ZTEIC

用“芯”捍卫安全

Z8D168 芯片

用户手册

Version 1.0

目　录

图 列 表

表 格 列 表

第一部分 系统概述

第1章 概述

Z8D168 系列是仙人球（Cacti）Z8 系列中一组带有 USB1.1 全速、ISO7816 主从接口和 SPI 接口的 8 位安全芯片，它与工业标准的 8051 单片机指令集完全兼容。Z8D168 片上集成了单周期高性能 8 位安全微控制器，拥有 256Byte 核内 RAM、3K 核外 RAM 和 1K 核外 PAE RAM，168K Byte Flash 程序/数据存储器。Z8D168 内置 DES/3DES、AES、SSF33、SCB2 和公钥算法引擎，内置了硬件真随机数发生器和安全防护单元，适用于 PKI 等安全应用。Z8D168 系列芯片具有功耗低、稳定性高和兼容性强等特点。

Z8D168 系列包含 Z8D168U、Z8D168C-Ⅰ和 Z8D168C-Ⅱ等，可应用于 USBKey 和智能 IC 卡等多个信息安全领域。

Z8D168 功能框图如 图 1-1所示：（具体接口配置详见产品芯片型号说明）

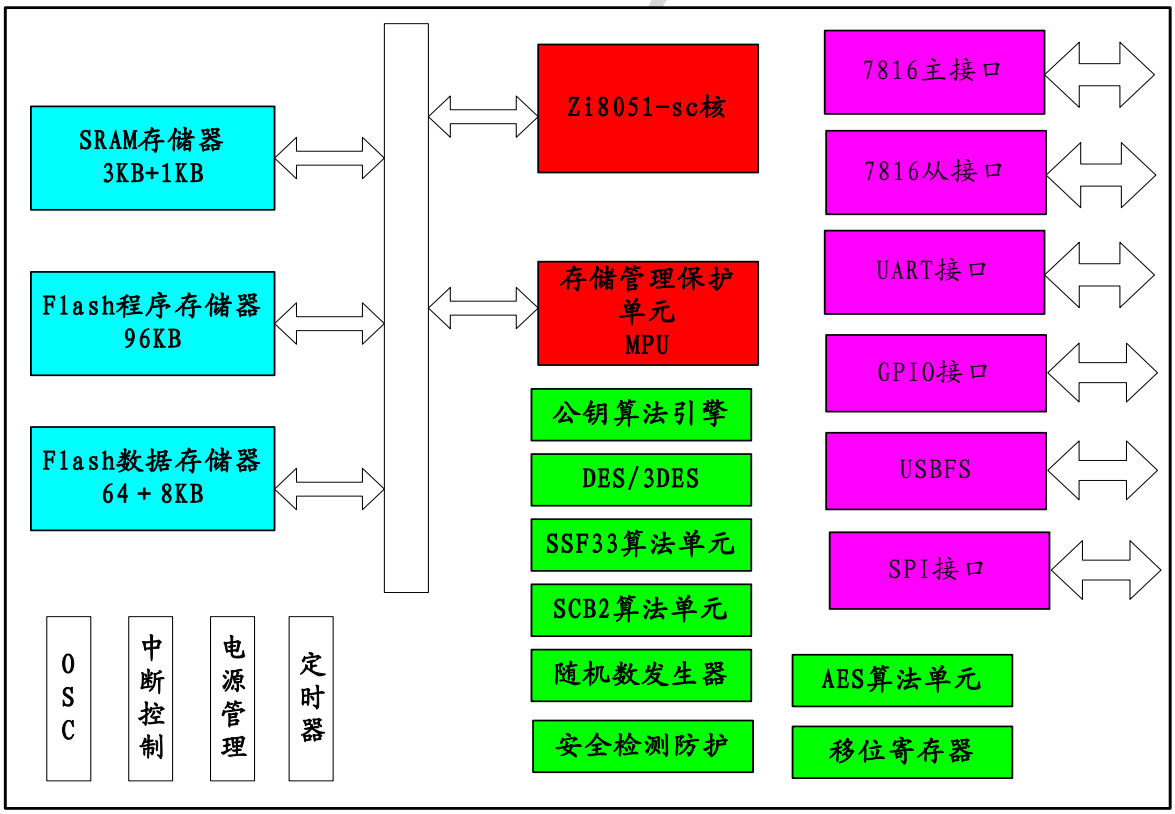


图 1-1 Z8D168 芯片功能框图

Z8D168 SSOP28 封装管脚图，如 图 1-2 所示：

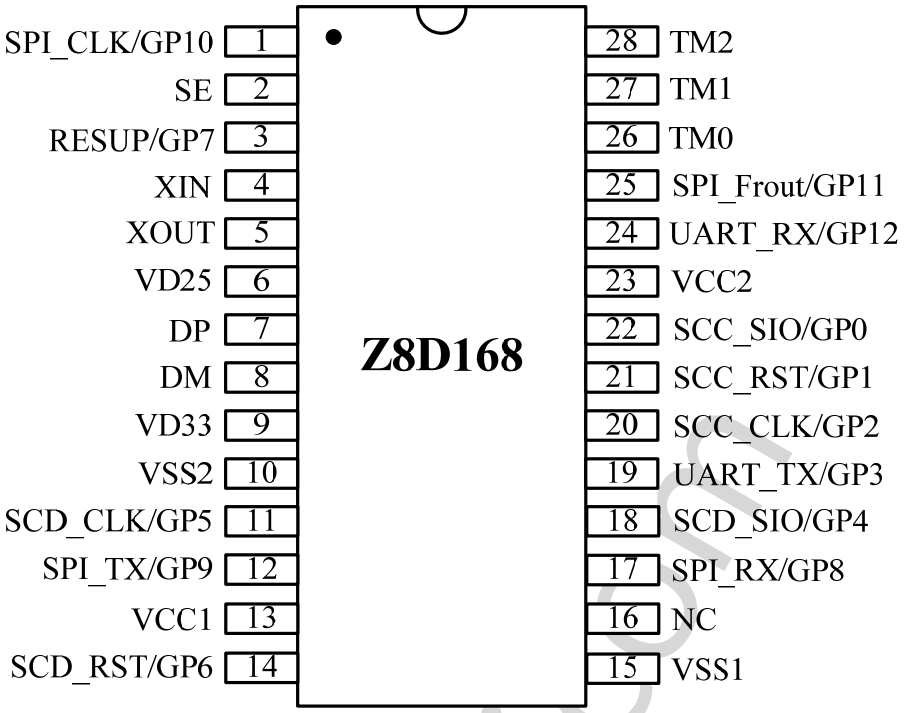


图 1-2 Z8D168 SSOP28 封装管脚分配图

各引脚对应功能如 表 1-1所示：

表 1-1 Z8D168 外部引脚信号定义

管脚名	管脚类型	功能描述
VCC1	电源输入	USB 应用： 4.5V~5.5V 电源； 7816 应用：2.7V~5.5V 电源
VCC2	电源输入	USB 应用： 4.5V~5.5V 电源； 7816 应用：2.7V~5.5V 电源 只需使用 VCC1 供电，本引脚悬空
VSS1、VSS2	地	
VD33	电源输出	3.3V 电压输出引脚，接 RC 电路到地，提高系统稳定性。不可用作外部芯片电源。
VD25	电源输出	2.5V 电压输出引脚，接 RC 电路到地，提高系统稳定性。不可用作外部芯片电源。
TM2	输入	芯片工作模式选择引脚，内部上拉 111，芯片工作在 7816 接口模式 101，芯片工作在 USB 接口全速模式 其他模式禁用
TM1	输入	
TM0	输入	
Xin	输入	外部无源晶体震荡输入端；参考外接电阻

		500K~2M 欧，参考外接电容 6pf~40pf
Xout	输出	外部无源晶体震荡输出端；
DP	输入\输出	USB 总线 D+模拟 IO。
DM	输入\输出	USB 总线 D-模拟 IO。
SCC_SIO/GP0	输入\输出	SCC 接口 IO 引脚；复用为 GPIO0
SCC_RST/GP1	输入\输出	SCC 接口复位引脚；复用为 GPIO1
SCC_CLK/GP2	输入\输出	SCC 接口时钟引脚；复用为 GPIO2
UART_TX/GP3	输入\输出	UART 接口 TX 引脚；复用为 GPIO3
SCD_SIO/GP4	输入\输出	SCD 接口 IO 引脚；复用为 GPIO4
SCD_CLK/GP5	输入\输出	SCD 接口 IO 引脚；复用为 GPIO5
SCD_RST/GP6	输入\输出	SCD 接口复位引脚；复用为 GPIO6
RESUP/GP7	输入\输出	USB 接口模式中作为 D+/D- 1.5K 欧姆上拉电阻的控制信号，复用为 GPIO7
SPI_RX/GP8	输入\输出	SPI 接口 RX 引脚；复用为 GPIO8
SPI_TX/GP9	输入\输出	SPI 接口 TX 引脚；复用为 GPIO9
SPI_CLK/GP10	输入\输出	SPI 接口 CLK 引脚；复用为 GPIO10
SPI_Frout/GP11	输入\输出	SPI 接口 Frout 引脚；复用为 GPIO11
UART_RX/GP12	输入\输出	UART 接口 RX 引脚；复用为 GPIO12

第2章 关键特性

2.1 处理器性能

- 单周期 Zi8051-SC 核
  - 指令集和工业标准 8051 指令集兼容
  - 增强型 8051 结构，指令周期为 1 个系统时钟周期
  - 中断结构：具有 5 个中断源，2 个优先级
  - 位寻址
  - 定时器：2 个 16 位可编程定时器
  - 看门狗定时电路
  - 低功耗模式：支持 IDLE 模式、深度主动休眠模式、深度被动休眠模式和主动唤醒休眠模式
  - 时钟频率：核时钟最大 20 MHz，PAE 运算时钟最大 40MHz
- 存储保护单元（MPU）
  - 支持总线加扰和解扰
  - 支持存储区安全访问控制

2.2 片上存储单元

- 168K Byte + 512 Byte Flash（Code Flash/ Data Flash + OTP Block）
  - 168K Byte Flash 用于程序、函数库、数据的存储
    - 8k Data Flash 页大小为 64 字节；160k Code Flash 页大小为 512 字节，可以配置作为 Data Flash
    - Data Flash 最少擦写次数 30 万次，Code Flash 最少擦写次数 10 万次
    - 室温（25℃）下数据保持时间最少为 10 年
    - 擦写性能
      - 单字节编程时间：小于 64us
      - 页擦除时间：小于 1.2ms
  - 512 Byte OTP Block 用于用户关键/敏感数据的存储
- RAM：256 Byte 核内 RAM + 3K Byte 核外 RAM+ 1K Byte 核外 PAE RAM

2.3 安全组件

- 公钥算法引擎
  - PAE 内部时钟：40MHz （最大）
  - 支持大数模乘（1024bit/2048bit）/模幂（1024bit）/乘法（1024bit）运算协处理
  - 支持多项式（511 位）模乘/乘法运算协处理
  - 1024 位 RSA 算法签名速度达到 95ms/次（不带 CRT）@CPU 20MHz, PAE 40MHz
  - 1024 位 RSA 算法签名速度达到 40ms/次（带 CRT） @CPU 20MHz, PAE 40MHz
  - 1024 位 RSA 密钥对生成速度达到 1.5s/次（带 CRT）@CPU 20MHz, PAE 40MHz
  - 1024 位 RSA 密钥对生成速度达到 1.8s/次（不带 CRT）@CPU 20MHz, PAE 40MHz
  - 192 比特 ECC 算法（p 域）点乘速度达到 0.5s/次以上@CPU 20MHz, PAE 40MHz
  - 192 比特 ECC 算法（2n 域）点乘速度达到 1s/次以上@CPU 20MHz, PAE 40MHz
- DES/3DES 引擎
  - 支持 DES、3DES（2 KEY 和 3 KEY）算法的加密和解密
  - 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 流加解密速度大于 3.0Mbps
- AES 算法引擎
  - 支持 AES 算法的加密和解密
  - 加解密速度大于 2.4Mbps
- SCB2 算法引擎
  - 支持 SCB2 算法的加密和解密
  - 加解密速度大于 2.4Mbps
- SSF33 算法引擎
  - 支持 SSF33 算法的加密和解密

- 加解密速度 2.4Mbps

- 随机数发生器
  - 内置真随机数发生器
- 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
  - 安全优化布线
  - 内部上电复位
  - 每一颗芯片具有唯一序列号

2.4 通讯接口

- USB1.1 全速控制器
  - 最高传输速率： FS/12Mbps
  - 最多支持 5 个硬件端点（1 个控制端点，2 个中断端点和 2 个 BULK 端点）
  - 控制端点 FIFO 深度 8 字节，中断端点 FIFO 深度 8 字节，BULK 端点 FIFO 深度 32 字节
  - 实际流加密速度最高可以达到 3.0Mbps
- ISO7816 主、从控制器
  - 具备 ISO7816 主、从控制器
  - 符合 ISO7816－3 标准，最高传送速率 300Kbps
- UART 控制器
  - 全双工

- 波特率： 9.6kbps ~115.2 kbps, 可编程

- SPI 控制器
  - 最高波特率 10Mbps
- GPIO
  - 双向可配置 GPIO
  - 13 个可配置 GPIO 和其他接口复用

2.5 电气特性

- 整个芯片最大功耗小于 48mA（5V@20M），芯片的平均功耗小于 18mA(5V@20M)，低功耗模式下低于 500uA
- 电源电压：2.7 ~5.5V
- ESD 保护：4KV 以上
- 符合 ISO7816-3 规范

2.6 芯片型号

	USB 接口	SCD 接口	SCC 接口	SPI 接口
Z8D168U	有	有	有	有
Z8D168C-Ⅰ	无	有	有	有
Z8D168C-Ⅱ	无	有	无	无

2.7 开发环境

- KEIL51 集成开发环境，推荐版本 Keil uVision2 V2.40a。
- 硬件实时仿真器
- 开发板

2.8 产品封装

- Wafer/DIE
- SSOP28
- 卡封装/module

2.9 应用产品

- 低成本身份认证 USBkey
- 电子签章
- 软件加密锁
- 智能卡

第二部分 主处理器单元

第3章 CPU核

3.1 概述

Z8D168 芯片采用了ZTEIC自行开发的Zi8051-SC 8 位安全MCU核，Zi8051-SC核是一款与工业标准 8051 指令集完全兼容的单周期高性能 8 位安全微控制器。该微控制器包括指令译码单元（IDU）、控制执行单元（EU）、算术逻辑单元(ALU)、中断处理单元（IPU）、2 个定时器/计数器、通用寄存器组(R0-R7)、特殊功能寄存器组（SFR）、核内RAM接口及核内 256 字节SRAM、扩展SFR总线接口、ROM总线接口、核外RAM总线接口和存储保护单元（MPU）。Zi8051-SC的指令集与工业标准 8051 指令集完全兼容，Zi8051-SC指令集详见 附录A 指令集。

3.1.1 模块图

Zi8051-SC核及存储部分的结构图如 图 3-1所示：

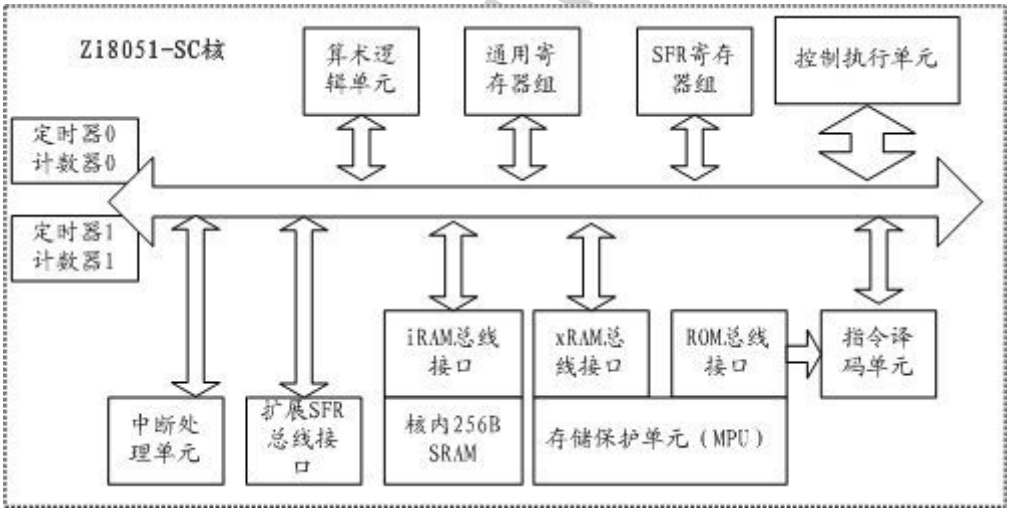


图 3-1 Zi8051-SC 内核及存储接口部分结构图

3.1.2 特性

- 程序存储器接口：程序存储器是 FLASH，支持多 Bank 技术
- 核内数据存储器接口：256 字节寻址空间，高 128 字节地址与 SFR 寄存器组共享

- 核外数据存储器接口：支持多 Bank 技术，寻址空间大于 64KB
- 两个 16 位的定时器
- 中断：5 个中断源（两个外部中断源和 3 个内部中断源），2 个优先级
- 指令集和工业标准 8051 指令集完全兼容
- 带看门狗定时电路
- 低功耗模式：支持 IDLE 模式、深度主动休眠模式和深度被动休眠模式以及主动唤醒休眠模式
- 时钟频率：最高达 20MHz
- 单周期指令，每个机器周期 1 个系统时钟周期(而工业标准 8051 每个机器周期 12 个系统时钟周期)

3.2 核内存储器

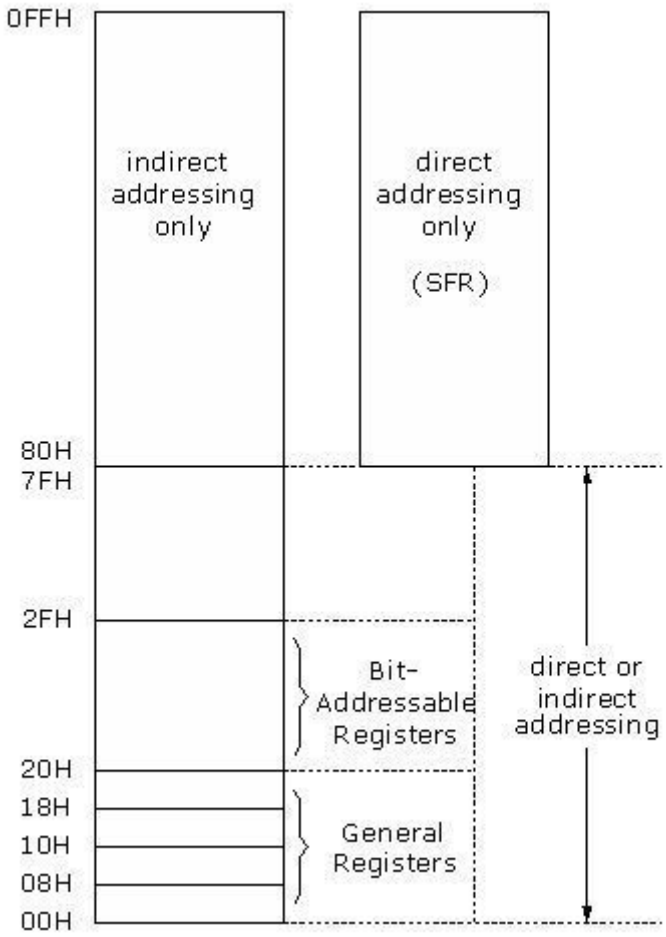


图 3-2 内部 RAM 空间分配

Zi8051-SC核具有 256 字节的内部随机存取存储器(RAM)，空间分配如 图 3-2所示。可以三种寻址方式使用：

- 低 128 字节(00H~7FH)可通过直接寻址或间接寻址方式来使用
- 高 128 字节(80H~0FFH)只能通过间接寻址模式来使用
- 特殊功能寄存器(SFR)全部分布在高 128 字节范围内，通过直接寻址方式来使用

核内低位的 128 个字节可以采用直接或间接寻址进行访问。最低的 32 个字节分为 4 组寄存器可以作为通用寄存器组 R0-R7（由 PSW 的 RS0、RS1 选择使用那一组寄存器）；接下来的 16 个字节是位寻址区，共 128 位(00－7FH)。

3.3 特殊功能寄存器

SFR中定义的寄存器包括：累加器ACC、B寄存器、程序状态字寄存器PSW、堆栈指针寄存器SP、数据指针寄存器DPTR（DPH、DPL）、端口寄存器P0、定时器寄存器（TH0、TL0）/（TH1、TL1）、中断优先级寄存器IP、中断使能寄存器IE、定时器模式寄存器TMOD、定时器控制寄存器TCON和功耗控制寄存器PCON。这些寄存器及其相应的地址详见 附录 B 特殊功能寄存器。寄存器具体功能在相应的模块中有详细描述。

Z8D168 使用了双 DPTR 技术，可以提高访问外部存储器的效率。软件设计时请注意在中断处理时保护 DPS，DPH，DPL，DPH2，DPL2 寄存器。

第4章 存储保护单元(MPU)

4.1 概述

MPU（Memory Protect Unit 存储保护单元）用于保护存储器中的代码和数据不会被非法访问，并指示存储器访问越界错误，包括如下功能：

- 程序空间访问限制
  - 程序运行越界出错指示
  - 不同 FLASH 区域访问限制
- 外部存储器分配以及访问控制
  - 实现 OTP 区域
  - 按地址限制对外部存储器的访问
  - 按地址限制特殊寄存器访问
- 总线加扰

4.2 存储器空间

Z8D168 的存储资源主要由以下几部分构成：

- 168K 字节的 FLASH 空间
- 256 字节核内部 RAM
- 3K 字节核外部 RAM
- 1K 字节核外部 PAE RAM，不作 PAE 运算时可以作为通用核外部 RAM 使用

4.2.1 Rom存储器分配

Z8D168 的 Code Flash 存储器空间分配如所示：

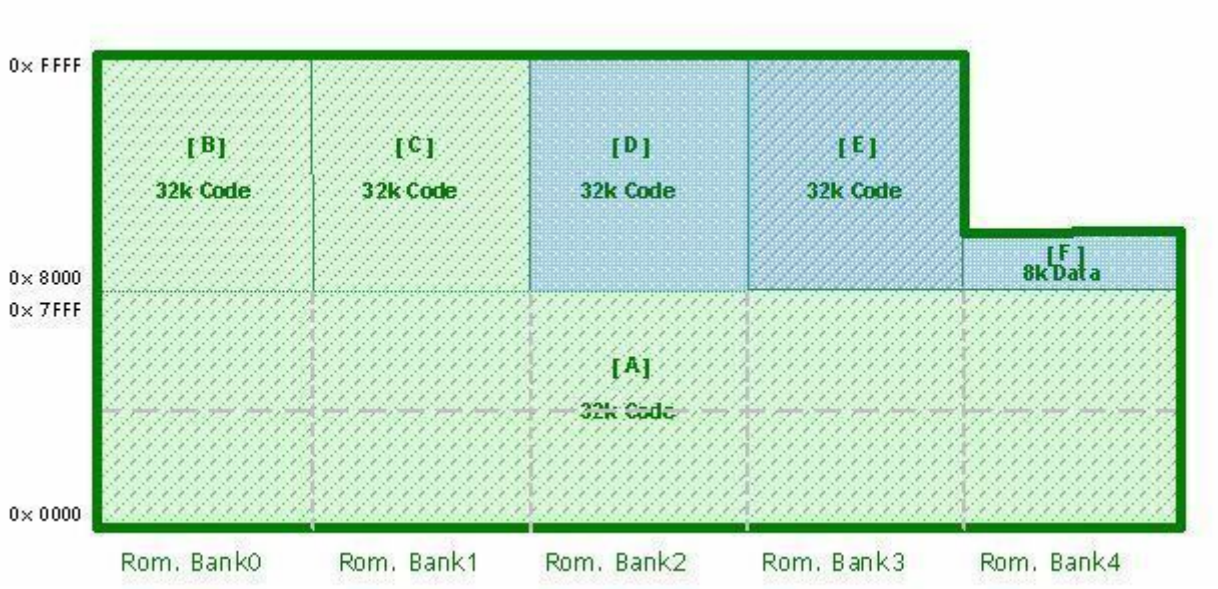


图 4-1 Rom Bank Assignment Map

表 4-1 Rom 地址分配表

ROM BUS			
ROMBANK		Start	End
*	A	0x0000	0x7FFF
ROM BANK 0	B	0x8000	0xFFFF
ROM BANK 1	C	0x8000	0xFFFF
ROM BANK 2	D	0x8000	0xFFFF
ROM BANK 3	E	0x8000	0xFFFF
ROM BANK 4	F	0x8000	0x9FFF

说明：

1、ROMBANK：指程序运行 PC 指针或 MOVC 指令运行时 DPTR 所指向的地址指针。

2、ROM Map映射如 图 4-1 Rom Bank Assignment Map，将 168K FLASH分为A、B、C、D、E、F六个区域，其中绿色区域是固定的程序存储器，蓝色区域既可以当程序存储器使用也可以当数据存储器使用。

3、由 表 4-1 Rom 地址分配表可见，ROM BUS低 32k寻址空间（ROM: 0x0000 ~ 0x7FFF）固定对应FLASH BANK [A]，高 32K寻址空间（ROM: 0x8000 ~ 0xFFFF）为Rom Bank Window，可通过配置RomBank来切换不同的Flash Memory Block（分别对应Code Flash的 4 个 32KB[B]-[E]和Data Flash的 8KB[F]）。

4.2.2 Ramx存储器分配

Z8D168 的Ramx存储器空间分配如 图 4-2 Ramx Bank Assignment Map所示：

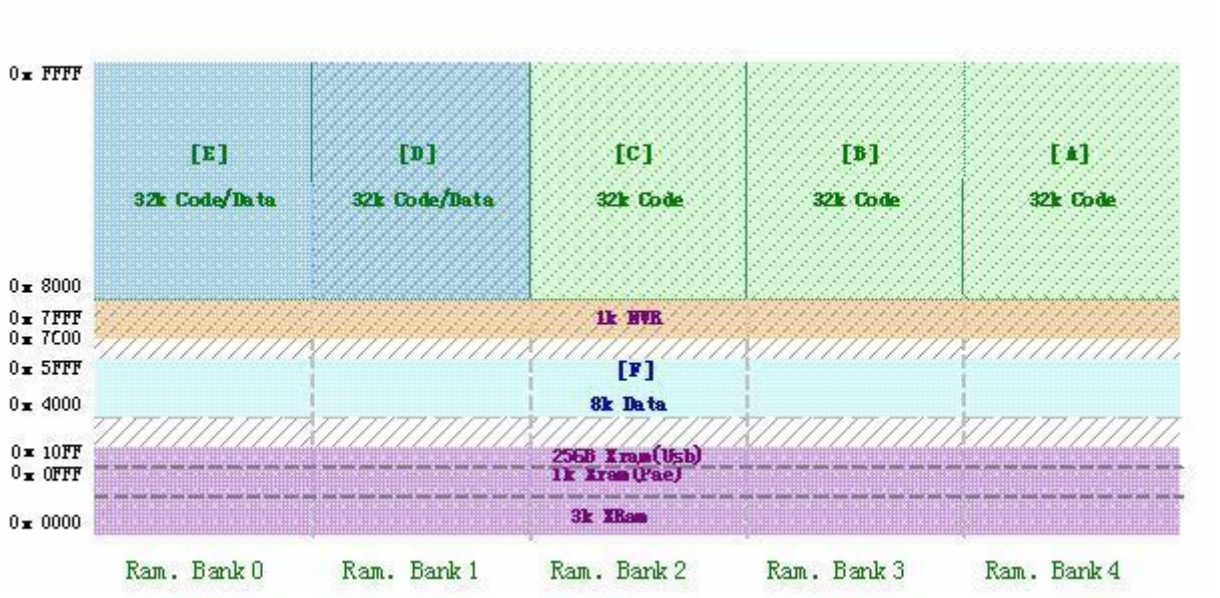


图 4-2 Ramx Bank Assignment Map

表 4-2 Ramx 地址分配表

Ramx BUS			
RAMX BANK		Start	End
扩展 Ram	*	0x0000	0x0BFF
PAE/Hash Ram	*	0x0C00	0x0FFF
DataFlash	F	0x4000	0x5FFF
OTP	*	0x7C00	0x7FFF
Ram Bank 0	E	0x8000	0xFFFF
Ram Bank 1	D	0x8000	0xFFFF
Ram Bank 2	C	0x8000	0xFFFF
Ram Bank 3	B	0x8000	0xFFFF
Ram Bank 4	A	0x8000	0xFFFF

说明：

1、 RAMBANK：指 MOVX 指令运行时 DPTR 所指向的地址总线。

2、 RAM Map映射如 图 4-2 Ramx Bank Assignment Map所示，由 表 4-2 Ramx 地址分配表可见在Ramx Space上，高 32k寻址空间（RAMX：0x8000 ~ 0xFFFF）为Ramx Bank Window，可通过配置RAMBANK来切换不同的Memory Block。

3、 OTP 区域仅 0x7D00-0x7DFF 区间用户可访问。

4.3 寄存器配置

4.3.1 MPU控制寄存器

表 4-3 MPU 控制寄存器(MPUCR－FFH)

MPUCR		MPU 控制寄存器				FFh	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rev	rev	rev	rev	rev	rev	rev	SAE
-	-	-	-	-	-	-	R/W
0	0	0	0	0	0	0	0
SAE	SectorAccessEn=1：表示不同 Sector 之间允许进行 Movc/Movx 操作 SectorAccessEn=0：表示不同 Sector 之间不允许进行 Movc/Movx 操作						

注意：MPUCR 仅 BOOT 区域可访问，由 BOOT 初始化时配置。

4.3.2 MPU状态寄存器

表 4-4 MPU 状态寄存器（MPUSR—FEH）

MPUSR		MPU 状态寄存器				FEh	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rev	GIDEF	SSXEF	BFEF	SFREF	INFEF	RAEF	ROEF
-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
GIDEF	GID 非法访问错误是指非 BOOT 区分成六个区域，不同区域根据 MPUGIDA、MPUGIDB 来配置 GID 号（每个区域可以配置成 0、1、2、3），GID 相同则可以相互 MOVC/MOVX 访问，不同时若 MPUCR.SAE 未置 1 的话，将不允许相互 MOVC/MOVX 访问。 GIDEF =1：表示 GID 非法访问错误 GIDEF =0：表示无 GID 非法访问错误							
SSXEF	该标志位仅在 SCB2/SSF33 模式时，非 BOOT 区程序读复用 PRAM 的数据时产生；非 BOOT 区应用程序写 PRAM 被屏蔽，但不会产生此错误标志 SSXEF =1：表示 SCB2/SSF33 算法专用 RAM 区非法读访问错误						

	SSXEF =0：表示无 SCB2/SSF33 算法专用 RAM 区非法读访问错误
BFEF	BFEF =1：表示 FLASH Boot 区非法访问错误 BFEF =0：表示无 FLASH Boot 区非法访问错误
SFREF	当非 BOOT 区程序读写 MPUCR、MPUGIDA、MPUGIDB 时产生。 SFREF =1：表示 SFR 非法访问错误 SFREF =0：表示无 SFR 非法访问错误
INFEF	INFEF =1：表示 INF 非法访问错误 INFEF =0：表示无 INF 非法访问错误
RAEF	RAEF =1：表示 RAMX 访问越界错误 RAEF =0：表示无 RAMX 访问越界错误
ROEF	ROEF =1：表示 ROM 访问越界错误 ROEF =0：表示无 ROM 访问越界错误
该寄存器所包含的所有状态位都是软件写 0 清零，硬件置 1。 （对软件写 1 不产生中断，但标志位改变）	

4.3.3 Rom Bank选择寄存器

表 4-5 Rom Bank 选择寄存器(ROMBANK－FDH)

ROMBANK		Rom Bank 选择寄存器				0FDh	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rev	rev	rev	rev	rev	ROMBANK[2:0]		
R	R	R	R	R	R/W		
0	0	0	0	0	0	0	0
ROM BANK [2:0]	ROMBANK =3’b000：表示选中[B]，ROM 总线选中 FLASH 的 BANKB。 ROMBANK =3’b001：表示选中[C]，ROM 总线选中 FLASH 的 BANKC。 ROMBANK =3’b010：表示选中[D]，ROM 总线选中 FLASH 的 BANKD。 ROMBANK =3’b011：表示选中[E]，ROM 总线选中 FLASH 的 BANKE。 ROMBANK =3’b100：表示选中[F]，ROM 总线选中 FLASH 的 BANKF。						

4.3.4 Ram Bank选择寄存器

表 4-6 Ram Bank 选择寄存器(RAMBANK－FCH)

RAMBANK		Ram Bank 选择寄存器				0FCh	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rev	rev	rev	rev	rev	RAMBANK[2:0]		
R	R	R	R	R	R/W		
0	0	0	0	0	0	0	0
RAM BANK [2:0]	RAMBANK =3'b000：表示选中[E]，XRAM 总线选中 FLASH 的 BANKE。 RAMBANK =3'b001：表示选中[D]，XRAM 总线选中 FLASH 的 BANKD。 RAMBANK =3'b010：表示选中[C]，XRAM 总线选中 FLASH 的 BANKC。 RAMBANK =3'b011：表示选中[B]，XRAM 总线选中 FLASH 的 BANKB。 RAMBANK =3'b100：表示选中[A]，XRAM 总线选中 FLASH 的 BANKA。						

4.3.5 SectorGid寄存器A

表 4-7 SectorGid 寄存器 A(MPUGIDA－F8h)

MPUGIDA		Ram Bank 选择寄存器				0F8h	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sector3Gid[1:0]		Sector2Gid[1:0]		Sector1Gid[1:0]		Sector0Gid[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Sector0Gid[1:0]	FLASH BANKE 的 Group ID。						
Sector1Gid[1:0]	FLASH BANKD 的 Group ID。						
Sector2Gid[1:0]	FLASH BANKC 的 Group ID。						
Sector3Gid[1:0]	FLASH BANKB 的 Group ID。						

MPUGIDA 仅 BOOT 区域可访问，由 BOOT 初始化时配置。

4.3.6 SectorGid寄存器B

表 4-8 SectorGid 寄存器 B(MPUGIDB－F9h)

MPUGIDB	Ram Bank 选择寄存器	0F9h

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rsv.		Rsv.		Sector5Gid[1:0]		Sector4Gid[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

Sector5Gid[1:0]	FLASH BANKF 的 Group ID
Sector4Gid[1:0]	FLASH BANKA（除去 BOOT 区）的 Group ID，地址区域为：BOOT+1~7fff

MPUGIDB 仅 BOOT 区域可访问，由 BOOT 初始化时配置。

4.4 访问控制规则

- FLASH BANKA 低 8k 为 BOOT 区。
- MPU 部分 SFR（如 MPUCR、MPUGIDA、MPUGIDB），只可 BOOT 区访问，由 BOOT 初始化时配置；
- BOOT 区外 FLASH 程序不能访问(Movc/Movx)BOOT 区；
- 168K Byte FLASH（5*32K+ 8K）分为 6 个区域，每个区域可以配置 2 位 Gid(group id)，最多可分为 4 个分组。相同的 Gid 可互相访问(Movc/Movx)，不相同的 Gid 之间是否可互相访问由配置决定。
- 8K Data Flash 的最后 64 字节仅允许 BOOT 程序可写，且受 OTP MASK 对应比特位保护。
- 当芯片工作在 SCB2/SSF33 算法模式下，共享的 PAE RAM 仅对 BOOT 程序开放权限，其他区域的程序均不能访问。

第5章 FLASH存储器控制器(CFC)

5.1 概述

Z8D168 片上集成了 168K 字节的 FLASH 存储器另外还有 512 字节用作 OTP 区域。CFC（Combo Flash Control）以 XRAM 总线接口完成 CPU 访问片上 FLASH 的操作。

- 实现 168K+1K FLASH 的访问
- 实现 512 字节 OTP 区域
- 通过 CPU XRAM 总线访问片上 FLASH
- 通过 CFC 模块，CPU 可以对 FLASH 进行单字节的读、写操作，页擦除操作（Date Flash（BANK F）页大小为 64 字节，Code Flash（BANK A\B\C\D\E）页大小为 512 字节），还可以执行 VREAD 操作验证擦除是否完全。
- 所有 FLASH 操作，硬件会自动等待操作完成后才向下执行，不需要软件额外插入等待时间。

5.2 寄存器配置

CFC 模块提供 2 个 8 位寄存器来控制 FLASH 操作。

5.2.1 FLASH写/擦除控制寄存器

CFCCSR 是用来控制对FLASH进行写或擦除操作的控制寄存器，其功能描述如 表 5-1 所示。

表5-1 FLASH 写/擦除控制寄存器(CFCCSR—9AH)

CFCCSR	**FLASH 写/擦除控制寄存器**					**9AH**	
FWE	FWMOD			FTOF	FVRS	FTOEN	OTPCW
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OTPCW	OTP 封口使能						
FTOEN	FLASH Program/Erase Time Out Check Enable						
FVRS	FLASH VRead Error						

FTOF	FLASH Program/Erase Time Out Flag
FW MOD	FLASH 写/擦除模式选择： FWMOD = 0，Byte Program，字节编程模式； FWMOD = 1，Sector VREAD FWMOD = 3，第一次 Sector Erase(30H); FWMOD = 4，第二次 Sector Erase(31H); FWMOD = 5，第三次 Sector Erase(32H)； FWMOD = 6，第四次 Sector Erase(33H)； FWMOD = 7，Suicide 自毁操作，该操作将清除芯片中所有数据，包括芯片出厂信息。
FWRE	FLASH 写操作使能

5.2.2 OTP测试寄存器

OTP测试寄存器反应OTP区域封口状态，表示OTP区域是否还可以写入数据。OTP测试寄存器的功能描述如 表 5-2所示。

表5-2　OTP 测试寄存器(CFCOTPR—9Bh)

CFCOTPR		**OTP 测试寄存器**				**9Bh**	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
1	1	1	1	1	1	1	1
OTPR	该寄存器为存储 OTP 信息的寄存器，该寄存器只读； CFCOTPR[0] = 1 表示 FLASH BANKF 最后一页还可以写入； CFCOTPR[0] = 0，表示 FLASH BANKF 最后一页无法再修改； CFCOTPR[7] = 1 表示 OTP 区 0x7DC0-0x7DFF 还可以写入； CFCOTPR[7] = 0，表示 OTP 区 0x7DC0-0x7DFF 无法再修改； OTP 区域仅 0x7D00-0x7DFF 区间用户可访问。						

第三部分 系统功能控制

第6章 中断控制器(INTC)

6.1 概述

中断控制器包含了 Z8D168 核内中断处理单元 IPU 以及扩展中断控制器 INTC，所有中断源可以通过 INTC 或直接向 IPU 发送中断请求。IPU 响应中断请求，停止当前指令执行过程，跳转到中断服务程序中去。INTC 把外部中断 0（INT0）、外部中断 1（INT1）、串口中断（SiInt）都进行了扩展，并将 USB 的中断状态标志复用在串口中断标志寄存器（SIV）中。

- INT0 包括：DES、PAE、三合一算法 EcifInt（AES/SCB2/SSF33 算法）中断。
- INT1 包括：SEC、MPU、WDT、RNG、EXINT（GPIO6/GPIO12）中断。
- SiInt 包括：Scd/Uart、Scc、Spi 接口中断。
- SIV 包括：USB、Scd/Uart、Scc、Spi 中断状态标志位。

其中 DES、PAE、EcifInt、SEC、MPU、WDT、RNG、EXINT（GPIO12）中断使能复用在扩展中断使能寄存器（XIE）中，中断标志位复用在中断标志寄存器（XIV）中，EXTINT（GPIO6）中断使能和中断标志在寄存器 XXIEV 中。

Scd/Uart、Scc、Spi 接口中断使能分散在各个接口模块寄存器中，USB、Scd/Uart、Scc、Spi 中断状态标志位复用在 SI 扩展中断标志寄存器（SIV）中。

6.2 Z8D168核中断处理单元IPU

Z8D168 核 IPU 类似标准 8051，有 5 个中断源，分别是外部中断 INT0、INT1、定时器 0 的中断 TF0、定时器 1 的中断 TF1 和串行中断 SI。通过设置中断使能寄存器 IE，这些中断可以单独被使能或者禁用。五个中断的中断优先级从高到低分别是：IE0、TF0、IE1、TF1 和 SI。

与IPU有关的寄存器是中断使能寄存器IE和中断优先级寄存器IP，这两个寄存器的详细定义如 表 6-1和 表 6-2所示：

6.2.1 中断使能寄存器

表 6-1 中断使能寄存器（IE－A8H）

IE		中断使能寄存器					A8H
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	Rev	Rev	ES	ET1	EX1	ET0	EX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
EA	全局中断使能，EA=0 则所有中断被禁止						
ES	串行端口中断使能，ES=0 则串行端口中断被禁止						
ET1	Timer1 溢出中断使能，ET1=0 则 Timer1 溢出中断被禁止						
EX1	外部中断 1 使能，EX1=0 则外部中断 1 被禁止						
ET0	Timer0 溢出中断使能，ET0=0 则 Timer0 溢出中断被禁止						
EX0	外部中断 0 使能，EX0=0 则外部中断 0 被禁止						

6.2.2 中断优先级寄存器

表 6-2 中断优先级寄存器（IP－B8H）

IP		中断优先级寄存器					B8H
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	PS	PT1	PX1	PT0	PX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PS	定义串行端口中断优先级，PS=1 则串行端口具有更高优先级						
PT1	定义 Timer1 溢出中断优先级，PT1=1 则 Timer1 溢出中断具有更高优先级						
PX1	定义外部中断 1 优先级，PX1=1 则外部中断 1 具有更高优先级						
PT0	定义 Timer0 溢出中断使能，PT0=1 则 Timer0 溢出中断具有更高优先级						
PX0	定义外部中断 0 使能，PX0=1 则外部中断 0 具有更高优先级						

6.3 寄存器配置

扩展中断控制器 INTC 模块提供 1 个扩展中断使能寄存器（XIE）、1 扩展中断标志寄存器（XIV）、1 个 SI 扩展中断标志寄存器和 1 个 EX1 扩展中断标志寄存器（XXIEV）。CPU 操作这 4 个寄存器，完成中断复用功能。其中 XIE 寄存器控制各模块中断的使能，中断产生后 XIV 寄存器会将对应模块的中断标志置 1，需要软件清 0；XXIEV 寄存器可以使能 GPIO6（SCD_RST 引脚）对应扩展中断和标志扩展中断产生。

6.3.1 扩展中断使能寄存器

表 6-3 扩展中断使能寄存器(XIE—D1H)

XIE		IEx 扩展中断使能寄存器					D1h
X0IE			X1IE				
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DesInt En	PaeInt En	EcifInt En	SecInt En	MpuInt En	WdtInt En	RngInt En	ExtInt En
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

DesIntEn	Des 模块中断使能
PaeIntEn	Pae 模块中断使能
EcifIntEn	Ecif 模块中断使能（包括 AES/SCB2/SSF33 算法中断）
SecIntEn	Sec 模块中断使能
MpuIntEn	Mmu 模块中断使能
WdtIntEn	Wdt 模块中断使能
RngIntEn	Rng 模块中断使能
ExtIntEn	ExtInt 中断使能（SCD 模式下 GPIO12 设置为输入时，引脚低电平产生中断）

6.3.2 扩展中断标志寄存器

表 6-4 扩展中断标志寄存器(XIV—D2H)

XIV		IEx 扩展中断标志寄存器					D2h
X0IV			X1IV				
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DesInt Vec	PaeInt Vec	EcifInt Vec	SecInt Vec	MpuInt Vec	WdtInt Vec	RngInt Vec	ExtInt Vec
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

DesIntVec	Des 模块中断标志，硬件置 1，软件清 0（写“0”清）
PaeIntVec	Pae 模块中断标志，硬件置 1，软件清 0（写“0”清）
EcifIntVec	Ecif 模块中断标志，硬件置 1，软件清 0 （包括 AES/SCB2/SSF33 算法中断） 需要再查询下一级中断标志寄存器请参考算法各章节描述
SecIntVec	Sec 模块中断标志，硬件置 1，软件清 0（写“0”清）

MpuIntVec	Mmu 模块中断标志，硬件置 1，软件清 0（写“0”清）
WdtIntVec	Wdt 模块中断标志，硬件置 1，软件清 0（写“0”清） 清零该寄存器位需要先置位 WDTCSR.CLRINT 为 1
RngIntVec	Rng 模块中断标志，硬件置 1，软件清 0（写“0”清）
ExtIntVec	ExtInt 中断标志，硬件置 1，软件清 0（写“0”清） （SCD 模式下 GPIO12 设置为输入时，引脚低电平产生中断）

6.3.3 SI扩展中断标志寄存器

表 6-5 SI 扩展中断标志寄存器(SIV—D3H)

SIV		SI 扩展中断标志寄存器					D3h
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SpiInt Vec	SccInt Vec	ScdInt Vec	Err2Int Vec	Err1Int Vec	TkInt Vec	EpInt Vec	DevInt Vec
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

SpiIntVec	SPI 模块中断标志，硬件置 1，软件清 0
SccIntVec	SCC 模块中断标志，硬件置 1，软件清 0
ScdIntVec	Scd 模块中断标志，硬件置 1，软件清 0
Err2IntVec	USB 模块错误类中断 2 标志，硬件置 1，软件清 0（写“0”清）
Err1IntVec	USB 模块错误类中断 1 标志，硬件置 1，软件清 0（写“0”清）
TkIntVec	USB 模块令牌类中断标志，硬件置 1，软件清 0（写“0”清）
EpIntVec	USB 模块端点类中断标志，硬件置 1，软件清 0（写“0”清）
DevIntVec	USB 模块设备类中断标志，硬件置 1，软件清 0（写“0”清）

6.3.4 EX1扩展中断标志寄存器

表 6-6 EX1 扩展中断标志寄存器(XXIEV—ABH)

XXIEV		EX1 扩展中断标志寄存器					ABh
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCDRstintEn	Rev	Rev	Rev	Rev	Rev	Rev	SCDRstintVec
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

SCDRstintEn	SCD Rst（GPIO6）引脚扩展中断使能
SCDRstintVec	SCD Rst（GPIO6）引脚扩展中断标志，硬件置 1，软件清 0

第7章 时钟控制（CGU）

7.1 概述

CGU（Clock Generate Unit）负责提供各种频率的时钟，对 CPU 时钟和各个功能模块的时钟进行控制。Z8D168 提供三种方式进行功耗管理：CPU 和 PAE 频率控制，低功耗模式以及模块时钟开关控制。

一、 CPU 和 PAE 频率控制

Z8D168 提供对 CPU 和 PAE 模块工作频率的配置。

Z8D168 内部时钟基准源(OSC)提供 40MHz 时钟，经分频后提供给 CPU 和 PAE 模块。通过配置 CGUFDR，CGUFCR 寄存器，可以改变 CPU 和 PAE 模块的工作频率。CPU 最大工作频率 20MHz，最小工作频率 5MHz；PAE 最大工作频率 40MHz，最小工作频率 5MHZ，要求 PAE 模块的工作频率大于或等于 CPU 的工作频率。

二、Z8D168 提供四种低功耗模式：

1. IDLE 模式
2. 深度主动休眠模式
3. 深度被动休眠模式
4. 主动唤醒休眠模式

通过配置 PCON 寄存器或检测外部停时钟（7816 模式下）以及配置 CGUFDR.ARPD 位，芯片会进入相应的休眠模式。

三、Z8D168 提供对多个功能模块的时钟进行开关控制。

通过配置 CGUFCR 寄存器，用户可以控制各功能模块时钟的开启或关闭，以达到降低功耗的目的。

7.2 寄存器配置

7.2.1 CGU分频寄存器

表7-1 CGU 分频寄存器(CGUFDR—E9H)

CGUFDR		CGU 分频寄存器					E9H
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PLLRDY/ARPD	PLLSoftEn	ClkGscRngEn	CSDE	Pae FreqDivNum		CoreFreqDivNum	
R/W	R/W	R/W	R/W	R/W		R/W	
1	0	0	0	0	0	1	0

PLLRDY/ARPD	读：PLL 是否处于正常运行状态，1 为正常，表示 PLL 有输出，0 表示 PLL 还没有输出； 写：主动唤醒省电模式使能信号，软件置 1 后，系统进入主动唤醒省电模式。
PLLSoftEn	1：在 SCD 模式，且 CPU 频率和 PAE 模块频率同时选择 5M 时，打开 PLL 0：在 SCD 模式，且 CPU 频率和 PAE 模块频率同时选择 5M 时，系统自动关闭 PLL，降低功耗。
ClkGscRngEn	ClkGsc 时钟加扰使能
CSDE	外部停时钟检测使能(在 7816 模式下，监测外部时钟停止)
Pae FreqDivNum	PAE 模块分频系数： 该值为 00 时表示 1 分频系数 该值为 01 时表示 2 分频系数 该值为 10 时表示 4 分频系数 该值为 11 时表示 8 分频系数 PAE 的工作频率要大于等于核频率。
CoreFreqDivNum	核分频系数： 该值为 00 或 01 时表示 2 分频系数 该值为 10 时表示 4 分频系数 该值为 11 时表示 8 分频系数 复位后默认为 4 分频系数 (注：核的实际频率为 40M/分频系数)

说明：

必须在 PAE 模块时钟使能的情况下（CGUFCR. PaeClkEn 为 1），才能改变 PAE 模块分频系数。关闭 PAE 模块前必须配置 CGUFDR. Pae FreqDivNum 为 0x00 或 0x03。

7816 模式下，核时钟配置 5M 和 PAE 模块时钟配置 5M 会关闭 PLL 降低功耗，如果修改 PAE 时钟，会自动打开 PLL，PLL 需要经过一段时间才能运行稳定，PLL 运行稳定后 CGUFDR.PLLRDY 会置位，也可以在切换频率前主动置位 CGUFDR.PLLSoftEn，提前开启 PLL。

7.2.2 CGU功能模块控制寄存器

表 7-2 CGU 功能模块控制寄存器(CGUFCR—E8H)

CGUFCR		CGU 功能模块控制寄存器					E8H
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UsbUdc En	UsbPhy En	SaeClk En	SPIClk En	RngClk En	DesClk En	ScdClk En	PaeClk En
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	0	0	0	0	1	0
UsbUdcEn		UsbUdc 时钟使能。0 ： 关闭 ； 1：开启					
UsbPhyEn		UsbPhy 时钟使能。0 ： 关闭 ； 1：开启					
SaeClkEn		Scb2、SSF33、AES 算法模块时钟使能。0 ： 关闭 ； 1：开启					
SPIClkEn		SPI、SCC 模块时钟使能。0 ： 关闭 ； 1：开启					
RngClkEn		RNG 模块时钟使能。0 ： 关闭 ； 1：开启					
DesClkEn		Des、KGE 模块时钟使能。0 ： 关闭 ； 1：开启					
ScdClkEn		Scd 模块时钟使能。0 ： 关闭 ； 1：开启					
PaeClkEn		Pae 模块时钟使能。0 ： 关闭 ； 1：开启					

7.2.3 电源控制寄存器

表 7-3 电源控制寄存器(PCON—87H)

PCON		电源控制寄存器					87H
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	Rev	Rev	PM[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	-	-	-	-	-	0	0
PM[1:0]	Power Mode 。 00：工作模式 01：IDLE 模式 10：PD (PowerDown) 模式						

7.3 低功耗模式的进入条件与退出条件

	特色	进入条件	退出条件	描述	应用
Idle Mode	中等主动休眠	软件配置 PCON.Idle	内部中断	core 相关部分休眠 Cfc,Wdt,Int 等模块不休眠 OSC 状态保持 FLASH 可进入 StandBy	可在 CPU 空转等待中断等场合应用
PowerDown Mode	深度主动休眠	软件配置 PCON.Pd	外部信号（7816 起始位）	全部数字逻辑休眠 FLASH 进入 StandBy	空闲时的低功耗模式，可用于完成一次任务，等待下一个命令的场合
ClkStp Mode	深度被动休眠	检测到 SCD 时钟停止	SCD 时钟沿	全部数字逻辑休眠 FLASH 进入 StandBy	来自 7816 主控制器的停时钟请求
Auto Resume Power Down	主动唤醒休眠模式	配置寄存器 CGUFDR.7	省电计数器计数满 5.12ms 倍数时间后自动退出，时间可软件配置； 外部信号（7816 起始位）	全部数字逻辑休眠 FLASH 进入 StandBy	

1、IDLE 模式： PCON=0x01，一个指令周期后芯片进入休眠状态。能够通过中断退出该模式（WDT 中断除外）。

2、主动 Power Down 模式：PCON=0x02，一个指令周期后芯片主动进入深度休眠，此时检测到通信线（7816 引脚）上的起始位将退出该模式，也可通过 USB 总线 RESUME 唤醒。

3、被动 Power Down 模式：7816 模式下，配置 CGUFDR.CSDE=1，且检测到外部 7816 时钟停止时，芯片进入深度休眠。检测到外部时钟线上的上升沿将退出该模式。

4、主动唤醒休眠模式：配置 CGUFDR.7 为 1，芯片进入休眠模式，计数满 5.12ms 整数倍（可配），芯片唤醒；检测到通信线（7816 引脚）上的起始位也将退出该模式

7.4 使用说明

CGU 相关寄存器的操作可能会使整个芯片的时钟发生变化，所以需慎重。以下列出操作时应注意的一些问题。

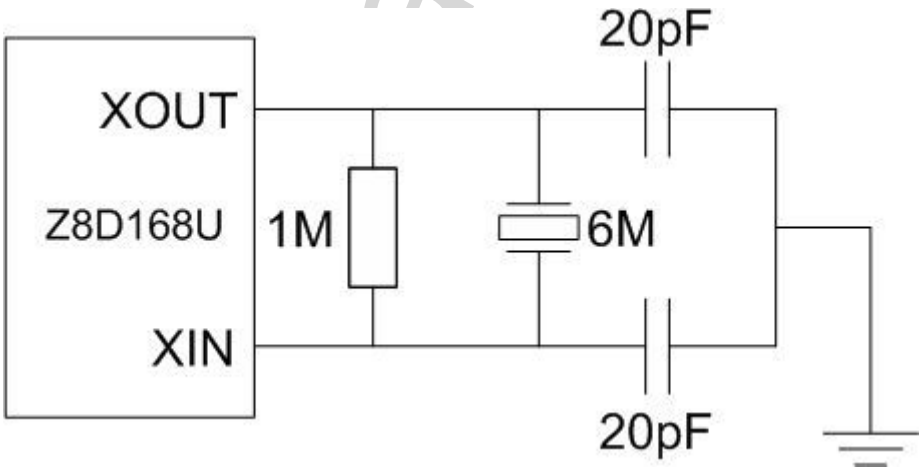
1、在做 FLASH 写操作时（编程、擦除、OTP 封口以及所有页操作）时，建议关闭外部停时钟检测使能 CGUCR.CSDE；

2、当 IO 模式配置为 UART 模式时，如果 CPU 时钟改变，则波特率计数器（UARTBPH、UARTBPL）的计数时间也随之发生改变，因为 UART 模式下，波特率分频的时钟源来自 CPU 时钟。IO 模式配置为 7816 模式时，CPU 时钟的改变不会影响 7816 的波特率，因为其分频的时钟源来自外部时钟。

3、当 CPU 时钟改变时，定时器（TIMER0、TIMER1）的计数时间也随之发生改变，因为定时器的时钟来自于 CPU 时钟 16 分频后的时钟。

7.5 外部时钟

USB 全速模式下芯片需要外部提供 6M 的时钟，这个 6M 外部时钟可以由 XIN 引脚直接引入，也可以由 XOUT、XIN 引脚外接 6M 晶体和内部振荡电路产生。此时内部 40M 基准时钟源准确频率为 38.4MHz。外接晶体电路如下图所示：



第8章 复位控制单元(RCU)

8.1 概述

RCU(RESET CONTROL UNIT)模块是复位控制模块，软件可以通过设置 RCU 控制寄存器来控制系统复位。Z8D168 提供 4 种复位方式：上电复位，WDT 复位，接口复位和软复位。

- 上电复位

当芯片上电后，引发上电复位过程。

- WDT 复位

当 WDT 溢出时，如果使能了 WDT 复位功能，则将引起系统复位。

- 接口复位

当 IO 接口配置为 SCD 模式时，SCD 复位引脚上的有效复位信号会引起系统复位。

当 IO 接口配置为 SCD 以外的模式时，如果把其中 RST 引脚配置为扩展复位功能时，此引脚上的有效复位信号将引起系统复位。

- 软复位

当给 RCUCR. RcuRst 位置 1 时，将引起系统复位。

8.2 寄存器配置

表 8-1 RCU 控制寄存器(RCUCR—EFH)

RCUCR		RCU 控制寄存器					EFh
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	FdHf RstEn	Lvd25 RstEn	Lvd23 RstEn	RsmPd RstEn	UsbS Rst	RcuRst
R	R	R/W	R/W	W/R	W/R	W	W
0	0	0	1	1	0	0	0

FdHfRstEn	频率检测高频复位使能信号，高电平有效
Lvd25RstEn	2.5V 低电压检测复位使能信号，高电平有效，缺省上电使能
Lvd23RstEn	2.3V 低电压检测复位使能信号，高电平有效，缺省上电使能
RsmPdRstEn	软件控制 USB 模式下 Resume 是唤醒 Power Down 模式，

	高电平有效
UsbSRst	USB 模块局部复位，高电平有效
RcuRst	软件复位，高电平有效

第9章 看门狗定时器（WDT）

9.1 概述

为避免软件运行时进入死循环，同时监视程序段运行时间是否正常，防止系统进入死锁状态，需要由 WDT(WatchDog Timer)对软件运行的时钟周期数进行监控，提供跳出软件死循环或系统锁死的功能。WDT 一旦启动，软件必需周期性的对它操作，否则 WDT 产生的溢出复位信号将会使整个系统复位。

9.2 寄存器配置

9.2.1 看门狗控制/状态寄存器

表 9-1 看门狗控制/状态寄存器（WDTCSR－91H）

WDTCSR		WDT 看门狗控制/状态寄存器					91H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WTRF	CLRINT	Rev	Rev	WD1	WD0	EWT	RWT
R	W	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WTRF	WDT 复位标志位，WDT 引起芯片复位后，该位由硬件置 1，软件清 0。						
CLRINT	CLRINT：清除 X1IV 寄存器 WDT 中断向量标志位前，需要先给该位写 1。						
WD1	计数溢出置的高位。（可参照 表 9-3计数溢出值表）						
WD0	计数溢出置的低位。（可参照 表 9-3计数溢出值表）						
EWT	复位输出使能信号。该位为 1 时，WDT 溢出后会产生系统复位；该位为 0 时，WDT 溢出后不会产生系统复位						
RWT	复位 WDT 模块中的计数器。WDT 模块中的计数器倒计数到 0 之前，软件要重新置该位为 1，使计数器根据设置的计数溢出值重新开始倒计数。若在计数器计到 0 之前，没有置该位为 1 时，WDT 模块将输出中断或复位信号。硬件会自动清除该位，同时启动 WDT 时应先对该位置 1。						

9.2.2 看门狗写控制寄存器

表 9-2 看门狗写控制寄存器（WDTTAP－92H）

WDTTAP		WDT 看门狗写控制寄存器					92H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
Bit 7~ 0		若 MCU 要向 WDTCSR(控制/状态寄存器)低两位进行写操作前，要先对 WDTTAP 先写入 0xaa 数据， 紧接的 1 个指令周期里向 WDTTAP 写入 0x55 数据后，再紧接的 1 个指令周期里对 WDTCSR(控制/状态寄存器)的写操作才有效，否则对 WDTCSR(控制/状态寄存器)低两位的写操作都是无效的。WDTTAP 是只写寄存器，读出来的值无效。					

9.3 使用说明

WDT 模块在启动内部计数器使能后，该计数器根据所选的计数值开始倒计数，计数的间隔为系统时钟周期。

在程序正常运行的情况下，在计数器还未计到 0 之前，软件要复位 WDT 计数器，并开始倒计数。若计数器计到 0，表示这时程序已出错，看门狗时钟模块（WDT）就会产生复位信号和中断信号（若复位使能和中断使能信号有效）。

若 MCU 对 WDT 控制寄存器中的复位输出使能信号 WdtEwt 和重新置初始值给计数器的信号 WdtRwt 进行写操作时，要先对写保护寄存器写入 0xaa，在之后的 1 个指令周期里向该寄存器写入 0x55，再紧接的 1 个指令周期（即 1 个系统时钟周期）里对 WDT 控制寄存器进行写操作，才能改变这两个信号的内容。若超过这 3 个指令周期后才对控制寄存器进行写操作，这时控制寄存器的内容将不会被改变，除非重新对写保护寄存器进行操作。

WDT 模块中产生中断和复位信号的 4 种时间溢出。

表 9-3 计数溢出值表

WDTCSR[3]	WDTCSR[2]	中断产生时间	复位产生时间
0	0	2^{17}clocks	2^{17} + 512 clocks
0	1	2^{20}clocks	2^{20} + 512 clocks
1	0	2^{23}clocks	2^{23} + 512 clocks
1	1	2^{26}clocks	2^{26} + 512 clocks

第10章 定时器单元(TMU)

9.1 概述

Z8D168 有两个 16 位的定时器：定时器 0 和定时器 1。定时器的时钟源来自于 CPU 核时钟 16 分频后的时钟，每 16 个核时钟定时器加 1，当定时器值到 0x10000 时产生溢出信号。

9.2 寄存器配置

10.1.1 TH0、TL0、TH1、TL1寄存器

表 10-1 TH0、TL0、TH1、TL1 寄存器

名称	功能	状态	初始值	地址
TH0	定时器 0（高位字节）	R/W	00H	8CH
TL0	定时器 0（低位字节）	R/W	00H	8AH
TH1	定时器 1（高位字节）	R/W	00H	8DH
TL1	定时器 1（低位字节）	R/W	00H	8BH

10.1.2 定时器模式寄存器

表 10-2 定时器模式寄存器（TMOD－89H）

TMOD	**定时器模式寄存器**						**89H**
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	M1	M0	Rev	Rev	M1	M0
Rev	Rev	R/W	R/W	Rev	Rev	R/W	R/W
0	0	0	0	0	0	0	0
[M1:M0] [Bit5:Bit4]	Timer1 工作模式选择 模式 0：做一个 13 位的定时器用，由 TH0/1 的低 5 位和 TL0/1 的 8 位构成。 模式 1：做一个 16 位的定时器用。 模式 2：做一个 8 位的定时器用，计数满时 TL0/1 自动从 TH0/1 加载数据。						

[M1:M0] [Bit1:Bit0]	Timer0 工作模式选择 模式 0：做一个 13 位的定时器用，由 TH0/1 的低 5 位和 TL0/1 的 8 位构成。 模式 1：做一个 16 位的定时器用。 模式 2：做一个 8 位的定时器用，计数满时 TL0/1 自动从 TH0/1 加载数据。

10.1.3 定时器控制寄存器

表 10-3 定时器控制寄存器（TCON－88H）

TCON	定时器控制寄存器						88H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TF1	Timer1 溢出标志，当定时器 1/计数器 1 溢出时，由硬件置 1，申请中断。进入中断服务程序被硬件自动清零						
TR1	Timer1 运行控制位，靠软件置位或清除，置 1 时，定时器 1/计数器 1 接通工作，清零时停止工作						
TF0	Timer0 溢出标志，当定时器 0/计数器 0 溢出时，由硬件置 1，申请中断。进入中断服务程序被硬件自动清零						
TR0	Timer0 运行控制位，靠软件置位或清除，置 1 时，定时器 1/计数器 1 接通工作，清零时停止工作						
IE1	外部沿触发中断 1 请求标志。检测到 INT1 引脚上出现的外部中断信号的下降沿时，由硬件置 1，请求中断。进入中断服务程序后被硬件自动清零						
IT1	外部中断 1 类型控制位。靠软件置 1 或清零，以控制外部中断的触发类型。IT1=1 时，下降沿触发，IT1=0 时，低电平触发						
IE0	外部沿触发中断 0 请求标志						
IT0	外部中断 0 类型控制位						

第四部分 安全控制

第11章 DES算法引擎

11.1 概述

DES 模块完成 DES、3DES 运算。

11.1.1 功能

- 实现 DES，3DES（2 KEY 和 3 KEY）加密和解密运算。
- 支持 EBC 模式和 CBC 模式的加密和解密。

11.2 支持模式

DES 模块实现 DES、3DES 算法的加密和解密运算。支持以下几种模式：

1. 单 DES ECB

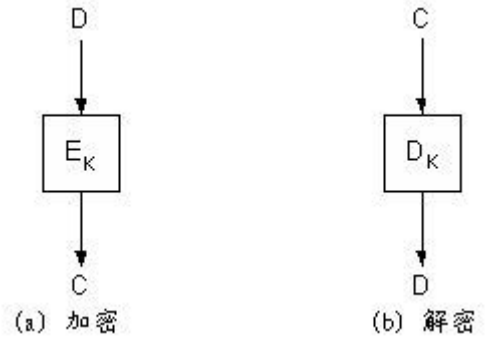


图 11-1 单 DES ECB 的运算流程图

2. 3DES 双密钥 ECB

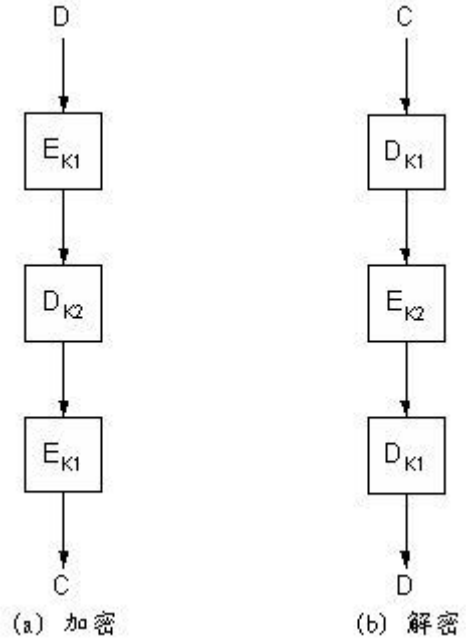


图 11-2　3DES 双密钥 ECB 的运算流程图

3. 3DES 三密钥 ECB

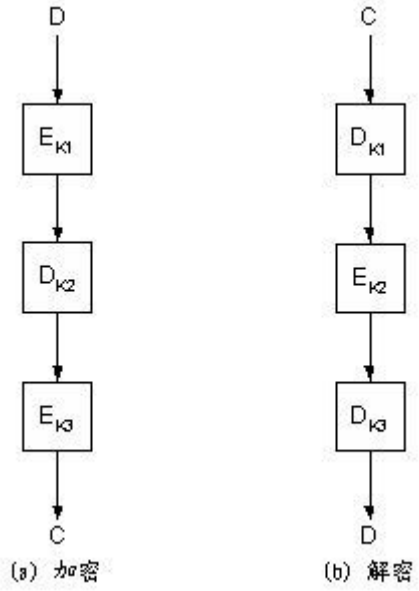


图 11-3　3DES 三密钥 ECB 的运算流程图

4. 单 DES CBC

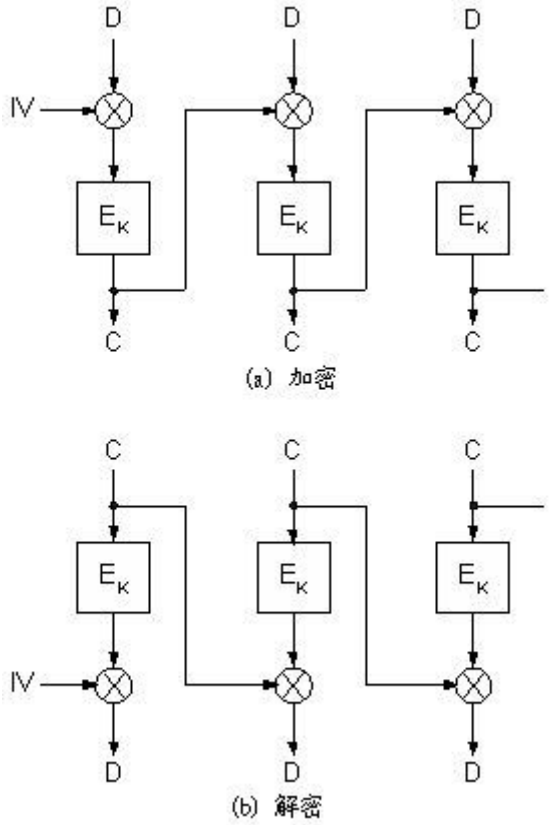


图 11-4　单 DES CBC 的运算流程图

5. 3DES 双密钥 CBC

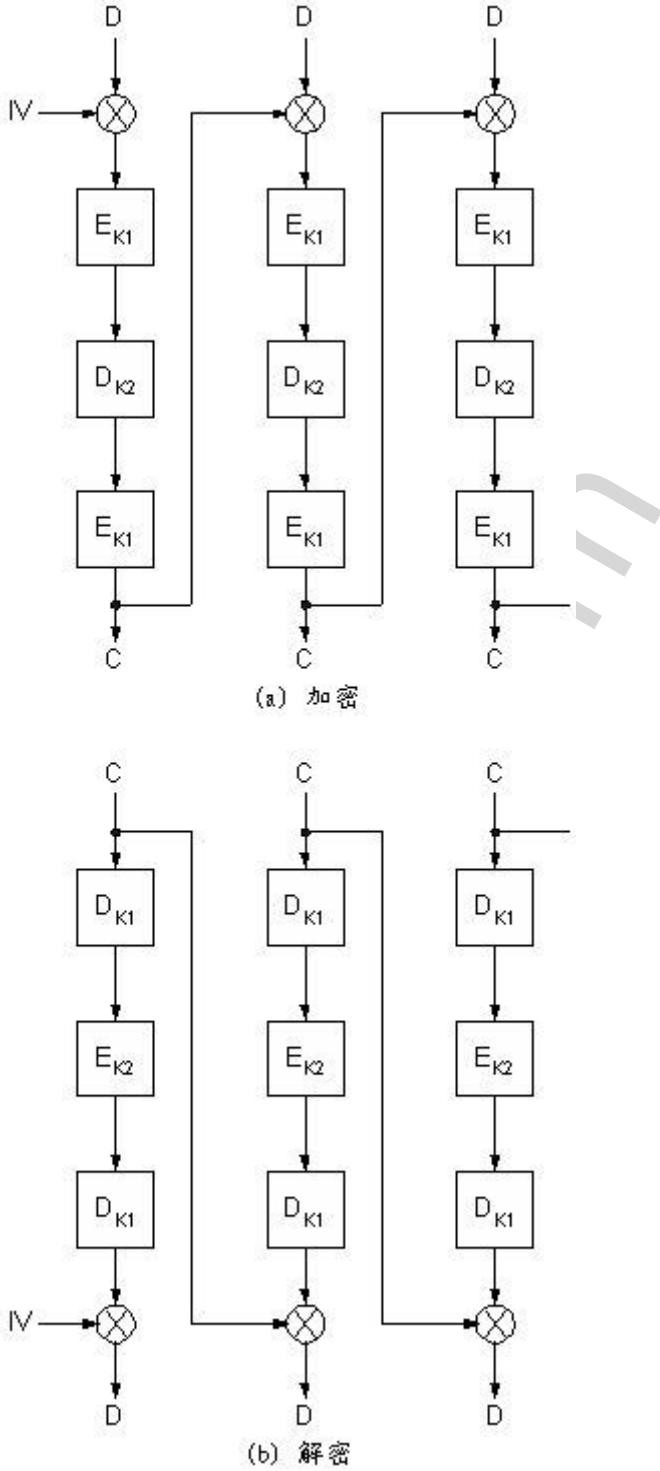


图 11-5　3DES 双密钥 CBC 的运算流程图

6. 3DES 三密钥 CBC

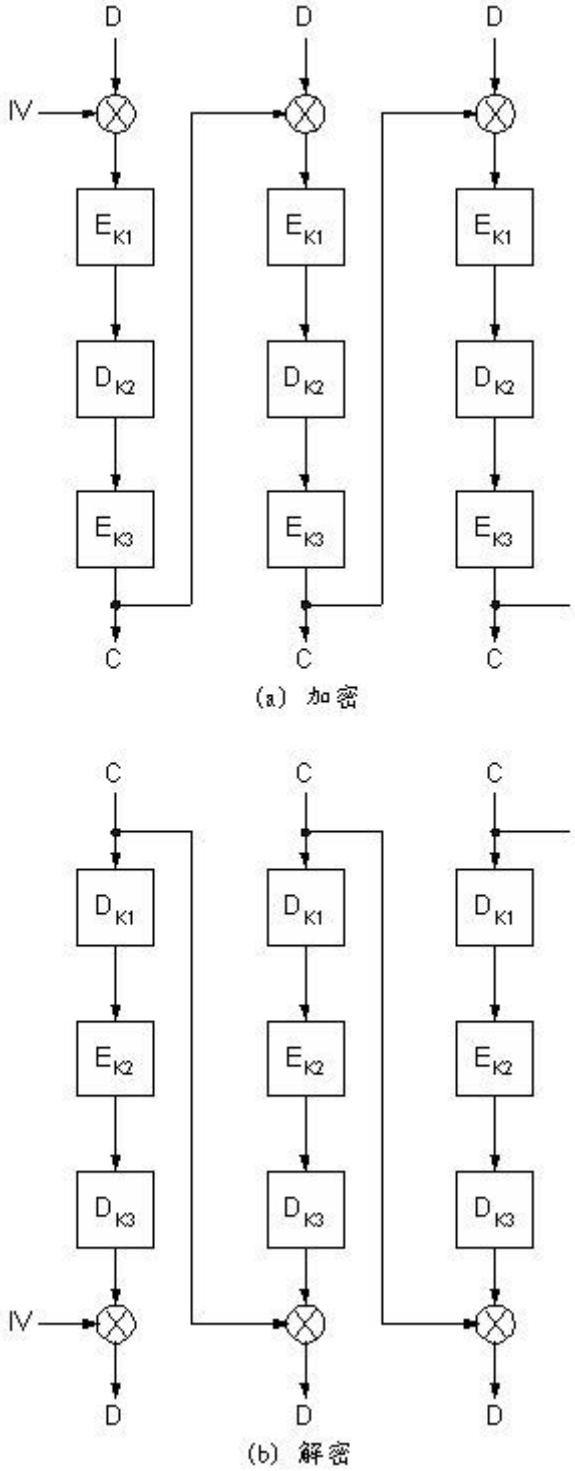


图 11-6　3DES 三密钥 CBC 的运算流程图

11.3 寄存器配置

11.3.1 DES数据寄存器

表 11-1 DES 数据寄存器（DESDR－E5H）

DESDR		DES 数据寄存器					E5H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DESDR[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R
0	0	0	0	0	0	0	0
DESDR[7:0]		加解密数据的装入和结果数据的读出					

DES 数据寄存器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~ bit8，再写入 bit9~ bit16，依次类推，最后写入 bit57~ bit64。结果数据读出时，也要顺序读出，先读出 bit1~ bit8，再读出 bit9~ bit16，依次类推，最后读出 bit57~ bit64。最多可以连续写 64bit。

11.3.2 密钥寄存器

表 11-2 密钥寄存器（DESKR－E7H）

DESKR		密钥寄存器					E7H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DESKR[7:0]							
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
DESKR[7:0]		加解密密钥的装入					

密钥寄存器 DESKR 用于加密和解密密钥的装入，长度为 8bit。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~ bit8，再写入 bit9~ bit16，依次类推，最后写入 bit57~ bit64。

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：

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。

11.3.3 控制寄存器

控制寄存器 DESCR 用于启动加解密运算以及一些特殊条件的控制。

表 11-3 DES 控制寄存器（DESCR－E4H）

DESCR		DES 控制寄存器					E4H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RUN	ENCRY	DES	KEY	EDR[1:0]		ECB	Rev
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
0	0	0	0	0	0	0	0
RUN		启动 DES 运算，软件置位，硬件清除 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 0，进行加密运算 1，进行解密运算					
DES		DES/TDES 指示位 0，进行 DES 运算 1，进行 TDES 运算					
KEY		TDES 运算中，2KEY/3 KEY 的指示位。DES 运算时，此位无效 0，2 密钥运算 1，3 密钥运算					
EDR[1:0]		指定执行的 DES 回合数 00，执行 16 个 DES 回合 01，执行 1 个 DES 回合 10，执行 2 个 DES 回合 11，执行 3 个 DES 回合					
ECB		ECB/CBC 模式指示位 0，ECB 模式 1，CBC 模式					

11.3.4 DES初始向量寄存器

表 11-4 DES 初始向量寄存器（DESIV－E6H）

DESIV		DES 初始向量寄存器					E6H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DESIV[7:0]							
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
DESIV [7:0]		初始向量数据：只允许用户进行写操作。 用于存储 CBC 模式加、解密所需的初始向量数据。					

11.4 DES模块运算流程

图 11-7描述DES模块的运算流程，用户可参照此流程对数据进行加解密操作。

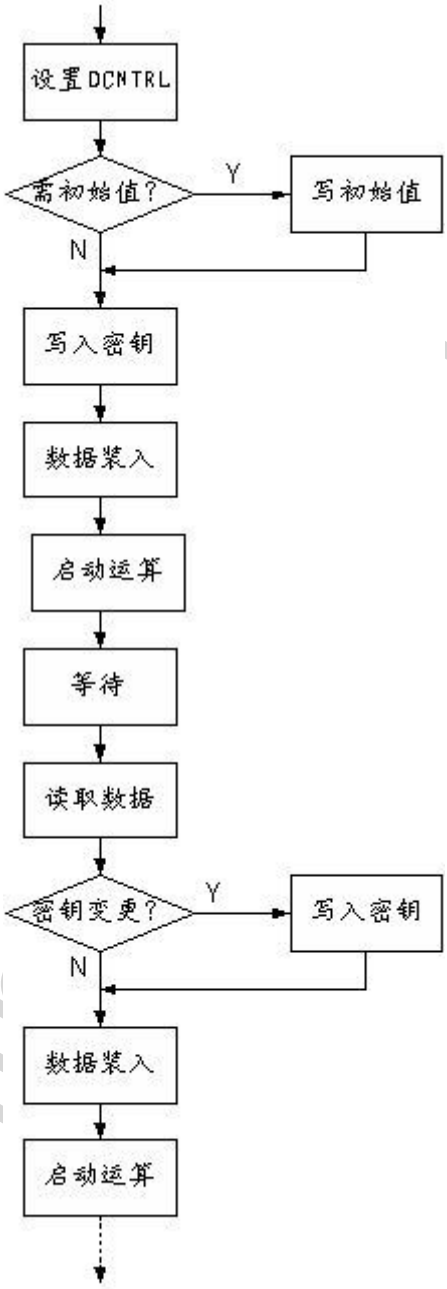


图 11-7 DES 运算流程

第12章 公钥算法引擎（PAE）

12.1 概述

PAE（Public Arithmetic Engine）公钥算法引擎硬件实现了公钥加密算法必需的模幂、模乘、乘法及多项式乘法等运算。

12.2 寄存器配置

12.2.1 PSAE控制状态寄存器

表 12-1 PSAE 控制状态寄存器(PSAECSR－94hH)

PSAECSR		PAE 控制状态寄存器					94h
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SAE RDY	PRAM RDY	SSFINT	SCBINT	AESINT	SHFT MOD	AEMOD[1:0]	
R	R	R/W	R/W	R/W	R/W	R/W	R/W
1	0	0	0	0	0	0	0

SAERDY	SAE 模块（AES/SCB2/SSF33）运算结束标志，该标志位会比各模块中的 RUN/BUSY 标志位晚了几个系统时钟置位。 1：SAE 算法空闲或者运行结束，若 PRAMRDY 也等于 1 则可以访问 PAE Ram 存储区； 0：SAE 算法模块忙，不能访问 PAE Ram 存储区
PRAMRDY	PAE Ram 自动清零状态指示位， 0：PAE Ram 自动清零忙中，不能访问 PAE Ram； 1：PAE Ram 不处于自动清零
SSFINT	SSF 模块中断标志位，硬件置 1，写 1 清零。
SCBINT	SCB 模块中断标志位，硬件置 1，写 1 清零。
AESINT	AES 模块中断标志位，硬件置 1，写 1 清零。
SHFTMOD	移位运算模式，该模式不能与 SSF33 模式同时并存。
AEMOD[1:0]	00：PAE 模式； 01：AES 模式 10：SCB2 模式 11：SSF33 模式

注意：

- SCB2/SSF33 模式切换到其它模式时（AEMOD[1]从 1 变为 0）会自动清零 PAE Ram 内容。建议程序不要随意改变 AEMOD 的设置，否则易导致 PAE Ram 中的有用数据丢失。
- 建议在设置 AEMOD 之后，先查询 PRAMRDY 标志位再对 PAE Ram 操作。

12.2.2 PAE控制命令寄存器

表 12-2 PAE 控制命令寄存器(PAECMD－F1H)

PAECMD		PAE 控制命令寄存器					F1H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	COM[3:0]			
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
COM[3:0]		运算命令控制位用于运算命令的控制： 0010，预计算，运算 H 参数 0110，模幂运算，其中包括 H 参数的运算 1010，模乘运算，其中包括 H 参数的运算 0100，模幂运算，其中不包括 H 参数的运算 1000，模乘运算，其中不包括 H 参数的运算 1101，GF(2^n)多项式乘法运算,A(x,m)*B(x,m) 1110，GF(2^n)多项式模余运算,C(x,2m-1)mod F(x,m) 1111，GF(2^n)多项式模乘运算, A(x,m)*B(x,m) mod F(x,m) 其它值，禁止设定				

12.2.3 PAE控制寄存器

表 12-3 PAE 控制寄存器(PAECR－F2H)

PAECR		PAE 控制寄存器					F2H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	Rev	Rev	Rev	RUN
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
RUN		RSA 模块运算控制状态位，此位软件置位，硬件清零					

	0，运算结束，RSA 处于空闲状态，可以读取运算结果 1，启动 RSA 后，保持到此次运算结束

12.2.4 PAE模长低位寄存器

表 12-4 PAE 模长低位寄存器（PAENLENL－F3H）

PAENLENL		PAE 模长低位寄存器					F3H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NLEN [7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
NLEN [7:0]	模长的低 8 位，与 PAENLENH 一起构成模长寄存器。模长寄存器最大值 2048。						

12.2.5 PAE模长高位寄存器

表 12-5 PAE 模长高位寄存器（PAENLENH－F4H）

PAENLENH		模长高位寄存器					F4H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	NLEN [11:8]			
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
NLEN [11:8]	模长的高 4 位 与 PAENLENL 一起构成模长寄存器。模长寄存器最大值 2048。						

12.2.6 PAE幂长低位寄存器

表 12-6 PAE 幂长低位寄存器（PAEELENL -F5H）

PAEELENL		PAE 幂长低位寄存器					F5H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ELEN [7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ELEN [7:0]	幂长的低 8 位。 与 PAEELENH 一起构成幂长寄存器。幂长寄存器用于表明 PAE 模幂运算时指数长度，其值介于 3～2048 之间。						

12.2.7 PAE幂长高位寄存器

表 12-7 PAE 幂长高位寄存器（PAEELENH－F6H）

PAEELENH		幂长高位寄存器					F6H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	ELEN [11:8]			
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ELEN [11:8]	幂长的高 4 位 与 PAEELENL 一起构成幂长寄存器。幂长寄存器用于表明 PAE 模幂运算时指数长度，其值介于 3～2048 之间。						

12.2.8 PAE模式寄存器

表 12-8 PAE 模式寄存器（PAEMOD－F7H）

PAEMOD		模式寄存器					F7H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	Rev	Rev	MODREG[1:0]	
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
MODREG[1:0]	设置 PAE 模块运算模式 00，缺省 1024Bit 模乘模幂运算模式 01，2048Bit 模乘运算模式 10，1024Bit 乘法运算模式 11，511Bit 内 GF(2^n) 多项式运算模式						

12.3 PAE Ram区域分配

Z8D168 片上核外有 1K Byte RAM 空间专门用于 PAE 等算法运算变量的存放，对应地址空间为 0x0C00 – 0x0FFF，数据在 PAE Ram 中按小端存放。在 RSA 运算过程中（PAECR.RUN＝1），PAE Ram 无法读写。

1、PAEMOD.MODREG[1:0]= 0x00, 1024Bit 模乘模幂运算时，PAE Ram 被分为 8 块逻辑地址，每块 128Byte，具体分配表如下：

地址	备注

0x0c00-0x0c7f	A 值和 A*B(modN)结果
0x0c80-0x0cff	内部 E 值
0x0d00-0x0d7f	内部暂存
0x0d80-0x0dff	内部暂存
0x0e00-0x0e7f	B 值 和 A^E(modN)结果
0x0e80-0x0eff	H 值
0x0f00-0x0f7f	N 值
0x0f80-0x0fff	内部暂存

注意：

- 计算模乘时运算结果放在 0x0C00-0x0C7F，计算模幂时运算结果放在 0x0E00-0x0E7F 中；
- 未说明的空间保留，严禁读写。

2、PAEMOD.MODREG[1:0]= 0x01，2048Bit 模乘运算时，PAE Ram 被分为 4 块逻辑地址，每块 256Byte，具体分配表如下：

地址	备注
0x0c00-0x0cff	A 值和 A*B(modN)结果
0x0d00-0x0dff	内部暂存
0x0e00-0x0eff	H 值 B 值
0x0f00-0x0fff	N 值

注意：

- 计算模乘时运算结果放在 0x0C00-0x0CFF；
- 未说明的空间保留，严禁读写。

3、PAEMOD.MODREG[1:0]= 0x2，1024Bit 乘法运算时，PAE Ram 被分为 4 块逻辑地址，每块 256Byte，具体分配表如下：

地址	备注
0x0c00-0x0cff	A 值 和 A*B 结果
0x0d00-0x0dff	内部暂存
0x0e00-0x0eff	B 值

0x0f00-0x0fff	内部暂存

注意：

- 计算乘法时运算结果放在 0x0C00-0x0CFF；
- 未说明的空间保留，严禁读写。

4、PAEMOD.MODREG[1:0]= 0x3，511Bit 内 GF(2^n) 多项式运算时，PAE Ram 被分为 6 块逻辑地址，每块 64Byte 及两块未用区域（分别为 256Byte 和 384Byte），具体分配表如下：

地址	备注
0x0c00-0x0c3f	A 值
0x0c40-0x0c7f	内部暂存
0x0c80-0x0cbf	C 低 64Byte,模结果 S
0x0cc0-0x0cff	C 高 64Byte
0x0d00-0x0dff	未用，可做缓存
0x0d00-0x0d3f	B 值
0x0d40-0x0d7f	R 值
0x0e80-0x0fff	未用，可做缓存

注意：

- 计算多项式乘法、模余及模乘时运算结果放在 0x0C80-0x0CBF，仅计算多项式乘法时计算结果放在 0x0CC0-0x0CFF 中。
- 0x0D40-0x0D7F 中存放的是 R 值，为模多项式 F 去掉最高比特位 1 的部分，即 F(x,m)=xm+R(x,m-1)；
- 未说明的空间保留，严禁读写。

12.4 软件操作流程

A. 1024Bit 模乘模幂运算，PAEMOD.MODREG[1:0]=0x00。

1) 写 PSAECSR 寄存器 AEMOD[1:0]位为 00；
2) 写 PAEMOD.MODREG[1:0]=0x00，设置 1024Bit 模乘模幂运算；
3) 写相应 PAE 参数寄存器，包括模长寄存器 PAENLEN 和幂长寄存器 PAENLEN；
4) 判断 PSAECSR[6]是否为 1，为 0 则等待，为 1 则进行下一步；

5) PAE Ram 写入待计算的数据。计算 A^Emod(N)包括模 N、幂 E 和计算数 A。计算 A*Bmod N 时包括模 N、计算数 A 和计算数 B；
6) 写计算命令寄存器 PAECMD，设置对应运算，写控制寄存器 PAECR 启动运算；
7) 查询 PAECR 寄存器判断是否运算结束，读取相应计算结果。运算结果放在 0x0C00-0x0C7F（计算模乘时）和 0x0E00-0x0E7F（计算模幂时）中。或者利用 PAEINT 中断处理程序中读取；
8) 在进行第一次模乘或者模幂运算时，必须载入 N 值，计算命令采用相应的 HAB 和 HAE 命令。以后运算如果 N 不变，则不需重新载入 N 值，计算命令可以采用不重新进行 H 值计算的 AB 和 AE 命令；
9) 进行模幂运算，如果 E 值不变，则也不需重新载入。进行模乘运算，如果 B 值不变，也不需要进行重新载入。

B. MODREG[1:0]=0x01，2048Bit 模乘运算模式

1) 写 PSAECSR 寄存器 AEMOD[1:0]位为 00；
2) 写 PAEMOD.MODREG[1:0]=0x01，设置 2048Bit 模乘运算模式；
3) 写模长 Nlen 寄存器 PAENLEN；
4) 判断 PSAECSR[6]是否为 1，为 0 则等待，为 1 则进行下一步；
5) 写数据到 PAE Ram，包括计算 A*Bmod N 时的模 N 和计算数 A。若是第一次模乘，必须载入 N 值，若以后的运算 N 不变，则不需要重新载入；
6) 写计算命令寄存器 PAECMD，设置 HAB 运算，写控制寄存器 PAECR 启动运算。
7) 查询 PAECR 寄存器或者利用 PAEINT 中断判断是否运算结束，结束则继续写计算 A*Bmod N 时的计算数 B 到相应内部缓冲区 RAM；
8) 写计算命令寄存器 PAECMD，设置 AB 运算；写控制寄存器 PAECR 启动运算。
9) 查询 PAECR 寄存器或者利用 PAEINT 中断判断是否运算结束，读取相应计算结果。运算结果放在 0x0C00-0x0CFF 中。

C. MODREG[1:0]=0x2，1024Bit 乘法运算模式

1) 写 PSAECSR 寄存器 AEMOD[1:0]位为 00；
2) 写 PAEMOD.MODREG[1:0]=0x02，设置 1024Bit 乘法运算模式；
3) 写模长 Nlen 寄存器 PAENLEN。（注：因为实际是采用模乘来完成乘法，故需要模数长度，Nlen=2*MAX(Alen,Blen)，N 值为全 2^Nlen-1）；
4) 判断 PAECSR[6]是否为 1，为 0 则等待，为 1 则进行下一步；

5) 写数据到 PAE Ram，包括计算 A*B 时的计算数 A 和计算数 B。（注：A 和 B 的输入均填充零按 2*MAX(Alen,Blen)长度输入）；
6) 写计算命令寄存器 PAECMD，设置 HAB 运算，写控制寄存器 PAECR 启动运算。
7) 查询 PAECR 寄存器或者利用 PAEINT 中断判断是否运算结束，读取相应计算结果，运算结果放在 0x0C00-0x0CFF 中。

D. MODREG[1:0]=0x3，512Bit 内 GF(2^n)多项式运算模式

1) 写 PSAECSR 寄存器 AEMOD[1:0]位为 00；
2) 写 PAEMOD.MODREG[1:0]=0x03，设置 512Bit 内 GF(2^n) ECC 多项式运算模式；
3) 写模长 Nlen 寄存器 PAENLEN；
4) 判断 PAECSR[6]是否为 1，为 0 则等待，为 1 则进行下一步；
5) 写相关数据到 PAE Ram，包括计算 A*B 或 A*B mod F 时的计算数 A 和计算数 B，计算 C mod F 的计算数 C，及计算 A*B mod F 或 C mod F 时的模 R；
6) 写计算命令寄存器 PAECMD，设置运算操作，写控制寄存器 PAECR 启动运算；
7) 查询 PAECR 寄存器或者利用 PAEINT 中断判断是否运算结束，读取相应计算结果，运算结果放在 0x0C80-0x0CBF 或 0x0CC0-0x0CFF 中。

第13章 随机数生成器（RNG）

13.1 概述

RNG 模块是一个完全硬件真随机数发生器。

13.2 特性

完全硬件真随机。

13.3 寄存器配置

13.3.1 RNGCOMMAND寄存器

表 13-1 RNGCOMMAND 寄存器(RNGCOMMAND—D5H)

RNGCOMMAND		RNGCOMMAND 寄存器					D5H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IPSEL	Rev	Rev	Seed GM	Reset	RNF	RDF	RMF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

IPSEL	1：表示选中真随机模块输出有效 0：表示选中伪随机模块输出有效，保留操作，一般不使用。
Seed GM	写 1 启动种子赋值操作，赋值完成硬件清零。 保留操作，一般不使用。
Reset	RNG 模块复位信号，初始化 RNG 模块时使用。
RNF	1：IPSEL 为 1 时，表示预读剩余随机数个数到 RNGNUM 寄存器。 0：无操作。
RDF	1：IPSEL 为 1 时，表示预读 1 字节随机数到 RNGDATA 寄存器。 0：无操作。
RMF	1：IPSEL 为 1 时，表示预读随机数模式配置到 RNGMODE 寄存器。 0：无操作。

13.3.2 随机数状态寄存器

表 13-2 随机数状态寄存器(RNGNUM—D4H)

RNGNUM		随机数状态寄存器					D4H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	RNGNUM[3:0]			
-	-	-	-	R			
0	0	0	0	0	0	0	0
RNGNUM[3:0]	可用真随机数的个数，最大值为 8。						

13.3.3 RNGMODE选择寄存器

表 13-3 RNGMODE 选择寄存器(RNGMODE—D6H)

RNGMODE		RNGMODE 选择寄存器					D6H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FD[3: 0]				LPF	Rev	MODE[1:0]	
R/W				R/W	-	R/W	
0	0	0	0	0	0	0	0
FD[3: 0]	分频因子：若 FD＝N（十进制表示），则 RNG 模块使用 2^N 分频核时钟工作						
LPF	节电模式：0——非节电模式，模拟电路工作 1——节电模式，模拟电路停止工作 低功耗模式下，使能节电模式可以关闭 RNG 模拟电路有效降低最低功耗。						
MODE[1:0]	工作模式：00——保留 01——正常模式 10——保留 11——保留 进行随机数生成时，必须配置成正常模式。						
在 RNGMODE 进行写操作之后，必须再等一个机器周期才可对 RNG 模块的其他寄存器进行读操作。							

13.3.4 RNG数据寄存器

表 13-4 RNG 数据寄存器（RNGDATA—D7H）

RNGDATA		RNG 数据寄存器					D7H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RNGDATA [7:0]							
R							
0	0	0	0	0	0	0	0
RNGDATA[7:0]		生成的真随机数					

13.4 软件操作流程

13.4.1 产生32bit真随机数

13.4.1.1 操作流程

1) 配置 RNGCOMMAND.IPSEL = 1，选择真随机模块输出有效；
2) 配置 RNGMODE 寄存器，选择正常模式，即 MODE＝0x01；
3) 配置 RNGCOMMAND. RNF = 1，预读剩余随机数个数到 RNGNUM 寄存器；
4) 从 RNGNUM 读取缓存数据剩余个数，判断是否有随机数待取；
5) 配置 RNGCOMMAND.RNF =0 , RNGCOMMAND. RDF＝1，预读 1 字节随机数到 RNGDATA 寄存器；
6) 从 RNGDATA 读取第 1 个字节；
7) 重复步骤 3-6，读取指定长度的随机数；
8) 配置 RNGCOMMAND.RNF =0，RNGCOMMAND.RDF=0，停止读数。

13.4.1.2 示例程序

```
void RNG_Init(void)
{
        CGUFCR |= 0x08; //RNG clk enable

    RNGCOMMAND = 0x88;
    RNGCOMMAND = 0x80;
    RNGMODE = 0x01;
}

void RNG_GetTrue(U8 *buf, U32 len)
{
    U32 data i;
    U8 data tmp;
```

```
        for(i=0; i<len; i++)
        {
            do
            {
                RNGCOMMAND = 0x84; // RNGCOMMAND.RNF = 1
                tmp = RNGNUM;
            }
            while(tmp == 0);

            RNGCOMMAND = 0x82; // RNGCOMMAND.RDF =1
            buf[i] = RNGDATA;
        }
    }
```

第14章 密钥生成引擎（KGE）

14.1 概述

KGE（Key Generate Engine）密钥对生成引擎，提供公钥算法密钥对生成过程中除法运算。

14.2 寄存器配置

14.2.1 被除数寄存器

表 14-1 被除数寄存器(KGEDND—ECH)

KGEDND		被除数寄存器					ECH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
KGEDND[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KGEDND[7:0]	被除数寄存器。2048 位被除数按照从高到低的顺序输入被除数寄存器，每次输入 8 位						

14.2.2 除数寄存器

表 14-2 除数寄存器 (KGESOR—EDH)

KGESOR		除数寄存器					EDH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
KGESOR [7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KGESOR [7:0]	除数寄存器						

14.2.3 余数寄存器

表 14-3 余数寄存器(KGERMN—EEH)

KGERMN	余数寄存器	EEH

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
KGERMN [7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KGERMN [7:0]		余数寄存器					

14.3 操作方法

单次求余运算：

```
CLR     A
MOV     A,   KGERMN                 清除余数寄存器
MOV     KGESOR,        #DATA        除数寄存器赋值
MOV     KGEDND,   #DATA             被除数寄存器赋值并启动运算
NOP
NOP
NOP
NOP
MOV     A,   KGERMN                 五个机器周期后读取余数结果
```

2048 位的正整数除以 8 位的素数：

```
CLR     A
MOV     A,   KGERMN                 清除余数寄存器
MOV     KGESOR,        #DATA        除数寄存器赋值
MOV     KGEDND,   #DATA             被除数寄存器赋值并启动运算
NOP
NOP
NOP
NOP
MOV     KGEDND,   #DATA             被除数寄存器赋值并启动运算
NOP
NOP
NOP
NOP
……
MOV     KGEDND,   #DATA             被除数寄存器赋值并启动运算
NOP
NOP
```

```
NOP
NOP
MOV     A,  KGERMN                  读取余数结果
```

注意：2048 位被除数按照从高到低的顺序输入被除数寄存器，每次输入 8 位。

第15章 AES算法单元

15.1 概述

AES（Advanced Encryption Standard）模块用来完成标准的 AES 算法加解密操作。

- 实现 128bit，192bit，256bit 密钥的 AES 的轮密钥扩展及加密解密运算
- 支持 EBC 模式和 CBC 模式的加密和解密

1. ECB 模式

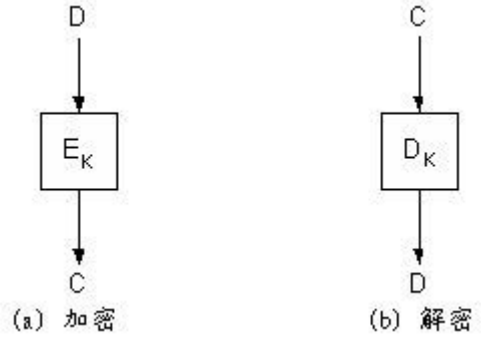


图 15-1 ECB 的运算流程图

2. CBC 模式

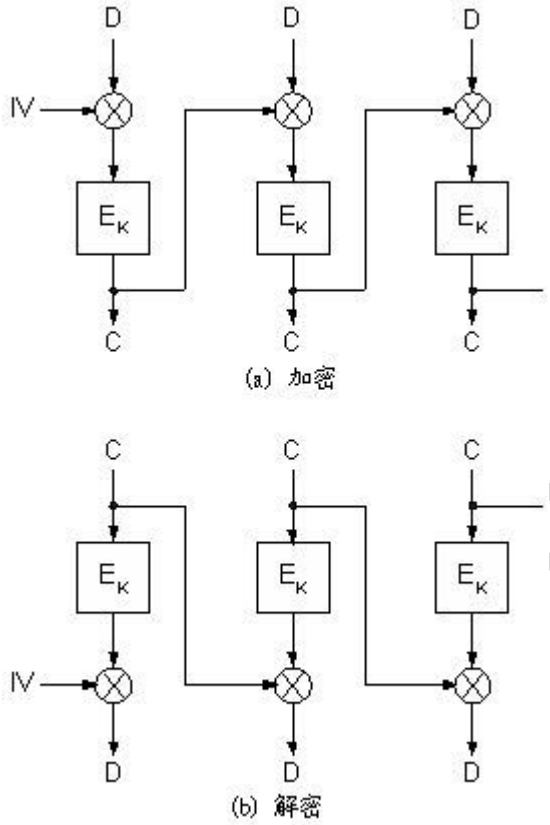


图 15-2 CBC 的运算流程图

15.2 寄存器配置

进行 AES 运算前，需要配置 PSAECSR.AEMOD[1:0]为 0x01，使能 AES 运算模式。

表 15-1 AES 控制寄存器(AESCSR—93H)

AESCSR		**AES 控制状态寄存器**					**93h**
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rsv.	Rsv.	Mode	Round[1:0]		OpMode[1:0]		Run
R	R	R/W	R/W		R/W		R/W
0	0	0	0		0		0
Mode		0：ECB 模式 1：CBC 模式					
Round[1:0]		00：128 比特密钥，加解密 10 轮 01：192 比特密钥，加解密 12 轮 10：256 比特密钥，加解密 14 轮 11：未定义					

OpMode[1:0]	00：轮密钥扩展运算 01：加密运算 10：解密运算 11：未定义
Run	0：AES 运算结束，处于空闲状态 1：启动 AES 运算，保持到此次运算结束

注意：

运算过程中，写 AES 寄存器无效。

运算完成，硬件自动清除 AESCSR.Run 位，其余位保留了上一次操作的状态。

15.3 RAM区域分配

AES 使用 PAE Ram 存放运算数据，0x0C00-0x0C3F 存放输入的运算数据和输出运算结果，0x0C40-0x0DFF 存放运算中间数据。

所有数据分组都是按照字节大端模式存储，即假设 1 个 128 比特的数 A：C79075B1……F43D12C8，最高字节数据 0x C7 将会存储在地址 0x0C00，数据 0x90 存储在地址 0x0C01，依次类推，数据 0x12 存储在地址 0x0C0E，数据 0xC8 存储在地址 0x0C0F；需要特别声明的是针对初始化密钥 IK，由于支持 128 比特、192 比特、256 比特不同密钥长度，它对应的存储区域也分别是：0x0C20~0x0C2F、0x0C20~0x0C37、0x0C20~0x0C3F。

地址	备注
0x0c00-0x0c0f	明文/密文
0x0c10-0x0c1f	IV（CBC 模式应用）
0x0c20-0x0c2f	初始化密钥 IK
0x0c30-0x0c3f	初始化密钥 IK/轮密钥 1 RK1
0x0c40-0x0d2f	保留为算法使用
0x0d30-0x0dff	未用

15.4 软件操作流程

1) 写 PSAECSR 寄存器 AEMOD[1:0]位为 0x01。
2) 写入 ROUND 控制和 CBC/ECB 模式控制到 AESCSR 寄存器。
3) 判断 PSAECSR. PRAMRDY 位是否为 1，为 0 则等待，为 1 则进行下一步

4) 写初始密钥到 0x0C20~0x0C2F/0x0C37/0x0C3F PAE Ram 区域。
5) 写 AESCSR 寄存器的 OpMode=00，并置位 AESCSR.Run 控制位启动轮密钥扩展运算。
6) 读 PSAECSR. SAERDY 位或者 AESCSR.Run 位（根据此位判断时需要软件再延时等待 10 个 NOP 左右）等待轮密钥扩展结束。
7) 如果是 CBC 模式操作且是第一个分组数据，写入 IV 到 0x0C10~0x0C1F PAE Ram 区域；否则进行下一步。
8) 写入明文/密文分组数据到 0x0C00~0x0C0F PAE Ram 区域。
9) 写 AESCSR 寄存器 OpMode=2’b01/10，并置位 AESCSR.Run 控制位启动加密/解密运算。
10) 读 PSAECSR. SAERDY 位或者 AESCSR.Run 位（根据此位判断时需要软件再延时等待 10 个 NOP 左右）等待加解密结束。
11) 从 0x0C00~0x0C0F PAE Ram 区域读出加密/解密结果。
12) 如果继续下一个分组加解密，重复步骤 7~10；如果需要更新密钥，重复步骤 4~11。

15.5 算法流程

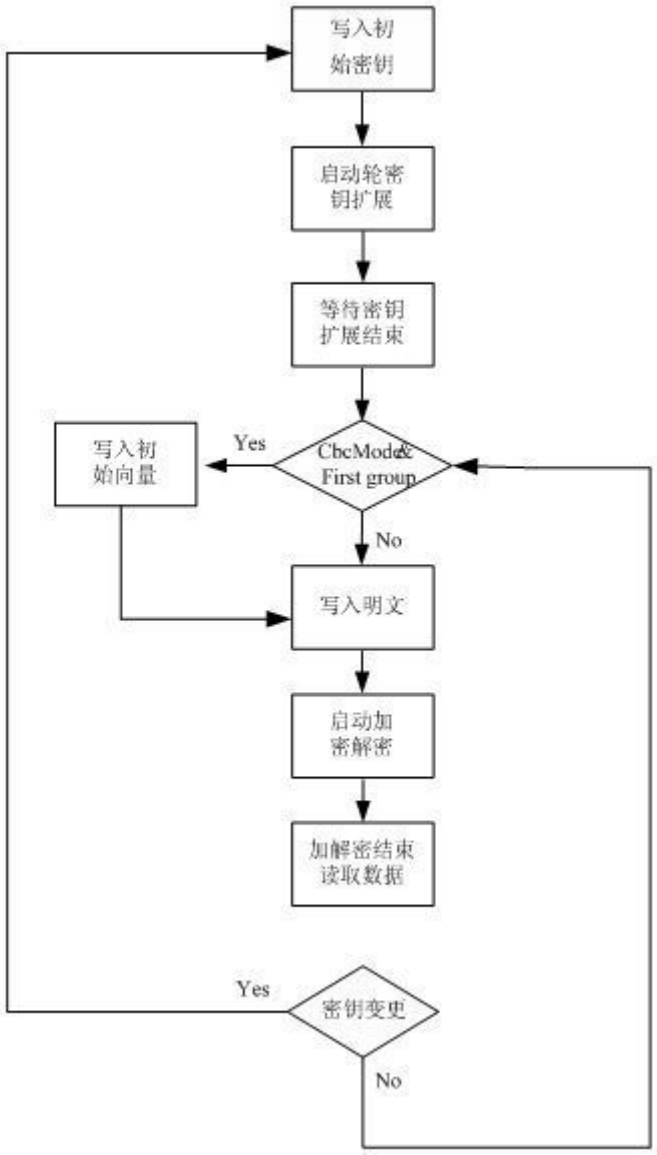


图 15-3 AES 运算流程

第16章 SSF33算法单元

16.1 概述

SSF33 算法单元，完成国密 SSF33 对称算法加解密运算。

16.2 寄存器配置

进行 SSF33 前运算，需要配置 PSAECSR.AEMOD[1:0]为 0x03，并确保 PSAECSR.SHFTMOD 位 0，使能 SSF33 运算模式。SSF33 控制寄存器和移位控制寄存器复用。

表 16-1 SSF33 控制寄存器(SSFCSR—96H)

SSFCSR		控制状态寄存器					96h
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RANGE					DIR	CMD	RUN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

RUN	运算或移位功能使能寄存器，由软件置位，硬件清零 0：运算或移位结束，功能模块处于空闲状态 1：启动运算，保持到此次运算结束
CMD	SSF33 加解密时，表示控制运算类型 移位功能时，表示移位类型 0，解密运算，非循环移位 1，加密运算，循环移位
DIR	加解密时，表示密钥扩展或者加解密；移位时，移位方向控制 0，密钥扩展操作；向左移位 1，加解密操作；向右移位
RANGE	00~31，表示移位的范围，向左或者向右移 RANGE+1（即 1~32）位

16.3 RAM区域分配

SSF33 模块使用 PAE Ram 存放运算数据。SSF33 运算模式下非 BOOT 区程序无法直接访问 PAE Ram 区，非 BOOT 区程序只可以通过 DMA 访问 PAE Ram 中加解密数据存储区，其他区域无法访问。

地址	备注
0x0f60-0x0f6f	明文
0x0f70-0x0f7f	密文

16.4 软件操作流程

参见《Z8D168 算法库使用手册》

第17章 SCB2算法单元

17.1 概述

SCB2 算法单元，完成国密 SCB2 对称算法加解密运算。

17.2 寄存器配置

进行 SCB2 运算，需要配置 PSAECSR.AEMOD[1:0]为 0x02，使能 SCB2 运算模式

表 17-1 SCB 控制状态寄存器(SCBCSR—95H)

SCBCSR		SCB 控制状态寄存器					95h
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rsv.	Rsv.	Rsv.	Rsv.	Enc Mode	TotalLoop		RUN
R	R	R	R	R/W	R/W		R/W
0	0	0	0	0	0	0	0

EncMode	表示进行解密运算 表示进行加密运算
TotalLoop[1:0]	00，进行 14 轮运算模式 01，进行 12 轮运算模式 10，进行 10 轮运算模式 11，进行 8 轮运算模式
RUN	运算使能寄存器，由软件置位，硬件清零 0：运算结束，SCB 处于空闲状态； 1：启动运算后，保持到此次运算结束；

17.3 RAM区域分配

SCB2 模块使用 PAE Ram 存放运算数据。SCB2 运算模式下非 BOOT 区程序无法直接访问 PAE Ram 区，非 BOOT 区程序只可以通过 DMA 访问 PAE Ram 中加解密数据存储区，其他区域无法访问。

地址	备注
0x0d40-0x0d43	明文/密文 13-16 字节
0x0d44-0x0d47	明文/密文 5-8 字节

0x0f40-0x0f43	明文/密文 9-12 字节
0x0f44-0x0f47	明文/密文 1-4 字节

17.4 软件操作流程

参见《Z8D168 算法库使用手册》

第18章 移位运算单元

18.1 概述

32 位移位运算单元，完成加密算法运算过程中常用的数据移位操作。

18.2 寄存器配置

进行 32 位移位运算前，需要配置 PSAECSR.SHFTMOD 位为 1，并确保 PSAECSR.AEMOD[1:0]不为 0x03，使能 32 位移位运算模式。SSF33 控制寄存器和移位控制寄存器复用。

表 18-1 SSF33 控制寄存器(SSFCSR—96H)

SSFCSR		控制状态寄存器					96h
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RANGE					DIR	CMD	RUN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

RUN	运算或移位功能使能寄存器，由软件置位，硬件清零。 0：运算或移位结束，功能模块处于空闲状态； 1：启动运算，保持到此次运算结束；
CMD	SSF33 加解密时，表示控制运算类型； 移位功能时，表示移位类型； 0，解密运算，非循环移位 1，加密运算，循环移位
DIR	加解密时，表示密钥扩展或者加解密；移位时，移位方向控制 0，密钥扩展操作；向左移位 1，加解密操作；向右移位
RANGE	00~31，表示移位的范围，向左或者向右移 RANGE+1（即 1~32）位

表 188-2 移位数据寄存器(SHFTDR—97H)

SHFTDR		移位数据寄存器					97h
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SHFTDAT[7:0]							

R/W							
0	0	0	0	0	0	0	0
SHFTDAT[7:0]		移位寄存器输入输出数据。					

18.3 软件操作流程

1) 写 PAECSR 寄存器 SHFTMOD 位为 1。（注：此时 AEMOD[1:0]不能为 2’b10）
2) 设置 SSFCSR 的 RANGE 和 DIR 控制位；
3) 将待移 32 位数据写入 SHFTDR，并写 SSFCSR.RUN 为 1 启动移位运算；

- 往 SHFTDR 寄存器写 32bit 待移位数据 X[31:24]；
- 往 SHFTDR 寄存器写 32bit 待移位数据 X[23:16]；
- 往 SHFTDR 寄存器写 32bit 待移位数据 X[15:8]；
- 往 SHFTDR 寄存器写 32bit 待移位数据 X[7:0]；
- SSFCSR.RUN=1

4) 读 SSFCSR.RUN 比特，等待移位运算结束；
5) 读出 SHFTDR 数据。

- 从 SHFTDR 寄存器读出 32bit 移位数据结果 Z[31:24]；
- 从 SHFTDR 寄存器读出 32bit 移位数据结果 Z [23:16]；
- 从 SHFTDR 寄存器读出 32bit 移位数据结果 Z [15:8]；
- 从 SHFTDR 寄存器读出 32bit 移位数据结果 Z [7:0]；

第19章 DMA传输单元

19.1 概述

DMA 传输单元，用于 PAE、AES 等运算过程中数据搬移的加速。

19.2 寄存器配置

19.2.1 DMA模式控制状态寄存器

表 19-1 DMA 模式控制状态寄存器 (DMACSR－CFH)

DMACSR		DMA 模式控制状态寄存器					CFh
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DMA RUN	.DMADIR		DMA LEN	PBASE			
R/W	R/W		R/W	R/W			
0	0		0	0	0	0	0

DMARUN	0：DMA 空闲 1：DMA 运行状态 该位由软件置 1，硬件清零
DMADIR[1:0]	00：XRAM -> PRAM 01：PRAM -> XRAM 10：XRAM -> USB EP3FIFO 11：USB EP4FIFO -> XRAM
DMALEN	0：DMA 传输长度为 16 字节； 1：DMA 传输长度为 32 字节；
PBASE[3:0]	该基址控制位仅在非 AES/SCB2 模式下有效。 当为 SSF33 DMA 模式时，仅第 0 位有效，其含义如下： PBASE[0] = 0 ：表示 DMA 操作基址为 0x0F60； PBASE[0] = 1 ：表示 DMA 操作基址为 0x0F70； 其它模式，即 PAE(RSA&ECC)或普通模式时对应的 PRAM 基址如下： 4'h0：对应 PRAM 基址为 0x0C00；

	4'h1：对应 PRAM 基址为 0x0C20； 4'h2：对应 PRAM 基址为 0x0C40； 4'h3：对应 PRAM 基址为 0x0C60； 4'h4：对应 PRAM 基址为 0x0D00； 4'h5：对应 PRAM 基址为 0x0D20； 4'h6：对应 PRAM 基址为 0x0D40； 4'h7：对应 PRAM 基址为 0x0D60； 4'h8：对应 PRAM 基址为 0x0E00； 4'h9：对应 PRAM 基址为 0x0E20； 4'ha：对应 PRAM 基址为 0x0E40； 4'hb：对应 PRAM 基址为 0x0E60； 4'hc：对应 PRAM 基址为 0x0F00； 4'hd：对应 PRAM 基址为 0x0F20； 4'he：对应 PRAM 基址为 0x0F40； 4'hf：对应 PRAM 基址为 0x0F60； 由于 DMALEN 最大支持 32 字节，因此可知 DMA 操作涵盖了 PRAM 以下地址范围： 0x0C00~0x0C7F，0x0D00~0x0D7F，0x0E00~0x0E7F，0x0F00~0x0F7F。 从本文的 RAM 分配可以看出，除去 1024 比特的模幂运算、1024 比特大数乘法运算、2048 比特的模乘运算参数无法全部用 DMA 传输外，其它模式的运算都可以采用 DMA 加速传输，节省 CPU 搬运数据时间，有效提高如 ECC 算法的加密速度。

19.2.2 DMA模式XRAM基址寄存器

表 19-2 DMA 模式 XRAM 基址寄存器 (DMAXBASE－DFH)

DMAXBASE		DMA 模式 XRAM 基址寄存器					DFh
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XBASE							
R/W							
0							

XBASE[7:0]	对应 XRAM 的[11：4]比特位，XRAM 寻址范围为：0x0000~0x0BFF. DMA 操作模式时，XRAM 中的数据全部按照字节小端模式存取。 注意：每进行一次 DMA 操作，XBASE 将会硬件自动加 1 或 2，由 DMA 长度决定，16 字节加 1，32 字节加 2。

19.2.3 DMA模式XRAM偏移地址寄存器

表 19-3 DMA 模式 XRAM 偏移地址寄存器 (DMAXOFS－EBH)

DMAXOFS		**DMA 模式 XRAM 偏移地址寄存器**					**EBh**
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rsv				XOFFSET			
R				R/W			
0				0			
XOFFSET[3:0]	对应 XRAM 的[3：0]比特位，DMA 操作模式时，XRAM 中的数据全部按照字节小端模式存取。偏移地址设置不变。						

19.3 软件操作流程

1. 配置 DMACSR.DMADIR 设置传输方向。
2. 配置 DMACSR.PBASE、DMAXBASE、DMAXOFS，设置 DMA 传输 XRAM 和 PAE Ram 地址。
3. 置位 DMACSR.DMARUN 位，启动 DMA 传输。
4. 查询 DMA.DMARUN 位，等待 DMA 传输完成。

第20章 安全防护单元（SEC）

20.1 概述

SEC 模块完成外部高低电压检测和频率异常检测，按照配置产生中断或者复位。

20.2 寄存器配置

20.2.1 SEC控制寄存器

表 20-1 SEC 控制寄存器(SECCR—E1H)

SECCR		SEC 控制寄存器					E1H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	HVDEN	Rev	LVDEN	Rev	FDEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
HVDEN	高电压检测使能 1：使能电压检测电路的高电压检测功能； 0：关闭电压检测电路的高电压检测功能，并清除状态寄存器中的高电压警号信号；						
LVDEN	低电压检测使能 1：使能电压检测电路的低电压检测功能； 0：关闭电压检测电路的低电压检测功能，并清除状态寄存器中的低电压警号信号；						
FDEN	频率检查模块使能 1：使能频率检测电路； 0：关闭并且复位频率检测电路，清除状态寄存器中的高频率以及低频率警号信号；						

20.2.2 SEC状态寄存器

表 20-2 SEC 状态寄存器（SECSR—E3H)

SECSR	SEC 状态寄存器	E3H

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	HFD	LFD	HVD	LVD	Rev
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

HFD	高频率告警标志 频率检测使能后，如果产生高频率告警，则 HFD 为 1，关闭频率检测使能后，HFD 清 0。
LFD	低频率告警标志 频率检测使能后，如果产生低频率告警，则 LFD 为 1，关闭频率检测使能后，LFD 清 0。
HVD	高电压告警标志： 高电压检测使能后，如果产生高电压告警，则 HVD 为 1，关闭高电压检测使能后，HVD 清 0。
LVD	低电压告警标志： 低电压检测使能后，如果产生低电压告警，则 LVD 为 1，关闭低电压检测使能后，LVD 清 0。

注意：

- 高、低压检测告警中断每次都会被响应。
- 高频率告警产生后，如果不清除高频告警标志，高频率中断就不会再产生
- 低频率告警产生后，如果不清除低频告警标志，低频率中断就不会再产生

第五部分 外部接口控制

第21章 IO控制器(IOM)

21.1 概述

IOM(IO Manager)负责管理 Z8D168 中的 IO，完成 IO 控制和复用。

21.2 寄存器配置

21.2.1 IO模式控制寄存器

表 21-1 IO 模式控制寄存器(IOMCR—9CH)

IOMCR		IO 模式控制寄存器					9CH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCDU	SPI	SCC	ResUp	RstLv	RstEn	RstMd	UART
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

UART	UART 模式选择： 0：非 UART 模式 1：UART 模式 UART 模式不能和 SCD 模式共存
RstMd	非 SCD 模式下 ScdRst_Pad 引脚工作模式： 0：作为复位 1：作为 GPIO SCD 模式下此控制位无效
RstEn	复位有效： 0：复位产生中断而不带来系统复位 1：复位带来系统复位
RstLv	非 SCD 模式下外部复位有效电平或者复位中断有效沿信号： 0：低电平/下降沿；1：高电平/上升沿。
ResUp	ResUp_Pad 输出控制，仅在 USB 模式下有效，SCD 模式下 ResUp_Pad 作为 GPIO7 使用。
SCC	SCC 模式选择： 0：非 SCC 模式 1：SCC 模式

SPI	SPI 模式选择 0：非 SPI 模式 1：SPI 模式
SCDU	SCD 模式下启动 USB_LS 模式 0：不使能 USB_LS 模式 1：使能 USB_LS 模式，只能在 SCD 模式下配置该位

21.2.2 IO模式控制寄存器2

表 21-2 IO 模式控制寄存器 2(IOMCR2—9EH)

IOMCR2		IO 模式控制寄存器 2					**9EH**
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GPIO Sel	ScdAs Scc	ClkPhy OE	ClkScc Sel	SccD2 Scd	SccO	ScdO	ScL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

ScL	Scc，Scd 接口连通模式选择： 0：Scc 与 Scd 接口不直接相连 1：Scc 与 Scd 接口直接相连 只有在 Scd 和 Scc 模式同时有效时才可以设置
ScdO	Scc，Scd 接口直接连接时，Scd SIO 输入输出控制位： 0：Scd 的 Sio 作为输入信号 1：Scd 的 Sio 作为输出信号 只有在 Scd 和 Scc 模式同时有效时才可以设置
SccO	Scc，Scd 接口直接连接时，Scc SIO（GP0_Pad）输入输出控制位： 0：Scc 的 Sio 作为输入信号 1：Scc 的 Sio 作为输出信号 只有在 Scd 和 Scc 模式同时有效时才可以设置
SccD2Scd	Scc，Scd 接口直接连接时，Scd 数据输入来自于 SIO_Pad 或者是 Scc SIO（GP0_Pad） 0：Scd 数据输入来自于 SIO_Pad 1：Scd 数据输入来自于 Scc SIO（GP0_Pad） 只有在 Scd 和 Scc 模式同时有效时才可以设置

ClkSccSel	在 Scd 和 Scc 模式同时打开时决定 SccClk_Pad 是由 SCC 模块时钟输出，或者是读卡器时钟 1：直接来自于读卡器 0：由 SCC 模块提供 只有在 Scd 和 Scc 模式同时有效时才起作用。
ClkPhyOE	在 USBLS 模式下，设置该位，则 ScdClkPad 输出 ClkPhy 用于检测 1：输出 ClkPhy 0：ScdClkPad 继续用于 GPIO
ScdAsScc	在 SCD 与 SCC 同时打开的时候选择 Scd 数据口为 GP0_Pad 或者是 Sio_Pad 1：数据口选择为 GP0_Pad 0：数据口选择为 Sio_Pad
GPIOSel	选择当特殊寄存器访问 2'b01 地址时访问的是 GPIOCR 还是 GPIOCR2 0：对 GPIOCR 操作及监测 1：对 GPIOCR2 操作及监测

21.2.3 GPIO方向控制寄存器

表 21-3 GPIO 控制寄存器(GPIOCR—9DH)

GPIOCR		GPIO 控制寄存器					9DH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GPIODIR[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
GPIODIR[7:0]	8 位 GPIO 输入输出方向控制。 0 为输入，1 为输出。 注意：GPIOCR[7]为 0，则 GPIO7 从 P2_7 读回的值为常 1，而 GPIOCR[7]为 1 的时候，可以从 P2_7 输出到 GPIO7 上。						

21.2.4 GPIO方向控制寄存器2

表 21-4 GPIO 控制寄存器 2(GPIOCR2—9DH)

GPIOCR2		GPIO 控制寄存器 2					9DH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Res	Res	Res	GPIODIR2[4:0]				
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
GPIODIR2[4:0]		5 位 GPIO 输入输出方向控制。 0 为输入，1 为输出。					

注意：GPIOCR2 和 GPIOCR 寄存器地址复用，置位 IOMCR2.GPIOSel 选择 GPIOCR2 寄存器。

21.3 GPIOCR对应的端口和访问控制方式

GPIOCR 对应的端口和访问控制方式：

表 21-5 GPIO 的访问控制

GPIO	Pad	方向控制	访问
0	GP0_Pad	Sfr.GPIOCR [0]	P2.0
1	GP1_Pad	Sfr.GPIOCR [1]	P2.1
2	SccClk_Pad	Sfr.GPIOCR [2]	P2.2
3	Tx_Pad	Sfr.GPIOCR [3]	P2.3
4	SIO_PAD	Sfr.GPIOCR [4]	P2.4
5	ClkScd_Pad	Sfr.GPIOCR [5]	P2.5
6	ScdRst_Pad	Sfr.GPIOCR [6]	P2.6
7	ResUp_Pad*	Sfr.GPIOCR [7]	P2.7
8	GP8_Pad	Sfr.GPIOCR2 [0]	P0.0
9	GP9_Pad	Sfr.GPIOCR2[1]	P0.1
10	GP10_Pad	Sfr.GPIOCR2[2]	P0.2
11	GP11_Pad	Sfr.GPIOCR2[3]	P0.3
12	GP12_Pad	Sfr.GPIOCR2[4]	P0.4

* ResUp_Pad 作为 GPIO 使用时，只能输出，作为输入设置后，读回的值常为 1。

在 SCD 模式下，设置 GPIO12 为输入，如果使能了 XIE.ExtIntEn 位，GPIO12 引脚上低电平将会产生中断。

第22章 SPI 接口

22.1 概述

SPI 接口完成芯片与外围器件同步串行通讯接口。

22.1.1 芯片特性：

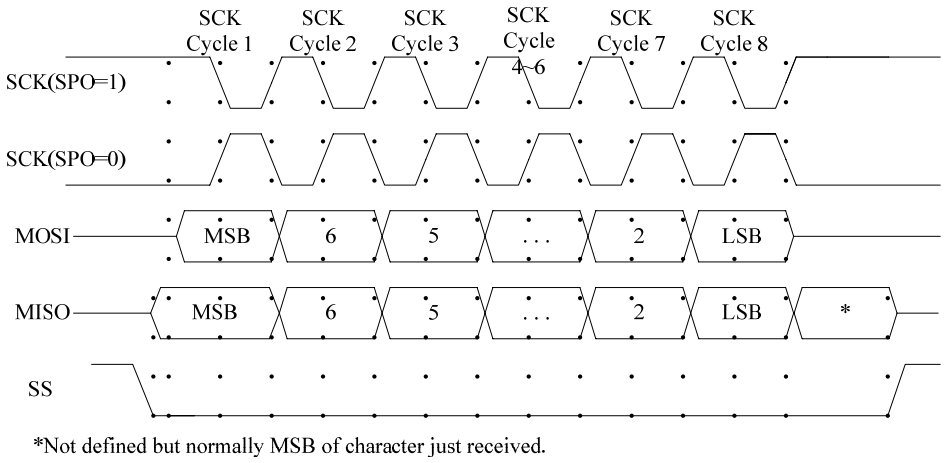
1. 全双工、四线同步传输
2. 工作在主模式下
3. 本地环路功能，可以方便地进行测试
4. 可编程的时钟相位和极性
5. 独立编程的波特率发生器
6. 最大位传输速度为 10Mbps
7. 可编程的传输位设置

22.1.2 通讯原理

SPI（Serial Peripheral Interface）是 Motorola 公司推出的一种同步串行接口，是一种全双工、三线通信的系统。它允许处理器与各种外围设备以串行方式进行通信。在 SPI 接口中，数据的传输需要一个时钟信号和两条数据线。

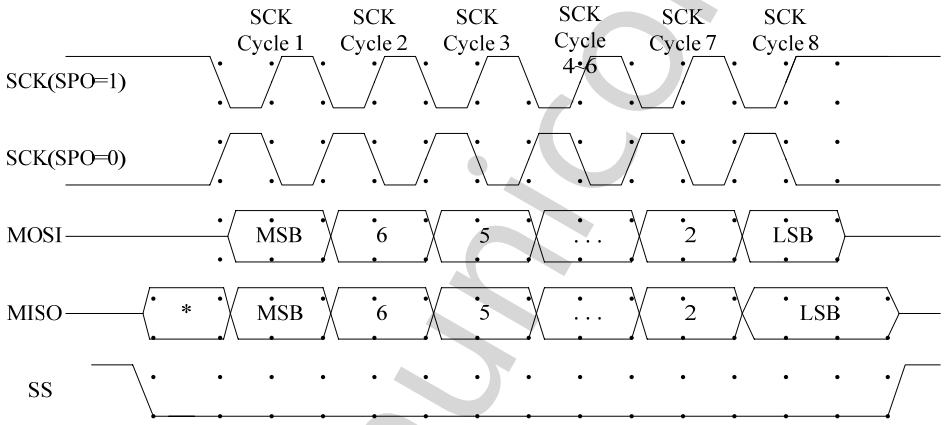
SPI 总线由四根线组成：串行时钟线（SCK），主机输出/从机输入线（MOSI），主机输入/从机输出线（MISO），还有一根是从机选择线（SS），它们在与总线相连的各个设备之间传送信息。

SPI 总线中所有的数据传输由串行时钟 SCK 来进行同步，每个时钟脉冲传送 1 比特数据。SCK 由主机产生，是从机的一个输入。时钟的相位（SPH）与极性（SPO）可以用来控制数据的传输。SPO=0 表示 SCK 处于空闲状态时为低电平，SPO=1 表示 SCK 处于空闲状态时为高电平。时钟相位（SPH）可以用来选择两种不同的数据传输模式，SPH=0 表示数据在信号 SS 声明后的第一个 SCK 边沿有效，SPH=1 表示数据在信号 SS 声明后的第二个 SCK 边沿才有效。因此，主机与从机中 SPI 设备的时钟相位和极性必须要一致才能通信。



*Not defined but normally MSB of character just received.

SPH=0 时 SPI 的数据传输模式



*Not defined but normally LSB of previously transmitted character.

SPH=1 时 SPI 的数据传输模式

传输过程包括 8 个 SCK 的时钟周期和开始阶段的初始化以及结束阶段。开始和结束阶段决定于 SPH 的模式选择而不用管 SPI 是工作从模式还是主模式。当 SPI 工作于主模式时，SPI 选择的时钟频率也会影响到其初始阶段。

从模式下，传输过程是从 SCK 的第一个时钟边沿还是从 SS 的下降沿开始决定于 SPH 的选择。当 SPH=0 时，SS 的下降沿表示传输开始。当 SPH=1 时，SCK 的第一个时钟沿表示传输的开始。无论在哪种 SPH 模式下，传输过程都可以将 SS 拉高来结束。

主模式下，传输开始于软件操作往 SPDR 寄存器里面写数据，SPH 对传输的开始过程的延迟没有影响，但是影响到了时钟信号的初始状态。当 SPH=0 时，在第一个 SCK 时钟周期的前半个周期保持在其处于空闲状态的值；当 SPH=1 时，SCK 的第一个时钟周期由 SCK 从空闲状态的值跳转到有效电平开始。传输过程结束于第 8 个时钟周期，对于

SPH=0，传输过程结束于最后一个时钟沿；而对于 SPH=1，SCK 的第 8 个时钟周期后半个周期始终处于空闲状态时的电平。

22.2 寄存器配置

SPI 模块共有 7 个寄存器。下面分别给出了各寄存器的数据结构和功能定义的说明。

22.2.1 SPI通讯状态控制寄存器

表 22-1 SPI 通讯状态控制寄存器 (SPICSCR－C8H)

SPICSCR		SPI 通讯状态控制寄存器					C8H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPH	SPO	rev	rev	DSS			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPH		SPH=0 表示在第一个时钟沿采样输出数据 SPH=1 表示在第二个时钟沿采样输出数据。					
SPO		SPO=0 表示传输空闲时输出时钟为低电平 SPO=1 表示传输空闲时输出时钟为高电平。					
DSS		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					

22.2.2 SPI波特率控制寄存器1

表 22-2 SPI 波特率控制寄存器 1(SPIBDCR1－C9H)

SPIBDCR1		SPI 波特率控制寄存器 1					C9H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

SCR							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SCR	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： CPU 时钟 CPSDVSR*(1+SCR) CPSDVSR 是在 SPIBDCR2.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。						

22.2.3 SPI工作状态控制寄存器

表 22-3 SPI 工作状态控制寄存器(SPIWSCR－CAH)

SPIWSCR		SPI 工作状态控制寄存器					CAH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rev	rev	rev	SSE	LBM	RORIE	TIE	RIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SSE	SSE =0：表示 SPI 操作不使能。 SSE =1：表示 SPI 操作使能。						
LBM	LBM =0：表示正常工作模式。 LBM =1：表示环路测试状态。						
RORIE	RORIE =0：表示接收 FIFO 溢出中断不使能。 RORIE =1：表示接收 FIFO 溢出中断使能。						
TIE	TIE =0：表示发送 FIFO 在半满或更少的时候，不产生发送中断。 TIE =1：表示发送中断使能，发送 FIFO 在半满或更少的时候，产生发送中断。						
RIE	RIE =0：接收 FIFO 在半满或更少的时候，不产生接收中断。 RIE =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。						

22.2.4 SPI数据寄存器

表 22-4 SPI 数据寄存器(SPIDR－CBH)

SPIDR		SPI 选择寄存器					CBH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPIDR							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIDR	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 发送 FIFO 深度 2 级，接收 FIFO 深度 4 级						

22.2.5 SPI状态寄存器

表 22-5 SPI 状态寄存器(SPISR－CCH)

SPISR		SPI 状态寄存器					CCH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rev	rev	rev	BSY	RFF	RNFF	TNF	TFE
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
BSY	BSY =0：表示 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。						
RFF	接收 FIFO 满时该位置 1，否则置 0						
RNFF	接收 FIFO 不满时该位置 1，否则置 0						
TNF	发送 FIFO 不满时该位置 1，否则置 0						
TFE	发送 FIFO 空时该位置 1，否则置 0						

22.2.6 SPI波特率控制寄存器2

表 22-6 SPI 波特率控制寄存器 2 (SPIBDCR2－CDH)

SPIBDCR2	SPI 时钟预定标器寄存器	CDH

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPSDVSR							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
CPSDVSR	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 该寄存器不允许设置为“0x00”和“0x01”						

22.2.7 SPI中断标识/清除寄存器

表 22-7 SPI 中断标识/清除寄存器(SPIIIR/SPIICR－CEH)

SPIIIR/SPIICR		SPI 中断标识/清除寄存器					CEH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rev	rev	rev	rev	rev	RTIS	TIS	RIS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
RTIS		接收 FIFO 溢出中断 SPI 接收中断产生时，此位置 1，否则置 0					
TIS		发送请求中断产生时，此位置 1，否则置 0					
RIS		接收请求中断产生时，此位置 1，否则置 0					
该中断寄存器所包含的所有状态位都是软件写 0 清零，硬件置 1。							

第23章 智能卡设备(SCD)接口

23.1 概述

智能卡设备（以下简称 SCD）支持与 ISO/IEC 7816-3 标准 T = 0 和 T = 1 传输协议兼容的智能卡读卡器。在智能卡芯片中，SCD 模块在智能卡 CPU 的控制下完成与读卡器的串行通讯。

注意：SCD 接口只有在卡封装的芯片中有效。

23.1.1 功能特性

- 符合 ISO / 7816-3 标准
- 支持异步字符(T0)传输协议：
  - 数据长度：8bit
  - 奇偶校验位生成以及奇偶错误检测
  - 自动错误检测并报告以及 I/O 线上数据重传
  - 支持正向约定(低位在前)，支持反向约定(高位在前)
- 支持异步块传输协议(T=1) (软件层实现)
- 8bit 4 级的接收 FIFO
- 可编程波特率(波特率可以根据参数 F/D 调整)
- 支持数据通讯及错误处理中断

23.2 基本原理

23.2.1 传送格式

一帧数据至少有 12 个ETU周期长(ETU即基本时间单位：这是传送一个位所需的时间)，传送器发出一个起始位，紧接着为一个数据字节加上一个奇偶位校验位。在发送模式中，发送完 8 位数据后，由硬件自动计算奇偶位；在奇偶位发完之后，I/O线返回高阻态，保持至少两个ETU。如 图 23-1所示。

	sb	b0	b1	b2	b3	b4	b5	b6	b7	pb	guard time	sb

sb: 起始位
pb: 奇偶校验位
b0～b7: 数据位
guard time: 保护时间,为至少2个ETU
I/O线空闲时期为高阻态

图 23-1 SCD ISO-7816 发送/接收数据格式

23.2.2 接收模式

通过清除 SCDCSR 中的 TRS 位来选择接收模式，接收器接收串行数据流并将其转换成一个并行字符。接收器等待起始位并确认它，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 SCDDR(接收 FIFO)。当发现奇偶校验错时，设置 SCDISR.TRE 位。接收器停止将接收字节写入接收 FIFO，同时在 10.5etu 到 12etu 间将 SIO 管脚驱动为低电平等待错误字符的重发。

23.2.3 发送模式

首先通过设置 SCDCSR 中的 TRS 位来选择发送模式，数据被装入 SCDDR 寄存器后，在 I/O 线上输出一个起始位，在起始位之后，数据和奇偶校验位被移出，IO 线在起始位之后在 10ETU 处返回高阻态。

在起始位之后 11ETU 处发生中断，IO 线被检测来发现可能的错误信号，SCDISR 中的状态位也被更新。软件检查错误标志位决定是否重传。

下一字节的发送，须经过 2 个 ETU 之后，清除 SCDISR 中的 TXEND 标志位, 装入数据后，就会开始发送，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，SCD 开始将 SCDDR 的数据传送发送器，然后通过 SIO 管脚发送数据，以起始位开始，奇偶校验位结束。

23.2.4 数据重传

■ SCD 接收模式下的重传：

图 23-2描述了SCD接收模式下的重传操作(SCDCSR.TRS = 0)。

1. 检查接收的奇偶校验位，如果发生奇偶校验错，SCDISR.TRE 位自动设置为 1，SIO 管脚发出错误信号通知读卡器重传当前数据。此时，接收器中接收的数据不会写入 SCDDR。在采样下一个奇偶校验位时确保 SCDISR.TRE 位已清零。

2. 检查接收的奇偶校验位，如果没有发生奇偶校验错，SCDISR.TRE 位不会被设置为 1，接收器中接收的数据自动写入 SCDDR。

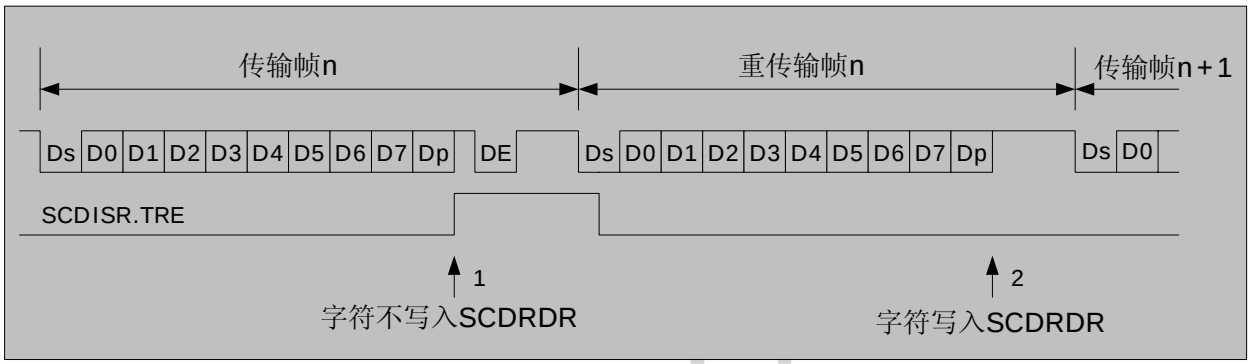


图 23-2 接收模式下数据重传(T = 0)

■ SCD 发送模式下的重传：

图 23-3描述了SCD发送模式下的重传(SCDCSR.TRS = 1)。

1. 当一帧传送完毕后，当从卡返回一个错误信号，则 SCDISR.TRE 位设置为 1。如果设置为允许硬件自动重传(SCDCSR. RE3_EN)，此时 SCD 发送器不从 SCDDR 加载新数据，如果重传次数没有超过 3 次，发送器自动重发当前数据。否则，SCDISR.RETR_3 设置为 1，重传结束。在采样下一个奇偶校验位前确保 SCDISR.TRE 和 SCDISR.RETR_3 被清零。

2. 如果没有错误信号从读卡器返回，SCDCSR.TRE 不会被设置为 1，SCD 发送器从 SCDDR 加载新数据并从 SIO 管脚上发送。

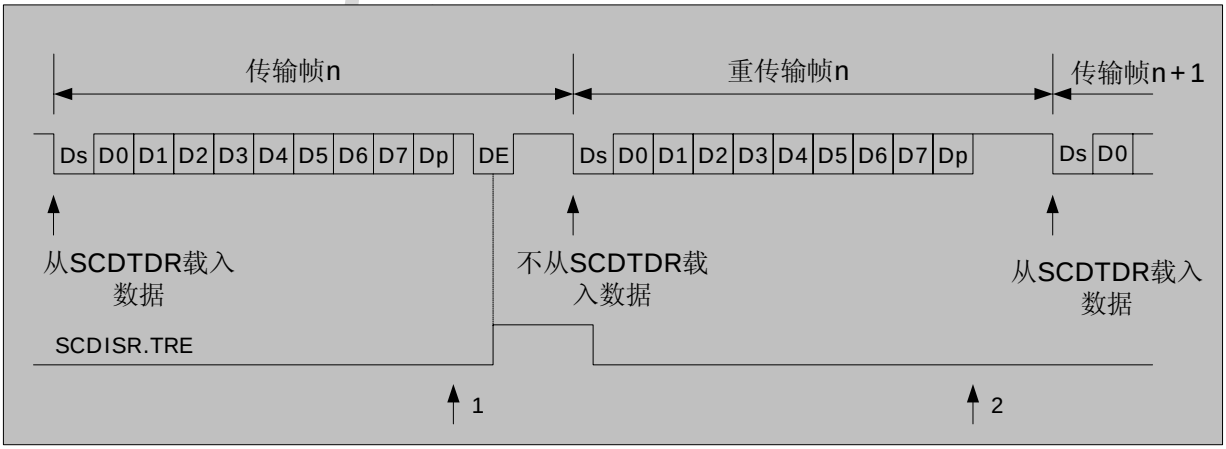


图 23-3 发送模式下数据重传(T = 0)

23.3 SCD寄存器配置

23.3.1 SCD中断状态寄存器

表 23-1 SCD 中断状态寄存器（SCDISR－C0H)

SCDISR		SCD 中断状态寄存器					C0H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ECNTO	FIFO_NE	FIFO_HF	FIFO_FU	FIFO_OV/T2R	TXEND	TRE	RETR_3
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
0	0	0	0	0	0	0	0

ECNTO	ETU TIMER 计数溢出标志： ECNTO =0 则没有溢出 ECNTO =1 则溢出
	此位表示指示用户设定的字符间的延时(字符等待时间)是否到，即，SCD 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 SCDECR 设定的值)。如果 SCDCSR.ETU_EN 为 1，则使能了 ETU 计数器溢出中断。写 0 清除这个标志，写 1 没影响。
FIFO_NE	FIFO 非空标志： FIFO_NE =0 则 FIFO 空 FIFO_NE =1 则 FIFO 非空，软件清除此位
FIFO_HF	FIFO 半满标志： FIFO_HF =0 则 FIFO 非半满 FIFO_HF =1 则 FIFO 半满，软件清除此位
FIFO_FU	FIFO 全满标志： FIFO_FU =0 则 FIFO 非全满 FIFO_FU =1 则 FIFO 全满，软件清除此位
FIFO_OV	Rx-FIFO 接收溢出错误： FIFO_OV =0 没有接收溢出错误发生 FIFO_OV =1 发生了接收溢出错误

	此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)SCDISR.FIFO_OV 被设置为 1，CPU 读此位为 1 后必须从接收 FIFO 读数据并且将 SCDISR.FIFO_OV 清零。 注：1).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。 2).当 SCDISR.FIFO_OV 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。
T2R	SCD 快速发送转接收标志(快速从发送模式切换到接收模式时使用)： T2R =0 表示发送没有完成 T2R =1 表示发送完成，软件查询到此标志后可立即设置当前工作模式为接收模式，软件清 0
	SCDISR.T2R 与 SCDISR.FIFO_OV 复用: 发送模式(SCDCSR.TRS=1): SCDISR[3] = SCDISR.T2R ; 接收模式(SCDCSR.TRS=0): SCDISR[3] = SCDIST.FIFO_OV.
TXEND	SCD 发送完成标志： TXEND =0 表示发送没有完成 TXEND =1 发送完成，硬件设置，软件清 0
	当 SCDCSR.TRS= 1 时，SCDISR.TXEND= 1 表示发送完 1 个字节
TRE	SCD 发送/接收奇偶校验错误标示： TRE =0 则 SCD 发送/接收完成时无奇偶校验错误 TRE =1 则 SCD 发送/接收完成时有奇偶校验错误
	此位表示在发送和接收数据时有一个奇偶校验错误发生。 当 SCDCSR.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，此时硬件设置 SCDISR.TRE = 1。对于 T= 0，有校验错误的接收数据将不会写到 FIFO 上。 当 SCDCSR.TRS = 1, SCDISR.TRE = 1 表明一个奇偶校验错误信号从读卡器反馈来；如果 同时 RETR_3 是 0，对于 T=0，硬件会自动重传输最后一个字节。
RETR_3	硬件自动重传 3 次： RETR_3 =0 则数据重传没有超过 3 次 RETR_3 =1 则有奇偶错的数据重传次数>=3 次

	此位表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,硬件设置这个位。在传送过程中，如果 SCDISR.RE3_EN 设置为 1，当 SCDISR.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 没影响。 当 SCDCSR.TRS = 0 时这个位被忽略(接收模式)

23.3.2 SCD中断允许寄存器

表 23-2 SCD 中断允许寄存器（SCDIER－C1H)

SCDIER		SCD 中断允许寄存器					C1H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Bit 7	ETU TIMER 计数溢出中断，0：禁止； 1：使能						
Bit 6	FIFO 非空中断，0：禁止； 1：使能						
Bit 5	FIFO 半满中断，0：禁止； 1：使能						
Bit 4	FIFO 全满中断，0：禁止； 1：使能						
Bit 3	Rx-FIFO 接收溢出中断/TX-快速发送转接收标志，0：禁止； 1：使能						
Bit 2	SCD 发送完成中断，0：禁止； 1：使能						
Bit 1	SCD 发送/接收奇偶校验错误中断，0：禁止； 1：使能						
Bit 0	硬件自动重传超过 3 次中断，0：禁止； 1：使能						

23.3.3 SCD控制寄存器

表 23-3 SCD 控制寄存器(SCDCSR—C2H)

SCDCSR		SCD 控制寄存器					C2H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	FLUSH	TRS	RE3_EN	ODD_EN	DIS
-	-	-	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

FLUSH (写 1 有效)	接收 FIFO 清除： FLUSH =0 不清空接收 FIFO FLUSH =1 则清空接收 FIFO 设置这个位将在一个 CLK 周期内清空接收/ FIFO，但不会清空接收器。 这个位只能被写‘1’，读出来常为‘0’。
TRS	SCD 发送/接收模式选择： TRS =0 选择接收模式 TRS =1 选择发送模式 指出传输为发送或者接收。
RE3_EN	硬件自动重传 3 次使能： RE3_EN =0 软件重发 RE3_EN =1 硬件重发 发送前设置此位，如出现奇偶校验错误，由硬件自动重发 3 次
ODD_EN	奇偶校验方式选择： ODD_EN =0 偶校验 Even Parity ODD_EN =1 奇校验 Odd Parity
DIS	正反向模式选择 Direct Inverse Mode： 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。还规定 ATR 中 TS 字节是 H’3B 还是 H’3F。 DIS =0 正向模式：发送数据不改变，LSB 先发；接收数据不改变，LSB 先收。TS 字节为 H’3B DIS =1 反向模式：发送数据发送前 Bit 反转，MSB 先发，接收数据 MSB 先收，存储前 Bit 反转。TS 字节为 H’3F

注意：SCDCSR 两次连续写操作之间的时间间隔要求大于 35 个系统时钟。

23.3.4 SCD数据寄存器

表 23-4 SCD 数据寄存器（SCDDR—C3H)

SCDDR		SCD 数据寄存器					C3H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCDDR[7:0]							
R/W							
00h							

SCDDR[7:0]	接收发送复用寄存器。 写时，写的数据放入发送缓冲区； 读时，读到的是数据是接收缓冲区的值。

23.3.5 SCD波特率参数低位寄存器

表 23-5 SCD 波特率参数低位寄存器(SCDBPRL—C4H)

SCDBPRL		SCD 波特率参数低位寄存器					C4H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCDBPRL[7:0]							
R/W							
74h							
SCDBPRL[7:0]	波特率参数寄存器 SCDBPRH/ECR、SCDBPRL 构成 12 位分频器。 例如：系统时钟为 3.57MHz，为获得 9600 波特率， 则 SCDBPR＝3.57×1000000÷9600＝372， 即 SCDBPRH/ECR＝01H，SCDBPRL＝74H。 例如：系统时钟为 3.57MHz，为获得 19200 波特率， 则 SCDBPR＝186，即 SCDBPRH/ECR＝00H，SCDBPRL＝baH。						

23.3.6 SCD波特率参数高位/ETU计数寄存器

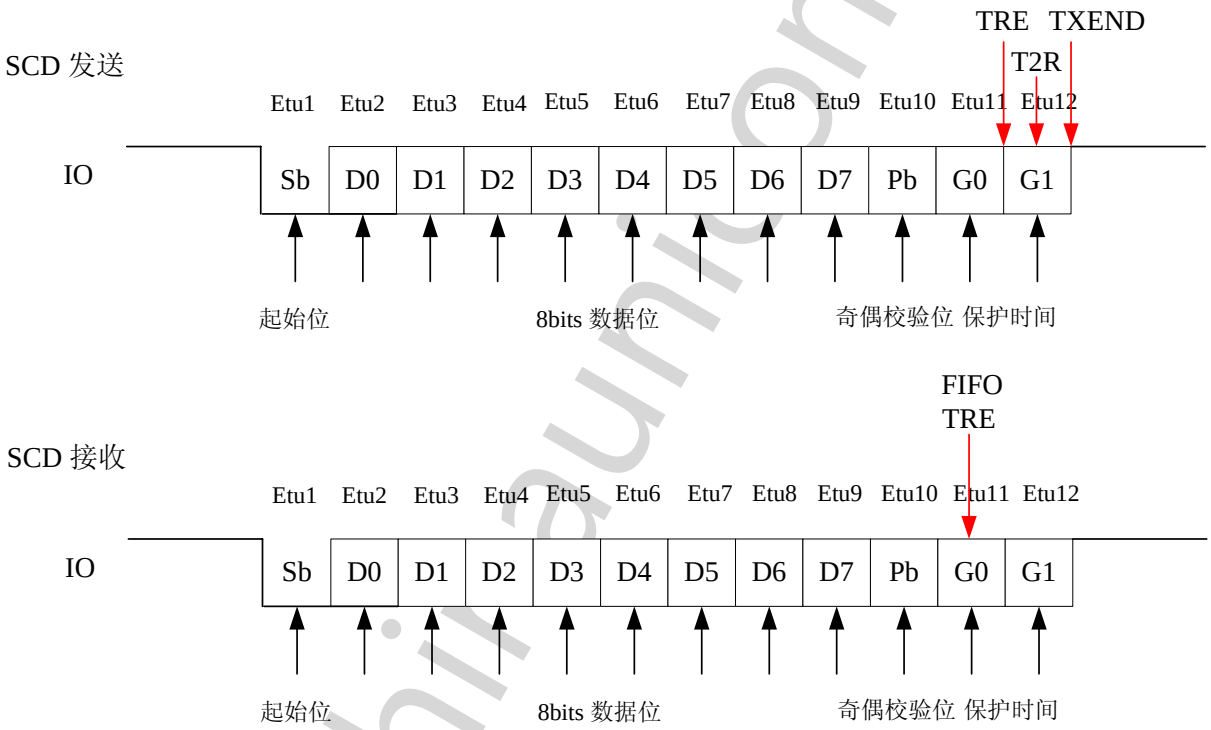
表 23-6 SCD 波特率参数高位/ETU 计数寄存器(SCDBPRH/ECR—C5H)

SCDBPRH/ECR		SCD 波特率参数高位/ETU 计数寄存器					C5H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCDECR [7:4]				SCDBPRH[3:0]			
R/W							
01h							
SCDECR [7:4]	用于设置字符之间间隔时间。系统复位将 SCDECR 初始化为 H’00。如果 SCDECR 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 SCDECR 的值，如果 SCDIER .ECNTO 使能，则产生一个 etu 计数溢出中断(ECI)。						

SCDBPRH[3:0]	波特率参数寄存器 SCDBPRH、SCDBPRL 构成 12 位分频器。 例如：系统时钟为 3.57MHz，为获得 9600 波特率， 则 SCDBPR＝3.57×1000000÷9600＝372， 即 SCDBPRH＝01H，SCDBPRL＝74H。 例如：系统时钟为 3.57MHz，为获得 19200 波特率， 则 SCDBPR＝186，即 SCDBPRH＝00H，SCDBPRL＝baH。

注意：写 SCDBPRH/ECR 将自动清除内部 etu 计数器

23.4 SCD标志位置位时间



23.5 SCD操作

23.5.1 初始化

- 置 SCDIER=0
- 在接收前，清空 Receive-FIFO(SCDCSR. FLUSH =1)
- 清 SCDISR 中的状态标志
- 配置 SCDBPRH/ECR 和 SCDBPRL
- 设置或清零 SCDCSR.TRS 位

- 中断使能（SCDIER 相应位置 1）

23.5.2 发送字节步骤

- 按照 23.5.1初始化中描述的步骤初始化SCD模块(连续发送可只在第一次作初始化)
- 置 SCDCSR.TRS=1
- 写入第 1 个字节到 SCDDR (开始发送第 1 个字节)
- 等待发送第 1 个字节完成标志 SCDISR.TXEND=1（或等待 ScdInt 中断）
- 清 SCDISR.TXEND=0
- 软件确保间隔大于 2 个 ETU (如无速度要求等特殊情况，尽量可在相邻数据发送间，加延时,以支持某些对方设备来不及接收数据的情况，而漏掉或错接.)
- 写入第 2 个字节到 SCDDR（开始发送第 2 个字节）
- 等待发送第 2 个字节完成标志 SCDISR.TXEND= 1（或等待 ScdInt 中断）

……

……

- 等待发送第 n 个字节完成标志 SCDISR.TXEND =1（或等待 ScdInt 中断）
- 清 SCDISR.TXEND=0

（发送结束）

23.5.3 接收字节步骤

- 初始化 SCD 模块，置 SCDCSR.TRS=0(连续接收可只在第一次作初始化)
- 接收错误处理：

读 SCDISR.TRE 和 SCDISR. FIFO_OV 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。

- 状态检测及读接收数据：

等待 FIFO 状态中断 SCDISR.FIFO_NE/ SCDISR.FIFO_HF/ SCDISR.FIFO_FU，或查询 FIFO 状态表示位，然后从 SCDDR 读接收数据

（接收结束）

23.5.4 接收转发送

- 按 23.5.3接收字节步骤接收结束后，需要先延时
- 然后按 23.5.2发送字节步骤发送数据以支持对方刚发完数据,还未处于有效接收状态的情况

23.5.5 发送转接收

- 发送最后一个字节数据
  a. 写入最后 1 个字节数据到 SCDDR (开始发送最后 1 个字节数据)
  b. 等待快速发送转接收标志 SCDISR.T2R=1（或等待 ScdInt 中断）
  c. 清 SCDISR.T2R=0
- 设置 SCD 为接收状态(SCDCSR.TRS=0),

按 23.5.3接收字节步骤接收数据(如有过多延时而不能马上进入接收状态，则可能错过接收对方刚发的数

23.6 编程要求

1. 连续对 SCDCSR 写操作之间时间间隔要求大于 35 个核时钟。

第24章 SCC 接口

24.1 概述

智能卡控制器（以下简称 SCC）支持与 ISO/IEC 7816-3 标准 T = 0 和 T = 1 传输协议兼容的智能卡设备。在芯片中，SCC 模块在 CPU 的控制下实现与智能卡的串行通讯。

24.2 功能特性

1. 符合 ISO / 7816-3 标准
2. 支持 T = 0 协议，即异步半双工字符传输协议

a) 数据长度 8 bit

b) 奇偶校验位自动生成及奇偶校验错误检测

c) 自动错误检测并报告

d) 支持正向约定（先传 LSB）和反向约定(先传 MSB)

e) 传输完成后自动接收模式转发送模式

3. 支持 T = 1 协议，即异步半双工块传输协议（软件层实现）
4. 8 比特 2 级固定深度接收 FIFO
5. 系统时钟输入为：5MHz/10MHz/20MHz
6. 输出给卡时钟，系统时钟的 1/2、1/4、1/8 分频
7. 停卡时钟功能
8. 波特率范围：2400bps~115200bps
9. 支持帧传输保护时间可配， 0 到 15 个 ETU
10. 支持数据传输处理中断
11. 支持奇偶校验错误中断
12. 数据帧格式错误中断
13. FIFO 溢出中断

24.3 数据传输

24.3.1 数据格式

一帧至少有 12 个ETU周期长(ETU即基本时间单位：这是传送 1bit数据所需的时间)，发送器发出一个起始位，紧接着是一个字节数据（8bits）加上 1bit奇偶校验位，当数据中包含奇数个 1 时，该位为 1。在发送模式中，发送完 8 位数据后，由硬件自动计算奇偶位；在奇偶位发完之后，I/O线返回高阻态，保持至少两个ETU。如 图 24-1所示。

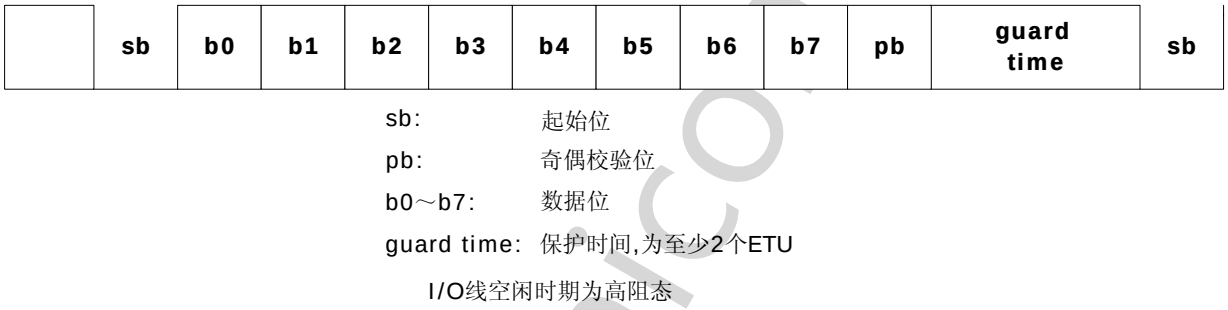


图 24-1 ISO-7816 发送/接收数据格式

24.3.2 接收模式

通过把 SCCUCR 中的 T/R 位清 0 来选择接收模式，或者在传输最后一个数据前把 SCCUCR 的 LCT 位置 1，使其发送完成自动转入接收模式。I/O 线会在每位的中心位置采样；当检测到起始位后，紧接的数据字节和奇偶位被移入内部移位寄存器中，通过比较奇偶位的正确与否，来确定是否需要对方重传，数据采样在位于起始位后 1.5 个 ETU 处开始。

当 SCC 处于接收模式，检测到有效起始位后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写入 FIFO。当发现奇偶校验错时，接收器不会将接收字节写入 FIFO，同时在 10.5etu 到 11.5etu 间将 SIO 管脚驱动为低电平等待错误字符的重发。并且在 10.5etu 时置位 SCCISR 的 PE 位，产生奇偶校验错误中断。

24.3.3 发送模式

通过设置 SCCUCR 中的 T/R 位来选择发送模式，数据被装入 SCCTDR 寄存器中，

在 I/O 线上输出一个起始位，在起始位之后，数据和奇偶校验位被移出，IO 线在起始位之后在 10ETU 处返回高阻态。

在起始位之后 11ETU 处发生中断，IO 线被检测来发现可能的错误信号，SCCISR 中的状态位也被更新。软件检查错误标志位决定是否重传。

下一字节的发送，必须经过 2 个 ETU 之后，数据被装入 SCCTDR 后，就会开始发送；发送器先发送起始位，然后一个字节的数据位，最后是奇偶校验位。这些数据通过 SIO 管脚发送数据，以起始位开始，奇偶校验位结束。

24.4 寄存器配置

SCC 模块共有 7 个寄存器。下面分别给出了各寄存器的数据结构和功能定义的说明。

24.4.1 SCC中断状态寄存器

表 24-1 SCC 中断状态寄存器(SCCSR－D8H)

SCCSR		SCC 中断状态寄存器					D8H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	PE	OVR	FER	TBE/RBF	FNE
-	-	-	R/W	R/W	R/W	R/W	R
-	-	-	0	0	0	0	0

Rev	保留位，软件不操作。
PE	SCC 发送/接收时奇偶校验错误标志位，接收模式下将在 10.5 个 ETU 时置 1，发送模式下将在 11.5 个 ETU 时置 1。 在 T=0 时，收到发生奇偶校验错误的数据，不会写到 FIFO 中。 在 T=1 时，收到发生奇偶校验错误的数据，会写到 FIFO 中。 CPU 对此位写 1 清 0
OVR	接收 FIFO 溢出错误：OVR 为 1，SCC 接收到新字符时 FIFO 已满，此时，至少会造成一个字符丢失。 CPU 读此位为 1 后，必须读取接收 FIFO 的数据。 CPU 对此位写 1 清 0
FER	帧错误中断：Frame Error，若 IO 在 Start bit 后的 10.25 个 ETU 之后没有处于高阻态，该位置 1。 CPU 对此位写 1 清 0

TBE/RBF （read only）	TBE 为高表示处于发送状态的 SCC 缓冲区已空，此时控制器可以继续向 SCCTDR 写入下一字符。当写入下一字符到 SCCTDR 或者 SCCUCR 中 T/R 自动或者人工清 0，TBE 将置 1。 RBF 为高表示处于接收状态的 SCC 接收 FIFO 已满，此时 CPU 可以通过读取 SCCTDR 使得该位清 0。 TBE/RBF 共用一位，当 SCDM 处于发送状态时，该位表示 TBE，当其处于接收状态时，该位表示 RBF。
FNE （read Only）	接收 FIFO 非空时，该位置 1。 当接收 FIFO 空时，该位自动清 0；
向 TDR 写数据或者从 RDR 读数据后，或者 SCC 从发送模式切换到接收模式并且接收 FIFO 未满时，TBE/RBF 将自动清 0。 若 LCT 用于发送最末字符时，TBE 在发送结束时不会清 0。	

24.4.2 SCC中断使能寄存器

表 24-2 SCC 中断使能寄存器(SCCIER－D9H)

SCCIER		SCC 中断使能寄存器					D9H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	PEEN	OVREN	FEREN	TBE/RBFEN	FNEEN
-	-	-	R/W	R/W	R/W	R/W	R/W
-	-	-	0	0	0	0	0
PEEN		PEEN=1，发送/接收奇偶检验错中断使能					
OVREN		OVREN=1，接收 FIFO 溢出中断使能					
FEREN		FEREN=1，帧错误中断使能					
TBE/RBFEN		TBE/RBFEN=1，发送完成和接收 FIFO 满中断使能					
FNEEN		FNEEN=1，接收 FIFO 中断非空使能					

24.4.3 SCC传输控制寄存器

表 24-3 SCC 传输控制寄存器(SCCTCR－DAH)

SCCTCR		SCC 传输控制寄存器					0DAH
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

Rev	FIP	AUTOCONV	PROT	T/R	LCT	SS	CONV
RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0
CONV	CONV 表示发送 MSB/LSB 的顺序约定。 CONV=1 表示顺序发送。 AUTOCONV=1 时，CONV 由软件设定。 AUTOCONV=0 时，CONV 由硬件自动检测。						
SS	SS 位在 ATR 之前置 1 用于自动检测 CONV， 在接收到初始字符后 10.5 ETU 处自动清 0。						
LCT	最后发送字符（last character for trans）. 在写入最后要发送的字符到 SCCTDR 前先将此位置 1。 SCC 发送完后将自动切换到接收模式。 成功发送完(（11+3/4）etu in T=0, (10+3/4)etu in T=1）硬件自动将此位清 0。 LCT=0 后，T/R 位也将自动清 0，SCC 进入接收模式。						
T/R	T/R 由软件置位使 SCC 处于发送模式。 T/R 由 0 变为 1 将使 USR 中的 TBE 置位。 若发送最后字符前 LCT 有效则发送完成后 T/R 由硬件自动清 0。						
PROT	PROT=1 表示 T=1； PROT=0 表示 T=0。						
AUTOCONV	AUTOCONV 为低有效。 AUTOCONV=1 时，发送顺序协定由软件配置 UCR1 的 CONV 位决定； AUTOCONV=0 时，发送顺序协定将根据（SS 位有效时）自动检测接收的第一个字符来决定。 该位在一个卡 session 中不可更改。						
FIP	奇偶校验控制位。 该位为 1 时，为奇校验。 该位为 0 是，为偶校验。						
Rev	保留位，软件不操作。						

24.4.4 SCC用户控制寄存器

表 24-4 SCC 用户控制寄存器(SCCUCR－DBH)

SCCUCR		SCC 用户控制寄存器					DBH
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

Rev	START	Rev	RSTIN	RIU	CST	AC1	AC0
	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0
AC[1:0]	系统时钟分频因子，分频后的时钟频率就是输出给卡的时钟频率 00 Clk 01 Clk/2 10 Clk/4 11 Clk/8						
CST	Clock stop. (停止给卡的时钟 Clk_AC) CST 有效时，CLK 停止； CST 复位时，CLK 由 AC[2:0]决定。						
RIU	低电平有效，SCC 局部复位（同步复位）。 该位清 0 时会将 SCC 内的寄存器恢复初始值。 卡激活前必须先复位。						
RSTIN	当卡被激活时，输出给卡的 RST 信号由该位决定。						
Rev	保留位，软件不操作。						
START	START=1，卡将被激活； START=0，卡停止激活。						

24.4.5 SCC传输数据寄存器

表 24-5 SCC 传输数据寄存器（SCCTDR－DCH）

SCCTDR		SCC 传输数据寄存器					DCH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCCTDR							
W							
00H							
SCCTDR	发送数据寄存器						

24.4.6 SCC接收数据寄存器

表 24-6 SCC 接收数据寄存器（SCCRDR－DCH）

SCCRDR		SCC 接收数据寄存器					DCH
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

R	
00H	
SCCRDR	接收数据寄存器

24.4.7 SCC波特率参数低位寄存器

表 24-6 SCC 波特率参数低位寄存器（SCCBPRL－DDH）

SCCBPRL		**SCC 波特率参数低位寄存器**					**DDH**
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BPRL[7:0]							
RW							
00H							
SCCBPRL	BPR 用于对卡时钟计数生成 ETU						
	波特率参数寄存器 SCCBPRH、SCCBPRL 构成 12 位分频器。 例如：系统时钟为 20MHz，输出给卡时钟 5M，为获得 9600 波特率， 则 SCDBPR＝5×1000000÷9600＝521=0x209， 即 SCDBPRH＝02H，SCDBPRL＝09H。 例如：输出给卡时钟为 5MHz，为获得 19200 波特率， 则 SCDBPR＝5×1000000÷19200＝260=0x104， 即 SCDBPRH＝01H，SCDBPRL＝04H。					

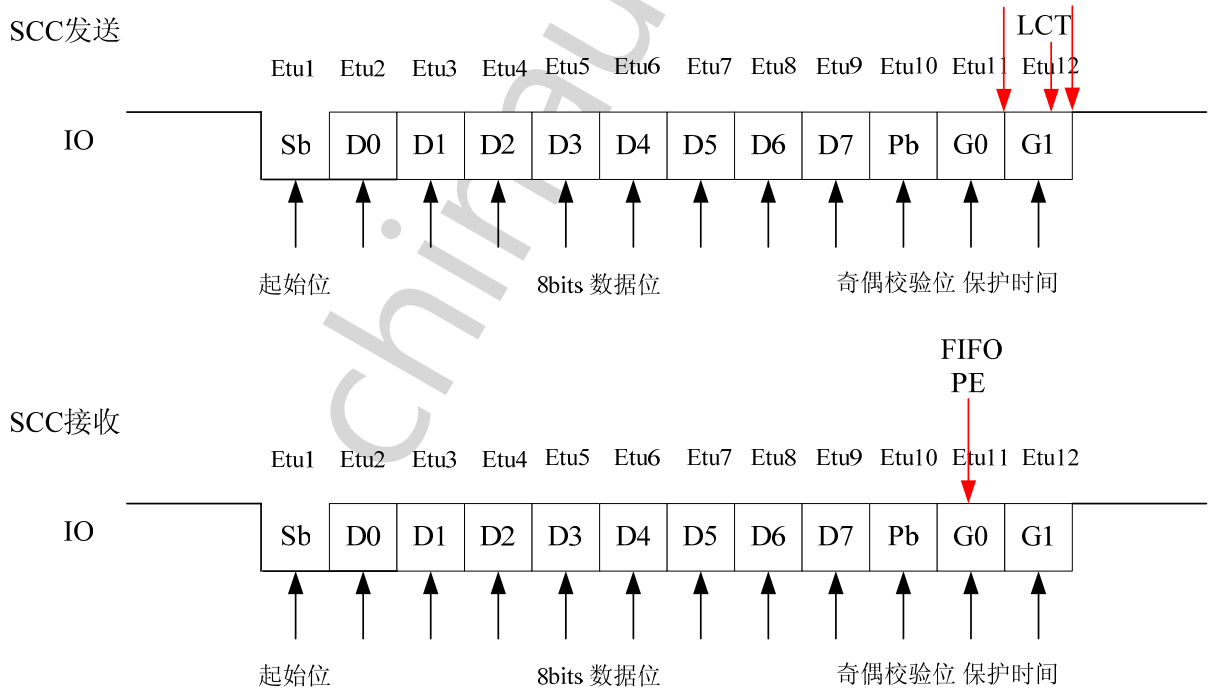
24.4.8 SCC波特率参数高位寄存器

表 24-7 SCC 波特率参数高位寄存器（SCCBPRH－DEH）

SCCBPRH		**SCC 波特率参数高位寄存器**					**DEH**
BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT9	BIT8
GT 3	**GT 2**	**GT 1**	**GT0**	**BPRH[11:8]**			
RW				RW			
0	0	0	0	0000			

GT	Guard Time，SCC 保护时间参数(1~E) 发送模式下，SCC 等待足够的 ETU 后再发送 SCCTDR 中的数据。 T = 1 时， GTR=F 表示在延时 11 个 ETU 后操作； T = 0 时， GTR=F 表示在延时 12 个 ETU 后操作。
BPRH	BPRH 用于对卡时钟计数生成 ETU
	波特率参数寄存器 SCCBPRH、SCCBPRL 构成 12 位分频器。 例如：系统时钟为 20MHz，输出给卡时钟 5M，为获得 9600 波特率， 则 SCDBPR＝5×1000000÷9600＝521=0x209， 即 SCDBPRH＝02H，SCDBPRL＝09H。 例如：输出给卡时钟为 5MHz，为获得 19200 波特率， 则 SCDBPR＝5×1000000÷19200＝260=0x104， 即 SCDBPRH＝01H，SCDBPRL＝04H。

24.5 SCC标志位置位时间



24.6 SCC操作

A. 初始化

- SCCUCR.RIU 清 0，保持 2 个 Clk 以上低电平然后置 1。目的是清 SFR 寄存器的标志位；
- 如果设置 SCCIER.AUTOCONV=1，则软件决定发送顺序，即设置 SCCUCR.CONV=1 则正向传输，SCCUCR.CONV=0 是反向传输；否则 ATR 的应答字 3b 或 3f 决定 SCCUCR.CONV 的值；
- 配置 SCCIER，中断使能寄存器，是否产生相应的中断脉冲。
- 配置 ATR 阶段的波特率计数器，设置 SCCPDRL[7:0]为 0x74，SCCPDRH[3:0]为 0x1；
- 配置 SCCUCR[1:0]，决定输出给卡的时钟频率；
- 配置 SCCBPRH[7:4]，发送数据字节的保护时间， 0~15 个 ETU 可配；
- 配置输出给卡的时钟使能，SCCUCR.CST=0，输出给卡的时钟有效。

B. 卡 Activation 过程

- SCCUCR.START 置 1 则开始激活卡，硬件自动使 Clk_Card 输出稳定，IO 拉高，START 置 1 后大约 100 个 clk，卡被激活。
- 设置 SCCTCR.SS=1，再设置 SCCUCR.RSTIN=1，等待卡的 ATR，在 400~40000 个 Clk 之内收到 3b 或 3f 数据，表示 ATR 正确。

C. 卡 Deactivation 过程

- SCCUCR.START 置 0 则开始 Deactive 卡，软件设置 SCCBPRH.RSTIN=0，硬件自动使 Clk_Card 输出为 0，然后把 IO 拉低。

D. 发送字节

- 发送数据前软件可以配置参数：波特率、分频系数、发送数据的间隔时间 GT；
- 置 SCCTCR.T/R=1；
- 写入第一个字节到 SCCTDR；
- 等待发送完成标志 SCCSR.TBE =1(或等待 ScrIntn 中断)；
- 如果发送出错：产生中断或者查询 SCCISR 寄存器标志，判断错误类型，执行相应的错误处理，处理完之后 CPU 对该位写 1，则该标志位清 0；
- 可以继续写入一个字节到 SCCTDR，此时 SCCSR.TBE 自动清 0，或者转接收模式（即 SCCTCR.T/R 自动或者手动清 0）后 SCCSR.TBE 自动清 0。

E. 接收字节

- 接收数据前软件可以配置参数：波特率、分频系数、发送数据的间隔时间 GT；连续接收配置一次即可；
- 置 SCCTCR.T/R=0；
- 接收数据，查询 SCCBPRH.FE!=0（即接收数据 FIFO 非空），则读 SCCRDR 数据；
- 接收错误处理：产生中断或者查询 SCCSR 寄存器标志位，判断错误类型，执行相应的错误处理，处理完之后 CPU 对该位写 1，则该标志位清 0；
- 继续接收数据。

第25章 UART接口

25.1 概述

通用异步串行接口（以下简称 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，或把从外设的 UART 串行接收来的数据转换为并行数据提供给处理器。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。

25.1.1 功能特性

- 提供标准的异步通讯位（起始位、奇偶校验位和停止位）
  - 生成 1 位起始位
  - 生成 1 位校验位（可设置奇校验或偶校验，或无校验位）
  - 生成 1 位停止位
- 8Bit 4 级的接收 FIFO
- 工作模式或闭环测试模式
- 可编程波特率(波特率可以根据参数 F/D 调整)
- 支持数据通讯及错误处理中断
- 具有起始位有效性检测功能
- 全双工通讯

25.2 基本原理

25.2.1 传送格式

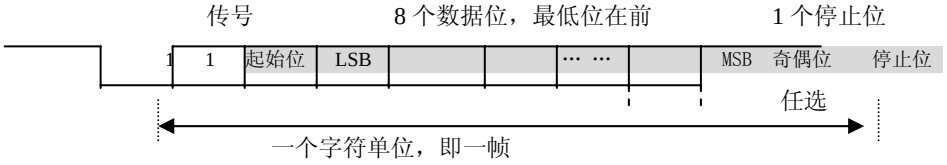


图 25-1 UART 发送/接收数据格式

一帧数据除表示字符信息的数据位（位长度 8bit）可选外还有若干附加位：起始位（1 位、值恒为 0）、奇偶校验位（可选择奇校验或偶校验）、停止位（长度为 1 位，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）。

异步通信的一帧经历以下步骤：

1）无传输　　发送方连续发送传号“1”，表明双方无数据传输。

2）起始传输　发送方在任何时候将传号变成空号（即由“1”变“0”），并持续 1 个数据位时间，表明发送方开始发送数据。与此同时接收方检测到空号后开始与发送方同步，并等待收到随后的数据。

3）数据传输　在起始空号之后连续发送或接收的数据位串称为数据传输。一帧数据位长度 8 位。一旦确定并在修改前，应确保每次发送时数据位数不变。数据传输规定最低位在前，最高位在后。

4）奇偶传输　数据传输之后可以选择奇偶校验位发送或接收。奇偶校验位的状态取决于选择的奇偶校验类型。一旦选择确定并修改前，应确保每个字符所选择的奇偶校验位的有无和校验类型必须一致。

5）停止传输　奇偶校验位（选择有奇偶校验）或数据位(选择无奇偶校验)之后发送或接收的停止位，其状态恒为传号（信息“1”）。停止位的长度 1。一旦选择确定并修改前，应确保每个字符所发送的停止位的长度相同。

发送方发送一帧字符后，可以有两种选择发送下一字符，即连续发送或随机发送。在连续发送的时候，下一帧的启始位接到上一帧的帧停止位，就这样一帧紧接着一帧连续的发送出去。所谓随机发送是指一帧发送完后停止一个随机的时间长度才发送下一帧的启始位。

25.2.2　接收模式

通过清除 UARTCR 中的 TRS 位来选择接收模式，I/O 线被在每位的中心位置采样；当检测到起始位后，紧接的数据字节和奇偶校验位被移入内部移位寄存器中，通过比较奇偶校验位的正确与否，来确定是否需要对方重传，数据采样在位于起始位后 1.5 个 ETU 处开始；

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 UARTRDR(接收 FIFO)。当发现奇偶校验错时，设置 UARTISR.TRE 位。接收器停止将接收字节写入接收 FIFO。

UARTISR 中的 FIFO 标志位必须被清除用来返回“等待状态”并容许下一个数据的接收。

25.2.3 发送模式

首先通过设置 UARTCR 中的 TRS 位来选择发送模式，数据被装入 UARTTDR 寄存器中，在 I/O 线上输出一个起始位，在起始位之后，数据和奇偶校验位被移出，IO 线在起始位之后在 10ETU 处返回高阻态。

当发送完成后，UARTISR 中的状态位也被更新。软件检查错误标志位决定是否重传。

下一字节的发送，需先清除 UARTISR 中的 TXEND 标志位，装入数据后，就会开始发送，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，UART 开始将 UARTTDR 的数据传送发送器，然后通过 TXD 管脚发送数据，以起始位开始，奇偶校验位结束。

25.3 寄存器配置

UART 模块提供 8 个 8 位 SFR 寄存器来实现与 CPU 接口，UART 模块与 SCD 模块寄存器为同一地址，UART 模块与 SCD 模式不能共存。

25.3.1 UART中断状态寄存器

表 25-1 UART 中断状态寄存器（UARTISR－C0H）

UARTISR		UART 中断状态寄存器					C0H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ECNTO	FIFO_NE	FIFO_HF	FIFO_FU	ORER	TXEND	TRE	STPB_ERR
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
0	0	0	0	0	0	0	0

ECNTO	ETU TIMER 计数溢出标志： ECNTO =0 则没有溢出 ECNTO =1 则溢出
	此位表示指示用户设定的字符间的延时(字符等待时间)是否到，即，UART 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 UARTECR 设定的值)。如果 UARTCR.ETU_EN 为 1，则使能了 ETU 计数器溢出中断。写 0 清除这个标志，写 1 没影响。
FIFO_NE	FIFO 非空标志： FIFO_NE =0 则 FIFO 空 FIFO_NE =1 则 FIFO 非空，软件清除此位
FIFO_HF	FIFO 半满标志： FIFO_HF =0 则 FIFO 非半满 FIFO_HF =1 则 FIFO 半满，软件清除此位
FIFO_FU	FIFO 全满标志： FIFO_FU =0 则 FIFO 非全满 FIFO_FU =1 则 FIFO 全满，软件清除此位
ORER	Rx-FIFO 接收溢出错误： ORER =0 没有接收溢出错误发生 ORER =1 发生了接收溢出错误 此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) UARTISR.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 UARTISR.ORER 清零。 注： 1).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。 2).当 UARTISR.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。
TXEND	UART 发送完成标志： TXEND =0 表示发送没有完成 TXEND =1 发送完成，硬件设置，软件清 0 当 UARTCR.TRS= 1 时，UARTISR.TXEND= 1 表示发送完 1 个字节

TRE	UART 发送/接收奇偶校验错误标示： TRE =0 则 UART 发送/接收完成时无奇偶校验错误 TRE =1 则 UART 发送/接收完成时有奇偶校验错误
	此位表示在发送和接收数据时有一个奇偶校验错误发生。 当 UARTCR.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，此时硬件设置 UARTISR.TRE = 1。
STPB_ERR	接收到错误的停止位标志(停止位为低电平)： STOPBIT_ERR =0 则当前帧接收到正确的停止位 STOPBIT_ERR =1 则当前帧接收到错误的停止位

25.3.2 UART中断允许寄存器

表 25-2 UART 中断允许寄存器（UARTIER－C1H）

UARTIER		UART 中断允许寄存器					C1H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
-	-	-	0	0	0	0	0
Bit 7		ETU TIMER 计数溢出中断， 0：禁止； 1：使能					
Bit 6		FIFO 非空中断，0：禁止； 1：使能					
Bit 5		FIFO 半满中断，0：禁止； 1：使能					
Bit 4		FIFO 全满中断，0：禁止； 1：使能					
Bit 3		Rx-FIFO 接收溢出中断，0：禁止； 1：使能					
Bit 2		UART 发送完成中断，0：禁止； 1：使能					
Bit 1		UART 发送/接收奇偶校验错误中断，0：禁止； 1：使能					
Bit 0		接收到错误的停止位中断，0：禁止； 1：使能					

25.3.3 UART状态与控制寄存器

表 25-3 UART 控制寄存器(UARTCS—C2H)

UARTCS		UART 控制寄存器					C2H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Rev	UART_LB	UART_PD	FLUSH	TRS	Rev	ODD_EN	Rev
-	R/W	R/W	R/W	R/W	-	R/W	-
0	0	0	0	0	0	0	0

UART_LB	UART 闭环测试模式选择： UART_LB =0 ，UART 正常工作模式 UART_LB =1，UART 闭环测试模式 当 UART_LB =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
UART_PD	UART 有无奇偶校验选择： UART_PD =0， UART 有奇偶校验 UART_PD =1， UART 无奇偶校验 UART_PD =1，发送和接收时不产生或接收奇偶校验位。
FLUSH (写 1 有效)	接收 FIFO 清除： FLUSH =0 不清空接收 FIFO FLUSH =1 清空接收 FIFO 设置这个位将在一个 CLK 周期内清空接收/ FIFO，但不会清空接收器。这个位只能被写‘1’，读出来常为‘0’。
TRS	UART 发送/接收模式选择： TRS =0 选择接收模式 TRS =1 选择发送模式 指出传输为发送或者接收。
ODD_EN	奇偶校验方式选择： ODD_EN =0 偶校验 Even Parity ODD_EN =1 奇校验 Odd Parity

25.3.4 UART数据寄存器

表 25-4 UART 数据寄存器（UARTDR—C3H）

UARTDR		UART 数据寄存器					C3H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UARTDR [7:0]							
WR							
00h							

UARTDR [7:0]	接收发送复用寄存器。 写时，写的数据放入发送缓冲区； 读时，读到的是数据是接收缓冲区的值。

25.3.5 UART波特率参数低位寄存器

表 25-5 UART 波特率参数低位寄存器(UARTBPRL—C4H)

UARTBPRL		UART 波特率参数低位寄存器					C4H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UARTBPRL[7:0]							
R/W							
74h							
UARTBPRL [7:0]	波特率参数寄存器 UARTBPRH、UARTBPRL 构成 **12** 位分频器。						

25.3.6 UART波特率参数高位/ETU计数寄存器

表 25-6 UART 波特率参数高位寄存器(UARTBPRH—C5H)

UARTBRPH/ECR		UART 波特率参数高位/ETU 计数寄存器					C5H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UARTECR [7:4]				UARTBPRH[3:0]			
R/W							
01h							
UART ECR [7:4]	用于设置字符之间间隔时间。系统复位将 UARTECR 初始化为 H'00。 如果 UARTECR 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 UARTECR 的值，如果 UARTIER .ECNTO 使能，则产生一个 etu 计数溢出中断(ECI)。						
UART BPRH [3:0]	波特率参数寄存器 UARTBPRH、UARTBPRL 构成 12 位分频器。 分频器的时钟源来自 CPU 核时钟。						

25.4 软件操作流程

25.4.1 初始化

1) 置 UARTIER=0
2) 在接收前，清空 Receive-FIFO(UARTCR. FLUSH =1)
3) 清 UARTISR 中的状态标志
4) 配置 UARTBPRH 和 UARTBPRL
5) 设置或清零 UARTCR.TRS 位
6) 中断使能（UARTIER 相应位置 1）

25.4.2 发送字节步骤

1) 按照 25.4.1初始化中描述的步骤初始化
2) 置 UARTCR.TRS=1
3) 写入第 1 个字节到 UARTTDR (开始发送第 1 个字节)
4) 等待发送第 1 个字节完成标志 UARTISR.TXEND=1（或等待 UARTInt 中断）
5) 清 UARTISR.TXEND=0
6) 写入第 2 个字节到 UARTTDR（开始发送第 2 个字节）
7) 等待发送第 2 个字节完成标志 UARTISR.TXEND= 1（或等待 UARTInt 中断）
8) ……
9) ……
10) 等待发送第 n 个字节完成标志 UARTISR.TXEND =1（或等待 UARTInt 中断）
11) 清 UARTISR.TXEND=0

（发送结束）

25.4.3 接收字节步骤

1) 初始化：置 UARTCR.TRS=0
2) 接收错误处理：

读 UARTISR.TRE 和 UARTISR. 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。

3) 状态检测及读接收数据：

FIFO 状态中断 UARTISR.FIFO_NE/ UARTISR.FIFO_HF/ UARTISR.FIFO_FU,然后从 UARTRDR 读接收数据

（接收结束）

第26章 USB接口

26.1 概述

USB 设备控制器(USB Device Controller - UDC)提供一个完全兼容 USB1.1 协议的设备接口。

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

26.1.1 特性

UDC具有如下的一些特性:

- 兼容 USB1.1 协议
- 支持全速/低速两种速度模式
- 硬件自动处理 USB Specification 中 Chapter9 的部分标准请求
- 支持悬挂/恢复以及远端唤醒功能
- 支持 5 个物理端点(一个默认控制端点、2 个中断端点和 2 个 BULK 端点)
- 支持控制传输，批量传输和中断传输

26.2 寄存器配置

26.2.1 USB设备配置寄存器

表 26-1 USB 设备配置寄存器（DEVCFG—BFH）

DEVCFG		**USB 设备配置寄存器**					**BFH**
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	URMOD	EP4VLD	EP3VLD	EP2VLD	EP1VLD	RSM	DEVMOD
R	R	R/W	R/W	R/W	R/W	-	R
0	0	0	0	0	0	0	0
URMOD	USB RAM 访问控制模式选择位， 1：可直接访问 USB RAM(256 字节，映射于 XRAM 空间 0x1000-0x10ff)； 0：不可直接访问 USB RAM。						

EP4VLD	端点 4 配置有效控制位，1：端点 4 可以接收数据；0：端点 4 不可用。
EP3VLD	端点 3 配置有效控制位，1：端点 3 可以发送数据；0：端点 3 不可用。
EP2VLD	端点 2 配置有效控制位，1：端点 2 可以接收数据；0：端点 2 不可用。
EP1VLD	端点 1 配置有效控制位，1：端点 1 可以发送数据；0：端点 1 不可用。
RSM	USB 接口远端唤醒 Resume 控制信号，由 CPU 置 1，在 Epc 返回 USB RSM 中断后硬件清零/CPU 清零。
DEVMOD	USB 接口速度模式位，1：全速；0：低速。由 IOM 根据模式选择信号硬件设置。

26.2.2 USB端点控制状态寄存器

表 26-2 USB 端点控制状态寄存器（EPCSR—A3H）

EPCSR		USB 端点控制状态寄存器					A3H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	EP4RDY	EP3RDY	EP2RDY	EP1RDY	EP0ORDY	EP0IRDY	EP0SRDY
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	1	0	1	1	1	0
EP4RDY	端点 4 OUT 传输当前缓冲区 FIFO Ready 指示标识 1：当前缓冲区 FIFO 数据已经接收成功，CPU 可以读取； 0：当前缓冲区 FIFO 正在等待接收数据，CPU 无法进行读操作。 该位只能由 CPU 在读取完毕 FIFO 数据后写 1 清零，表示当前数据已经读取，该 FIFO 可以重新等待接收新的数据，CPU 写 0 会被屏蔽；当成功接收完成 HOST 下传的一包数据并返回 ACK 之后，硬件将此位置 1。 注意在 ONEBUF=0 时，在清零之后此位有可能还会保持为 1，它实际是内部另一块 OUT FIFO 的 RDY 指示位。						
EP3RDY	端点 3 IN 传输缓冲区 FIFO Ready 指示标识 1：当前缓冲区 FIFO 可被 CPU 写入数据； 0：当前缓冲区 FIFO 禁止 CPU 写操作。 该位只能由 CPU 在写入数据到 FIFO 完成后写 1 清零，表示当前缓冲区 FIFO 数据已写入，可以上传给 USB HOST CPU 写 0 会被屏蔽；当 HOST 成功接收到该数据包返回 ACK 之后，硬件将此位置 1。 注意在 ONEBUF=0 时，在清零之后此位有可能还会保持为 1，它实际是内部另一块 IN FIFO 的 RDY 指示位。						

EP2RDY	端点 2 OUT 传输缓冲区 FIFO Ready 指示标识 1：当前缓冲区 FIFO 数据已经接收成功，CPU 可以读取； 0：当前缓冲区 FIFO 正在等待接收数据，CPU 无法进行读操作。 该位只能由 CPU 在读取完毕 FIFO 数据后写 1 清零，表示当前数据已经读取，该 FIFO 可以重新等待接收新的数据，CPU 写 0 会被屏蔽；当成功接收完成 HOST 下传的一包数据并返回 ACK 之后，硬件将此位置 1。
EP1RDY	端点 1 IN 传输缓冲区 FIFO Ready 指示标识 1：当前缓冲区 FIFO 可被 CPU 写入数据； 0：当前缓冲区 FIFO 禁止 CPU 写操作。 该位只能由 CPU 在写入数据到 FIFO 完成后写 1 清零，表示当前缓冲区 FIFO 数据已写入，可以上传给 USB HOST CPU 写 0 会被屏蔽；当 HOST 成功接收到该数据包返回 ACK 之后，硬件将此位置 1。
EP0O RDY	端点 0 OUT 传输当前缓冲区 FIFO Ready 指示标识 1：当前缓冲区 FIFO 数据已经接收成功，CPU 可以读取； 0：当前缓冲区 FIFO 正在等待接收数据，CPU 无法进行读操作。 该位只能由 CPU 在读取完毕 FIFO 数据后写 1 清零，表示当前数据已经读取，该 FIFO 可以重新等待接收新的数据，CPU 写 0 会被屏蔽；当成功接收完成 HOST 下传的一包数据并返回 ACK 之后，硬件将此位置 1。 注意在 ONEBUF=0 时，在清零之后此位有可能还会保持为 1，它实际是内部另一块 OUT FIFO 的 RDY 指示位。
EP0I RDY	端点 0 IN 传输缓冲区 FIFO Ready 指示标识 1：当前缓冲区 FIFO 可被 CPU 写入数据； 0：当前缓冲区 FIFO 禁止 CPU 写操作。 该位只能由 CPU 在写入数据到 FIFO 完成后写 1 清零，表示当前缓冲区 FIFO 数据已写入，可以上传给 USB HOST CPU 写 0 会被屏蔽；当 HOST 成功接收到该数据包返回 ACK 之后，硬件将此位置 1。 注意，在 ONEBUF=0 时: 1. 在清零之后此位有可能还会保持为 1，它实际是内部另一块 IN FIFO 的 IRDY 指示位。 2. 在低速模式，在软件两次清零操作之间，必须保证有大于 3 个 UDC 时钟(2us)的间隔(比如软件写 8 字节的 IN 包后，又发零包的情况下)。

EP0S RDY	端点 0 Setup 缓冲区 FIFO Ready 指示标识 1：当前缓冲区 FIFO 数据已经接收成功，CPU 可以读取； 0：当前缓冲区 FIFO 正在等待接收数据，CPU 无法进行读操作。 该位只能由 CPU 在读取完毕 FIFO 数据后写 1 清零，表示当前数据已经读取，该 FIFO 可以重新等待接收新的数据，CPU 写 0 会被屏蔽；当成功接收完成 HOST 下传的一包数据并返回 ACK 之后，硬件将此位置 1。 注意在 ONEBUF=0 时，在清零之后此位有可能还会保持为 1，它实际是内部另一块 SETUP FIFO 的 SRDY 指示位。

26.2.3 端点控制状态寄存器

表 26-3 USB 端点 0 控制状态寄存器（EP0CSR—A4H）

EP0CSR		端点 0 控制状态寄存器					A4H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ONE BUF	Rev	Rev	Rev	Rev	Rev	CLR	STALL
R/W	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0

ONE BUF	1：端点 0 IN/OUT 分别只使用 1 个缓冲区传送数据模式。 0：端点 0 IN/OUT 分别用多个缓冲区传送数据模式。
CLR	初始化清零控制位，写 1 清零端点 0 IN/OUT FIFO 指针，硬件自动将此位归零。
STALL	端点 0 Stall 设置信号，接收到 HOST 的 Clear Feature 命令或者复位后被清除。

表 26-4 端点 0 状态寄存器 2（EP0BCR—A5H）

EP0BCR		端点 0 状态寄存器 2					A5H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	EP0BCR[3:0]			
R	R	R	R	R			
0	0	0	0	0			

EP0BCR[3:0]	当前端点 0 OUT 传输缓冲区 FIFO 有效数据长度，对 CPU 只读。

表 26-5 端点 1 控制状态寄存器（EP1CSR—A6H）

EP1CSR		端点 1 控制状态寄存器					A6H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Rev	Rev	Rev	Rev	Rev	Rev	CLR	STALL
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
CLR	初始化清零控制位，写 1 清零端点 1 IN FIFO 指针，硬件自动将此位归零。						
STALL	端点 1 Stall 设置信号，接收到 HOST 的 Clear Feature 命令或者复位后被清除。						

表 26-6 端点 2 控制状态寄存器（EP2CSR—A7H）

EP2CSR		端点 2 控制状态寄存器					A7H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	Rev	Rev	CLR	STALL
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
CLR	初始化清零控制位，写 1 清零端点 2 OUT FIFO 指针，硬件自动将此位归零。						
STALL	端点 2 Stall 设置信号，接收到 HOST 的 Clear Feature/Stall 命令或者复位后被清除。						

表 26-7 端点 2 有效数据长度寄存器（EP2BCR—ACH）

EP2BCR		端点 2 有效数据长度寄存器					ACH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	EP2BCR[3:0]			
R	R	R	R	R			
0	0	0	0	0			
EP2BCR[3:0]	当前端点 2 OUT 传输缓冲区 FIFO 有效数据长度，对 CPU 只读。						

表 26-8 端点 3 控制状态寄存器（EP3CSR—ADH）

EP3CSR		端点 3 控制状态寄存器					ADH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ONEBUF	Rev	Rev	Rev	Rev	Rev	CLR	STALL
R/W	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
ONEBUF	1：端点 3 只使用 1 个缓冲区传送数据模式。 0：端点 3 用多个缓冲区传送数据模式。						
CLR	初始化清零控制位，写 1 清零端点 3 IN FIFO 指针，硬件自动将此位归零。						

STALL	端点 3 Stall 设置信号，接收到 HOST 的 Clear Feature/Stall 命令或者复位后被清除。

表 26-9 端点 4 控制状态寄存器（EP4CSR—AEH）

EP4CSR		端点 4 控制状态寄存器					AEH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ONEBUF	Rev		Rev	Rev	Rev	CLR	STALL
R/W	R		R	R	R	R/W	R/W
0	0		0	0	0	0	0
ONEBUF	1：端点 4 只使用 1 个缓冲区传送数据模式。 0：端点 4 用多个缓冲区传送数据模式。						
CLR	初始化清零控制位，写 1 清零端点 4 OUT FIFO 指针，硬件自动将此位归零。						
STALL	端点 4 Stall 设置信号，接收到 HOST 的 Clear Feature/Stall 命令或者复位后被清除。						

表 26-10 端点 4 有效数据长度寄存器（EP4BCR—AFH）

EP4BCR		端点 4 有效数据长度寄存器					AFH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	EP4BCR[5:0]					
R	R	R					
0	0	0					
EP4BCR[5:0]	当前端点 4 OUT 传输缓冲区 FIFO 有效数据长度，对 CPU 只读。						

26.2.4 USB设备中断寄存器

表 26-11 USB 设备中断使能寄存器（USBIE—B9H）

USBIE		USB 设备中断使能寄存器					B9H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	RSM	SUSP	SOF	URES
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
RSM	USB Resume 中断使能控制位。1：使能 RSM 中断；0：屏蔽 RSM 中断。						
SUSP	USB Suspend 中断使能控制位。1：使能 SUSP 中断；0：屏蔽 SUSP 中断。						
SOF	USB 帧头标识中断使能控制位。1：使能 SOF 中断；0：屏蔽 SOF 中断。						
URES	USB 设备复位中断使能控制位。1：使能 URES 中断；0：屏蔽 URES 中断。						

表 26-12 USB 设备中断请求/状态寄存器（USBIR—BAH）

USBIR		USB 设备中断请求/状态寄存器					BAH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	RSM	SUSP	SOF	URES
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

RSM	USB Resume 中断请求/状态位。CPU 写 1 清零。 1：有 RSM 中断请求； 0：无 RSM 中断请求。
SUSP	USB Suspend 中断请求/状态位。CPU 写 1 清零。 1：有 SUSP 中断请求； 0：无 SUSP 中断请求。
SOF	USB 帧头标识中断请求/状态位。CPU 写 1 清零。 1：有 SOF 中断请求； 0：无 SOF 中断请求。
URES	USB 设备复位中断请求/状态位。CPU 写 1 清零。 1：有 URES 中断请求； 0：无 URES 中断请求。

表 26-13 USB 端点中断使能寄存器（EPIE—BBH）

EPIE		USB 端点中断使能寄存器					BBH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	EP4 OUT	EP3IN	EP2 OUT	EP1IN	EP0 OUT	EP0IN	SUDAV
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

EP4 OUT	端点 4 OUT FIFO 非空中断使能控制位。 1：使能 EP4OUT 中断； 0：屏蔽 EP4OUT 中断。
EP3IN	端点 3 IN FIFO 空中断使能控制位。 1：使能 EP3IN 中断； 0：屏蔽 EP3IN 中断。
EP2 OUT	端点 2 OUT FIFO 非空中断使能控制位。 1：使能 EP2OUT 中断； 0：屏蔽 EP2OUT 中断。

EP1IN	端点 1 IN FIFO 空中断使能控制位。 1：使能 EP1IN 中断； 0：屏蔽 EP1IN 中断。
EP0 OUT	端点 0 OUT FIFO 非空中断使能控制位。 1：使能 EP0OUT 中断； 0：屏蔽 EP0OUT 中断。
EP0IN	端点 0 IN FIFO 空中断使能控制位。 1：使能 EP0IN 中断； 0：屏蔽 EP0IN 中断。
SUDAV	端点 0 Setup 包数据有效中断使能控制位。 1：使能 SUDAV 中断； 0：屏蔽 SUDAV 中断。

表 26-14 USB 端点中断请求/状态寄存器（EPIR—BCH）

EPIR		**USB 端点中断请求/状态寄存器**					**BCH**
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	EP4OUT	EP3IN	EP2OUT	EP1IN	EP0OUT	EP0IN	SUDAV
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
EP4OUT	端点 4 OUT 中断请求/状态位。 1：有 EP4OUT 中断请求/状态位； 0：无 EP4OUT 中断请求/状态位。 该位对 CPU 只读。该状态位的清零与 EPCSR 寄存器的 EP4RDY 清零同时发生。						
EP3IN	端点 3 IN 中断请求/状态位。 1：有 EP3IN 中断请求/状态位； 0：无 EP3IN 中断请求/状态位。该位对 CPU 只读。 该状态位的清零与 EPCSR 寄存器的 EP3RDY 清零同时发生。						
EP2OUT	端点 2 OUT 中断请求/状态位。 1：有 EP2OUT 中断请求/状态位； 0：无 EP2OUT 中断请求/状态位。该位对 CPU 只读。 该状态位的清零与 EPCSR 寄存器的 EP2RDY 清零同时发生。						

EP1IN	端点 1 IN 中断请求/状态位。 1：有 EP1IN 中断请求/状态位； 0：无 EP1IN 中断请求/状态位。该位对 CPU 只读。 该状态位的清零与 EPCSR 寄存器的 EP1RDY 清零同时发生。
EP0OUT	端点 0 OUT 中断请求/状态位。 1：有 EP0OUT 中断请求/状态位； 0：无 EP0OUT 中断请求/状态位。 该位对 CPU 只读。该状态位的清零与 EPCSR 寄存器的 EP0ORDY 清零同时发生。
EP0IN	端点 0 IN 中断请求/状态位。 1：有 EP0IN 中断请求/状态位； 0：无 EP0IN 中断请求/状态位。 该位对 CPU 只读。该状态位的清零与 EPCSR 寄存器的 EP0IRDY 清零同时发生。
SUDAV	端点 0 Setup 包数据有效中断请求/状态位。 1：有 SUDAV 中断请求/状态位； 0：无 SUDAV 中断请求/状态位。该位对 CPU 只读。 该状态位的清零与 EPCSR 寄存器的 EP0SRDY 清零同时发生。

表 26-15 USB 端点令牌中断使能寄存器（TKIE—BDH）

TKIE		**USB 端点令牌中断使能寄存器**					**BDH**
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	EP3TK	Rev	EP1TK	Rev	EP0ITK	SUTK
R	R	R/W	R	R/W	R	R/W	R/W
0	0	0	0	0	0	0	0

EP3TK	端点 3 IN 令牌中断使能控制位。 1：使能 EP3TK 中断；0：屏蔽 EP3TK 中断。
EP1TK	端点 1 IN 令牌中断使能控制位。 1：使能 EP1TK 中断；0：屏蔽 EP1TK 中断。
EP0ITK	端点 0 IN 令牌中断使能控制位。 1：使能 EP0ITK 中断；0：屏蔽 EP0ITK 中断。
SUTK	端点 0 Setup 包令牌中断使能控制位。 1：使能 SUTK 中断；0：屏蔽 SUTK 中断。

表 26-16 USB 端点中断请求/状态寄存器（TKIR—BEH）

TKIR		USB 端点中断请求/状态寄存器					BEH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	EP3TK	Rev	EP1TK	Rev	EP0ITK	SUTK
R	R	R/W	R	R/W	R	R/W	R/W
0	0	0	0	0	0	0	0
EP3TK	端点 3 IN 令牌中断请求/状态位。CPU 写 1 清零。 1：有 EP3TK 中断请求/状态位； 0：无 EP3TK 中断请求/状态位。						
EP1TK	端点 1 IN 令牌中断请求/状态位。CPU 写 1 清零。 1：有 EP1TK 中断请求/状态位； 0：无 EP1TK 中断请求/状态位。						
EP0ITK	端点 0 IN 令牌中断请求/状态位。CPU 写 1 清零。 1：有 EP0ITK 中断请求/状态位； 0：无 EP0ITK 中断请求/状态位。						
SUTK	端点 0 Setup 包令牌中断请求/状态位。CPU 写 1 清零。 1：有 SUTK 中断请求/状态位； 0：无 SUTK 中断请求/状态位。						

表 26-17 USB 端点错误中断使能寄存器（ERRIE—A1H）

ERRIE		USB 端点错误中断使能寄存器					A1H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	EP4ERR	EP3ERR	EP2ERR	EP1ERR	EP0OERR	EP0IERR	SUERR
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
EP4ERR	端点 4 OUT FIFO 读空错误中断使能控制位。 1：使能 EP4ERR 中断；0：屏蔽 EP4ERR 中断。						
EP3ERR	端点 3 IN FIFO 写满错误中断使能控制位。 1：使能 EP3ERR 中断；0：屏蔽 EP3ERR 中断。						
EP2ERR	端点 2 OUT FIFO 读空错误中断使能控制位。 1：使能 EP2ERR 中断；0：屏蔽 EP2ERR 中断。						
EP1ERR	端点 1 IN FIFO 写满错误中断使能控制位。 1：使能 EP1ERR 中断；0：屏蔽 EP1ERR 中断。						
EP0OERR	端点 0 OUT FIFO 读空错误中断使能控制位。 1：使能 EP0OERR 中断；0：屏蔽 EP0OERR 中断。						

EP0I ERR	端点 0 IN FIFO 写满错误中断使能控制位。 1：使能 EP0IERR 中断；0：屏蔽 EP0IERR 中断。
SUERR	端点 0 Setup 包读空错误中断使能控制位。 1：使能 SUERR 中断；0：屏蔽 SUERR 中断。

表 26-18 USB 端点错误中断请求/状态寄存器（ERRIR—A2H）

ERRIR		USB 端点错误中断请求/状态寄存器					A2H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	EP4 ERR	EP3 ERR	EP2 ERR	EP1 ERR	EP0O ERR	EP0I ERR	SUERR
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
EP4 ERR	端点 4 OUT FIFO 读空错误中断请求/状态位。CPU 写 1 清零。 1：有 EP4ERR 中断请求/状态位；0：无 EP4ERR 中断请求/状态位。						
EP3 ERR	端点 3 IN FIFO 写满错误中断请求/状态位。CPU 写 1 清零。 1：有 EP3ERR 中断请求/状态位；0：无 EP3ERR 中断请求/状态位。						
EP2 ERR	端点 2 OUT FIFO 读空错误中断请求/状态位。CPU 写 1 清零。 1：有 EP2ERR 中断请求/状态位；0：无 EP2ERR 中断请求/状态位。						
EP1 ERR	端点 1 IN FIFO 写满错误中断请求/状态位。CPU 写 1 清零。 1：有 EP1ERR 中断请求/状态位；0：无 EP1ERR 中断请求/状态位。						
EP0O ERR	端点 0 OUT FIFO 读空错误中断请求/状态位。CPU 写 1 清零。 1：有 EP0OERR 中断请求/状态位；0：无 EP0OERR 中断请求/状态位。						
EP0I ERR	端点 0 IN FIFO 写满错误中断请求/状态位。CPU 写 1 清零。 1：有 EP0IERR 中断请求/状态位；0：无 EP0IERR 中断请求/状态位。						
SUERR	端点 0 Setup 包错误中断请求/状态位。CPU 写 1 清零。 1：有 SUERR 中断请求/状态位；0：无 SUERR 中断请求/状态位。						

表 26-19 USB 端点错误中断使能寄存器 2（ERR2IE—A9H）

ERR2IE		USB 端点错误中断使能寄存器 2					A9H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SUDW	EP4R	EP3W	EP2R	EP1W	EP0R	EP0W	SUDR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SUDW	端点 0 Setup 缓冲区满 UDC 继续写入 Setup 包数据出错中断使能控制位。 1：使能 SUDW 中断；0：屏蔽 SUDW 中断。						

EP4R	端点 4 OUT FIFO 读空后清 RDY 出错中断使能控制位。 1：使能 EP4R 中断；0：屏蔽 EP4R 中断。
EP3W	端点 3 IN FIFO 写满后清 RDY 出错中断使能控制位。 1：使能 EP3W 中断；0：屏蔽 EP3W 中断。
EP2R	端点 2 OUT FIFO 读空后清 RDY 出错中断使能控制位。 1：使能 EP2R 中断；0：屏蔽 EP2R 中断。
EP1W	端点 1 IN FIFO 写满后清 RDY 出错中断使能控制位。 1：使能 EP1W 中断；0：屏蔽 EP1W 中断。
EP0R	端点 0 OUT FIFO 读空后清 ORDY 出错中断使能控制位。 1：使能 EP0R 中断；0：屏蔽 EP0R 中断。
EP0W	端点 0 IN FIFO 写满后清 IRDY 出错中断使能控制位。 1：使能 EP0W 中断；0：屏蔽 EP0W 中断。
SUDR	端点 0 Setup 包读空后清 SRDY 出错中断使能控制位。 1：使能 SUDR 中断；0：屏蔽 SUDR 中断。

表 26-20 USB 端点错误中断请求/状态寄存器 2（ERR2IR—AAH）

ERR2IR		**USB 端点错误中断请求/状态寄存器 2**					**AAH**
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SUDW	EP4R	EP3W	EP2R	EP1W	EP0R	EP0W	SUDR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

SUDW	端点 0 Setup 缓冲区满 UDC 继续写入 Setup 包数据出错中断请求/状态位。CPU 写 1 清零。 1：有 SUDW 中断请求/状态位；0：无 SUDW 中断请求/状态位。
EP4R	端点 4 OUT FIFO 读空后清 RDY 出错中断请求/状态位。CPU 写 1 清零。 1：有 EP4R 中断请求/状态位；0：无 EP4R 中断请求/状态位。
EP3W	端点 3 IN FIFO 写满后清 RDY 出错中断请求/状态位。CPU 写 1 清零。 1：有 EP3W 中断请求/状态位；0：无 EP3W 中断请求/状态位。
EP2R	端点 2 OUT FIFO 读空后清 RDY 出错中断请求/状态位。CPU 写 1 清零。 1：有 EP2R 中断请求/状态位；0：无 EP2R 中断请求/状态位。
EP1W	端点 1 IN FIFO 写满后清 RDY 出错中断请求/状态位。CPU 写 1 清零。 1：有 EP1W 中断请求/状态位；0：无 EP1W 中断请求/状态位。
EP0R	端点 0 OUT FIFO 读空后清 ORDY 出错中断请求/状态位。CPU 写 1 清零。 1：有 EP0R 中断请求/状态位；0：无 EP0R 中断请求/状态位。

EP0W	端点 0 IN FIFO 写满后清 IRDY 出错中断请求/状态位。CPU 写 1 清零。 1：有 EP0W 中断请求/状态位；0：无 EP0W 中断请求/状态位。
SUDR	端点 0 Setup 包读空后清 SRDY 出错中断请求/状态位。CPU 写 1 清零。 1：有 SUDR 中断请求/状态位；0：无 SUDR 中断请求/状态位。

26.2.5 端点数据FIFO数寄存器

表 26-21 端点 0 Setup 包数据 FIFO 数据寄存器（SUDFIFO—B1H）

SUDFIFO		端点 0 Setup 包数据 FIFO 数据寄存器					B1H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SUDFIFO [7:0]							
R							
0							
SUDFIFO [7:0]	端点 0 当前 Setup 包数据缓冲区 FIFO 数据寄存器，对 CPU 只读。						

表 26-22 端点 0 IN 缓冲区 FIFO 数据寄存器（EP0INFIFO—B2H）

EP0INFIFO		端点 0 IN 缓冲区 FIFO 数据寄存器					B2H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP0INFIFO [7:0]							
R/W							
0							
EP0INFIFO [7:0]	端点 0 IN 当前缓冲区 FIFO 数据寄存器，对 CPU 可读写。						

表 26-23 端点 0 OUT 缓冲区 FIFO 数据寄存器（EP0OUTFIFO—B3H）

EP0OUTFIFO		端点 0 OUT 缓冲区 FIFO 数据寄存器					B3H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP0OUTFIFO [7:0]							
R							
0							
EP0OUTFIFO [7:0]	端点 0 OUT 当前缓冲区 FIFO 数据寄存器，对 CPU 只读。						
EP0OUTFIFO [7:0]							
R							
0							

表 26-24 端点 1 IN 缓冲区 FIFO 数据寄存器（EP1FIFO—B4H）

EP1FIFO		端点 1 IN 缓冲区 FIFO 数据寄存器					B4H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP1FIFO[7:0]							
R/W							
0							
EP1FIFO[7:0]	端点 1 IN 缓冲区 FIFO 数据寄存器，对 CPU 可读写。						

表 26-25 端点 2 OUT 缓冲区 FIFO 数据寄存器（EP2FIFO—B5H）

EP2FIFO		端点 2 OUT 缓冲区 FIFO 数据寄存器					B5H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP2FIFO[7:0]							
R							
0							
EP2FIFO[7:0]	端点 2 OUT 缓冲区 FIFO 数据寄存器，对 CPU 只读。						

表 26-26 端点 3 IN 缓冲区 FIFO 数据寄存器（EP3FIFO—B6H）

EP3FIFO		端点 3 IN 缓冲区 FIFO 数据寄存器					B6H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP3FIFO [7:0]							
R/W							
0							
EP3FIFO [7:0]	端点 3 IN 当前缓冲区 FIFO 数据寄存器，对 CPU 可读写。						

表 26-27 端点 4 OUT 缓冲区 FIFO 数据寄存器（EP4FIFO—B7H）

EP4FIFO		端点 4 OUT 缓冲区 FIFO 数据寄存器					B7H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP4FIFO [7:0]							
R							
0							
EP4FIFO [7:0]	端点 4 OUT 当前缓冲区 FIFO 数据寄存器，对 CPU 只读。						

26.2.6 UFM模块控制/状态寄存器

表 26-28 UFM 模块控制/状态寄存器(UFMSR—C7H)

UFMSR	UFM 模块控制/状态寄存器	C7H

Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	CLRF EN	EP[2:0]			CLRF	SRDY
R	R	R/W	R			R	R
0	0	0	0	0	0	0	0

CLRF EN	Clear Feature 命令硬件自动清除端点 STALL 使能位，1：使能，0：禁止
EP[2:0]	Clear Feature 对应的端点号
CLRF	Clear Feature 命令指示位，1：当前命令为 Clear Feature，0：当前命令不是 Clear Feature
SRDY	端点 0 控制传输有无数据阶段指示位，1：有数据阶段，0：无数据阶段

UFM (USB Filter Module) 模块是 USB 模块的一个子模块，用于过滤一些标准的 USB 请求。硬件会自动处理这些标准请求，不需要软件的参与。

26.3 使用流程

系统上电之后，USB 模块除端点 0 外都是未使能状态，端点 0 IN FIFO 为空，CPU 可以写入数据到端点 0 IN FIFO 中；端点 0 OUT FIFO 为非空状态，初始化时 CPU 需要主动清除端点 0 FIFO 非空状态。Setup FIFO 为空状态，随时可接收 USB 主机下发的 Setup 包。其他 IN 端点缺省都是 FIFO 为空状态，CPU 可以写入数据；其他 OUT 端点缺省都是 FIFO 为空状态，随时可以接收主机下发的 OUT 包。

USB 速度模式由 IOM 模块根据外部 TM2、TM1 和 TM0 引脚信号判断，TM2、TM1、TM0 为 3b”101”时是 USB 全速模式，TM2、TM1、TM0 为 3b”110”时是 USB 低速模式。

26.3.1 初始化流程：

1) 写 DEVCFG 使能相应的端点；
2) 写 1 清除 EPCSR.EP0ORDY 位；
3) 将相关中断使能开启，如系统串行中断，USB 模块中断；
4) 置位 IOMCR.ResUp 位使能 D+/D-信号线上拉，通知主机设备插入。

26.3.2 Setup包软件处理流程

1) 预开启 EPIR.SUDAV 中断使能；

2) 当接收到一个 Setup 包，USB 模块产生中断，CPU 先查询系统中断扩展寄存器 SIV，判断中断源为 EpInt；
3) CPU 再查询 EPIR 寄存器，判断中断请求为 SUDAV 中断（此时 EPCSR.EP0SRDY 也应该为 1）；
4) CPU 写 EP0CSR.CLR 归零当前 FIFO 读指针，然后读出 8 字节 SUDFIFO 数据解析，之后清零系统中断状态寄存器 SIV.EPINT，接着向 EPCSR.EP0SRDY 写 1 清零该标志位（同时也是清除 EPIR.SUDAV 中断标志）；
5) 注：对 8 字节数据的解析也可以放到清除两级中断请求之后；
6) 根据 Setup 包解析结果，判断控制传输类型，进入控制传输下一阶段。

26.3.3 端点0 CTL IN包软件处理流程

由于 IN 端点有两类中断/状态产生，一是当前 IN FIFO 空状态，二是新的 IN 包令牌中断/状态 EPxTK，因此对所有的 IN 类型包都可以有两种处理机制：

1) 根据 Setup 包解析或者其他协议预知需要上传数据，查询 EPCSR.EP0IRDY 位等待端点 0 IN FIFO 为空，将数据预先写入 IN FIFO，并写 1 清零 EPCSR 相应端点 RDY 位。主机读走数据后，IN FIFO 会为空，CPU 再根据是否已传完数据决定是否继续写入数据到 FIFO，直到数据传输完。
2) 根据 Setup 包解析或者其他协议预知需要上传数据，但并不急于写入数据到 IN FIFO，而是等待 IN 包令牌中断请求 EpxTK 之后写入数据到 FIFO，直到数据传输完。

现在具体以机制一介绍端点 0 CTL IN 包的处理，假定解析 Setup 包后知道需要上传端点 0 数据（如 Get_Descriptor）：

3) CPU 写 EP0CSR.CLR 归零当前 FIFO 写指针；
4) 如果需上传数据个数 N<=8，写入 N 个数据到 EP0INFIFO，写 1 清 EPCSR.EP0IRDY 就结束了；
5) 如果 N>8，类似前面写入 8 字节数据之后，等待 EPCSR.EP0IRDY 查询该标志，该标志有效后，继续重复写数据到 EP0INFIFO 的过程（包括指针归零，写数据，清 EP0IRDY），直到全部数据传输完成。

26.3.4 端点0 CTL OUT包软件处理流程

假定CPU解析Setup包之后知道主机要下传N个字节端点0数据（如Set_Descriptor），流程如下：

1) 开启 EPIE.EP0OUT 或者采用查询方式
2) 等待 EPIR.EP0OUT 中断请求/查询该标志位，该标志位有效后，表明 EP0 OUT FIFO 非空。读 EP0BCR 寄存器确定有效数据长度，根据该长度在写 EP0CSR.CLR=1 归零 FIFO 指针后读出缓冲区数据，写 1 清 EPCSR.EP0ORDY；
3) 再由总长度判断是否接收完毕数据，如果还需要接收，等待 EPIR.EP0OUT 中断请求/查询该标志位。

26.3.5 端点1 INT IN包软件处理流程

假设 CPU 通过 Setup 或者其他协议知道需要通过端点 1 上传数据。

1) CPU 写 EP1CSR.CLR 归零当前 FIFO 写指针；
2) 如果需上传数据个数 N<=8，写入 N 个数据到 EP1FIFO，写 1 清 EPCSR.EP1RDY 就结束了；
3) 如果 N>8，类似前面写入 8 字节数据之后，等待 EPIR.EP1IN 中断请求/查询该标志，该标志有效后，继续重复写数据到 EP1FIFO 的过程（包括指针归零，写数据，清 EP1RDY），直到全部数据传输完成。最后一次收到的 EPIR.EP1IN 中断请求可以在清除系统级中断请求 SIV.EPINT 后不做其他处理。

26.3.6 端点2 INT OUT包软件处理流程

假设 CPU 通过 Setup 或者其他协议知道需要通过端点 2 下发数据。

1) 开启 EPIE.EP2OUT 中断或者采用查询方式；
2) 等待 EPIR.EP2OUT 中断请求/查询该标志位，该标志位有效后，读 EP2BCR 寄存器确定有效数据长度，根据该长度在写 EP2CSR.CLR=1 归零 FIFO 指针后读出缓冲区数据；
3) CPU 写 1 清零 EPCSR.EP2RDY 交出缓冲区控制权兼清除中断请求标志位，重复 2 的等待。

26.3.7 端点3 BULK IN包软件处理流程

端点 3 BULK IN 处理与端点 1 非常类似，差别在于单包最大长度由 8 字节变成 32 字节，假设 CPU 通过 Setup 或者其他协议知道需要通过端点 3 上传数据。

1) CPU 写 EP3CSR.CLR 归零当前 FIFO 写指针；
2) 如果需上传数据个数 N<=32，写入 N 个数据到 EP3FIFO，写 1 清 EPCSR.EP3RDY 就结束了；
3) 如果 N>32，类似前面写入 32 字节数据之后，等待 EPIR.EP3IN 中断请求/查询该标志，该标志有效后，继续重复写数据到 EP3FIFO 的过程（包括指针归零，写数据，清 EP3RDY），直到全部数据传输完成。最后一次收到的 EPIR.EP3IN 中断请求可以在清除系统级中断请求 SIV.EPINT 后不做其他处理。

26.3.8 端点4 BULK OUT包软件处理流程

端点 4 BULK OUT 处理则与端点 2 非常类似，差别也是在最大包长变为 32 字节。

1) 开启 EPIE.EP4OUT 中断或者采用查询方式；
2) 等待 EPIR.EP4OUT 中断请求/查询该标志位，该标志位有效后，读 EP4BCR 寄存器确定有效数据长度，根据该长度在写 EP4CSR.CLR=1 归零 FIFO 指针后读出缓冲区数据；
3) CPU 写 1 清零 EPCSR.EP4RDY 交出缓冲区控制权兼清除中断请求标志位，重复 2 的等待。

第六部分 电气特性

第27章 电气特性

27.1 最高绝对限额

此部分提供 Z8D168 芯片的绝对最高限额，在实际操作时不要超过这些参数，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 27-1 最高绝对限额

符号	描述	最小	最大	单位
TS	存储温度	-25	85	°C
VCC	电源电压	0	6.0	V
VESD	最大 ESD 电压，HBM		5000	V

27.2 操作条件

此部分显示Z8D168芯片的电压、频率以及温度特性。

表 27-2 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	20	85	°C
V_{Vcc}	电源电压	2.7	3.0/5.0	5.5	V
I_{Vcc}	Frequency:20Mhz， Vcc=5v	13	18	48	mA
$F_{inter-cpu}$	内部 CPU 核频率范围	5	20	40	MHz
Fclk	用户模式外部接口时钟		6		MHz

27.3 DC参数

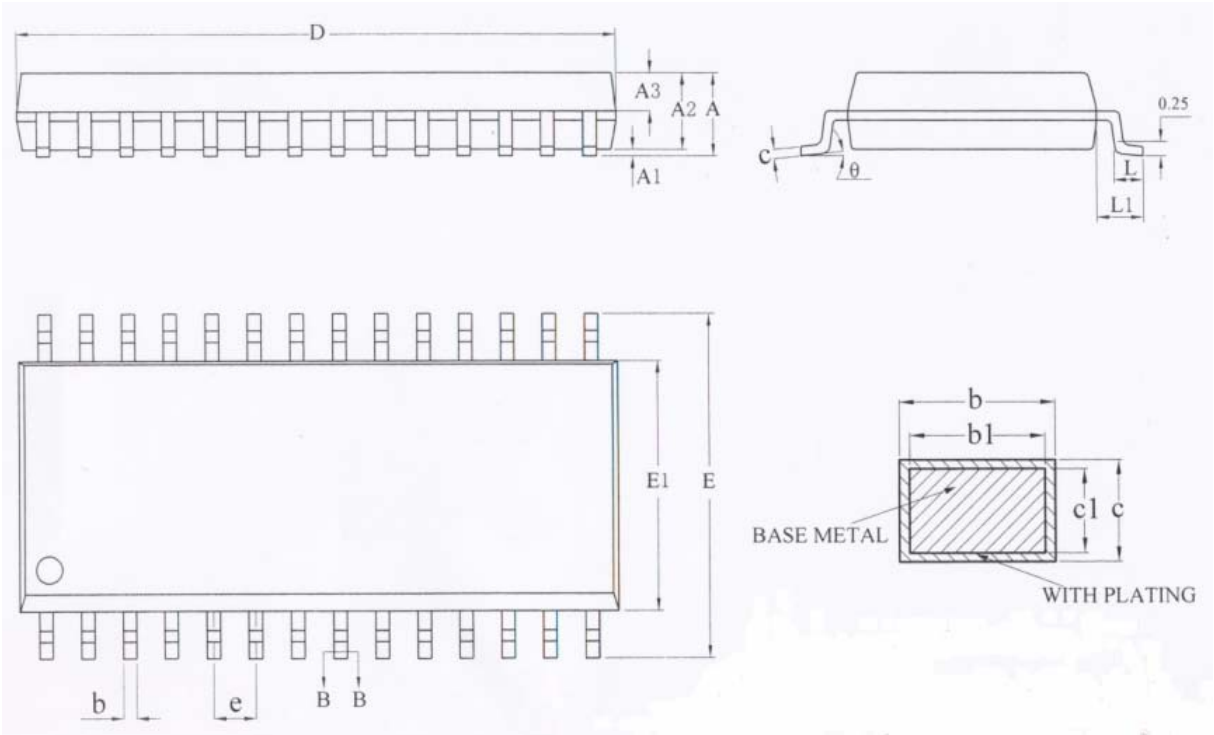
DC特性包括每一个引脚地输入门限以及输出驱动电压及电流。这些参数能够决定最大的DC负载，并决定给定负载的条件下的最大的传送时间。表 27-3显示了高低电压输入、输出以及IO引脚情况下的DC操作条件，所有的DC参数值在整个温度范围内有效。

表 27-3 标准输入、输出以及 IO 引脚 DC 操作条件

符号	描述	最小	典型	最大	单位
输入 DC 操作条件					

V_{IH}	输入高电压，所有标准输入和双向端口	0.7Vcc			V
V_{IL}	输入低电压，所有标准输入和双向端口			0.2*Vcc	V
ILIH	输入漏电流（输入高电压）			250	uA
ILIL	输入漏电流（输入低电压）			250	uA
输出 DC 操作条件					
V_{OH}	输出高电压，所有标准输出和双向端口@3mA	VDD-1.0			V
V_{OL}	输出低电压，所有标准输出和双向端口@3mA			0.8	V
I_{OH}	输出高电流，所有标准、高强度输出以及双向端口(V_O=VDD-1.0V)	3			mA
I_{OL}	输出低电流，所有标准、低强度输出以及双向端口 (V_O=0.8V)	3			mA
IO 上下拉电阻					
R_{PU}	内置上拉电阻	-	78	-	KΩ
R_{PD}	内置下拉电阻	-	-	-	KΩ

27.4 封装



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	2.00
A1	0.05	—	0.25
A2	1.65	1.75	1.85
A3	0.75	0.80	0.85
b	0.29	—	0.37
b1	0.28	0.30	0.33
c	0.15	—	0.20
c1	0.14	0.15	0.16
D	10.00	10.20	10.40
E	7.60	7.80	8.00
E1	5.10	5.30	5.50
e	0.65BSC		
L	0.75	—	1.05
L1	1.25BSC		
θ	0	—	8°
L/F载体尺寸 (mil)	153*200		

附录A 指令集

表 A-1 指令中常用的符号和标识

Rn	Register R7-R0 of the currently selected Register Bank.
Direct	8-Bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., control register, status register, etc. (128-255)].
@Ri	8-Bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.
#data	8-Bit constant included in instruction.
#data 16	16-Bit constant included in instruction.
Addr 16	16-Bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64 Kbyte Program Memory address space.
Addr 11	11-Bit destination address. Used by ACALL and AJMP. The branch will be within the same 2 Kbyte page of program memory as the first byte of the following instruction.
Rel	Signed (two's complement) 8-Bit offset byte. Used by SJMP and all conditional jumps.Range is –128~+127 bytes relative to first byte of the following instruction.
Bit	Direct Addressed Bit in Internal Data RAM or Special Function Register.

表 A-2 数据转移指令

No	Mnemonic	Description	Bytes	Clks	Opcode
1	MOV A,Rn	Rn -> A	1	2	E8~EF
2	MOV A,direct	(direct) -> A	2	3	E5
3	MOV A,@Ri	(Ri) -> A	1	2	E6~E7
4	MOV A,#data	data-> A	2	2	74
5	MOV Rn,A	A -> Rn	1	1	F8~FF
6	MOV Rn,direct	(direct) -> Rn	2	3	A8~AF
7	MOV Rn,#data	Data -> Rn	2	2	78~7F
8	MOV direct,A	A -> (direct)	2	2	F5
9	MOV direct,Rn	Rn -> (direct)	2	2	88~8F
10	MOV direct,direct	(direct) -> (direct)	3	3	85
11	MOV direct,@Ri	(Ri) -> (direct)	2	2	86~87
12	MOV direct,#data	Data -> (direct)	3	3	75
13	MOV @Ri,A	A -> (Ri)	1	1	F6~F7
14	MOV @Ri,direct	(direct) -> (Ri)	2	3	A6~A7
15	MOV @Ri,#data	data-> (Ri)	2	2	76~77
16	MOV	data16 -> DPTR	3	3	90

	DPTR,#data16				
17	MOVC A,@A+DPTR	(A+DPTR) -> A	1	2	93
18	MOVC A,@A+PC	PC+1 -> PC (A+PC) -> A	1	2	83
19	MOVX A,@Ri	(Ri) -> A	1	2	E2~E3
20	MOVX A,@DPTR	(DPTR) -> A	1	2	E0
21	MOVX @Ri,A	A -> (Ri)	1	1	F2~F3
22	MOVX @DPTR,A	A -> (DPTR)	1	1	F0
23	PUSH direct	(direct) -> STACK	2	3	C0
24	POP direct	STACK -> (direct)	2	2	D0
25	XCH A,Rn	A <--> Rn	1	3	D8~DF
26	XCH A,direct	A <--> (direct)	2	4	C5
27	XCH A,@Ri	A <--> (Ri)	1	3	C6~C7
28	XCHD A,@Ri	(A3,A2,A1,A0) <--> (Ri.3,Ri.2,Ri.1,Ri.0)	1	3	D6~D7

表 A-3 算术运算指令

No	Mnemonic	Description	Bytes	Clks	Opcode
1	ADD A,Rn	A+Rn -> A	1	2	28~2F
2	ADD A,direct	A+(direct) -> A	2	3	25
3	ADD A,@Ri	A+(Ri) -> A	1	2	26~27
4	ADD A,#data	A+data -> A	2	2	24
5	ADDC A,Rn	A+Rn+CY -> A	1	2	38#F
6	ADDC A,direct	A+(direct)+CY ->A	2	3	35
7	ADDC A,@Ri	A+(Ri)+CY -> A	1	2	36~37
8	ADDC A,#data	A+(data)+CY -> A	2	2	34
9	SUBB A,Rn	A-Rn-CY -> A	1	2	98~9F
10	SUBB A,direct	A-(direct)-CY -> A	2	3	95
11	SUBB A,@Ri	A-(Ri)-CY -> A	1	2	96~97
12	SUBB A,#data	A-data-CY -> A	2	2	94
13	INC A	A+1 -> A	1	1	04
14	INC Rn	Rn+1 -> Rn	1	2	08~0F
15	INC direct	(direct)+1 -> (direct)	2	3	05
16	INC @Ri (Ri=00~7FH)	(Ri)+1 -> (Ri)	1	2	06~07
17	DEC A	A-1 -> A	1	1	14
18	DEC Rn	Rn-1 -> Rn	1	2	18~1F
19	DEC direct	(direct)-1 -> (direct)	2	3	15

20	DEC @Ri (Ri=00~7FH)	(Ri)-1 -> (Ri)	1	2	16~17
21	INC DPTR	DPTR+1 -> DPTR	1	4	A3
22	MUL AB	A*B ->	1	3	A4
23	DIV AB	A/B ->	1	4	84
24	DA A (A=00-99H)		1	3	D4

表 A-4 布尔运算和移位指令

No	Mnemonic	Description	Bytes	Clks	Opcode
1	ANL A,Rn	A and Rn -> A	1	2	58~5F
2	ANL A,direct	A and (direct) -> A	2	3	55
3	ANL A,@Ri	A and (Ri) -> A	1	2	56~57
4	ANL A,#data	A and data -> A	2	2	54
5	ANL direct,A	(direct) and A -> (direct)	2	3	52
6	ANL direct,#data	(direct) and data -> (direct)	3	3	53
7	ORL A,Rn	A or Rn -> A	1	2	48~4F
8	ORL A,direct	A or (direct) -> A	2	3	45
9	ORL A,@Ri	A or (Ri) -> A	1	2	46~47
10	ORL A,#data	A or data -> A	2	2	44
11	ORL direct,A	(direct) or A -> (direct)	2	3	42
12	ORL direct,#data	(direct) or data -> (direct)	3	3	43
13	XRL A,Rn	A xor Rn -> A	1	2	68~6F
14	XRL A,direct	A xor (direct) -> A	2	3	65
15	XRL A,@Ri	A xor (Ri) -> A	1	2	66~67
16	XRL A,#data	A xor data -> A	2	2	64
17	XRL direct,A	(direct) xor A -> (direct)	2	3	62
18	XRL direct,#data	(direct) xor data -> (direct)	3	3	63
19	CLR A	0 -> A	1	1	E4
20	CPL A	~A -> A	1	1	F4
21	RL A	A7~A1,A7 -> A6~A0,A7	1	1	23
22	RLC A	CY,A7~A0 -> A7~A0,CY	1	1	33
23	RR A	A7,A6~A0 -> A0,A7~A1	1	1	03
24	RRC A	CY,A7~A0 >A0,CY,A7~A	1	1	13
25	SWAP A	AL < -- > AH	1	1	C4

表 A- 5　程序转移指令

No	Mnemonic	Description	Bytes	Clks	Opcode
1	ACALL addr11	PC+1 -> STACK? PC(15:11), addr11 -> PC	2	2	XXX10001
2	LCALL addr16	PC+1-> STACK? Addr16 -> PC	3	3	12
3	RET	STACK -> PC?	1	3	22
4	RETI	STACK -> PC?	1	3	32
5	AJMP addr11	PC(15:11),addr11 -> PC	2	2	XXX00001
6	LJMP addr16	Addr16 -> PC	3	3	02
7	SJMP rel	PC+1+ rel -> PC?	2	2	80
8	JMP @A+DPTR	A+DPTR -> PC	1	1	73
9	JZ rel	PC=(Acc==0)?PC+rel:PC+1	2	2	60
10	JNZ rel	PC=(Acc==1)?PC+rel:PC+1	2	2	70
11	CJNE A,direct,rel	PC=(Acc!=(direct))?PC+rel:PC+1	3	4	B5
12	CJNE A,#data,rel	PC=(Acc!==data)?PC+rel:PC+1	3	3	B4
13	CJNE Rn,#data,rel	PC=(Rn!=data)?PC+rel:PC+1	3	3	B8~BF
14	CJNE @Ri,#data,rel	PC=((Ri)!=data)?PC+rel:PC+1	3	3	B6~B7
15	DJNZ Rn,rel	Rn-1 -> Rn PC=(Rn!=0)?PC+rel:PC+1	3	3	D8~DF
16	DJNZ direct,rel	(direct)-1->(direct) PC=((direct)!=0)?PC+rel:PC+1	3	4	D5
17	NOP		1	1	00

表 A- 6 位操作指令

No	Mnemonic	Description	Bytes	Clks	Opcode
1	CLR C	Clear CY	1	1	C3
2	CLR Bit	Cleat Bit	2	2	C2
3	SETB C	set CY	1	1	D3
4	SETB Bit	set Bit	2	2	D2
5	CPL C	~CY -> CY	1	1	B3
6	CPL Bit	~Bit -> Bit	2	3	B2
7	ANL C, Bit	CY and Bit -> CY	2	3	82
8	ANL C, /Bit	CY and ~Bit -> CY	2	3	B0
9	ORL C,Bit	CY or Bit -> CY	2	3	72
10	ORL C,/Bit	CY or ~Bit -> CY	2	3	A0
11	MOV C,Bit	Bit -> CY	2	3	A2
12	MOV Bit,C	CY -> Bit	2	2	92
13	JC rel	PC=(CY==1)?PC+rel:PC+1	2	2	40
14	JNC rel	PC=(CY==0)?PC+rel:PC+1	2	2	50
15	JB Bit,rel	PC=(Bit==1)?PC+rel:PC+1	3	3	20
16	JNB Bit,rel	PC=(Bit==0)?PC+rel:PC+1	3	3	30
17	JBC Bit,rel	PC=(Bit==1)?PC+rel:PC+1 0 ->Bit	3	3	10

附录B 特殊功能寄存器

表 B-1 Z8D168 特殊功能寄存器(SFR)定义

寄存器名		功能描述	地址	初始值
B		B 寄存器	F0H	00H
ACC		累加器	E0H	00H
PSW		程序状态字	D0H	00H
P0		P0 口	80H	00H
DPH		数据指针（高位字节）	83H	00H
DPL		数据指针（低位字节）	82H	00H
SP		堆栈指针	81H	07H
DPL2		数据指针 2（低位字节）	84H	00H
DPH2		数据指针 2（高位字节）	85H	00H
DPS		双 DPTR 切换寄存器	86H	00H
MPU	MPUCR	MPU 控制寄存器	FFH	00H
	MPUSR	MPU 状态寄存器	FEH	00H
	ROMBANK	ROM 总线 bank 选择寄存器	FDH	00H
	RAMBANK	RAM 总线 bank 选择寄存器	FCH	00H
	MPUGIDA	SectorGid 寄存器 A	F8H	00H
	MPUGIDB	SectorGid 寄存器 B	F9H	00H
CFC	CFCCSR	FLASH 写/擦除控制寄存器	9AH	00H
	CFCOTPR	FLASH OTP 标志字节寄存器	9BH	FFH
中断控制	IE	中断允许控制	A8H	00H
	IP	中断优先级控制	B8H	00H
	XIE	扩展外部中断使能寄存器	D1H	00H
	XIV	扩展外部中断向量寄存器	D2H	00H
	SIV	SI 扩展中断标志寄存器	D3H	00H
	XXIEV	EX1 扩展中断标志寄存器	ABH	00H
CGU	CGUFDR	CPU 核时钟分频系数寄存器	E9H	01H
	CGUFCR	功能模块时钟控制寄存器	E8H	C2H
	PCON	电源控制	87H	00H
WDT	WDTCSR	看门狗控制及状态寄存器	91H	00H
	WDTTAP	看门狗写控制寄存器	92H	00H
TMU	TH1	定时器 1（高位字节）	8DH	00H

	TL1	定时器 1（低位字节）	8BH	00H
	TH0	定时器 0（高位字节）	8CH	00H
	TL0	定时器 0（低位字节）	8AH	00H
	TMOD	定时器方式控制	89H	00H
	TCON	定时器控制	88H	00H
RCU	RCUCR	复位控制寄存器	EFH	00H
DES	DESKR	DES 密钥寄存器	E7H	00H
	DESIV	DES 初始向量寄存器	E6H	00H
	DESDR	DES 数据寄存器	E5H	00H
	DESCR	DES 控制寄存器	E4H	00H
PAE	PSAECSR	PAE 控制状态寄存器	94H	00H
	PAEMOD	PAE 模式寄存器	F7H	00H
	PAEELENH	PAE 幂长寄存器高 8 位	F6H	00H
	PAEELENL	PAE 幂长寄存器低 8 位	F5H	00H
	PAENLENH	PAE 模长寄存器高 8 位	F4H	00H
	PAENLENL	PAE 模长寄存器低 8 位	F3H	00H
	PAECR	PAE 启动运算寄存器	F2H	00H
	PAECMD	PAE 控制命令寄存器	F1H	00H
RNG	RNGDATA	RNG 数据寄存器	D7H	00H
	RNGMODE	RNGMODE 选择寄存器	D6H	00H
	RNGNUM	随机数状态寄存器	D4H	00H
	RNGCOMMAND	RNGCOMMAND 寄存器（片选）	D5H	00H
KGE	KGEDND	KGE 低 8 位被除数寄存器	ECH	00H
	KGESOR	KGE 除数寄存器	EDH	00H
	KGERMN	KGE 余数寄存器	EEH	00H
AES	AESCSR	AES 控制寄存器	93H	00H
SSF33	SSFCSR	SSF33 算法控制模块控制状态寄存器	96H	00H
SCB2	SCBCSR	SCB2 算法控制模块控制状态寄存器	95H	00H
SHF	SHFTDR	移位数据寄存器	97H	00H
DMA	DMACSR	DMA 模式控制状态寄存器	CFH	00H
	DMAXBASE	DMA 模式 XRAM 基址寄存器	DFH	00H
	DMAXOFS	DMA 模式 XRAM 偏移地址寄存器	EBH	00H
SEC	SECCR	SEC 控制寄存器	E1H	00H
	SECSR	SEC 状态寄存器	E3H	00H
IOM	GPIOCR2	GPIO 方向控制寄存器 2	9DH	00H

	IOMCR2	IO 模式控制寄存器 2	9EH	00H
	IOMCR	IOM 模式控制寄存器	9CH	00H
	GPIOCR	GPIO 方向控制寄存器	9DH	00H
SPI	SPICR0L	SPI 控制寄存器 0 低 8 位	C8H	00H
	SPICR0H	SPI 控制寄存器 0 高 8 位	C9H	00H
	SPICR1	SPI 控制寄存器 1	CAH	00H
	SPIDR	SPI 数据寄存器	CBH	00H
	SPISR	SPI 状态寄存器	CCH	00H
	SPICPSR	SPI 时钟预定标器寄存器	CDH	00H
	SPIIIR/SPIICR	SPI 中断标识/清除寄存器	CEH	00H
SCD	SCDISR	SCD 中断状态寄存器	C0H	00H
	SCDIER	SCD 中断允许寄存器	C1H	00H
	SCDCSR	SCD 控制寄存器	C2H	00H
	SCDDR	SCD 数据寄存器	C3H	00H
	SCDBPRL	SCD 波特率参数低位寄存器	C4H	74H
	SCDBPRH/ECR	SCD 波特率参数高位/ETU 计数寄存器	C5H	01H
SCC	SCCSR	SCC 中断状态寄存器	D8H	00H
	SCCIER	SCC 中断使能寄存器	D9H	00H
	SCCTCR	SCC 传输控制寄存器	DAH	00H
	SCCUCR	SCC 用户控制寄存器	DBH	00H
	SCCTDR	SCC 传输数据寄存器	DCH	00H
	SCCRDR	SCC 接收数据寄存器	DCH	00H
	SCCBPRL	SCC 波特率参数低位寄存器	DDH	00H
	SCCBPRH	SCC 波特率参数高位寄存器	DEH	00H
UART	UARTISR	UART 中断状态寄存器	C0H	00H
	UARTIER	UART 中断允许寄存器	C1H	00H
	UARTCS	UART 控制与状态寄存器	C2H	00H
	UARTDATA	UART 数据寄存器	C3H	00H
	UARTBPRL	UART 波特率寄存器低位	C4H	74H
	UARTBPRH	UART 波特率寄存器高位	C5H	01H
USB	DEVCFG	USB 设备配置寄存器	BFH	00H
	EPCSR	USB 端点控制状态寄存器	A3H	2EH
	EP0CSR	端点 0 控制状态寄存器	A4H	00H
	EP0BCR	端点 0 状态寄存器 2	A5H	00H
	EP1CSR	端点 1 控制状态寄存器	A6H	00H

	EP2CSR	端点 2 控制状态寄存器	A7H	00H
	EP2BCR	端点 2 有效数据长度寄存器	ACH	00H
	EP3CSR	端点 3 控制状态寄存器	ADH	00H
	EP4CSR	端点 4 控制状态寄存器	AEH	00H
	EP4BCR	端点 4 有效数据长度寄存器	AFH	00H
	USBIE	USB 设备中断使能寄存器	B9H	00H
	USBIR	USB 设备中断请求/状态寄存器	BAH	00H
	EPIE	USB 端点中断使能寄存器	BBH	00H
	EPIR	USB 端点中断请求/状态寄存器	BCH	00H
	TKIE	USB 端点令牌中断使能寄存器	BDH	00H
	TKIR	USB 端点中断请求/状态寄存器	BEH	00H
	ERRIE	USB 端点错误中断使能寄存器	A1H	00H
	ERRIR	USB 端点错误中断请求/状态寄存器	A2H	00H
	ERR2IE	USB 端点错误中断使能寄存器 2	A9H	00H
	ERR2IR	USB 端点错误中断请求/状态寄存器 2	AAH	00H
	SUDFIFO	端点 0 Setup 包数据 FIFO 数据寄存器	B1H	00H
	EP0INFIFO	端点 0 IN 缓冲区 FIFO 数据寄存器	B2H	00H
	EP0OUTFIFO	端点 0 OUT 缓冲区 FIFO 数据寄存器	B3H	00H
	EP1FIFO	端点 1 IN 缓冲区 FIFO 数据寄存器	B4H	00H
	EP2FIFO	端点 2 OUT 缓冲区 FIFO 数据寄存器	B5H	00H
	EP3FIFO	端点 3 IN 缓冲区 FIFO 数据寄存器	B6H	00H
	EP4FIFO	端点 4 OUT 缓冲区 FIFO 数据寄存器	B7H	00H

附录C 术语与缩略语

CFC	Combo Flash Controller，FLASH 控制器
CGU	Colck Generate Unit,时钟产生单元
DES	Data Encryption Standard，数据加密标准
IOM	IO Manager。IO 管理单元
KGE	Key Generate Engine 密钥对生成引擎
MPU	Memory Protecting Unit ,存储管理单元
OTP	One Time Program
PAE	Public Algorithm Engine 公钥算法引擎
PD	Power Down
PKI	Public Key Infrastructure，公钥基础设施
RCU	Recet Control Unit,复位控制单元
RNG	Random Number Generator，随机数生成器
SCD	Smart Card Device，智能卡设备接口
SEC	安全防护单元
UART	Universal Asynchronous Receiver Transmitter，通用异步收发器，也称串口。
WDT	Watch Dog Timer,看门狗定时器
XI	Extended Interrupt，扩展中断

附录E 版本历史

版本号	日期	人员	描述
V1.0	2008-07-13	L.HQ	完成正式版文档。