ZTEIC

用“芯”捍卫安全

8位USB-KEY芯片

Z8D64U用户手册

Version 1.2

目 录

图 列 表

表 格 列 表

第一部分 系统概述

第1章 概述

Z8D64U 是仙人球（Cacti）Z8 系列中一款带有 USB 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。Z8D64U 片上集成了单周期高性能 8 位安全微控制器、256Byte 片上核内 RAM、2KByte 片上核外 RAM、1KByte 片上核外 PAERAM、64KByte Flash 程序/数据存储器，及 USB、UART 和 7816 等接口电路。Z8D64U 内置 DES/3DES、SSF33、SCB3 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。Z8D64U 芯片具有功耗低、稳定性高和兼容性强等特点。USB 接口完全兼容 USB1.1 协议。

Z8D64U功能框图如 图 1-1所示：

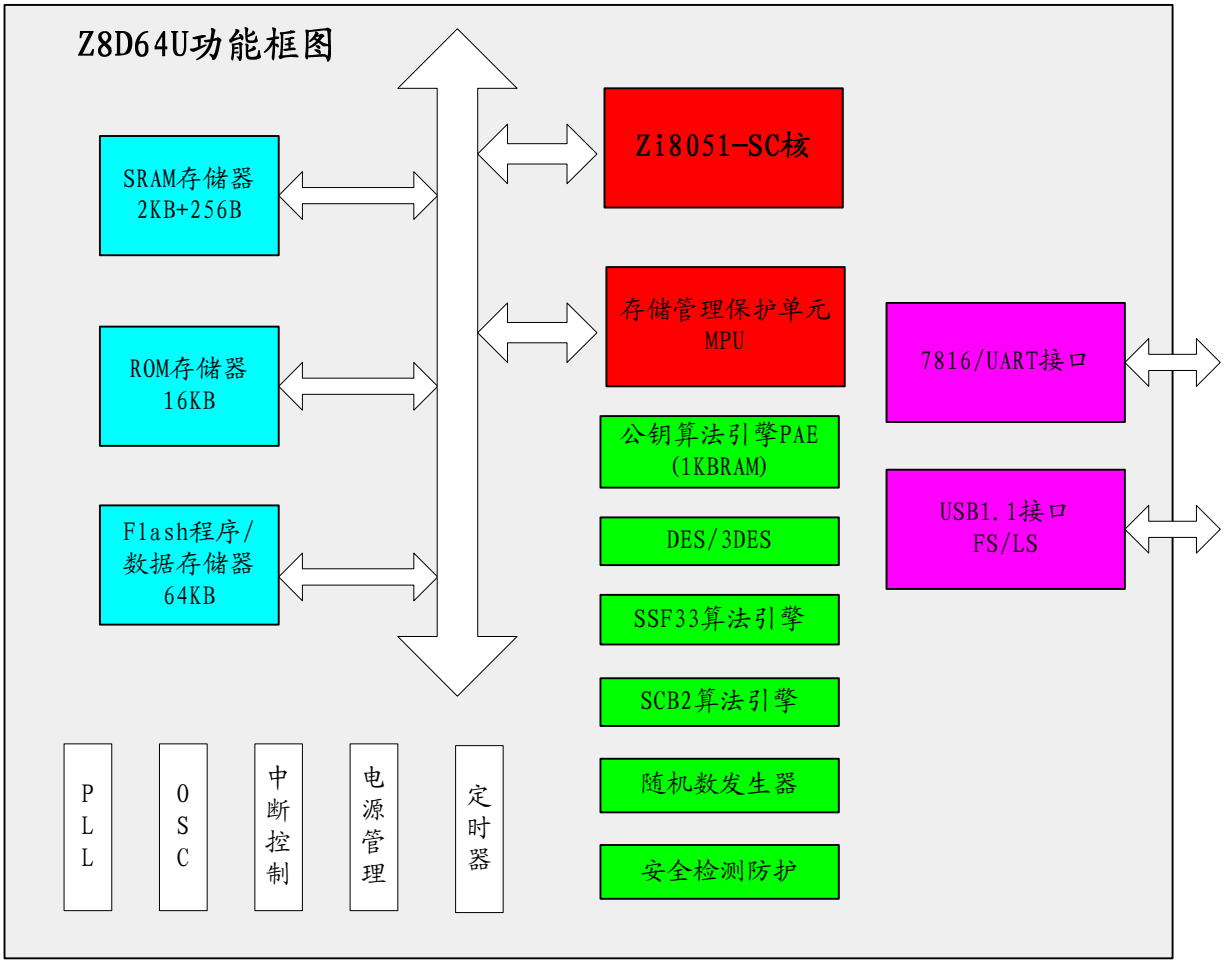


图 1-1 Z8D64U 芯片功能框图

Z8D64U封装管脚图，如 图 1-所示：

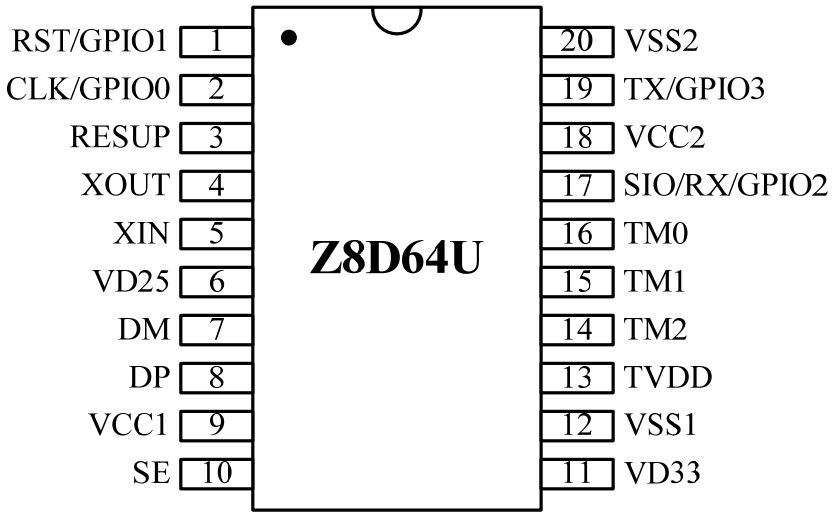


图 1-2 Z8D64U管脚分配

各引脚对应功能如 表 1-1所示：

表 1-1 Z8D64U 外部引脚信号定义

管脚名	功 能 描 述	管脚类型
VCC1 VCC2	系统电源，工作范围 3.0V-5.5V	电源
VSS1 VSS2	系统地	地
DP	USB 总线 D+模拟 IO	输入/输出
DM	USB 总线 D-模拟 IO	输入/输出
XIN	外部无源晶体震荡输入端	输入
XOUT	外部无源晶体震荡输出端	输出
CLK/GPIO0	7816 接口时钟引脚；复用为 GPIO0	输入/输出
RST/GPIO1	7816 接口复位引脚；复用为 GPIO1	输入/输出
SIO/RX/GPIO2	7816 接口复位引脚，复用为 UART 接口接收引脚，复用为 GPIO2	输入/输出
TX/GPIO3	UART 接口发送引脚，复用为 GPIO3	输入/输出
Resup	USB 应用中作为 D+/D-上拉电阻的控制信号	输出
TM0 TM1 TM2	芯片工作模式配置引脚	输入
SE	芯片工作时，本引脚需要固定为低	输入
VD33	3.3V 电压输出引脚，可以接电容到地，提高系统稳定性	电源
VD25	2.5V 电压输出引脚，可以接电容到地，提高系统稳定性	电源
TVDD	测试电压输出引脚，可以接电容到地，提高系统稳定性	电源

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
  - 可位寻址
  - 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（POWER DOWN）和深度被动休眠模式（CLKSTP）
  - 内部时钟频率：核时钟最大 20MHz，PAE 运算时钟最大 40M
- 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

- 16KB ROM
- 64KB Code Flash/ Data FLASH（64KB Flash + 512Byte OTP Block）
  - 64KB Flash 用于程序、函数库、数据的存储
    - 页面大小为 64 字节
    - 最少擦写次数 10 万次 25℃
    - 擦写性能
      - 单字节编程时间：51us（最大）
      - 页擦除时间：1.01ms（最快）
    - 室温（25℃）下数据保持时间最少 10 年；
  - 512Byte OTP Block 用于用户关键/敏感数据的存储
- RAM：3.25KB（256Byte 核内 RAM+2KByte 核外 RAM+1KByte 核外

PAERAM）

2.3 安全组件

- 公钥算法引擎
  - 内部时钟：40MHz （最大）
  - 支持大数（1024 位）模乘/模幂/乘法运算协处理
  - 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 12 次/秒（不带 CRT）
  - 1024 位 RSA 算法签名速度达到 24 次/秒（带 CRT）
  - 1024 位 RSA 密钥对生成时间小于 1.5 秒（带 CRT）
  - 1024 位 RSA 密钥对生成时间小于 2.0 秒（不带 CRT）
- 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 加解密速度大于 3.0Mbps
- SCB2 算法引擎
  - 支持 SCB2 算法的加密和解密
  - 加解密速度大于 3.0Mbps
- SSF33 算法引擎
  - 支持 SSF33 算法的加密和解密
  - 加解密速度 3.0Mbp s
- 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 FS/LS
  - 最高传输速率： FS/12Mbps，LS/1.5Mbps
  - 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
  - 低速模式下可以不使用外部晶振
  - 实际流加密速度最高可以达到 3.0Mbps
- ISO7816 从控制器
  - 具备一路 ISO7816 从控制器，可以作为从设备接读卡器
  - 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
- GPIO
  - 双向可配置 GPIO

2.5 电气特性

- 整个芯片的功耗小于 40mA（3.3V@20M 主频情况下），芯片的平均功耗小于 23mA(3.3V@20M 主频情况下)，低功耗模式下低于 500uA

- USB 低速模式支持内部时钟模式，无需外部时钟电路即可工作。
- 电源电压：2.7~5.5V
- ESD 保护：4KV 以上
- 符合 ISO7816-2 规范

2.6 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

2.7 产品封装

- Wafer/DIE
- SSOP20-209Mil
- 卡封装

2.8 应用产品

- 低成本身份认证 USBkey
- 电子签章
- 软件加密锁

第二部分 主处理器单元

第3章 CPU核

3.1 概述

Z8D64U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：

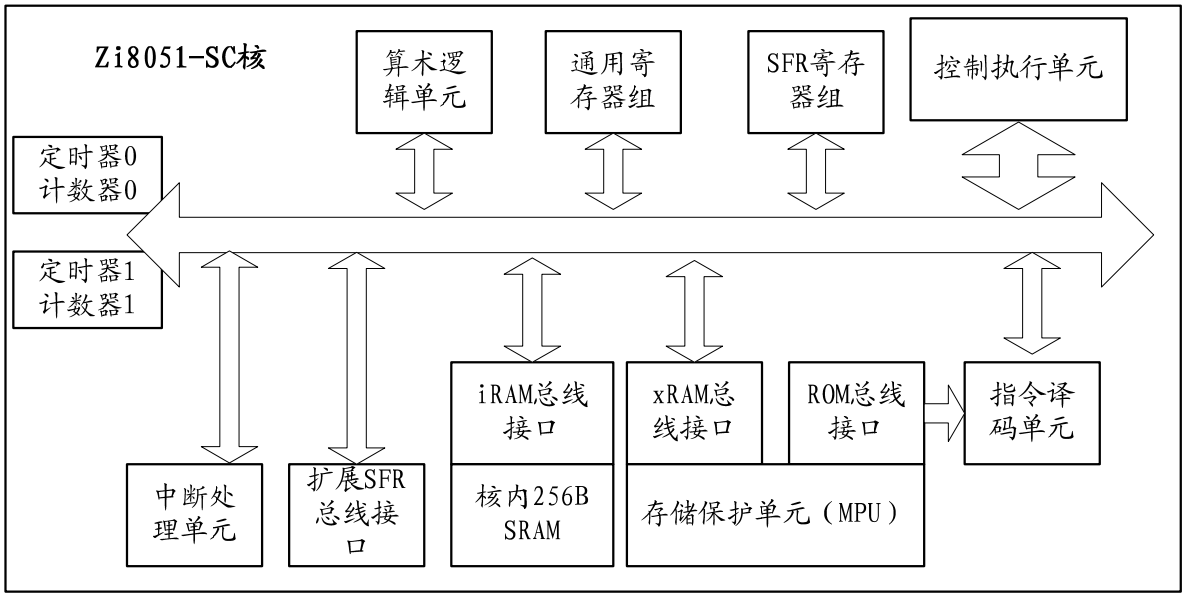


图 3-1 Zi8051-SC 内核及存储接口部分结构图

3.1.2 特性

➢ 程序存储器接口：程序存储器可以是 ROM 或 FLASH，支持多 Bank 技术，最大空间大于 64KB

- 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（POWER DOWN）和深度被动休眠模式（CLKSTP）
- 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

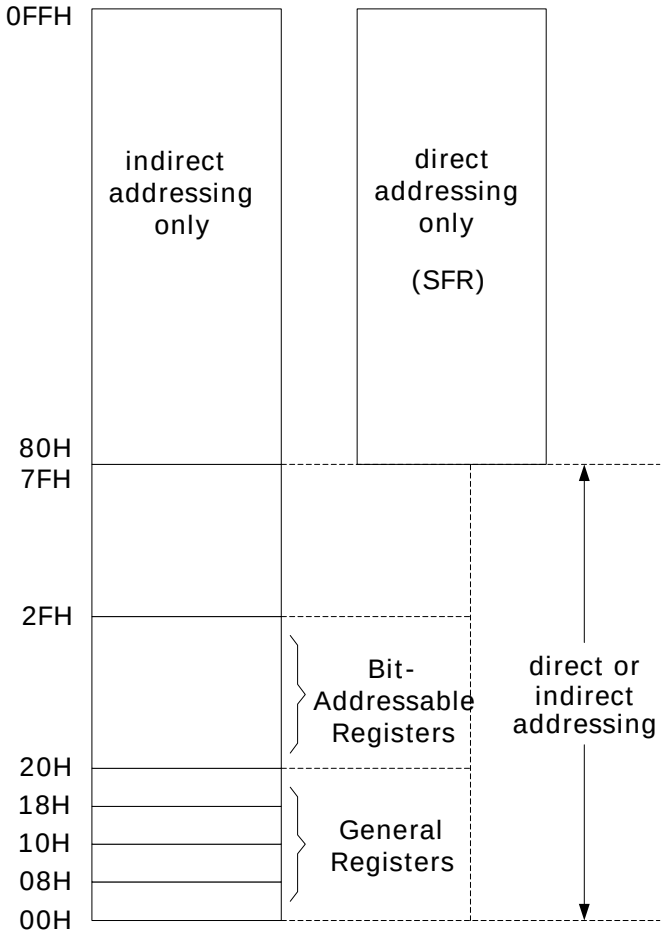


图 3-2 内部 RAM 空间分配

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和端口控制寄存器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。

Z8D64U 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 寄存器配置

4.2.1 MPU控制寄存器

表 4-1 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	SectorAccessEn	Rev
-	-	-	-	-	-	R/W	—
0	0	0	0	0	0	0	0
SectorAccessEn	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						

4.2.2 MPU状态寄存器

表 4-2 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	MFEF	MREF	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	GIDEF =1：表示 GID 非法访问错误 GIDEF =0：表示无 GID 非法访问错误						
MFEF	MFEF =1：表示 Flash 非法访问错误 MFEF =0：表示无 Flash 非法访问错误						
MREF	MFEF =1：表示 16KRom 非法访问错误 MFEF =0：表示无 16KRom 非法访问错误						
SFREF	SFREF =1：表示 SFR 非法访问错误 SFREF =0：表示无 SFR 非法访问错误						
INFEF	OTPEF =1：表示 INF 非法访问错误 OTPEF =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。							

4.2.3 Rom Bank选择寄存器

表 4-3 Rom Bank 选择寄存器(ROMBANK－FDH)

ROMBANK		Rom Bank 选择寄存器					FDH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	Rev	Rev	Rev	ROMBANK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ROMBANK		ROMBANK =0：表示选中 ROMBANK0，选中 Flash 低 32K 空间。 ROMBANK =1：表示选中 ROMBANK1，选中 Flash 高 32K 空间。					

4.2.4 Ram Bank选择寄存器

表 4-4 Ram Bank 选择寄存器(RAMBANK－FCH)

RAMBANK		Ram Bank 选择寄存器					FCH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	Rev	Rev	Rev	RAMBANK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
RAMBANK		RAMBANK =0：表示 RAMBANK0，选中 Flash 低 32K 空间。 RAMBANK =1：表示 RAMBANK1，选中 Flash 高 32K 空间。					

4.2.5 SectorGid寄存器A

表 4-5 SectorGid 寄存器 A(MPUSGA－F8h)

MPUSGA		SectorGid 寄存器 A					F8H
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
Sector3Gid[1:0]		Flash 中扇区 3 的 Group ID，地址区域为：6000~7fff					
Sector2Gid[1:0]		Flash 中扇区 2 的 Group ID，地址区域为：4000~5fff					
Sector1Gid[1:0]		Flash 中扇区 1 的 Group ID，地址区域为：2000~3fff					
Sector0Gid[1:0]		Flash 中扇区 0 的 Group ID，地址区域为：0000~1fff					

4.2.6 SectorGid寄存器B

表 4-6 SectorGid 寄存器 B(MPUSGB－F9h)

MPUSGB		SectorGid 寄存器 B					F9H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sector7Gid[1:0]		Sector6Gid[1:0]		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
Sector7Gid[1:0]		Flash 中扇区 7 的 Group ID，地址区域为：e000~ffff					
Sector6Gid[1:0]		Flash 中扇区 6 的 Group ID，地址区域为：c000~dfff					

Sector5Gid[1:0]	Flash 中扇区 5 的 Group ID，地址区域为：a000~bfff
Sector4Gid[1:0]	Flash 中扇区 4 的 Group ID，地址区域为：8000~9fff

4.3 访问控制规则

- MPUSR（MPU 状态寄存器），只可 ROM 区访问；
- ROM 区程序不可访问(Movc/Movx)FLASH 区；
- FLASH 区不可访问 ROM 区；
- INFO 区（OTP，512 字节）只能由 ROM 访问，权限全开；
- 64k FLASH 按 8k 分为 8 个 Sector, 每个 Sector 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。

4.4 存储器空间

Z8D64U 的存储资源主要由以下几部分构成：

- 16K 字节的程序 ROM 空间（16K 字节 ROM）
- 64K 字节的 Flash 空间（64K 字节 Flash）
- 256 字节 MCU 内部 RAM
- 2K 字节 MCU 外部 RAM
- 1K 字节外部 PAE RAM，不作 PAE 运算时可以作为 MCU 外部 RAM 使用

4.4.1 Rom存储器分配

Z8D64U的Rom存储器空间分配如 图 4-1所示：

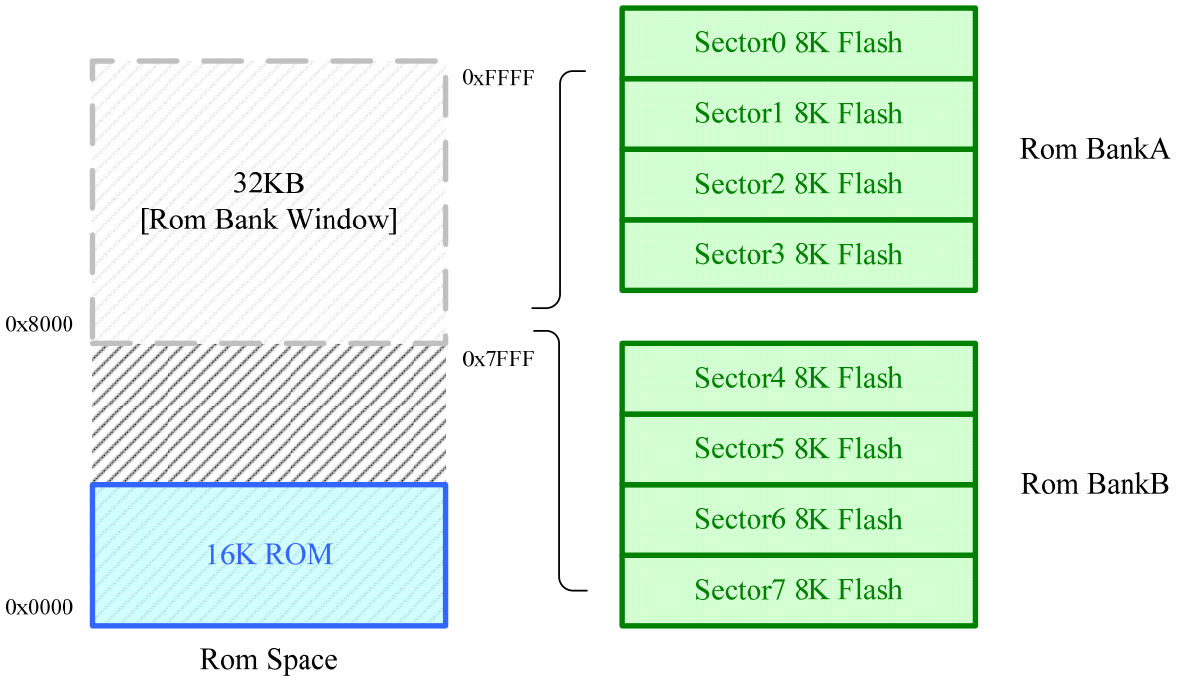


图 4-1 Rom Bank Assignment Map

表 4-7 Rom 地址分配表

Rom Bus			
	ROMBANK	***Start***	***End***
ROM	*	0x0000	0x7FFF
BANK0	0	0x8000	0xFFFF
BANK1	1	0x8000	0xFFFF

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，图中绿色区域A0~A3, B4~B7 块分别对应 64K Flash由低到高的 8 个 8K字节存储块，蓝色为 16K ROM。

3、由 图 4-1可见在RomSpace上，低 16k寻址空间（ROM： 0x0000 ~ 0x3FFF）固定对应 ROM，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Memory Block。

4.4.2 Ramx存储器分配

Z8D64U的Ramx存储器空间分配如 图 4-2所示：

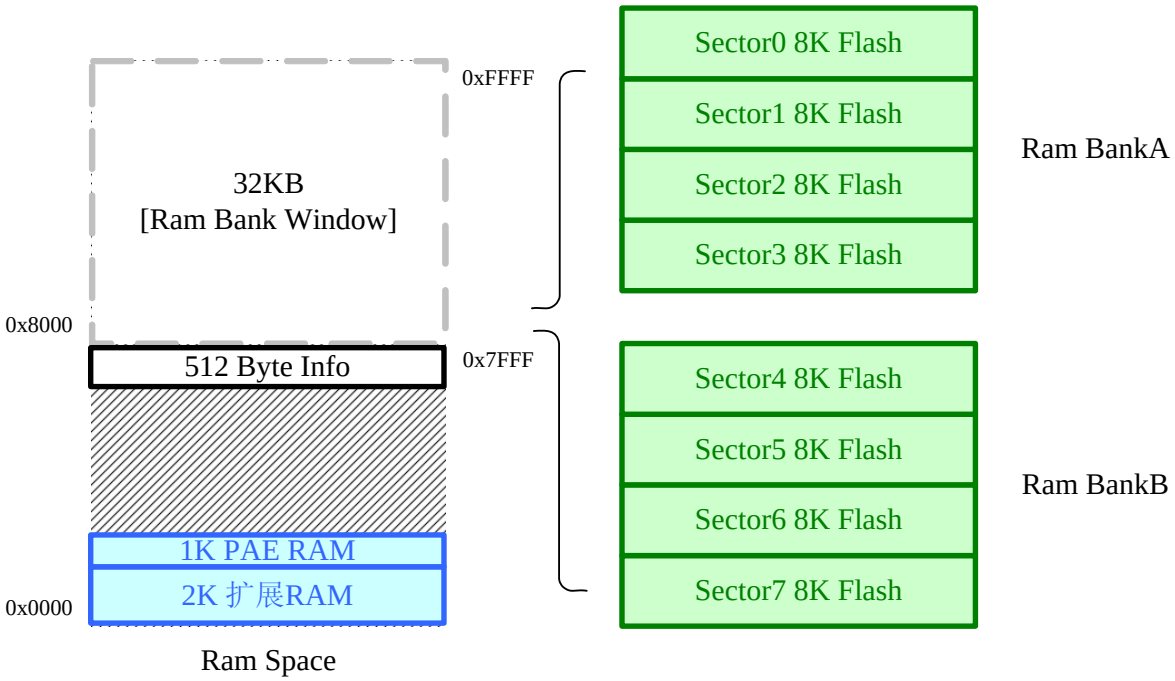


图 4-2 Ramx Bank Assignment Map

表 4-8 Ramx 地址分配表

RAMX 地址空间	容量	说明
0x0000 ~ 0x07FF	2KB	扩展 Ram
0x0800 ~ 0x0BFF	1KB	PAE/Hash Ram
0x7E00 ~ 0x7FFF	512 B	Flash Info (OTP)
0x8000 ~ 0xFFFF	32KB	Ramx Bank Window

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所示，由 图 4-2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。

第5章 FLASH存储器控制器(RFC)

5.1 概述

Z8D64U 片上集成了 16K 字节的 ROM 存储器，64K 字节的 Flash 存储器另外还有 512 字节用作 OTP 区域。RFC（ROM Flash Control）以 XRAM 总线接口完成 CPU 访问片上 Flash 的操作。

- 实现 64K+512 Byte Flash 的访问
- 实现 512Byte OTP 区域
- 通过 CPU XRAM 总线访问片上 Flash
- 通过 RFC 模块，CPU 可以对 Flash 进行单字节的读、写操作，扇区擦除操作（可以擦除 64Byte），块擦除操作（可以擦除 4Kbyte），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

RFC 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

表 5-1 FLASH 写/擦除控制寄存器（RFCCSR－CAH）

RFCCSR		**FLASH 写/擦除控制寄存器**					**CAH**
Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FWRE	FWMOD			FPETO	FVRS	FTOEN	OTPCW
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FWRE	Flash 写操作使能						

FWMOD	Flash 写/擦除模式选择： FWMOD = 0，Byte Program，字节编程模式； FWMOD = 1，保留 FWMOD = 2，Sector Erase； FWMOD = 3，Block Erase，块擦除模式； FWMOD = 4，Sector VREAD； FWMOD = 5，保留 FWMOD = 6，保留 FWMOD = 7，保留
FPETO	FLASH 擦除/编程操作超时标志
FVRS	FLASH VREAD 校验错误标志
FTOEN	FLASH 擦除/编程操作超时检测使能
OTPCW	OTP 封口使能

5.2.2 OTP测试寄存器

表 5-2 OTP 测试寄存器（RFCOTPR－CBH）

RFCOTPR		**OTP 测试寄存器**					**CBH**
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFCOTPR [7:0]							
R							
FFH							
RFCOTPR [7:0]	该寄存器为 FLASH INFO 区 OTP 信息的寄存器； 该寄存器只读； 执行 OTP 封口后，该寄存器的值会自动 Update。 该寄存器的每一位对应 FLASH INFO 区的一个扇区，为 0 的位对应的扇区被封口，不能再进行擦除/编程操作。例如本寄存器值为 0xfe，则 FLASH INFO 区第一个扇区（0x7e00-0x7e3f）被封口						

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 Zi8051-SC 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 DES、PAE 和 Ssf33 模块中断复用到扩展中断 INT0，SEC、MPU、WDT 和 RNG 模块中断以及 RST 引脚上的中断复用到 INT1。

6.2 Zi8051-SC核中断处理单元IPU

Zi8051-SC 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：

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						

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）和 1 个 SI 扩展中断标志寄存器。CPU 操作这 3 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。

6.1.1 扩展中断使能寄存器

表 6-3 扩展中断使能寄存器(XIE—D1H)

XIE		扩展中断使能寄存器					D1H
X0IE			X1IE				
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DesIntEn	PaeIntEn	SsfIntEn	SecIntEn	MpuIntEn	WdtIntEn	RngIntEn	ExtIntEn
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						
SsfIntEn	Ssf33 模块中断使能						
SecIntEn	Sec 模块中断使能						
MpuIntEn	Mpu 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

6.1.2 扩展中断标志寄存器

表 6-4 扩展中断标志寄存器(XIV—D2H)

XIV		扩展中断标志寄存器					D2H
X0IV			X1IV				
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DesIntVec	PaeIntVec	SsfIntVec	SecIntVec	MpuIntVec	WdtIntVec	RngIntVec	ExtIntVec
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
PaeIntVec	Pae 模块中断标志，硬件置 1，软件清 0
SsfIntVec	Ssf33 模块中断标志，硬件置 1，软件清 0
SecIntVec	Sec 模块中断标志，硬件置 1，软件清 0
MpuIntVec	Mpu 模块中断标志，硬件置 1，软件清 0
WdtIntVec	Wdt 模块中断标志，硬件置 1，软件清 0
RngIntVec	Rng 模块中断标志，硬件置 1，软件清 0
ExtIntVec	ExtInt 中断标志，硬件置 1，软件清 0

6.1.3 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
SIV							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Err2IntVec	Err1IntVec	TkIntVec	EpIntVec	DevIntVec
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

Err2IntVec	USB 模块错误类中断 2 标志，硬件置 1，软件清 0
Err1IntVec	USB 模块错误类中断 1 标志，硬件置 1，软件清 0
TkIntVec	USB 模块令牌类中断标志，硬件置 1，软件清 0
EpIntVec	USB 模块端点类中断标志，硬件置 1，软件清 0
DevIntVec	USB 模块设备类中断标志，硬件置 1，软件清 0

第7章 时钟控制和复位管理(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。Z8D64U 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。

一、 CPU 和 PAE 频率控制

Z8D64U 提供对 CPU 和 PAE 模块工作频率的配置。

Z8D64U 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、Z8D64U 提供三种种低功耗模式：

1. IDLE 模式
2. 深度主动休眠模式（POWER DOWN）
3. 深度被动休眠模式（CLKSTP）

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

三、Z8D64U 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
Rev	VRStandbyEn	ClkGscRngEn	CSDE	Pae FreqDivNum		CoreFreqDivNum	
R/W	R/W	R/W	R/W	R/W		R/W	
0	0	0	0	0		1	

VRStandbyEn	VR Standby 使能信号 为 1 时，在 powerdown 时，VR 进入 standby 状态 为 0 时，在 powerdown 时，VR 不进入 standby 状态
ClkGscRngEn	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 2'b00 时表示 1 分频系数 该值为 2'b01 时表示 2 分频系数 该值为 2'b10 时表示 4 分频系数 该值为 2'b11 时表示 8 分频系数 (注：核的实际频率为 OSC 频率/分频系数 2’bxx 详见 附录D 注释) PAE 的工作频率要大于等于核频率。
CoreFreqDivNum	核分频系数： 该值为 2'b01 时表示 2 分频系数 该值为 2'b10 时表示 4 分频系数 该值为 2'b11 时表示 8 分频系数 (注：核的实际频率为 OSC 频率/分频系数)

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
UsbUdcEn	UsbPhyEn	ScbClkEn	KgeClkEn	RngClkEn	DesClkEn	ScdClkEn	PaeClkEn
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W
1	1	-	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						
ScbClkEn	Scb 时钟使能。0：关闭 ； 1：开启						
KgeClkEn	Kge 时钟使能。 0：关闭 ； 1：开启						
RngClkEn	Rng 时钟使能。 0：关闭 ； 1：开启						
DesClkEn	Des 时钟使能。 0：关闭 ； 1：开启						
ScdClkEn	Scd 时钟使能。 0：关闭 ； 1：开启						
PaeClkEn	Pae 时钟使能。 0：关闭 ； 1：开启						

注意：CGU 模块寄存器访问受 MPU 保护，Flash 中程序不能直接访问，需要调用 ROM 中 BOOT API 来访问。

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

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 CGUCR.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。

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。

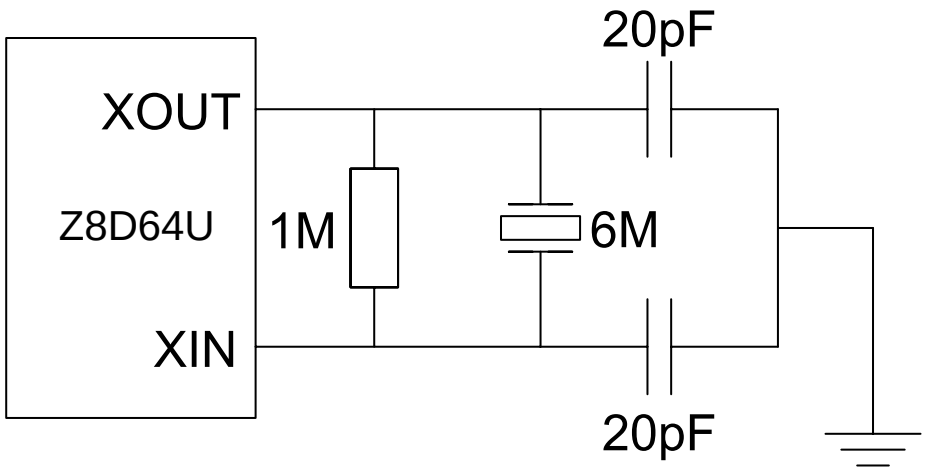
1、在做 Flash 写操作时（编程、擦除、OTP 封口以及所有页操作）时，建议关闭外部停时钟检测使能 CGUCR.CSDE；

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。外接晶体电路如下图所示：



7.6 复位管理

Z8D64U 提供 4 种复位方式：上电复位，WDT 复位，接口复位和软复位。

7.6.1 上电复位

当芯片上电后，引发上电复位过程，该过程将持续 50 个基准源时钟，约 300us。

7.6.2 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。

7.6.3 接口复位

当 IO 接口配置为 SCD 模式时通过，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。

7.6.4 软复位

当给 RCUCR 寄存器与 1 时，将引起系统复位。

第8章 看门狗定时器（WDT）

8.1 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。

8.2 寄存器配置

8.2.1 看门狗控制/状态寄存器

表 8-1 看门狗控制/状态寄存器（WDTCSR－D9H）

WDTCSR		**WDT 看门狗控制/状态寄存器**					**D9H**
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 8-3计数溢出值表）						
WD0	计数溢出置的低位。（可参照 表 8-3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。						

8.2.2 看门狗写控制寄存器

表 8-2 看门狗写控制寄存器（WDTTAP－DAH）

WDTTAP		WDT 看门狗写控制寄存器					DAH
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。					

8.3 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 8-3 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9章 定时器单元(TMU)

9.1 概述

Zi8051 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 0xFFFF 时产生溢出信号。

9.2 寄存器配置

9.1.1 TH0、TL0、TH1、TL1寄存器

表 9-1 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

9.1.2 定时器模式寄存器

表 9-2 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。						

9.1.3 定时器控制寄存器

表 9-3 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10章 复位控制单元(RCU)

10.1 概述

RCU(RESET CONTROL UNIT)模块是 Z8D64U 的复位控制模块，软件可以通过设置 RCU 控制寄存器来控制系统复位。

10.2 寄存器配置

10.2.1RCU控制寄存器

表 10-1 RCU 控制寄存器(RCUCR—C8H)

RCUCR		RCU 控制寄存器					C8H
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	Rev	RsmPdRst	UsbSRst	RcuRst
-	-	-	-	-	R/W	R/W	R/W
0	0	0	0	0	0	0	0
RsmPdRst	软件控制 USB 模式下 Resume 是否唤醒 Power Down，置 1 有效						
UsbSRst	USB 模块复位，置 1 有效						
RcuRst	软件控制系统复位，置 1 有效						

第四部分 安全控制

第11章 DES算法引擎

11.1 概述

DES 模块为 Z8D64U 芯片中的一个模块。在 Z8D64U 芯片中，以 SFR 总线为接口与 CPU 进行数据交换。

CPU 通过 SFR 总线接口操作数据寄存器 DESDR 和密钥寄存器 DESKR 装入数据字节和密钥字节，然后在控制寄存器 DESCR 控制下、完成单 DES 或 3DES 的运算。

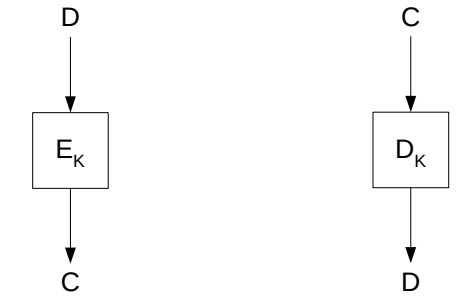
11.1.1功能

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

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. 单密钥 ECB



(a) 加密　　(b) 解密

图 11-1 单密钥 ECB 的运算流程图

2. 密钥 ECB

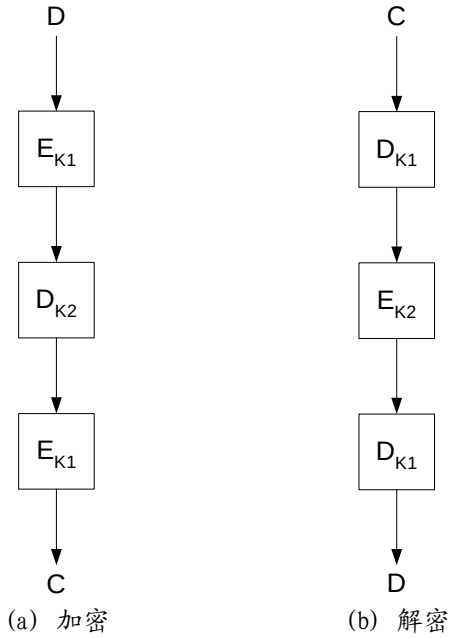


图 11-2 2 密钥 ECB 的运算流程图

3. 密钥 ECB

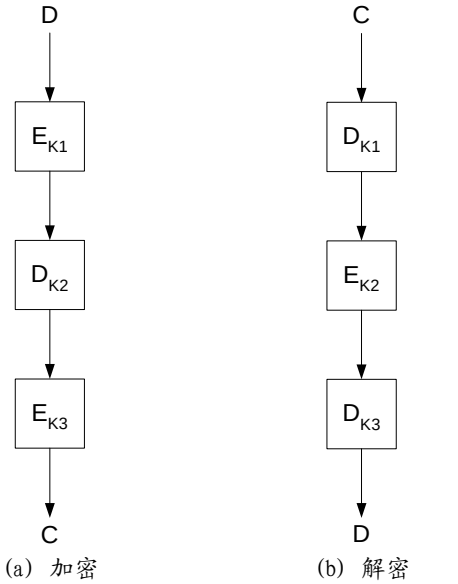


图 11-3 3 密钥 ECB 的运算流程图

4. 单密钥 CBC

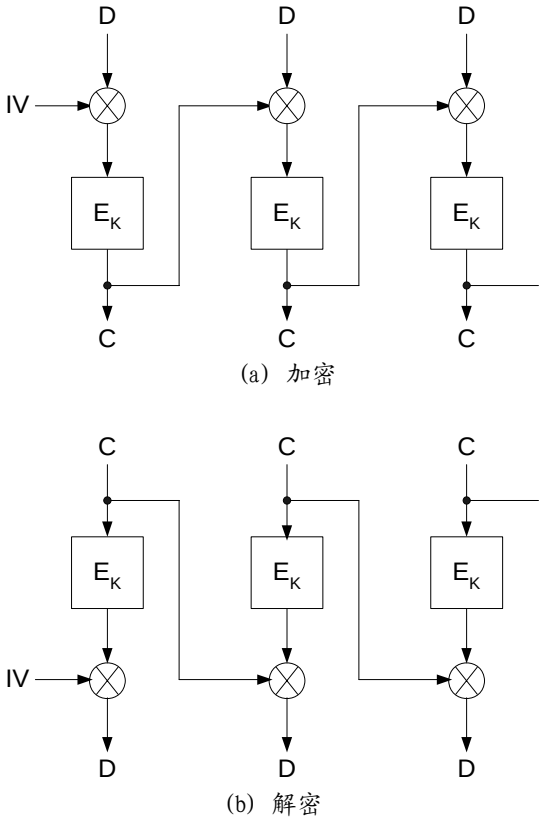
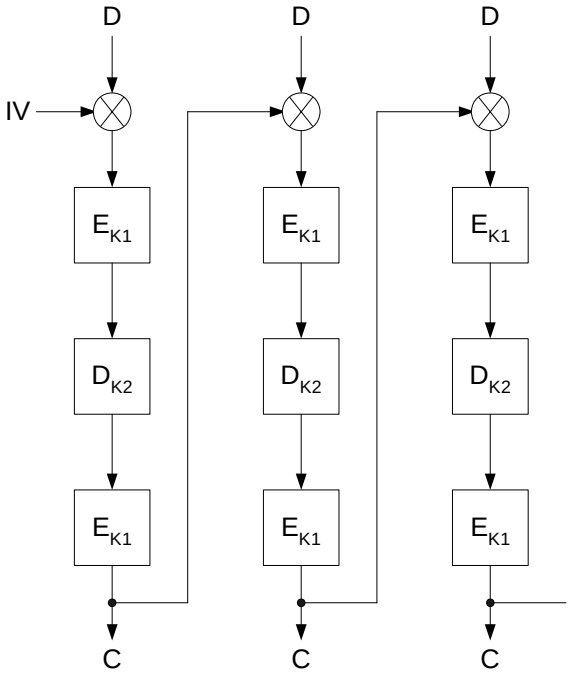
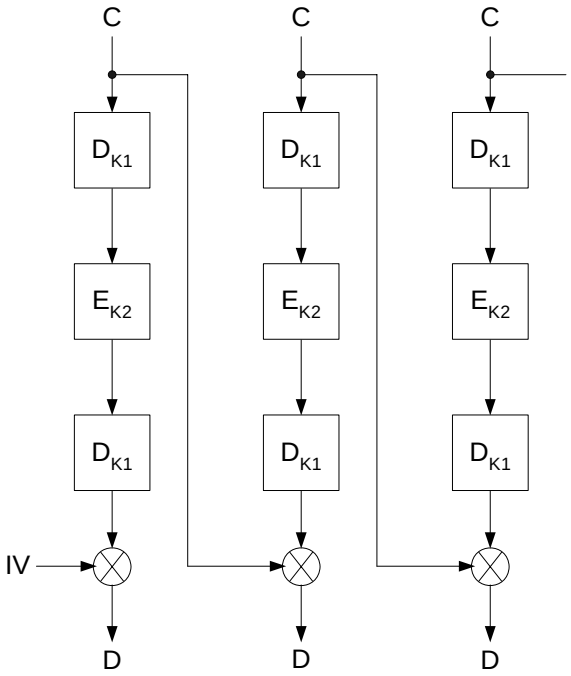


图 11-4 单密钥 CBC 的运算流程图

5. 密钥 CBC



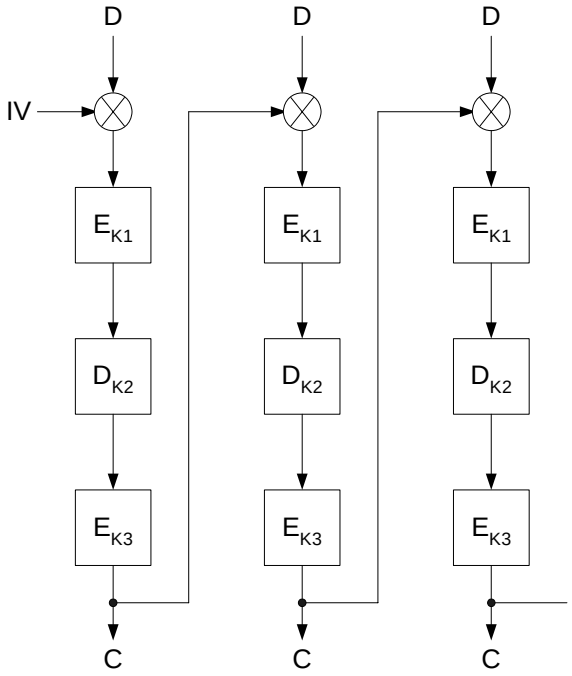
(a) 加密



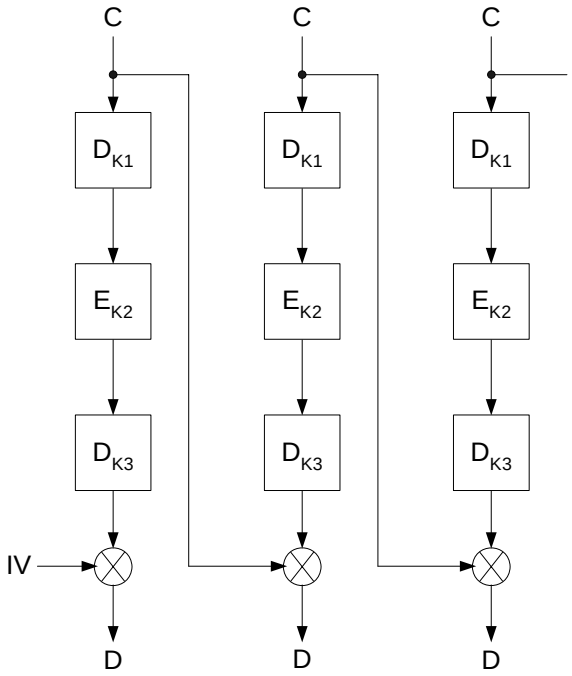
(b) 解密

图 11-5 2 密钥 CBC 的运算流程图

6. 密钥 CBC



(a) 加密



(b) 解密

图 11-6 3 密钥 CBC 的运算流程图

11.3 寄存器配置

CPU 操作 DES 模块的 4 个寄存器，控制 DES 模块的运行和交换数据。

11.3.1数据寄存器

表 11-1 数据寄存器（DESDR－E5H）

DESDR		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					

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 64bits。

写操作时，装入数据到协处理器，由 LSB 开始，连续 8 个字节为一个 BLOCK；读操作时，从协处理器读出数据，由 LSB 开始，连续 8 个字节为一个 BLOCK。初始值为 0。

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]		加解密数据的装入					

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。

装入加解密密钥到协处理器，由 LSB 开始。强制进行读动作时，数据不确定。需用多密钥运算时，各密钥要顺序写入这一统一的密钥寄存器中：

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 控制寄存器（DESCR－E4H）

DESCR		密钥寄存器					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 协处理器，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初始向量寄存器

表 11-4 初始向量寄存器（DESIV－E6H）

DESIV		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。只在 CBC 模式使用，ECB 模式时，即使写入数据，也不起作用。						

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。

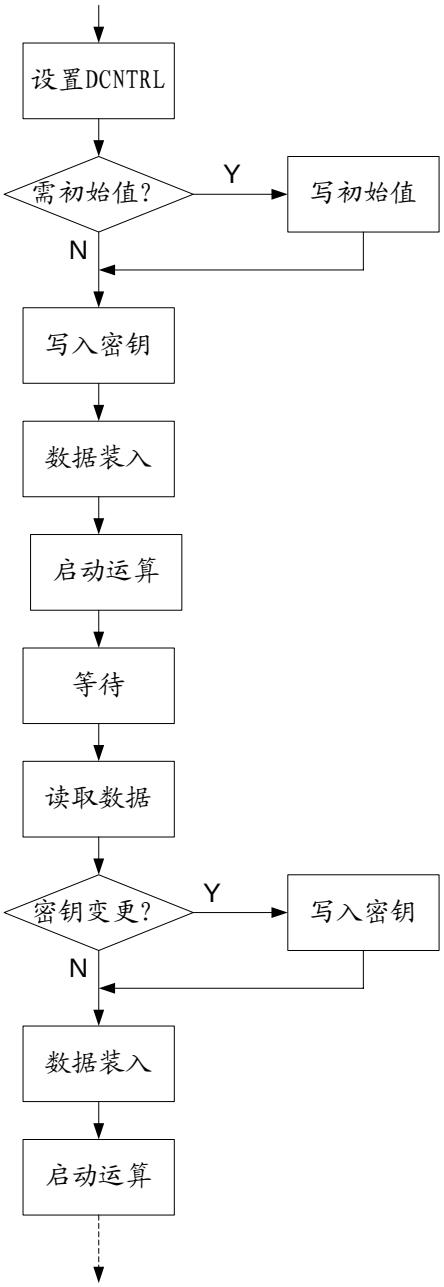


图 11-7 DES 运算流程

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 ECC 等运算。

12.2 寄存器配置

12.2.1PAE控制命令寄存器

表 12-1 PAE 控制命令寄存器(PAECMD－F1H)

PAECMD		密钥寄存器					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				

12.2.2PAE控制寄存器

表 12-2 PAE 控制寄存器(PAECR－F2H)

PAECR		密钥寄存器					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					

12.2.3PAE模长低位寄存器

表 12-3 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 RSANLENH 一起构成模长寄存器。模长寄存器最大值 2048。					

12.2.4PAE模长高位寄存器

表 12-4 PAE 模长高位寄存器（PAENLENH－F4H）

RSANLENH		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 RSANLENL 一起构成模长寄存器。模长寄存器最大值 2048。					

12.2.5PAE幂长低位寄存器

表 12-5 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 3～1024 之间。

12.2.6PAE幂长高位寄存器

表 12-6 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 3～1024 之间。					

12.2.7PAE模式寄存器

表 12-7 PAE 模式寄存器（PAEMOD－F7H）

RSAMOD		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 RSA 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) ECC 运算模式					

12.3 内部RAM配置

Z8D64U 片上核外有 1K Byte RAM 空间专门用于 PAE 运算变量的存放，对应地址空间为 0x0800-0x0bff。在 RSA 运算过程中（RUN＝1），PAERAM 无法读写。

1、 MODREG[1:0]= 2’b00，1024Bit 模乘模幂运算模式运算时，PAERAM 被分为 8 块逻辑地址，每块 128Byte；物理地址为 2 块，DPRAM0 和 DPRAM1。具体分配表如下：

内部 RAM	地址	备注
DPRAM0_0	10'h000-10'h07f	A 值 A*B(modN)结果
DPRAM0_1	10'h080-10'h0ff	内部 E 值
DPRAM0_2	10'h100-10'h17f	内部暂存
DPRAM0_3	10'h180-10'h1ff	内部暂存
DPRAM1_0	10'h200-10'h27f	B 值 A^E(modN)结果
DPRAM1_1	10'h280-10'h2ff	H 值
DPRAM1_2	10'h300-10'h37f	N 值
DPRAM1_3	10'h380-10'h3ff	内部暂存

注意：

- **计算模乘时运算结果放在 DPRAM0_0，计算模幂时 DPRAM1_0 中；**
- **未说明的空间保留，严禁读写。**

2、MODREG[1:0]= 2'b01，2048Bit 模乘运算模式运算时，PAERAM 被分为 4 块逻辑地址，每块 256Byte；物理地址为 2 块，DPRAM0 和 DPRAM1。具体分配表如下：

内部 RAM	地址	备注
DPRAM0_0	10'h000-10'h0ff	A 值 A*B(modN)结果
DPRAM0_1	10'h100-10'h1ff	内部暂存
DPRAM1_0	10'h200-10'h2ff	H 值 B 值
DPRAM1_1	10'h300-10'h3ff	N 值

注意：

- **计算模乘时运算结果放在 DPRAM0_0；**
- **未说明的空间保留，严禁读写。**

3、MODREG[1:0]= 2'b10，1024Bit 乘法运算模式运算时，PAERAM 被分为 4 块逻辑地址，每块 256Byte；物理地址为 2 块，DPRAM0 和 DPRAM1。具体分配表如下：

内部 RAM	地址	备注
DPRAM0_0	10'h000-10'h0ff	A 值 A*B 结果
DPRAM0_1	10'h100-10'h1ff	内部暂存
DPRAM1_0	10'h200-10'h2ff	B 值
DPRAM1_1	10'h300-10'h3ff	内部暂存

注意：

- **计算乘法时运算结果放在 DPRAM0_0；**

- **未说明的空间保留，严禁读写。**

4、MODREG[1:0]= 2’b11，511Bit 内 GF(2^n) ECC 运算模式运算时，PAERAM 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），物理地址为 2 块，DPRAM0 和 DPRAM1。具体分配表如下：

内部 RAM	地址	备注
DPRAM0_0	10’h000-10’h03f	A 值
DPRAM0_1	10’h040-10’h07f	内部暂存
DPRAM0_2	10’h080-10’h0bf	C 低 64Byte,模结果 S
DPRAM0_3	10’h0c0-10’h0ff	C 高 64Byte
DPRAM0_4	10’h100-10’h1ff	未用，可做缓存
DPRAM1_0	10’h200-10’h23f	B 值
DPRAM1_1	10’h240-10’h27f	R 值
DPRAM1_2	10’h280-10’h3ff	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 DPRAM0_2，仅计算多项式乘法时计算结果 DPRAM0_3 中。**
- **DPRAM1_1 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 RSA使用流程

- **MODREG[1:0]=2’b00，1024Bit 模乘模密运算模式**

1)：填写 Rsa 模式寄存器为 0（可省略，缺省为 0）。

2)：填写相应 Rsa 内部寄存器，包括模长 Nlen 寄存器 RSANLEN 和幂长 Elen 寄存器 RSANLEN。

3)：Rsa 内部缓冲区写入待计算的数据。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。

4)：填写计算命令寄存器 RSACMD，写 Ctl 控制寄存器 RSACR 启动运算。

5)：查询 RSACR 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

DPRAM0_0（计算模乘时）和 DPRAM1_0（计算模幂时）中。或者利用 RSAINT 中断处理程序中读取。

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。

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。

- **MODREG[1:0]=2'b01，2048Bit 模乘运算模式**

1)：填写 Rsa 模式寄存器为 1，设置为 2048Bit 模乘运算模式

2)：填写 Rsa 内部模长 Nlen 寄存器 RSANLEN。

3)：写数据到 Rsa 内部缓冲区 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。

4)：填写计算命令寄存器 RSACMD，写 HAB 命令到 Ctl 控制寄存器 RSACR 启动运算。

5)：查询 RSACR 寄存器或者利用 RSAINT 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。

6)：填写计算命令寄存器 RSACMD，写 AB 命令到 Ctl 控制寄存器 RSACR 启动运算。

7)：查询 RSACR 寄存器或者利用 RSAINT 中断判断是否运算结束，读取相应计算结果。运算结果放在 DPRAM0_0 中。

- **MODREG[1:0]=2'b10，1024Bit 乘法运算模式**

1)：填写 Rsa 模式寄存器为 2，设置为 1024Bit 乘法运算模式

2)：填写 Rsa 内部模长 Nlen 寄存器 RSANLEN。（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）

3)：写数据到 Rsa 内部缓冲区 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）

4)：填写计算命令寄存器 RSACMD，写 HAB 命令到 Ctl 控制寄存器 RSACR 启动运算。

5)：查询 RSACR 寄存器或者利用 RSAINT 中断判断是否运算结束，读取相应计算结果。运算结果放在 DPRAM0_0 中。

- **MODREG[1:0]=2'b01，2048Bit 模乘运算模式**

1)：填写 Rsa 模式寄存器为 1，设置为 2048Bit 模乘运算模式

2)：填写 Rsa 内部模长 Nlen 寄存器 RSANLEN。

3)：写数据到 Rsa 内部缓冲区 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。

4)：填写计算命令寄存器 RSACMD，写 HAB 命令到 Ctl 控制寄存器 RSACR 启动运算。

5)：查询 RSACR 寄存器或者利用 RSAINT 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。

6)：填写计算命令寄存器 RSACMD，写 AB 命令到 Ctl 控制寄存器 RSACR 启动运算。

7)：查询 RSACR 寄存器或者利用 RSAINT 中断判断是否运算结束，读取相应计算结果。运算结果放在 DPRAM0_0 中。

- **MODREG[1:0]=2'b11，512Bit 内 GF(2^n) ECC 多项式运算模式**

1)：填写模式寄存器为 3，设置为 512Bit 内 GF(2^n) ECC 多项式运算模式

2)：填写内部模长 Nlen 寄存器 RSANLEN。

3)：填写相关数据到内部缓冲区 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。

4)：填写计算命令寄存器 RSACMD，启动运算。

5)：查询 RSACR 寄存器或者利用 RSAINT 中断判断是否运算结束，读取相应计算结果。运算结果放在 DPRAM0_2/3 中。

第13章 随机数生成器（RNG）

13.1 概述

Z8D64U 内部有 1 个 32 位的全硬件随机数发生器。随机数发生器模块由控制寄存器 RNGCR、数据寄存器 RNGDATA 组成。

13.2 特性

支持真随机、伪随机两种模式。

伪随机的种子可由真随机产生或者由外部输入。

每产生 8bit 就存入 RNGDATA 寄存器，并设置 RNGF 标志或产生中断通知 CPU 读取。

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

13.3 寄存器配置

RNG 模块提供两个 8 位寄存器来实现与 CPU 接口，CPU 操作这两个寄存器，控制 RNG 模块并读取随机数据。

13.3.1COMMAND寄存器（片选）

表 13-1 COMMAND 寄存器（片选）(COMMAND—D5H)

COMMAND		**COMMAND 寄存器（片选）**					**D5H**
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IPSEL	Rev	Rev	SeedGM	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
IPSEL	1：表示选中 RNGIP 真随机模块输出有效，地址映射到 RNGIP，R*有效 0：表示选中 RNGIP 伪随机模块输出有效，地址映射到 RNGIP，R*无效						
SeedGM	软件可作读操作和写 1 操作；种子赋值完毕，硬件自动清 0 或 Reset 清 0。						
Reset	送给国密 IP 的 Reset 信号，高有效.						

RNF	1：IPSEL 为 1 时，表示从 RNGIP 中预读 NUM 寄存器的值 0：无操作。
RDF	1：IPSEL 为 1 时，表示从 RNGIP 中预读 DATA 寄存器的值 0：无操作。
RMF	1：IPSEL 为 1 时，表示从 RNGIP 中预读 MODE 寄存器的值 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（IP中未被读走的数组）真随机数组（32 Bits/组）的个数；最大值为 8'h8。（8'hxx详见 附录D 注释）						

13.3.3RNGMODE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），则模拟电路使用 2^N 分频总线时钟工作						
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						
MODE[1:0]	工作模式：00——保留 01——正常模式 10——高速模式 11——高速模式						

由于国密 IP 的写读时序需要两个机器周期完成，这里要求：

在进行 RNGMODE 进行写操作之后，必须再等一个机器周期完成，才可对 RNGIP 的三组寄存器进行读操作；读操作使用软件策略进行预取，需配置 COMMAND 的相应标志位完成预取，使用 SFR 读时无需等待一个机器周期。

13.3.4RNG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							
RNGDATA[7:0]	产生的真随机数，供 CPU 读取						

13.3.5RNG控制寄存器

表 13-5 RNG 控制寄存器（RNGCR—D6H）

RNGCR		RNG 控制寄存器					D6H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RNGF	Rev	Rev	TESTMODE	TESTVALUE	PNSEL	OSCEN	RUN
R/W	-	-	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
RNGF	随机数有效标志，硬件置 1，软件清除，清除后才继续产生下一随机数						
TESTMODE	测试模式选择。 当＝1 为测试模式 当＝0 为工作模式						
TESTVALUE	测试值。在测试模式下（TESTMODE＝1）， 当＝1，读 RNGDATA 的值为 8’hFF 当＝0，读 RNGDATA 的值为 8’h0						
PNSEL	真、伪随机数选择。 当＝1，产生伪随机数 当＝0，产生真随机数						

OSCEN	模拟电路 RNG_IP 模块工作使能控制。 当 OSCEN =0 时，模拟电路 RNG_IP 模块不工作； 当 OSCEN =1 时，模拟电路 RNG_IP 模块正常工作。 当 PNSEL＝1 且 OSCEN =1 时，伪随机数发生器的种子为真随机数由 RNG_IP 模块产生； 当 PNSEL＝1 且 OSCEN =0 时，完全伪随机。
RUN	模块启动信号 =0，RNG 停止工作； =1，RNG 正常工作。

13.3.6RNG种子数据寄存器

表 13-6 RNG 种子数据寄存器（RNGSEED—D7H）

RNGSEED		RNG 种子数据寄存器					D7H
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RNGSEED [7:0]							
W							
RNGSEED[7:0]	8bit 种子数据寄存器，该寄存器只写。当写入一个值后，模块中 59bit 的用户输入种子寄存器就会左移 8 位，并将写入的这个值写入低 8 位数据。连写 8 次种子数据寄存器后，就可以配置好用户设置的种子用于完全伪随机产生模式。（注意，第一次写入数据的高 5 位无效。）						

13.4 软件操作流程

13.4.1产生32位伪随机数（完全伪随机）：

13.4.1.1 操作流程

以下出现的 UserSeed 表示用户准备输入的 59bit 种子。

- 配置 COMMAND.IPSEL＝0（默认值）
- 往 RNG 种子数据寄存器写入{5’h0, UserSeed[58:56]}
- 往 RNG 种子数据寄存器写入 UserSeed[55:48]
- 往 RNG 种子数据寄存器写入 UserSeed[47:40]
- 往 RNG 种子数据寄存器写入 UserSeed[39:32]
- 往 RNG 种子数据寄存器写入 UserSeed[31:24]

- 往 RNG 种子数据寄存器写入 UserSeed[23:16]
- 往 RNG 种子数据寄存器写入 UserSeed[15:8]
- 往 RNG 种子数据寄存器写入 UserSeed[7:0]
- 配置 RNGCR 寄存器，置 RNGCR.PNSEL=1, RNGCR.OSCEN=0, RNGCR.RUN=1（开始产生第 1 个字节）
- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 1 个字节
- 清 RNGCR. RNGF =0（开始产生第 2 个字节）
- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 2 个字节
- 清 RNGCR. RNGF =0（开始产生第 3 个字节）
- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 3 个字节
- 清 RNGCR. RNGF =0（开始产生第 4 个字节）
- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 4 个字节
- 同时清 RNGCR.RNGF=0, RNGCR.RUN=0（结束）

13.4.1.2 示例程序

```
    mov COMMAND,#00h            ;COMMAND[7]=0
    mov R3,  #00h          ;R3 Repeat Counter
    mov RNGCR,  #03h            ;True Random 7b
    mov P3,  #7bh
    lcall Seed0
    mov R3,  #00h
    mov RNGCR,  #05h            ;Generate Pure      PN0 79
    mov P3,  #79h
    lcall RngMove2
    mov R3,  #00h
Seed0:
    mov RNGDR,  #00h            ;Seed Input0
    SeedI0:
    mov RNGDR,  #00h
    inc   R3
    CJNE     R3,  #07h,     SeedI0
    Ret
RngMove2:
    mov RNGCR,  #05h
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,  #7eh          ; Byte 1
    anl   RNGCR,  #7fh
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,  #7dh          ; Byte 2
```



```
    anl   RNGCR, #7fh
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,   #7ch          ; Byte 3
    anl   RNGCR, #7fh
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,   #70h          ; Byte 4 7c finish 4 Bytes once
    mov RNGCR, #00h
    inc   R3
    mov A,    R3
    mov P3,   #60h          ;Print Repeat Time 70
    CJNE      R3,  #0ah,    RngMove2        ;Repeat
    ret
WaitRngf:
    mov A,    RNGCR
    anl   A,  #80h          ;CR[7]=1
    jz    WaitRngf
    ret
```

13.4.2产生32bit真随机数

13.4.2.1 操作流程

- 配置 COMMAND.IPSEL = 1，则表示选中 RNGIP 模块寄存器有效，COMMAND 控制读数据
- 配置 RNGIPMODE 寄存器，选择工作模式，现在的 IP 只支持正常模式，即 MODE＝8’h01；
- 配置 COMMAND. RNF = 1，完成从国密 IP 的 NUM 寄存器预取数据到 RNGIPNUM 中
- 从 RNGIPNUM 读取数据，查看缓存数据剩余个数，最多存储 8 个字节
- 配置 COMMAND.RNF =0 , COMMAND. RDF＝1
- 从 RNGIPDATA 读取第 1 个字节
- 再次查看 NUM，再次配置预读 RDF，每次配置只能取出一个 RNG 数（8bit）
- 从 RNGIPDATA 读取第 2 个字节
- ……
- 从 RNGIPDATA 读取第 n 个字节
- 配置 COMMAND.RDF=0，停止读数

13.4.2.2 示例程序

```
    RngIP_GM:
    mov       COMMAND,#88h        ;COMMAND[7]=1, choose GM IP
    nop
    nop
    mov COMMAND,#80h
    mov RNGCR, #01h
    Zero:
    mov COMMAND,#84h
    mov A,    RNGNUM
```



```
cjne A,   #08h,     Zero
mov P3,  #0ffh
mov R3,  #02fh
Delt:
mov COMMAND,#82h
mov A,   RNGDR
mov P3,  #0feh
dec  R3
mov P3,  #00h
mov      A,   R3
jz   Fin
mov COMMAND,#84h
mov A,   RNGNUM
jnz  Delt
Fin:
mov COMMAND,#84h
mov A,   RNGIPCR
mov COMMAND,#80h
ret
```

13.4.3产生32位伪随机数（自动以国密IP的真随机数做种子）

13.4.3.1 操作流程

- 配置 COMMAND.IPSEL = 1，则表示选中 RNGIP 模块寄存器有效，COMMAND 控制读数据
- 配置 RNGIPMODE 寄存器，选择工作模式，现在的 IP 只支持正常模式，即 MODE＝8’h01；
- 配置 COMMAND. RNF = 1，完成从国密 IP 的 NUM 寄存器预取数据到 RNGIPNUM 中
- 从 RNGIPNUM 读取数据，查看缓存数据剩余个数，直到其值不为 0，进行下一步
- 配置 COMMAND.RNF =0 , COMMAND. RDF＝1，预读种子的高位
- 配置 COMMAND＝8’h10，COMMAND.IPSEL = 0, COMMAND.RNF =0 , COMMAND. SeedGM＝1，同时读种子的低位；
- 查询 COMMAND.SeedGM 是否清零，如果清零继续下一步，否则继续查询
- 配置 RNGCR 寄存器，置 RNGCR.PNSEL=1, RNGCR.OSCEN=0, RNGCR.RUN=1（开始产生第 1 个字节）
- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 1 个字节
- 清 RNGCR. RNGF =0（开始产生第 2 个字节）
- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 2 个字节
- 清 RNGCR. RNGF =0（开始产生第 3 个字节）
- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 3 个字节
- 清 RNGCR. RNGF =0（开始产生第 4 个字节）

- 等待完成标志 RNGCR.RNGF=1
- 从 RNGDATA 读取第 4 个字节
- 同时清 RNGCR.RNGF=0, RNGCR.RUN=0（结束）

13.4.3.2 示例程序

```
    mov COMMAND,#00h
    mov RNGNUM,#00h
    mov R3,  #00h
    ;lcall      delays
    lcall Seed2
    mov R3,  #00h
    mov RNGCR,  #05h          ;Generate Pure      PN1 78
    mov P3,  #78h
    lcall RngMove2
    mov R3,  #00h
Seed2:                              ;  Automated Get Seed from GMIP
    mov COMMAND,#80h
    mov RNGCR,  #01h          ;RNGCR's Addr D6 is Duplicated As IPMODE
    Zer:
    mov COMMAND,#84h
    mov A,    RNGNUM
    cjne A,   #08h,     Zer
    mov COMMAND,#82h              ;Seed_higher    from GMIP
    mov COMMAND,#10h              ;Seed_lower    from GMIP, & Start the Analog IP
    SeedI2:
    mov A,    COMMAND
    jnz   SeedI2
    ret
RngMove2:
    mov RNGCR,  #05h      ;Generate PN code From UserSeed
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,  #7eh         ; Byte 1
    anl   RNGCR,  #7fh
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,  #7dh         ; Byte 2
    anl   RNGCR,  #7fh
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,  #7ch         ; Byte 3
    anl   RNGCR,  #7fh
    lcall WaitRngf
    mov A,    RNGDR
    mov P3,  #70h         ; Byte 4 7c finish 4 Bytes once
    mov RNGCR,  #00h
    inc   R3
    mov A,    R3
    mov P3,  #60h         ;Print Repeat Time 70
    CJNE      R3,  #01h,      RngMove2          ;Repeat
    ret
WaitRngf:
    mov A,    RNGCR
    anl   A,    #80h          ;CR[7]=1
```



```
jz      WaitRngf
ret
```

第14章 密钥生成引擎（KGE）

14.1 概述

KGE（Key Generate Engine）密钥对生成引擎硬件自动生成公钥算法中需要的密钥对。

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：

```
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：

```
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 安全防护单元（SEC）

15.1 概述

SEC 模块完成外部高低电压检测和频率异常检测，将当前状态通知 CPU。

SEC 模块工作后，将检测到的状态存入 SECSR 寄存器，由 CPU 查询获取状态或产生中断通知 CPU。

15.2 寄存器配置

15.2.1 SEC控制寄存器

表 15-1 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；						

15.2.2 SEC状态寄存器

表 15-2 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，高频率告警就不会再产生
- 低频率告警产生后，如果不清除低频告警标志，低频率告警就不会再产生

第五部分 外部接口控制

第16章 IO控制器(IOM)

16.1 概述

IOM(IO Manage)模块控制 4 个 IO 口的功能。

16.2 寄存器配置

16.2.1 IO模式控制寄存器

表 16-1 IO 模式控制寄存器(IOMCR—8EH)

IOMCR		**IO 模式控制寄存器**					**8EH**
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	ResUp	RstLv	RstEn	RstMd	UART
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

ResUp	ResUp_Pad 输出控制，用于 USB D+/D- 信号线上拉
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。
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
RstMd	非 SCD 模式下 RST 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。
UART	UART 模式选择： 0：非 UART 模式 1：UART 模式，在该模式下 TX 和 SIO 引脚作为 UART 的发送和接收引脚

注意：IOMCR 寄存器访问受 MPU 保护，Flash 中程序不可以直接访问，需要调用 ROM 中 BOOT API 来访问。

16.2.2 GPIO方向控制寄存器

表 16-2 GPIO 控制寄存器(GPIOCR—8FH)

GPIOCR		GPIO 方向控制寄存器					8FH
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	Rev	Rev	GPIODIR[3:0]			
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
GPIODIR[3:0]		4 位 GPIO 输入输出方向控制。 0 为输入，1 为输出。					

16.2.3 GPIOCR对应的端口和访问控制方式

表 16-3 GPIO 的访问控制

GPIO	Pad	方向控制	访问
0	ClkScd	GPIOCR [0]	P2_0
1	ScdRst	GPIOCR [1]	P2_1
2	Sio	GPIOCR [2]	P2_2
3	Tx	GPIOCR [3]	P2_3

16.3 接口及GPIO的配置

16.3.1接口模式

- SCD：此时只有 SCD 接口工作
- USB＋UART：此时 USB 与 UART 同时工作，其中 USB 包括 USB 全速模式(USB FS)和 USB 低速模式(USB LS)

注：接口模式在芯片封装时就已经确定，为以上两种接口模式中的一种，不能通过软件来修改

16.3.2各接口模式下的GPIO

在SCD，或者UART，或者USB FS，或者USB LS模式下，通过对IOMCR[3:1]位进行设置，可以控制RST引脚的功能，具体使用可以查看 表 16-4：

表 16-4 各个模式下设置 RST 引脚功能

地址	8EH				
位数			IOMCR [3]	IOMCR [2]	IOMCR [1]
位			RstLv	RstEn	RstMd
用法	模式	作用	相应值	相应值	相应值
	SCD	进行系统复位	x	1	x
		产生复位中断	x	0	x
	UART	进行系统低电平复位	0	1	0
		进行系统高电平复位	1	1	0
		低电平唤醒 Power Down 状态	0	x	0
		高电平唤醒 Power Down 状态	1	x	0
		产生由 RST 下降沿触发的复位中断	0	0	0
		产生由 RST 上升沿触发的复位中断	1	0	0
		RST 作为 GPIO，映射为 P2_1	x	x	1
	USB FS USB LS	进行系统低电平复位	0	1	0
		进行系统高电平复位	1	1	0
		低电平唤醒 Power Down 状态	0	x	0
		高电平唤醒 Power Down 状态	1	x	0
		产生由 RST 下降沿触发的复位中断	0	0	0
		产生由 RST 上升沿触发的复位中断	1	0	0
		RST 作为 GPIO，映射为 P2_1	x	x	1

注意事项：

表中的 x 表示不用关心其值。

如果是UART或者USB（USBFS, USBLS），又或者UART和USB两者共存的模式，那么还必须根据需要对GPIOCR的状态位进行设置，设置方法如下 表 16-5所示：

表 16-5 UART 和 USB 模式下的 GPIO

地址	8FH							
位数	模式状态				GPIOCR [3]	GPIOCR [2]	GPIOCR [1]	GPIOCR [0]
用法	作为 GPIO 输入							
	UART	USB	输出	输入	相应值	相应值	相应值	相应值
	1	x	P2[0]作为 GPIO 输入使用		x	x	x	0
	x	1			x	x	x	0
	1	x	P2[1]作为 GPIO 输入使用		x	x	0	x
	x	1			x	x	0	x
	1	x	P2[2]作为 GPIO 输入使用		x	0	x	x
	x	1			x	0	x	x
	1	x	P2[3]作为 GPIO 输入使用		0	x	x	x
	x	1			0	x	x	x
	作为 GPIO 输出							
	1	x	P2[0]作为 GPIO 输出使用		x	x	x	1

	x	1		x	x	x	1
	1	x	P2[1]作为 GPIO 输出使用	x	x	1	x
	x	1		x	x	1	x
	1	x	P2[2]作为 GPIO 输出使用	x	1	x	x
	x	1		x	1	x	
	1	x	P2[3]作为 GPIO 输出使用	1	x	x	x
	x	1		1	x	x	x

第17章 UART接口

17.1 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。

17.1.1功能特性

- 提供标准的异步通讯位（起始位、奇偶位和停止位）
  - 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

17.2 基本原理

17.2.1 传送格式

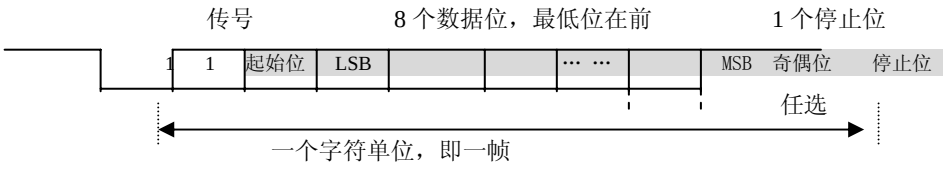


图 17-1 UART 发送/接收数据格式

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　数据传输之后是可选择的奇偶位发送或接收。奇偶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有无和校验类型必须一致。

5）停止传输　奇偶位（选择有奇偶）或数据位(选择无奇偶)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。

17.2.2 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；

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。

17.2.3 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。

17.3 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。

17.3.1 UART中断状态寄存器

表 17-1 UART 中断状态寄存器（UARTISR－C0H）

UARTISR		UART 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	ORER	TXEND	TRE	STOPBIT_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。对于 T= 0，有校验错误的接收数据将不会写到 FIFO 上。 当 UARTCR.TRS = 1, UARTISR.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。
STOPBIT_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

17.3.2 UART中断允许寄存器

表 17-2 UART 中断允许寄存器（UARTIER－C1H）

UARTIER		UART 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，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						

17.3.3 UART状态与控制寄存器

表 17-3 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

17.3.4 UART数据寄存器

表 17-4 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。

17.3.5 UART波特率参数低位寄存器

表 17-5 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。						

17.3.6 UART波特率参数高位/ETU计数寄存器

表 17-6 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							
UARTECR [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)。						
UARTBPRH[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。						

17.4 UART操作

17.4.1 初始化

- 置 UARTIER=0
- 在接收前，清空 Receive-FIFO(UARTCR. FLUSH =1)

- 清 UARTISR 中的状态标志
- 配置 UARTBPRH 和 UARTBPRL
- 设置或清零 UARTCR.TRS 位
- 中断使能（UARTIER 相应位置 1）

17.4.2 发送字节步骤

- 按照 17.4.1初始化中描述的步骤初始化
- 置 UARTCR.TRS=1
- 写入第 1 个字节到 UARTTDR (开始发送第 1 个字节)
- 等待发送第 1 个字节完成标志 UARTISR.TXEND=1（或等待 UARTInt 中断）
- 清 UARTISR.TXEND=0
- 写入第 2 个字节到 UARTTDR（开始发送第 2 个字节）
- 等待发送第 2 个字节完成标志 UARTISR.TXEND= 1（或等待 UARTInt 中断）

……

……

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

（发送结束）

17.4.3 接收字节步骤

- 初始化：置 UARTCR.TRS=0
- 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。

- 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18章 USB接口

18.1 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》。

18.1.1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

18.2 寄存器配置

18.2.1 USB设备配置寄存器

表 18-1 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	Rev	EP4VLD	EP3VLD	EP2VLD	EP1VLD	Rev	DEVMOD
R	R	R/W	R/W	R/W	R/W	-	R
0	0	0	0	0	0	0	0

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。
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。

18.2.2 USB端点控制状态寄存器

表 18-2 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 为空，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；当成功接收完成 HOST 下传的一包数据返回 ACK 之后，硬件将此位置 1，CPU 写 0 会被屏蔽。注意在清零之后此位有可能迅速变 1，表明已经接收到新的数据包。						
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；当 HOST 成功接收到该数据包返回 ACK 之后，硬件将此位置 1，CPU 写 0 会被屏蔽。注意在清零之后此位有可能迅速变 1，表明已经有新的空缓冲区 FIFO 可写入数据。						
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 为空，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；当成功接收完成 HOST 下传的一包数据返回 ACK 之后，硬件将此位置 1，CPU 写 0 会被屏蔽。						
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；当 HOST 成功接收到该数据包返回 ACK 之后，硬件将此位置 1，CPU 写 0 会被屏蔽。						

EP0ORDY	端点 0 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 为空，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；当成功接收完成 HOST 下传的一包数据返回 ACK 之后，硬件将此位置 1，CPU 写 0 会被屏蔽。注意在清零之后此位有可能迅速变 1，表明已经接收到新的数据包。
EP0IRDY	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；当 HOST 成功接收到该数据包返回 ACK 之后（即 Ep0In 中断），硬件将此位置 1，CPU 写 0 会被屏蔽。注意在清零之后此位有可能迅速变 1，表明已经有新的空缓冲区 FIFO 可写入数据。
EP0SRDY	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 没有接收好数据，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，CPU 写 0 会被屏蔽。注意在清零之后此位有可能迅速变 1，表明已经接收到新的数据包。

18.2.3 端点控制状态寄存器

表 18-3 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
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 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 18-4 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 18-5 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 18-6 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 18-7 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
未用	未用	未用	未用	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 18-8 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 18-9 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 18-10 端点 4 有效数据长度寄存器（EP4OUTBCR—AFH）

EP4OUTBCR		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	EP4OUTBCR[5:0]					
R	R	R					
0	0	0					
EP4OUTBCR[5:0]	当前端点 4 OUT 传输缓冲区 FIFO 有效数据长度，对 CPU 只读。						

18.2.4 USB设备中断寄存器

表 18-11 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 18-12 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。1：有 RSM 中断请求；0：无 RSM 中断请求。CPU 写 1 清零。						
SUSP	USB Suspend 中断请求/状态位。1：有 SUSP 中断请求；0：无 SUSP 中断请求。CPU 写 1 清零。						
SOF	USB 帧头标识中断请求/状态位。1：有 SOF 中断请求；0：无 SOF 中断请求。CPU 写 1 清零。						
URES	USB 设备复位中断请求/状态位。1：有 URES 中断请求；0：无 URES 中断请求。CPU 写 1 清零。						

表 18-13 USB 端点中断使能寄存器（USBEPIE—BBH）

USBEPIE		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	EP4OUT	EP3IN	EP2OUT	EP1IN	EP0OUT	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
EP4OUT	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。						
EP2OUT	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。
EP0OUT	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 18-14 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 18-15 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 18-16 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。1：有 EP3TK 中断请求/状态位；0：无 EP3TK 中断请求/状态位。CPU 写 1 清零。						
EP1TK	端点 1 IN 令牌中断请求/状态位。1：有 EP1TK 中断请求/状态位；0：无 EP1TK 中断请求/状态位。CPU 写 1 清零。						
EP0ITK	端点 0 IN 令牌中断请求/状态位。1：有 EP0ITK 中断请求/状态位；0：无 EP0ITK 中断请求/状态位。CPU 写 1 清零。						
SUTK	端点 0 Setup 包令牌中断请求/状态位。1：有 SUTK 中断请求/状态位；0：无 SUTK 中断请求/状态位。CPU 写 1 清零。						

表 18-17 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。						
EP0IERR	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 18-18 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	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 读空错误中断请求/状态位。1：有 EP4ERR 中断请求/状态位；0：无 EP4ERR 中断请求/状态位。CPU 写 1 清零。						
EP3ERR	端点 3 IN FIFO 写满错误中断请求/状态位。1：有 EP3ERR 中断请求/状态位；0：无 EP3ERR 中断请求/状态位。CPU 写 1 清零。						
EP2ERR	端点 2 OUT FIFO 读空错误中断请求/状态位。1：有 EP2ERR 中断请求/状态位；0：无 EP2ERR 中断请求/状态位。CPU 写 1 清零。						
EP1ERR	端点 1 IN FIFO 写满错误中断请求/状态位。1：有 EP1ERR 中断请求/状态位；0：无 EP1ERR 中断请求/状态位。CPU 写 1 清零。						

EP0OERR	端点 0 OUT FIFO 读空错误中断请求/状态位。1：有 EP0OERR 中断请求/状态位；0：无 EP0OERR 中断请求/状态位。CPU 写 1 清零。
EP0IERR	端点 0 IN FIFO 写满错误中断请求/状态位。1：有 EP0IERR 中断请求/状态位；0：无 EP0IERR 中断请求/状态位。CPU 写 1 清零。
SUERR	端点 0 Setup 包错误中断请求/状态位。1：有 SUERR 中断请求/状态位；0：无 SUERR 中断请求/状态位。CPU 写 1 清零。

表 18-19 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 18-20 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。1：有 SUDW 中断请求/状态位；0：无 SUDW 中断请求/状态位。CPU 写 1 清零。
EP4R	端点 4 OUT FIFO 读空后清 RDY 出错中断请求/状态位。1：有 EP4R 中断请求/状态位；0：无 EP4R 中断请求/状态位。CPU 写 1 清零。
EP3W	端点 3 IN FIFO 写满后清 RDY 出错中断请求/状态位。1：有 EP3W 中断请求/状态位；0：无 EP3W 中断请求/状态位。CPU 写 1 清零。
EP2R	端点 2 OUT FIFO 读空后清 RDY 出错中断请求/状态位。1：有 EP2R 中断请求/状态位；0：无 EP2R 中断请求/状态位。CPU 写 1 清零。
EP1W	端点 1 IN FIFO 写满后清 RDY 出错中断请求/状态位。1：有 EP1W 中断请求/状态位；0：无 EP1W 中断请求/状态位。CPU 写 1 清零。
EP0R	端点 0 OUT FIFO 读空后清 ORDY 出错中断请求/状态位。1：有 EP0R 中断请求/状态位；0：无 EP0R 中断请求/状态位。CPU 写 1 清零。
EP0W	端点 0 IN FIFO 写满后清 IRDY 出错中断请求/状态位。1：有 EP0W 中断请求/状态位；0：无 EP0W 中断请求/状态位。CPU 写 1 清零。
SUDR	端点 0 Setup 包读空后清 SRDY 出错中断请求/状态位。1：有 SUDR 中断请求/状态位；0：无 SUDR 中断请求/状态位。CPU 写 1 清零。

18.2.5 端点数据FIFO数寄存器

表 18-21 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 18-22 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 18-23 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。						

表 18-24 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 18-25 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 18-26 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 18-27 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。						

18.2.6 UFM模块控制/状态寄存器

表 18-28 UFM 模块控制/状态寄存器(UFMSR—9FH)

UFMSR		UFM 模块控制/状态寄存器					9FH
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rev	Rev	CLRFEN	EP[2:0]			CLRF	SRDY
R	R	R/W	R			R	R
0	0	0	0	0	0	0	0
CLRFEN	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。

18.3 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。

18.3.1初始化流程：

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

18.3.2Setup包软件处理流程

- 预开启 EPIR.SUDAV 中断使能；
- 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；
- 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）；
- 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）；
- 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；
- 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。

18.3.3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：

- 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。
- 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）：

- CPU 写 EP0CSR.CLR 归零当前 FIFO 写指针；
- 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；
- 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。

18.3.4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：

- 开启 EPIE.EP0OUT 或者采用查询方式
- 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；
- 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。

18.3.5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。

- CPU 写 EP1CSR.CLR 归零当前 FIFO 写指针；
- 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；
- 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。

18.3.6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。

- 开启 EPIE.EP2OUT 中断或者采用查询方式；
- 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；
- 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。

18.3.7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。

- CPU 写 EP3CSR.CLR 归零当前 FIFO 写指针；
- 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；
- 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。

18.3.8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。

- 开启 EPIE.EP4OUT 中断或者采用查询方式；
- 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；
- 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。

第五部分 电气特性

第19章 电气特性

19.1 最高绝对限额

此部分提供 Z8D64U 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 19-1 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	2.7	5.5	V
VESD	最大 ESD 电压，HBM	-	-	V

19.2 操作条件

此部分显示Z8D64U芯片的电压、频率以及温度特性。

表 19-2 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	-	-	-	°C
V_{Vcc}	电源电压	4.5	5.0	5.5	V
I_{Vcc}	Frequency:10Mhz, Vcc=5v	-	-	-	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	-	-	33	MHz
B_{SCD}	ISO7816 接口最高通讯带宽	-	-	-	bps
Cin	输入电容	-	-	-	PF

19.3 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 19-3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 19-3 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_{Vcc}+0.3	V
V_{IL}	输入低电压，所有标准输入和双向端口	-0.3		1.2Vcc	V
ILIH	输入漏电流（输入高电压）			+5	uA
ILIL	输入漏电流（输入低电压）		100	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	VDD-1.0			V
V_{OL}	输出低电压，所有标准输出和双向端口			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=V_{OH})			+5	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=V_{OH})		-100	-250	MA
IO 上下拉电阻					
R_{PU}	内置上拉电阻	-	50	-	KΩ
R_{PD}	内置下拉电阻	-	-	-	KΩ

19.4 AC参数

一个引脚地AC特性包括输入以及输出电容，它决定了外部驱动或其他驱动的负载分析。AC特性包括一个 因子，它决定不同负载情况下的AC时序的快慢。表 19-4显示了高低电压输入、输出以及IO引脚情况下的AC操作条件，所有的AC参数值在整个温度范围内有效。

表 19-4 标准输输入、输出以及双向端口 AC 操作条件

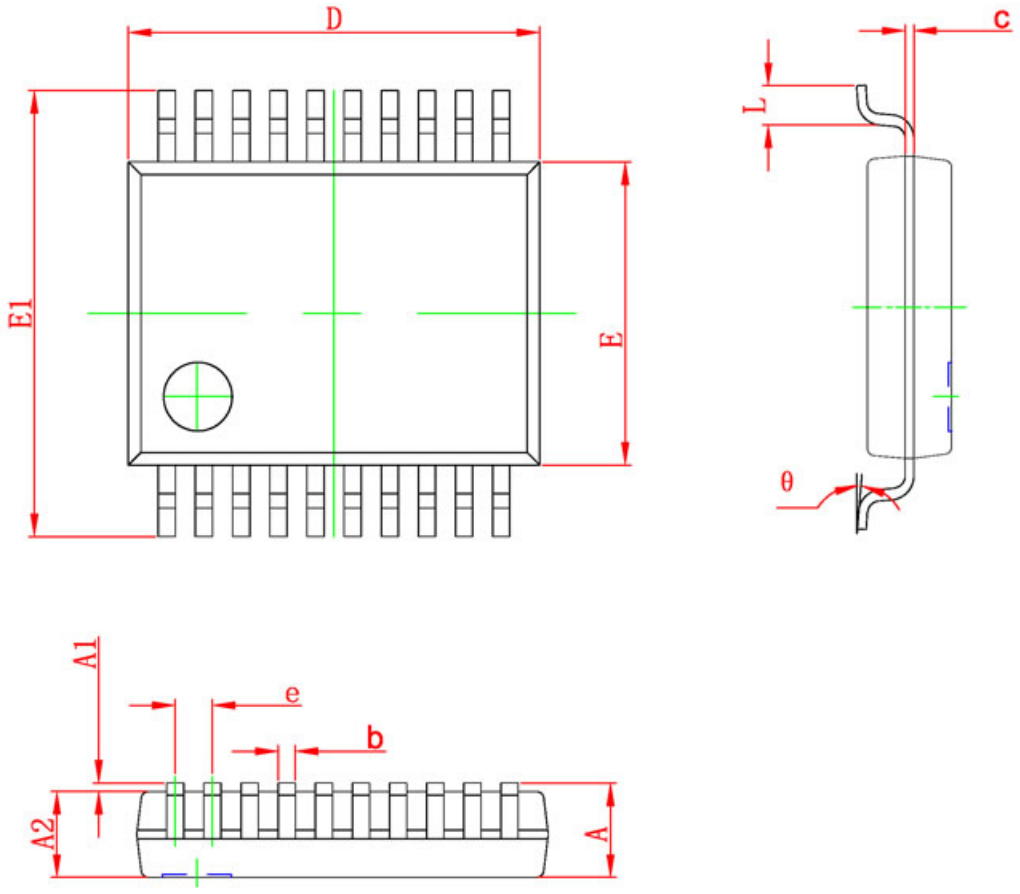
符号	描述	最小	典型	最大	单位
C_{IN}	输入电容，所有标准输入以及双向端口	-	-	-	pF

C_{OUT_H}	输出电容，所有标准高强度输出以及双向端口	-		-	pF

注：在此负载范围内AC特性能够保证，所有的测试在50pf的条件下。

19.5 封装

SSOP20(209mil) PACKAGE OUTLINE DIMENSIONS



Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A		1.730		0.068
A1	0.050	0.230	0.002	0.009
A2	1.400	1.600	0.055	0.063
b	0.220	0.380	0.009	0.015
c	0.090	0.250	0.004	0.010
D	7.000	7.400	0.276	0.291
E	5.100	5.500	0.201	0.217
E1	7.600	8.000	0.299	0.315
e	0.65(BSC)		0.026(BSC)	
L	0.550	0.950	0.022	0.037
θ	0°	8°	0°	8°

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 DPTR,#data16	data16 -> DPTR	3	3	90
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 Z8D64U 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
	SectorGidA	SectorGid 寄存器 A	F8H	00H
	SectorGidB	SectorGid 寄存器 B	F9H	00H
RFC	RFCCSR	Flash 写/擦除控制寄存器	CAH	00H
	RFCOTPR	Flash OTP 标志字节寄存器	CBH	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
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	WDTCLR	看门狗状态清除寄存器	D8H	00H
	WDTCSR	看门狗控制及状态寄存器	D9H	00H
	WDTTAP	看门狗写控制寄存器	DAH	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	C8H	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
SSF33	SSFPR	SSF33 算法控制模块参数寄存器	9CH	00H
	SSFKR	SSF33 算法控制模块密钥寄存器	9DH	00H
	SSFDR	SSF33 算法控制模块数据寄存器	9EH	00H
	SSFCSR	SSF33 算法控制模块控制状态寄存器	9FH	00H
SCB2	SCBCSR	SCB2 算法控制模块控制状态寄存器	DBH	00H
	SCBEKR	SCB2 算法控制器 EK 参数寄存器	DCH	00H
	SCBAKR	SCB2 算法控制模块 AK 参数寄存器	DDH	00H
	SCBSKR	SCB2 算法控制模块 SK 参数寄存器	DEH	00H
	SCBDR	SCB2 算法控制模块数据寄存器	DFH	00H
PAE	PAMOD	PAE 模式寄存器	F7H	00H
	RSAELENH	PAE 幂长寄存器高 8 位	F6H	00H
	RSAELENL	PAE 幂长寄存器低 8 位	F5H	00H
	RSANLENH	PAE 模长寄存器高 8 位	F4H	00H
	RSANLENL	PAE 模长寄存器低 8 位	F3H	00H
	RSACR	PAE 启动运算寄存器	F2H	00H
	RSACMD	PAE 控制命令寄存器	F1H	00H
RNG	RNGDATA	RNG 数据寄存器	D7H	00H
	RNGSEED	RNG 种子数据寄存器	D7H	00H
	RNGCR	RNG 控制寄存器	D6H	00H
	RNGIPCR	RNGIP 选择寄存器	D4H	00H
	RNGDR	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
	COMMAND	COMMAND 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
SEC	SECCR	SEC 控制寄存器	E1H	00H
	SECSR	SEC 状态寄存器	E3H	00H
IOM	IOMCR	IOM 模式控制寄存器	8EH	00H
	GPIOCR	GPIO 方向控制寄存器	8FH	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
	EP4OUTBCR	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D 注释

2’bxx	两位的二进制数 xx
8’hxx	八位的十六进制数 xx

附录E 版本历史

版本号	日期	人员	描述
V1.0	2007-01-08		建立文档
V1.1	2007-04-13		修改文档，并正式发布
V1.2	2007-06-03		修改文档