSR 中 UIE 比位控制是否产生 TCNT 下溢中断。
- 设置 TRDR 值控制计数时间。（具体时钟数为设置值加 1）
- 设置 TCNT 初始值。
- 等待 BUSY 位被清除。
- 设置 TER 中相应 TE 位启动草纸。

计数器操作具体图示如下：

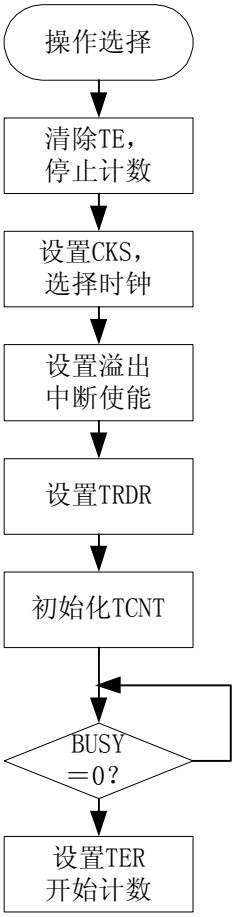


图 11-1　TMU 操作

第四部分 外部接口控制

第12章 USB 设备控制器(UDC)

12.1 概述

USB 设备控制器(USB Device Controller - UDC)提供 USB 功能设备与 USB 主机之间的接口。

对 USB 协议和操作的完全描述，请参考 Universal Serial Bus Specification, Revision 1.1

12.1.1 特性

UDC 具有如下的一些特性:

兼容 USB1.1 协议

支持全速率(12Mbps)的发送速率

支持硬件处理 USB Specification 中 Chapter9 的部分标准请求

支持悬挂/恢复以及远端唤醒逻辑

支持一个配置和两个具有一个可替换设置的接口

支持 4 个物理端点(端点 0，1，2，3)和 5 个逻辑端点(IN 端点 0, 1 ,3 OUT 端点 0, 2)

支持控制传输，批量传输和中断传输

- 缺省情况下，端点 0 在复位或者挂起后，只用于配置 UDC 的控制传输通讯。
- 端点 1 用于执行批量传输的 IN 数据处理。
- 端点 2 用于执行批量传输的 OUT 数据处理。
- 端点 3 用于执行中断传输的 IN 数据处理。

注:由于 UDC 是一个全速率 USB 设备控制器，USB 设备时钟为 12Mhz，故 AHB 总线时钟必须大于或等于 18Mhz，否则 UDC 无法正常工作。

12.2 寄存器配置

表 12-1 UDC 寄存器

寄存器名	描述	读/写	初始值	地址	宽度
EP0InCR	IN 端点 0 控制寄存器	读/写	0X00000000	0X18000000 0X000C0000	32
EP0InSR	IN 端点 0 状态寄存器	读/写	0X00000000[1]	0X18000004 0X000C0004	32
EP0InBSR	IN 端点 0 缓存大小寄存器	读/写	0X00000000[2]	0X18000008 0X000C0008	32
EP0InMPSR	IN 端点 0 最大包长寄存器	读/写	0X00000000	0X1800000C 0X000C000C	32
EP0InDesR	IN 端点 0 描述符寄存器	读/写	0X00000000	0X18000014 0X000C0014	32
EP1InCR	IN 端点 1 控制寄存器	读/写	0X00000000	0X18000020 0X000C0020	32
EP1InSR	IN 端点 1 状态寄存器	读/写	0X00000000[1]	0X18000024 0X000C0024	32
EP1InBSR	IN 端点 1 缓存大小寄存器	读/写	0X00000000[2]	0X18000028 0X000C0028	32
EP1InMPSR	IN 端点 1 最大包长寄存器	读/写	0X00000000	0X1800002C 0X000C002C	32
EP1InDesR	IN 端点 1 描述符寄存器	读/写	0X00000000	0X18000034 0X000C0034	32
EP0OutCR	OUT 端点 0 控制寄存器	读/写	0X00000000	0X18000200 0X000C0200	32
EP0OutSR	OUT 端点 0 状态寄存器	读/写	0X00000000[1]	0X18000204 0X000C0204	32
EP0OutPFNR	OUT 端点 0 包的帧数寄存器	读/写	0X00000000[2]	0X18000208 0X000C0208	32
EP0OutMPSR	OUT 端点 0 最大包长寄存器	读/写	0X00000000	0X1800020C 0X000C020C	32
EP0OutSBPR	OUT 端点 0 Setup 缓存指针	读/写	0X00000000	0X18000210 0X000C0210	32
EP0OutDesR	OUT 端点 0 描述符寄存器	读/写	0X00000000	0X18000214 0X000C0214	32

EP2OutCR	OUT 端点 2 控制寄存器	读/写	0X00000000	0X18000240 0X000C0240	32
EP2OutSR	OUT 端点 2 状态寄存器	读/写	0X00000000[1]	0X18000244 0X000C0244	32
EP2OutPFNR	OUT 端点 2 包的帧数寄存器	读/写	0X00000000[2]	0X18000248 0X000C0248	32
EP2OutMPSR	OUT 端点 2 最大包长寄存器	读/写	0X00000000	0X1800024C 0X000C024c	32
EP2OutDesR	OUT 端点 2 描述符寄存器	读/写	0X00000000	0X18000254 0X000C0254	32
DevCFGR	设备配置寄存器	读/写	0X00000000	0X18000400 0X000C0400	32
DevCR	设备控制寄存器	读/写	0X00000000	0X18000404 0X000C0404	32
DevSR	设备状态寄存器	读/写	0X00000000	0X18000408 0X000C0408	32
DevIntR	设备中断寄存器	读/写	0X00000000	0X1800040C 0X000C040c	32
DevIntMR	设备中断屏蔽寄存器	读/写	0X00000000	0X18000410 0X000C0410	32
EPIntR	端点中断寄存器	读/写	0X00000000	0X18000414 0X000C0414	32
EPIntMR	端点中断屏蔽寄存器	读/写	0X00000000	0X18000418 0X000C0418	32
EP0InfR	端点 0 信息寄存器	读/写	0X00000000	0X18000504 0X000C0504	32
EP1InfR	端点 1 信息寄存器	读/写	0X00000000	0X18000508 0X000C0508	32
EP2InfR	端点 2 信息寄存器	读/写	0X00000000	0X1800050C 0X000C050C	32
EP3InfR	端点 3 信息寄存器	读/写	0X00000000	0X18000510 0X000C0510	32
RXFIFO		读	0Xxxxxxxxx	0X18000800 0X000C0800	32
TXFIFO EP0		写	0Xxxxxxxxx	0X18000880 0X000C0880	32

TXFIFO EP1		写	0Xxxxxxxxx	0X180008A0 0X000C08A0	32
TXFIFO EP3		写	0Xxxxxxxxx	0X180008E0 0X000C08E0	32

注: 上述所有寄存器只能以 32 位方式存取。上表中的地址包括软件使用的虚拟地址 (0X18000 xxx) 以及硬件实现和软件映射的物理地址(0X000C0xxx)。

- 向 TXFIFO 写入数据后为了确认发送数据，写 0X1800041C 一次，UDC 然后开始发送。
- 接收一个长度为 0 的包后为了确认 0 长度的包的接受，读 0X1800041C 一次。
- 发送一个长度为 0 的包，首先写 0X18000420 一次，然后写 TXFIFO EP0 或 TXFIFO EP1 或 TXFIFO EP3 一次，并写 0X1800041C 一次，然后 UDC 将向 USB 总线发送一个长度为 0 的包。

12.2.1 设备配置寄存器(DevCFGR)

设备配置寄存器用于配置设备。当接收到 USB 主机的设置配置请求时设置这个寄存器；否则这个寄存器只在初始配置时设置。上电后这个寄存器的初始值为 0X00000003。

Bit:	31							16
Read:	……							
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
Read:					SP	RW	SPD	
Write:								
Reset:	0	0	0	0	0	0	1	1

- **Bit 31 ~ 3:** 保留位。读这些位得到 0，对这些位只能写 0。
- **自供电(Self-Powered—SP):** 当这个位设置时表示设备是自己供电的。我们不支持自供电，所以这位只读，为 0。
- 远端唤醒(Remote Wakeup—RW): 当这个位设为 1 时表示设备具有远端唤醒功能。

Bit 2: RW	描述	
0	不能远端唤醒	(初始值)
1	可以远端唤醒	

- **速率(SPD1-0):** 这两个只读位表示设备支持的速率：11 表示全速率。

12.2.2 设备控制寄存器(DevCR)

UDC 工作时，在配置后任何时候都可以设置设备控制寄存器以控制设备。上电后寄存器的初始值为 0x00006000。

Bit:	31	30	29	28	27	26	25	24
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	23	22	21	20	19	18	17	16
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	15	14	13	12	11	10	9	8
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
Read:							TF	RES
Write:								
Reset:	0	0	0	0	0	0	0	0

- **Bit 31~2**: 保留位。读这些位得到 0，对这些位只能写 0。

- **发送 FIFO 清空(Transmit FIFO Flush—TF):** 当这个位被设置时表示当检测到一个 USB 复位后所有发送 FIFO 应该被清空。

Bit 1: TF	描述	
0	发送 FIFO 没有清空	(初始值)
1	发送 FIFO 清空	

- **恢复(Resume -- RES):** 当这个位被设置时表示系统在 USB 上将处理一个

恢复信令。

Bit 0: RES	描述	
0	USB 没有恢复	(初始值)
1	USB 上产生恢复	

注: 用户在设置 RES 位之前 USB 主机必须发送 set_feature 请求来激活 UDC DEVICE_REMOTE_WAKEUP，否则远端唤醒不会发生。

12.2.3 设备状态寄存器(DevSR)

设备状态寄存器反映了设备的状态信息，这些状态信息在一些中断服务程序中要用到。这个寄存器只读。上电后寄存器的初始值为 0X00000000。

Bit:	31	……	……	……	……	18	17	16
读:								
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	15	14	13	12	11	10	9	8
读:		ENUM SPD		SUSP	ALT			
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
读:	INTF				CFG			
写:								
复位:	0	0	0	0	0	0	0	0

- **Bit 31~15:保留位。**读这些位得到 0，对这些位只能写 0。

- **枚举的速度(Enumerated Speed—ENUM SPD 1-0):** 2'b11 为全速率。

- **悬挂状态(Suspend Status—SUSP):** 这个位表示悬挂状态，当 USB 上检测到悬挂条件时设置这个位。

Bit 12: SUSP	描述	
0	没有检测到悬挂	(初始值)
1	检测到悬挂	

- **可改变设置(Alternate Setting—ALT 3-0):** 这 4 位表示接口转换的可改变设置。
- **接口号(Interface Number—INTF 3-0):** 这 4 位表示使用 *SetInterface* 请求的接口设置。
- **配置号(Configuration Number—CFG 3-0):** This 这 4 位表示使用 *SetConfiguration* 请求的配置设置。

12.2.4 设备中断寄存器(DevIntR)

当系统事件发生时，设置设备中断。软件使用这些中断做一些系统级决定。上电复位后寄存器的初始值是 0X00000000。

Bit:	31	……	……	……	……	……	……	8
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
Read:	URSM		SOF	US	UR		SI	SC
Write:								
Reset:	0	0	0	0	0	0	0	0

- **Bit 31－8,6,2:** 保留位。读这些位得到 0，对这些位只能写 0。

- **USB 恢复（RUSM）：**当 USB 恢复，这一位将被置 1。

Bit 7: URSM	描述	
0	没有检测到恢复	(初始值)
1	检测到一个恢复	

- **SOF 令牌(SOF):** 当 USB 上检测到 SOF 令牌时设置这个位。

Bit 5: SOF	Description	
0	没有检测到 SOF 令牌	(初始值)
1	检测到 SOF 令牌	

- **USB 悬挂(US):**当 USB 上检测到悬挂状态设置这个位。

Bit 4: US	描述	
0	没有检测到悬挂状态	(初始值)
1	检测到悬挂状态	

- **USB 复位(UR):**当 USB 上检查到 USB 复位时设置这个位。

Bit 3: UR	描述	
0	没有检测到 USB 复位	(初始值)
1	检测到 USB 复位	

- **设置接口(SI):** 当 USB 上检测到设备接收到一个 *SetInterface* 请求时设置这个位。

Bit 1: SI	描述	
0	没有检测到设置接口请求	(初始值)
1	检测到设置接口请求	

- **设置配置(SC):** 当 USB 上检测到 *SetConfiguration* 请求时设置这个位。

Bit 0: SC	描述	
0	没有检测到设置配置请求	(初始值)
1	检测到设置配置请求	

12.2.5 设备中断屏蔽寄存器

设备中断屏蔽寄存器用于屏蔽系统级中断。寄存器中相应位置 1 将屏蔽设备中断寄存器中相应的中断。上电后寄存器的初始值为 0X00000000。

Bit:	31	……	……	……	……	……	……	8
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
Read:	Mask		Mask				Mask	
Write:								
Reset:	0	0	0	0	0	0	0	0

- **Bit 31~8，Bit 6 和 Bit 2:** 保留位。读这些位得到 0，对这些位只能写 0。

- **中断屏蔽(Mask):** 在寄存器相应位置 1 将屏蔽设备中断寄存器中相应的中断。

12.2.6 端点中断寄存器

端点中断寄存器用于设置端点级中断。端点 0 是全双工，有两个中断位，每个方向一个中断位。上电后寄存器的初始值为 0X00000000。

Bit:	31	……	……	……	……	……	……	16
读:	OUT EP							
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	15	……	……	……	……	……	……	0
读:	IN EP							
写:								
复位:	0	0	0	0	0	0	0	0

- **OUT 端点(OUT EP 15-0):** 16 位中的每一位表示从相应的 OUT 端点有一个中断发生， 只有 outep[2]和 outep[0]有效，其他位保留。

- **IN 端点(IN EP 15-0):** 16 位中的每一位表示从相应的 IN 端点有一个中断发生， 只有 inep[3]，inep[1]和 inep[0]有效，其他位保留。

12.2.7 端点中断屏蔽寄存器

端点中断屏蔽寄存器用于屏蔽端点中断。往寄存器中任何一位置 1 将屏蔽相应端点的中断寄存器中的任何中断。上电后寄存器的初始值位 0X00000000。

Bit:	31	……	……	……	……	……	……	16
读:	OUT EP Mask							
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	15	……	……	……	……	……	……	0
读:	IN EP Mask							
写:								
复位:	0	0	0	0	0	0	0	0

- **OUT 端点屏蔽(OUT EP 15-0):** 16 位中的每一位表示从相应的 OUT 端点有一个中断发生， 只有 outep[2]和 outep[0]有效，其他位保留。
- **IN 端点屏蔽 (IN EP 15-0):** 16 位中的每一位表示从相应的 IN 端点有一个中断发生， 只有 inep[3]，inep[1]和 inep[0]有效，其他位保留。

12.2.8 端点控制寄存器(EPnInCR, n = 0, 1, 3 EPnOutCR, n = 0, 2)

端点控制寄存器用于 CPU 设定端点的特性。端点 0 是一个全双工端点，有两个这样的寄存器，分别对应 IN 和 OUT。这是一个 32 位可读可写的寄存器。上电后寄存器的初始值由断点类型决定。

Bit:	31	……	……	……	……	……	……	8
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
Read:	SA		ET				F	S
Write:								
Reset:	0	0	0	0	0	0	0	0

- **Bit 31-8, 6,3：** 保留位。读这些位得到 0，对这些位只能写 0。

注意：SA 只在 OUT 端点 0 时可用。

- **Status Ack (SA):**对于 OUT 端点 0，如果 USB 主机初始化一个 3 个阶段的控制 OUT 传输，UDC 在状态阶段将返回 NAK 给 USB 主机，直到软件对这一位置 1。在这一位被置 1 后，并且在发送 OUT 状态之后，UDC 将对这一位清

0。

Bit 7: SA	描述	
0	在状态阶段的 Nak	(初始值)
1	在状态阶段的 Ack	

- 端点类型(ET1-0): 这两位表示端点类型。

Bit 5: ET1	Bit 4: ET0	Endpoint Type	
0	0	控制端点	(初始值)
0	1	ISO 端点	
1	0	批量端点	
1	1	中断端点	

注：对端点 0，ET 固定为 0；对端点 1、2，ET 固定为 2；对端点 3，ET 固定为 3。

- **清空(Flush—F):** T 当这个位设置为 1 时将清空 IN 端点的数据 FIFO。对 OUT 端点保留。

Bit 1: F	描述	
0	不清空 IN 端点的数据 FIFO	(初始值)
1	清空 IN 端点的数据 FIFO	

- **终止（Stall—S）:** 当这个位设置时，来自 USB 主设备的任何请求都被 Stall。

Bit 0: S	描述	
0	不终止 USB 主设备请求	(初始值)
1	终止 USB 主设备请求	

12.2.9 端点状态寄存器 (EPnInSR, n = 0, 1,3 EPnOutSR, n = 0, 2)

端点状态寄存器用于 CPU 判断端点状态。端点 0 是一个全双工端点，有两个这样的寄存器，分别对应 IN 和 OUT。这是个 32 位的可读可写寄存器。上电后寄存器的初始值为 0X00000000 或 0X00000010。

Bit:	31	……	22	……	11	10	9	8
读:			Rx Pkt Size					
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
读:		IN	OUT					
写:								
复位:	0	0	0	0/1	0	0	0	0

注:

- 对端点 0 和端点 2，复位后位 5-4 为 2'b01。
- 对 IN 端点 0 和端点 1 和端点 3，复位后位 5-4 为 2'b00，只读。

  - **位 31－23、位 10 和位 8:** 保留位。读这些位得到 0，对这些位只能写 0。
  - **接收包大小(Rx Pkt Size 11-0):** 这 12 位表示端点当前接收包的字节数。对 IN 包保留。
  - **IN 包状态(IN):** 当这个位设置时表示端口接收了一个 IN 令牌。对 OUT 端点这个位保留。

Bit 6: IN	描述	
0	没有接收 IN 令牌	(初始值)
1	接收 IN 令牌	

  - **OUT 包状态(OUT1-0):** 当这个位设置时表示端口接收了一个 OUT 包。这两位编码表示接收包的类型。这些位对 IN 端点保留。

Bit 5：OUT1	Bit 4：OUT0	端点类型	
0	0	None	(端点 0 IN 和端点 1 的初始值)
0	1	接收的数据	(端点 0 OUT 和端点 2 的初始值)
1	0	接收 Setup 数据(8 字节)	
1	1	保留	

12.2.10 IN 端点缓存大小寄存器(EPnInBSR, n = 0, 1，3)

对 IN 端点，端点缓存大小寄存器的位 15－0 保存着端点缓存的大小；对端点 0 固定为 8，对端点 1 固定为 16，对端点 3 固定为 4。

12.2.11 端点最大包长寄存器(EPnInMPSR,n=0,1,3 EPnOutMPSR, n =0,2)

端点最大包长寄存器规定了该端点支持的最大包长。端点 0 的 IN 和 OUT 固定为 32，端点 1 和端点 2 固定为 64，端点 3 固定为 16。

12.2.12 端点信息寄存器(EpnInfR, n = 0, 1, 2，3)

对每个端点，它的端点信息寄存器保存端点的信息，例如端点最大包长，端点类型，端点方向，以及端点的配置号，接口号和可替换设置。

上电后寄存器的初始值为 0X00000000。

Bit:	31	……	29	28	27	26	25	24
读:				MPS				
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	23	22	21	20	19	18	17	16
读:	MPS					ALTS		
写:								
复位:	0	0	0	0	0	0	0	0
Bit:	15	14	13	12	11	10	9	8
读:		IFN				CGN		
写:								
复位:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
读:		EPT		EPD	EPN			
写:								
复位:	0	0	0	0	0	0	0	0

- **Bit 31－29:** 保留位。读这些位得到 0，对这些位只能写 0。
- **最大包长(MPS 9-0):** 这 10 位为端点的最大包长，由 CPU 编程，按字节计算。
- **可替换设置 (ALTS3-0):** 这 4 位由 CPU 根据此端点所属的可替换设置编程。
- **接口号(IFN3-0):** 这 4 位由 CPU 根据此端点所属的接口号编程。
- **配置号(CGN3-0):** 这 4 位由 CPU 根据此端点所属的配置号编程。
- **可替换设置(ALTS3-0):** 这 4 位由 CPU 根据此端点所属的可替换设置编程。
- **端点类型(EPT1-0):** 这两位由 CPU 根据此端点的类型编程。

Bit 6: EPT1	Bit 5: EPT0	端点类型	
0	0	控制	(初始值)
0	1	同步	
1	0	批量	
1	1	中断	

- **端点方向(EPD):** 这个位由 CPU 根据端点方向编程。

Bit 4: EPD	**描述**	
0	OUT	(初始值)
1	IN	

- **端点号(EPN):** 这 4 位由 CPU 根据端点所属的端点号编程。

12.3 操作

UDC 相当于内部系统总线（AHB）的从设备。CPU 通过读/写映射到 UDC 的 FIFO 的内存来在 USB 总线和系统内存之间传输数据。所有数据传输是由中断驱动的。

USB 主机初始化 USB 传输，UDC 响应从 USB 主机发的各种命令。在一切操作之前，软件必须在 UDC 中完成可能的 CSRs 的编程。

12.3.1 断点列表

UDC 支持一个配置，两个接口，四个物理端点。

端点号	单点类型	方向	最大包的大小
0	控制	IN/OUT	32 字节
1	批量	IN	64 字节
2	批量	OUT	64 字节
3	中断	IN	16 字节

12.3.2 设备请求支持

参考 USB1.1 协议了解更多关于设备请求的格式的细节。下表指明了是硬件自动确认还是软件确认是什么设备请求。

请求	硬件/软件支持
CLEAR_FEATURE	硬件
GET_CONFIGURATION	硬件
GET_DESCRIPTOR	软件
GET_INTERFACE	硬件
GET_STATUS	硬件
SET_ADDRESS	硬件
SET_CONFIGURATION	硬件
SET_DESCRIPTOR	软件
SET_FEATURE	硬件
SET_INTERFACE	硬件
SYNC_FRAME	不支持因为没有 ISO 端点

12.3.3 接收和发送 FIFO

CPU 通过读/写 UDC 的 FIFO 来在 USB 总线和系统内存之间传输数据。

UDC 支持一个 32 字深度接-FIFO RAM，所有 OUT 端点共享这个公共 FIFO，接收 FIFO 的内存映射地址是 0X18000800。

对于 IN 端点，UDC 支持一个 28 字深度的发送-FIFO RAM。每一个 IN 端点都有一个 FIFO 控制器和由 FIFO 控制器产生的地址，地址是 RAM 中可以形成环路的一小块地址范围。地址范围的大小和 IN 端点缓存大小寄存器中说明的缓存大小匹配。FIFO 的基地址是前一个 FIFO 的输出。对于理想的控制端点 0 的 IN 端,它的基地址是 0X18000880（发送-RAM 的起始地址）。这个基地址加上 IN 端点 0 缓存大小寄存器中的可编程 FIFO 大小的四倍（等于 0X180008A0），便是端点 0 FIFO 控制器的输出，后继的端点 FIFO 控制器（端点 1）用这个输出为它的基地址。对于端点 3，情况类似。

12.3.4 IN 事务

对于一个 IN 端点，如果 UDC 接收到一个 IN 令牌后，检查发送（IN）数据 FIFO 中有无可用的数据。如果有可用的数据，读这个 FIFO，UDC 将发送数据到 USB 总线上，如果 FIFO 中没有任何数据，产生一个中断，回应 NAK 信号。一旦 CPU 发现一个中断，它将检查中断请求寄存器（Interrupt Request Register），查明那个端点请求了一个中断。在确定是哪个端点以后，CPU 检查端点状态寄存器，了解为什么会发生这个中断。一旦 CPU 知道是因为端点接收到一个 IN 令牌，它将把包的数据直接写在端点 IN 数据 FIFO 所映射到的那个地址上。当所有包的数据读写到了 FIFO 上，CPU 向一个预先定义的地址（0X1800041C）写一个数据，向 UDC 表明系统存储器向端点 FIFO 的一个传输已经完成。这个写操作确认在断点发送 FIFO 中的包。一旦 USB 主机重发一次 IN 令牌，UDC 把端点发送 FIFO 的数据传输到 USB 总线上。如果软件想清除端点 IN 数据 FIFO 上的包，可以通过设置 UDC 中的 CSRs 来实现。

一个端点上顺序发生的事件，如下所述：

注意：当接收到 Set Configuration 和 Set Interface 命令，UDC 的 DevIntR 将产生一个中断。当接收到 Set Descriptor，UDC 的 EPIntR 将产生一个中断。但是 Set Address 和 Set Feature 将不会向 CPU 产生中断。

- 初始化 UDC 中必需的 CSRs。
- 等待 USB 主机开始一个事务。
- 接收到 IN 事务。
- UDC 检查端点号并设置断点中断寄存器相应的位。检查端点缓存是否为空，如果不为空而且数据已经被确认，UDC 把数据直接传输到 USB 总线上；如果为空并且中断没有被屏蔽，给 CPU 产生一个中断，否则，CPU 需要检查端点中断寄存器。
- CPU 通过读端点 0 或 1 或 3 的端点寄存器知道 IN 令牌的到来，并向端点 0 或 1 的基地址直接写入包数据，如 1.3.3 所述。
- 把数据写入发送 FIFO 之后，CPU 应该写操作存储器映射地址 0X1800041C 一次来确认包数据，然后当 USB 主机重发 IN 令牌时，UDC 开始把发送 FIFO 中的数据传输到 USB 总线。
- 如果 CPU 想通过端点 0 或 1 或 3，发送一个 0 长度的数据包，首先写操作存储器映射地址 0X18000420 一次，然后写操作端点 0 或 1 的发送 FIFO 一次，在后，写操作存储器映射地址 0X1800041C 一次来确认包数据，最后 UDC 将向 USB 总线上发送一个 0 长度的数据包。

12.3.5 OUT 事务

一旦 UDC 从 USB 主机接收到 OUT 或 SETUP 包，如果接收 FIFO 对于包来说有可用的空间，UDC 会把数据传输到接收 FIFO 中。一旦包传输到接收 FIFO 中，UDC 产生一个中断通知 CPU 接收了一个包。CPU 读中断寄存器和该端点的包长寄存器，从此知道接收包中有多少个字节，然后 CPU 从接收-FIFO 中读出该字节数。必须注意 CPU 读的地址是接收-FIFO 映射的地址。从 USB 主机接收 OUT 数据包事务流程如下所述。

- 初始化 UDC 中必需的 CSRs。
- 等待 USB 主机开始一个事务。
- 接收到 OUT 事务
- 如果接收 FIFO 对于包来说有可用的空间，UDC 会把数据传输到接收 FIFO 中。否则，UDC 将返回一个 NAK，USB 主机将会重传这个包。

如果中断没有被屏蔽，一旦 UDC 把数据传输到接收-FIFO 中，UDC 将给 CPU 产生一个中断，否则，CPU 需要检查端点中断寄存器。

CPU 知道 OUT 令牌是给端点 0 或端点 2，并且从该端点的包长寄存器得知接收包中有多少个字节，然后 CPU 从接收-FIFO 中读出该字节数。

对于 OUT 端点 0，如果 USB 主机初始化一个有 3 个阶段的控制 OUT 事务，UDC 在状态阶段，将给 USB 主机返回一个 NAK，一直到这一位被软件置 1，当这一位被置 1 之后，并且发送完状态状态，UDC 将会将会清除这一位。

如果 CPU 知道接收到一个 0 包，必须独 0X1800041C 一次来确认 0 长度包的接受。

第13章 智能卡控制器(SCC)

13.1 概述

智能卡控制器（以下简称 SCC）支持符合 ISO/IEC 7816-3 标准的普通智能卡或 UIM 卡接口。

13.1.1 特性

SCC 的特性如下：

支持普通智能卡或 UIM 卡

8 比特 16 级的接受/发送 FIFO

支持异步字符(T = 0)通讯模式

- 数据长度：8 比特
- 奇偶校验位生成以及奇偶错误检测
- 自动错误报告以及 I/O 线上数据重传
- 支持正向约定以及反向约定

支持异步块(T = 1)通讯模式

波特率可以根据参数 F/D 调整

可编程外部时钟输出 SCC_CLK

支持时钟的停止

支持数据通讯和错误处理中断

13.2 管脚配置

表 13-1 SCC 管脚

管脚名	全称	I/O	功能
SCC_DATA	发送/接收数据管脚	输入/输出	连接 SCC 和卡的发送/接收数据管脚
SCC_CLK	串行时钟管脚	输出	连接 SCC 和卡的串行时钟管脚

13.2.1 SCC_DATA 管脚

SCC_DATA 是串行数据脚，双向。

13.2.2 SCC_CLK 管脚

SCC 的时钟输入脚，支持时钟停止。

13.3 寄存器配置

表 13-2 SCC 寄存器

积存器名	全名	读/写	初始值*1	虚拟地址	宽度
SCCDR	SCC 接收/发送 FIFO 数据寄存器	读	不确定	0X1803F500	8
SCCFDR	SCC FIFO 数据记数寄存器	读	0X00	0X1803F504	8
SCCCR	SCC 控制寄存器	读/写	0X00000000	0X 1803F508	32
SCCSR	SCC 状态寄存器	读/写	0X 8200	0X1803F50C	16
SCCTFR	SCC 发送因子寄存器	读/写	0X0173	0X 1803F510	16
SCCEGTR	SCC 额外保护时间寄存器	读/写	0X00	0X 1803F514	8
SCCECR	SCC Etu 计数器值寄存器	读/写	0X00000000	0X 9803F518	32
SCCRTOR	SCC 接收暂停时间寄存器	读/写	0X00	0X 9803F51C	8

13.3.1 SCC 发送/接收 FIFO 数据寄存器(SCCDR)

SCCDR 是一个 8 比特寄存器，做为 16 级接收/发送 FIFO 的入口。读 SCCDR 产生读接收 FIFO 数据操作，写 SCCDR 产生把数据写入发送 FIFO 操作。当 SCCCR.TRS = 0，FIFO 作为接收 FIFO，当 SCCCR.TRS = 1，FIFO 作为发送 FIFO。

当读 SCCDR，如果接收 FIFO 非空，得到接收 FIFO 中最早放入的数据项，相应的数据入队计数减 1；如果接收 FIFO 空，得到不可预测值。

当写 SCCDR，如果发送 FIFO 没有满，SCCDR 中的数据作为最后输入项复制到发送 FIFO 中，相应的数据入队计数加 1；如果发送 FIFO 满，写 SCCDR 对 FIFO 没有影响。

在上电复位时接收/发送 FIFO 自动清除。上电复位后读 SCCDR 得到不可预测值。

Bit:	7	6	5	4	3	2	1	0
读:								
写:								
复位:	—	—	—	—	—	—	—	—

注： 当 SCCCR.TRS = 0 时 SCCDR 对 CPU 只读，当 SCCCR.TRS = 1 时 SCCDR 对 CPU 只写。

13.3.2 SCC FIFO 数据记数寄存器(SCCFDR)

SCCFDR 是一个 8 位只读寄存器，它表明有多少个数据字节存储在 SCCDR 中。

Bit:	7	6	5	4	3	2	1	0
Read:				R4	R3	R2	R1	R0
Write:								
Reset:	0	0	0	0	0	0	0	0

- **Bit 7~5：**保留。这些位读时为 0，对这些位的写操作将被忽略。
- **Bit 4~0：**表明在接收/发送 FIFO 中，接收多少个数据字节或要发送多少个数据字节。值 0X00 表明接收/发送 FIFO 为空，值 0X10 表明接收/发送 FIFO 已满。

13.3.3 SCC 控制寄存器(SCCCR)

寄存器 SCCCR 是一个 32 比特寄存器，主要用于控制操作模式和发送格式模式。SCCCR 在上电复位时初始化为 0X00000000。

Bit:	31	30	29	28	27	26	25	24
Read:	SCCE	TRS	T2R				FDIV	
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	23	22	21	20	19	18	17	16
Read:	FLUSH*2						TRIG	
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	15	14	13	12	11	10	9	8
Read:	TP	CONV	TXIE*1	RXIE*1	TENDIE*1	RTOIE*1	ECIE*1	EPIE*1
Write:								
Reset:	0	0	0	0	0	0	0	0

Bit:	7	6	5	4	3	2	1	0
Read:	RETIE*1	EOIE*1			TSEND	PX		CLKSTP
Write:								
Reset:	0	0	0	0	0	0	0	0

注意:

1，对有关 SCC 中断的细节，请参考中断那一章节。

2，SCCCR.FLUSH 只能写“1”，常读为“0”。

- **Bit 28~26，22~18，和 5~4：**保留。这些位读时为 0，对这些位的写操作将被忽略。
- **SCC 使能(SCCE):** 使能或者禁用 SCC 操作。SCC 在空闲的状态下只消耗一点能量。清除 SCCCR.SCCE 将立即复位 SCC 内部状态、接收器/发送器以及复位 SCCSR 的某些位，但不会立即自动复位 SCC FIFO，SCCCR，SCCTFR，SCCECR 和 SCCRTOR。在 SCC 重新使能之前，软件应该确保 FIFO/寄存器可适当的配置而且可以在需要的时候手工的清 0/置 1。当 SCC 禁用时，AHB 总线读/写接口是可以用的。

Bit 15: SCCE	描述	
0	SCC 禁用	(初始值)

1	SCC 使能	

注意: 如果这一位被置 1 时，当发送数据已写入发送-FIFO（当 SCCCR.TRS=1）或开始位被检查到，串行发送将开始。串行接受将开始，（当 SCCCR.TRS=0）。所以在这一位被置 1 之前，传输模式的设置，FIFO 的清除，SCCCR 其他位的配置都必须完成。

- **发送或接收选择(TRS):**指示传输为发送或者接收。当 SCCCR.T2R = 1 并且 SCCCR.TRS = 1 时，SCC 在发送 FIFO 中所有数据发送结束后，发送 FIFO 为空而且发送器为空(并且 SCCSR.RETR_3 = 0)，并自动转入接收进程并且清 SCCCR.TRS 为 0。软件写 SCCCR.TRS 位在任何时候都有更高的优先级，当软件写一个不同的值时 SCCCR.TRS 位改变。软件必须保证 SCC 空闲或者在 SCCCR.TRS 位改变时 SCCCR.SCCE 被清除。

Bit 14: TRS	描述	
0	选择接收进程	(初始值)
1	选择发送进程	

注意:

- 在 SCCCR.TRS 被写 1 后，如果 SCCCR.T2R = 1，在至少一个数据正确发送后并且发送 FIFO 为空才会自动转入接收进程。
- 如果 SCC 由于校验错重发了 3 次，设置 SCCSR.RETR_3 = 1，并且不会自动转入接收进程(即使是发送 FIFO 为空，发送器为空且 SCCCR.T2R = 1)。 在软件处理 RETR_3 错误后，紧接的 SCC 传输是接收还是发送由软件写 SCCCR.TRS 位来控制。

- **自动从发送转入接收(T2R):** 只在 SCCCR.TRS = 1 并且 SCCSR.RETR_3 = 0 时有效， 在 SCCCR.TRS = 0 或者 SCCSR.RETR_3 = 1 时被忽略。这个比特控制当所有发送 FIFO 中数据发送完毕后(发送 FIFO 和发送器为空) ，SCC 是否自动转入接收进程.。如果 SCCSR.RETR_3 为 1，即使发送 FIFO 为空并且 SCCCR.T2R = 1，SCC 也不会自动转入接收，直到软件对 SCCCR.TRS 写 0。

Bit 13: T2R	描述	
0	SCC 传输方向完全由软件控制，在发送 FIFO 所有数据发送完毕(发送 FIFO 和发送器为空)后 SCCCR.TRS 不会改变，除非软件修改它。	(初始值)
1	在发送 FIFO 所有数据发送完毕(发送 FIFO	

	和发送器为空)后 SCC 自动转入接收进程并且同时自动对 SCCCR.TRS 和 SCCCR.T2R 清 0。	

注意: 软件写 SCCCR.T2R 位在任何时候都有较高的有先权

- **分频器选择 (FDIV):**这些位定义了设备时钟经过分频器得到怎样的 SCC_CLK 。

Bit 25-24: FDIV[1:0]	**描述**	
00	SCC_CLK 频率和设备时钟一样	(初始值)
01	SCC_CLK 频率是设备时钟的一半	
10	保留	
11	保留	

注意:

1. 在 UIM 卡的应用时, 它应该设置产生一个在 13/4 MHz (或者 13/8 MHz) 和 5 MHz 之间的 SCC_CLK 时钟频率 (了解更多的细节, 清参考文件 “gsm 11.11”。
2. 在普通的智能卡应用时, SCC_CLK 频率应该遵守 ISO7816-3 协议。

- **接收/发送 FIFO 清除(FLUSH):** 设置这个位将在一个 PCLK 周期内清空接收/发送 FIFO，但不会清空接收器/发送器。这个位只能被写‘1’，读出来常为‘0’。

Bit 8: FLUSH	描述	
0	不清空接收/发送 FIFO	(初始值)
1	在一个 PCLK 周期内清空接收/发送 FIFO	

- **接收/发送 FIFO 触发器(TRIG):** 这个位定义了接收/发送 FIFO 触发器值。当 SCCCR.TRS = 0， 如果接收 FIFO 中的接收字符数等于或者多于设置值， SCCSR.FFTRG 将被设置为‘1’；当 SCCCR.TRS = 1，如果发送 FIFO 中的空位置等于或者多于设置值，SCCSR.FFTRG 将被设置为‘1’。

Bit 17-16: TRIG [1:0]	**SCCSR.RFTG/TFTG 触发条件 (在接收/发送-FIFO 中数据记录值)**	
00	触发值是 1	(初始值)
01	触发值是 4	

10	触发值是 8	
11	触发值是 14	

- **卡传输协议(TP):** 规定异步模式下正在运行的（普通智能卡或 UIM 卡）卡的传输协议

Bit 7: TP	描述	
0	传输协议 T = 0	(初始值)
1	传输协议 T = 1	

- **卡数据传输约定(CONV):** 选择卡数据字节编/解码传输约定。规定是否将接收/发送数据的逻辑反转，接收/发送数据是 LSB 先还是 MSB 先。

Bit 6: CONV	描述	
0	发送数据不改变，LSB 先发；接收数据不改变，LSB 先收。	(初始值)
1	发送数据发送前比特反转，MSB 先发；接收数据 MSB 先收，存储前比特反转。	

注意：在 TS 字符接收之前，SCCCR.CONV 必须清零。之后，根据接收到的 TS 字符，SCCCR.CONV 相应的置 1 或清 0。

- **发送触发中断使能(TXIE):**对于 SCCCR.TRS = 1, 使能或禁止发送-FIFO 数据数量符合触发值的中断 (TXI)。当发送-FIFO 剩余的空间值等于或大于 SCCCR.TRIG 设置的触发值, 而且如果 SCCCR.TXIE = 1，一个 TXI 将发生。

Bit 13: TXIE	描述	
0	发送-FIFO 数据数量符合触发值中断(TXI) 禁止。	(初始值)
1	发送-FIFO 数据数量符合触发值中断(TXI)使能。	

注意:当把发送数据写入发送-FIFO 使 SCCSR.TFTG = 0 或者 SCCCR.TXIE 清 0，或者 SCCCR.TRS 清 0 时，TXI 中断请求可以被清除。

- **接收触发中断使能(RXIE):**对于 SCCCR.TRS = 0, 使能或禁止接收-FIFO 数据数量符合触发值的中断 (RXI)。当接收-FIFO 中的字符等于或大于 SCCCR.

TRIG 设置的触发值, 而且如果 SCCCR.RXIE = 1，一个 RXI 将发生。

Bit 12: RXIE	描述	
0	接收-FIFO 数据数量符合触发值中断(RXI)禁止。	(初始值)
1	接收-FIFO 数据数量符合触发值中断 (RXI)使能	

注意:当从接收-FIFO 中读出数据，使 SCCSR.RFTG = 0 或者 SCCCR.RXIE 清 0，或者 SCCCR.TRS 置 1 时，RXI 中断请求可以被清除。

- **发送停止中断使能(TENDIE):**使能或禁止发送停止中断 (TENDI)。一旦发送停止而且 SCCSR.TEND = 1 (发送-FIFO 和发送器都为空), 如果 SCCCR.TENDIE = 1，一个 TENDI 将会发生。

Bit 11: TENDIE	描述	
0	发送停止中断(TENDI) 禁止。	(初始值)
1	发送停止中断(TENDI) 使能。	

- **SCC 接收结束中断使能(RTOIE):**使能或禁止 SCC 接收结束中断(RTOI)。对于 SCCCR.TRS = 0, 当接收完最后一个接收数据的奇偶校验位 (没有其他数据到来),如果接收-FIFO 数据数量既不为空，也没有符合触发值，并保持一个预期等待时间，一个 RTOI 中断将发生如果 SCCCR.RTOIE = 1。 预期等待时间通过 etu 模块中的 SCCRTOR 来改变。

Bit 10: RTOIE	**Description**	
0	RTOI 中断禁止	(初始值)
1	RTOI 中断势能.	

- **SCC ETU 计数器溢出中断势能(ECIE):** 使能或禁止“SCC etu 计数器溢出”中断(ECI)。 内部的 etu 计数器是为了计算字符/块的等待时间，当溢出时通知使用者而设计的。预备等待时间是通过软件设置 etu 单元中的 SCCECR。

Bit 9: ECIE	**Description**	

0	ECI interrupt is Disabled.	(Initial value)
1	ECI interrupt is Enabled.	

- **奇偶校验错误中断使能(EPIE):** 使能或禁止产生奇偶校验错误中断势能(EPI)。

Bit 8: EPIE	描述	
0	EPI 中断禁止。	(初始值)
1	EPI 中断使能	

- **重发 3 次中断使能(RETIE):** 使能或禁止产生重发 3 次中断 (RETI)。

Bit 7: RETIE	描述	
0	RETI 中断禁止	(初始值)
1	RETI 中断使能	

- **接收溢出错误中断使能:** 使能或禁止产生接收溢出错误中断 (EOI)。

Bit 6: EOIE	描述	
0	接收溢出错误中断(EOI)禁止	(初始值)
1	接收溢出错误中断(EOI) 使能	

- **开始字符 TS 结束(TSEND):**确定接收 TS 字符是否结束。当 TSEND 为 0 时，硬件不会报告奇偶校验错误。如果在 SCC 接收完 TS 字符之后，用户想要通过 RXI 中断被通知到，那么用户应该配置 SCCCR.TRIG [1:0] = 2'b00 和 SCCCR.RXIE = 1, 这样用户可以及时的读到 TS 字符。

Bit 3: TSEND	描述	
0	接收 TS 字符阶段没有结束。	(初始值)
1	接收 TS 字符阶段结束。	

注意:软件应该确保在接收 TS 字符之前， TSEND 被清 0； 在接收 TS 字符之后，TSEND 立即被置 1。

- **参数 X (PX):**确定 SCC 是否支持“时钟停止”模式，如果支持“时钟停止”模式， SCC_CLK 管脚将停止在高电平状态或低电平状态。

Bit 2-1: PX [1:0]	描述	

00	SCC 不支持“时钟停止”。	(初始值)
01	SCC_CLK 停止在低电平状态。	
10	SCC_CLK 停止在高电平状态。	
11	保留	

注意: 在 UIM 卡应用时，根据“file characteristics” (参考 gsm11.11)确定的条件，时钟只能停止。所以软件写参数 X(PX) 时，应该根据上面所述的要求，不应该根据 ISO7816-3 中规定普通智能卡所用的 Ta (i)。

- **智能卡时钟停止(CLKSTP):**当 SCC 支持时钟停止而且将要进入时钟停止时，软件应该置 SCCCR.CLKSTP = 1, 然后 SCC_CLK 将在 1920 个周期后停止 (SCCSR.TRANS 位将被清除 0)。如果 SCC_CLK 已经停止, 软件清除 SCCCR.CLKSTP 将重新开始 SCC_CLK 并且传输在 1023 个周期后可能继续 (SCCSR.TRANS 位将被置 1)。传输只在 SCCSR.TRANS = 1 时执行。

Bit 0: CLKSTP	**描述**	
0	SCC 已经离开或正在离开时钟停止模式。	(初始值)
1	SCC 已经进入或正在进入时钟停止模式	

13.3.4 SCC 状态寄存器(SCCSR)

状态寄存器 SCCSR 是一个 16 比特的寄存器，主要用于指示 SCC 的当前状态。 SCCSR 在上电复位时设置为 0X8200。

Bit:	15	14	13	12	11	10	9	8
Read:	TRANS*1			ORER*2	RTO	PER*2	TFTG	RFTG
Write:								
Reset:	1	0	0	0	0	0	1	0

Bit:	7	6	5	4	3	2	1	0
Read:	TEND*1,2,3			RETR_3*2				ECNTO*2
Write:								
Reset:	0	0	0	0	0	0	0	0

注意:

1，当 SCCCR.SCCE 被禁止， SCCSR.TRANS 将被自动的置 1；当 SCCCR.SCCE 被禁止时，如果 SCCCR.TRS = 1 而且 SCCFDR = 0X00, SCCSR.TEND 将被自动的置 1，SCCSR 的其他位将保持不变。

2，ORER, PER, TEND, RETR_3 和 ECNTO 只能被写入 0，写入 1 将被忽略。

3，当 SCCCR.TRS = 1 时，TEND 只读；当 SCCCR.TRS = 0 时，TEND 是一个读/写位（只能被写 0）。

- **Bit 14~13、Bit 6~5 和 Bit 3-1:** 保留。这些位读为 0，写这些位将被忽略。
- **传输状态(TRANS):**确定传输是否停止。当 SCCSR.TRANS 为 0 时，SCC 不能传输数据。在 SCCCR.CLKSTP 位被置 1 并且没有传输的 1920 个周期之后，SCCSR.TRANS 被置 1。当 SCCSR.TRANS 为 0 时，清除 SCCCR.CLKSTP 位后的 1023 个周期 SCCSR.TRANS 将被置为 1。

Bit 15: TRANS	描述	
0	SCC 在时钟停止状态或者从时钟停止状态恢复。SCC 不能进行正常的传输。	
1	SCC 在正常的传输状态。	(初始值)

注意:

- 如果 SCCCR.CLKSTP 保持 1 的时间小于 1920 周期,时钟将不会真正的停止,并且 SCCSR.TRANS 将不会清 0；如果时钟停止， SCCSR.TRANS 将在时钟恢复的 1023 个周期后被置为 1。
- 当 SCCCSR.TRANS = 0 时，如果 SCCCR.SCCE 被禁用(= 0), 因为 SCCSR.TRANS 将自动置 1 而且 1023 个周期传输恢复时间将不会满足 ,用户应该确保在卡恢复期间在足够的延迟后 SCC 是在使能状态。否则，用户应该确保当 SCCSR.TRANS = 1 时，SCCCR.SCCE 被禁用(= 0)。
  - **接收溢出错误 (ORER):** 这个位表示有一个接收溢出错误发生。当接收 FIFO 满，完成一个新数据接收后，(表示接收器和接收 FIFO 同时满) SCCSR.ORER 被设置为 1。CPU 读此位为 1 后必须从接收 FIFO 读数据并且将 SCCSR.ORER 清零。

Bit 10: ORER	描述	
0	没有接收溢出错误发生	(初始值)
1	发生了接收溢出错误	

注意:

- 在接收溢出错误发生前接收的数据保存在接收 FIFO 中，之后的数据被丢弃。
- 当 SCCSR.ORER 设为 1 时，接收器接收的数据不会写入接收 FIFO，但接收器仍然继续接收数据。

  - **接收暂停(RTO):** 指示接收暂停状态是否发生，对于 SCCCR.TRS = 0, 当接收完最后一个接收数据的奇偶校验位 (),如果接收-FIFO 数据数量没有符合触发值（接收-FIFO 数据数量既不为空）而且没有进一步的数据到来，这个状态保持一个预期等待时间（预期等待时间通过 etu 模块中的 SCCRTOR 来改变）SCCSR.RTO 将被置 1。当 SCCCR.TRS=1 或者接收-FIFO 为空或者接收-FIFO 中的数据满足了触发值， SCCSR.RTO 将自动被清 0。

Bit 11: RTO	描述	
0	没有接收暂停	(初始值)
1	有接收暂停	

注意:用户应该注意 SCCSR.RTO = 1 并不表示接受的结束。SCCSR.RTO = 1 只是用来当接收-FIFO 中的数据没有满足触发值并保持一个期望的等待时间而且没有进一步的数据到来时，通知用户区读接收-FIFO。

  - **奇偶校验错误(PER):**表明在发送和接收数据时有一个奇偶校验错误发生。当 SCCCR.TRS = 0, 当接收数据中的 1 的个数加上校验位和根据翻转的奇偶校验检查不相同的时候，奇偶校验错误发生。对于 T = 0,有校验错误的接收数据将不会写到 FIFO 上, 一个字符再传输将发生;对于 T = 1, 有校验错误的接收数据将会写到 FIFO 上, 一个字符再传输不会发生

当 SCCCR.TRS = 1, SCCSR.PER = 1 表明一个奇偶校验错误信号从卡反馈来; 如果 同时 RETR_3 是 0，当 T=0 时，硬件会自动重传输最后一个字节，当 T=1 时，不会有字符的重传输发生。

Bit 11: PER	描述	
0	没有校验错误发生或者校验错误被软件通过写 1 而清除了。	(初始值)
1	有一个校验错误发生	

- **发送-FIFO 数量满足触发值(TFTG):**当 SCCCR.TRS = 1 时，SCCSR.FFTRG = 1 表示发送 FIFO 中剩余地址空间数据大于或者等于 SCCCR.TRIG 设定的触发值。

Bit 8: FFTRG	描述	
0	Transmit-FIFO 中数据数量不满足触发值，或者 SCCCR.TRS = 0	(初始值)
1	FIFO 中数据数量满足触发值	

注: SCCDR 是一个 16 位的 FIFO 寄存器。当 SCCSR.TFTG = 1 时，至少发送触发器设置的数据字节数可写。在发送-FIFO 中可以发送的数据字节是由 SCCFDR 指出。

- **接收-FIFO 数量满足触发值(RFTG):** 当 SCCCR.TRS = 0 时， SCCSR.FFTRG = 1 表示接收 FIFO 中接收数据的数量等于或大于 SCCCR.TRIG 设定的触发值。

Bit 8: FFTRG	描述	
0	Receive-FIFO 中数据数量不满足触发值，或者 SCCCR.TRS = 1	(初始值)
1	FIFO 中数据数量满足触发值	

注: SCCDR 是一个 4 字节 FIFO 寄存器。当 SCCSR.RFTG = 1 时，至少接收送触发器设置的数据字节数可读。在接收-FIFO 中可以接收的数据字节是由 SCCFDR 指出。

- **发送停止(TEND):** 当 SCCCR.TRS = 1, 一旦发送停止 (发送-FIFO 和发送器都为空) SCCSR.TEND 将会被置 1。在 SCCSR.TEND 将会被置 1 后，当 SCCCR.TRS = 1,一旦用户向发送-FIFO 写数据， SCCSR.TEND 将自动清除；如果 SCCCR.TRS 变为 0, SCCSR.TEND 只能用软件写 0 清除它(= 0) 。

Bit 7: TEND	描述	
0	表示发送-FIFO 不为空或者发送_FIFO 为空但发送器不为空；或者被软件清	(初始值)
1	表示发送结束，发送_FIFO 发送器都为空	

注意:如果发送-FIFO 为空 (SCCCR.TRS = 1 并且 SCCFDR = 0X00), ，当 SCCCR.SCCE(=0) 禁用或者重传结束 (SCCSR.RETR_3 = 1)SCCSR.TEND 将被自动置 1。用户应该保证发送的正常结束。

- **重发 3 次 (RETR_3):** 指示有奇偶错的数据已经重传了 3 次或者更多。当有奇偶错的数据连续重传 3 次并且第三次重传的数据还有奇偶错时设置这个位。在传送过程中，如果 SCCCR.RETIE 设置为 1，当 SCCSR.RETR_3 被设置为 1 时产生 RETI 中断。当这个位为 1 时硬件重传将停止。当这个位被软件清零后新的传送可以继续。写 0 将清除这个标志，写 1 没有影响。

当 SCCCR.TRS = 0 时这个位被忽略(接收模式).

Bit 4: RETR_3	描述	
0	数据重传没有超过 3 次	(初始值)
1	有奇偶错的数据重传次数等与或者超过 3 次。	

- **SCC ETU 计数溢出(ECNTO):**指示用户设定的字符间的延时是否到，即，SCC 间隔 etu 计数器是否溢出。(溢出在这里是指间隔 etu 计数器的值等于 SCCECR 设定的值)。如果 SCCCR.ECIE 为 1，则使能了 etu 计数器溢出中断(ECI)。写 0 将清除这个标志，写 1 没有影响。

Bit 0: ECNTO	描述	
0	没有溢出	(初始值)
1	溢出	

13.3.5 SCC 传输因子寄存器(SCCTFR)

SCCTFR 寄存器是一个 16 位读/写寄存器， 主要用于控制传输因子。寄存器的位 15 到位 11 保留，读这些位得到 0，对这些位只能写 0。CPU 可以在任何时候读 SCCTFR， 在 ATR 之后后续字节传输之前 CPU 应该写这个寄存器。在上电复位时 SCCTFR 被初始化为 0X0173 (D’371)。

寄存器 SCCTFR [10:0]的值应该等于(F/D)计算的值减 1。

传输数据的 etu (elementary time units: the period for 1 bit transfer)根据传输因子 F 和 D 的实际值确定 。(参考 ISO 7816-3 的 6.5.2 节了解更多 F 和 D 的信息)

1etu = (F/D) × (1/f)

其中:

- F: 传输因子，ATR 参数；
- D: 波特率调整因子，ATR 参数；
- f: SCC 时钟频率

Bit:	15	14	13	12	11	10	9	8
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	1

Bit:	7	6	5	4	3	2	1	0
Read:								
Write:								
Reset:	0	1	1	1	0	0	1	1

13.3.6 SCC 额外保护时间寄存器(SCCEGTR)

寄存器 SCCEGTR 是一个 8 位的读写寄存器，设置它相当于设置 ATR 中 TC(1)中的参数，参数 N 是从 SCC 发送字符到读卡器的额外的保护时间。如果没有 TC(1)， SCCEGTR 被设置为 0X00。上电复位把 SCCEGTR 初始化为 0X00。

在 UIM 卡应用时， TC(1) 只能被设置成 0X00 或者 0XFF, 设置为其他值是被禁止的。

Bit:	7	6	5	4	3	2	1	0
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

注意:

1. 当 TC(1) 为 0X00（SCCEGTR = 0X00），在 T = 0 和 T = 1 两种情况下，两个连续的字符电平前沿之间的延时都是 12 etu 。
2. 当 TC(1) 为 0XFF（SCCEGTR = 0Xff），两个连续的字符电平前沿之间的最小延时时间在两个传输方向上都是相同的，在 T = 0 情况下最小延时时间是 12 etu ，在 T = 1 情况下最小延时时间是 12 etu 。

13.3.7 SCC ETU 计数器值寄存器(SCCECR)

SCCECR 是一个 32 位读/写寄存器，用于设置字符之间间隔时间。位 31~20 保留，这些位读一直为 0，只能写 0。时间间隔是通过 etu 单元计数实现的，在 ATR 期间，卡发的两个连续的字符电平前沿之间的延时应该不超过 9600 etus。

如果 SCCECR 的值不为零，则它是 etu 计数器的计数目标值。当内部 etu 计数器的计数值等于 SCCECR 的值，如果 SCCCR.SCCIE 使能，则产生一个 etu 计数溢出中断(ECI)。

Bit:	31			~				20
Read:								
Write:								
Reset:	0	.	.	.	.	.	.	0

Bit:	19			~				0
Read:								
Write:								
Reset:	0	.	.	.	.	.	.	0

注意:

1. 在当卡停止供给时钟使时钟停止(SCCSR.TRANS = 0) 期间，内部 etu 计数器将自动清除。
2. 写 SCCECR 将自动清除内部 etu 计数器。

13.3.8 SCC 接收超时寄存器 (SCCRTOR)

The SCCRTOR 寄存器是一个 8 位的读/写寄存器，它是用来设置当接收-FIFO 既不为空也不满足触发值，并接收完最后一个奇偶校验位（没有进一步的数据到来）时，期望等待时间(在 etu 单元里设置)。上电复位将把 SCCRTOR 初始化为 0X00 。

如果 SCCRTOR 的值不为零，则它是内部结束计数器的计数目标值。当内部结束计数器的计数值等于 SCCRTOR 的值（除了期望等待时间终止），如果 SCCCR. RTOIE 使能，则产生一个计数溢出中断(RTOI)。

Bit:	7	6	5	4	3	2	1	0
Read:								
Write:								
Reset:	0	0	0	0	0	0	0	0

注意: SCCRTOR 设置的值应该大于两个连续的字符电平前沿之间的延时时间的值（在 etu 单元里）。

13.4 接收器/发送器

13.4.1 接收器(SCCCR.TRS = 0)

接收器接收串行数据流并将其转换成一个并行字符。当被使能时，它搜索起始位并确认它，并且在位中间采样后续的数据位。接收器对数据进行奇偶校验。当接收到一个有效的数据字节后，接收器自动将该字节传递到 SCCDR (接收 FIFO)。当发现奇偶校验错时，对 T＝0 和 T＝1 模式都设置 SCCSR.PER 位。对 T = 0 模式，接收器停止将接收字节写入接收 FIFO，同时在 10.5etu 到 12etu 间将 SCC_DATA 管脚驱动为低电平等待错误字符的重发。对 T = 1 模式，SCC 不会返回错误信号，有奇偶错的数据将写入接收 FIFO。

13.4.2 发送器(SCCCR.TRS = 1)

发送器从 CPU 接收一个并行字符并且将它串行发送出去包括增加的起始位和奇偶校验位。在执行串行数据传送时，SCC 开始将 SCCDR（发送 FIFO）的数据传送发送器，然后通过 SCC_DATA 管脚发送数据，以起始位开始，奇偶校验位结束。对 T=0 模式，如果 SCC 接收到一个错误信号，SCC 设置 SCCSR.PER 位，停止从发送 FIFO 加载新数据，并重传发送器中的当前数据。发送器对一个数据最多重传 3 次，重传 3 次以后 SCCSR.RETR_3 设置位 1 并停止重传。对 T = 1，不会接收到错误信号，也不进行字符重传。

当所有数据都正确传送完后(发送 FIFO 为空，SCCSR.RETR_3 = 0)，如果 SCCCR.T2R = 1，SCC 将自动转入接收状态，SCCCR.TRS 和 SCCCR.T2R 自动清零。

13.5 TS 字符和 SCCCR.CONV 位

SCCCR.CONV 应该在 TS 字符接收之后再配置（置 1/清 0）。

图 13-1 开始 TS 字符的波形。描述了两种类型卡(正向约定和正向约定)的起始字符 TS 的波形，以及 SCCCR.CONV 的配置。详细细节参考 ISO7816 标准。

在正向约定中，SCCCR.CONV = 0，数据的电平不反转，传输时 LSB 先。TS 字符是 0X3B。奇偶校验是偶校验的，所以校验位为 1。

在反向约定中，SCCCR.CONV = 1，数据电平反转，传输时 MSB 在先。TS 字符为 0X3F。奇偶校验是偶校验的，所以校验位为 0，相当于逻辑电平‘1’。对接收和发送对如此规定。

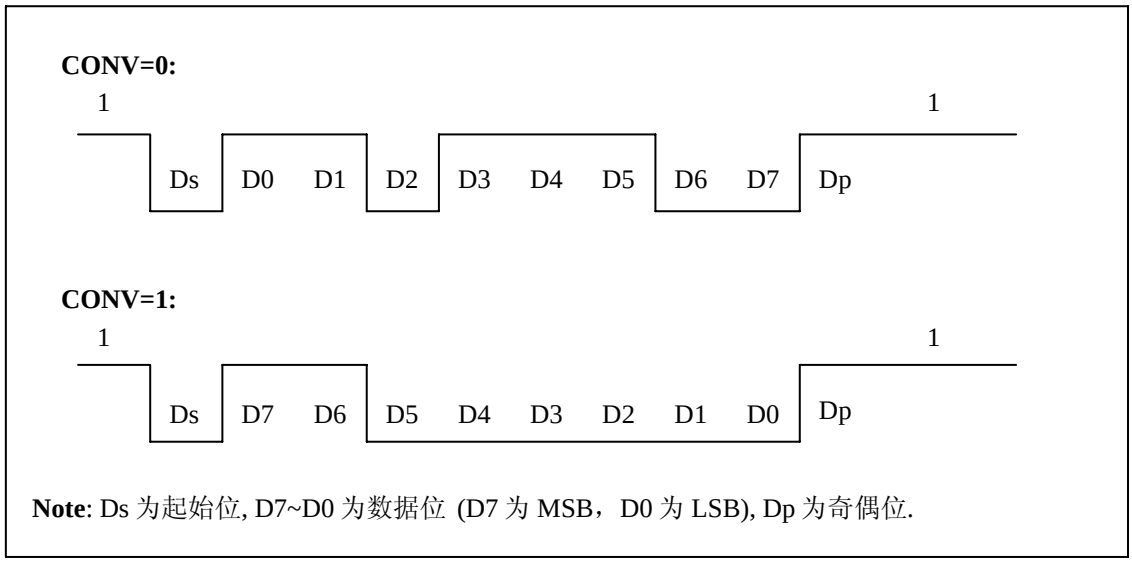


图 13-1 开始 TS 字符的波形

13.6 管脚连接

下图，描述了智能卡控制器和卡的连接。

用片上的一个 GPIO 当作复位信号，数据线通过一个电阻上拉到 Vcc。除了图中的管脚外还有电源和地的管脚。

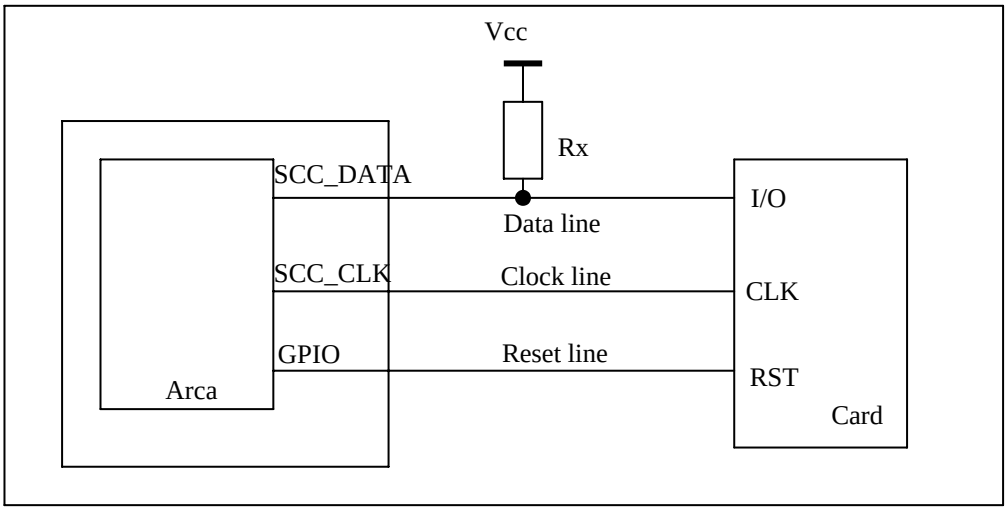


图 13-2 SCC 管脚连接图

13.7 SCC 操作

对于一般的智能卡或者 UIM 卡的应用,用户应该注意 SCCCR.FDIV 和 SCCEGTR 配置可能上的不同。对于一般的智能卡或者 UIM 卡的应用,其他的配置相同。

13.7.1 初始化

在接收或者发送前采用如下流程初始化 SCC。在下列的步骤里，当传输空闲时，清除 SCCCR.SCCE 将被忽略。下图给出了一个初始化流程的例子。

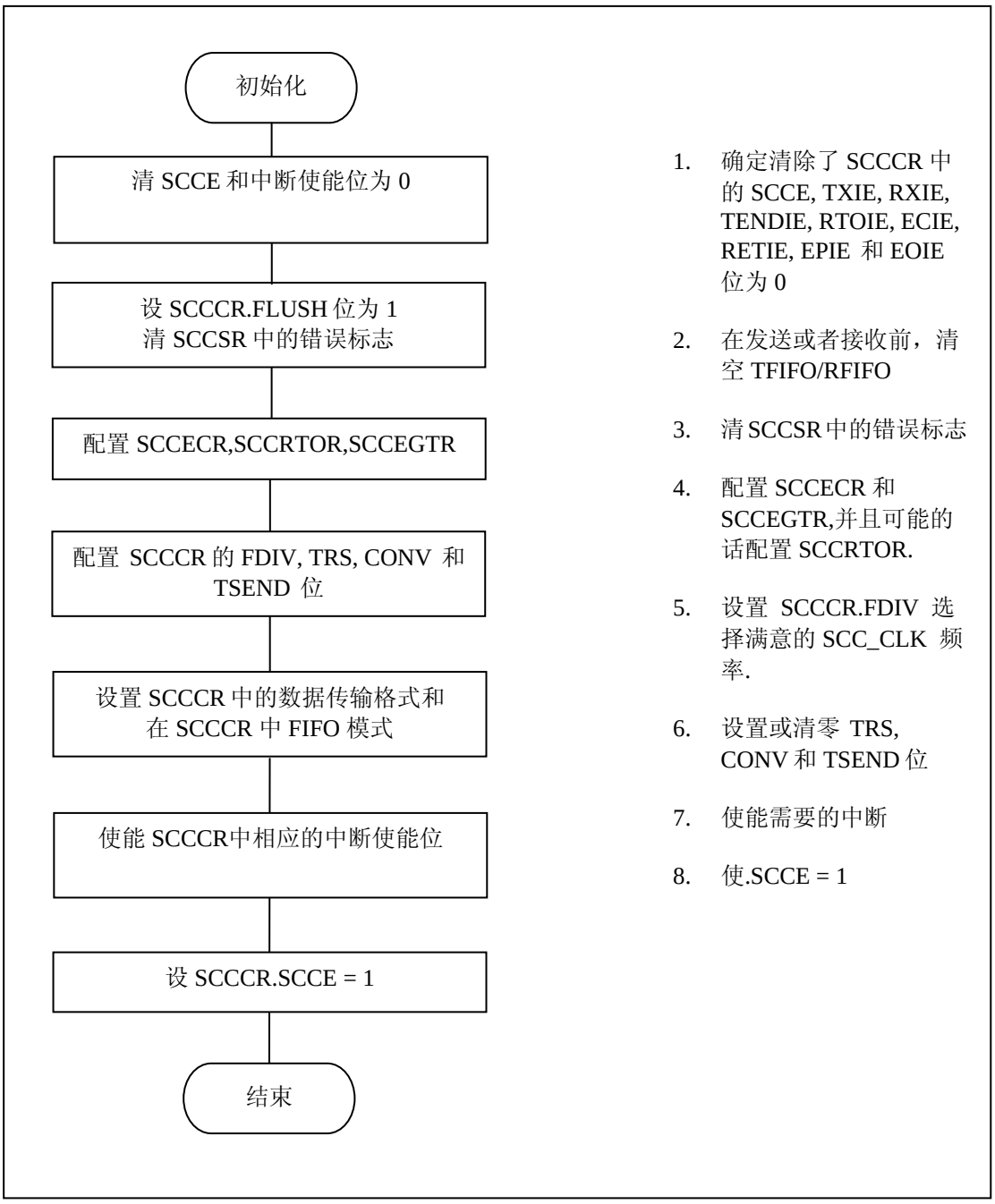
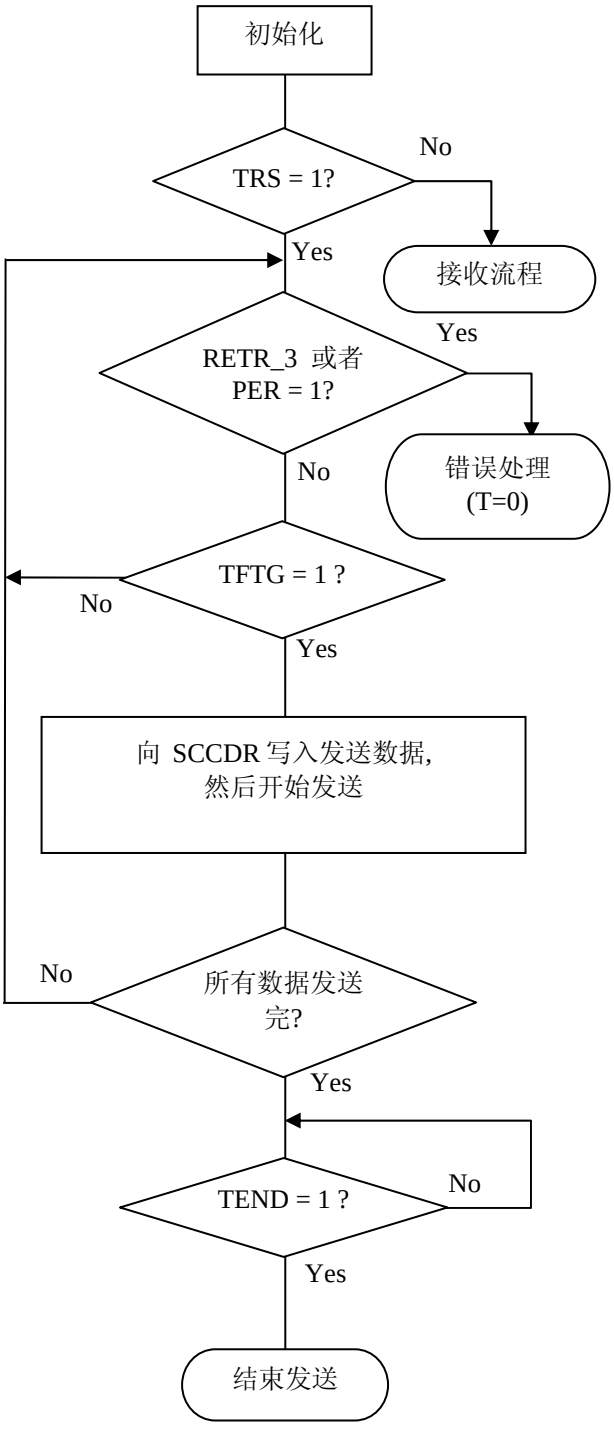


图 13-3 初始化流程例子

13.7.2 串行数据传送

下图给出了传送过程的一个例子。假定卡的激活和 ATR 阶段已经结束，TSEND 和 CONV 已经设置为恰当的值。



1，如上所述初始化 SCC
2，判断是否为发送处理
3，检查错误状态，如果找到错误便处理。
4，状态检查和发送数据写：读 SCCSR 并检查 TFTG 是否为 1。如果 TFTG 置 1,然后写发送数据到 SCCDR 里，可以写的发送数据数量是发送触发器设置的数目。

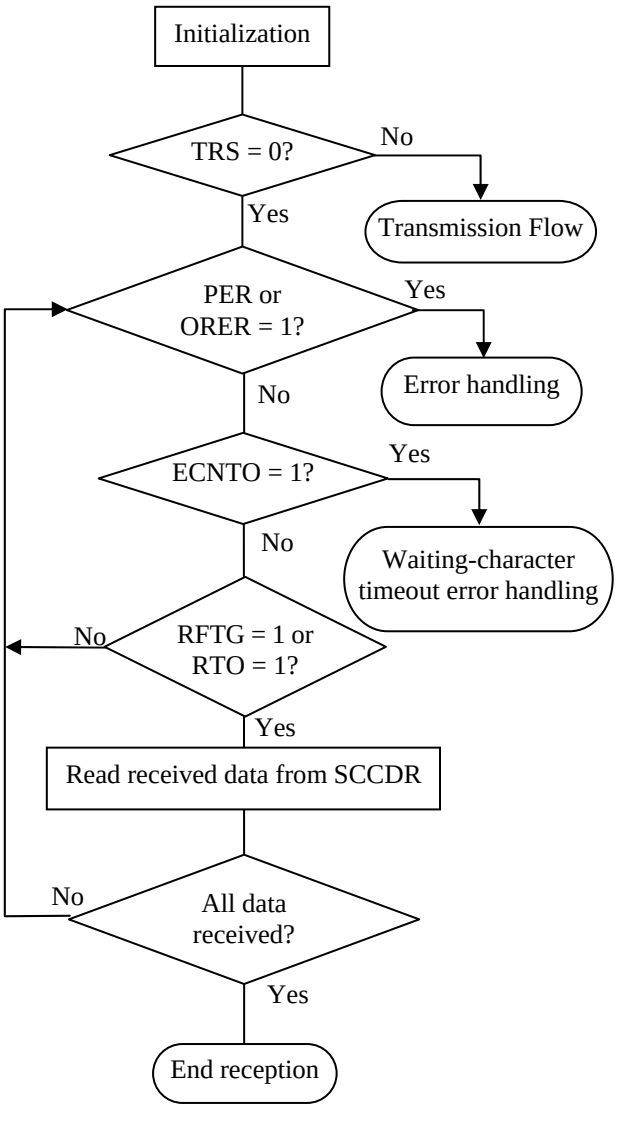
5，发送继续步骤：为了继续发送，从 TFTG 读 1 来确定向发送 FIFO 写是可能的，然后向 SCCDR 写入数据。

6，当最后的一个字节写入了发送 FIFO,等待 SCCSR.TEND 被置为 1 然后结束当前的发送。

图 13-4 传送流程的例子

13.7.3 串行数据接收

下图给出了接收过程的一个例子。假设卡已经被激活。



1. 如初始化中描述的初始化 smart card 设备
2. 判断是否是一个接受处理.
3. 接收错误处理：读 SCCSR 中的 PER 和 ORER 标志，判断是否发生错误，执行相应的错误处理，然后清错误标志为 0。
4. 等待字符超时错误或处理:读 ECNTO 看 etu 计数器超时是否发生，执行相应的错误处理，然后在恰当时间清 ECNT
5. 接收继续步骤：状态检测及读接收数据：读 SCCSR 并且如果 FFTRG = 1 时从 SCCDR 读至少接收触发设置值规定字节的接收数据。
6. 接收继续步骤：状态检测及读接收数据：读 SCCSR 并且如果 RFTG = 0 并且 RTO=1 时,读 SCCDR 中接收到的数据，知道接收-FIFO 为空。读 SCCFKR 可以确定 SCCDR 中接收到的数据字节数量。
7. 在所有数据接收之后，结束当前的接收。

图 13-5 接收流程的例子

13.8 使用注意事项

13.8.1 中断操作

在 SCC 中有 8 个中断源：TXI, RXI, TENDI, RTOI, EPI, EOI, RETI and ECI 。这些中断组合起来产生一个 SCC 中断。每个 SCC 中断可以通过对 INTC 中断屏蔽寄存器（IMR）相应的屏蔽位写‘1’来屏蔽。

表 13-3 SCC 操作状态及中断源

模式	状态	标志	寄存器	掩码	寄存器	中断源	DMAC 激活
发送 /接收模式	发送/接收奇偶错误	PER	SCCSR	EPIE	SCCCR	EPI	不行
	SCCECR 计数器溢出	ECNTO	SCCSR	ECIE	SCCCR	ECI	不行
发送模式	发送-FIFO 满足 TRIG 值	TFTG	SCCSR	TXIE	SCCCR	TXI	不行
	重传次数等于或超过 3 次	RETR_3	SCCSR	RETIE	SCCCR	RETI	不行
	发送结束	TEND	SCCSR	TENDIE	SCCCR	TENDI	不行
接收模式	接收-FIFO 满足 TRIG 值	RFTG	SCCSR	RXIE	SCCCR	RXI	不行
	接收超时	RTO	SCCSR	RTOIE	SCCCR	RTOI	不行
	接收溢出错误	ORER	SCCSR	EOIE	SCCCR	EOI	不行

推荐的中断优先级排序是 EOI > RXI/TXI > EPI > RETI > TENDI/RTOI > ECI。

第14章 通用输入输出口(GPIO)

14.1 综述

通用输入输出口（GPIO）是用作控制和在系统和外围电路间的握手，它可以生成输出信号和可以捕获为特定的应用的输入信号。所有的 GPIO 引脚都有复合的功能并且可以和其它芯片的模块相复合。

这块芯片支持 27 个 GPIO 引脚。有 32 位的 7 个特殊功能的 32 位寄存器来服务于这 27 个 GPIO 口。这 7 个寄存器对应每个 GPIO 口有着不同的功能，下面来解释一下：

GPIO 数据寄存器 (GPDR)

GPIO 定向控制寄存器 (GPDIR)

GPIO 选择功能 L / U 寄存器 (GPALR and GPAUR)

GPIO 中断检测模式寄存器 (GPIDR)

GPIO 中断使能寄存器 (GPIER)

GPIO 中断标志寄存器 (GPFR)

对于复合接口，当功能为可编程 IO 接口时，该芯片的模块功能不能被用做同步。芯片模块的输出端口被屏蔽并且输入到模块的接口被转换到无效值。

IO 编程功能是基于所有的 GPIO 都支持的基本功能。当引脚被配置成输出，他将被相应的在 GPDR 寄存器的相应位对应的值驱动，而且引脚的电压值不能被读出。GPDR 可以像普通寄存器一样被读，写。当配置成输入，引脚电压任何时候都可以从 GPDR 中读出，但是 GPDR 寄存器保持上次从 AHB 总线上写入的值。

GPIO 引脚 18 和 23 能通过设置 GPIER 寄存器 IEm (m = 0, 1)为 1 来配置成异步外部中断请求。它们能配置成高/低电平或者上升/下降沿来检测中断。被检测到的中断作为 2 个中断源送达 INTC 模块。对于电平或者沿的检测模式，检测到的有效电平或者有效沿可以在 GPFR 中反映。当 GPIO 的引脚没有配置成中断功能或者 GPIO 引脚检测机制变化，硬件将自动清除在 GPFR 的相应位。

注意:

如果端口引脚处于芯片功能可变换的状态(GPALR.An [1:0] ≠ 0* 或者 GPAUR.An [1:0] ≠ 0*) 或者中断功能(GPIER.IEm = 1)，管脚方向将自动被硬件设置到适当的值。相应的方向控制位(GPDIR.DIn)被忽略，软件也用不着配置或者检测它。

在下面的描述中，An[1:0]＝0 代表管脚 n 已经有效的配置到正常的 GPIO 了，An [1:0] ≠ 0 代表管脚 n 配置到内部芯片功能 01/2.。对于 An [1:0] ≠ 0 表示选择了保留功能，管脚 n 的功能同配置到正常 GPIO 功能一样。

14.1.1 特性

芯片中的 GPIO 管脚数是 27。每个管脚能被配置成通用输入输出管脚或者输出或者与内不芯片功能复合使用。

GPIO 管脚 18 和 23 能用作 2 级中断源并且每个中断能够配置成上升下降沿或者高低电平检测模式。

在复位以后，和 SCD 复合的功能将被初始化到 SCD 功能而其它的管脚将初始化到通用 I/O 。

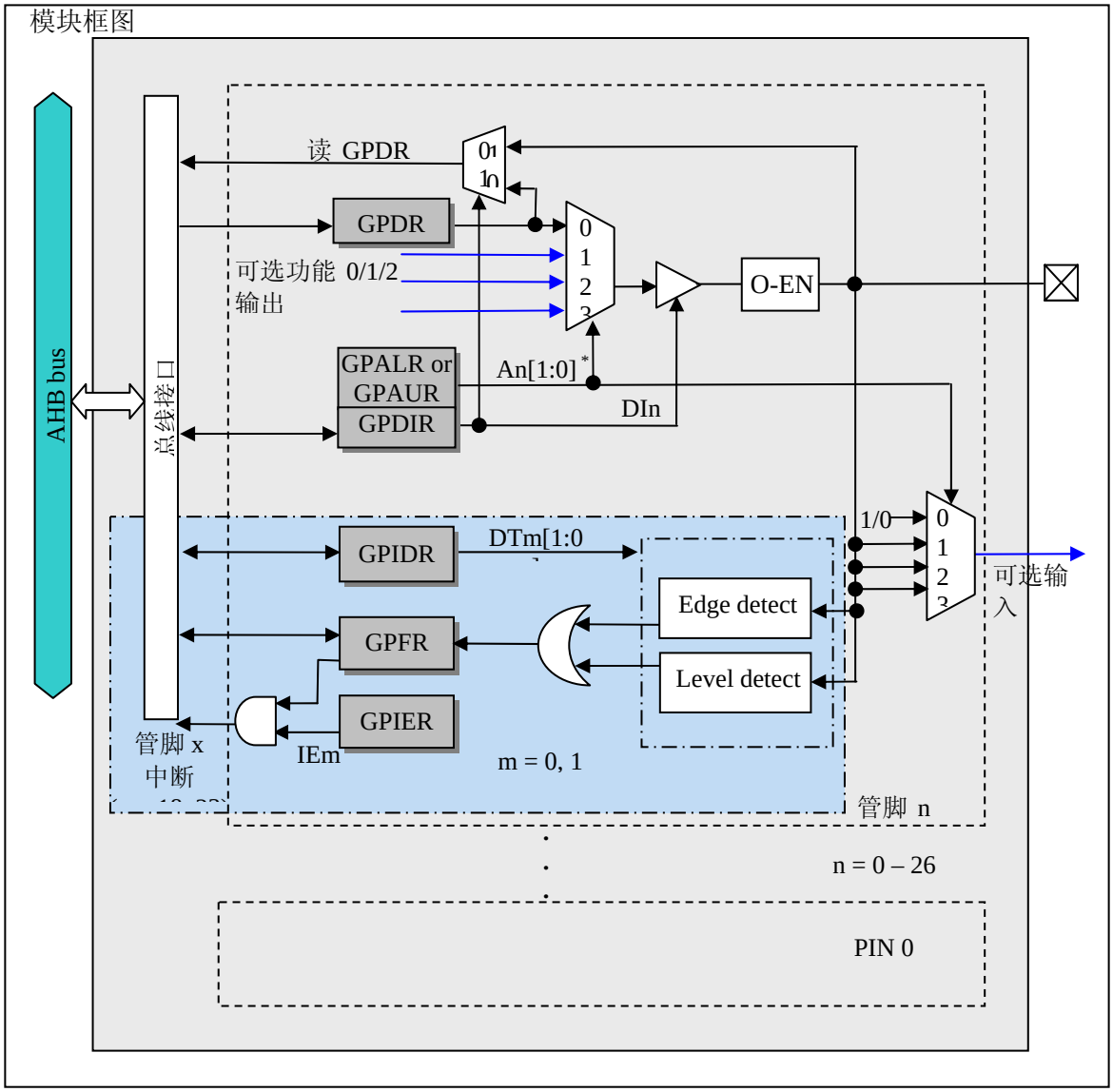


图 14-1 GPIO 模块框图

注意：

只有 GPIO 管脚 18 和 23 有中断功能。在 GPIER 的相应位 0 和 GPFR 作用管脚 18 并将位 1 作用管脚 23，GPIDR [1:0]作用于管脚 18 和 GPIDR [3:2]作用于管脚 23。从 GPIO18 和 23 分别有两个中断源送到 INTC。

寄存器配置

表 14-1 GPIO 管脚寄存器

名称	全名	R/W	初始值	地址	寻址位宽
GPDR	GPIO 数据寄存器	R/W	0X00000000*1	0x1803F500	32
GPDIR	GPIO 方向寄存器	R/W	0X00000000	0x1803F504	32
GPALR	GPIO 功能选择 L 寄存器	R/W	0X00000000	0x1803F50C	32
GPAUR	GPIO 功能选择 U 寄存器	R/W	0X00000004	0x1803F510	32
GPIDR	GPIO 中断检测模式寄存器	R/W	0X00000000	0x1803F514	32
GPIER	GPIO 中断使能寄存器	R/W	0X00000000	0x1803F518	32
GPFR	GPIO 中断标志寄存器	R/W	0X00000000	0x1803F51c	32

注意:

这值是 GPDR 寄存器的初始值，但是复位后，读 GPDR 将得到大部分管脚值，这是因为大部分的 GPIO 管脚已经初始化到输入方向。

对于可选功能寄存器，它指明了 L 被寄存器配置 GPIO 的低 16 位和 U 指明了被寄存器配置的剩下的高 11 位。

14.1.2 GPIO 端口数据寄存器 (GPDR)

GPIO 端口数据寄存器 (GPDR)是一个 32 位寄存器总是存储通过 AHB 总线接口写入的数据。GPDR 位 26～0 代表 GPIO 的 26～0 的管脚，位 31～27 保留。当某位配置成输入管脚，当前的管脚将被读出。否则，这位将保存在 GPDR 的值将被读出。这位寄存器可以一直被写而不管它的方向和功能。

位:	31	…	27	26	…	2	1	0
读:		…			…			

写:								
复位:	0	…	0	0	…	0	0	0

GPDR 通过复位将初始化到 0X00000000。这些是保留位。写到这些位无效并且读出始终为 0。

注意: 当任何一个 GPIO 管脚转换到内部芯片作为输入管脚或者 IO 管脚时，读 GPDR 将始终得到当前管脚 n 的值。

14.1.3 GPIO 端口方向寄存器 (GPDIR)

GPIO 端口方向寄存器 (GPDIR)是一个 32 位的负责控制管脚方向的寄存器。GPDIR 位 26～0 代表 GPIO 的 26～0 的管脚，位 31～27 保留。当某位配置成通用输入 GPALR/GPAUR.An [1:0] = 0 和 GPDIR.DIn = 0），GPIER.IEm 位决定当前的管脚功能是可编程输入还是外部中断输入。

位:	31	…	27	26	…	2	1	0
读:		…		DI	…	DI	DI	DI
写:								
复位:	0	…	0	0	…	0	0	0

GPDIR 通过复位将初始化到 0X00000000。这些是保留位。写到这些位无效并且读出始终为 0。

- **方向(DI):** DI 位定义了相应 GPIO 管脚的输入输出方向。

位 26-0: DIn	**描述**	
0	管脚 n 是输入	(初始值)
1	管脚 n 是输出	

注意: 当相应的 GPALR/GPAUR.An [1:0] ≠ 0 或者 GPIER.Iem 为 1 时，Din 没有效果。在这些情况下，硬件自动控制方向来适应这些功能。

14.1.4 GPIO 管脚功能转换 L 寄存器 (GPALR)

GPIO 管脚转换功能 L 寄存器 (GPALR) 是一个 32 位的负责控制管脚功能转换的寄存器。每一个 GPIO 管脚用 2 位去配置 GPIO，并且高达三种可选择功能（当某个管脚被定义为其它功能而不是正常的 GPIO 功能时，相应的 GPDIR 位将被忽略）。GPALR 位 31～0 代表 GPIO 的 15～0 的管脚。

位:	31	30	…	…	3	2	1	0
读:	A[1:0]		…	…	A[1:0]		A[1:0]	
写:								
复位:	0	0	…	…	0	0	0	0

GPALR 通过复位将初始化到 0X00000000。

- **可变功能控制 (A[1:0]):** 每 2 位指明是否为正常的 GPIO 的管脚或者被其三种可选功能的一种所替代。

位 31-0: An[1:0]	**描述**	
00	管脚 n 作为一个正常的 GPIO 接口。	(初始值)
01	管脚 n 改变为内部芯片功能 0。	
10	管脚 n 改变为内部芯片功能 1。	
11	管脚 n 改变为内部芯片功能 2。	

注意: 如果选择的内部芯片功能 0/1/2 是被管脚所保留的，这个管脚的功能将为通用 I/O 口。

14.1.5 GPIO 口可变功能 U 寄存器 (GPAUR)

GPIO 口可变功能 U 寄存器(GPAUR) 是一个 32 位的负责控制功能转变的寄存器。每一个 GPIO 管脚用 2 位去配置，高达三种可选择的功能（当某个管脚被定义为复合功能而不是正常的 GPIO 功能时，相应的 GPDIR 位将被忽略）。GPAUR 的位 21～0 代表 GPIO 的 26～16 的管脚。位 31～22 是保留位。

位:	31	…	5	4	3	2	1	0
读:		…	A[1:0]		A[1:0]		A[1:0]	
写:								
复位:	0	…	0	0	0	1	0	0

GPAUR 通过复位将初始化到 0X00000000。这些是保留位。写到这些位无效并且读出始终为 0。

- **可变功能控制 (A[1:0]):** 每 2 位指明是否为正常的 GPIO 的功能或者被其三种可选功能的一种所替代。

位 21-0: An[1:0]	描述	
00	管脚 n 作为一个正常的 GPIO 接口或者专门用作中断输入由 GPIER.Iem 位决定)。	(初始值)
01	管脚 n 改变为内部芯片功能 0。	
10	管脚 n 改变为内部芯片功能 1。	
11	管脚 n 改变为内部芯片功能 2。	

注意: 如果选择的内部芯片功能 0/1/2 是为管脚保留的，这个管脚的功能将为通用 I/O 口或者作为专用的中断输入（由 GPIER.Iem 位决定）。

14.1.5.1 可变功能列表

下面展示了 27 个 GPIO 管脚的可选功能。An [1:0] 代表 GPALR.An [1:0] 或者 GPAUR.An [1:0]。 当 An [1:0] = B’01, 管脚 n (n = 0-26)转变到它内部芯片功能 0；当 An [1:0] = B’10, 管脚 n 改变到它的内部芯片功能 1；当 An [1:0] = B’11, 管脚 n 改变到它的内部芯片功能 2。

GPIO 管脚 23 是特殊的 GPIO 管脚并且没有可选择的功能。对于其它的 GPIO 管脚，如果选定的内部芯片功能是保留的，GPIO 管脚 18 和 23 能作为专用中断输入（由 GPIER.Iem 位决定），对于剩余的 GPIO 管脚，他们的作用是通用 I/O 接口。

当一个管脚转变到它的内部芯片功能是，它相应的方向控制寄存器位 GPDIR.Din 将被硬件忽略。

表 14-2 GPIO 接口可变功能列表(总共 27 个)

GPIO Pin		7	6	5	4	3	2	1	0
An [1:0] = B’01	名称								
	I/O	IO	IO	IO	IO	IO	IO	IO	IO
An [1:0] = B’10	名称								
	I/O	IO	IO	IO	IO	IO	IO	IO	IO

An [1:0] = B'11	名称	保留	保留	保留	保留	保留	保留	保留	保留
	I/O	保留	保留	保留	保留	保留	保留	保留	保留

GPIO Pin		15	14	13	12	11	10	9	8
An [1:0] = B'01	名称	SCC0. SCC0_ CLK							
	I/O	O	O	I	O	O	O	O	O
An [1:0] = B'10	名称	SPI SPICLK Out	保留						
	I/O	O	保留	I	O	O	O	O	O
An [1:0] = B'11	名称		保留	保留	保留	保留	保留	保留	保留
	I/O	保留	保留	保留	保留	保留	保留	保留	保留

GPIO Pin		23	22	21	20	19	18	17	16
An [1:0] = B'01	名称		SCC1. SCC1_ DATA		SCC1. SCC1_ CLK	SPI SPIFra Out	SPI. SPI_ RxD	SCD. SCD_ DATA	SPI. SPI_ TxD
	I/O	保留	IO	O	O	O	I	IO	O
An [1:0] = B'10	名称	保留		保留				SCC0. SCC0_ DATA	
	I/O	保留	O	保留	O	保留	保留	IO	保留
An [1:0] = B'11	名称	保留	保留	保留	保留	保留	保留		保留
	I/O	保留	保留	保留	保留	保留	保留	保留	保留

GPIO Pin	26	25	24

An [1:0] = B'01	**名称**	UART. UART_ TxD	UART. UART_ RxD	
	I/O	O	I	O
An [1:0] = B'10	**名称**	保留	保留	保留
	I/O	保留	保留	保留
An [1:0] = B'11	**名称**	保留	保留	保留
	I/O	保留	保留	保留

14.1.6 GPIO 接口中断检测方式寄存器 (GPIDR)

GPIO 接口中断检测方式寄存器寄存器 (GPIDR)实现配置检测电平/上升下降沿中断模式。每个 GPIO18 和 23 的管脚用 2 位来配置中断检测方式(当某管脚的中断功能禁用时，相应的 GPIDR.DTm [1:0]位将被忽略。)GPIDR 的 3～0 位代表 GPIO 的 23 和 18 管脚而位 31~4 是保留位。

Bit:	31	30	…	4	3	2	1	0
读:			…		DT [1:0]		DT[1:0]	
写:								
复位:	0	0	…	0	0	0	0	0

GPDIR 通过复位将初始化到 0X00000000。这些是保留位。写到这些位无效并且读出始终为 0。

- **中断检测模式 (DT [1:0]):**当被配置为中断功能时，每 2 位将指明 GPIO 的中断检测模式。

位 3-0:DTm[1:0]	**描述**	
00	管脚 x (x = 18, 23)中断是低电平触发	(初始值)
01	管脚 x (x = 18, 23)中断是高电平触发	
10	管脚 x (x = 18, 23)中断是下降沿触发	
11	管脚 x (x = 18, 23)中断是上升沿触发	

14.1.7 GPIO 端口中断使能寄存器 (GPIER)

GPIO 端口中断使能寄存器寄存器 (GPIER)是为了当 GPIO 管脚作为通用目的 I/O 口(GPALR/GPAUR An [1:0] = 0)时使能 GPIO 的中断功能。GPIER 位 1-0 代表 GPIO 管脚的 23 和 18，位 31-2 是保留位。

位:	31	30	…	…	…	2	1	0
读:			…	…	…		IE	IE
写:								
复位:	0	0	…	…	…	0	0	0

GPIER 通过复位将初始化到 0X00000000。这些是保留位。写到这些位无效并且读出始终为 0。

- **中断使能 (IE):** IEm 位使能中断功能。

位 1-0: **IEm**	**描述**	
0	对管脚 X 禁止中断功能。	(初始值)
1	对管脚 X 使能中断功能（结果将反映到 GPFR 中）。	

注意: 当 GPIER.IEm = 1 时，管脚的方向将自动设置为输入， GPDIR.Dix 将被忽略。

14.1.8 GPIO 管脚中断标志寄存器 (GPFR)

GPIO 管脚中断标志寄存器是为了当管脚设置为中断时(GPIER.IEm is 1)反映在电平检测模式下的有效电平状态或者在沿检测模式下有效沿的锁存状态。对于电平检测 GPIO 的管脚模式，在 GPFR 的相应位只读。对于沿检测 GPIO 管脚的模式，写 1 到某位将对该位清 0，写 0 进去没有效果。GPFR 位 1-0 代表 GPIO 管脚 23 和 18，位 31-2 是保留位。

位:	31	30	…	…	…	2	1	0
读:			…	…	…		FLAG	FLAG
写:								
复位:	0	0	…	…	…	0	0	0

GPFR 通过复位将初始化到 0X00000000。这些是保留位。写到这些位无效并且读出始终为 0。

- **中断标志 (FLAG):** FLAG 展现了是否中断已经被检测出来。

位 1-0: FLAGm	描述	
0	管脚 X 当 GPIDR.DTm [1:0]指明检测模式时，没有检测到中断。	(初始值)
1	管脚 X 当 GPIDR.DTm [1:0]指明检测模式时，检测到了中断。	

注意: 禁止中断功能 (GPIER.IEm = 0)或者改变中断检测模式(通过改变 GPIDR DTm [1:0] 的位)将导致硬件自动清除 GPFR 的相应位。

第15章 同步串口(SPI)

15.1 同步串口 SPI

15.1.1 同步串口 SPI 特性

同步串口主要有以下特征：

工作在主模式下。

可编程的传输波特率

具有相互独立的接收 FIFO，发送 FIFO，都是 16 位宽，16 字深的

可以编程设置通信对象是 SPI,TI SSI

可编程设置每一帧的位数从 4 ~ 16 位

提供 3 个可屏蔽中断发送中断，接收中断，接收溢出中断

15.1.2 具有内部环路测试功能同步串口 SPI 描述

同步串口 SPI 是一个可以同 motorola SPI，TI SS 二种被广泛采用的同步串行接口相互通信的串口,在系统中可以作为主模式。

发送 FIFO

发送 FIFO 是一个 16 字深，16 位宽的先进先出缓冲器。在 SPI 复位后，SPI 使能以前可以向发送 FIFO 预装一些数据，一旦使能 SPI, 这些数据就顺序的串行发送出去。发送时，把读指针所指的单元的数据并行的置入发送移位寄存器 TSR 中，然后每来一个 SPICLKIN(从模式下)或 SPICLKOUT（主模式下）的有效沿，在 TSR 移位的同时发送出去一位，先发送最高有效位，最后发送最低有效位。发送 FIFO 空时，发送停止，即不允许发送数据，发送 FIFO 空的标志是写计数器和读计数器的值相同，即写指针和读指针指向统一个单元。发送 FIFO 满的标志是发送 FIFO 写计数器和读计数器的差值是 15。在发送 FIFO 满了以后，就不能再向发送 FIFO 写入数据了，如果仍然向寄存器 SPIDR 中写数据，刷新的是寄存器 SPIDR 的值，而不会更新发送 FIFO 中的值。

接收 FIFO

接收 FIFO 是一个 16 字深，16 位宽的先进先出缓冲器，其内部结构与发送 FIFO 一样。具体的操作过程与发送 FIFO 的操作过程是相反的。初始化以后，接收 FIFO 的读指针和写指针都是指向 0 单

元的。启动后，接收移位寄存器 RSR 每一个同步时钟的有效沿就从外设收到一位数据，然后移位，接收完成一帧后，把接收移位寄存器 RSR 的数据并行置入接收 FIFO 写指针当前所指的单元。ARM core 或 SDMA 读数据寄存器 SPIDR 相当于从接收 FIFO 中取数据，当检测到读 SPIDR 后，接收 FIFO 中当前读指针所指的单元的数值并行置入 SPIDR，然后被取走。当接收 FIFO 满了，如果再接收到数据就不能置入接收 FIFO 了，接收移位寄存器的值不断被覆盖，同时产生接收溢出中断。

15.1.3 同步串口 SPI 寄存器描述

同步串口 SPI 寄存器组中包括 6 个寄存器，如下图所示。

SPI 基地址：0x18002E000

偏移地址	类型	宽度	复位值	名 字	描 述
SPI Base +0x00	读/写	16	0x00	SPICR0	控制寄存器 0
SPI Base +0x04	读/写	5	0x00	SPICR1	控制寄存器 1
SPI Base +0x08	读/写	16	0x--	SPIDR	接收 FIFO（读） / 发送 FIFO 数据寄存器（写）
SPI Base +0x0c	读	5	0x00	SPISR	状态寄存器
SPI Base +0x10	读/写	8	0x00	SPICPSR	时钟预定标寄存器
SPI Base +0x14	读/写	3/0	0x00	SPIIIR/ SPIICR	中断识别寄存器（读）/ 中断清除寄存器（写）
SPI Base +0x18-0x3c			-		保留
SPI Base +0x40-90					保留
SPI Base +0x94-ff					保留

同步串口中寄存器组的地址安排

15.1.3.1 SPICR0 [16](+0x00)

SPICR0 是 0 号控制寄存器 ，共有 5 部分，复位后各位清 0。

位	名称	类型	功 能
15：8	SCR	读/写	串行时钟频率。SCR 是用于产生有效的 SPI 传输波特率的，这个频率也是同步时钟的频率。传输波特率的公式为： $\frac{FSPICLK}{CPSDVSR*(1+SCR)}$ CPSDVSR 是在 SPICPSR 寄存器中设定的，是一个 2~254 的偶数，SCR 的值是一个 0~255 的数。
7	SPH	读/写	SCLKOUT phase。 这一位是用以控制在于 Mototola SPI 通信是的时钟相位的。
6	SPO	读	SCLKOUT polarity。 这一位是用以控制在于 Mototola SPI 通信是的时

		/写	钟极性的
5:4	FRF	读/写	Frame format。用于选择通信对象的类型。 00 Motorola SPI frame format 01 TI synchronous serial frame format 10 Reserved 11 Reserved ,
3:0	DSS	读/写	Data Size Select: 用于选择传输一帧的数据位数。 0000 Reserved,undefined operation 0001 Reserved,undefined operation 0010 Reserved,undefined operation 0011 4-bit 0100 5-bit 0101 6-bit 0110 7-bit 0111 8-bit 1000 9-bit 1001 10-bit 1010 11-bit 1011 12-bit 1100 13-bit 1101 14-bit 1110 15-bit 1111 16-bit

15.1.3.2 SPICR1 [7]（+0 x04）

SPICR1 是 1 号控制寄存器，包括五个部分。

位	名 称	类型	功 能
15：7	--	--	保留
6	SOD	--	保留(设置成 0)
5	MS	--	保留(设置成 0)
4	SSE	读/写	Synchronous serial port enable 0 = SPI 操作不使能 1 = SPI 操作使能
3	LBM	读/写	Loop back mode: 0 = SPI 正常工作模式 1 = SPI 环路测试状态，在这种状态下，SPITxD 与 SPIRxD 在内部连接到一起。从串口发出去的数据，被串行的输入自己的接收端，进而置入接收 FIFO 中，可以被取到 ARM core 中进行检测，在上电复位以后，要检测 SPI 内部的数据路径是否正常，就要用这种方法。
2	RORIE	读/写	Receicve FIFO Overrun interrupt enable: RORIE = 0 接收 FIFO 溢出不使能。这位被置为 0 后，在发生了接收溢出后，不产生接收溢出中断 SPIRORINTR。 如果接收溢出中断已经产生后，清除 RORIE 位可以清除接收溢出中断位。 RORIE = 1 。接收溢出中断使能。

1	TIE	读/写	Transmit FIFO interrupt enable. TIE= 0 发送 FIFO 在半满或更少的时候，不产生发送中断 SPITXINTR TIE= 1 发送中断使能。发送 FIFO 在半满或更少的时候，产生发送中断 SPITXINTR
0	RIE	读/写	Receive FIFO interrupt enable. RIE= 0 接收 FIFO 在半满或更少的时候，不产生发送中断 SPIRXINTR TIE= 1 接收中断使能。接收 FIFO 在半满或更少的时候，产生接收中断 SPIRXINTR

15.1.3.3 SPIDR[16]（+0x08）

SPIDR 是 16 位宽的数据寄存器。

当 CPU 想从通过 SPI 接收外设的数据时，CPU 读寄存器 SPIDR， 同时接收 FIFO 的当前读指针所指的单元中的一帧数据被并行置入 SPIDR 中，被 CPU 从 SPIDR 中取到存储器或 CPU 中。SPI 从 SPIRxD 端串行接收数据，在接收到一帧数据后，并行的打入接收 FIFO 中的当前写指针所指的单元中，然后当前指针指向下一个单元，直到接收 FIFO 被写满。

当 CPU 欲通过 SPI 向外设发送数据时，CPU 就向 SPIDR 写数据, 这帧数据立即从 SPIDR 并行置入发送 FIFO 的当前写指针所指的单元。在上一帧数据从 SPITxD 发送出去以后，发送 FIFO 中当前读指针所指单元的数据并行置入发送移位寄存器，读指针指向下一个单元，然后数据从 SPITxD 端串行发送出去，在发送完一帧数据后，再置入下一帧数据到移位寄存器中，直到发送 FIFO 被发空。

接收 FIFO 和发送 FIFO 都是 16 字深,16 位宽的，但是所发送或接收的数据可能时 4~16 位的，用户需要把写到发送 FIFO 中的数据右对齐,发送的时候忽略高位无效数据,从外设接收来的数据,也要右对齐。在与 National 的 Microwire 接口通信的时候，SPI 的发送数据时固定的 8 位，接收的数据的位数时 4~16 可编程的。

SPI 复位后 SPIDR 为不定值。在 SPI 不使能的时候（SSE=0），可以向发送 FIFO 预先写入多个数据，一旦使能就可发送出去。

位	名称	类 型	功 能
15：0	DATA	读/写	接收数据（读）,通过 SPIDR 从接收 FIFO 中接收数据 发送数据（写）,通过 SPIDR 向发送 FIFO 中写数据

15.1.3.4 SPISR[5]（+0x0c）

SPISR 是一个对程序员来讲只读的状态寄存器，主要是来指明 SPI 的填充状态，和是否忙的状态寄存器。

位	名 称	类型	功 能
15：5	--	- -	保留
4	BSY	读	SPI 的忙标志 BSY = 0 SPI 是空闲的。 BSY = 1 说明 SPI 正在发送和/或接收数据，或者是发送 FIFO 不空。
3	RFF	读	接收 FIFO 满时该位置 1，否则置 0
2	RNFF	读	接收 FIFO 不满时该位置 1，否则置 0
1	TNF	读	发送 FIFO 不满时该位置 1，否则置 0
0	TFE	读	发送 FIFO 空时该位置 1，否则置 0

15.1.3.5 SPICPSR[8] (+0x10)

SPICPSR 是时钟预定标器寄存器（clock prescale register），主要的作用是给系统时钟 SPICLK 提供一个分频因子，来与控制寄存器 0 的 SCR 一同来产生同步时钟信号。在编程的时候，SPICPSR 在内部是作为一个 2~254 之间的偶数来用的，最低位被强行设置为 0，

位	名 称	类型	功 能
15：8	--	- -	保留
7:0	CPSDVSR	读/写	Clock prescale divisor. 在写的时候被设为一个 0~254 之间的偶数。 在读这个寄存器的时候，第 0 位被强行设为 0

15.1.3.6 SPIIIR/SPIICR[3/0]（+0x14）

SPIIIR 是中断标识（寄存器，SPIICR 中断清除寄存器。这两个寄存器的物理上是一个，地址也是一样的。这个地址空间具有不同的功能，向 SPIICR 写任何一个值都会清除接收溢出中断。复位时 SPIIIR 所有位为 0。

位	名 称	类 型	功 能
15：0	- -	写	无论写何值， 接收溢出中断状态被清除
15：3	- -	读	保留
2	RTIS	读	接收 FIFO 溢出中断 SPIRTINTR 产生时，此位置 1，否则置 0
1	TIS	读	发送请求中断 SPITINTR 产生时，此位置 1，否则置 0
0	RIS	读	接收请求中断 SPIRINTR 产生时，此位置 1，否则置 0

第16章 通用异步串口 UART

16.1.1 通用异步串口 UART 的特性：

发送与接收通道各有 16 字节深的 FIFO，以减少对 CPU 工作的中断；

FIFO 深度可由程序控制为 1 个字或 16 个字；

可编程的波特率产生器。对参考时钟可进行 2×16 到 65536×16 分频，并产生一个内部 16 倍波特率时钟；

提供标准的异步通讯位（起始位、停止位和奇偶位）。

提供发送中断、接收中断、接收超时中断、调制解调状态中断可独立屏蔽；

具有断线的产生与检测功能；

具有起始位有效性检测功能；

在 UART 控制器中，以下参数是可编程的：

波特率分频系数（可从 1 到 65535）

一帧的数据位的位数（5、6、7 或 8 位）

帧停止位个数（1 位或 2 位）

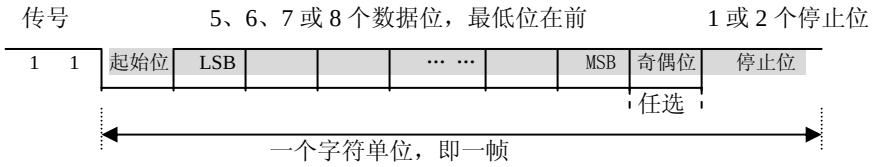
奇偶模式（奇模式（奇偶位为 1）、偶模式（奇偶位为 0）或不使能奇偶模式（无奇偶位））

FIFO 深度可编程，使能（16 个字深度）或不使能（1 个字深度）

工作模式或闭环测试模式

16.1.2 通用异步串口 UART 协议说明

异步串行传输格式如下图所示。



1. 异步串行传输格式

一帧数据除表示字符信息的数据位（位长度 5—8）可选外还有若干附加位：起始位（1 位、值恒为 0）、奇偶位（可选择有无）、停止位（长度为 1，1.5 和 2 位可选，值恒为 1）。传送一个字符必须以起始位开始，以停止位结束。这个过程称为一帧。除此之外，异步通信协议还规定：信号“1”称为传号（或称标记状态 MARK），信号“0”称为空号（或称间隔状态 SPACE）。

16.1.3 通用异步串口 UART 描述：

UART 的主要功能是把从存储器或处理器中并行传输传来的数据串行的发送到外设的 UART 接收端，或从外设的 UART 串行接收来的数据转换为并行数据提供给处理器传输到存储器中。UART 控制器采用全双工方式，发送器和接收器可同时工作，完成点到点的双向数据传输。为了减少对 CPU 核的中断次数，提高数据的传输效率， UART 控制器的输入和输出通道上都配置了 16 个字深度的 FIFO，作为发送数据或接收数据的缓冲器。

16.1.4 通用异步串口 UART 寄存器说明

UART 控制器的寄存器共有六个。见下表所示。

每个寄存器在存储器中映射有地址，寄存器组的基址是可变的，但相对地址是不变的，每个寄存器都有其独有的地址。寄存器组的地址分配见下表所示。

在寄存器组中，每个寄存器都是 8 位的, 线路控制寄存器 UARTLCR 为 24 位，实际上占用三个地址空间。

UART 基地址：0x1802E400

偏移地址	类型	宽度	复位值	名 字	描 述
+0x00	读/写	8	0x--	UARTDR	数据寄存器
+0x04	读/写	4/0	0x00	UARTRSR/ UARTECT	接收状态寄存器（读）/ 错误清除寄存器（写）

+0x08	读/写	7	0x00	UARTLCR_H	线路控制寄存器的高位字节
+0x0c	读/写	8	0x00	UARTLCR_M	线路控制寄存器的中间字节
+0x10	读/写	8	0x00	UARTLCR_L	线路控制寄存器的低位字节
+0x14	读/写	8	0x00	UARTCR	控制寄存器
+0x18	读	8	0x9-	UARTFR	标志寄存器（只读）
+0x1c	读/写	4/0	0x00	UARTIIR/ UARTICR	中断识别寄存器（读）/ 中断清除寄存器（写）
+0x24	读/写	4	0x00	UARTREIER	接收错误中断使能寄存器
+0x284-ff					保　　留

寄存器组的地址安排

下面逐个介绍寄存器每位的功能。在描述寄存器时采用这样的格式：

寄存器名[位宽]（偏移地址）

16.1.4.1 UARTDR[8]（+0x00）

UARTDR 是 8 位宽的数据寄存器，复位时为不定态。

对于数据发送过程，数据先写到 UARTDR，然后 UARTDR 中的数据立即送入发送 FIFO 中。

对于接收过程，有效的 8 位数据位从接收 FIFO 并行装入 UARTDR 中。CPU 通过读寄存器 UARTDR 获得 FIFO 中的数据。

位	名称	类 型	功 能
8：0	DATA	读/ 写	接收数据（读） 发送数据（写）

16.1.4.2 UARTRSR/UARTECR[4/0]（+0x04）

接收状态从 UARTRSR 中读得，若对寄存器 UARTECR 写操作，则帧格式错误、奇偶错误、中断错误和接收溢出错误位被清除。所有位在复位后均为 0 值。

位	名　称	类型	功　　能

7:0		写	写操作清除帧格式错误、奇偶错误、中断错误和溢出错误位，至于写的数值并不重要
7:4		读	保留，读的时候值为 0
3	Overrun Error (OE)	读	接收溢出错： 若在接收 FIFO 满时又接收到数据则此位置 1 当对 UARTECR 写时此位清零。 当 FIFO 满时又接收到数据，FIFO 中的数据保持有效，只是移位寄存器中的数据被覆盖。这时 ARMcore 应读接收 FIFO 以清空接收 FIFO
2	Break error (BE)	读	终止错： 若接收数据输入持续为低的时间超过了一个完整的帧格式（包括起始位、数据串、奇偶位和停止位）的时间，即中止情况发生则此位置 1。 当对 UARTECR 写时此位清零。 当中止错误发生仅一个为 0 的字节送入 FIFO 中，下一个字节仅在接收数据输入变为 1 后且一个有效的起始位被接收到时才被接收。
1	Parity Error (PE)	读	奇偶校验错： 当此位置 1 表示接收到的数据的奇偶性和奇偶位不符。 当对 UARTECR 写时此位清零。
0	Rraming Error (FE)	读	帧停止位错： 当此位置 1 表示接收的一帧数据没有一个有效的停止位（有效的停止位为 1）。 当对 UARTECR 写时此位清零。

需要注意的是在从状态寄存器 UARTSR 中读出错误中断之前，先要把数据从 UARTDR 中读出去，这时 UARTSR 中的值是刚刚读出数据的错误状态。这个顺序是不能相反的。

16.1.4.3 UARTLCR_H[7](+0x08)

UARTLCR_H 是线路控制寄存器 UARTLCR 的高位字节。复位后各位清 0。

位	名 称	类型	功 能
7		读/写	保留

6：5	Word length (WLEN[1 ： 0])	读/写	这两位用于选择发送或接收时一帧中数据位的位数： 11=8 位 10=7 位 01=6 位 00=5 位
4	Enable FIFO (FEN)	读/写	若此位置 1，则发送或接收 FIFO 使能 FIFO 是 16 字深；否则发送或接收 FIFO 不使能，FIFO 只有单字深度
3	Two stop bits select (STP2)	读/写	若此位置 1，则帧格式中一帧有两个停止位；若此位置 0，则只有一个停止位； 接收逻辑不检测第二个停止位。
2	Even Parity select (EPS)	读/写	如此位置 1，则奇偶校验采用偶校验方式；否则采用奇校验方式。
1	Parity enable (PEN)	读/写	这一位置 1，则奇偶校验使能，发送和接收时要产生或接收奇偶位。
0	Send break (BRK)	读/写	若此位置 1，则中止当前帧的发送，插入发送至少一整帧长度的低电位，产生中止情况。为了产生中止中断，这位必须保持高电平至少一个帧长度以上。建议保持 2 个时钟帧长度；在产生中止情况时，接收方产生中止错误中断。 接收判别 break 时，当前帧接收完毕后，判断下一帧整个是不是低电平，如果时，则 break 判别成功； 如果遇到一下状况不会判别为 break ：如果在正在接收的一帧没有结束之前，就接收到了低电平，然后经过 1 个帧长度的低电平，就接收到了高电平；虽然整个接收线上的低电平的长度有一个帧长度，但是没有一个完整的帧是接收到了一帧低电平，所以判别不是 break，只是接收帧格式错，或奇偶错或停止位错。 break 的接收判别是以帧为单位的，接收到满一帧低电平才判别为 break; break 的发送是只要往 break 寄存器位写 1，就立即中止正在发送的帧，在发送线上发出低电平。

注：在发送和接收使能时（即 UART 控制器使能信号 UARTEN 为高）时，控制器 UARTLCR-H 中的 LBE、WLEN0、WLEN1、FEN、STP2、EPS、PEN 的值都不能改变；要改变它们的值只有在不使能 UART 控制器后再对 UARTLCR-H 的各位重新赋值。

BRK 位是可以随时由软件改变的。

16.1.4.4 UARTLCR_M[8]（+0 x0c）和UARTLCR_L[8]（+0 x10）

UARTLCR_H 和 UARTLCR_L 是线路控制寄存器 UARTLCR 的中间字节和低位字节。复位后各位清 0。这 16 位确定了波特率分频系数。

位	名 称	类 型	功 能
7：0	BAUD DIVMS[7：0]	读/写	波特率分频系数的高 8 位
7：0	BAUD DIVLS[7：0]	读/写	波特率分频系数的低 8 位

注：在发送和接收使能时（即 UART 控制器使能信号 UARTEN 为高）时，控制器 UARTLCR-M、UARTLCR-L 中的波特率分频系数不能改变。

刷新 UARTLCR 的 3 个寄存器的顺序是有要求的,有两种顺序

- 先 UARTLCR_L 再 UARTLCR_M 最后 UARTLCR_H
- 先 UARTLCR_M 再 UARTLCR_L 最后 UARTLCR_H

若低两个寄存器只刷新一个，顺序如下

- 先 UARTLCR_L （或 UARTLCR_M ）然后 UARTLCR_H

16.1.4.5 UARTCR[8]（+0x14）

UARTCR 是控制器寄存器，在复位时所有位清零。

位	名 称	类型	功 能
7	Loopback enbable (lBE)	读 / 写	当 LBE=0,正常工作状态 当 LBE=1,发送端口和接收端口在内部连接到一起,用于完成闭环测试功能
6	RTIE	读 / 写	若此位为 1，则接收超时中断使能
5	TIE	读 /	若此位为 1，则发送中断使能

		写	
4	RIE	读 / 写	若此位为 1，则接收中断使能
3			保留
2			保留
1			保留
0	UARTEN	读 / 写	若此位为 1，则 UART 使能；否则 UART 不使能

16.1.4.6 UARTFR[8]（+0x18）

UARTFR 是标志寄存器，在复位时 TXFE 和 RXFE 为 1，其余位为 0

位	名 称	类 型	功 能
7	TXFF	读	发送 FIFO 满时该位置 1
6	RXFF	读	接收 FIFO 满时该位置 1
5	TXFE	读	发送 FIFO 空时该位置 1
4	RXFE	读	接收 FIFO 空时该位置 1
3	BUSY	读	发送 FIFO、发送缓冲器和发送移位器任一不空则该位置 1
2			保留
1			保留
0			保留

16.1.4.7 UARTIIR/UARTICR[8]（+0x1c）

UARTIIR/UARTICR 是中断辩识寄存器/中断清除寄存器。这个地址空间具有不同的功能，写 UA RTICR ，overrun 中断被清除。复位时 UARTIIR 所有位为 0。

位	名 称	类型	功 能
7：0		写	无论写何值，超时中断，接收中断，发送中断状态被清除
7	Overrun error interrupt status (OEIS)	读	当发生 overrun interrupt 时，这位被置 1 当写 UARTICR 时，overrun interrput 被清除，这位也被清除。

6	Break error interrupt status (BEIS)	读	当发生 break interrupt 时，这位被置 1 当接收 FIFO 中的所有字都没有 break 错时，break 中断被清除，这位也被清除。
5	Parity error interrupt status (PEIS)	读	当发生 Parity interrupt 时，这位被置 1 当接收 FIFO 中的所有字都没有 parity 错时，parity 中断被清除，这位也被清除
4	Frame error interrput status (FEIS)	读	当发生 Frame interput 时，这位被置 1 当接收 FIFO 中的所有字都没有 frame 错时，frame 中断被清除，这位也被清除
3	Receive timeout interrupt status (RTIS)	读	接收超时中断标志 中断 UARTRTINTR 产生此位置 1 当访问接收寄存器时超时中断清除，这个状态位也取消。
2	Transmit interrupt （TIS）	读	发送中断标志 中断 UARTTINTR 产生此位置 1
1	Receive intrerrupt (RIS)	读	接收中断标志 中断 UARTRINTR 产生此位置 1
0			保留

16.1.4.8 UARTREIER[4]（+0x24）

UARTREIER 寄存器是接收错误中断使能寄存器,复位时 UARTREIER 所有位为 0。

位	名 称	类型	功 能
7：4		读写	保留
3	Overun error interrupt enable (OEIE)	读写	当 OEIE 为 1， 使能 OEI 中断。 复位后这位是 0
2	Break error interrput enabel (BEIE)	读写	当 BEIE 为 1， 使能 BEI 中断。 复位后这位是 0
1	Parity error interrupt enable (PEIE)	读写	当 PEIE 为 1， 使能 PEI 中断。 复位后这位是 0
0	(Frame error interput enable (FEIE)	读写	当 FEIE 为 1， 使能 FEI 中断。 复位后这位是 0

第五部分 安全控制

第17章 DES 算法引擎

17.1 概述

DES 模块为 Z32U256 系列芯片中的一个模块。在 Z32U256 系列芯片中，以 AHB 总线为接口与 CPU 进行数据交换。在 CPU 的操作下，DES 模块完成：将 USB 接口接收的数据，进行加密或解密运算，然后再从 USB 接口发送出去。

CPU 通过 AHB 总线接口操作数据 DDAT 和密钥寄存器 DKEY 装入数据字节和密钥字节，然后在控制寄存器 DCNTRL 控制下、完成寄存器单 DES 或 3DES 的运算。

17.1.1 功能

实现 DES，3DES（2 KEY 和 3 KEY）加密解密运算

支持 EBC 模式和 CBC 模式的加密和解密

17.2 支持模式

DES 模块实现 DES、3DES、算法的加密和解密运算。

支持的模式有：

单密钥 ECB、2 密钥 ECB、3 密钥 ECB、单密钥 CBC、2 密钥 CBC 和 3 密钥 CBC。

17.3 寄存器说明

DES 模块提供 4 个 32 位 寄存器来实现与 AHB 总线接口，即 4 个寻址空间。通过 AHB 总线，CPU 操作这 4 个寄存器，控制 DES 模块的运行和交换数据。

17.3.1 数据寄存器(DDAT 32bit)

数据寄存器 DDAT 用于加解密数据的装入和结果数据的读出。每组数据 8 个字节，装入数据要顺序写入，如数据为 bit1~ bit64 时，应先写入 bit1~ bit32，再写入 bit33~ bit64。结果数据读出时，也要顺序读出，先读出 bit1~ bit32，再读出 bit33~ bit64。

简称	**地址**	**31~0**
DDAT		DDAT31~DDAT0
初始值		0

说明	R/W

- **DDAT31~DDAT0：**写操作时，装入数据到协处理器，由 LSB 开始，连续写 2 个双字为一个 BLOCK。读操作时，从协处理器读出数据，由 LSB 开始，连续读 2 个双字为一个 BLOCK。

17.3.2 密钥寄存器（DKEY）

密钥寄存器 DKEY 用于加、解密密钥的装入,长度为 32bit，DKEY0、DKEY8、DKEY16、DKEY24 为奇偶校验位。每个密钥为 8 个字节，要顺序写入，如密钥为 bit1~ bit64 时，应先写入 bit1~ bit32，再写入 bit33~ bit64。

简称	**地址**	**31~0**
DKEY		DKEY31~DKEY0
初始值		0
说明		W

- **DKEY31~DKEY0：**只允许写操作。装入加解密密钥到协处理器，由 LSB 开始。强制进行读动作时，数据不确定。

需用多密钥运算时，各密钥要顺序写入这一统一的密钥寄存器中。即 2 密钥运算时，要先写入密钥 2，再写入密钥 1。3 密钥运算时，要先写入密钥 3，再写入密钥 2。再写入密钥 1。

17.3.3 控制寄存器（DCNTRL）

控制寄存器 DCNTRL 用于启动加解密运算以及一些特殊条件的控制。

简称	**地址**	**31~10**	**9~8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DCNTRL		保留	BLE_MSLB	RUN	ENCRY	DES	KEY	EDR1	EDR0	ECB	保留
初始值		0	0	0	0	0	0	0	0	0	0
说明		R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

- **BLE_MSLB：**CPU 对数据寄存器 DDAT、密钥寄存器 DKEY、初始向量寄存器 DESIV 进行 32bit 读写操作时，可根据 BLE_MSLB 的配置，实现 32bit 数据的大小端调整和高低位调整。

BLE_MSLB	功　能 设 CPU 读写数据为{bit31~bit24，bit23~bit16，bit15~bit8，bit7~0} 则调整后的结果为：
00	不变。{bit31~bit24，bit23~bit16，bit15~bit8，bit7~0}
01	高低位调整。{bit24~bit31，bit16~bit23，bit8~bit15，bit0~7}
10	大小端调整。{bit7~0，bit15~bit8，bit23~bit16，bit31~bit24}
11	大小端和高低位都调整。{ bit0~7，bit8~bit15，bit16~bit23，bit24~bit31}

- **RUN**：启动 DES 协处理器，软件置位，硬件清除。

RUN	功　能
0	运算结束，协处理器处于空闲状态。
1	启动 DES 协处理器后，保持到此次运算结束。

- **ENCRY**：加密/解密指示位。

ENCRY	功　能
0	进行加密运算
1	进行解密运算

- **DES**：DES/TDES 指示位。

DES	功　能
0	进行 DES 运算
1	进行 TDSE 运算

- **KEY**：TDES 运算中，2KEY/3 KEY 的指示位。DES 运算时，此位无效。

KEY	功　能
0	2 密钥运算
1	3 密钥运算

- **EDR1, EDR0**：指定执行的 DES 回合数

EDR1	EDR0	功　能
0	0	执行 16 个 DES 回合
0	1	执行 1 个 DES 回合
1	0	执行 2 个 DES 回合
1	1	执行 3 个 DES 回合

- **ECB**：ECB/CBC 模式指示位。

ECB	功　能
0	ECB 模式
1	CBC 模式

17.3.4 初始向量寄存器(DESIV)

初始向量寄存器 DESIV 用于存储 CBC 模式加、解密所需的初始向量数据。只在 CBC 模式使用，ECB 模式时，即使写入数据，也不起作用。

简称	**地址**	**31~0**
DESIV		DESIV31~DESIV0
初始值		0
说明		W

- **DESIV31~DESIV0**：初始向量数据。只允许用户进行写操作。初始向量数据为 64bit，如同加解密数据一样，分两次写入。

第18章 公钥算法引擎(PAE)

18.1 概述

公钥算法引擎（PAE）模块实现 RSA 和 ECC 等公钥算法所必须的模乘、模幂运算。

18.1.1 技术指标

关键的技术指标如下：

支持 128～1024bit 大数乘法、模幂运算。

核心算法基于 Montgomery 算法，支持 H 值计算。

支持 AHB 总线与运算核双时钟，运算核时钟可以是 AHB 总线时钟的 1、2、4 倍，最高 96MHz。

计算 1024bit 模幂最大计算速度为 23 次/秒/96MHz 主频。

不启动运算时，同 EPI 模块共用的内部 1kByte RAM 可用作数据缓冲区。

AHB 总线接口，支持中断方式。

PAE 详细使用方法请参考应用解决方案文档。

第19章 随机数发生器(RNG)

19.1 概述

Z32U256 系列内部有 1 个 32 位的随机数发生器用于密码。随机数发生器模块由控制寄存器 RNGCTRL、数据寄存器 RNGDATA、控制电路以及模拟电路 rng_ip 组成。模拟电路 rng_ip 采用非线性的 RC 振荡以产生非固定频率的时钟，发生器的初始值不定。

随机数发生器工作通过随机数控制寄存器 RNGCTRL 设定并启动，产生的随机数保存在数据寄存器 RNGDATA。

19.2 特性

支持真随机、伪随机两种模式

伪随机的种子可由真随机（模拟电路 rng_ip）产生

真随机数输出速率为 50kbps～100kbps

伪随机数输出速率为 AHB 速率，最高可到 100Mbps

产生的随机数每 32bit 存入 RNGDATA 并通知 CPU 读取

当 RNGDATA 的数没有被 CPU 取走时，随机数输出将暂停，等候 CPU 把数取走再输出。

当 RNGDATA 的数准备好时，RNGF 标志置 1（供 CPU 查询），并产生中断

19.3 随机数功能寄存器

表 19-1 RNG 寄存器

名称	描述	R/W	初始值	地址	位宽
RNGCTRL	控制寄存器	R/W	0X0FA0	0X1802F000	15
RNGDATA	数据寄存器	R	0X00000000	0X1802F004	32

19.3.1 控制寄存器（RNGCTRL）

控制寄存器 RNGCTRL 控制发生器的启动/停止等。

Bit:	31~15	14	13	12	11	10	9	8
Read:		TESTMODE	TESTVALUE	DIVPAR				
Write:								
Reset:	0	0	0	0	1	1	1	1

Bit:	7	6	5	4	3	2	1	0
Read:	DIVPAR				PNSEL	IPEN	RNGF	RUN
Write:								
Reset:	1	0	1	0	0	0	0	0

- **TESTMODE：**——测试模式选择。
  - 当＝1，读 **RNGDATA** 的值为 32’hFFFFFFFF 或 32’h0（由 TESTVALUE 决定）
  - 当＝0，为正常模式，读 **RNGDATA** 的值为随机数
- **TESTVALUE：**——测试值。在测试模式下（TESTMODE＝1），
  - 当＝1，读 **RNGDATA** 的值为 32’hFFFFFFFF
  - 当＝0，读 **RNGDATA** 的值为 32’h0
- **DIVPAR：**——分频参数，将 AHB 时钟分频至 200~400KHz 提供给模拟真随机 IP 模块用。初始值 250（9’hFA）
- **PNSEL：**——真、伪随机数选择。
  - 当 PNSEL＝1，产生伪随机数
  - 当 PNSEL＝0，产生真随机数
- **IPEN：**——模拟电路 rng_ip 模块使能
  - 当 IPEN =0 时，模拟电路 rng_ip 模块不工作；
  - 当 IPEN =1 时，模拟电路 rng_ip 模块正常工作。
  - 当 PNSEL＝1 且 IPEN =1 时，伪随机数的种子由模拟电路 rng_ip 模块产生；

- 当 PNSEL＝1 且 IPEN =0 时，完全伪随机。

➢ **RNGF**：——随机数产生电路的状态标志位。

- RNGF =0，忙。**RNGDATA** 中随机数无效;
- RNGF =1，**RNGDATA** 中随机数有效。

➢ **RUN:** ——RNG 控制位。

- RUN =0，RNG 停止；
- RUN =1，RNG 启动。

19.3.2 数据寄存器（RNGDATA）

数据寄存器 RNGDATA 用于存储随机数据。

Bit:	31	30	……	4	3	2	1	0
Read:	RNGDATA							
Write:								
Reset:	0	0	0	0	0	0	0	0

➢ RNGDATA[31:0]——随机数。

第20章 安防模块(SEC)

20.1 概述

安防模块 SEC 用于高低频率检测、高低电压检测，防止攻击，将当前状态通知 CPU，并可产生保护性复位。安防模块由控制寄存器、状态寄存器、控制电路、模拟电路 100k 频率参考源、模拟电路高低电压检测组成。

20.2 特性

可分别控制频率检测、高电压检测、低电压检测使能 ；

可分别控制产生系统复位的使能 ；

若输入电压低于 2.5V 或高于 5.8V，系统会检测到非正常电压，并设置状态寄存器的相应位；

若被检频率在高低阈值范围之外，系统会检测到非正常频率，并设置状态寄存器的相应位；

可设置频率检测高低阈值，缺省值为高阈值 5MHz、低阈值 1MHz；

通过状态寄存器显示当前检测状态提供给 CPU；

产生中断告警；

第六部分 电气特性

第21章 电气特性

21.1 最高绝对限额

此部分提供 Z32U256 系列芯片的绝对最高限额，在实际操作的时候不要超过这些参数，否则将永久地损坏芯片。另外，在此范围内运行其功能也不能保证。

表 21-1 最高绝对限额

符号	描述	最小	最大	单位
TS	存储温度	-40	125	^{o} C
VCC	电源电压	2.5	6.0	V
VESD	最大 ESD 电压，人体模型		4000	V
IEOS	最大非电源引脚 DC 输入电流		5	mA

21.2 操作条件

此部分显示 Z32U256 系列芯片的电压、频率以及温度特性。

表 21-2 电压、温度以及频率电气特性

符号	描述	最小	典型	最大	单位
T_A	环境温度——正常温度	0	-	70	°C
V_{Vcc}	电源电压	2.7	5.0	6.0	V
V_{VDDIO}	VDDIO 电压	3.0	3.3	3.6	V
I_{VDDIO}	主频：160Mhz， VDDIO 电压为 3.3v			100	MA
V_{VDD}	VDDcore, AVDD, VDDG, VDDRTC, VDD 电压	2.275	2.5	2.75	V
I_{VDD}	Frequency:120Mhz， VDDIO=3.3v			200	MA
$F_{inter-cpu}$	内部 CPU 核频率范围			100	MHz
$F_{inter-PAE}$	内部 PAE 模块频率范围			100	MHZ
$F_{inter-DES}$	内部 DES 模块频率范围			100	MHZ
B_{SCD}	ISO7816 接口最高通讯带宽			310K	bps
B_{USB}	USB 接口最高通讯带宽			12M	bps
Cin	输入电容		5		PF

21.3 DC 参数

DC 特性包括每一个引脚地输入门限以及输出驱动电压及电流。这些参数能够决定最大的 DC 负载，并决定给定负载的条件下的最大的传送时间。下表显示了高低电压输入、输出以及 IO 引脚情况下的 DC 操作条件，所有的 DC 参数值在整个温度范围内有效。

表 21-3 标准输入、输出以及 IO 引脚 DC 操作条件

Symbol	Parameters	Min.	Nom.	Max.
VDD	Pre-driver supply Voltage	2.25V	2.5V	2.75V
VD33	I/O Supply Voltage	3.0V	3.3V	3.6V
Vil	Input Low Voltage	-0.3V		0.8V
Vih	Input High Voltage	2.4V		5.5V
Ii	Input leakage current @Vi=VD33 or 0V			±1uA
Ioz	Tri-state output leakage current@Vo=VD33or 0V			±10uA
Rpu	Pull-up resistor（GP16、GP17、TDI、TMS、SCD_RST）	74k	104k	177k
Rpd	Pull-down resistor（GP8、GP9、GP10、TAP_MD、TEST_EN、TEST_SEL2、RST_MD、TRSTN、TCK、CLK_MD、TEST_SEL1、TEST_SEL0 ）	43k	61k	119k
Vol	Output Voltage @Iol=5mA			0.4V
Voh	Output Voltage @Ioh=8mA	2.4V		
Iol1	Low level Output current @Vol=0.4V (All GPIOS except GP15、GP16、GP17、GP20、GP22)	3.5mA	5.7mA	7mA
Iol2	Low level Output current @Vol=0.4V (GP15、GP16、GP17、GP20、GP22)	6.9mA	11.3mA	13.9mA
Ioh1	High level Output current @Voh=2.4V (All GPIOS except GP15、GP16、GP17、GP20、GP22)	3.9mA	8mA	12.8mA
Ioh2	High level Output current @Voh=2.4V (GP15、GP16、GP17、GP20、GP22)	7.8mA	15.9mA	25.6mA

注：在现有的测试条件，某些参数是无法测出的，包括 Ioz、Rpu、Rpd。这些参数给出是所设计 PAD 的典型参数。

21.4 AC 参数

一个引脚地 AC 特性包括输入以及输出电容，它决定了外部驱动或其他驱动的负载分析。AC 特性包括一个 因子，它决定不同负载情况下的 AC 时序的快慢。下表显示了高低电压输入、输出以及 IO 引脚情况下的 AC 操作条件，所有的 AC 参数值在整个温度范围内有效。

表 21-4 标准输输入、输出以及双向端口 AC 操作条件

符号	描述	最小	典型	最大	单位

C_{IN}	输入电容，所有标准输入以及双向端口		5	10	pF
$C_{OUT\ H}$	输出电容，所有标准高强度输出以及双向端口	20		200	pF

注：在此负载范围内 AC 特性能够保证，所有的测试在 50pf 的条件下。

21.5 复位以及电源 AC 时序分析

Z32U256 系列芯片在下列情况下复位：

- 上电；
- 看门狗复位；
- 安防保护复位；
- 休眠模式。

21.6 USB2.0 全速收发器 AC/DC 特性

下表总结了 USB 收发器的功能需求，在 USB2.0 全速接口规范有详细的描述，Z32U256 系列 USB I/O 单元设计能够完全的满足这些要求。

测试条件是在室温情况下，TAVD33=3.3v 且 VDD=2.5v。

表 21-5 USB2.0 全速接口 DC 电气特性

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	Input high (driven)		2.0			V
V_{IL}	Input low				0.8	V
V_{DI}	Differential input sensitivity	$\|PADP - PADM\|$	0.2			V
V_{CM}	Differential common-mode range	includes V_{DI} range	0.8		2.5	V
V_{SE}	Single-ended receiver threshold		0.8		2.0	
	Receiver hysteresis			200		mV
V_{OL}	Output low (driven)		0		0.3	V
V_{OH}	Output high (driven)		2.8		3.6	V
V_{CRS}	Output signal cross voltage		1.3		2.0	V
R_{PU}	Pull-up resistor		1.425		1.575	$k\Omega$
R_{PD}	Pull-down resistor		14.25		15.75	$k\Omega$
V_{TRM}	Termination voltage for upstream port pull up (R_{PU})		3.0		3.6	V
Z_{DRV}	Driver output resistance	steady state drive*		10		Ω
C_{IN}	Transceiver capacitance	pin to GND			20	pF

*Driver output resistance doesn't include series resistor resistance.

表 21-6 全速情况下电气特性

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_{FR}	Rise time	$C_L = 50p$	4		20	ns
T_{FF}	Fall time	$C_L = 50p$	4		20	ns
T_{FRFF}	Rise and fall time matching	$T_{LRLF} = T_{LR}/T_{LF}$	90		111.11	%