Ver 1.8

SPARC V8 辐照加固微处理器

产品使用手册

产品型号：BM3803FMGRH

北京微电子技术研究所

版本控制页

版本号	发布日期	更改章节	更改说明	备注
1.0	2013.5	——	——	
1.1	2014.6.10	5.4.6（P41）	补充对存储器控制器 EDAC 功能详细说明。	
		5.4.6（P46—P48）	完善对“SRAM 和 SDRAM 出错处理”中 SRAM 和 SDRAM 出错处理的详细说明。	
		5.4.6（P48）	完善对“8 位、32 位 PROM 出错处理”中 8 位、32 位 PROM 出错处理的详细说明。	
		附录 4（P123—P210）	1、更正全部寄存器字段描述，增加建议软件初始化值； 2、将原附录 5“寄存器堆地址列表”并入附录 4。	
		附录 5（P212—P215）	增加 SRAM 与 I/O 访问时序图。	
		附录 6.5.3（P231—P233）	完善对陷阱与中断的详细描述。	
		附录 6.5.4（P234）	更正 Cache“刷新步骤”中刷新流程。	
		附录 6.6.2（P236）	1、更正 Regfile“造错”中造错流程； 2、增加“注意事项”中 ASR16 说明。	
		附录 6.6.3（P239）	更正 Cache“造错”中造错流程。	
		附录 6.6.4（P243）	更正存储器控制器“造错”中造错流程。	
		附录 6	增加 6.7 节“BM3803FMGRH 中断处	

		（P243—P246）	理”。	
		附录 7 （P248—P258）	完善“启动说明”，并改名为“处理器寄存器初始化”。	
		附录 8 （P259—P263）	1、将原附录 9、附录 10 并入附录 8，作为 8.1“浮点 IEEE-754 异常处理过程说明”、8.2“特定指令序列双精度浮点运算问题解决办法”； 2、增加 8.3“浮点异常陷阱返回地址问题解决办法”。	
		附录 9 （P265—P267）	完善“电源说明”，并改名为“对电源的要求”。	
		附录 10 （P268—P269）	完善“使用注意事项”。	
1.2	2014.11.17	5.2.2（P18）	增加 LDA/STA 指令 ASI 为 0xC—0xF 的说明。	
		5.4.6（P48—P50）	1、完善“其他”中字节写和半字写情况下 EDAC 功能详细说明； 2、增加“其他”中校验位生成公式； 3、将“其他”内容具体划分为几个段落，并删除“其他”标题。	
		5.5.9（P63）	增加“PCI 中断使用说明”。	
		5.6.1（P66）	增加“GPIO 中断”中 GPIO 电平中断说明。	
		5.6.2（P67—P68）	1、更正“发送/接收操作”中 UART 接收操作的过程描述； 2、更正“中断产生”中产生中断的条件； 3、增加 “UART 使用说明”。	
		附录 4（P124）	1、“特殊寄存器”改名为“状态寄存器”；	

			2、更正写操作指令是三指令延迟指令的寄存器为“PSR、WIM、TBR、ASR16、ASR17 寄存器”。	
		附录 4（P210）	更正“寄存器使用注意事项”中写状态寄存器需要三指令延迟的寄存器为“PSR、WIM、TBR、ASR16、ASR17 寄存器”。	
		附录 6.4（P223）	1、更正写状态寄存器需要三指令延迟的寄存器为 “%psr、%wim、%tbr、%asr16、%asr17 寄存器”； 2、删除 load 浮点状态寄存器指令（LDFSR）是延迟指令的描述。	
		附录 6.6.3（P239）	完善 Cache“造错”中造错流程。	
		附录 6.6.4（P243）	1、完善存储器控制器“造错”中造错流程； 2、更正“注意事项”中 PROM 校验码说明。	
		附录 6（P246—P247）	1、增加 6.8 节“引发 Trap 0x2b 陷阱条件”； 2、增加 6.9 节“引发 Trap 0x09 陷阱条件”。	
		附录 11（P270—P271）	增加“与 AT697E 差异性”。	
1.3	2015.1.28	附录 8.2（P260）	更正“……通过二进制检查工具对编译后的文件进行检查”为“……通过二进制检查工具对编译链接后的文件（含操作系统程序）进行检查”。	

		附录 8（P263—P264）	增加 8.4 节“Ticc 指令清除中断问题解决办法”。	
	2015.2.3	3（P3）	更正“读操作指示（read）”为“读/写操作指示（read）”。	
		附录 1（P91）	完善 J33 管脚说明为“外部存储器读/写指示，不作控制信号。高电平表示读，低电平表示写”。	
	2015.02.13	附录 8.2（P260-P261）	完善“软件修改方法”。	
1.4	2015.03.20	3（P3）	更正“DSU 串口发送信号(dsurx)”为“DSU 串口发送信号(dsutx)”。	
		附录 10（P267）	增加注意事项第 9 条“处理器工作在运行模式时，必须将 DSUEN 和 DSUBRE 接低电平，DSURX 接高电平”。	
1.5	2015.07.30	5.4.6（P49）	EDAC 编码公式修改，PD2 和 PD3 前面增加“~”。	
		5.4.4（P40）	增加 I/O 设备不同数据总线宽度的管脚连接与读写访问说明。	
		5.4.6（P40） 附录 10（P267）	增加表述：SDRAM 存储器 EDAC 功能不支持内部 AHB 总线的 burst 传输模式。	
	2015.11.06	附录 10 P268）	增加表述：LDDx 指令在 AHB 总线上发起的传输是 burst 传输模式	
	2016.03.09	附录 1（P99）	表格 A1-3，更正 cpci_en 为 testin0。	
		附录 1（P100）	增加说明：注：这些上拉下拉输入管脚的上拉和下拉在片内 pad 上。	
	2016.10.18	附录 4（P165）	更正“GPIO 方向寄存器”的“ LOWDIR[16]对应数据总线 D[15:8]”为“LOWDIR[16]对应数据总线 D[7:0]”。	

	2016.12.5	5.4.7（P49-P51）	完善“写保护说明”。	
1.6	2017.5.2	5.4.4(P40)	完善“BRDYN 时序说明”。	
		5.6.1（P67）	完善“GPIO 复用管脚说明”。	
		5.6.3（P72）	完善“定时器与看门狗计数值说明”。	
		6.6.4（P241）	增加“存储器控制器 EDAC 注意事项”。	
		6.9（P245）	增加“Trap 0x09 陷阱条件”。	
		附录 12（P269）	增加“PCI 不使用时 PCI 管脚接法说明”。	
1.7	2018.09.17	全文	调整章节和附录顺序，更新全文格式。	
		1.3	增加“惯例与术语表”。	
		4.11	增加“时钟倍频”。	
		4.12	增加“JTAG 接口”。	
		7.5	增加“产品防护”。	
		第八章	增加“用户关注产品信息”。	
		附录 1	增加“警示页”。	
1.8	2019.05.06	4.6.3（P37、P39）	增加“硬件不支持中断嵌套，需软件实现中断嵌套”说明。	
		附录 5.3(P260)	增加“硬件不支持中断嵌套，需软件实现中断嵌套”说明。	

目 录

一、 产品概述 1

1.1 产品特点 1
1.2 产品用途及应用范围 2
1.3 惯例和术语表 3
1.4 免责声明 4

二、 产品工作条件 5

2.1 绝对最大额定值 5
2.2 推荐工作条件 5

三、 封装及引出端说明 6

3.1 引出端排列 6
3.2 引脚信号描述 6
3.3 外形尺寸说明 19

四、 产品功能 21

4.1 产品结构框图 21
4.2 处理器核心 21
4.2.1 整数单元 22
4.2.2 指令集 22
4.2.3 浮点处理单元 23
4.2.4 工作模式 23
4.2.5 产品复位 23
4.2.6 产品异常 23
4.2.6 容错设计 25
4.3 CACHE 25
4.3.1 概述 25
4.3.2 Cache 缓存操作 25
4.3.3 Cache 的容错功能 27
4.3.4 Cache 的寄存器地址映射 27
4.4 定时器单元 27
4.4.1 预分频器 27
4.4.2 定时器 28
4.4.3 看门狗 28
4.4.4 定时器和看门狗的寄存器地址映射 28
4.5 GPIO 29
4.5.1 概述 29
4.5.2 GPIO 低 16 位 29
4.5.3 GPIO 高 16 位 29
4.5.4 复用功能 30
4.5.5 GPIO 中断 31

4.5.6 GPIO 寄存器地址映射 31
4.6 中断和陷阱 32
4.6.1 概述 32
4.6.2 陷阱类型 33
4.6.3 中断 36
4.7 UART 39
4.7.1 发送操作 39
4.7.2 接收操作 40
4.7.3 流控 40
4.7.4 回环 41
4.7.5 波特率产生 41
4.7.6 中断产生 41
4.7.7 UART 使用说明 41
4.7.8 UART 的寄存器地址映射 41
4.8 外部存储器接口 42
4.8.1 概述 42
4.8.2 外部存储器控制器地址空间 43
4.8.3 RAM 接口 49
4.8.4 SRAM 接口 49
4.8.5 SDRAM 接口 54
4.8.6 PROM 接口 59
4.8.7 I/O 设备接口 63
4.8.8 访问错误 66
4.8.9 错误管理 66
4.8.10 写保护 76
4.9 PCI 接口和仲裁器 77
4.9.1 概述 77
4.9.2 AHB-PCI 模块结构 78
4.9.3 PCI 接口复位 79
4.9.4 PCI 接口 AHB 侧地址空间 79
4.9.5 PCI 仲裁器 81
4.9.6 PCI 侧寄存器空间 81
4.9.7 地址空间映射 83
4.9.8 DMA 85
4.9.9 PCI 模块中断 89
4.10 调试支持单元 91
4.10.1 概述 91
4.10.2 调试通讯链接 98
4.10.3 启动调试 100
4.10.4 调试支持单元的寄存器地址映射 100
4.11 时钟倍频 101
4.11.1 概述 101
4.11.2PLL 使用要求 101

4.12 JTAG 接口 101
4.12.1 概述 101
4.12.2JTAG 使用要求 102
五、 产品电特性 104
5.1 电特性 104
六、 典型应用 116
6.1 开发板说明 116
6.2 开发板参考电路 118
6.2.1 电源模块 118
6.2.2 时钟复位电路 118
6.2.3 DSU 串口电路 120
6.2.4 测试管脚 120
6.2.5 FLASH 121
6.2.6 SRAM 122
6.2.7 SDRAM 122
七、 应用注意事项 124
7.1 产品应用说明 124
7.2 对电源的要求 125
7.3 常见问题软件规避 128
7.3.1 浮点 IEEE-754 异常处理方法 128
7.3.2 特定指令序列双精度浮点运算问题解决办法 128
7.3.3 浮点异常陷阱返回地址问题解决办法 131
7.3.4 Ticc 指令清除中断问题解决办法 132
7.4 编程说明 133
7.4.1 BM3803FMGRH 汇编编程注意事项 133
7.4.2 BM3803FMGRH 软件示例 137
7.5 产品防护 143
7.5.1 电装及防护措施 143
7.5.2 包装 144
7.5.3 运输和贮存 144
八、 用户关注产品信息 145
8.1 产品鉴定信息 145
8.2 产品标识 145
8.3 研制生产单位联系方式 146
附录 1 警示页 147
附录 2 寄存器列表 148
2.1 外设寄存器列表 148
2.1.1 0x80000000 地址上的寄存器 148
2.1.2 0x90000000 地址上的寄存器 149

2.1.3 0xC0000000 地址上的寄存器 149
2.1.4 PCI 配置空间的寄存器 150

附录 3 寄存器描述 151

3.1 IU 寄存器 151
3.1.1 全局寄存器 151
3.1.2 窗口寄存器 151
3.2 浮点寄存器 155
3.3 状态寄存器 155
3.4 系统寄存器 167
3.5 CACHE 寄存器 169
3.6 外部存储器接口寄存器 172
3.7 写保护寄存器 180
3.8 定时器和看门狗寄存器 181
3.9 UART 寄存器 185
3.10 中断寄存器 192
3.11 GPIO 寄存器 194
3.12 PCI 寄存器 196
3.13 DSU 寄存器 222
3.14 寄存器堆地址列表 227
3.14.1 全局寄存器 227
3.14.2 窗口寄存器 228
3.14.3 浮点寄存器 230
3.15 寄存器使用注意事项 230

附录 4 寄存器初始化 232

4.1 寄存器初始化简介 232
4.2 推荐寄存器初始化过程 232
4.2.1 内核寄存器初始化 232
4.2.2 片上寄存器初始化 238
4.2.3 用户程序相关寄存器初始化 240

附录 5 软件开发规范 242

5.1 通用编程规范 242
5.1.1 寄存器窗口与函数调用 242
5.1.2 堆栈 247
5.1.3 陷阱与中断 248
5.1.4 Cache 刷新 251
5.2 BM3803FMGRH 软件容错设计 251
5.2.1 Regfile EDAC 252
5.2.2 Cache 奇偶校验 253
5.2.3 外部存储器控制器 EDAC 256
5.3 BM3803FMGRH 中断处理 260
5.3.1 中断处理流程 260

5.3.2 中断处理中的中断级别设置 263
5.4 引发 TRAP0X2B 陷阱条件 264
5.5 引发 TRAP0X09 陷阱条件 264
附录 6 SPARC V8 介绍 266
6.1 SPARC V8 概述 266
6.2 数据类型 266
6.3 寄存器 268
6.3.1 IU 寄存器 268
6.3.2 FPU 寄存器 269
6.3.3 寄存器推荐使用方法汇总 269
6.4 指令 270
6.4.1 指令格式 270
6.4.2 指令分类 270
6.5 陷阱 271
6.5.1 精确陷阱 271
6.5.2 中断陷阱 272
6.5.3 延迟陷阱 272
6.5.4 陷阱控制 272
6.5.5 陷阱识别 273
附录 7 PROM、SRAM、IO、SDRAM 访问时序 274
附录 8 与 AT697E 差异性 282
附录 9 PCI 不使用时 PCI 管脚接法说明 284

一、 产品概述

BM3803FMGRH 处理器是基于 SPARC V8 体系结构的抗辐射 32 位 RISC 嵌入式处理器，内核包含整数处理单元、浮点处理单元、独立的指令和数据 Cache、硬件乘/除法器，外设包括中断控制器、定时器、看门狗、UART、通用 I/O 接口、外部存储器控制器、PCI 总线控制器等。

1.1 产品特点

- 最大工作频率：100MHz
- 兼容 SPARC V8 体系结构的 32 位微处理器内核：
  - 符合 IEEE754 标准的 32/64 位浮点处理单元 FPU
  - 32KB 指令 Cache，16KB 数据 Cache
- 内嵌硬件乘/除法器
- 外部存储器控制器：
  - 支持 8/16/32 位宽 PROM、SRAM 和 I/O 空间
  - 支持 32 位宽 SDRAM
- 中断控制器：
  - 15 路中断，其中 4 路外部中断
- 3 路 UART
- 2 路 24 位通用定时器
- 1 路 24 位看门狗
- 32 路 GPIO
- PCI 接口
- 容错设计：
  - 三模冗余
  - 纠一检二的 EDAC 编解码
  - 奇偶校验
- 硬件调试单元：

- 缓存跟踪功能
- 4 个硬件观测点

➢ 工作电压：
- I/O 电源电压：3.3V±0.30V
- 内核电源电压：1.8V±0.15V

➢ 工作温度：-55℃～+125℃

➢ 存储温度：-65℃～+150℃

➢ 功耗：小于 1W

➢ ESD 抗静电能力：2000V

➢ 抗辐性能：
- 抗总剂量：≥ 1×10^{3} Gy（Si）
- 单粒子锁定（SEL）阈值：≥ 75MeV cm^{2}/mg
- 单粒子功能在轨错误率（GEO 轨道）：$\leqslant 8\times 10^{-5}$次/天 器件

➢ 质量等级：CASTC 级

➢ 封装形式：CPGA391

➢ 重量：不大于 33g

➢ IEEE 1149.1 JTAG 接口

➢ 软件开发环境：SPE-C
- SPARC 集成开发环境（SPE-C）
- SPARC 软件开发环境（SparcSDE）

1.2 产品用途及应用范围

BM3803FMGRH 处理器是基于 SPARC V8 体系结构的抗辐射 32 位 RISC 嵌入式处理器，内核包含整数处理单元、浮点处理单元、独立的指令和数据 Cache、硬件乘/除法器，外设包括中断控制器、定时器、看门狗、UART、通用 I/O 接口、外部存储器控制器、PCI 总线控制器等，能够满足空间恶劣环境应用的抗辐射、长寿命、高可靠指标要求，与存储器和相关外围电路相连，可构成完整的嵌入式单板计算机系统。

1.3 惯例和术语表

本节定义了本手册使用的惯例和术语表。如表 1-3-1 所示。

表 1-3-1 惯例和术语表

惯例/术语	描述
ASI	Address Space Identifyier，地址空间标志符
ASR	Ancillary State Registers，辅助状态寄存器
Cache	高速缓冲存储器
CPU	Central Processing Uni，中心处理单元
DSU	Debug Support Unit，调试支持单元
Flash	闪存
FPU	Floating Point Unit，浮点单元
FSR	Floating Point State Register，浮点状态寄存器
GPIO	General Purpose Input Output，通用输入输出接口
IU	Integer Unit，整数单元
JTAG	Joint Test Action Group，联合测试工作组
LMR	Load Mode Register，加载模式寄存器
LRU	Least Recently Used，最近最少使用
LSB	Least Significant Bit，最低有效位
MSB	Most Significant Bit，最高有效位
PC/nPC	Program Counter/next Program Counter，程序计数器
PCI	Peripheral Component Interconnect，外设部件互连
PLL	Phase Locked Loop，锁相环
PROM	Programmable Read-Only Memory，可编程只读存储器
PSR	Processor State Register，处理器状态寄存器
RISC	Reduced Instruction Set Computer，精简指令集处理器
SDRAM	Synchronous Dynamic Random Access Memory，同步动态随机存储器

SPARC	Scalable Processor ARChitectur，可扩充处理器架构
SRAM	Static Random Access Memory，静态随机存取存储器
TBR	Trap Base Register，异常基址寄存器
UART	Universal Asynchronous Receiver and Transmitter，通用异步收发器
WIM	Windows Invalid Mask Register，窗口无效寄存器

⚠注意

1、定义“惯例/术语”之后，本文其他地方使用“惯例/术语”时均统一引用方式，例如按照本表格，表达“浮点状态寄存器”时，均使用“FSR”。

2、寄存器复位值按照十六进制统一写法“0xXXX0XXXX”，“0x”表示 16 进制，每 4bit 中有 1bit 不定值，即写为“X”，4bit 全部为固定值则写为相应 16 进制数值。

3、**本文中的寄存器名字的表述或引用使用加粗的表示方式**。

4、*本文中对寄存器中的字段的表述或引用使用斜体的表示方式*。

1.4 免责声明

本手册版权归北京微电子技术研究所所有，并保留一切权利。未经书面许可，任何单位、组织和个人不得将此文档中的任何部分公开、转载或以其他方式散发给第三方，否则将追究其法律责任。

本手册版本将不定期更新，请在使用本产品之前联系本单位销售部门获取本手册的最新版本。

用户因未严格按本手册要求保存、使用本产品，致使产品工作异常或损坏，造成任何直接或间接损失，本单位不承担任何责任。

除本手册说明之外，请勿接受第三方指导或参考第三方资料对本产品进行操作，用户对本手册有疑问之处请与本单位销售部门联系。

二、 产品工作条件

2.1 绝对最大额定值

绝对最大额定值如下：

数字核电源电压范围：-0.3V～+2.0V；

模拟核电源电压范围：-0.3V～+2.0V；

数字 I/O 电源电压范围：-0.3V～+4.0V；

模拟 I/O 电源电压范围：-0.3V～+4.0V；

存储温度范围：-65℃～+150℃。

2.2 推荐工作条件

推荐工作条件如下：

数字核电源电压范围：1.65V～1.95V；

模拟核电源电压范围：1.65V～1.95V；

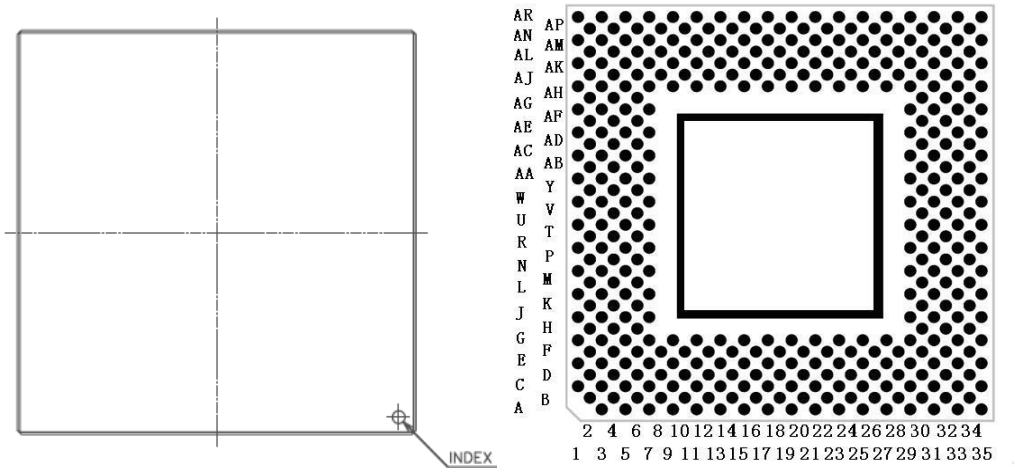
数字 I/O 电源电压范围：3.0V～3.6V；

模拟 I/O 电源电压范围：3.0V～3.6V；

工作温度范围：-55℃～+125℃。

三、 封装及引出端说明

3.1 引出端排列



顶视图　　底视图

图 3-1-1 引出端排列图

3.2 引脚信号描述

表 3-2-1 引脚信号描述

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
1	resetn	—	P4	I	系统复位，低电平有效，需在时钟有效后至少保持 10 个时钟周期
2	clk	—	D28	I	系统时钟
3	errorn	—	P6	OD	处理器错误模式指示，低电平输出有效
4	address[0]	—	M2	O	地址总线第 0 位，最低位
5	address[1]	—	N5	O	地址总线第 1 位
6	address[2]	—	M4	O	地址总线第 2 位
7	address[3]	—	K2	O	地址总线第 3 位
8	address[4]	—	M6	O	地址总线第 4 位
9	address[5]	—	J1	O	地址总线第 5 位
10	address[6]	—	L5	O	地址总线第 6 位
11	address[7]	—	J3	O	地址总线第 7 位

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
12	address[8]	—	G1	O	地址总线第 8 位
13	address[9]	—	J5	O	地址总线第 9 位
14	address[10]	—	F2	O	地址总线第 10 位
15	address[11]	—	K6	O	地址总线第 11 位
16	address[12]	—	G3	O	地址总线第 12 位
17	address[13]	—	D2	O	地址总线第 13 位
18	address[14]	—	F4	O	地址总线第 14 位
19	address[15]	—	E3	O	地址总线第 15 位
20	address[16]	—	B2	O	地址总线第 16 位
21	address[17]	—	C5	O	地址总线第 17 位
22	address[18]	—	D6	O	地址总线第 18 位
23	address[19]	—	B4	O	地址总线第 19 位
24	address[20]	—	F8	O	地址总线第 20 位
25	address[21]	—	C7	O	地址总线第 21 位
26	address[22]	—	B6	O	地址总线第 22 位
27	address[23]	—	A5	O	地址总线第 23 位
28	address[24]	—	E9	O	地址总线第 24 位
29	address[25]	—	C9	O	地址总线第 25 位
30	address[26]	—	D10	O	地址总线第 26 位
31	address[27]	—	B8	O	地址总线第 27 位，最高位
32	bexcn	上拉	N3	I	总线异常，低电平有效，用于 PROM、SRAM 和 I/O 空间
33	brdyn	上拉	G35	I	外部数据准备好，低电平有效，用于第 5 个 SRAM 和 I/O 空间
34	data[0]	—	E11	I/O	数据总线第 0 位，最低位
35	data[1]	—	A9	I/O	数据总线第 1 位
36	data[2]	—	C11	I/O	数据总线第 2 位
37	data[3]	—	A11	I/O	数据总线第 3 位
38	data[4]	—	E13	I/O	数据总线第 4 位
39	data[5]	—	B12	I/O	数据总线第 5 位
40	data[6]	—	D14	I/O	数据总线第 6 位
41	data[7]	—	A13	I/O	数据总线第 7 位
42	data[8]	—	B14	I/O	数据总线第 8 位
43	data[9]	—	E15	I/O	数据总线第 9 位
44	data[10]	—	C15	I/O	数据总线第 10 位
45	data[11]	—	D16	I/O	数据总线第 11 位
46	data[12]	—	B16	I/O	数据总线第 12 位

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
47	data[13]	—	C17	I/O	数据总线第 13 位
48	data[14]	—	E17	I/O	数据总线第 14 位
49	data[15]	—	A17	I/O	数据总线第 15 位
50	data[16]	—	D18	I/O	数据总线第 16 位
51	data[17]	—	A19	I/O	数据总线第 17 位
52	data[18]	—	E19	I/O	数据总线第 18 位
53	data[19]	—	C19	I/O	数据总线第 19 位
54	data[20]	—	B20	I/O	数据总线第 20 位
55	data[21]	—	F20	I/O	数据总线第 21 位
56	data[22]	—	C21	I/O	数据总线第 22 位
57	data[23]	—	E21	I/O	数据总线第 23 位
58	data[24]	—	B22	I/O	数据总线第 24 位
59	data[25]	—	A23	I/O	数据总线第 25 位
60	data[26]	—	D22	I/O	数据总线第 26 位
61	data[27]	—	E23	I/O	数据总线第 27 位
62	data[28]	—	A25	I/O	数据总线第 28 位
63	data[29]	—	C25	I/O	数据总线第 29 位
64	data[30]	—	D24	I/O	数据总线第 30 位
65	data[31]	—	B26	I/O	数据总线第 31 位，最高位
66	dsuact	—	U1	O	高电平有效，表示处理器处于调试模式
67	dsubre	—	V4	I	硬件调试模式控制输入，当 dsuen 为高电平有效时，在系统复位结束前三个时钟周期 dsubre 为高电平,处理器在复位结束后运行四条指令后立即进入调试模式
68	dsuen	—	W3	I	硬件调试模式使能，当调试模式触发条件满足时，处理器进入调试模式
69	dsurx	上拉	V2	I	硬件调试串口输入
70	dsutx	—	W5	O	硬件调试串口输出
71	pci_66	—	AK20	I	测试管脚接低
72	sim_fast_reset	下拉	AE1	I	测试管脚，板级设计时接 1KΩ电阻下拉
73	testin0	下拉	AC5	I	测试管脚，板级设计时接 1KΩ电阻下拉
74	testin1	—	AH4	I	测试管脚，板级设计时接 1KΩ电阻下拉
75	testout0	—	AD2	O	测试管脚
76	testout1	—	AL1	I/O	测试管脚，板级设计时接 1KΩ电阻上拉
77	pci_ad[0]	—	AR27	I/O	PCI 数据/地址总线最低位
78	pci_ad[1]	—	AK24	I/O	PCI 数据/地址总线
79	pci_ad[2]	—	AP26	I/O	PCI 数据/地址总线

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
80	pci_ad[3]	—	AM24	I/O	PCI 数据/地址总线
81	pci_ad[4]	—	AN25	I/O	PCI 数据/地址总线
82	pci_ad[5]	—	AP24	I/O	PCI 数据/地址总线
83	pci_ad[6]	—	AK22	I/O	PCI 数据/地址总线
84	pci_ad[7]	—	AN23	I/O	PCI 数据/地址总线
85	pci_ad[8]	—	AR23	I/O	PCI 数据/地址总线
86	pci_ad[9]	—	AN21	I/O	PCI 数据/地址总线
87	pci_ad[10]	—	AR21	I/O	PCI 数据/地址总线
88	pci_ad[11]	—	AM20	I/O	PCI 数据/地址总线
89	pci_ad[12]	—	AP20	I/O	PCI 数据/地址总线
90	pci_ad[13]	—	AN19	I/O	PCI 数据/地址总线
91	pci_ad[14]	—	AL19	I/O	PCI 数据/地址总线
92	pci_ad[15]	—	AR19	I/O	PCI 数据/地址总线
93	pci_ad[16]	—	AM14	I/O	PCI 数据/地址总线
94	pci_ad[17]	—	AN13	I/O	PCI 数据/地址总线
95	pci_ad[18]	—	AK14	I/O	PCI 数据/地址总线
96	pci_ad[19]	—	AP12	I/O	PCI 数据/地址总线
97	pci_ad[20]	—	AL13	I/O	PCI 数据/地址总线
98	pci_ad[21]	—	AR11	I/O	PCI 数据/地址总线
99	pci_ad[22]	—	AP10	I/O	PCI 数据/地址总线
100	pci_ad[23]	—	AK12	I/O	PCI 数据/地址总线
101	pci_ad[24]	—	AP8	I/O	PCI 数据/地址总线
102	pci_ad[25]	—	AM8	I/O	PCI 数据/地址总线
103	pci_ad[26]	—	AR7	I/O	PCI 数据/地址总线
104	pci_ad[27]	—	AL9	I/O	PCI 数据/地址总线
105	pci_ad[28]	—	AR5	I/O	PCI 数据/地址总线
106	pci_ad[29]	—	AP6	I/O	PCI 数据/地址总线
107	pci_ad[30]	—	AK10	I/O	PCI 数据/地址总线
108	pci_ad[31]	—	AR3	I/O	PCI 数据/地址总线最高位
109	pci_arb_gnt_n[0]	—	AL29	O	PCI 总线占用允许（主桥模式）
110	pci_arb_gnt_n[1]	—	AR33	O	PCI 总线占用允许（主桥模式）
111	pci_arb_gnt_n[2]	—	AK28	O	PCI 总线占用允许（主桥模式）
112	pci_arb_gnt_n[3]	—	AP32	O	PCI 总线占用允许（主桥模式）
113	pci_arb_gnt_n[4]	—	AM30	O	PCI 总线占用允许（主桥模式）
114	pci_arb_gnt_n[5]	—	AN31	O	PCI 总线占用允许（主桥模式）
115	pci_arb_gnt_n[6]	—	AM32	O	PCI 总线占用允许（主桥模式）
116	pci_arb_req[0]	—	AM26	I	PCI 总线占用请求（主桥模式）

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
117	pci_arb_req[1]	—	AN27	I	PCI 总线占用请求（主桥模式）
118	pci_arb_req[2]	—	AM28	I	PCI 总线占用请求（主桥模式）
119	pci_arb_req[3]	—	AR29	I	PCI 总线占用请求（主桥模式）
120	pci_arb_req[4]	—	AL27	I	PCI 总线占用请求（主桥模式）
121	pci_arb_req[5]	—	AR31	I	PCI 总线占用请求（主桥模式）
122	pci_arb_req[6]	—	AP30	I	PCI 总线占用请求（主桥模式）
123	pci_cbe[0]	—	AM22	I/O	PCI 命令和字节使能复用信号
124	pci_cbe[1]	—	AK18	I/O	PCI 命令和字节使能复用信号
125	pci_cbe[2]	—	AR13	I/O	PCI 命令和字节使能复用信号
126	pci_cbe[3]	—	AL11	I/O	PCI 命令和字节使能复用信号
127	pci_clk	—	AM4	I	PCI 时钟输入，33MHz
128	pci_devsel	—	AM16	I/O	PCI 设备选择信号
129	pci_frame	—	AN15	I/O	PCI 帧周期信号
130	pci_gnt	—	AP4	I	PCI 总线占用允许（从桥模式）
131	pci_host	上拉	AE3	I	PCI 桥主桥/从桥模式配置,接低电平为主桥模式，接高电平为从桥模式
132	pci_idry	—	AK16	I/O	PCI 主设备准备好信号
133	pci_idsel	—	AR9	I	PCI 初始化器件选择信号
134	pci_inta_n	—	AK2	I/O	PCI 总线中断 A，低电平有效（从桥模式输出信号，主桥模式输入信号）
135	pci_intb_n	—	AJ3	I	PCI 总线中断 B
136	pci_intc_n	—	AH6	I	PCI 总线中断 C
137	pci_intd_n	—	AM2	I	PCI 总线中断 D
138	pci_par	—	AR17	I/O	PCI 奇偶校验信号
139	pci_perr	—	AN17	I/O	PCI 校验错误指示
140	pci_req	—	AK8	O	PCI 总线占用请求（从桥模式）
141	pci_rst	—	AK4	I/O	PCI 复位信号，低电平有效。（从桥模式输入信号，主桥模式输出信号）
142	pci_serr	—	AL17	I/O	PCI 系统错误指示
143	pci_stop	—	AP16	I/O	PCI 停止数据传送信号
144	pci_trdy	—	AR15	I/O	PCI 目标设备准备好信号
145	pdata[0]	—	E25	I/O	校验数据总线最低位
146	pdata[1]	—	B28	I/O	校验数据总线
147	pdata[2]	—	D26	I/O	校验数据总线
148	pdata[3]	—	E27	I/O	校验数据总线
149	pdata[4]	—	A31	I/O	校验数据总线
150	pdata[5]	—	B30	I/O	校验数据总线

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
151	pdata[6]	—	F26	I/O	校验数据总线最高位
152	pio[0]	—	AE33	I/O	通用 I/O 与 PROM 宽度定义复用，上电复位时为输入，用于配置 PROM 位宽，pio[1:0]=“00”表示 8 位宽，“01”表示 16 位宽，“1x”表示 32 位宽。
153	pio[1]	—	AD32	I/O	通用 I/O 与 PROM 宽度定义复用，上电复位时为输入，用于配置 PROM 位宽，pio[1:0]=“00”表示 8 位宽，“01”表示 16 位宽，“1x”表示 32 位宽。
154	pio[2]	—	AF34	I/O	通用 I/O
155	pio[3]	—	AD30	I/O	通用 I/O 与串口外部可选时钟复用
156	pio[4]	—	AG35	I/O	通用 I/O
157	pio[5]	—	AE31	I/O	通用 I/O
158	pio[6]	—	AH34	I/O	通用 I/O
159	pio[7]	—	AH32	I/O	通用 I/O
160	pio[8]	—	AJ35	I/O	通用 I/O 与 cts2 复用
161	pio[9]	—	AG31	I/O	通用 I/O 与 rts2 复用
162	pio[10]	—	AL35	I/O	通用 I/O 与 rxd2 复用
163	pio[11]	—	AJ33	I/O	通用 I/O 与 txd2 复用
164	pio[12]	—	AJ31	I/O	通用 I/O 与 cts1 复用
165	pio[13]	—	AN35	I/O	通用 I/O 与 rts1 复用
166	pio[14]	—	AH30	I/O	通用 I/O 与 rxd1 复用
167	pio[15]	—	AM34	I/O	通用 I/O 与 txd1 复用
168	pll_bp	上拉	K30	I	PLL 时钟旁路控制信号，高电平时旁路 PLL，片内时钟直接由晶振提供；低电平时，片内时钟由晶振经 PLL 提供
169	pll_min[0]	下拉	G33	I	PLL 倍频系数输入，[0]为最低位，[3]为最高位，四位不可同时配置成 0
170	pll_min[1]	下拉	F28	I	PLL 倍频系数输入，[0]为最低位，[3]为最高位，四位不可同时配置成 0
171	pll_min[2]	下拉	A33	I	PLL 倍频系数输入，[0]为最低位，[3]为最高位，四位不可同时配置成 0
172	pll_min[3]	下拉	E29	I	PLL 倍频系数输入，[0]为最低位，[3]为最高位，四位不可同时配置成 0
173	pll_out	—	E35	O	PLL 输出，测试管脚，不作控制用
174	pll_rst	—	C29	I	PLL 复位信号，高电平有效，板级设计时接 1KΩ电阻下拉

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
175	ramoen[0]	—	M34	O	SRAM 读使能，低电平有效
176	ramoen[1]	—	N33	O	SRAM 读使能，低电平有效
177	ramoen[2]	—	P32	O	SRAM 读使能，低电平有效
178	ramoen[3]	—	R31	O	SRAM 读使能，低电平有效
179	ramoen[4]	—	R33	O	SRAM 读使能，低电平有效
180	ramsn[0]	—	H34	O	SRAM 片选，低电平有效
181	ramsn[1]	—	L31	O	SRAM 片选，低电平有效
182	ramsn[2]	—	K34	O	SRAM 片选，低电平有效
183	ramsn[3]	—	M32	O	SRAM 片选，低电平有效
184	ramsn[4]	—	L33	O	SRAM 片选，低电平有效
185	read	—	J33	O	外部存储器读/写指示，高电平表示读，低电平表示写，不作控制用。
186	rombsd[0]	—	AE5	I	Prom_bank 大小定义，用于 8 位宽 PROM 的 EDAC 功能。上电复位时用于配置 PROM 容量。rombsd[3:0]，[0]为最低位[3]为最高位。
187	rombsd[1]	—	AH2	I	Prom_bank 大小定义，用于 8 位宽 PROM 的 EDAC 功能。上电复位时用于配置 PROM 容量。rombsd[3:0]，[0]为最低位[3]为最高位。
188	rombsd[2]	—	AF4	I	Prom_bank 大小定义，用于 8 位宽 PROM 的 EDAC 功能。上电复位时用于配置 PROM 容量。rombsd[3:0]，[0]为最低位[3]为最高位。
189	rombsd[3]	—	AG3	I	Prom_bank 大小定义，用于 8 位宽 PROM 的 EDAC 功能。上电复位时用于配置 PROM 容量。rombsd[3:0]，[0]为最低位[3]为最高位。
190	romsn[0]	—	U35	O	PROM 片选，低电平有效
191	romsn[1]	—	V30	O	PROM 片选，低电平有效
192	iosn	—	V34	O	I/O 片选，低电平有效
193	oen	—	H32	O	读使能，低电平有效，PROM 和 I/O 空间共用。
194	writen	—	K32	O	写使能，低电平有效，PROM 和 I/O 空间共用。
195	rwen[0]	—	T30	O	SRAM 字节控制的写使能，低电平有效，对应数据管脚 data[31:24]。

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
196	rwen[1]	—	R35	O	SRAM 字节控制的写使能，低电平有效，对应数据管脚 data[23:16]。
197	rwen[2]	—	T32	O	SRAM 字节控制的写使能，低电平有效，对应数据管脚 data[15:8]。
198	rwen[3]	—	T34	O	SRAM 字节控制的写使能，低电平有效，对应数据管脚 data[7:0]。
199	sdclk	—	AC31	O	SDRAM 时钟输出
200	sdcsn[0]	—	W31	O	SDRAM 片选，低电平有效
201	sdcsn[1]	—	W33	O	SDRAM 片选，低电平有效
202	sddqm[0]	—	AB34	O	SDRAM 字节选通，低电平有效，对应数据管脚 data[7:0]
203	sddqm[1]	—	AC35	O	SDRAM 字节选通，低电平有效，对应数据管脚 data[15:8]
204	sddqm[2]	—	AB32	O	SDRAM 字节选通，低电平有效，对应数据管脚 data[23:16]
205	sddqm[3]	—	AC33	O	SDRAM 字节选通，低电平有效，对应数据管脚 data[31:24]
206	sdrasn	—	Y32	O	SDRAM 行地址选择，低电平有效
207	sdcasn	—	AA35	O	SDRAM 列地址选择，低电平有效
208	sdwen	—	Y34	O	SDRAM 写使能，低电平有效
209	scan_enable	下拉	G31	I	scan 使能，高电平有效，板级设计时接 1KΩ电阻下拉
210	scan_mode	下拉	E1	I	scan 模式的选择，高电平有效，板级设计时接 1KΩ电阻下拉
211	bist_mode	下拉	C35	I	mbist 测试模式使能信号，高电平有效，板级设计时接 1KΩ电阻下拉
212	skew[0]	下拉	AG1	I	时钟树延时输入，板级设计时接 1KΩ电阻下拉
213	skew[1]	下拉	AD6	I	时钟树延时输入，板级设计时接 1KΩ电阻下拉
214	tclk	下拉	T2	I	JTAG 时钟，不使用时接 1KΩ电阻下拉
215	tdi	上拉	T4	I	JTAG 数据输入,不使用时接 1KΩ电阻上拉
216	tdo	—	R1	tri_state_O	JTAG 数据输出
217	tms	上拉	P2	I	JTAG 模式选择,不使用时接 1KΩ电阻上拉
218	trst	上拉	N1	I	JTAG 复位，低电平有效，不使用时接 1KΩ电阻下拉
219	timer0	—	Y4	O	定时器 timer0 输出,溢出时输出一个系统

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
					时钟周期的高电平脉冲
220	timer1	—	Y2	O	定时器 timer1 输出,溢出时输出一个系统时钟周期的高电平脉冲
221	wdogn	—	AB6	OD	看门狗输出，低电平有效
222	rxd3	上拉	AB2	I	串口 3 数据输入
223	txd3	—	AC1	O	串口 3 数据输出
224	rts3	—	AA5	O	串口 3 请求发送
225	cts3	上拉	AA3	I	串口 3 清除发送
226	pll_corevdd	—	D32	P	PLL 模拟核电源
227	pll_corevss	—	E33	P	PLL 模拟核地
228	pll_iovdd	—	D30	P	PLL 模拟 IO 电源
229	pll_iovss	—	C31	P	PLL 模拟 IO 地
230	vdd_core	—	A15	P	数字核电源
231	vdd_core	—	A21	P	数字核电源
232	vdd_core	—	A27	P	数字核电源
233	vdd_core	—	D12	P	数字核电源
234	vdd_core	—	E7	P	数字核电源
235	vdd_core	—	F34	P	数字核电源
236	vdd_core	—	K4	P	数字核电源
237	vdd_core	—	L35	P	数字核电源
238	vdd_core	—	R5	P	数字核电源
239	vdd_core	—	U31	P	数字核电源
240	vdd_core	—	AA1	P	数字核电源
241	vdd_core	—	AD4	P	数字核电源
242	vdd_core	—	AE35	P	数字核电源
243	vdd_core	—	AJ1	P	数字核电源
244	vdd_core	—	AJ5	P	数字核电源
245	vdd_core	—	AK34	P	数字核电源
246	vdd_core	—	AL15	P	数字核电源
247	vdd_core	—	AM10	P	数字核电源
248	vdd_core	—	AP22	P	数字核电源
249	vdd_core	—	AP28	P	数字核电源
250	vdd_io	—	B18	P	数字 I/O 电源
251	vdd_io	—	B24	P	数字 I/O 电源
252	vdd_io	—	C1	P	数字 I/O 电源
253	vdd_io	—	C13	P	数字 I/O 电源

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
254	vdd_io	—	C27	P	数字 I/O 电源
255	vdd_io	—	D4	P	数字 I/O 电源
256	vdd_io	—	D8	P	数字 I/O 电源
257	vdd_io	—	H30	P	数字 I/O 电源
258	vdd_io	—	J35	P	数字 I/O 电源
259	vdd_io	—	L3	P	数字 I/O 电源
260	vdd_io	—	N35	P	数字 I/O 电源
261	vdd_io	—	U5	P	数字 I/O 电源
262	vdd_io	—	V6	P	数字 I/O 电源
263	vdd_io	—	V32	P	数字 I/O 电源
264	vdd_io	—	Y30	P	数字 I/O 电源
265	vdd_io	—	AB4	P	数字 I/O 电源
266	vdd_io	—	AF32	P	数字 I/O 电源
267	vdd_io	—	AL3	P	数字 I/O 电源
268	vdd_io	—	AL33	P	数字 I/O 电源
269	vdd_io	—	AN7	P	数字 I/O 电源
270	vdd_io	—	AN11	P	数字 I/O 电源
271	vdd_io	—	AN29	P	数字 I/O 电源
272	vdd_io	—	AP18	P	数字 I/O 电源
273	vdd_io	—	AR25	P	数字 I/O 电源
274	vss_core	—	A3	P	数字核地
275	vss_core	—	B10	P	数字核地
276	vss_core	—	D20	P	数字核地
277	vss_core	—	F16	P	数字核地
278	vss_core	—	F24	P	数字核地
279	vss_core	—	H2	P	数字核地
280	vss_core	—	J31	P	数字核地
281	vss_core	—	N31	P	数字核地
282	vss_core	—	R3	P	数字核地
283	vss_core	—	U33	P	数字核地
284	vss_core	—	Y6	P	数字核地
285	vss_core	—	AD34	P	数字核地
286	vss_core	—	AF2	P	数字核地
287	vss_core	—	AF30	P	数字核地
288	vss_core	—	AG5	P	数字核地
289	vss_core	—	AL21	P	数字核地
290	vss_core	—	AL25	P	数字核地

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
291	vss_core	—	AN1	P	数字核地
292	vss_core	—	AN9	P	数字核地
293	vss_core	—	AP14	P	数字核地
294	vss_io	—	A7	P	数字 I/O 地
295	vss_io	—	A29	P	数字 I/O 地
296	vss_io	—	C3	P	数字 I/O 地
297	vss_io	—	C23	P	数字 I/O 地
298	vss_io	—	D34	P	数字 I/O 地
299	vss_io	—	F14	P	数字 I/O 地
300	vss_io	—	F18	P	数字 I/O 地
301	vss_io	—	G5	P	数字 I/O 地
302	vss_io	—	L1	P	数字 I/O 地
303	vss_io	—	M30	P	数字 I/O 地
304	vss_io	—	P34	P	数字 I/O 地
305	vss_io	—	U3	P	数字 I/O 地
306	vss_io	—	W1	P	数字 I/O 地
307	vss_io	—	W35	P	数字 I/O 地
308	vss_io	—	AA33	P	数字 I/O 地
309	vss_io	—	AC3	P	数字 I/O 地
310	vss_io	—	AG33	P	数字 I/O 地
311	vss_io	—	AK26	P	数字 I/O 地
312	vss_io	—	AL7	P	数字 I/O 地
313	vss_io	—	AL23	P	数字 I/O 地
314	vss_io	—	AL31	P	数字 I/O 地
315	vss_io	—	AM12	P	数字 I/O 地
316	vss_io	—	AM18	P	数字 I/O 地
317	vss_io	—	AN3	P	数字 I/O 地
318	N/C	—	A35	—	N/C
319	N/C	—	B32	—	N/C
320	N/C	—	B34	—	N/C
321	N/C	—	C33	—	N/C
322	N/C	—	E5	—	N/C
323	N/C	—	E31	—	N/C
324	N/C	—	F6	—	N/C
325	N/C	—	F10	—	N/C
326	N/C	—	F12	—	N/C
327	N/C	—	F22	—	N/C

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
328	N/C	—	F30	—	N/C
329	N/C	—	F32	—	N/C
330	N/C	—	G7	—	与盖板相连，板级设计时接数字地
331	N/C	—	G9	—	N/C
332	N/C	—	G11	—	N/C
333	N/C	—	G13	—	N/C
334	N/C	—	G15	—	N/C
335	N/C	—	G17	—	N/C
336	N/C	—	G19	—	N/C
337	N/C	—	G21	—	N/C
338	N/C	—	G23	—	N/C
339	N/C	—	G25	—	N/C
340	N/C	—	G27	—	N/C
341	N/C	—	G29	—	N/C
342	N/C	—	H4	—	N/C
343	N/C	—	H6	—	N/C
344	N/C	—	J7	—	N/C
345	N/C	—	J29	—	N/C
346	N/C	—	L7	—	N/C
347	N/C	—	L29	—	N/C
348	N/C	—	N7	—	N/C
349	N/C	—	N29	—	N/C
350	N/C	—	P30	—	N/C
351	N/C	—	R7	—	N/C
352	N/C	—	R29	—	N/C
353	N/C	—	T6	—	N/C
354	N/C	—	U7	—	N/C
355	N/C	—	U29	—	N/C
356	N/C	—	W7	—	N/C
357	N/C	—	W29	—	N/C
358	N/C	—	AA7	—	N/C
359	N/C	—	AA29	—	N/C
360	N/C	—	AA31	—	N/C
361	N/C	—	AB30	—	N/C
362	N/C	—	AC7	—	N/C
363	N/C	—	AC29	—	N/C
364	N/C	—	AE7	—	N/C

序号	管脚名称	上拉/下拉	引出端	I/O 类型	说明
365	N/C	—	AE29	—	N/C
366	N/C	—	AF6	—	N/C
367	N/C	—	AG7	—	N/C
368	N/C	—	AG29	—	N/C
369	N/C	—	AJ7	—	N/C
370	N/C	—	AJ9	—	N/C
371	N/C	—	AJ11	—	N/C
372	N/C	—	AJ13	—	N/C
373	N/C	—	AJ15	—	N/C
374	N/C	—	AJ17	—	N/C
375	N/C	—	AJ19	—	N/C
376	N/C	—	AJ21	—	N/C
377	N/C	—	AJ23	—	N/C
378	N/C	—	AJ25	—	N/C
379	N/C	—	AJ27	—	N/C
380	N/C	—	AJ29	—	与盖板相连，板级设计时接数字地
381	N/C	—	AK6	—	N/C
382	N/C	—	AK30	—	N/C
383	N/C	—	AK32	—	N/C
384	N/C	—	AL5	—	N/C
385	N/C	—	AM6	—	N/C
386	N/C	—	AN5	—	N/C
387	N/C	—	AN33	—	N/C
388	N/C	—	AP2	—	N/C
389	N/C	—	AP34	—	N/C
390	N/C	—	AR1	—	N/C
391	N/C	—	AR35	—	N/C
注：所有 vdd_core 统称为 V_{DDD}；所有 vdd_io 统称为 V_{DDDIO}；pll_corevdd 统称为 V_{DDA}；pll_iovdd 统称为 V_{DDAIO}。					

注：(1)“上/下拉”列中的“－”表示内部 pad 无上拉/下拉；

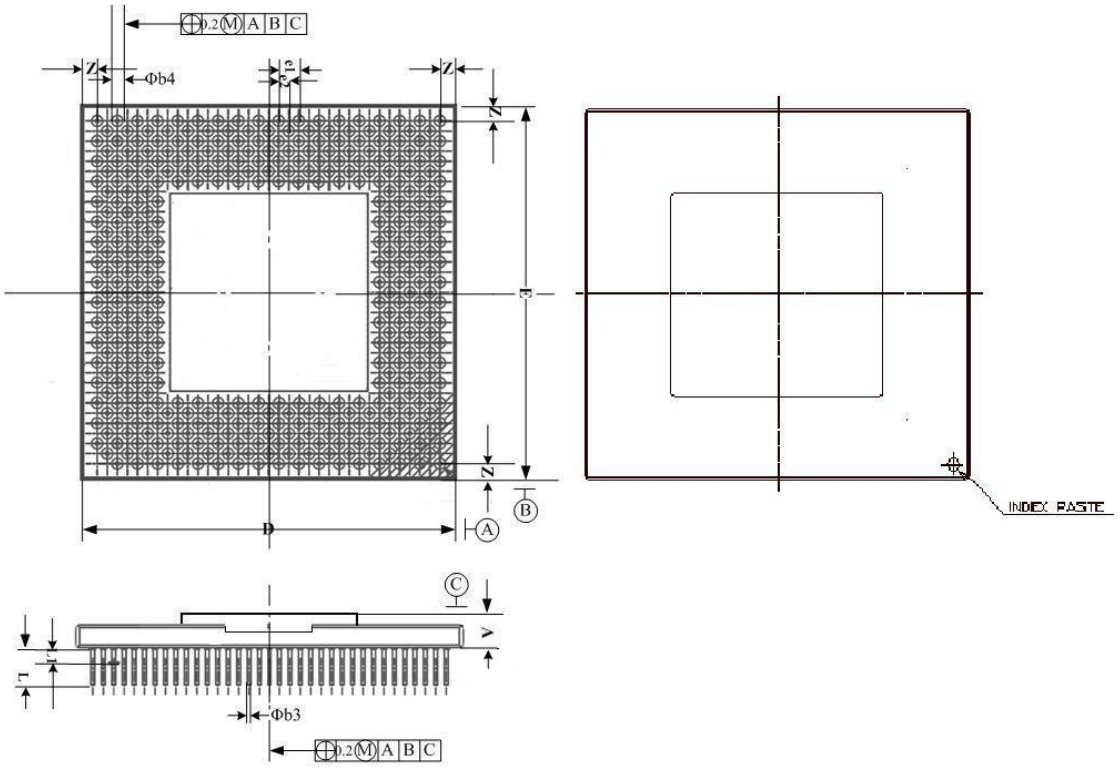
(2)“上/下拉”列中的“上拉/下拉”表示内部 pad 有上拉/下拉，上拉/下拉阻值为 15KΩ～200 KΩ。

表 3-2-2 输出驱动列表

I_{OH}	-0.5mA	-4mA	-8mA	-12mA	-16mA	无 (OD 双向)	无 (OD 输出)	-4mA (三态输出)
I_{OL}	1.5 mA	4mA	8mA	12mA	16mA	1.5mA (OD 双向)	4mA (OD 输出)	4mA (三态输出)
管脚名	pci_arb_gnt_n[6:0] pci_ad[31:0] pci_cbe_n[3:0] pci_par pci_frame_n pci_idry_n pci_trdy_n pci_stop_n pci_devsel_n pci_req_n pci_perr_n pci_rst_n	data[31:0] Pardata[6:0] ramsn[4:0] ramoen[4:0] read romsn[1:0] iosn rts3 timer0 timer1 pll_out	address[27:0] oen writen rwen[3:0] sdcsn[1:0] sddqm[3:0] sdrasn sdcasn sdwen pio[15:0]	txd3 dsutx dsuact	Sdclk	pci_serr_n pci_inta_n	errorn wdogn	Tdo

3.3 外形尺寸说明

BM3803FMGRH 采用 391 引出端交错陶瓷针栅阵列封装（CPGA391），外壳外形及尺寸符合下图 3-3-1 规定。



单位：mm

尺寸符号	数　值		
	最　小	公　称	最　大
A	1.78	2.74	3.68
φb3	0.41	0.46	0.51
φb4	1.30	1.40	1.50
e1	——	2.54	——
e2	——	1.27	——
L_1	1.02	1.27	1.52
L	2.54	4.57	5.08
D	46.76	47.24	47.72
E	46.76	47.24	47.72
Z	1.91	2.03	2.54

图 3-3-1 外形尺寸图

四、 产品功能

BM3803FMGRH 处理器是基于 SPARC V8 体系结构的抗辐射 32 位 RISC 嵌入式处理器，内核包含整数处理单元、浮点处理单元、独立的指令和数据 Cache、硬件乘/除法器，外设包括中断控制器、定时器、看门狗、UART、通用 I/O 接口、外部存储器控制器、PCI 总线控制器等，能够满足空间恶劣环境应用的抗辐射、长寿命、高可靠指标要求，与存储器和相关外围电路相连，可构成完整的嵌入式单板计算机系统。

4.1 产品结构框图

结构框图如图 4-1-1 所示。

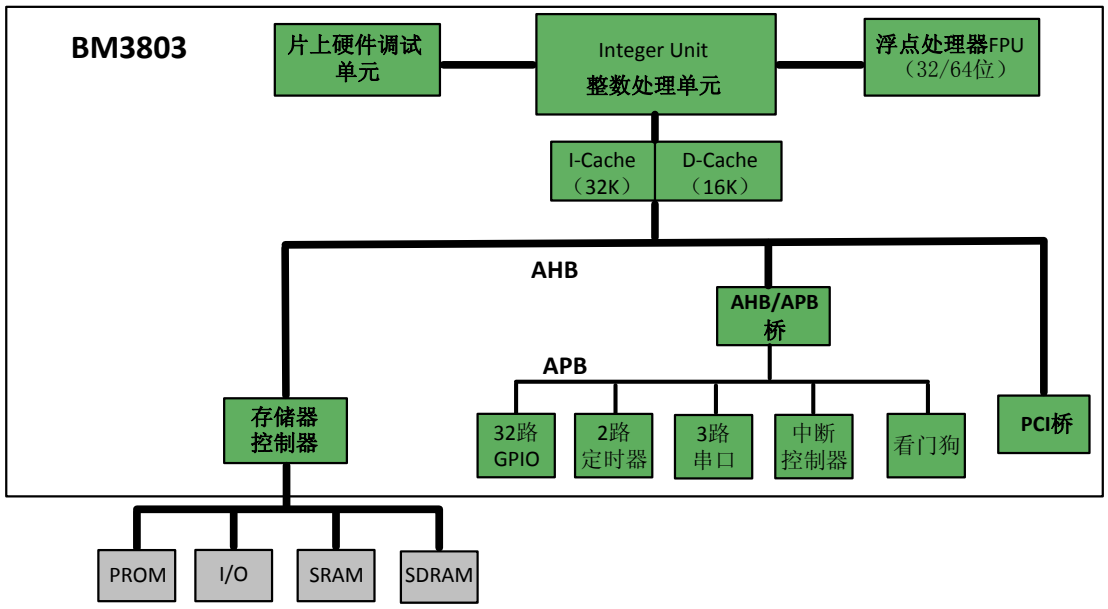


图 4-1-1 结构框图

4.2 处理器核心

BM3803FMGRH 处理器核心的主要功能是确保程序的正确执行。BM3803FMGRH 处理器核心具有两种状态：用户态和管理态。通过**处理器状态寄存器**中 *S* 字段可配置处理器处于用户态还是管理态。在管理态下，处理器能够执行包括特权指令在内的所有指令；在用户态下，试图执行特权指令将会触发管理软件的陷阱。

本节除了介绍处理器核心部件整数单元（IU）和浮点处理单元（FPU）之外，还介绍了

处理器的启动方式、复位、异常和容错处理。

4.2.1 整数单元

BM3803FMGRH 包含一个整数单元 IU，IU 有自己独立的 32 位寄存器。

IU 相关的寄存器说明详见附录 3.1，状态寄存器说明详见附录 3.3。

IU 实现了 SPARC V8 标准定义的整数指令集，其特性如下：

- **5 级指令流水：**

IU 单元使用 5 级指令流水线，以接近每时钟周期执行一条指令的速率执行指令；

- **8 个寄存器窗口；**
- **硬件乘/除法器：**

可以实现两个 32 位相乘，得到 64 位乘积，32/64 位除以 32 位，得到 32 位商；

- **寄存器三模冗余结构，寄存器堆 EDAC 容错。**

4.2.2 指令集

BM3803FMGRH 指令分为 6 种功能类别：加载/存储，算术/逻辑/移位，控制转移，读/写控制寄存器，浮点操作及其它指令。指令的执行周期见下表 4-2-1。

表 4-2-1 IU 指令执行周期（假设 Cache 命中）

指令名称	执行周期
JMPL（跳转指令）	2
LDD（双字加载指令）	2
ST（单字存储指令）	2
STD（双字存储指令）	3
SMUL/UMUL（有符号/无符号整数乘法指令）	4
SDIV/UDIV（有符号/无符号整数除法指令）	35
TICC（陷阱指令）	4
LDSTUB（原子加载/存储指令）	3
其他指令	1

4.2.3 浮点处理单元

BM3803FMGRH 包含一个浮点处理单元 FPU，FPU 有自己独立的 32 位寄存器。

FPU 实现了 SPARC V8 标准定义的浮点指令。FPU 可以处理单精度数和双精度数，但不支持四精度浮点指令操作。浮点指令的执行与 IU 串行，当 FPU 中指令没有执行完时，IU 处于等待状态。

浮点寄存器说明详见附录 3.2，状态寄存器说明详见附录 3.3。

4.2.4 工作模式

BM3803FMGRH 有两种工作模式：运行模式和调试模式。

- **运行模式**

dsuen 和 dsubre 管脚设置为低电平，复位后进入运行模式，从地址 0x0 开始执行程序。0x0 地址属于 PROM，需要通过 pio[1：0]正确配置 PROM 宽度：“00”表示 8 位，“01”表示 16 位，“1x”表示 32 位。

当使用带 EDAC 功能的 8 位 PROM 启动时，因为校验位的空间起始于 PROM Bank 的 3/4 地址处，所以需要在启动前确定 PROM 空间范围，即需要使用外部管脚 rombsd[3:0]来定义 PROM Bank 大小。

- **调试模式**

dsuen 和 dsubre 管脚设置为高电平，复位后从地址 0x0 开始执行 4 条指令后进入调试模式，此时可以通过调试串口调试 BM3803FMGRH。

4.2.5 产品复位

BM3803FMGRH 为低电平复位。在时钟有效后 resetn 信号保持 10 个时钟周期的低电平，就可以完成复位。复位信号拉高后，经过 3 个 sdclk 时钟周期的上升沿，处理器给出 PROM 地址信号，再经过 1 个 sdclk 时钟周期的上升沿，处理器给出 PROM 片选和读使能信号。

4.2.6 产品异常

BM3803FMGRH 的陷阱以及优先级见表 4-2-2。

表 4-2-2 BM3803FMGRH 的陷阱

陷阱	陷阱类型	优先级	说明
Reset	0x00	1	上电复位
write_error	0x2B	2	写错误
instruction_access_error	0x01	3	无法被纠正的取指错误
privileged_instruction	0x03	4	用户模式下执行特权指令
illegal_instruction	0x02	5	UNIMP 或其他未实现的命令
fp_disabled	0x04	6	不使能 FPU 时执行浮点指令
cp_disabled	0x24	6	不使能协处理器时执行协处理器指令
watchpoint_detected	0x0B	7	探测到观察点
Window_overflow	0x05	8	SAVE 进入无效窗口
window_underflow	0x06	8	RESTORE 进入无效窗口
register_hardware_error	0x20	9	检测到不可纠正的寄存器堆错误
mem_address_not_aligned	0x07	10	访问存储器时地址未对齐
fp_exception	0x08	11	FPU 异常
data_access_exception	0x09	13	LOAD 指令访问错误
tag_overflow	0x0A	14	Tagged 运算溢出
divide_exception	0x2A	15	被 0 除
trap_instruction	0x80~0xFF	16	软件异常指令 TA
interrupt_level_15	0x1F	17	异步中断 15
interrupt_level_14	0x1E	18	异步中断 14
interrupt_level_13	0x1D	19	异步中断 13
interrupt_level_12	0x1C	20	异步中断 12
interrupt_level_11	0x1B	21	异步中断 11
interrupt_level_10	0x1A	22	异步中断 10
interrupt_level_9	0x19	23	异步中断 9
interrupt_level_8	0x18	24	异步中断 8

interrupt_level_7	0x17	25	异步中断 7
interrupt_level_6	0x16	26	异步中断 6
interrupt_level_5	0x15	27	异步中断 5
interrupt_level_4	0x14	28	异步中断 4
interrupt_level_3	0x13	29	异步中断 3
interrupt_level_2	0x12	30	异步中断 2
interrupt_level_1	0x11	31	异步中断 1

4.2.6 容错设计

容错设计采用三模冗余和 EDAC 技术，对 IU 内部寄存器采用三模冗余结构，对寄存器堆 Regfile 中的数据采用 Hamming 码进行 EDAC 保护，可以检测两位错，纠正一位错。

当检测到可以纠正的数据错误时，流水线断流，将纠正后的数据按原读出地址写回 Regfile，然后重启流水线，从断流位置重新运行程序；当检测到不可纠正的数据错误时，采用了陷阱方式处理，引起 0x20 陷阱。

ASR16 和 ASR17 控制 Regfile 的 EDAC 功能，相关寄存器说明详见附录 3.3。

4.3 Cache

4.3.1 概述

BM3803FMGRH 的处理器内核采用哈佛结构，具有独立的指令 Cache 和数据 Cache。为了提高处理器内核的速度，指令 Cache 和数据 Cache 都采用了多路组相联（multi-set-caches）技术。Cache 的替换算法采用 LRU 算法。数据 Cache 容量为 16KB，指令 Cache 容量为 32KB。

4.3.2 Cache 缓存操作

外部存储器的部分区域能通过 Cache 缓存，可缓存的地址空间包括 PROM 和 RAM 区域，见表 4-3-1。

表 4-3-1 能否通过 Cache 缓存的地址空间列表

地址空间	区域	能否通过Cache缓存
0x0000-0000 ~ 0x1FFF-FFFF	PROM区域	能

0x2000-0000 ~ 0x3FFF-FFFF	IO空间区域	否
0x4000-0000 ~ 0x7FFF-FFFF	RAM区域	能
0x8000-0000 ~ 0xFFFF-FFFF	其他存储区域	否

Cache 在使用时通过使用不同的 ASI 映射表示不同的 Cache 操作，ASI 映射方式见表 4-3-2。

表 4-3-2 ASI 映射情况说明

ASI	Usage
0x0, 0x1, 0x2, 0x3	强制Cache不命中，若数据被Cache缓存过则更新Cache，若数据不在Cache中但地址可缓存则分配新Cache块进行替换
0x4, 0x7	强制Cache不命中，若数据被Cache缓存过则更新Cache
0x5	刷新指令Cache
0x6	刷新数据Cache
0x8, 0x9, 0xA, 0xB	正常Cache访问
0xC	访问指令Cache tags
0xD	访问指令Cache data
0xE	访问数据Cache tags
0xF	访问数据Cache data

利用 LDA/STA 指令，可以访问不同的地址空间。如果 Cache 使能，通过 ASI 0x8 - ASI 0xB 可以正常访问指令和数据；通过 ASI 0xC - ASI 0xF 可以诊断指令 Cache 和数据 Cache。指令 Cache 采用 4 路组相联方式，每组（set）的大小为 8KB。数据 Cache 采用 2 路组相联方式，每组（set）的大小为 8KB。通过 ASI 0xC - ASI 0xF 访问指令 Cache 和数据 Cache 时，除了要给 ASI 命令提供物理地址外，还需提供 set 的值，具体公式如下所示：

$$\textbf{ASI命令使用的地址} = (\textbf{set值} \ll \textbf{13}) + (\textbf{物理地址低13位})$$

其中，“set 值 ≪ 13”是 set 空间的基地址，“物理地址低 13 位”是偏移地址。对于数据 Cache，set 值可以为“0”或“1”，对于指令 Cache，set 值可以为“0”或“1”或“2”或“3”。

从数据 Cache 中读取 tag 和 data 时，需分别读 set 为“0”和“1”两种情况下的结果，其中有且仅有一种情况下结果正确。

从指令 Cache 中读取 tag 和 data 时，需分别读 set 为“0”、“1”、“2”和“3”四种情况下的结果，其中有且仅有一种情况下结果正确。

4.3.3 Cache 的容错功能

Cache 采用了奇偶校验的容错策略，当奇偶校验检测出数据/指令出错时，将直接从外部存储器中读取数据/指令，并将正确的数据写入 Cache 中。

Cache 容错功能的诊断主要通过 **Cache 容错控制寄存器**实现。主要功能包括对不同 Cache 区域的造错以及错误计数，测试 Cache 容错功能的正确性。

4.3.4 Cache 的寄存器地址映射

Cache 寄存器地址映射见表 4-3-3，详细说明见附录 3.5。

表 4-3-3 Cache 寄存器地址映射

寄存器名称	地址	复位值
Cache 控制寄存器	0x80000014	0xfd0000X0(1)
Cache 容错控制寄存器 1	0x80000110	0x20000000
Cache 容错控制寄存器 2	0x80000114	0x00000000
Cache 容错控制寄存器 3	0x80000118	0x00000000

注：(1)“复位值”按照十六进制统一写法“0xfd0000X0”，“0x”表示 16 进制，每 4bit 中有 1bit 不定值，即写为“X”，4bit 全部为固定值则写为相应 16 进制数值。

⚠注意 Cache 在使能前要首先对 Cache 进行刷新，防止由于之前 Cache 里已经缓存的内容导致指令或数据出错；Cache 刷新完成以后，**Cache 控制寄存器**的 *IP* 和 *DP* 位变为’0’，此时才可以对 Cache 进行操作。

4.4 定时器单元

4.4.1 预分频器

BM3803FMGRH 的定时器单元包含两个普通定时器和一个看门狗定时器。这三个定时器均为 24 位定时器，共用同一个 10 位预分频器。预分频器由系统时钟控制，系统时钟 clk 上升沿到达时，预分频器的计数值减 1，判断预分频器的计数值是否溢出，如果溢出，**定时器预分频重载寄存器**自动重载计数值，同时为定时器和看门狗产生一个计数 TICK 信号。预分

频器有效的分频值等于预分频重载计数值加 1。

两个普通定时器和看门狗定时器共用同一个减一计数器，预分频器重载计数值最小为 3。

4.4.2 定时器

定时器 0、定时器 1 分别包含 3 个寄存器：计数寄存器，重载寄存器，控制寄存器。**定时器控制寄存器**通过设置使能位 *EN* 来控制定时器是否使能。当预分频器产生 TICK 信号时，定时器的计数值减 1，定时器下溢时，产生中断，该中断能够被**中断屏蔽和优先级寄存器**屏蔽。此时，如果设置**定时器控制寄存器**的自动重载使能位 *RL* 为 1，定时器将重新装载寄存器值，定时器继续工作；如果 RL 为 0，定时器停止工作。通过设置**定时器控制寄存器**中的 *LD* 位，定时器可在任何时候强制重载计数值。定时器有效的分频值等于定时器重载计数值加 1。

4.4.3 看门狗

看门狗定时器在系统复位后，处于不使能状态，通过设置**定时器 0 控制寄存器**中的 *WTDEN* 使能位使能看门狗，当预分频器产生 TICK 信号时，**看门狗计数值寄存器**的计数值减 1，计数值减为 0 时，看门狗输出的外部信号 WDOG 为低电平。系统应在看门狗计数值减为 0 之前，更新**看门狗计数值寄存器**的计数值。看门狗有效的分频值等于看门狗计数值寄存器值加 1。

4.4.4 定时器和看门狗的寄存器地址映射

定时器和看门狗的寄存器地址映射如表 4-4-1 所示。

表 4-4-1 定时器和看门狗寄存器地址映射对照表

寄存器名称	地址	复位值(1)
定时器 0 计数寄存器	0x80000040	0x00XXXXXX
定时器 0 重载寄存器	0x80000044	0x00XXXXXX
定时器 0 控制寄存器	0x80000048	0x000000XX
看门狗计数值寄存器	0x8000004C	0x00FFFFFF
定时器 1 计数寄存器	0x80000050	0x00XXXXXX
定时器 1 重载寄存器	0x80000054	0x00XXXXXX
定时器 1 控制寄存器	0x80000058	0x0000000X
定时器预分频计数寄存器	0x80000060	0x00000000
定时器预分频重载寄存器	0x80000064	0x000003FF

注(1)：“复位值”按照十六进制统一写法“0xXXX0XXXX”，“0x”表示 16 进制，每 4bit 中有 1bit 不定

值，即写为“X”，4bit 全部为固定值则写为相应 16 进制数值。

寄存器说明详见附录 3.8。

4.5 GPIO

4.5.1 概述

通用接口 GPIO 实现了 32 路供用户定义使用的双向 IO 接口，分为高 16 位和低 16 位，其中高 16 位与数据总线复用，低 16 位部分端口与 UART、PROM 等功能复用。

4.5.2 GPIO 低 16 位

GPIO 低 16 位对应 PIO[15:0]，可与其他功能复用。当作为通用 I/O 使用时，每个端口都可读/写且相互独立。PIO[15:0]的功能通过两个寄存器控制：**GPIO 数据寄存器**和 **GPIO 方向寄存器**，**GPIO 方向寄存器**中 *IODIR* 控制每个端口的方向，若 *IODIRx* 位置 1，则相应管脚配置成输出，置 0 则为输入。当管脚配置成输入时，读 **GPIO 数据寄存器**中 *IODATx* 位，则返回当前管脚值；当配置成输出时，若 *IODATx* 位置 1，端口 x 被拉高，若 *IODATx* 位置 0，端口 x 被拉低。

当配置 *IODIRx* 值，使得端口 x 从输入切换到输出模式时，相应管脚立即被 *IODATx* 驱动。当配置 *IODIRx* 位使得端口 x 从输出切换到输入模式时，管脚值立即被写入 *IODATx*。

4.5.3 GPIO 高 16 位

GPIO 高 16 位对应数据总线 D[15:0]。仅当外部存储器（PROM，RAM，I/O 空间）数据配置为 8 位或 16 位时，GPIO 高 16 位才会用到。若 SDRAM 控制器使能，则高 16 位不可用。

高 16 位仅能配置成基于字节的输出或输入。当 D[7:0]作低字节时，D[15:8]作为中间字节。

通过写 **GPIO 方向寄存器**中的 *MEDDIR* 和 *LOWDIR*，可改变 GPIO 高 16 位的方向，若 *MEDDIR* 或 *LOWDIR* 为 1，则 D[15:0]中相应的字节配置成输出，若为 0，则配置成输入。

高 16 位的数据 *MEDDAT* 和 *LOWDAT* 通过 **GPIO 数据寄存器**读写。

当配置成输入时，读 **GPIO 数据寄存器**的 *MEDDAT*，返回 D[15:8]当前值，读 **GPIO 数据寄存器**的 *LOWDAT*，返回 D[7:0]当前值；当配置成输出时，*MEDDAT* 的值成为 D[15:8]的总线输

出，*LOWDAT*的值成为 D[7:0]的总线输出。

当改变**GPIO 方向寄存器**中 *MEDDIR* 或 *LOWDIR* 值时，若 *MEDDAT/LOWDAT* 从输入切换到输出，*MEDDAT/LOWDAT* 的值会立即驱动至相应引脚；当从输出切换到输入时，引脚值立即被更新至**GPIO 数据寄存器**的 *MEDDAT/LOWDAT*。

4.5.4 复用功能

低 16 位 GPIO 除了作为通用 I/O，部分端口可复用为其他功能。处于输入状态时，管脚上的信号同时送入 GPIO 模块和复用功能模块。处于输出状态时，若复用功能使能，输出到管脚上的信号来自于复用功能模块，若复用功能不使能，输出管脚上的信号来自于 GPIO 模块。复用功能如表 4-5-1 所示。

表 4-5-1 GPIO 管脚复用说明

GPIO 管脚	复用管脚	类型	复用管脚说明	复用管脚使能条件
pio[15]	txd1	O	UART1 发送数据	UART1 发送使能
pio[14]	rxd1	I	UART1 接收数据	UART1 接收使能
pio[13]	rts1	O	UART1 请求发送	UART1 流控使能
pio[12]	cts1	I	UART1 清除发送	UART1 流控使能
pio[11]	txd2	O	UART2 发送数据	UART2 发送使能
pio[10]	rxd2	I	UART2 接收数据	UART2 接收使能
pio[9]	rts2	O	UART2 请求发送	UART2 流控使能
pio[8]	cts2	I	UART2 清除发送	UART2 流控使能
pio[3]	uart clk	I	可选 UART 时钟	串口外部时钟使能
pio[2]	prom edac	I	PROM EDAC 使能	上电复位时，该管脚值会配置给 MECFG1 的 romEEn，为 1 表示 PROM EDAC 使能，为 0 表示 PROM EDAC 不使能。复位后该管脚作为正常 IO 使用。

pio[1:0]	prom width	I	定义启动 PROM 宽度	从外部 PROM 启动时，通过设置 PIO[1:0]来配置 PROM 存储器的数据宽度。00:8bit， 01:16bit，1X:32bit。启动后该管脚作为正常 IO 使用。

4.5.5 GPIO 中断

GPIO 端口可以配置成输入，用于捕获外部器件中断，通过设置 **GPIO 中断控制寄存器**，可支持四个中断。

通过读/写 **GPIO 数据寄存器**完成数据读/写操作，通过配置 **GPIO 方向寄存器**完成端口输入/输出方向设置，通过设置 **GPIO 中断控制寄存器**完成 32 路端口的中断使能和状态配置。**GPIO 中断控制寄存器**每 8bit 一组，共分 4 组：[7:0]、[15:8]、[23:16]、[31:24]，每组功能包括中断使能位 *EN*、边沿/电平触发方式位 *LE*、极性位 *PL*、端口选择位 *ISEL*，其中端口选择共 5 位信号，用于选择 32 位 GPIO 端口中任意一路相应管脚作为中断输入。

⚠注意 不要用同一路 GPIO 管脚同时触发多个 GPIO 中断，会给中断处理带来麻烦。

当 GPIO 中断采用电平触发方式时，以低电平为例，只要相应的 GPIO 端口接收到低电平，就会产生中断请求，处理器响应中断后，会自动清除中断请求，但如果此时 GPIO 端口仍接收到低电平，会再次产生中断请求，此时，用户中断服务程序中须增加清除中断请求操作，即写**中断清除寄存器**相应位，且要确保清除中断请求操作前 GPIO 端口的低电平撤销。

4.5.6 GPIO 寄存器地址映射

表 4--5-2 GPIO 寄存器地址映射

寄存器名称	地址	复位值(1)
GPIO 数据寄存器	0x800000A0	0xXXXXXXXX
GPIO 方向寄存器	0x800000A4	0x00000000

GPIO 中断控制寄存器	0x800000A8	0xXXXXXXXX

注(1)：“复位值”按照十六进制统一写法“0xXXX0XXXX”，“0x”表示 16 进制，每 4bit 中有 1bit 不定值，即写为“X”，4bit 全部为固定值则写为相应 16 进制数值。

寄存器说明详见附录 3.11。

4.6 中断和陷阱

4.6.1 概述

BM3803FMGRH 支持异常处理机制，即陷阱处理。陷阱分为精确陷阱、延迟陷阱和中断。精确陷阱和延迟陷阱是由与特定指令相关的硬件引起，在引起陷阱产生的指令执行期间发生，即同步陷阱；中断是由外部事件引起的，中断的产生不与任何特定的指令相关，并在指令执行完成后发生。

陷阱被几个寄存器控制：**处理器状态寄存器** PSR 的陷阱使能位 *ET* 控制陷阱和外部中断请求，PSR 的处理器中断等级域 *PIL* 指定可响应的外部中断请求级别，**浮点状态寄存器** FSR 的陷阱使能屏蔽域 *TEM* 控制浮点陷阱请求。

处理器状态寄存器 PSR 中的 *ET* 位必须为 1，陷阱才会被响应。*ET*=1 时，IU 在执行指令期间根据未决的陷阱优先次序来响应陷阱，只有最高优先级的陷阱请求被响应。当在同一时钟周期内发生多个同步陷阱时，具有最高优先级的陷阱被响应，而较低优先级陷阱被忽略，而当有多个未处理的外部中断请求时，较低优先级的中断请求会被保持。

- **ET 和 PIL 控制**

陷阱要正常响应，PSR 中的 *ET* 位必须为 1。

ET=1，如果发生陷阱，处理器按照陷阱处理程序响应陷阱。*ET*=0，如果发生陷阱，处理器进入错误模式。

IU 根据陷阱优先级区分陷阱响应次序，只响应最高优先级的陷阱。当处理最高优先级中断时，低优先级中断请求保持在**中断请求寄存器**中。

ET=1，如果发生中断，处理器比较 PSR 中的 *PIL* 字段和**中断级别/屏蔽控制寄存器**中的 *ILEVEL* 和 *IMASK* 字段，如果中断请求的优先级别 *ILEVEL* 和中断序号最高，且中断序号大于 *PIL* 字段值，且**中断级别/屏蔽控制寄存器**中的 *IMASK* 字段为 1，即未屏蔽此中断，则处理器

响应该中断。*ET*=0，所有中断被屏蔽。

◆ TEM 控制

如果浮点陷阱发生时，相应**浮点状态寄存器** FSR 中的 *TEM* 字段置 1，则使能浮点陷阱，此时若 *ET*=1，则处理器按照浮点陷阱处理程序响应浮点陷阱，且引起陷阱的指令的目的寄存器、FSR 的 *FCC* 和 *AEXC* 字段保持不变。如果 *TEM* 字段置 0，则屏蔽浮点陷阱，此时，产生的陷阱会被记录在 FSR 的 *AEXC* 字段中，且引起陷阱的指令的目的寄存器、FSR 的 *FCC* 字段被更新。

4.6.2 陷阱类型

陷阱类型符合 SPARC V8 规范，主要陷阱见表 4-6-1。

表 4-6-1 陷阱类型

陷阱	TT	描述	优先级
reset	0x00	复位	1
write_error	0x2b	写错误	2
instruction_access_exception	0x01	无法被纠正的取指错误	3
privileged_instruction	0x03	用户模式下执行特权指令	4
illegal_instruction	0x02	UNIMP或其他未实现的命令	5
fp_disabled	0x04	不使能FPU时执行浮点指令	6
cp_disabled	0x24	不使能协处理器时执行协处理器指令	6
watchpoint_detected	0x0B	探测到观察点	7
window_overflow	0x05	SAVE进入无效窗口	8
window_underflow	0x06	RESTORE进入无效窗口	8
register_hardware_error	0x20	检测到不可纠正的寄存器堆错误	9
mem_address_not_aligned	0x07	访问存储器时地址未对齐	10
fp_exception	0x08	FPU陷阱	11
data_access_exception	0x09	load指令访问错误	13
tag_overflow	0x0A	Tagged运算溢出	14
divide_exception	0x2A	被0除	15
trap_instruction	0x80 -0xFF	软件异常指令(TA)	16

陷阱描述：

◆ reset：复位陷阱由外部复位请求引起。它使得处理器在实际地址 0x0 开始执行。复位陷阱产生后，将使得 PSR 中的 *ET* 和 *S* 位分别被初始化为 0

和 1。

◆ write_error：数据存储到存储器时发生错误。

◆ instruction_access_exception：无法被纠正的取指错误。

◆ privileged_instruction：当 PSR 中的管理位为 0（即处于用户模式）而执行特权指令时产生的异常。

◆ illegal_instruction：执行带有未实现操作数的指令，或者 UNIMP 指令以及会引起处理器非法状态的指令时发生的陷阱。

◆ fp-disabled：当 FPU 无效或者不存在时执行 FPU 指令产生的陷阱。

◆ cp-disabled：当协处理器无效或者不存在时执行协处理器指令产生的陷阱。

◆ watchpoint_detected：指令返回存储器地址或加载/存储数据存储器地址，这些地址与以前加载的实现相关的观察点寄存器的内容相匹配。

◆ window_overflow：SAVE 指令使得当前窗口指针 CWP 指向无效窗口时引起的陷阱。

◆ window_underflow：RESTORE 或 RETT 指令使得 CWP 指向无效窗口引起的陷阱。

◆ register_hardware_error：检测到不可纠正的寄存器堆错误，在读寄存器时发生。

◆ mem_address_not_aligned：加载/存储指令产生与指令地址未正确对齐的存储器地址或 JMPL，RETT 指令产生字未对齐地址时产生的陷阱。

◆ fp_exception：当陷阱使能屏蔽 *TEM* 为 1 时，FPU 指令产生 IEEE_754 陷阱，或 FPU 指令未实现，或 FPU 指令不完整，或在 FPU 中存在序列或者硬件错误。

◆ data_access_exception：执行加载数据指令时，发生数据访问错误而产生的陷阱。

◆ tag_overflow：执行标记的算术指令并且算术上溢发生，或操作数的标记位至少有一位不为 0 时，产生的陷阱。

◆ divide_exception：执行整数除指令时被 0 除产生的陷阱。

◆ trap_instruction：软件陷阱，执行 Ticc 指令并且陷阱条件为真时产生的陷阱。

当在同一时钟周期内发生多个同步陷阱时，具有最高优先级的陷阱被命中，而较低优先级陷阱被忽略。

在**浮点状态寄存器** FSR 中，有 IEEE_754 浮点数异常相关字段（Floating-Point Exception Fields），实现如下：

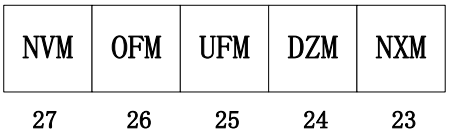


图 4-6-1 FSR 中的陷阱允许屏蔽字段 *TEM*

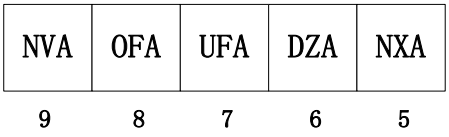


图 4-6-2 FSR 已发生的异常位字段 *AEXC*

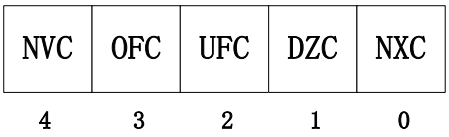


图 4-6-3 FSR 当前异常位字段 *CEXC*

FSR_invalid(NVC,NVA)：对于将要执行的操作，操作数为非法操作数。比如，0÷0 和∞－∞运算都是无效运算。该位为 1 表示非法操作数，为 0 表示合法操作数。

FSR_overflow(OFC,OFA)：在某种数据格式中，舍入后的结果可能会在数量级上大于正常数值的上限。1 表示上溢，0 表示未上溢。

FSR_underflow(UFC,UFA)：在某种数据格式中，舍入后的结果可能会不精确而且会在数量级上小于正常数值的下限。1 表示下溢，0 表示未下溢。

FSR_division-by-zero(DZC,DZA)：X÷0 ，X 为非正常操作数或者正常操作数时，对该位置位。注意 0÷0 运算不会将 DZC 置位。DZC 在除数为 0 被置 1，除数不为 0 时置 0。

FSR_inexact(NXC,NXA)：舍入后的结果与正确的无限精确结果不同时进行置位。置 1 表示不精确结果，置 0 表示精确结果。

FSR 的 *TEM* 位，五种浮点异常使能屏蔽位，*TEM* 字段如图 4-6-1 所示，如果浮点操作指令产生了一个或多个异常，并且对应于这一个或多个异常的 *TEM* 位置 1，则允许产生 IEEE_754 类型的浮点异常陷阱，*TEM* 置 0 可以禁止 IEEE_754 类型的陷阱。

FSR 的 *AEXC* 位，已发生异常位，如图 4-6-2 所示，当浮点陷阱被 *TEM* 屏蔽时，浮点陷阱就会累计在 *AEXC* 中。一条浮点操作指令执行时，将 *TEM* 和 *CEXC* 两个字段进行逻辑与操作，如果结果非零，将会产生 fp_exception 陷阱；否则，新的 *CEXC* 字段与 *AEXC* 字段进行或操作后将结果保留到 *AEXC* 字段中。

FSR 的 *CEXC* 位，当前异常位，如图 4-6-3 所示，表示由于执行浮点操作指令，产生了一个或者多个 IEEE_754 浮点异常。如果没有异常，对应的位会被清 0。

4.6.3 中断

BM3803FMGRH 支持 15 种中断及两个优先层次。这些中断都是由外部请求引起，与指令操作无关。中断控制器的结构框图见图 4-6-4。

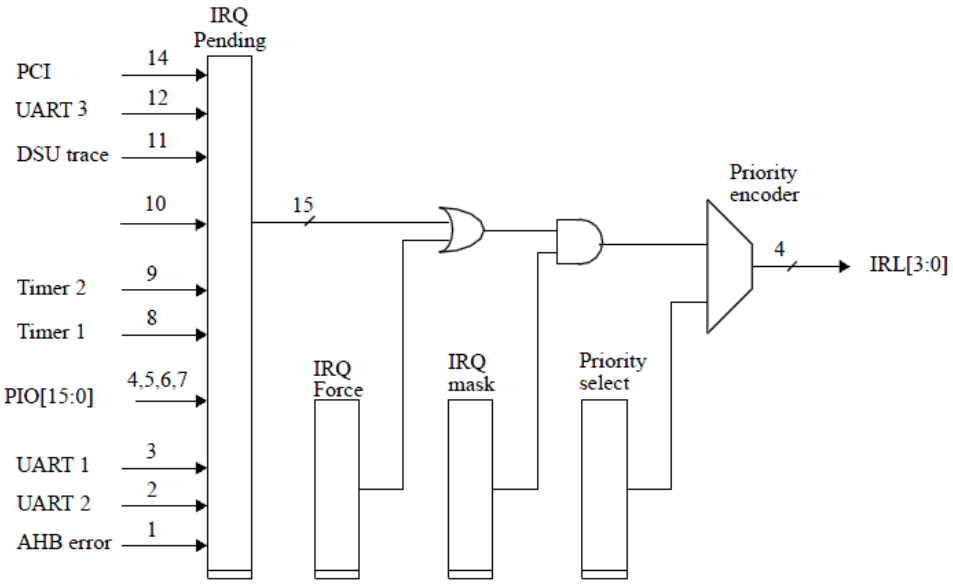


图 4-6-4 中断控制器结构框图

中断产生时，**中断请求寄存器**中的相应 *IPEND* 位被置位。中断控制器将当前优先级最高的未屏蔽的中断送给 IU。当 IU 响应中断后，则**中断请求寄存器**中的相应 *IPEND* 位自动清零。在**中断强制寄存器**中可以通过置位 *IFORCE* 强制中断。这时 IU 响应中断后，**中断强制寄存器**中的相应 *IFORCE* 位清零。

中断响应后，应用程序跳转到中断处理程序。BM3803FMGRH 硬件不支持嵌套，若应用程序需要实现中断嵌套，则必须手动修改中断处理程序的%psr.PIL 位和**中断级别/屏蔽寄存器**的 *IMASK* 位，软件实现中断嵌套。

复位后，**中断级别/屏蔽寄存器**的 *IMASK* 位清零，*ILEVEL* 位为不定值，其他的中断相关寄存器为不定值，用户在使用中断功能时，需要初始化全部中断寄存器。

4.6.3.1 中断列表

中断列表如表 4-6-2 所示。

表 4-6-2 中断列表

中断号	TT	中断源	优先级
15	0x1F	unused	17
14	0x1E	PCI 中断	18
13	0x1D	unused	19
12	0x1C	UART3	20
11	0x1B	DSU trace	21
10	0x1A	Unused	22
9	0x19	定时器 2	23
8	0x18	定时器 1	24
7	0x17	I/O中断3	25
6	0x16	I/O中断2	26
5	0x15	I/O中断1	27
4	0x14	I/O中断0	28
3	0x13	UART 1	29
2	0x12	UART 2	30
1	0x11	AHB 总线错误	31

中断寄存器映射表如表 4-6-3 所示。

表 4-6-3 中断寄存器地址映射对照表

寄存器名称	寄存器地址	复位值(1)
中断级别/屏蔽寄存器	0x80000090	0xXXXX000X
中断请求寄存器	0x80000094	0xXXXXXXXX
中断强制寄存器	0x80000098	0xXXXXXXXX
中断清除寄存器	0x8000009C	0xXXXXXXXX

注：(1)寄存器复位值按照十六进制统一写法“0xXXX0XXXX”，“0x”表示 16 进制，每 4bit 中有 1bit 不定值，即写为“X”，4bit 全部为固定值则写为相应 16 进制数值。

寄存器说明详见附录 3.10。

4.6.3.2 外部中断信号

BM3803FMGRH 在同一时刻支持四个来自 GPIO 的外部中断输入，四个外部中断的中断号是 4、5、6、7。通过 **GPIO 中断控制寄存器**进行配置，可以配置中断的触发方式：低电平、高电平、上升沿、下降沿。

GPIO 中断控制寄存器分为四组（[31:24]对应中断 7，[23:16]对应中断 6，[15:8]对应中断 5，[7:0]对应中断 4），每组八位且包含四个域：*EN*，*LE*，*PL*，*ISEL*。*EN*位被置 1，中断有效，为 0 则中断无效。*ISEL* 位定义了被选中 GPIO 的端口号码[31:0]。32 位端口包括 PIO[15:0]和 PIO[31:16]（与数据总线的 D[15:0]复用）。同时，每个 GPIO 中断的触发模式可配置，由 *LE* 和 *PL* 字段配置。具体如下表 4-6-4 所示。

表 4-6-4　GPIO 中断配置

LE	PL	GPIO中断触发模式
0	0	低电平
0	1	高电平
1	0	下降沿
1	1	上升沿

4.6.3.3 中断优先级

BM3803FMGRH 的中断优先级可以配置，**中断级别/屏蔽寄存器**的中断级别 *ILEVEL* 字段配置中断的优先级，级别 1 的优先级高于级别 0。在同一级别中，序号大的具有较高优先级，中断 15（TT=0x1F）具有最高优先级，中断 1（TT=0x11）具有最低优先级。对于 15 个中断，优先级通过 *ILEVEL* 和中断序号来决定，优先级次序为 *ILEVEL*>中断序号。

中断优先级的作用在于：若两个或两个以上中断请求同时产生时，中断控制器将当前优先级最高的未屏蔽的中断送给 IU。BM3803FMGRH 硬件不支持嵌套，若应用程序需要实现中断嵌套，则必须手动修改中断处理程序的%psr.PIL 位和**中断级别/屏蔽寄存器**的 *IMASK* 位，软件实现中断嵌套。

4.7 UART

UART 是非常灵活的串行通信模块，BM3803FMGRH 实现了三个通用异步收发器。其中两个的管脚是与 GPIO 复用。每个 UART 由发送保持寄存器、接收保持寄存器、发送移位寄存器和接收移位寄存器组成；控制主要通过控制寄存器、状态寄存器、分频寄存器、数据寄存器组成。串口数据的发送和接收通过对上述寄存器的操作实现。

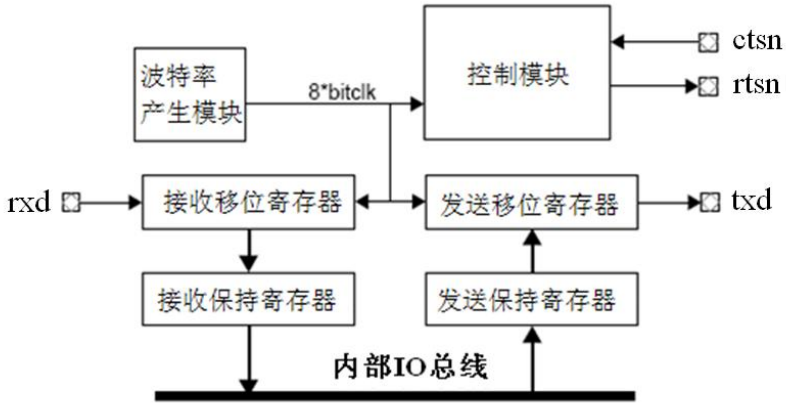


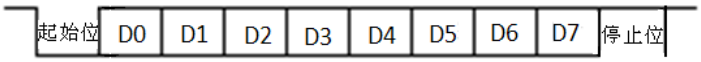
图 4-7-1 UART 结构框图

4.7.1 发送操作

在 **UART 控制寄存器**中置发送使能 *TE* 位，可以进行发送操作。当进行发送时，数据将从发送保持寄存器传输到发送移位寄存器，并在串行输出口 txd 管脚转化为串行数据流，同

时自动在每 8 位数据前增加一位起始位，在数据后增加可选的奇偶校验位和停止位。首先发送的是数据最低位。

数据帧，无奇偶校验：



数据帧，包含奇偶校验：

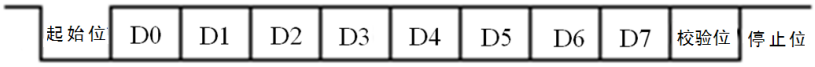


图 4-7-2　UART 数据帧格式

在停止位之后，如果在发送保持寄存器中没有出现新的字符，那么发送串行数据输出端口仍然保持高电平，并使 **UART 状态寄存器**的 *TS* 位为 1。

当在发送保持寄存器中还有待发送的字符时，将继续发送并清除 TS 位。如果发送中关闭发送功能，则将在完成本次发送后停止发送。

4.7.2 接收操作

在 **UART 控制寄存器**置 *RE* 位，可以进行接收操作。接收端等待在接收数据输入引脚上出现一个从高电平到低电平变化的起始位。若接收使能，则将在半个时钟之后采样串行输入。若串行输入采样为高，则起始位无效，将继续等待有效的起始位；若串行输入采样为低，起始位有效，接收端在每个时钟间隔内采样串行输入，接收移位寄存器串行接收到一帧数据后，将数据送入接收保持寄存器。

接收时，首先接收数据的最低位。收到的数据送入接收保持寄存器后，同时 **UART 状态寄存器**的数据准备好 *DR* 位变为 1。当接收保持寄存器和接收移位寄存器中都有未读取的数据，且又检测到新的数据时，溢出错 *OV* 位置 1，并且新数据将写入接收移位寄存器。

4.7.3 流控

若使能流控，则 ctsn 输入必须为低电平时才能进行发送。若发送过程中，ctsn 变高，将完成当前帧的发送，之后串行输出保持无效状态，直到 ctsn 重新变为低电平；接收时，当监测到有效的起始位时，rtsn 变为高电平。当从接收保持寄存器中读出数据后，并且没有要接收的数据，rtsn 变为低电平。如果 ctsn 连接到了接收端的 rtsn，则可以有效地避免数据溢出。

4.7.4 回环

若 **UART 控制寄存器**的 LB 位置位，UART 将进入回环模式。在该模式下，发送端输出在内部与接收端输入相连，从而可进行回环测试，验证接收/发送端以及相关软件的工作情况。在该模式下，输出仍然保持无效状态，不能向外发送数据。

4.7.5 波特率产生

UART 通过一个 12 位递减计数的分频器来产生需要的波特率。通过设置 **UART 控制寄存器**的外部时钟使能位 *EC*，来决定 UART 是由系统时钟还是外部时钟定时。若 *EC* 位为 0，分频器由系统时钟控制。若 *EC* 位为 1，分频器由外部时钟控制，外部时钟由 PIO[3]提供，PIO[3]的频率必须小于系统时钟频率的一半。

分频器在每次下溢时产生 UART TICK 信号，同时重载 UART 分频值。UART TICK 的频率是所需波特率的 8 倍。

波特率产生公式：

分频器重载值=（（串口时钟*10）/（波特率*8）-5）/10

4.7.6 中断产生

在以下的情况下，UART 将产生一个中断：当发送使能、发送中断使能、流控不使能，发送保持寄存器数据不为空，产生一个中断；当接收使能、接收中断使能，接收移位寄存器接收到一个完整的数据后，产生一个中断。

4.7.7 UART 使用说明

在进行串口初始化时，须对串口的状态寄存器以及控制寄存器进行配置，并读两次数据寄存器来清空数据寄存器，保证初始状态正确。若使用中断模式接收数据，且在中断屏蔽状态下接收了数据，则在串口中断服务程序中须增加两次读数据寄存器的操作，保证数据接收正确。

4.7.8 UART 的寄存器地址映射

UART 寄存器地址映射如表 4-7-1 所示。

表 4-7-1 UART 寄存器地址映射对照表

寄存器名称	地址	复位值(1)
UART1 数据寄存器	0x80000070	0x000000XX

UART1 状态寄存器	0x80000074	0x00000006
UART1 控制寄存器	0x80000078	0x000000XX
UART1 分频器重载寄存器	0x8000007C	0x00000XXX
UART2 数据寄存器	0x80000080	0x000000XX
UART2 状态寄存器	0x80000084	0x00000006
UART2 控制寄存器	0x80000088	0x000000XX
UART2 分频器重载寄存器	0x8000008C	0x00000XXX
UART3 数据寄存器	0x800000E0	0x000000XX
UART3 状态寄存器	0x800000E4	0x00000006
UART3 控制寄存器	0x800000E8	0x000000XX
UART3 分频器重载寄存器	0x800000EC	0x00000XXX

注(1)：“复位值”按照十六进制统一写法“0xXXX0XXXX”，“0x”表示 16 进制，每 4bit 中有 1bit 不定值，即写为“X”，4bit 全部为固定值则写为相应 16 进制数值。

寄存器说明详见附录 3.9。

4.8 外部存储器接口

4.8.1 概述

BM3803FMGRH 通过外部存储器控制器支持访问 PROM、存储器映射的 I/O 设备、SRAM 和 SDRAM。支持片选数量和数据位宽见表 4-8-1。

表 4-8-1 外部存储器类型表

存储器类型	片选数量	数据位宽
PROM	2	32、16、8
I/O	1	32、16、8
SRAM	5	32、16、8
SDRAM	2	32

外部存储器接口配置通过三个外部存储器配置寄存器进行：MCFG1、MCFG2、MCFG3。其中 MCFG1 主要用于配置 PROM 和 I/O 设备，MCFG2 用于配置 SRAM 和 SDRAM，MCFG3 用于配置 SDRAM 和 I/O 设备。配置寄存器说明详见附录 3.6。

4.8.2 外部存储器控制器地址空间

外部存储器的地址空间为 2G 字节，地址空间映射见表 4-8-2。

表 4-8-2 外部存储器地址空间

地址范围	大小	映射
0x0000-0000 ~ 0x1FFF-FFFF	512MB	PROM
0x2000-0000 ~ 0x3FFF-FFFF	256MB	I/O
0x4000-0000 ~ 0x7FFF-FFFF	1GB	SRAM/SDRAM

4.8.2.1 PROM 地址空间分配

PROM 最大支持 2 个片选，每个片选容量无需配置，每个片选容量最大为 256MB。

表 4-8-3 PROM 地址空间

大小	起始地址	结束地址	片选信号
256MB	0x0000-0000	0x0FFF-FFFF	romsn[0]
256MB	0x1000-0000	0x1FFF-FFFF	romsn[1]

4.8.2.2 I/O 地址空间分配

I/O 设备只支持 1 个片选，容量无需配置，片选容量最大为 256MB。

表 4-8-4 I/O 地址空间

大小	起始地址	结束地址	片选信号
256MB	0x2000-0000	0x2FFF-FFFF	iosn
256MB	0x3000-0000	0x3FFF-FFFF	iosn

注意 地址空间 0x2000-0000~0x2FFF-FFFF 同 0x3000-0000~0x3FFF-FFFF 重叠。

4.8.2.3 SRAM 和 SDRAM 空间分配

SRAM 和 SDRAM 地址分配见表 4-8-5。

表 4-8-5 SRAM 和 SDRAM 地址分配

SRAM 不使能位（*SRDIS*）	SDRAM 使能位（*SDREN*）	地址空间	RAM 类型
0 或 1	0	0x4000-0000 ~ 0x5FFF-FFFF	SRAM0、1、2、3
		0x6000-0000 ~ 0x7FFF-FFFF	SRAM4
0	1	0x4000-0000 ~ 0x5FFF-FFFF	SRAM0、1、2、3
		0x6000-0000 ~ 0x7FFF-FFFF	SDRAM0、1
1	1	0x4000-0000 ~ 0x7FFF-FFFF	SDRAM0、1

SRAM 最大支持 5 个片选，前 4 个片选容量大小可配置，范围是 8KB~256MB。第 5 个片选容量无需配置，第 5 个片选容量最大 256MB。

SDRAM 最大支持 2 个片选，每个片选容量大小可配置，范围是 4MB~512MB。

SRAM 使能、SDRAM 使能

1. SRAM 地址空间分配

表 4-8-6 SRAM 地址空间（SRAM 使能、SDRAM 使能）

SRBS 配置	大小	起始地址	结束地址	片选信号
1111	256MB	0x4000-0000	0x4FFF-FFFF	ramsn[0]
	256MB	0x5000-0000	0x5FFF-FFFF	ramsn[1]
1110	128MB	0x4000-0000	0x47FF-FFFF	ramsn[0]
	128MB	0x4800-0000	0x4FFF-FFFF	ramsn[1]
	128MB	0x5000-0000	0x57FF-FFFF	ramsn[2]
	128MB	0x5800-0000	0x5FFF-FFFF	ramsn[3]
1101	64MB	0x4000-0000	0x43FF-FFFF	ramsn[0]
	64MB	0x4400-0000	0x47FF-FFFF	ramsn[1]
	64MB	0x4800-0000	0x4BFF-FFFF	ramsn[2]

	64MB	0x4C00-0000	0x4FFF-FFFF	ramsn[3]
1100	32MB	0x4000-0000	0x41FF-FFFF	ramsn[0]
	32MB	0x4200-0000	0x43FF-FFFF	ramsn[1]
	32MB	0x4400-0000	0x45FF-FFFF	ramsn[2]
	32MB	0x4600-0000	0x47FF-FFFF	ramsn[3]
1011	16MB	0x4000-0000	0x40FF-FFFF	ramsn[0]
	16MB	0x4100-0000	0x41FF-FFFF	ramsn[1]
	16MB	0x4200-0000	0x42FF-FFFF	ramsn[2]
	16MB	0x4300-0000	0x43FF-FFFF	ramsn[3]
1010	8MB	0x4000-0000	0x407F-FFFF	ramsn[0]
	8MB	0x4080-0000	0x40FF-FFFF	ramsn[1]
	8MB	0x4100-0000	0x417F-FFFF	ramsn[2]
	8MB	0x4180-0000	0x41FF-FFFF	ramsn[3]
1001	4MB	0x4000-0000	0x403F-FFFF	ramsn[0]
	4MB	0x4040-0000	0x407F-FFFF	ramsn[1]
	4MB	0x4080-0000	0x40BF-FFFF	ramsn[2]
	4MB	0x40C0-0000	0x40FF-FFFF	ramsn[3]
1000	2MB	0x4000-0000	0x401F-FFFF	ramsn[0]
	2MB	0x4020-0000	0x403F-FFFF	ramsn[1]
	2MB	0x4040-0000	0x405F-FFFF	ramsn[2]
	2MB	0x4060-0000	0x407F-FFFF	ramsn[3]
0111	1MB	0x4000-0000	0x400F-FFFF	ramsn[0]
	1MB	0x4010-0000	0x401F-FFFF	ramsn[1]
	1MB	0x4020-0000	0x402F-FFFF	ramsn[2]
	1MB	0x4030-0000	0x403F-FFFF	ramsn[3]
0110	512KB	0x4000-0000	0x4007-FFFF	ramsn[0]
	512KB	0x4008-0000	0x400F-FFFF	ramsn[1]
	512KB	0x4010-0000	0x4017-FFFF	ramsn[2]
	512KB	0x4018-0000	0x401F-FFFF	ramsn[3]
0101	256KB	0x4000-0000	0x4003-FFFF	ramsn[0]
	256KB	0x4004-0000	0x4007-FFFF	ramsn[1]
	256KB	0x4008-0000	0x400B-FFFF	ramsn[2]
	256KB	0x400C-0000	0x400F-FFFF	ramsn[3]
0100	128KB	0x4000-0000	0x4001-FFFF	ramsn[0]
	128KB	0x4002-0000	0x4003-FFFF	ramsn[1]
	128KB	0x4004-0000	0x4005-FFFF	ramsn[2]
	128KB	0x4006-0000	0x4007-FFFF	ramsn[3]
0011	64KB	0x4000-0000	0x4000-FFFF	ramsn[0]
	64KB	0x4001-0000	0x4001-FFFF	ramsn[1]

	64KB	0x4002-0000	0x4002-FFFF	ramsn[2]
	64KB	0x4003-0000	0x4003-FFFF	ramsn[3]
0010	32KB	0x4000-0000	0x4000-7FFF	ramsn[0]
	32KB	0x4000-8000	0x4000-FFFF	ramsn[1]
	32KB	0x4001-0000	0x4001-7FFF	ramsn[2]
	32KB	0x4001-8000	0x4001-FFFF	ramsn[3]
0001	16KB	0x4000-0000	0x4000-3FFF	ramsn[0]
	16KB	0x4000-4000	0x4000-7FFF	ramsn[1]
	16KB	0x4000-8000	0x4000-BFFF	ramsn[2]
	16KB	0x4000-C000	0x4000-FFFF	ramsn[3]
0000	8KB	0x4000-0000	0x4000-1FFF	ramsn[0]
	8KB	0x4000-2000	0x4000-3FFF	ramsn[1]
	8KB	0x4000-4000	0x4000-5FFF	ramsn[2]
	8KB	0x4000-6000	0x4000-7FFF	ramsn[3]

2. SDRAM 地址空间分配

表 4-8-7 SDRAM 地址空间（SRAM 使能、SDRAM 使能）

SDRBS 配置	大小	起始地址	结束地址	片选信号
111	512MB	0x6000-0000	0x7FFF-FFFF	sdcsn[1]
110	256MB	0x6000-0000	0x6FFF-FFFF	sdcsn[0]
	256MB	0x7000-0000	0x7FFF-FFFF	sdcsn[1]
101	128MB	0x6000-0000	0x67FF-FFFF	sdcsn[0]
	128MB	0x6800-0000	0x6FFF-FFFF	sdcsn[1]
100	64MB	0x6000-0000	0x63FF-FFFF	sdcsn[0]
	64MB	0x6400-0000	0x67FF-FFFF	sdcsn[1]
011	32MB	0x6000-0000	0x61FF-FFFF	sdcsn[0]
	32MB	0x6200-0000	0x63FF-FFFF	sdcsn[1]
010	16MB	0x6000-0000	0x60FF-FFFF	sdcsn[0]
	16MB	0x6100-0000	0x61FF-FFFF	sdcsn[1]
001	8MB	0x6000-0000	0x607F-FFFF	sdcsn[0]
	8MB	0x6080-0000	0x60FF-FFFF	sdcsn[1]
000	4MB	0x6000-0000	0x603F-FFFF	sdcsn[0]
	4MB	0x6040-0000	0x607F-FFFF	sdcsn[1]

注意 当 SDRAM 和 SRAM 都配置使能，并且 SDRAM 的 bank 的 size 配置为 512MB 时，应当用处理器的片选 sdcsn[1]连接 SDRAM 存储器的片选。

SRAM 使能、SDRAM 不使能

3. SRAM 地址空间分配

表 4-8-8 前 4 个 SRAM 地址空间（SRAM 使能、SDRAM 不使能）

SRBS 配置	大小	起始地址	结束地址	片选信号
1111	256MB	0x4000-0000	0x4FFF-FFFF	ramsn[0]
	256MB	0x5000-0000	0x5FFF-FFFF	ramsn[1]
1110	128MB	0x4000-0000	0x47FF-FFFF	ramsn[0]
	128MB	0x4800-0000	0x4FFF-FFFF	ramsn[1]
	128MB	0x5000-0000	0x57FF-FFFF	ramsn[2]
	128MB	0x5800-0000	0x5FFF-FFFF	ramsn[3]
1101	64MB	0x4000-0000	0x43FF-FFFF	ramsn[0]
	64MB	0x4400-0000	0x47FF-FFFF	ramsn[1]
	64MB	0x4800-0000	0x4BFF-FFFF	ramsn[2]
	64MB	0x4C00-0000	0x4FFF-FFFF	ramsn[3]
1100	32MB	0x4000-0000	0x41FF-FFFF	ramsn[0]
	32MB	0x4200-0000	0x43FF-FFFF	ramsn[1]
	32MB	0x4400-0000	0x45FF-FFFF	ramsn[2]
	32MB	0x4600-0000	0x47FF-FFFF	ramsn[3]
1011	16MB	0x4000-0000	0x40FF-FFFF	ramsn[0]
	16MB	0x4100-0000	0x41FF-FFFF	ramsn[1]
	16MB	0x4200-0000	0x42FF-FFFF	ramsn[2]
	16MB	0x4300-0000	0x43FF-FFFF	ramsn[3]
1010	8MB	0x4000-0000	0x407F-FFFF	ramsn[0]
	8MB	0x4080-0000	0x40FF-FFFF	ramsn[1]
	8MB	0x4100-0000	0x417F-FFFF	ramsn[2]
	8MB	0x4180-0000	0x41FF-FFFF	ramsn[3]
1001	4MB	0x4000-0000	0x403F-FFFF	ramsn[0]
	4MB	0x4040-0000	0x407F-FFFF	ramsn[1]
	4MB	0x4080-0000	0x40BF-FFFF	ramsn[2]
	4MB	0x40C0-0000	0x40FF-FFFF	ramsn[3]
1000	2MB	0x4000-0000	0x401F-FFFF	ramsn[0]
	2MB	0x4020-0000	0x403F-FFFF	ramsn[1]
	2MB	0x4040-0000	0x405F-FFFF	ramsn[2]
	2MB	0x4060-0000	0x407F-FFFF	ramsn[3]
0111	1MB	0x4000-0000	0x400F-FFFF	ramsn[0]
	1MB	0x4010-0000	0x401F-FFFF	ramsn[1]

	1MB	0x4020-0000	0x402F-FFFF	ramsn[2]
	1MB	0x4030-0000	0x403F-FFFF	ramsn[3]
0110	512KB	0x4000-0000	0x4007-FFFF	ramsn[0]
	512KB	0x4008-0000	0x400F-FFFF	ramsn[1]
	512KB	0x4010-0000	0x4017-FFFF	ramsn[2]
	512KB	0x4018-0000	0x401F-FFFF	ramsn[3]
0101	256KB	0x4000-0000	0x4003-FFFF	ramsn[0]
	256KB	0x4004-0000	0x4007-FFFF	ramsn[1]
	256KB	0x4008-0000	0x400B-FFFF	ramsn[2]
	256KB	0x400C-0000	0x400F-FFFF	ramsn[3]
0100	128KB	0x4000-0000	0x4001-FFFF	ramsn[0]
	128KB	0x4002-0000	0x4003-FFFF	ramsn[1]
	128KB	0x4004-0000	0x4005-FFFF	ramsn[2]
	128KB	0x4006-0000	0x4007-FFFF	ramsn[3]
0011	64KB	0x4000-0000	0x4000-FFFF	ramsn[0]
	64KB	0x4001-0000	0x4001-FFFF	ramsn[1]
	64KB	0x4002-0000	0x4002-FFFF	ramsn[2]
	64KB	0x4003-0000	0x4003-FFFF	ramsn[3]
0010	32KB	0x4000-0000	0x4000-7FFF	ramsn[0]
	32KB	0x4000-8000	0x4000-FFFF	ramsn[1]
	32KB	0x4001-0000	0x4001-7FFF	ramsn[2]
	32KB	0x4001-8000	0x4001-FFFF	ramsn[3]
0001	16KB	0x4000-0000	0x4000-3FFF	ramsn[0]
	16KB	0x4000-4000	0x4000-7FFF	ramsn[1]
	16KB	0x4000-8000	0x4000-BFFF	ramsn[2]
	16KB	0x4000-C000	0x4000-FFFF	ramsn[3]
0000	8KB	0x4000-0000	0x4000-1FFF	ramsn[0]
	8KB	0x4000-2000	0x4000-3FFF	ramsn[1]
	8KB	0x4000-4000	0x4000-5FFF	ramsn[2]
	8KB	0x4000-6000	0x4000-7FFF	ramsn[3]

表 4-8-9 第 5 个 SRAM 地址空间（SRAM 使能、SDRAM 不使能）

大小	起始地址	结束地址	片选信号
256MB	0x6000-0000	0x6FFF-FFFF	ramsn[4]
256MB	0x7000-0000	0x7FFF-FFFF	ramsn[4]

注意 地址空间 0x6000-0000~0x6FFF-FFFF 同 0x7000-0000~0x7FFF-FFFF 重叠。

SRAM 不使能、SDRAM 使能

4. SDRAM 地址空间分配

表 4-8-10 SDRAM 地址空间（SRAM 不使能、SDRAM 使能）

SDRBS 配置	大小	起始地址	结束地址	片选信号
111	512MB	0x4000-0000	0x5FFF-FFFF	sdcsn[0]
	512MB	0x6000-0000	0x7FFF-FFFF	sdcsn[1]
110	256MB	0x4000-0000	0x4FFF-FFFF	sdcsn[0]
	256MB	0x5000-0000	0x5FFF-FFFF	sdcsn[1]
101	128MB	0x4000-0000	0x47FF-FFFF	sdcsn[0]
	128MB	0x4800-0000	0x4FFF-FFFF	sdcsn[1]
100	64MB	0x4000-0000	0x43FF-FFFF	sdcsn[0]
	64MB	0x4400-0000	0x47FF-FFFF	sdcsn[1]
011	32MB	0x4000-0000	0x41FF-FFFF	sdcsn[0]
	32MB	0x4200-0000	0x43FF-FFFF	sdcsn[1]
010	16MB	0x4000-0000	0x40FF-FFFF	sdcsn[0]
	16MB	0x4100-0000	0x41FF-FFFF	sdcsn[1]
001	8MB	0x4000-0000	0x407F-FFFF	sdcsn[0]
	8MB	0x4080-0000	0x40FF-FFFF	sdcsn[1]
000	4MB	0x4000-0000	0x403F-FFFF	sdcsn[0]
	4MB	0x4040-0000	0x407F-FFFF	sdcsn[1]

SRAM 不使能、SDRAM 不使能

此种情况等同于 SRAM 使能、SDRAM 不使能的情况。

4.8.3 RAM 接口

RAM 访问支持两种 RAM 类型：SRAM 和 SDRAM。

4.8.4 SRAM 接口

SRAM 接口支持 5 个 SRAM 片选，SRAM 控制信号包括：片选 ramsn[4:0]、读使能 ramoen[4:0]和写使能 rwen[3:0]。片选 ramsn[4:0]与读使能 ramoen[4:0]一一对应，写使能 rwen[3:0]为所有片选共用。

表 4-8-11 SRAM 片选和读写使能对应关系表

片选	读使能	写使能
ramsn[4]	ramoen[4]	rwen[3:0]
ramsn[3]	ramoen[3]	
ramsn[2]	ramoen[2]	
ramsn[1]	ramoen[1]	
ramsn[0]	ramoen[0]	

前 4 个 SRAM 片选容量大小通过配置寄存器 MCFG2[12:9]进行配置，范围为 8K 字节到 256M 字节，当配置为 256M 字节时，片选 ramsn[3:0]只留下 ramsn[1:0]。第 5 个 sram 片选为 ramsn[4]，容量大小固定为 256M 字节，起始地址为 0x6000-0000。

SRAM 数据总线的位宽，支持 8 位、16 位和 32 位，通过配置寄存器 MCFG2[5:4]进行配置，其值是“00”表示 8 位总线，“01”表示 16 位总线，“1x”表示 32 位总线。

当 SRAM 配置为 8 位数据宽度时，SRAM 存储体数据管脚连接处理器的数据管脚为 data[31:24]，SRAM 存储体地址管脚最低位连接处理器的地址管脚为 address[0]。8 位数据宽度 SRAM 的 EDAC 功能同管脚 pdata[6:0]无关。

当 SRAM 配置为 16 位数据宽度时，SRAM 存储体数据管脚连接处理器的数据管脚为 data[31:16]，SRAM 存储体地址管脚最低位连接处理器的地址管脚为 address[1]。

当 SRAM 配置为 32 位数据宽度时，SRAM 存储体数据管脚连接处理器的数据管脚为 data[31:0]，SRAM 存储体地址管脚最低位连接处理器的地址管脚为 address[2]。如果 32 位数据宽度 SRAM 使用 EDAC 功能，处理器的校验位管脚 pdata[6:0]连接 SRAM 校验位存储体。

SRAM 连接示意图见图 4-8-1、图 4-8-2、图 4-8-3。

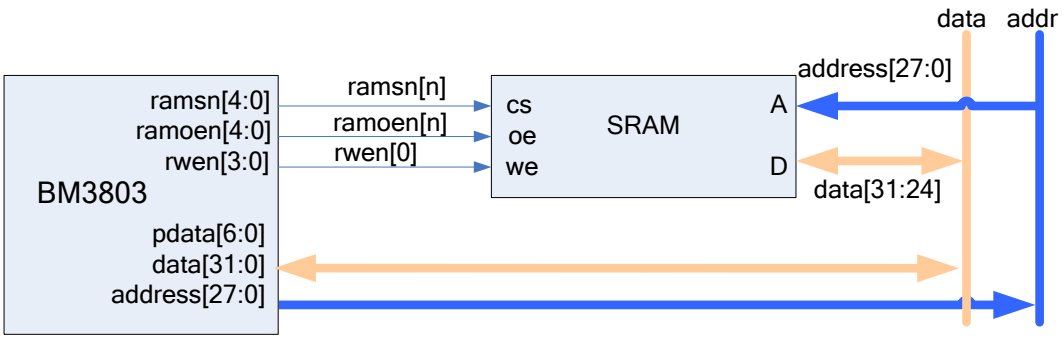


图 4-8-1 8 位宽 SRAM 连接示意图

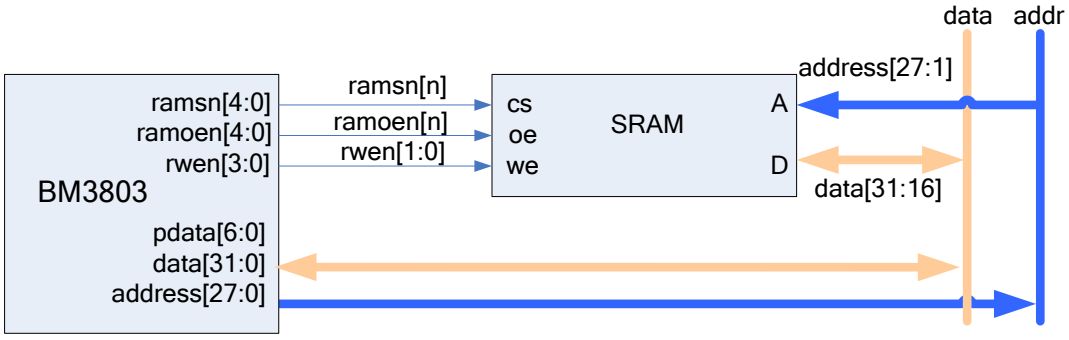


图 4-8-2 16 位宽 SRAM 连接示意图

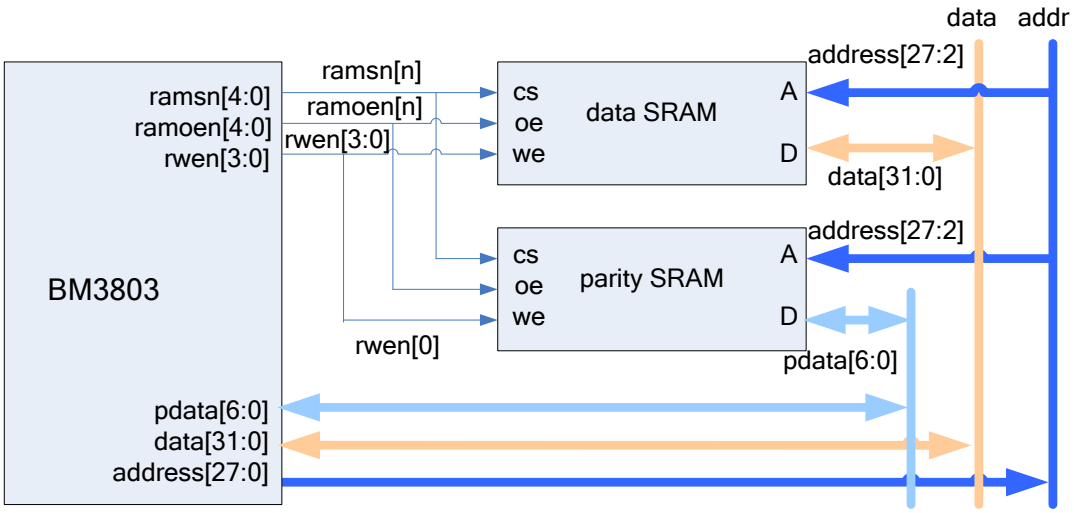


图 4-8-3 32 位宽 SRAM 连接示意图

SRAM 写操作分“读改写”模式和“写选通”模式两种，通过配置寄存器 MCFG2[6]进行配置。

MCFG2[6]配置为高时，SRAM 写操作为“读改写”模式，rwen[3:0]信号会同时有效，或同时无效，当 SRAM 配置为 32 位数据宽度时进行字节或半字写访问，先读字节或半字所在的字，然后控制器在片内改写字内的字节或半字，再把整个字写到 SRAM 中；当 SRAM 配置为 16 位数据宽度时进行字节写访问时，先读字节所在的半字，然后控制器在片内改写半字内的字节，再把整个半字写到 SRAM 中。

MCFG2[6]配置为低时，SRAM 写操作为“写选通”模式，rwen[3:0]信号会分别有效或无效，当出现字节或半字写访问时，字节或半字对应的 rwen 信号会有效，其它 rwen 信号会无效。

rwen[3:0]与 data[31:0]对应关系，见表 4-8-12。

表 4-8-12 SRAM 写信号与数据字节对应表

SRAM 写信号	对应数据字节
rwen[0]	data[31:24]
rwen[1]	data[23:16]
rwen[2]	data[15:8]
rwen[3]	data[7:0]

对于第五个 SRAM，还可以通过 brdyn 管脚，来延迟适配读写访问时间。第五个 SRAM 对 brdyn 管脚的使用需要通过配置寄存器 MCFG2[7]进行配置，当 MCFG2[7]配置为“1”时，SRAM 读写结束前，需要 brdyn 管脚为低电平。

SRAM 的读写访问时序，见图 4-8-4、图 4-8-5、图 4-8-6、图 4-8-7、图 4-8-8。

SRAM 读访问时序中“lead_out”周期中的地址值，不保持上一个时钟周期（图中“data2”周期）中的地址值，而是上一个时钟周期（图中“data2”周期）中处理器内部总线地址值。

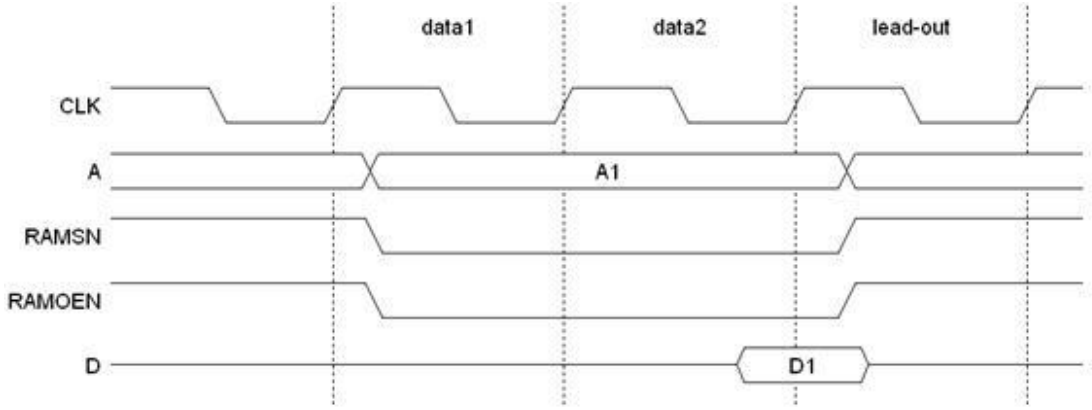


图 4-8-4 SRAM 读访问时序（EDAC 不使能，0 等待周期）

SRAM 读访问，EDAC 不使能时，在片选从有效变为无效前的时钟上升沿采集数据。

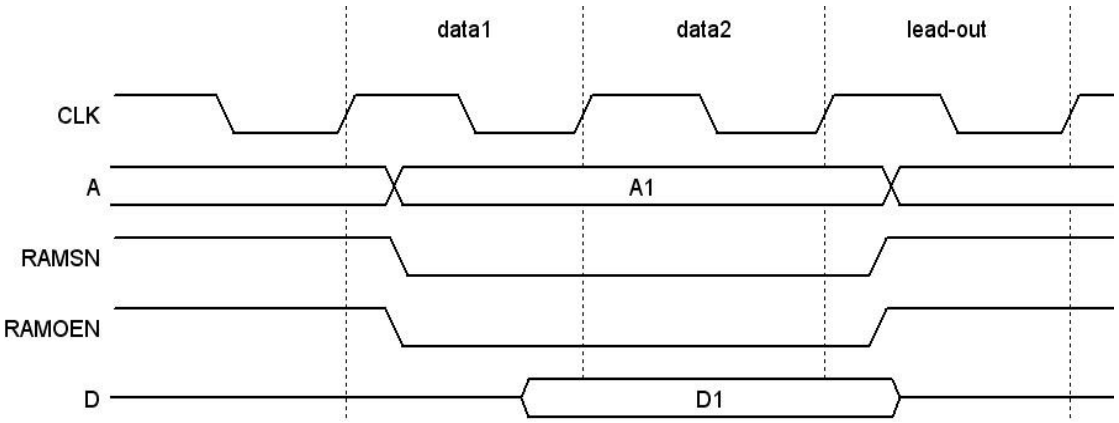


图 4-8-5 SRAM 读访问时序（EDAC 使能，0 等待周期）

SRAM 读访问，EDAC 使能时比不使能时，早一个时钟周期采集数据。

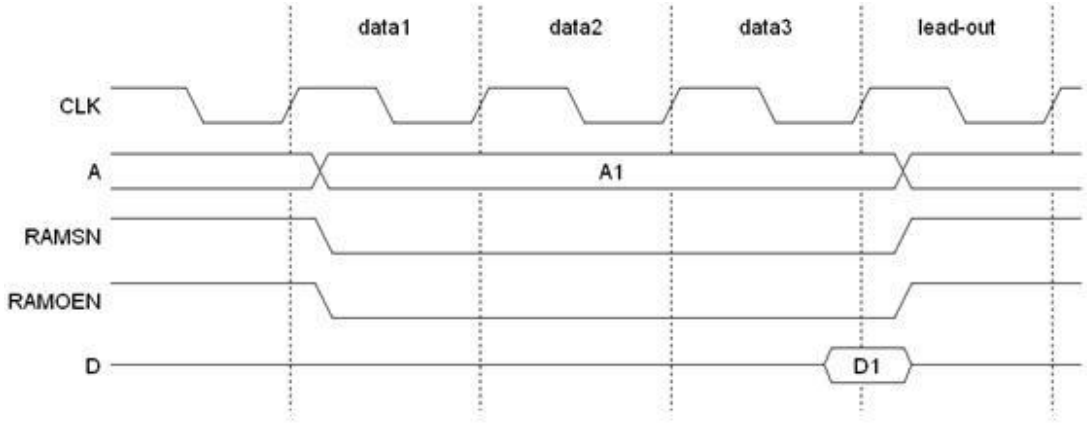


图 4-8-6 SRAM 读访问时序（EDAC 不使能，1 等待周期）

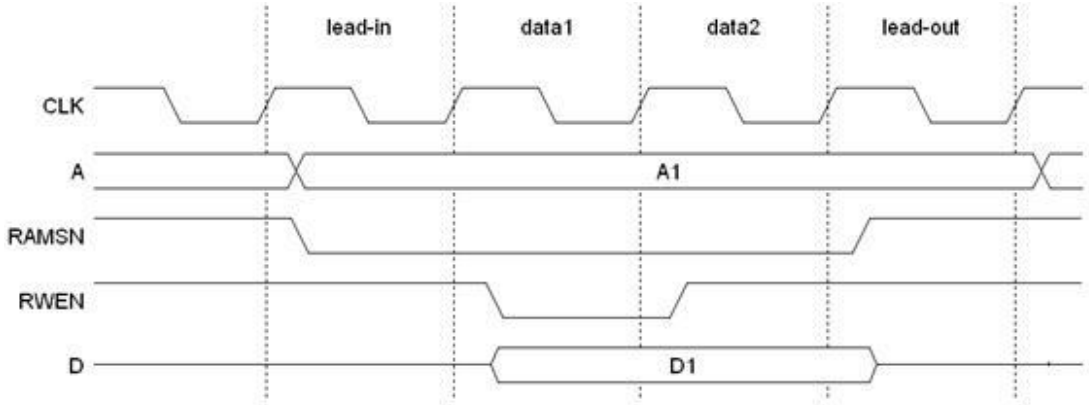


图 4-8-7 SRAM 写访问时序（0 等待周期）

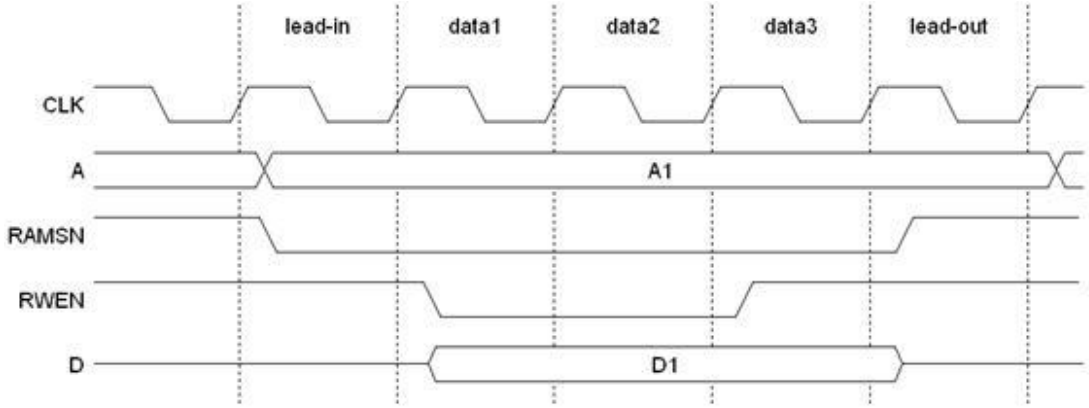


图 4-8-8 SRAM 写访问时序（1 等待周期）

4.8.5 SDRAM 接口

同步动态随机存储器接口支持 2 个片选，SDRAM 控制信号包括：片选 sdcsn[1:0]、行地址选择 sdrasn、列地址选择 sdcasn、写使能 sdwen、数据屏蔽 sddqm[3:0]和时钟 sdclk。每个 SDRAM 片选容量通过配置寄存器 MCFG2[25:23]配置，范围按照两倍递进从 4M 到 512M 字节。控制器支持 64M、256M 和 512M 的器件，这些器件列地址范围从 8 到 12 位，行地址位最大 13，片内 4 个块。SDRAM 只支持 32 位数据总线。

SDRAM 存储体的地址管脚应连接到处理器的 address[14:2]，SDRAM 存储器的块地址应连接到处理器的 address[16:15]。存储体地址管脚少于 13 位时，连接处理器管脚 address[14:2]的低位。

SDRAM 存储体的数据管脚应连接到处理器的 data[31:0]，如果 SDRAM 使用 EDAC 功能，处理器的校验位管脚 pdata[6:0]连接 SDRAM 校验位存储体。

SDRAM 连接示意图见图 4-8-9。

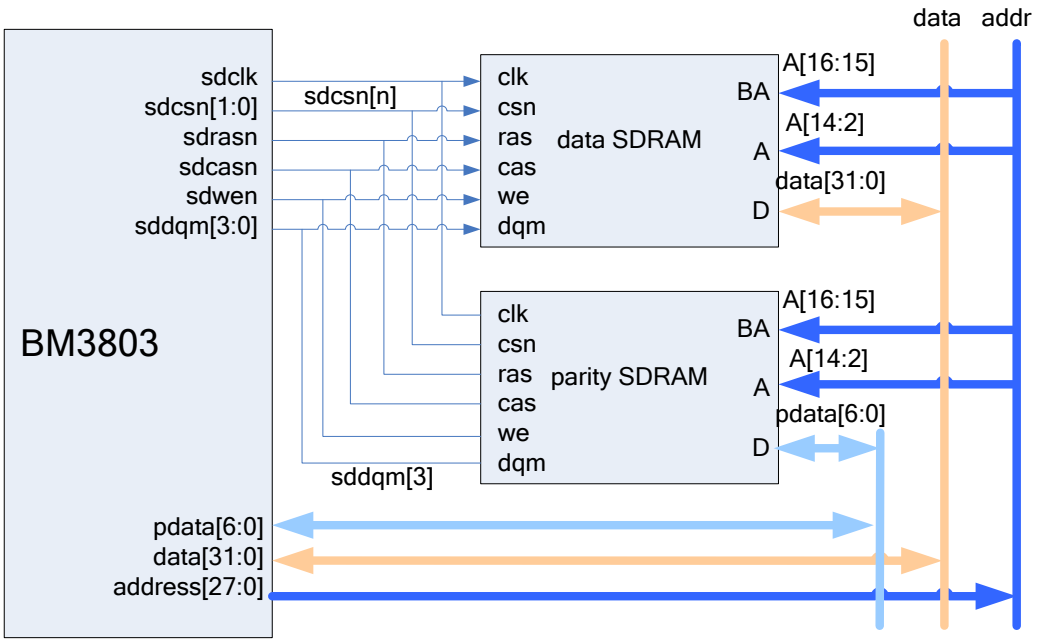


图 4-8-9 SDRAM 连接示意图

数据屏蔽 sddqm 管脚，也叫数据选通管脚，“1”表示屏蔽，“0”表示选通。sddqm[3:0]与 data[31:0]对应关系，见表 4-8-13。

表 4-8-13 SDRAM 数据屏蔽与数据字节对应表

数据屏蔽信号	对应数据字节
sddqm[3]	data[31:24]
sddqm[2]	data[23:16]
sddqm[1]	data[15:8]
sddqm[0]	data[7:0]

4.8.5.1 SDRAM 时序参数的配置

根据不同的SDRAM器件，通过配置寄存器MCFG2、MCFG3配置SDRAM时序参数见表4-8-14。

表 4-8-14 SDRAM 可编程时序参数表

功能	参数	范围	单位
CAS 延迟	T_{CAS}	2 - 3	时钟周期
预充	T_{RP}	2 - 3	时钟周期
自动刷新命令周期	T_{RFC}	3 - 11	时钟周期
自动刷新间隔	$T_{REFRESH}$	10 - 32768	时钟周期

4.8.5.2 SDRAM 命令

SDRAM 控制器能够发送三种 SDRAM 命令。通过对配置寄存器 MCFG2[20:19]写入非零值来发送 SDRAM 命令：

“01” ：发送预充命令；

“10” ：发送自动刷新命令；

“11” ：发送加载模式寄存器 LMR 命令。

每个命令执行完后，MCFG2[20:19]自动清零。

通过执行 LMR 命令，为 SDRAM 存储体配置访问模式，包括 CAS 延时周期和其它固定模式。其它固定模式为全页 burst 读、单一写、顺序 burst。CAS 延时周期会按照配置寄存器 MCFG2[26]，即 SdrCAS，为“0”，对应 CAS 延时周期为 2，为“1”，对应 CAS 延时周期为 3。如果要修改 CAS 延时配置，必须执行 LMR 命令。

执行 LMR 命令时处理器地址、输出值以及它们与 SDRAM 存储器的地址、模式寄存器位对应关系见表 4-8-15。

表 4-8-15 执行 LMR 命令时地址管脚情况

SDRAM		处理器	
地址	模式寄存器	地址	输出值
BA0	M13	address[15]	0
A12	M12	address[14]	0
A11	M11	address[13]	0
A10	M10	address[12]	0
A9	M9	address[11]	1
A8	M8	address[10]	0
A7	M7	address[9]	0
A6	M6	address[8]	0
A5	M5	address[7]	1
A4	M4	address[6]	SDRCAS
A3	M3	address[5]	0
A2	M2	address[4]	1
A1	M1	address[3]	1
A0	M0	address[2]	1

控制器提供周期刷新功能。通过对 MCFG2[31]刷新使能位 SDRREF 配置为“1”来使能周期刷新功能。刷新命令会对两个 SDRAM 片选同时发送。通过 MCFG3[26:12]刷新计数器重载值配置两个自动刷新命令之间的间隔周期。

根据 SDRAM 类型，自动刷新命令间隔周期为 7.8μs 或 15.6μs，即对应 100MHz 频率下的 780 或 1560 个时钟周期。

自动刷新命令间隔周期计算公式如下：

刷新间隔周期=（重载值+1）/ 时钟频率

4.8.5.3 SDRAM 初始化

通过对 MCFG2[14]配置为“1”来使能 SDRAM 后，同时对两个 SDRAM 片选自动执行 SDRAM 初始化序列命令：依次执行预充命令、两个自动刷新命令和 LMR 命令。

4.8.5.4 SDRAM 读访问

读周期包括三个步骤：

第一步：对所需的块和行执行 ACTIVE 命令；

第二步：经过 CAS 延时后，执行 READ 命令；

第三步：以预充命令结束读周期。

一次访问结束后，会关闭当前行，如果内部总线发出突发模式访问请求则执行突发模式读操作。

SDRAM 读时序（Burst length = 1; CL = 3），见图 4-8-10。

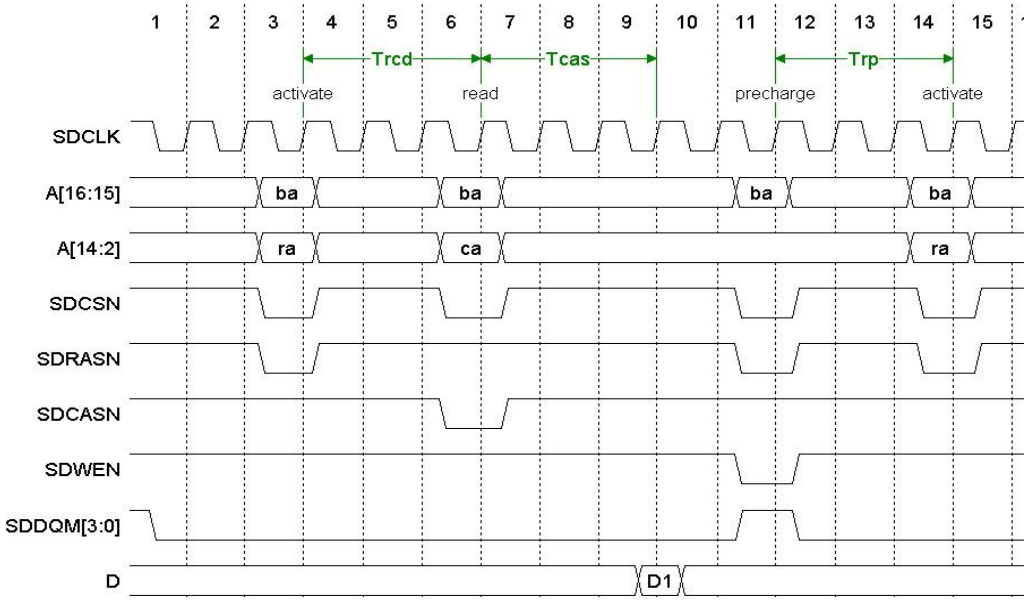


图 4-8-10 SDRAM 读访问时序

4.8.5.5 SDRAM 写访问

写周期包括三个步骤：

第一步：对所需的块和行执行 ACTIVE 命令；

第二步：经过 CAS 延时后，执行 WRITE 命令；

第三步：以预充命令结束写周期。

内部总线的突发写操作会产生没有空闲周期的突发写命令。

SDRAM 写时序（Burst length = 1; CL = 3），见图 4-8-11。

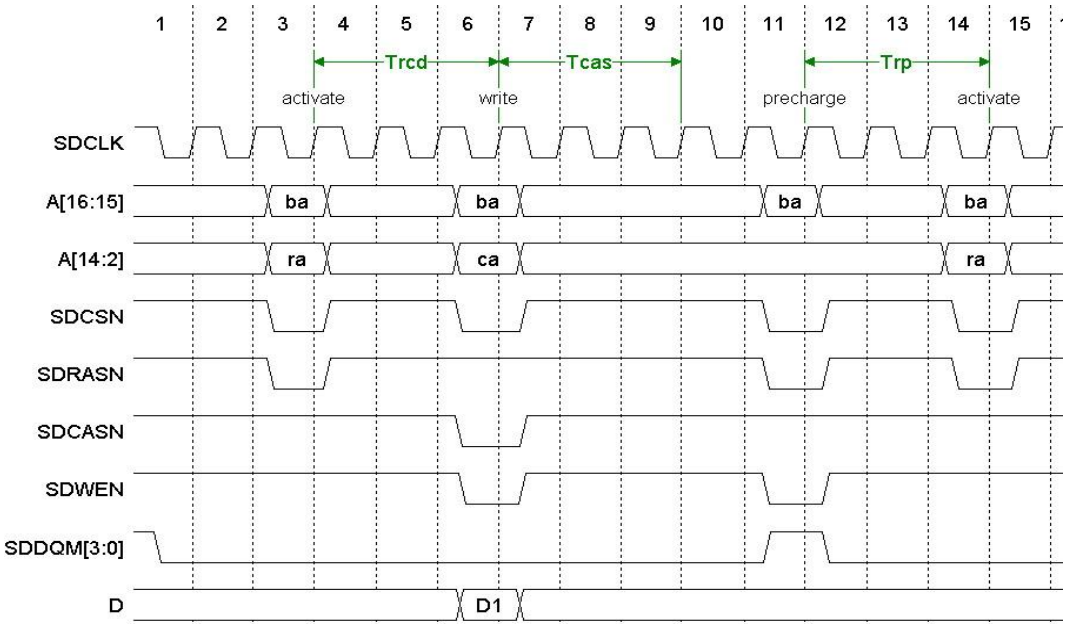


图 4-8-11 SDRAM 写访问时序

4.8.6 PROM 接口

PROM 接口支持 2 个 PROM 片选，每个片选容量无需配置，每个片选容量最大 256M 字节。PROM 控制信号包括：片选 romsn[1:0]、读使能 oen、写使能 writen。

通过配置寄存器 MCFG1[16:12]配置 PROM 写等待周期，通过配置寄存器 MCFG1[4:0]配置 PROM 读等待周期，等待周期配置范围为 0 到 31。通过配置寄存器 MCFG1[11]配置 PROM 的写使能。通过配置寄存器 MCFG1[9:8]配置 PROM 的数据宽度。

当 PROM 配置为 8 位数据宽度时，PROM 存储体数据管脚连接处理器的数据管脚为 data[31:24]，PROM 存储体地址管脚最低位连接处理器的地址管脚为 address[0]。8 位数据宽度 PROM 的 EDAC 功能同管脚 pdata[6:0]无关。

当 PROM 配置为 16 位数据宽度时，PROM 存储体数据管脚连接处理器的数据管脚为 data[31:16]，PROM 存储体地址管脚最低位连接处理器的地址管脚为 address[1]。

当 PROM 配置为 32 位数据宽度时，PROM 存储体数据管脚连接处理器的数据管脚为 data[31:0]，PROM 存储体地址管脚最低位连接处理器的地址管脚为 address[2]。如果 32 位数据宽度 PROM 使用 EDAC 功能，处理器的校验位管脚 pdata[6:0]连接 PROM 校验位存储体。

PROM 连接示意图见图 4-8-12、图 4-8-13、图 4-8-14。

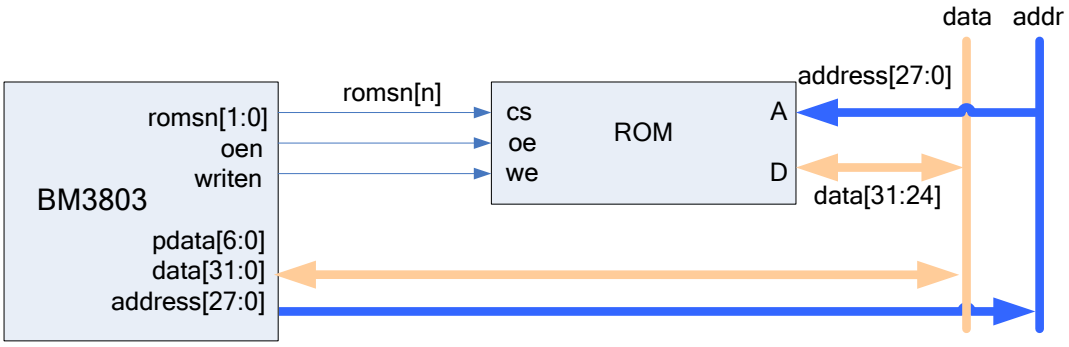


图 4-8-12 8 位宽 PROM 连接示意图

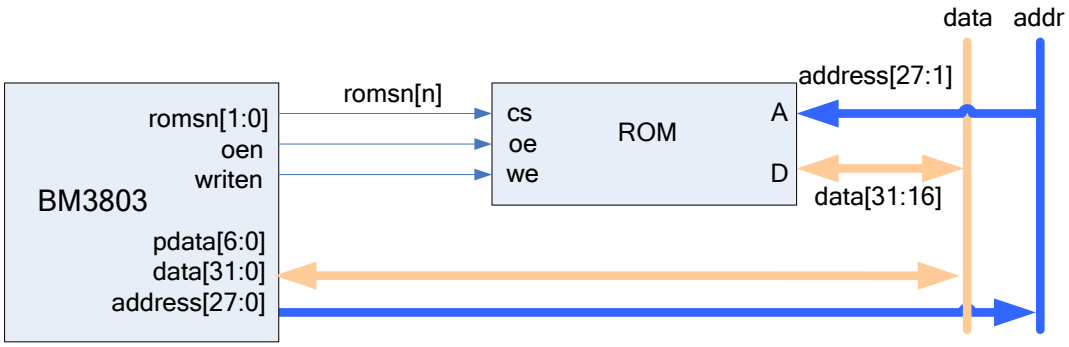


图 4-8-13 16 位宽 PROM 连接示意图

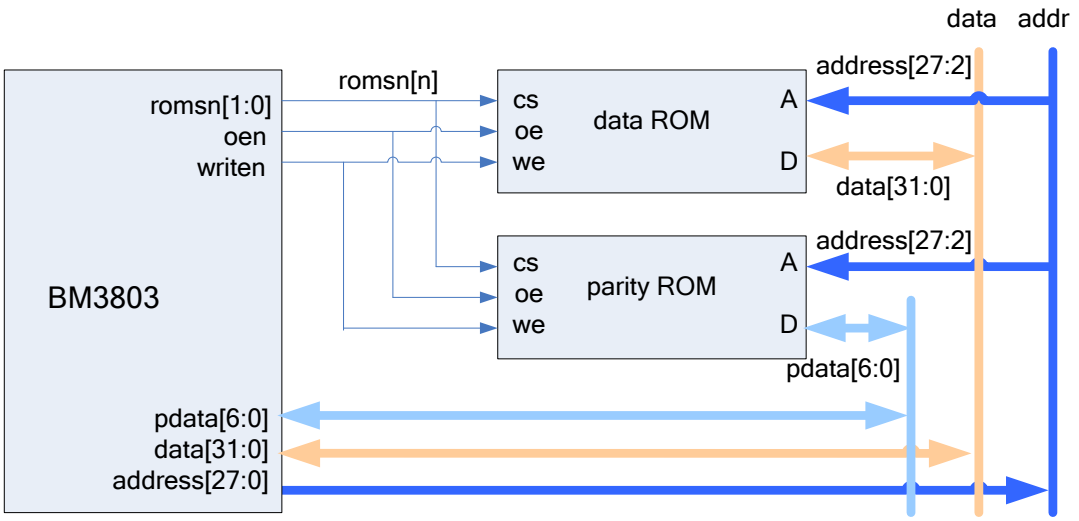


图 4-8-14 32 位宽 PROM 连接示意图

PROM 的读写访问接口时序，见图 4-8-15、图 4-8-16、图 4-8-17、图 4-8-18、图 4-8-19。

PROM 读访问时序中“lead_out”周期中的地址值，不保持上一个时钟周期（图中“data2”周期）中的地址值，而是上一个时钟周期（图中“data2”周期）中处理器内部总线地址值。

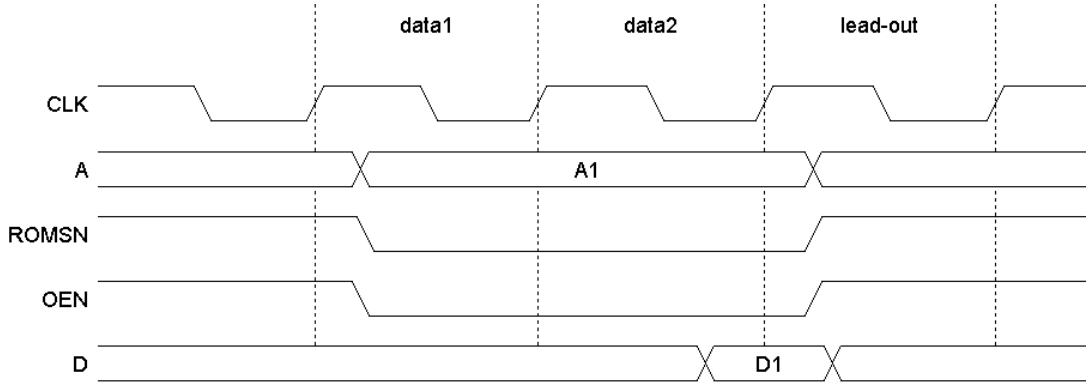


图 4-8-15 PROM 读访问时序（EDAC 不使能，0 个等待周期）

PROM 读访问，EDAC 不使能时，在片选从有效变为无效前的时钟上升沿采集数据。

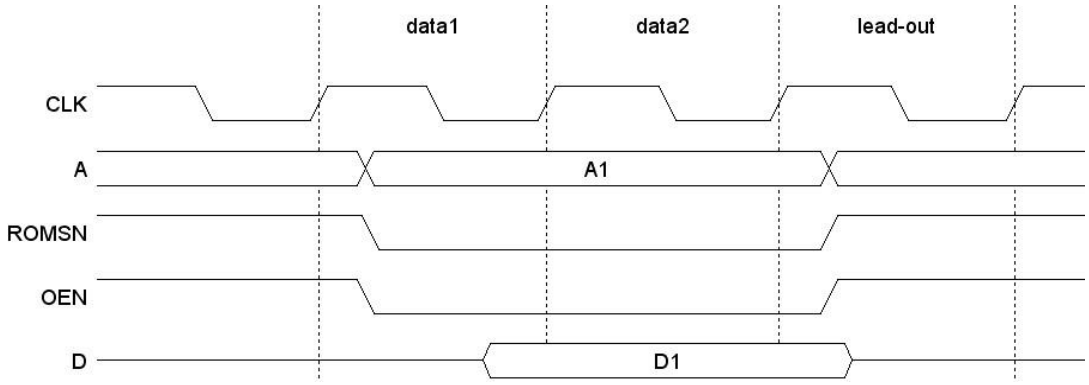


图 4-8-16 PROM 读访问时序（EDAC 使能，0 等待周期）

PROM 读访问，EDAC 使能时比不使能时，早一个时钟周期采集数据。

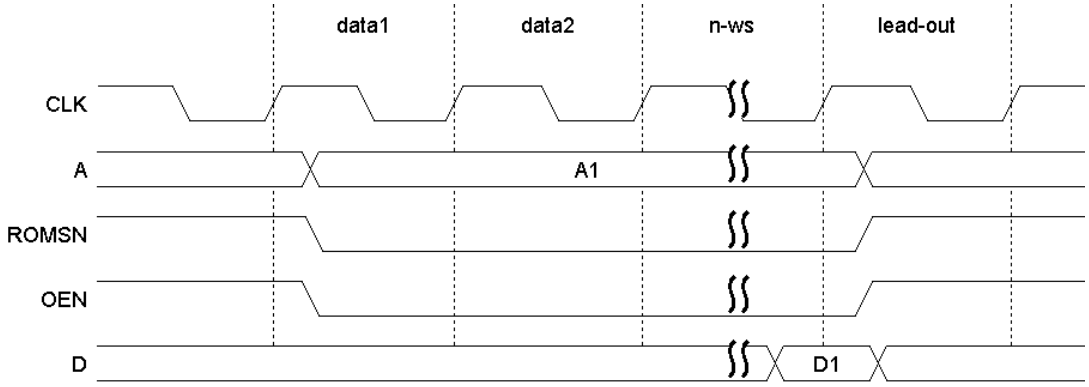


图 4-8-17 PROM 读访问时序（EDAC 不使能，n 个等待周期）

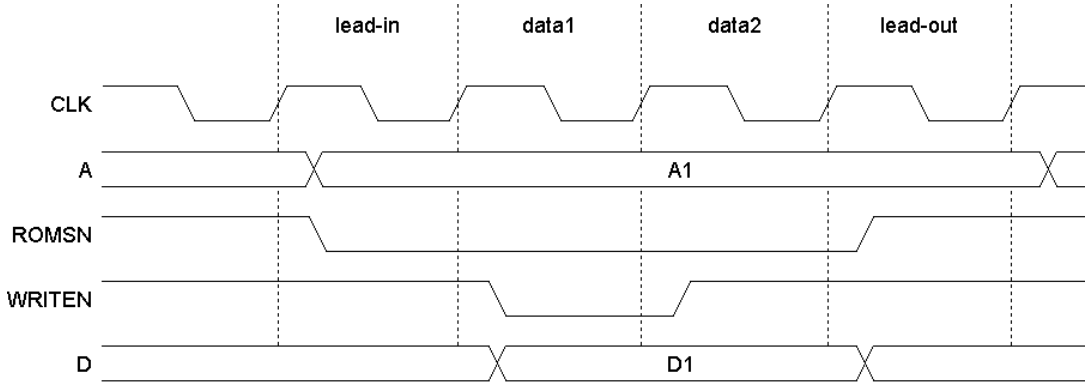


图 4-8-18 PROM 写访问时序（0 个等待周期）

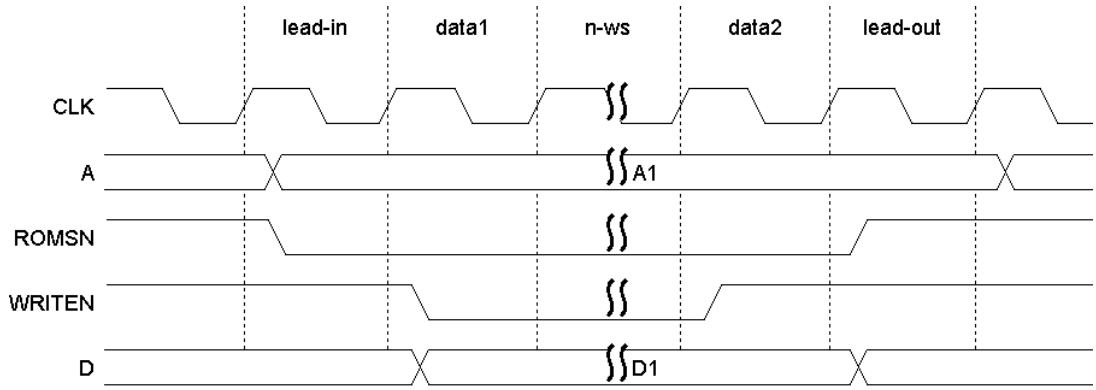


图 4-8-19 PROM 写访问时序（n 个等待周期）

4.8.7 I/O 设备接口

I/O 设备接口支持一个片选，容量大小无需配置，最大 256M 字节。I/O 设备地址范围是 0x2000-0000 ~ 0x3FFF-FFFF，其中 0x2000-0000 ~ 0x2FFF-FFFF 和 0x3000-0000 ~ 0x3FFF-FFFF 地址范围是重叠的，即访问 0x3000-0000 等同于访问 0x2000-0000，访问 0x3000-0001 等同于访问 0x2000-0001，依此类推。I/O 设备访问控制信号包括片选 iosn、读使能 oen、写使能 writen。

通过配置寄存器 MCFG1[28:27]配置 I/O 设备数据总线的宽度（“00”=8，“01”=16，“1x”=32）。

当 I/O 设备配置为 8 位数据宽度时，I/O 设备数据管脚连接处理器的数据管脚为 data[31:24]，I/O 设备地址管脚最低位连接处理器的地址管脚为 address[0]。

当 I/O 设备配置为 16 位数据宽度时，I/O 设备数据管脚连接处理器的数据管脚为 data[31:16]，I/O 设备地址管脚最低位连接处理器的地址管脚为 address[1]。

当 I/O 设备配置为 32 位数据宽度时，I/O 设备数据管脚连接处理器的数据管脚为 data[31:0]，I/O 设备地址管脚最低位连接处理器的地址管脚为 address[2]。

I/O 设备数据总线宽度配置为 8、16 位，跟配置为 32 位的读写访问时序相同，区别仅仅是数据总线宽度不同。

I/O 设备数据总线宽度配置为 8 位时，只支持 LDUB 和 STB 指令。

I/O 设备数据总线宽度配置为 16 位时，只支持 LDUH 和 STH 指令。

I/O 设备连接示意图见图 4-8-20、图 4-8-21、图 4-8-22。

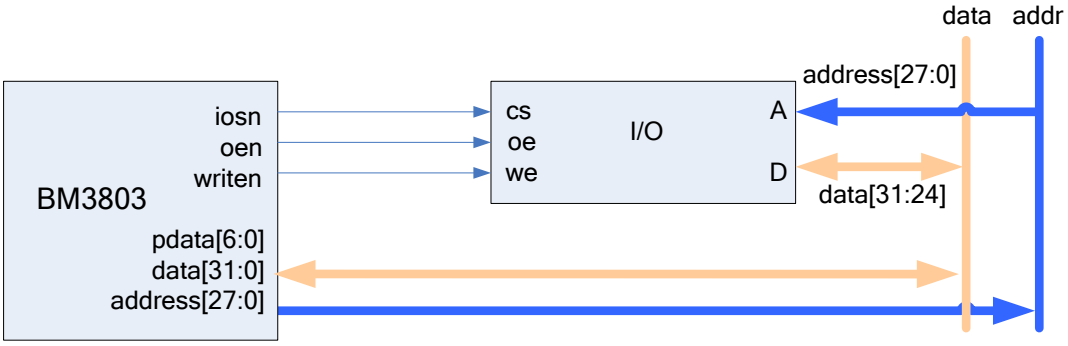


图 4-8-20 8 位宽 I/O 设备连接示意图

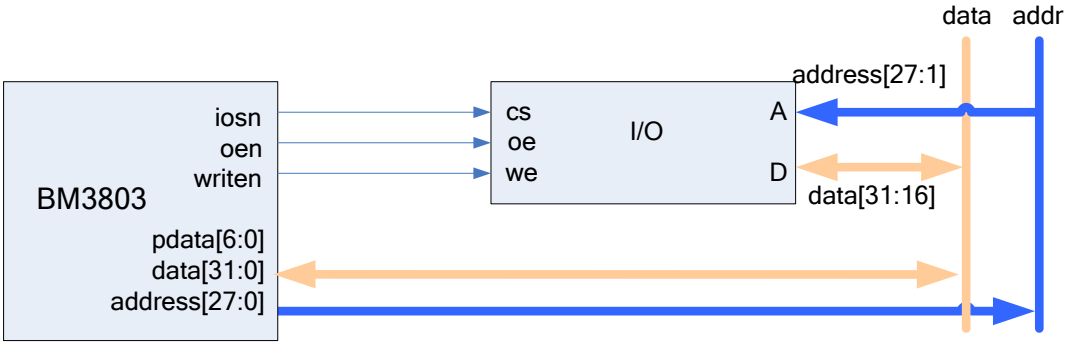


图 4-8-21 16 位宽 I/O 设备连接示意图

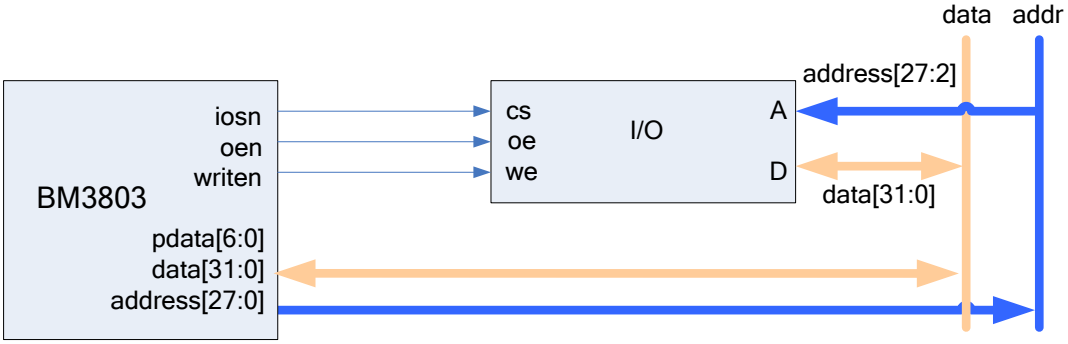


图 4-8-22 32 位宽 I/O 设备连接示意图

通过配置寄存器 MCFG1[28:27]配置数据总线的宽度，MCFG1[26]配置 I/O 设备 BRDYN 使能，MCFG1[24:20]配置读写等待周期，即读使能周期和写使能周期，MCFG1[19]配置 I/O 设备使能。I/O 设备访问时序，见图 4-8-23、图 4-8-24。

I/O 空间读访问时序中“lead_out”周期中的地址值，不保持上一个时钟周期（图中“data2”周期）中的地址值，而是上一个时钟周期（图中“data2”周期）中处理器内部总线地址值。

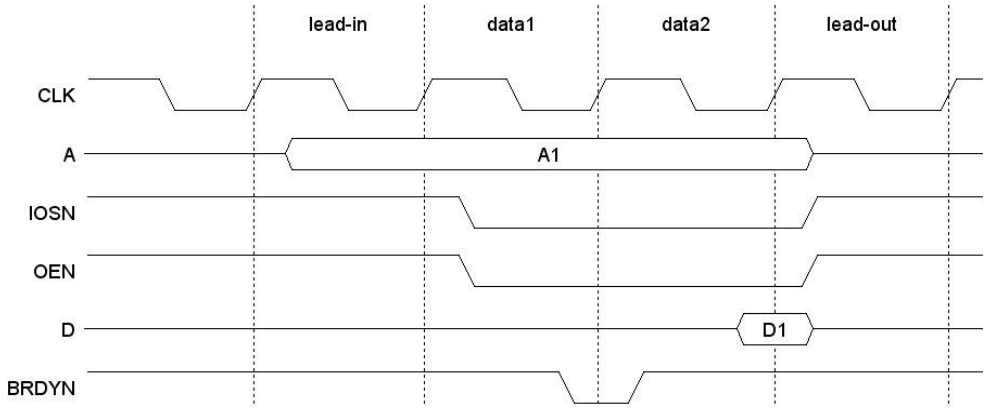


图 4-8-23 I/O 读访问时序（0 等待周期）

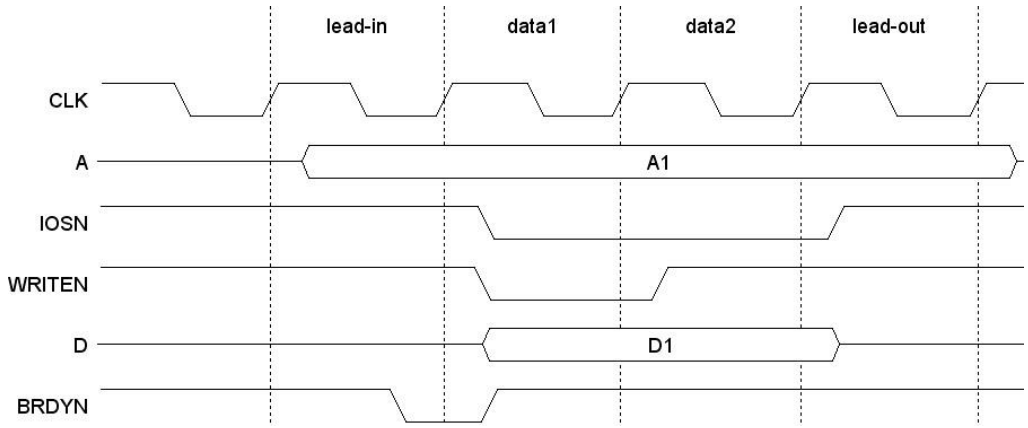


图 4-8-24 I/O 写访问时序（0 等待周期）

I/O 设备访问的读使能周期和写使能周期可以通过 MCFG1[23:20] IOWS 配置位来配置，最大可以配置为“15”，如果通过配置 IOWS 最大“15”仍不能满足 I/O 空间设备访问，可以使用 BRDYN 功能。I/O 空间的 BRDYN 功能通过 MCFG1[26] Brdy 配置位来配置，“1”表示使能 I/O 空间的 BRDYN 功能，“0”表示不使能 I/O 空间的 BRDYN 功能。在使能 I/O 空间 BRDYN

功能情况下，I/O 空间读写访问时，经过 IOWS 配置的等待周期后，采集 brdyn 管脚，如果采集到 brdyn 管脚为低电平后，结束等待，且在 brdyn 管脚为低电平后一个时钟周期后，读写信号结束。

4.8.8 访问错误

处理器通过 bexcn 信号指示访问错误，适用的外部存储器类型包括 PROM、SRAM、I/O 设备。如果通过设置 MCFG1[25]*BEXC*位为“1”使能访问错误，在接受数据的同时接受 bexcn 信号，如果 bexcn 管脚为低，内部总线会收到错误响应：

若处于取指操作，则进入 Trap0x01；

若处于读数据操作，则进入 Trap0x09；

若处于写数据操作，则进入 Trap0x2B。

4.8.9 错误管理

BM3803FMGRH 处理器对外部存储器访问具有 EDAC（纠错检错）功能，能够纠正 32 位字中的一个错误，能够检测 32 位字中的两个错误。一个错误具体是指 32 位数据位和 7 位校验位总共出现 1 位错误，两个错误具体是指 32 位数据位和 7 位校验位总共出现 2 位错误。检错和纠错时，对数据位和校验位同等对待，当检错时，不论错误出现在数据位或校验位都检错，当纠正 1 位错误时，不论 1 位错误出现在数据位或校验位都纠正。

处理器对 8 位的 PROM、32 位的 RPOM、8 位的 SRAM、32 位的 SRAM 和 SDRAM 存储器访问具有 EDAC 功能。

SDRAM 存储器访问，在使能 EDAC 功能时，不支持内部 AHB 总线的 burst 传输模式。

32 位的 PROM、32 位的 SRAM 和 32 位的 SDRAM 实现 EDAC 功能需要分别搭配 7 位的 PROM、7 位的 SRAM 和 7 位的 SDRAM 校验位存储器。

8 位的 PROM 和 8 位的 SRAM 实现 EDAC 功能不需要搭配额外的校验位存储器，而是数据位和校验位存放在同一个存储体当中，数据位存放在存储体的前四分之三，校验位存放在存储体的后四分之一，8 位存储体的前 4 个字节，即第一个字，对应的 7 位校验位（占一个字节）存放在存储体的四分之三起始位置，以此类推。

通过如下的外部存储器容错配置寄存器实现对存储器的纠检错功能的控制。

5. 外部存储器容错配置寄存器 MECFG1（0x8000-0100）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SR1EEN	SR2EEN	SR3EEN	SR4EEN	SR5EEN	ROMEEN	保留			EWB2	ERB	EWB1	SDEEN	EWB0	PRBS				保留				TCBAREA		保留	TCB							
r/w	r/w	r/w	r/w	r/w	r/w	r			r/w	r/w	r/w	r/w	r/w	r/w				r				r/w		r	r/w						
0	0	0	0	0	0	000			1	0	0	0	0	0000				0000				11		0	x						

实现对 SRAM 和 SDRAM 不同 BANK 的 EDAC 使能位的控制，同时在验证 EDAC 使能模式下记录数据的校验位。

- [31]：SRAM bank1 EDAC 使能，“1”为使能，“0”为不使能。
- [30]：SRAM bank2 EDAC 使能，“1”为使能，“0”为不使能。
- [29]：SRAM bank3 EDAC 使能，“1”为使能，“0”为不使能。
- [28]：SRAM bank4 EDAC 使能，“1”为使能，“0”为不使能。
- [27]：SRAM bank5 EDAC 使能，“1”为使能，“0”为不使能。
- [26]：PROM EDAC 使能，高有效。此字段的初值在上电复位时由 gpio 管脚的第 2 位配置。
- [25:23]：保留。
- [22]：EDAC 写旁路（EWB2），EWB2、1、0=“011”时有效。
- [21]：EDAC 读旁路(ERB)，“1”为使能，“0”为不使能。
- [20]：EDAC 写旁路（EWB1），EWB2、1、0=“011”时有效。
- [19]：SDRAM EDAC 使能，“1”为使能，“0”为不使能。
- [18]：EDAC 写旁路(EWB0)，EWB2、1、0=“011”时有效。
- [17:14]：PROM bank size。定义每个 PROM 大小。（“0000”=8 Kbyte，“0001”=16 Kbyte , … ,“1111”=256 Mbyte)。用于 8 位 PROM 的 EDAC。此字段的初值在上电复位时由外部管脚 rombsd[3:0]配置。
- [13:10]：保留。
- [9:8]：TCB area。将每个 bank 的 SRAM 和每个 bank 的 SDRAM 平均分成 4 份区域，读取此字段指向的区域中的数据，其对应的校验位，在 ERB 使能时存入 TCB 字段。

第 5 片选的 SRAM 不受 TCBarea 字段控制。RPOM 不受 TCBarea 字段控制。

- [7]：保留。
- [6:0]：Test check bits(TCB)。用于测试校验，如果 MECFG1 中 ERB 被使能，且对于 8、32 位数据宽度的 PROM 和 SRAM，以及 32 位数据宽度的 SDRAM，那么在读数据时，根据 TCBarea 字段，读得的正常的校验位存入此字段。

MECFG1[26]*romEEn* 此配置位，可以通过写 MECFG1 来配置，同时还可以在上电复位时通过管脚 gpio[2]来配置。如果 gpio[2]为‘1’，则 *romEEn* 配置位在上电复位时为‘1’。

MECFG1[17:14]*PRBS* 此配置位，可以通过写 MECFG1 来配置，同时还可以在上电复位时通过管脚 rombsd[3:0]来配置。

6. 存储器容错配置寄存器 MECFG2（0x8000-0104）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR																															
r/w																															
0																															

- [31:0]DR：数据位翻转。“1”为翻转，“0”为不翻转。

MECFG2 用于数据位造错。当 MECFG1 中 *EWB* 设置有效时，对于 32 位数据宽度 SRAM 和 SDRAM 整个字的写操作，则写入数据位按 MECFG2 来翻转。

7. 存储器容错配置寄存器 MECFG3（0x8000-0108）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																									PR						
r																									r/w						
0																									0						

- [31:7]：保留。
- [6:0]PR：校验位翻转。“1”为翻转，“0”为不翻转。

MECFG3 用于校验位造错。当 MECFG1 中 *EWB* 设置有效时，对于 8、32 位数据宽度 SRAM

和 SDRAM 整个字的写操作，则写入校验位按 MECFG3 来翻转。

8. EWB 写旁路说明（对于 PROM）

- EWB 写旁路有效或无效，对 PROM 不发生作用。

9. EWB 写旁路说明（对于 SRAM 和 SDRAM）

- 不论 EDAC 是否使能，只需 EWB 有效，且 32 位数据宽度 SRAM 和 SDRAM，且整个字的写操作，则写数据位和写校验位制造错误，即数据位按 MECFG2 翻转来写和校验位按 MECFG3 的 PR 字段翻转来写。
- 不论 EDAC 是否使能，只需 EWB 有效，且 8 位数据宽度 SRAM，且整个字的写操作，则写校验位制造错误，即校验位按 MECFG3 的 PR 字段翻转来写。
- 对于 8 位数据宽度 SRAM，只需 EWB 有效，且 stb 或 sth 指令，则用于直接进行数据位造错，即造错数据就是所写数据。
- 只要 EWB 有效，则纠错不使能。
- 只要 EWB 无效，对于 32 位 SRAM 和 SDRAM，则写正常的数据位和写正常的校验位，即不制造错误。
- 当 EDAC 使能时，如果 EWB 无效，对于 8 位 SRAM，则写正常的数据位和写正常的校验位，即不制造错误。
- 当 EDAC 使能时，如果 EWB 无效，则纠错使能，又叫 EDAC 写使能。

10. ERB 读旁路说明（对于 RPOM）

- RPOM 不受 TCBarea 字段控制，当对 8、32 位数据宽度的 PROM 进行读操作时，当 ERB 读旁路有效时，读得的正常的校验位将存入 MECFG1 中的 TCB 字段。

11. ERB 读旁路说明（对于 SRAM 和 SDRAM）

- 当 8 位、32 位数据宽度 SRAM 和 32 位数据宽度的 SDRAM 读访问时，不论 EDAC 是否使能，MECFG1[9:8]TCBarea 指向区域的数据对应的校验位在 ERB 使能时存入 MECFG1 中的 TCB 字段。
- 第 5 片选的 SRAM 不受 TCBarea 字段控制，其全部数据对应的校验位在 ERB 使能时都

存入 MECFG1 中的 TCB 字段。

- 只要 ERB 有效，则检错不使能。
- 只要 ERB 无效，则 TCB 字段保持。
- 当 EDAC 使能时，如果 ERB 无效，则检错使能，又叫 EDAC 读使能。

12. MECFG1 配置中 EDAC、EWB、ERB 详细说明（对于 SRAM 和 SDRAM）

- 组合 1（MECFG1_1）：当设置 EDAC 使能，且 EWB 无效，且 ERB 清零时，属于正常的 EDAC 检错和纠错设置。可以写入正确的数据位和正确的校验位，不能读出实际的校验位。
- 组合 2（MECFG1_2）：当设置 EDAC 使能，且 EWB 无效，且 ERB 置位时，可以写入正确的数据位和正确的校验位，以及读出实际的数据位和实际的校验位。此组合设置在正常工作中和测试过程中，用于写入正确的数据位和正确的校验位，以及读出实际的数据位和实际的校验位。此组合设置不检错，也不纠错。
- 组合 3（MECFG1_3）：当设置 EDAC 使能，且 EWB 有效，且 ERB 清零时，配合设置全 0 的 MECFG2 和 MECFG1 的 PR 字段，属于正常的 EDAC 检错和不纠错设置。
- 组合 4（MECFG1_4）：当设置 EDAC 使能，且 EWB 有效，且 ERB 置位时，对于 32 位 SRAM 和 SDRAM 可以写入错误的数据位和错误的校验位，以及读出实际的数据位和实际的校验位，校验位将存入 TCB 字段。对于 8 位 SRAM 不能写入错误的数据位，可以写入错误的校验位，可以读出实际的数据位和实际的校验位，校验位将存入 TCB 字段。此组合设置在测试过程中，对于 32 位 SRAM 和 SDRAM 用于写入错误的数据位和错误的校验位，以及读出实际的数据位和实际的校验位。对于 8 位 SRAM 用于写入错误的校验位，以及读出实际的数据位和实际的校验位。此组合设置不检错，也不纠错。
- 组合 5（MECFG1_5）：当设置 EDAC 不使能，且 EWB 无效，且 ERB 清零时，对于 32 位 SRAM 和 SDRAM 可以写入正确的数据位和正确的校验位，不能读出实际的校验位。对于 8 位 SRAM 可以写入正确的数据位，不能写校验位，不能读出实际的校验位。此组合设置可以在测试过程中，对于 32 位 SRAM 和 SDRAM 用于写入正确的数据位和正确的校验位，不能读出实际的校验位。此组合设置不检错，也不纠错。此组合

设置通常不用。

- 组合 6（MECFG1_6）：当设置 EDAC 不使能，且 EWB 无效，且 ERB 置位时，对于 32 位 SRAM 和 SDRAM 可以写入正确的数据位和正确的校验位，以及读出实际的数据位和实际的校验位。对于 8 位 SRAM 可以写入正确的数据位，不能写校验位，可以读出实际的数据位和实际的校验位。此组合设置在测试过程中，对于 32 位 SRAM 和 SDRAM 用于写入正确的数据位和正确的校验位，以及读出实际的数据位和实际的校验位。此组合设置不检错，也不纠错。
- 组合 7（MECFG1_7）：当设置 EDAC 不使能，且 EWB 有效，且 ERB 清零时，对于 32 位 SRAM 和 SDRAM 可以写入错误的数据位和错误的校验位，不能读出实际的校验位。对于 8 位 SRAM 不能写入错误的数据位，可以写入错误的校验位，不能读出实际的校验位。此组合设置在测试过程中，对于 32 位 SRAM 和 SDRAM 用于写入错误的数据位和错误的校验位。对于 8 位 SRAM 用于写入错误的校验位，以及可以通过 stb 和 sth 指令来直接进行数据位造错。此组合设置不检错，也不纠错。此组合设置通常不用。
- 组合 8（MECFG1_8）：当设置 EDAC 不使能，且 EWB 有效，且 ERB 置位时，对于 32 位 SRAM 和 SDRAM 可以写入错误的数据位和错误的校验位，以及读出实际的数据位和实际的校验位，校验位将存入 TCB 字段。对于 8 位 SRAM 不能写入错误的数据位，可以写入错误的校验位，可以读出实际的数据位和实际的校验位，校验位将存入 TCB 字段。此组合设置在测试过程中，对于 32 位 SRAM 和 SDRAM 用于写入错误的数据位和错误的校验位，以及读出实际的数据位和实际的校验位。对于 8 位 SRAM 用于写入错误的校验位，以及读出实际的数据位和实际的校验位。此组合设置不检错，也不纠错。

13. PROM 的 EDAC 功能配置说明

- 当访问 8、32 位 PROM，读操作时，当 ERB 读旁路有效时，读得的正常的校验位将存入 MECFG1 中的 TCB 字段。
- PROM 的 EDAC 纠错检错功能，与 EWB 写旁路和 ERB 读旁路无关。
- PROM 在 EDAC 使能时，检错并纠错，纠错时不写回。

- 8 位宽 PROM 不支持写校验位。
- PROM 没有造错功能。

14. SRAM 和 SDRAM 出错处理

- 如果 MECFG1 配置成 mecfg1_1（EDAC 配置成使能状态，即 MECFG1 中的 Sr1EEn、Sr2EEn、Sr3EEn、Sr4EEn、Sr5EEn 和 SdEEn 置位，并且 EWB 无效和 ERB 清零），处理情况如下：
  - 如果读存储器时出现单位错误，外部存储器控制器将此数据进行纠正并回写存储器。这种情况不产生任何陷阱和中断。
  - 如果读存储器时出现非单位错误，处理情况如下：

    如果此时处于读数据操作，将导致进入数据访问异常 trap0x09；

    如果此时处于取指操作，将导致进入指令访问异常 trap0x01。
  - 如果对存储器进行字节或半字写，要写入的字节或半字所在的字出现单位错误，外部存储器控制器将要写入的字节或半字所在的字进行纠正，再改写这个字要写入的字节或半字，再将整个字写入存储器。这种情况不产生任何陷阱和中断。
  - 如果对存储器进行字节或半字写，要写入的字节或半字所在的字出现非单位错误，将导致进入写访问异常 trap0x2b。
- 如果 MECFG1 配置成 mecfg1_3（MECFG1 中的 Sr1EEn、Sr2EEn、Sr3EEn、Sr4EEn、Sr5EEn 和 SdEEn 置位，并且 EWB 有效和 ERB 清零，并且 MECFG3 中的 PR 字段和 MECFG2 配置为全 0），这种情况，写操作是写入正确的数据位和正确的校验位。读操作的数据校验位不存入 MECFG1 中的 TCB 字段。这种配置的处理情况如下：
  - 如果读存储器时出现单位错误，处理情况如下：

    如果此时处于读数据操作，将导致进入数据访问异常 trap0x09；如果此时处于取指操作，将导致进入指令访问异常 trap0x01。
  - 如果读存储器时出现非单位错误，处理情况如下：

    如果此时处于读数据操作，将导致进入数据访问异常 trap0x09；

    如果此时处于取指操作，将导致进入指令访问异常 trap0x01。
  - 对于 32 位的 sram，如果对存储器进行字节或半字写，要写入的字节或半字所在的字出现单位错误，处理情况如下：

如果设置 rmw 使能，将导致进入写访问异常 trap0x2b；

如果设置 rmw 不使能，将不作任何处理。

- 对于 8 位的 sram 和 32 位 sdram，如果对存储器进行字节或半字写，要写入的字节或半字所在的字出现单位错误：将不作任何处理，同设置 rmw 使能与否无关。
- 对于 32 位的 sram，如果对存储器进行字节或半字写，要写入的字节或半字所在的字出现非单位错误，处理情况如下：

  如果设置 rmw 使能，将导致进入写访问异常 trap0x2b；

  如果设置 rmw 不使能，将不作任何处理。
- 对于 8 位的 sram 和 32 位 sdram，如果对存储器进行字节或半字写，要写入的字节或半字所在的字出现非单位错误：将不作任何处理，同设置 rmw 使能与否无关。

- 如果 EDAC 配置成不使能状态（即 MECFG1 中的 Sr1EEn、Sr2EEn、Sr3EEn、Sr4EEn、Sr5EEn 和 SdEEn 清零），不论 ERB、EWB 如何配置，这种情况不检错和不纠错，不产生任何陷阱和中断。
- 如果 ERB 置位，而不论 EWB 如何配置，也不论 EDAC 使能如何配置，这种情况不检错和不纠错，不产生任何陷阱和中断。当读存储器时，读得的正常的校验位存入 MECFG1 中的 TCB 字段。

15. 8 位、32 位 PROM 出错处理

- 如果 PROM 的 EDAC 配置成使能状态，即 MECFG1 中的 romEEn 置位，那么当对 8 位、32 位 PROM 进行访问时，
  - 如果读操作出现单位错误，那么存储器控制器将此数据进行纠正并向下传递，但不回写存储器。这种情况不产生任何陷阱和中断。
  - 如果读操作时出现非单位错误，处理情况如下：

    如果此时处于读数据操作，将导致进入数据访问异常 trap0x09；

    如果此时处于取指操作，将导致进入指令访问异常 trap0x01。
- 如果 PROM 的 EDAC 配置成不使能状态，即 MECFG1 中的 romEEn 清零，那么当对 8

位、32 位 PROM 进行访问时，不检错和不纠错，

- 如果读 PROM 数据，不论数据位或校验位出错，不产生任何陷阱和中断。
- 如果读 PROM 指令，出现单位错误或非单位错误，使得指令变成非法指令，将导致进入指令访问异常 trap0x01。（校验位不理会）

16. 造错补充说明

对于 32 位 SRAM 和 SDRAM，字节写或半字写不能通过 MECFG2 和 MECFG3 对数据位和校验位造错，整字写可以通过 MECFG2 和 MECFG3 对数据位和校验位造错（需要 MECFG1 中 EWB 设置有效）。

对于 8 位 SRAM，字节写、半字写或整字写都不能通过 MECFG2 对数据位造错。可以通过字节写或半字写直接写入错误数据。

对于 8 位 SRAM，字节写或半字写不能通过 MECFG3 对校验位造错，整字写可以通过 MECFG3 对校验位造错（需要 MECFG1 中 EWB 设置有效）。

17. 纠错补充说明

作纠错处理的前提是纠错使能并检出单位错。如果作纠错处理，则不报错。

18. 字节和半字写补充说明

对于 8 位、32 位 SRAM，在 EDAC 纠错模式情况下，如果进行字节或半字写操作，则自动进入“读改写”操作而不论 rmw 配置位如何配置。

对于 SDRAM，在 EDAC 纠错模式情况下，如果进行字节或半字写操作，则自动进入“读改写”操作。

19. EDAC 编码匹配注意事项说明

对 8 位 PROM、32 位 PROM、8 位 SRAM、32 位 SRAM 和 32 位 SDRAM 的访问，打开 EDAC 使能前，要保证存储器中数据位和校验位的 EDAC 编码匹配，避免由于字节或半字写操作，造成存储器中数据位和校验位的 EDAC 编码不匹配。

20. 8 位 PROM 和 8 位 SRAM 存储体数据位和校验位注意事项说明

8 位 PROM 和 8 位 SRAM，在使用 EDAC 功能时，存储体内四分之三后用于存放校验位，不能存放程序和数据。

21. 存储器管脚连接补充说明

32 位宽数据位 PROM 存储器同它的 8 位宽校验位 PROM 存储器，共用相同的片选、地址、读使能、写使能信号，数据管脚分别接处理器的 data 和 pdata。

32 位宽数据位 SRAM 存储器同它的 8 位宽校验位 SRAM 存储器，共用相同的片选、地址、读使能信号，数据管脚分别接处理器的 data 和 pdata。校验位 SRAM 存储器的写使能信号用处理器的 rwen[0]。

32 位数据宽度数据位 SDRAM 存储器同它的 8 位数据宽度校验位 SDRAM 存储器，共有相同的片选、地址、行选择、列选择、写使能信号，数据管脚分别接处理器的 data 和 pdata。校验位 SDRAM 存储器的 DQM[0]接处理器的 sddqm[3]信号，校验位 SDRAM 存储器的 DQM[3:1]接高。

22. EDAC 编码

32 位数据位对应 7 位校验位生成公式如下：

PD0 = D0 ^ D4 ^ D6 ^ D7 ^ D8 ^ D9 ^ D11 ^ D14 ^ D17 ^ D18 ^ D19 ^ D21 ^ D26 ^ D28 ^ D29 ^ D31。

PD1 = D0 ^ D1 ^ D2 ^ D4 ^ D6 ^ D8 ^ D10 ^ D12 ^ D16 ^ D17 ^ D18 ^ D20 ^ D22 ^ D24 ^ D26 ^ D28。

~PD2 = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31。

~PD3 = D0 ^ D1 ^ D5 ^ D6 ^ D7 ^ D11 ^ D12 ^ D13 ^ D16 ^ D17 ^ D21 ^ D22 ^ D23 ^ D27 ^ D28 ^ D29。

PD4 = D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D14 ^ D15 ^ D18 ^ D19 ^ D20 ^ D21 ^ D22 ^ D23 ^ D30 ^ D31。

PD5 = D8 ^ D9 ^ D10 ^ D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^

D30 ^ D31。

PD6 = D0 ^ D1 ^ D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31。

公式中“~”表示“非”，“^”表示“异或”。

4.8.10 写保护

写保护功能对 RAM 空间(0x4000-0000 到 0x7FFF-FFFF)的写操作进行保护，支持两个写保护寄存器（写保护寄存器 WPR1 和写保护寄存器 WPR2），分别对不同的地址空间进行写保护。

WPR[31]，即 EN，“1”表示使能，“0”表示不使能。

WPR[30]，即 BP，“1”表示 block 保护模式，“0”表示 segment 保护模式。

WPR[29:15]，即 TAG，比较地址。

WPR[14:0]，即 MASK，掩码地址。

对 RAM 进行写操作，RAM 对应的内部总线地址 ADDR[29:15]同 TAG 进行“异或”，再同 MASK 进行“与”，结果为 result。写保护结果见表 4-8-16。

表 4-8-16 写保护结果表

*EN*配置	*BP*配置	result	写操作是否保护
1	1	为 0	是
1	0	不为 0	是

注意 如果写保护要使用 segment 模式，即 BP 配置为“0”，只能使能一个写保护寄存器，两个写保护寄存器不能同时使用 segment 模式。

图示举例说明

RAM 写保护举例图示，见图 4-8-25、图 4-8-26。

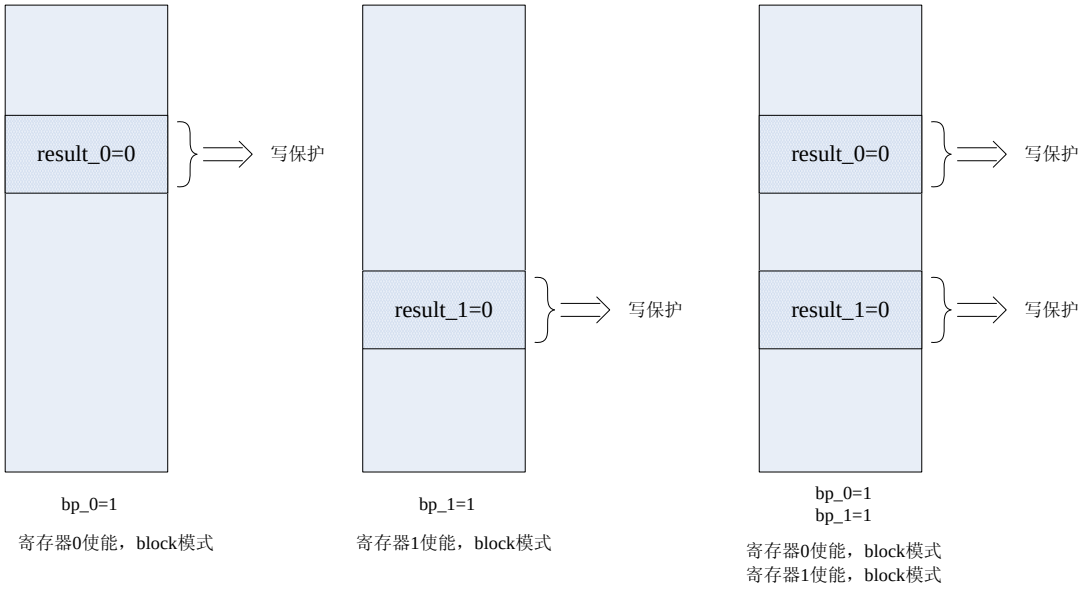


图 4-8-25 block 模式写保护

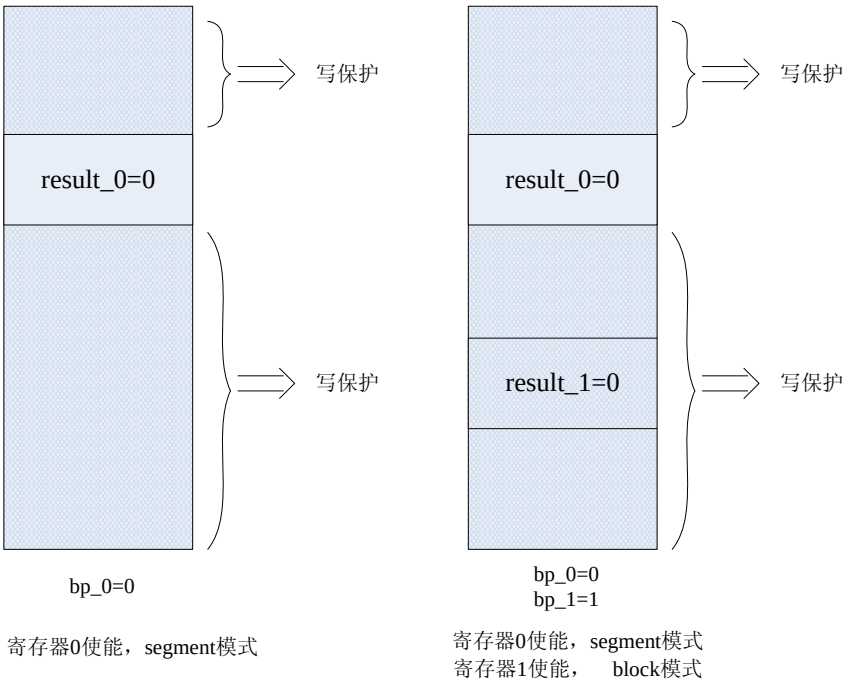


图 4-8-26 segment 模式以及分别两种模式写保护

4.9 PCI 接口和仲裁器

4.9.1 概述

BM3803FMGRH PCI 接口为 32 位 AHB to PCI 总线桥，实现了 AHB2.0 总线和 PCI2.3 33MHz PCI 总线之间的协议转换。BM3803FMGRH PCI 接口可配置成主桥形式即 Host Bridge 或从桥形式即 Guest Bridge，对应 PCI 总线上的 Host 设备和 Guest 设备。此模块实现了标准 32 位 PCI 接口的所有必选信号和可选信号中的全部 PCI 中断信号。

PCI 总线上的 Host 设备和 Guest 设备如图 4-9-1。图 4-9-1 左侧，BM3803FMGRH 处于 Host 模式，配置总线上其他 PCI 设备，并为其他设备提供总线仲裁。下图右侧，BM3803FMGRH 处于 Guest 模式，被 PCI Host 设备配置，在总线 Host 的管理下进行总线通信。

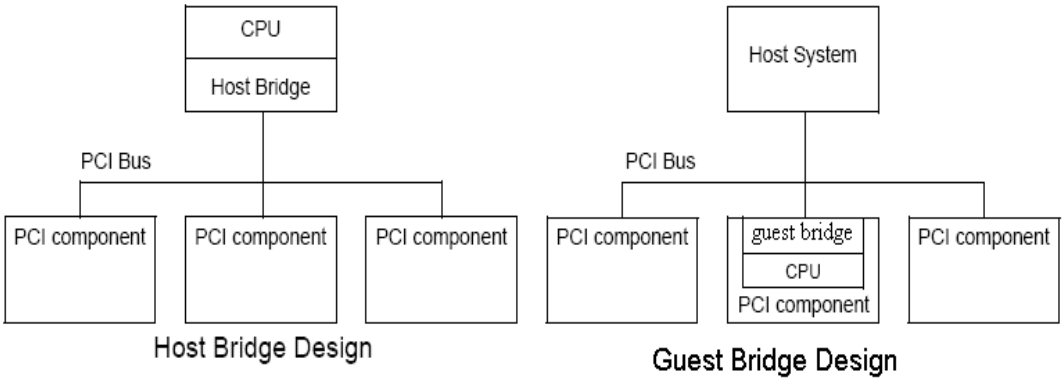


图 4-9-1 PCI 总线设备

4.9.2 AHB-PCI 模块结构

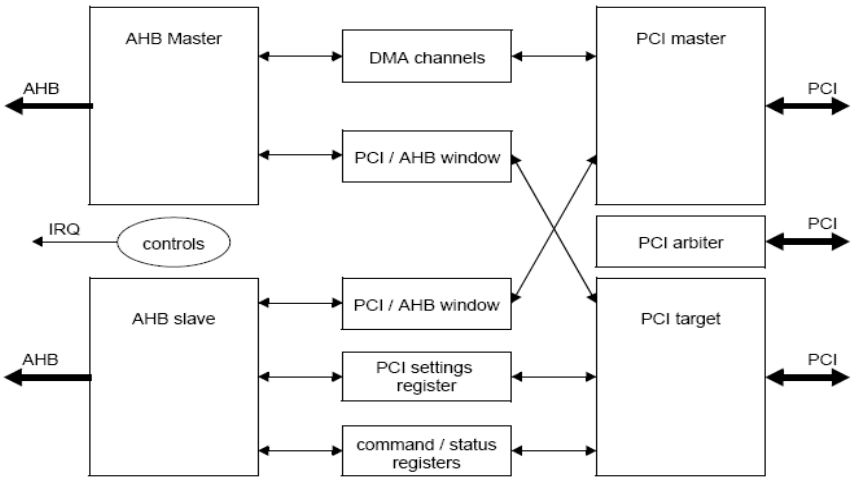


图 4-9-2 AHB-PCI 桥模块结构框图

PCI 接口结构如图 4-9-2 所示，PCI master/target 接口将 AHB 系统连接到 PCI 总线上；

BM3803FMGRH 处理器通过 AHB slave 模块控制和配置 AHB-PCI 桥模块；BM3803FMGRH 处理器可通过 DMA 通道配置和访问 PCI 总线设备。Host 模式下的 AHB-PCI 桥包含一个内嵌的 PCI 仲裁模块，此模块可管理 7 个外部 PCI 设备。PCI-AHB 地址空间映射即 PCI-AHB window 使 PCI 总线 Master 设备可以直接访问 CPU 存储器和 AHB 设备。

AHB 总线 Master 设备可以通过 AHB-PCI 地址空间映射即 AHB-PCI window 读写存储器和 I/O 空间来访问 PCI 总线设备。

4.9.3 PCI 接口复位

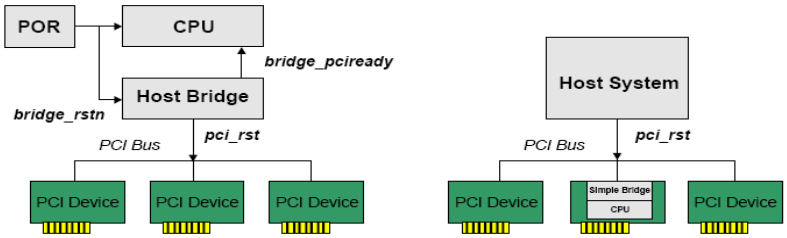


图 4-9-3 PCI 总线复位

如图 4-9-3 所示。Host 模式下的 BM3803FMGRH 的 PCI 接口，在 BM3803FMGRH 电路复位时发出 PCI 总线复位信号，并报告 PCI 总线复位是否完成。BM3803FMGRH 向 AHB 地址 0x8000-00D0 寄存器写入 0x05 可以使 PCI 总线复位。

Guest 模式下的 BM3803FMGRH 的 PCI 接口，在接收到 PCI 总线上的复位信号时 PCI 侧逻辑复位，在接收到 BM3803FMGRH 复位信号时 AHB 侧逻辑复位。

4.9.4 PCI 接口 AHB 侧地址空间

BM3803FMGRH PCI 接口，映射到四块独立的 AHB 总线空间，如图 4-9-4 所示。

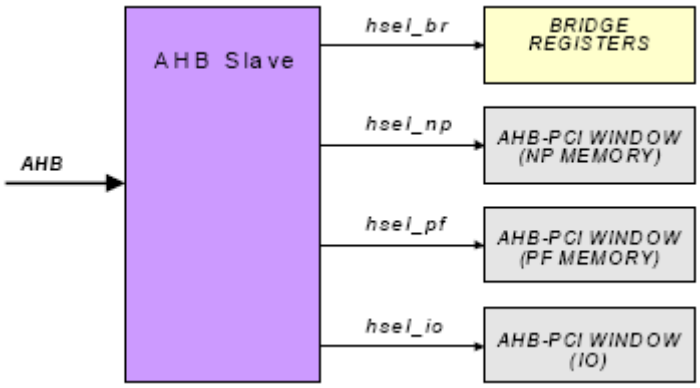


图 4-9-4 AHB-PCI 地址映射

空间分配如下：

寄存器空间即 bridge registers： 0xC000-0000 ~ 0xC7FF-FFFF

AHB-PCI 不可预取地址空间映射，即 NP：0xD000-0000 ~ 0xDFFF-FFFF

AHB-PCI 可预取地址空间映射，即 PF：0xE000-0000 ~ 0xFFFF-FFFF

AHB-PCI I/O 地址空间映射，即 I/O：0xC800-0000 ~ 0xCFFF-FFFF

其中 AHB 侧桥寄存器空间，基地址为 0xC000-0000，偏移地址如表 4-9-1 所示。

表 4-9-1 AHB 侧 PCI 相关映射寄存器

Byte Offset	Mode	Name	Description
000h	R/W	WDMA_PCI_ADDR	Write DMA start address on the PCI bus
004h	R/W	WDMA_AHB_ADDR	Write DMA transfer start address on the AHB bus
008h	R/W	WDMA_CONTROL	Write DMA size & control
00Ch ... 01Ch	reserved		
020h	R/W	RDMA_PCI_ADDR	Read DMA start address on the PCI bus
024h	R/W	RDMA_AHB_ADDR	Read DMA transfer start address on the AHB bus
028h	R/W	RDMA_CONTROL	Read DMA size & control
02Ch ... 03Ch	reserved		
040h	R/W	CPU_IMASK	Interrupt mask
044h	R/W/C	CPU_ISTATUS	Interrupt status
048h	W	CPU_ICMD	Interrupt command
04Ch	R	CPU_VERSION	Bridge version and miscellaneous information
050h ... 06Ch	reserved		
070h	R/W	PCIAHB_ADDR_NP	PCI-AHB window non-prefetchable range control

074h	R/W	PCIAHB_ADDR_PF	PCI-AHB window prefetchable range control
078h	R/W	PCIAHB_TIMER	PCI-AHB window discard timer
07Ch	R/W	AHBPCI_TIMER	AHB-PCI window discard timer
080h	R/W/C	PCI_CONTROL	PCI control bits
084h	R or R/W	PCI_DV	PCI device and vendor ID
088h	R or R/W	PCI_SUB	PCI subsystem device and vendor ID
08Ch	R or R/W	PCI_CREV	PCI class code and revision ID
090h	R/W/C	PCI_BROKEN	PCI arbiter broken master register
094h	R or R/W	PCIAHB_SIZ_NP	PCI-AHB window non-prefetchable range size
098h	R or R/W	PCIAHB_SIZ_PF	PCI-AHB window prefetchable range size
09Ch ... 3FCh	reserved		

4.9.5 PCI 仲裁器

PCI 仲裁模块在 BM3803FMGRH 处于 PCI Host 模式时有效，此模块提供 BM3803FMGRH PCI 接口自身的仲裁信号和最多七个其他 PCI 设备的仲裁信号。此模块的功能是：

- 使能 PCI 总线 Master 设备以开始 DMA 传输；
- 避免 PCI 总线冲突；
- 保证所有的 PCI 总线 Master 设备以同等的机会有规则地传送数据。

仲裁器能发现屏蔽并报告出错的 PCI 总线设备，以使其不影响 PCI 系统的性能。如果一个 PCI Master 设备请求 PCI 总线 16 个时钟周期以上而不开始一次总线操作，仲裁器就认为此 PCI 总线设备功能不正常。

PCI_Broken 寄存器的低 8 位反映 PCI 总线 Master 设备的状态，其中任 1 位为 1 则说明对应的 PCI Master 设备功能不正常而且被禁止访问 PCI 总线。向此位写 1，则重新使能相应的 PCI Master 设备。

4.9.6 PCI 侧寄存器空间

Guest 模式下，BM3803FMGRH PCI 接口实现了 PCI 标准配置空间，除此之外，PCI 侧寄存器还包括标准 PCI 中断的桥寄存器和 PCI 版本寄存器即这些寄存器地址在 0x80 ~ 0x8c 之间。PCI 侧寄存器的地址空间如表 4-9-2 所示。

表 4-9-2 PCI 配置寄存器地址空间

Offset	Mode	Name	Description
00h…3Ch	—	—	Standard PCI Header
40h…7Ch	—	—	reserved
80h	R/W	PCI_IMASK	Interrupt mask
84h	R/W/C	PCI_ISTATUS	Interrupt status
88h	W	PCI_ICMD	Interrupt command
8Ch	R	PCI_VERSION	Bridge version and miscellaneous information
90h…FCh	—	—	reserved

PCI 接口处于 Guest 模式时，实现了 PCI2.3 协议规定的标准 PCI 配置寄存器中，六个 PCI 地址空间定义寄存器 bar0---bar5 中的 bar3-bar5，如表 4-9-3 所示。

表 4-9-3 PCI 目标接口地址空间

Space	Size	Resource
BAR0...BAR2		reserved
BAR3	user defined	PCI-AHB window non-prefetchable area
BAR4	user defined	PCI-AHB window prefetchable area
BAR5	256 bytes	same as configuration space (Guest Bridge only)

当 BM3803FMGRH PCI 接口处于 Host 模式时，并不存在 PCI 侧寄存器，但可以通过配置读写对其他 PCI 总线设备进行配置。

PCI Host 设备为 PCI 总线上的 BM3803FMGRH PCI Guest 设备资源分配地址空间，包括 Memory 和 I/O，需设置 PCI 配置空间的 BAR3、BAR4、BAR5 三个基地址寄存器，BM3803FMGRH PCI 设备配置空间中，PCI 头标区偏移 0x1c、0x20、0x24 的三个寄存器，即 BAR0，BAR1，BAR2 三个基地址寄存器未实现；其中 BAR5 为配置空间 PCI 总线基地址，这样可以让 BM3803FMGRH PCI Guest 通过 Memory 读写取代通过配置读写来访问其他设备配置空间中的各种寄存器的方式，如读写中断寄存器，或者重新配置配置空间中的 BAR3 或/和 BAR4 寄存器以实现地址空间重新分配。中断相关寄存器使用如图 4-9-10 所示。

4.9.7 地址空间映射

4.9.7.1 AHB-PCI 地址空间映射

AHB-PCI 地址空间映射将 AHB 总线读写操作转化为 PCI 总线读写操作。

AHB 总线 Master 设备使用 AHB-PCI 地址空间映射直接访问 PCI 设备。AHB-PCI 地址空间映射捕获指向 PCI 接口 AHB 空间的 AHB 操作，并将这些操作传输到 PCI 总线上。这些操作并不需要 BM3803FMGRH 的 IU 等其他模块介入。

AHB-PCI 地址空间映射如图 4-9-5 所示，包括不可预取存储器映射空间、可预取存储器映射空间，可预取 I/O 映射空间三部分，AHB 侧地址范围在 PCI 接口 AHB 侧地址范围之内。

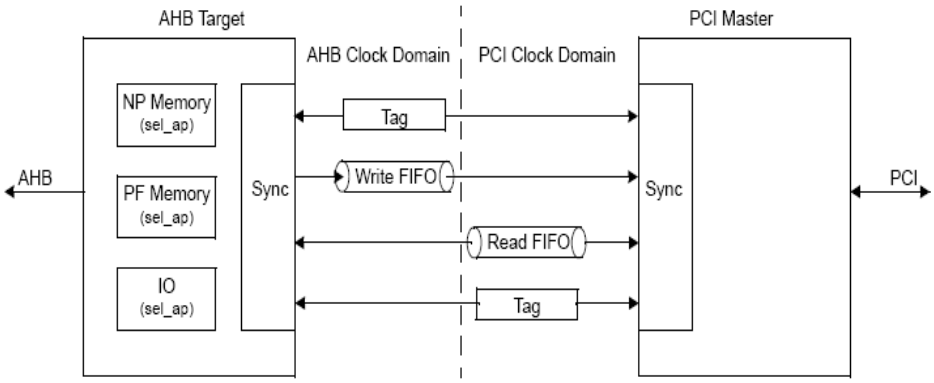


图 4-9-5 AHB-PCI 地址空间映射

AHB-PCI 地址空间映射使用扁平寻址方式，AHB 总线操作的地址与其转化成的 PCI 总线操作的地址相同。

具体说，BM3803FMGRH 访问 AHB 地址空间段 0xC800-0000 ~ 0xFFFF-FFFF 时，可以自动产生相应的 PCI 总线操作，即 PCI 接口访问 PCI 地址空间 0xC800-0000 ~ 0xFFFF-FFFF 地址时：

- 0xC800-0000 ~ 0xCFFF-FFFF 落在 AHB-PCI I/O 地址空间映射；将产生 PCI I/O 读写操作。
- 0xE000-0000 ~ 0xFFFF-FFFF 落在 AHB-PCI 可预取地址空间映射；将产生 PCI 可预取存储器读写操作。
- 0xD000-0000 ~ 0xDFFF-FFFF 落在 AHB-PCI 不可预取地址空间映射。将产生 PCI 不可预

取存储器读写操作。

如上所述，BM3803FMGRH 可以利用地址空间映射机制，通过 AHB 总线读写操作直接访问 0xC800-0000 ~ 0xFFFF-FFFF 范围的 PCI 总线地址，这个范围内的资源也可以通过 DMA 访问，包括可预取地址空间映射和不可预取地址空间映射以及 I/O 地址空间映射，其余的 PCI 地址空间则只能通过 PCI 接口的 DMA 机制访问。

4.9.7.2 PCI-AHB 地址空间映射

PCI-AHB 地址空间映射将 PCI 总线读写操作转化为 AHB 总线读写操作。

PCI 总线的其他 Master 设备通过 PCI-AHB 地址空间映射直接访问 BM3803FMGRH 本地存储器和片内 AHB 总线设备。PCI-AHB 地址空间映射捕获指向自己 PCI 空间的 PCI 操作，并将这些操作传输到 AHB 总线上。同 AHB-PCI 地址空间映射类似，这些操作并不将 IU 等其他模块牵连其中。

PCI-AHB 地址空间映射如图 4-9-6 所示，包括 Bar3、Bar4、Bar5 三部分，分别为不可预取 PCI 存储器映射空间、可预取 PCI 存储器映射空间和 PCI 接口 PCI 侧寄存器空间。地址转换的机制随 PCI 接口模式不同而不同。

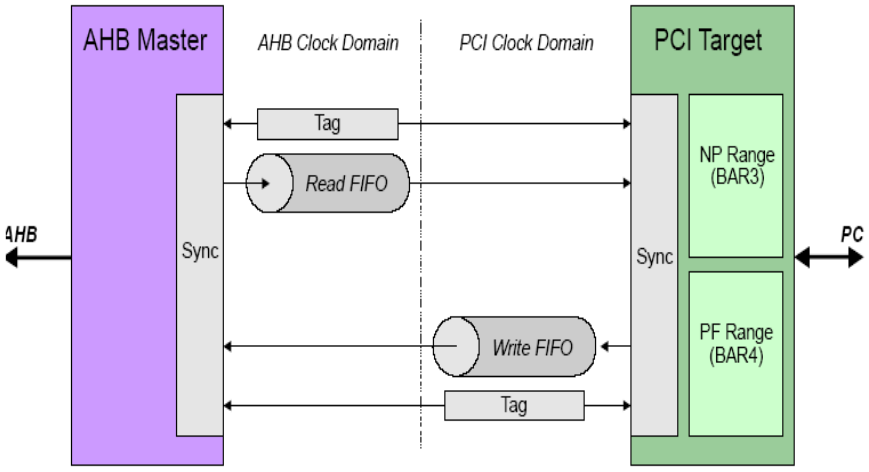


图 4-9-6 PCI-AHB 地址映射

PCI-AHB 地址空间映射包括 Bar3、Bar4、Bar5 三部分：

- 不可预取存储器映射空间被映射到 PCI 设备的 BAR3，处理指向此空间范围内的 PCI 操作不会进行数据预取。
- 可预取存储器映射空间被映射到 PCI 设备的 BAR4，处理指向此空间范围内的 PCI 操作会自动进行数据预取。
- PCI 接口 PCI 侧寄存器空间在 PCI 总线上被映射成 BAR5。这样使用存储器读写也可以修改 PCI 接口配置寄存器。

Bar3 和 Bar4 的大小在 256 字节和 1G 字节之间。其对应的 AHB 侧地址使用 AHB 侧寄存器定义。分别由 PCIAHB_ADDR_NP 和 PCIAHB_ADDR_PF 定义其 AHB 基地址，PCIAHB_SIZ_NP 和 PCIAHB_SIZ_PF 两个寄存器定义其大小。

在 Host 模式下，PCI-AHB 地址空间映射不进行地址转换，即 PCI 总线读写操作和其转换成的 AHB 总线读写操作目标地址一致。反过来说 Host 模式下 PCI 接口的 PCI-AHB 地址空间映射的 PCI 侧地址范围由 AHB 侧寄存器定义。

在 Guest 模式下，PCI 接口的 PCI-AHB 地址空间映射的 PCI 侧地址范围由 BAR3 base address、BAR4 base address、BAR5 base address 决定。PCI 空间和 AHB 空间之间存在如下转换：

不可预取空间：

AHB address = PCIAHB_ADDR_NP + (PCI address - BAR3 base address)；

可预取空间：

AHB address = PCIAHB_ADDR_PF + (PCI address - BAR4 base address)。

4.9.8 DMA

PCI 接口的很大一部分功能是通过 DMA 机制进行的，DMA 通道如图 4-9-7 所示，在 DMA 机制中 DMA 控制器在控制 PCI 总线的同时也控制 AHB 总线，实现独立于 CPU 的数据传输。

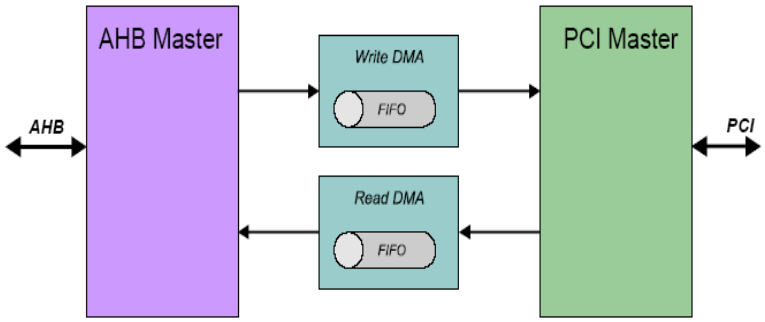


图 4-9-7 DMA 通道

DMA 机制可以在 AHB 总线和 PCI 总线之间实现大量的 Memory 或 I/O 数据传输而仅需要少量的 CPU 干预，也可以执行配置周期和 I/O 操作。

CPU 分配和初始化一段存储器缓冲区，然后初始化设置 DMA 寄存器，并指定 PCI 交易的类型和目标地址。交易被初始化为 AHB Master 后，CPU 再无任何动作，PCI Master 会自动执行所需要的 PCI 总线数据交易，除非用户终止 DMA 传输。DMA 传输状态将报告给 CPU_ISTATUS 寄存器，可以通过此寄存器引起 BM3803FMGRH 处理器中断。

在 DMA 操作中，DMA 读是指从 PCI 地址空间中向 BM3803FMGRH 的 AHB 地址空间中读取数据；DMA 写是指把 BM3803FMGRH AHB 地址空间中的数据写入 PCI 地址空间。

发起 DMA 操作需要三个寄存器：PCI 地址寄存器、AHB 地址寄存器和 DMA 控制寄存器。DMA 写传输需要配置的寄存器有：PCI 地址寄存器，地址为 0xC000_0000；AHB 地址寄存器，地址为 0xC000_0004；DMA 写控制寄存器，地址为 0xC000_0008。DMA 读传输需要配置的寄存器有：PCI 地址寄存器，地址为 0xC000_0020；AHB 地址寄存器，地址为 0xC000_0024；DMA 读控制寄存器，地址为 0xC000_0028。DMA 传输的 AHB 地址寄存器 WDMA_AHB_ADDR、RDMA_AHB_ADDR 会随着 DMA 的进行自动增长，直至 DMA 结束。

DMA 控制寄存器 DMA_CONTROL 定义了 PCI 总线上数据传输的 PCI 命令和控制操作，结构如图 4-9-8 所示。

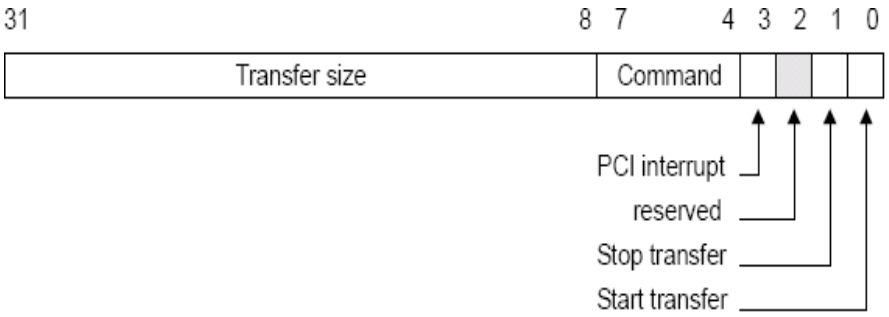


图 4-9-8 DMA 控制寄存器

- Bit[3] 置 1 可以使 DMA 完成时在 PCI 总线上产生中断信号。
- Bit[1] 置 1 可以终止正在进行的 DMA。
- Bit[0] 置 1 可以启动 DMA，完成时自动清零， DMA 完成/退出时会在中断状态寄存器相应的位置 1，使用时在 DMA 开始之后，等待并判断 DMA 完成时可以查询此位。

Transfer size 为 DMA 传输字节长度，最大为 16MB，取值为 0x1-0xFFFFFC；

Command 部分表示发起的 DMA 操作类型，如表 4-9-4 所示。

表 4-9-4 DMA 操作类型

Channel	Command	Description
Read DMA	0000	Interrupt Acknowledge
	0010	I/O Read
	0110	Memory Read
	1010	Configuration Read
	1100	Memory Read Multiple
Write DMA	0001	Special Cycle
	0011	I/O Write
	0111	Memory Write
	1011	Configuration Write

DMA 写操作示例：

*(unsigned int *)0xc0000000 = 0xc0000000; // Write DMA start address on the PCI bus

*(unsigned int *)0xc0000004 = 0x40000000; // Write DMA transfer start address on the AHB

bus

*(unsigned int *)0xc0000008 = 0x10071; // Write DMA size & control

发起把 AHB 地址 0x4000-0000 处 0x100 字节写往 PCI 地址 0xC000-0000 处的 DMA Memory Write 操作。

DMA 读操作示例：

*(unsigned int *)0xc0000020 = 0x40000000;

*(unsigned int *)0xc0000024 = 0x40001000;

*(unsigned int *)0xc0000028 = 0x100a1;

则读出了槽位地址在 0x4000-0000 处的 PCI 设备配置空间中 0x100 字节的内容。

BM3803FMGRH PCI Host 设备通过配置 DMA 进行 PCI 配置时，应尽量使用单字 DMA 读写。配置 DMA 读写时 PCI 地址的确定由 PCI 总线底板的槽位及选择关系决定。

通过 DMA 操作产生 PCI 总线类型 0 和类型 1 配置周期时的 PCI 总线地址格式如图 4-9-9 所示。

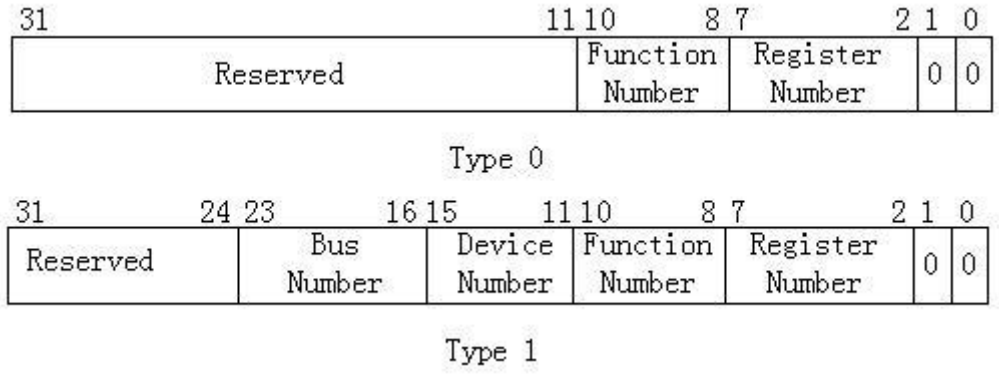


图 4-9-9 Type 0 和 Type 1 地址空间配置

BM3803FMGRH AHB 总线为大端总线，而 PCI 总线为小端总线，所以他们之间的 DMA 传输需按照一定的要求来保证数据顺序一致。

当 BM3803FMGRH DMA 目标或源地址为 32 位的 RAM 空间且进行 DMA 传输的 PCI 设备为两个 BM3803FMGRH PCI 设备时，需遵循以下原则以避免字节序错位：

- PCI 地址和 AHB 地址后两位相等时，进行任意类型字节长度的 DMA；

- PCI 地址和 AHB 地址 2 字节即半字对齐时，DMA 长度为 2 或 2 的倍数；
- PCI 地址和 AHB 地址 4 字节即字对齐时，DMA 长度为 4 或 4 的倍数。

当 BM3803FMGRH DMA 目标或源地址为 8 位 RAM 空间时，则需要 PCI 地址和 AHB 地址 4 字节对齐，DMA 长度为 4 字节或 4 的倍数。

推荐使用 PCI 地址和 AHB 地址 4 字节对齐，DMA 长度为 4 字节或 4 的倍数的 DMA 方式，此时无论何种情况都不会被大小端问题影响。

4.9.9 PCI 模块中断

PCI 模块中断既可以通过 AHB 总线向 BM3803FMGRH 的 IU 模块发出中断请求，也可以在 Guest 模式下通过 PCI 总线发出中断或响应来自 Host 模式 PCI 总线的中断请求。前者称为处理器中断，后者称为 PCI 中断。

4.9.9.1 处理器中断

PCI 接口的处理器中断为 BM3803FMGRH 处理器的 14 号中断，PCI 部分配置为 Host 或 Guest 时，其实现稍有不同，如图 4-9-10 所示。AHB 侧有三个中断相关寄存器：

- 0xC000-0040 ： CPU_IMASK　　CPU 中断掩码寄存器。
- 0xC000-0044 ： CPU_ISTATUS　　CPU 中断状态寄存器。
- 0xC000-0048 ： CPU_ICMD　　CPU 中断命令寄存器。

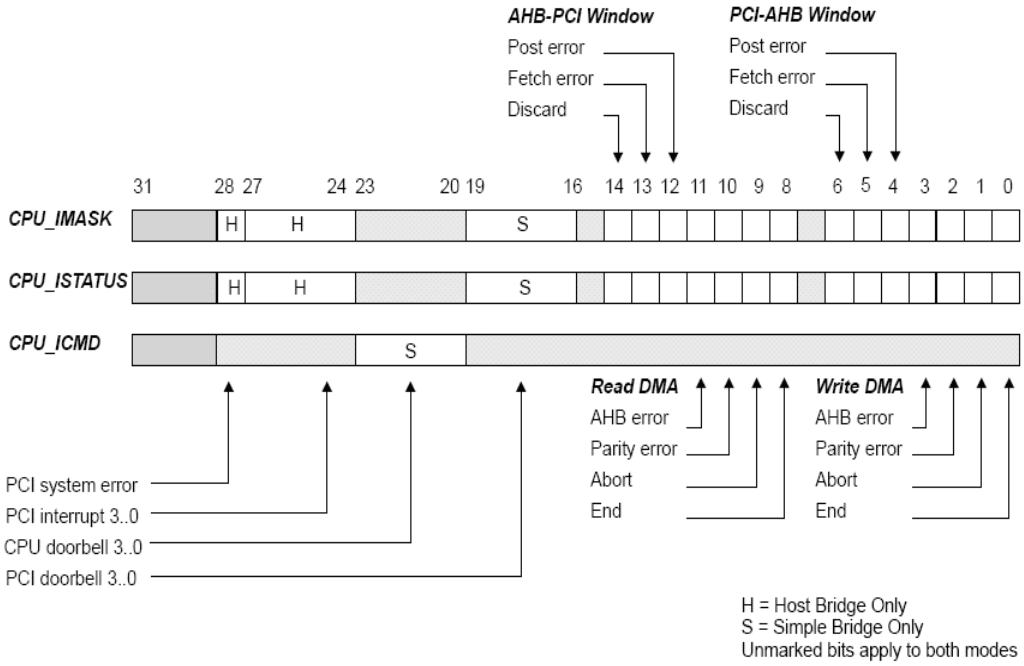


图 4-9-10 CPU 中断寄存器

- CPU_ISTATUS：CPU 中断状态寄存器，此寄存器可读/可写/可清零。相应的中断源被激活，状态寄存器相应位自动置位为 1，各个中断源是相互独立的且多个中断源可以同时触发。CPU 可以监控并清除状态位，写 1 清除相应状态位，写 0 无效。
- CPU_IMASK：CPU 中断掩码寄存器，此寄存器可读/可写。寄存器置位使能相应的中断源，清零屏蔽相应的中断源。上述操作不影响 CPU_ISTATUS 寄存器各位的值：也就是说被屏蔽的中断仍会报告给相应的状态位。如果一个或多个中断源被触发且未被屏蔽，则 PCI 桥就会向 CPU 申请中断。
- CPU_ICMD：CPU 中断命令寄存器，只写寄存器，用于人为使能一些中断源。读操作返回 0。写 1 置位，写 0 无效。

4.9.9.2 PCI 中断

PCI 侧配置空间中也有三个中断寄存器，PCI_IMASK、PCI_ISTATUST、PCI_ICMD，分别为 PCI 中断掩码寄存器，PCI 中断状态寄存器，PCI 中断命令寄存器。这三个寄存器的 CPU Doorbell 位和 PCI Doorbell 位和 CPU 中断寄存器的对应位配合使用产生中断请求，这些位只在 Guest 模式下有效。这三个寄存器只能通过 PCI 总线读写，如图 4-9-11 所示。

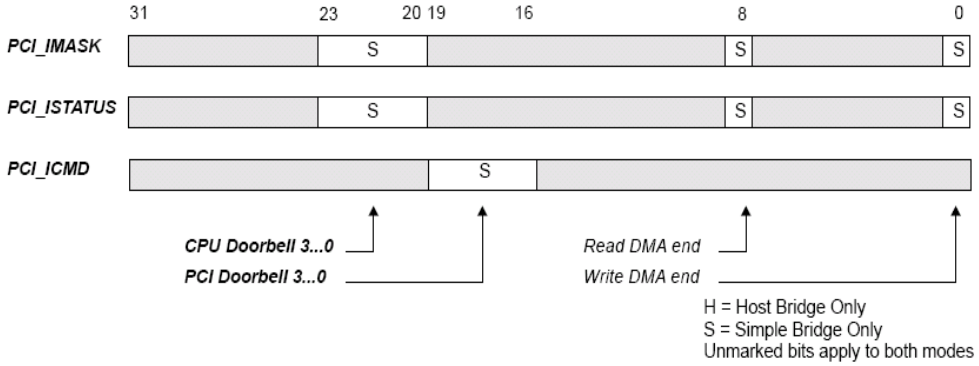


图 4-9-11 PCI 中断寄存器

- PCI Doorbell：PCI 设备可以通过访问此位向 BM3803FMGRH 中断控制器发出中断请求。若 PCI Doorbell 中断在 CPU_IMASK 寄存器中使能，此时置位 PCI Doorbell 位，则 BM3803FMGRH 的 CPU 中断中的 PCI Doorbell 中断有效。该位只写，写 0 无效，写 1 置位。此位只在 BM3803FMGRH 配置为 Guest bridge 模式下有效。
- CPU Doorbell：CPU Doorbell 中断由 BM3803FMGRH 写 CPU_ICMD 对应位触发。若此时 PCI_IMASK 寄存器中断源被使能，则 BM3803FMGRH 可以通过 PCI 总线 INTA#发送用户自定义的中断，并访问 PCI_ISTATUS 寄存器查询中断类型。此位只在 BM3803FMGRH 配置为 Guest bridge 模式下有效。
- Read DMA end 和 Write DMA end 的使用和 CPU Doorbell 类似，也是通过 PCI 总线的 INTA#信号向 PCI 总线 Host 设备发出中断信号，中断由 DMA 完成事件触发。此位只在 BM3803FMGRH 配置为 Guest bridge 模式下有效。
- PCI 中断使用说明：PCI 中断为电平中断，处理器响应 14 号中断即 PCI 中断后，需要软件首先清除中断源 PCI 模块内的中断请求，即向 CPU_ISTATUS 寄存器中相应位写 1，然后清除 14 号中断，从而清除相应 PCI 中断。

4.10 调试支持单元

4.10.1 概述

BM3803FMGRH 具有调试支持单元，方便软件在目标硬件上进行调试。通过 DSU 可以将 BM3803FMGRH 置于调试模式，允许对 BM3803FMGRH 中所有寄存器及 Cache 存储器进行

读写访问。DSU 同时具有一个大小为 512*16 字节的跟踪缓冲区 Tracebuffer，保存已经被执行的指令和总线上传输过来的数据。调试通讯链接实现了一个简单的读/写协议并使用标准的异步串口通讯。

DSU 控制寄存器在任何时候都可以被访问。而 BM3803FMGRH 的寄存器和 Cache 只能在 BM3803FMGRH 进入调试模式时被访问。Tracebuffer 只能在跟踪被禁用或者跟踪完成时被访问。在调试模式下，DSU 控制 BM3803FMGRH，流水线暂停。

仅当 DSU 的外部引脚 dsuen 使能的情况下，才能进入调试模式。dsuen 引脚拉高，使能调试模式。

使能调试模式后，通过下列任一事件 BM3803FMGRH 可以进入调试模式：

- 执行一个断点指令 ta 1；
- 观测点命中；
- 在系统复位结束前三个时钟周期 dsubre 为高电平，处理器在复位结束后运行四条指令后立即进入调试模式；
- 设置 **DSU 控制寄存器**中的 *BN* 位；
- 进入错误模式的陷阱；
- **DSU 控制寄存器**中定义的一个或多个陷阱触发；
- 执行一个单步操作；
- DSU 断点命中。

当进入调试模式后，BM3803FMGRH 发生以下改变：

- **程序计数器 PC** 和 **nPC** 被存入临时寄存器（可通过调试单元访问）；
- DSU 状态指示信号 dsuact 输出高电平；
- 时钟单元停止，冻结定时器和看门狗。

通过清除 **DSU 控制寄存器**中的 *BN* 位或取消 dsuen 信号可以使 BM3803FMGRH 恢复运行模式，此时定时器恢复计时，BM3803FMGRH 从保存的 **PC** 和 **nPC** 处开始执行。BM3803FMGRH 进入陷阱后也可以进入调试模式，例如：应用程序运行中，发生了非屏蔽陷阱，此时应用程序中止，暂停执行。

4.10.1.1 时间标志

DSU 时间标志计数器的值作为执行操作的时间参考。BM3803FMGRH 运行时，计数器每时钟周期加一；BM3803FMGRH 进入调试模式时计数器停止，重新开始运行时计数器重新开始计数。

4.10.1.2 Tracebuffer

Tracebuffer 是一个环形缓冲区，存储执行的指令或者内部总线上传输的数据，其大小为 512 行，每行 16 字节。Tracebuffer 由 **DSU 控制寄存器**和 **Tracebuffer 控制寄存器**控制。当 BM3803FMGRH 进入调试模式时，跟踪暂停。

Tracebuffer 有三种工作模式：

- 指令跟踪模式（只记录执行的指令）
- 总线跟踪模式（只记录内部总线上传输的数据）
- 混合跟踪模式（同时记录执行指令和总线数据）

Tracebuffer 控制单元包含了两个计数器（总线计数器和指令计数器），表示当前跟踪记录写入 Tracebuffer 的位置。由于 Tracebuffer 是环形的，因此计数器值实际上指向 Tracebuffer 当前记录位置，每次对 Tracebuffer 的记录完成后，计数器值自动增加。

（1）指令跟踪

Tracebuffer 控制寄存器中的 *TI* 位置 1，使能指令跟踪模式。

在指令跟踪期间，除多周期指令，一条指令占用缓冲区的一行。多周期指令占用跟踪缓冲区两行或三行：

- 对于存储指令，Tracebuffer 中第一行[95:64]用于记录存储地址；第二行相同字段存储写入的数据。对 STD 指令，需要使用三行记录，第一行[95:64]字段，记录存储地址，第二行相同字段存储写入数据的高 32bit，第三行相同字段存储写入数据的低 32bit。当一行的 Bit 126 置 1 时，表示这是一条多周期指令的第二行或第三行记录。
- 双字加载指令(LDD)使用两行记录，第一行[95:64]字段存储读出数据高 32bits，第二行相同字段存储读出数据的低 32bits。
- 乘法和除法指令使用两行记录。对乘法指令，第一行[95:64]字段记录计算结果的高

32bits，第二行[95:64]字段记录计算结果的低 32bits；对除法指令，第一行[95:64]字段记录为符号扩展，第二行[95:64]字段记录 32bits 运算结果。Bit 126 置位，表示同一指令记录的第二行。

◆ 双精度浮点指令使用两行记录。第一行[95:64]字段存储计算结果的高 32bits，第二行[95:64]字段存储计算结果的低 32bits。Bit 126 置位，表示同一指令记录的第二行。

记录格式如图 4-10-1 和表 4-10-1 所示。

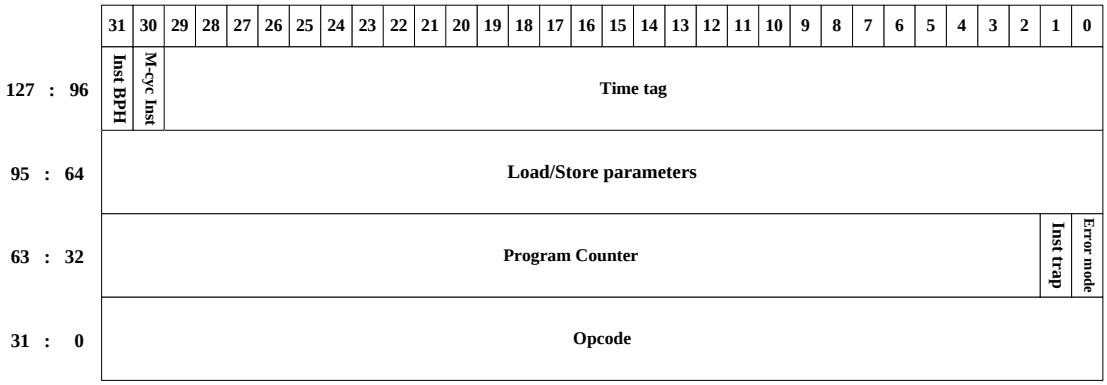


图 4-10-1 指令跟踪记录格式

表 4-10-1 指令跟踪记录格式

比特位	含义	说明
127	指令断点命中	如果出现 DSU 指令断点命中，设置为‘1’
126	多周期指令	多周期指令(LDD,ST 或 FPOP)的第二、三行设置为‘1’
125:96	时间标志	时间标志计数器的值
95:64	Load/Store 参数	指令结果、存储地址或者存储数据
63:34	程序计数器	程序计数器（不记录最低两位，因为它们总是 0）
33	指令陷阱	如果被跟踪的指令进入陷阱则设置为‘1’
32	处理器错误模式	如果被跟踪的指令引起处理器进入错误模式则设置为‘1’
31:0	Opcode	指令操作码

当跟踪被冻结时，产生 11 号中断。

（2）总线跟踪

将 **Tracebuffer 控制寄存器**的 *TA* 设置为 1，可以使能总线跟踪模式。

在总线跟踪期间，Tracebuffer 的一行记录一次内部总线上的操作。记录格式如图 4-10-2 和表 4-10-2 所示。

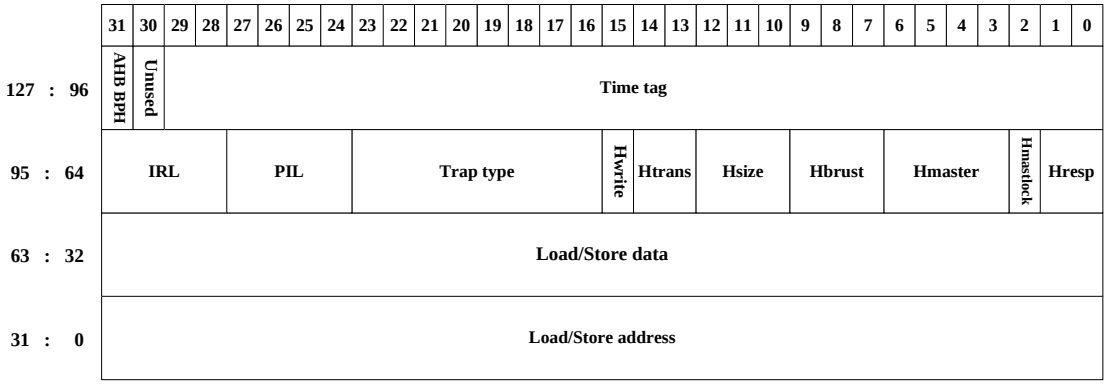


图 4-10-2 总线跟踪记录格式

表 4-10-2 总线跟踪记录格式

比特位	含义	说明
127	AHB 断点命中	如果 DSU AHB 断点命中则设为‘1’
126	-	未使用
125:96	Time tag	时间标志计数器的值
95:92	IRL	中断请求输入
91:88	PIL	中断级别(psr.pil)
87:80	Trap type	陷阱类型(psr.tt)
79	Hwrite	AHB HWRITE
78:77	Htrans	AHB HTRANS
76:74	Hsize	AHB HSIZE
73:71	Hburst	AHB HBURST
70:67	Hmaster	AHB HMASTER
66	Hmastlock	AHB HMASTLOCK
65:64	Hresp	AHB HRESP
63:32	Load/Store data	AHB HRDATA or HWDATA
31:0	Load/Store address	AHB HADDR

（3）混合跟踪

在混合模式下，Tracebuffer 被分为两半：指令存在低一半 Tracebuffer，总线传输存在高一半 Tracebuffer。AHB 索引计数器的 MSB 位自动保持为高，指令索引计数器的 MSB 保持为低，即 AHB 跟踪记录的起始地址为 0x9001-1000，指令跟踪记录的起始地址为 0x9001-0000。

（4）DSU 映射

表 4-10-3 DSU 映射表

地址	寄存器
0x8000-00c4	DSU 串口状态寄存器
0x8000-00c8	DSU 串口控制寄存器
0x8000-00cc	DSU 串口波特率重载寄存器
0x9000-0000	DSU 控制寄存器
0x9000-0004	Trace buffer 控制寄存器
0x9000-0008	DSU Trace 时间标志寄存器
0x9000-0010	AHB 断点寄存器 1
0x9000-0014	AHB 屏蔽地址寄存器 1
0x9000-0018	AHB 断点寄存器 2
0x9000-001C	AHB 屏蔽地址寄存器 2
0x9001-0000 ~ 0x9001-1FFF	Trace buffer(1)
0x9002-0000 ~ 0x9002-029F	IU/FPU 寄存器堆
0x9008-0000	Y 寄存器
0x9008-0004	PSR 寄存器
0x9008-0008	WIM 寄存器
0x9008-000C	TBR 寄存器
0x9008-0010	PC 寄存器
0x9008-0014	NPC 寄存器
0x9008-0018	FSR 寄存器
0x9008-001C	DSU 陷阱寄存器(2)
0x9008-0060	ASR24
0x9008-0064	ASR25
0x9008-0068	ASR26

0x9008-006C	ASR27
0x9008-0070	ASR28
0x9008-0074	ASR29
0x9008-0078	ASR30
0x9008-007C	ASR31
0x9010-0000 ~ 0x9010-7FFF	指令 Cache tag(3)
0x9010-0000 ~ 0x9010-1FFF	第一路
0x9010-2000 ~ 0x9010-3FFF	第二路
0x9010-4000 ~ 0x9010-5FFF	第三路
0x9010-6000 ~ 0x9010-7FFF	第四路
0x9014-0000 ~ 0x9014-7FFF	指令 Cache data
0x9014-0000 ~ 0x9014-1FFF	第一路
0x9014-2000 ~ 0x9014-3FFF	第二路
0x9014-4000 ~ 0x9014-5FFF	第三路
0x9014-6000 ~ 0x9014-7FFF	第四路
0x9018-0000 ~ 0x9018-3FFF	数据 Cache tag (4)
0x9018-0000 ~ 0x9018-1FFF	第一路
0x9018-2000 ~ 0x9018-3FFF	第二路
0x901C-0000 ~ 0x901C-3FFF	数据 Cache data
0x901C-0000 ~ 0x901C-1FFF	第一路
0x901C-2000 ~ 0x901C-3FFF	第二路

注：1、每行 128 位，共 512 行。

2、**DSU 陷阱寄存器**是一个只读寄存器，指出哪种 SPARC 陷阱导致处理器进入调试模式。当通过设置 **DSU 控制寄存器**中的 *BN* 位强制进入调试模式时，陷阱类型是 0x0b。

DSU 陷阱寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																			EM	TT								R			
r																			r	r								r			
0																			0	x								0			

位号	助记符	描述

31:29	R	保留位，读出为 0
12	EM	错误模式（EM），如果陷阱使得处理器进入错误模式，则置该位。
11:4	TT	8 位 SPARC 陷阱类型。
3:0	R	保留位，读出为 0

3、由于 Cache 结构，一行指令 Cache 的 tag 在 Tracebuffer 中的映射为连续的 8 个字，读写 8 个字中的任何一个都是对同一个 tag 操作。

4、由于 Cache 结构，一行数据 Cache 的 tag 在 Tracebuffer 中的映射为连续的 4 个字，读写 4 个字中的任何一个都是对同一个 tag 操作。

（5）DSU 断点

DSU 包含了两个断点寄存器来匹配任何内部的 AHB 总线地址或者执行的指令。断点命中的典型应用是冻结 Tracebuffer，但也可以把 BM3803FMGRH 置于调试模式。

冻结操作可以被延迟，通过把 **DSU 控制寄存器**中的 *TDELAY* 字段置为非零值来实现。这种情况下，*TDELAY* 的值会在每一次额外的跟踪后减一，递减至零时冻结 Tracebuffer。如果 **DSU 控制寄存器**中的 *BT* 置 1，则 Tracebuffer 冻结时，DSU 强制 BM3803FMGRH 进入调试模式。因为流水延迟，在 BM3803FMGRH 被置于调试模式前，最多有 4 个额外的指令可能被执行。针对每个断点，都可以使用屏蔽寄存器，针对某块地址进行屏蔽，仅有屏蔽寄存器中相应位为 1 时，才会在断点检测时进行比较。在执行指令时中断，应置 *EN* 位；在 AHB 加载或存储访问时中断，应置 *LD* 或 *ST* 位。

4.10.2 调试通讯链接

调试通讯链接包含一个串口，支持异步通信协议传输访问参数和数据。链接命令由一个控制字节、32 位地址和可选的写入数据组成。如果 **DSU 控制寄存器**的 *LR* 位置位，则每次 AHB 传输之后发送响应的字节。如果未设置 *LR* 位，则写访问不返回任何数据，读访问只返回读到的数据。

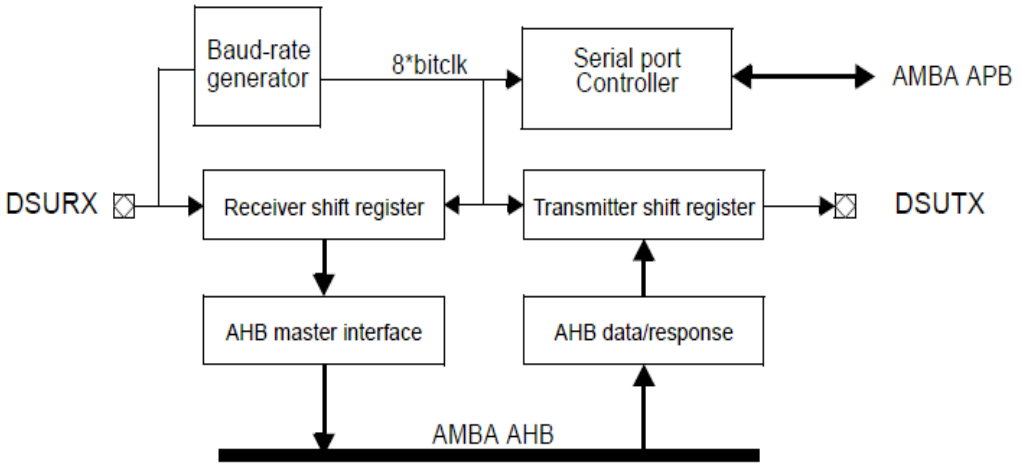


图 4-10-3 DSU 通讯链接框图

4.10.2.1 数据帧

数据以 8-bit 为基础进行发送，数据帧格式如图 4-10-4 所示。

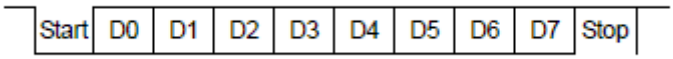


图 4-10-4 DSU 串口数据帧

4.10.2.2 调试控制命令

通过通讯链路，可以对内部总线上任何地址进行读/写操作。通过设置长度字段为 n-1（其中 n 标明传输字的长度），可以进行块传输。写访问时，发送控制字节、目标地址段首地址以及需要写入的数据，数据写入地址以目标地址段首地址为基地址，以字为单位自动递增。读访问时，发送控制字节和目标地址段首地址，返回相应大小的数据。DSU 命令如图 4-10-5 所示。

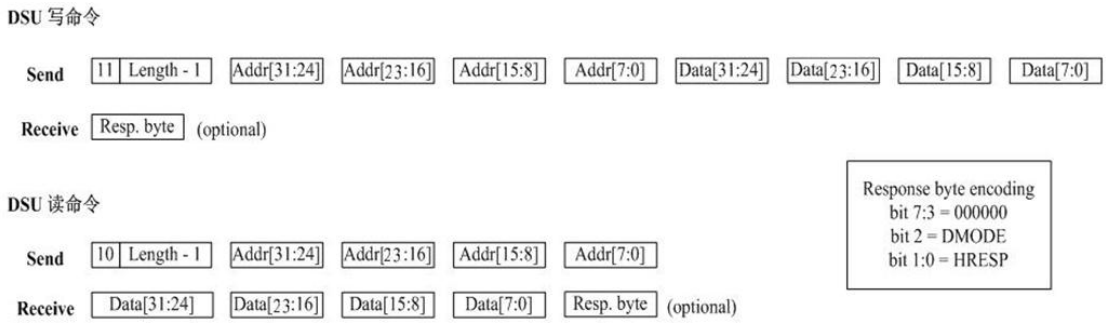


图 4-10-5 DSU 命令

4.10.2.3 串口通信时钟

时钟控制，每次下溢时产生一个串口 TICK。每次下溢之后分频器重新载入**串口重载寄存器**的值。串口 TICK 频率应该是期望波特率的 8 倍。

如果不进行软件编程，波特率能够被自动检测，这通过寻找接收数据(2-bit 周期)的两个下降沿的最短周期实现。当发现三个相同的 2-bit 周期时，相应的分频器重载值锁存入**重载寄存器**，设置 **DSU 串口控制寄存器**中的 *BL* 位。如果 *BL* 位通过软件复位，波特率检测过程重新开始。当接收器收到一个'break'时，波特率检测重新开始，这允许外部的发送器改变波特率。为了正确的检测波特率，在复位或者发送'break'之后应该向接收器发送值 0x55。

分频值计算公式如下：

$$\text{scaler} = \frac{\frac{\text{sysclk} \times 10}{\text{baudrate} \times 8} - 5}{10}$$

4.10.3 启动调试

将 dsuen 和 dsubre 管脚同时置为高电平时，在复位后 BM3803FMGRH 执行 0x0 地址处的四条指令后进入调试模式。此时可以通过 DSU 通信接口初始化系统，下载程序进行调试。

4.10.4 调试支持单元的寄存器地址映射

调试支持单元的寄存器地址映射见表 4-10-4，详细说明见附录 3.13。

表 4-10-4 DSU 寄存器地址映射对照表

寄存器名称	地址	复位值(1)
DSU 控制寄存器	0x90000000	0xXXX0XXXX
Trace buffer 控制寄存器	0x90000004	0x0XXXXXXX
DSU Trace 时间标志寄存器	0x90000008	0x00000000
AHB 断点寄存器 1	0x90000010	0xXXXXXXXX
AHB 屏蔽地址寄存器 1	0x90000014	0xXXXXXXXX
AHB 断点寄存器 2	0x90000018	0xXXXXXXXX
AHB 屏蔽地址寄存器 2	0x9000001c	0xXXXXXXXX
DSU 串口状态寄存器	0x800000c4	0x00000006
DSU 串口控制寄存器	0x800000c8	0x00000000

DSU 串口波特率重载寄存器	0x800000cc	0x0003FFFF

注：(1)“复位值”按照十六进制统一写法“0xXXX0XXXX”，“0x”表示 16 进制，每 4bit 中有 1bit 不定值，即写为“X”，4bit 全部为固定值则写为相应 16 进制数值。

4.11 时钟倍频

4.11.1 概述

BM3803FMGRH 的时钟倍频是通过 PLL 实现，系统时钟可以由外部时钟 clk 直接提供或者 clk 经过内部 PLL 提供。

4.11.2PLL 使用要求

PLL 的接口电平采用 3.3V，PLL 相关的管脚有 7 个，分别是：pll 旁路（pll_bp）、pll 倍频系数输入（pll_min[3:0]）、pll 复位（pll_rst）、pll 测试输出（pll_out）。

pll_bp 设置为低电平时，系统时钟由外部时钟 clk 经过内部 PLL 提供，pll_bp 设置为高电平时 pll 被旁路，系统时钟由外部时钟 clk 直接提供。

pll_bp 设置为低电平时，pll_min[3:0]表示 pll 的倍频系数，pll_min[3:0]不能同时为零（取零时无法保证 pll 频率的锁定）。pll_min[3:0]倍频系数为 1 到 15。

⚠注意

1、pll 复位信号 pll_rst，高电平有效，板级设计时需要下拉。

2、pll 输出时钟信号 pll_out，板级设计时悬空。

3、当 pll_bp 设置为低电平时，系统时钟由内部 PLL 提供。为保证 PLL 稳定工作，外部时钟 clk 的选择必须同时满足以下三个条件：1、输入频率范围 2MHz-30MHz；2、占空比范围 45%-55%；3、倍频后的时钟范围 40MHz-100MHz。

4、PLL 输出稳定时钟需要一定时间间隔，BM3803FMGRH 的 3.3V 和 1.8V 电源稳定后经过 10ms 时间间隔 PLL 输出稳定时钟。

4.12 JTAG 接口

4.12.1 概述

BM3803FMGRH 实现了符合 IEEE 1149.1 JTAG 规范的标准接口，此接口可用于板级测试。

4.12.2JTAG 使用要求

JTAG 接口包括 5 个管脚，如表 4-12-1 所示。

表 4-12-1 JTAG 管脚

管脚名称	I/O 类型	上拉/下拉	说明
tck	I	下拉	tck 是时钟信号，tck 与系统时钟独立。
tms	I	上拉	tms 是模式控制信号，与 tck 同步。
tdi	I	上拉	tdi 是串行输入数据信号，tdi 与 tck 同步。
tdo	O	——	tdo 是串行输出数据信号，tdo 与 tck 同步。
trst	I	上拉	trst 用于复位 JTAG，低电平有效，trst 与 tck 异步。不使用 JTAG 时下拉。

不使用 JTAG 时需确保 trst 管脚下拉，推荐接法如图 4-12-1 所示。

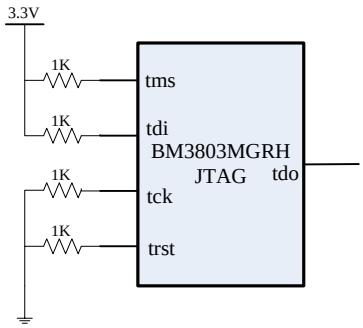


图 4-12-1 不使用 JTAG 时推荐的板级连接方式

tms 和 tdi 管脚通过 1K 上拉电阻连接到 3.3V 的电源，tck 和 trst 通过 1K 下拉电阻接地。

使用 JTAG 时，四个输入管脚中的 tms、tdi、tck 直接连接其外部 JTAG 输入信号，trst 管脚推荐采用如图 4-12-2 的接法。

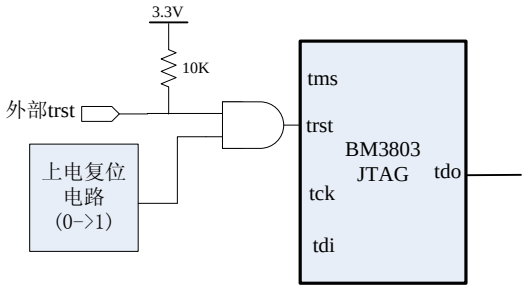


图 4-12-2 使用 JTAG 时推荐的 trst 管脚连接方式

图 4-12-2 中上电复位电路为板级上电复位电路，“外部 trst”为外部 JTAG 输入信号中的 trst，并通过 10K 电阻上拉，上电复位电路的输出和“外部 trst”作为与门的输入，与门的输出连接 BM3803FMGRH 的 trst 管脚。

五、 产品电特性

5.1 电特性

表 5-1-1 电特性

参数	符号	条件 (除另有规定外， -55℃≤T_A≤125℃， 1.65V≤V_{DDD}≤1.95V， 1.65V≤V_{DDA}≤1.95V， 3.0V≤V_{DDDIO}≤3.6V， 3.0V≤V_{DDAIO}≤3.6V)		分组	极限值		单位
					最小	最大	
输出高电平电压	V_{OH}	V_{DDD}=1.65V，V_{DDA}=1.65V，V_{DDDIO}=3.0V，V_{DDAIO}=3.0V，测所有适用的输出端。[a]	I_{OH}=-0.5 mA	A1 A2 A3	2.7	—	V
			I_{OH}= -4、-8、-12、-16 mA		2.6	—	
输出低电平电压	V_{OL}		I_{OL}=1.5 mA		—	0.3	V
			I_{OL}= 4、8、12、16 mA		—	0.4	
输入高电平电压	V_{IH}	测所有适用的输入端。			2.0	—	V
输入低电平电压	V_{IL}				—	0.8	V
输入高电平漏电流	I_{IH}	V_{DDD}= 1.95V，V_{DDA}=1.95V，V_{DDDIO}= 3.6V，V_{DDAIO}= 3.6V，V_I= 3.6V			—	1	μA
上拉输入高电平漏电流	I_{IHPU}				—	5	μA
下拉输入高电平漏电流	I_{IHPD}				20	200	μA
输入低电平漏电流	$\lvert I_{IL} \rvert$	V_{DDD}=1.95V，V_{DDA}=1.95V，V_{DDDIO}=3.6V，V_{DDAIO}= 3.6V，V_I= 0V			—	1	μA
上拉输入低电平漏电流	$\lvert I_{ILPU} \rvert$				20	200	μA
下拉输入低电平漏电流	$\lvert I_{ILPD} \rvert$				—	5	μA

表 5-1-1 （续）

参数	符号	条件 (除另有规定外， -55℃≤T_A≤125℃， 1.65V≤V_{DDD}≤1.95V， 1.65V≤V_{DDA}≤1.95V， 3.0V≤V_{DDDIO}≤3.6V， 3.0V≤V_{DDAIO}≤3.6V)		分组	极限值		单位
					最小	最大	
三态输出高电平漏电流	I_{OZH}	V_{DDD}=1.95V ， V_{DDA}=1.95V ，V_{DDDIO}=3.6V ， V_{DDAIO}= 3.6V ，V_O=3.6V		A1 A2 A3	—	1	μA
三态输出低电平漏电流	$\lvert I_{OZL} \rvert$	V_{DDD}=1.95V ， V_{DDA}=1.95V ，V_{DDDIO}=3.6V ， V_{DDAIO}= 3.6V ，V_O=0V			—	1	μA
待机电源电流	$I_{DDD(SB)}$	V_{DDD}=1.95V V_{DDA}=1.95V V_{DDDIO}=3.6V V_{DDAIO}=3.6V clk 和 pci_clk 时钟信号不工作，负载为空。	测 V_{DDD} 端		—	10	mA
			测 V_{DDDIO} 端		—	5	
输入电容 [b]	C_{IN}	T_A =25℃		A4	—	12	pF
输出电容 [b]	C_{IO}	T_A =25℃			—	12	pF
动态功耗	$I_{DD(dy)}$	f=100MHz，负载为空	测 V_{DDD} 端	A4	—	240	mA
			测 V_{DDA} 端	A5		10	
			测 V_{DDDIO} 端	A6		90	
功能测试频率	f	V_{DDD}=1.65V、1.8V、1.95V V_{DDA}=1.65V、1.8V、1.95V V_{DDDIO}=3.0V、3.3V、3.6V V_{DDAIO}=3.0V、3.3V、3.6V V_{IH} =V_{DDDIO} V_{IL} = 0V V_{OUT}= V_{DDDIO}/2 [C]		A7 A8A A8B	—	100	MHz

表 5-1-1 （续）

参数	符号	条件 (除另有规定外， -55℃≤T_A≤125℃， 1.65V≤V_{DDD}≤1.95V， 1.65V≤V_{DDA}≤1.95V， 3.0V≤V_{DDDIO}≤3.6V， 3.0V≤V_{DDAIO}≤3.6V)	分组	极限值		单位
				最小	最大	
Pll 旁路情况 Sdclk 输出延时	t_4	V_{DDD}＝1.65V， V_{DDA}＝1.65V， V_{DDDIO}＝3.0V， V_{DDAIO}＝3.0V， SKEW[1:0]=“00”[d]	A9 A10 A11	2	7	ns
PLL 建立时间	t_5			—	10^7	ns
address[27:0]输出延时	t_{10}			1.5	11	ns
data[31:0] and pardata[6:0] 写输出延时	t_{11}			2	13	ns
data[31:0] and pardata[6:0] 写输出保持时间	t_{14}			2	13	ns
Oen 输出延时	t_{15}			2	11	ns
writen 输出延时	t_{17}			1.5	11	ns
Romsn[1:0]输出延时	t_{18}			2	13	ns
Ramsn[4:0]输出延时	t_{19}			2	13	ns
Ramoen[4:0]输出延时	t_{20}			2	13	ns
rwen[3:0] 输出延时	t_{21}			2	13	ns
iosn 输出延时	t_{22}			2	13	ns
sdcasn 输出延时	t_{25}			3	9	ns
sdcsn[1:0] 输出延时	t_{26}			2	8.5	ns
sdrasn 输出延时	t_{27}			2	8.5	ns
sdwen 输出延时	t_{28}			2	8.5	ns
sddqm[3:0] 输出延时	t_{29}			2	8.5	ns
pio[15:0]输出延时	t_{31}			2.5	10	ns
pio[15:0]写保持时间	t_{34}			2.5	—	ns
Pci_ad_n[31:0] 和 pci_cbe_n[3:0] 输出延时	t_{110}			4	12	ns
Pci_frame_n, pci_par_n, pci_perr_n, pci_serr_n, pci_stop_n, 和 pci_devsel_n 输出延时	t_{113}			4	11	ns
pci_irdy_n 和 pci_trdy_n 输出延时	t_{114}			4	11	ns
pci_req_n 输出延时	t_{115}			4	12	ns

[a] V_{OH} 和 V_{OL} 参数的适用管脚的输出驱动分类列表见表 14。

[b] C_{IN}、C_{IO}仅在初始鉴定或设计工艺更改有影响时进行。抽取 3 只器件，1 只器件抽取 C_{IN}、C_{IO}全部引线，另外两只器件中每只各抽样 116（0）。这两只器件中 1 只 C_{IN} 必须全抽，1 只 C_{IO}必须全抽。

[c] 条件 V_{OUT}=V_{DDDIO}/2 适用于一般双向和输出管脚，不包括 OD 双向、OD 输出和三态输出管脚。

[d] 测试频率为 20MHz。

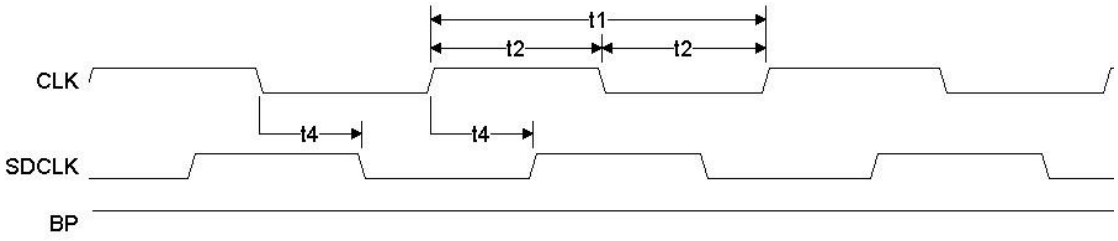


图 5-1-1　PLL 旁路情况下的时钟信号

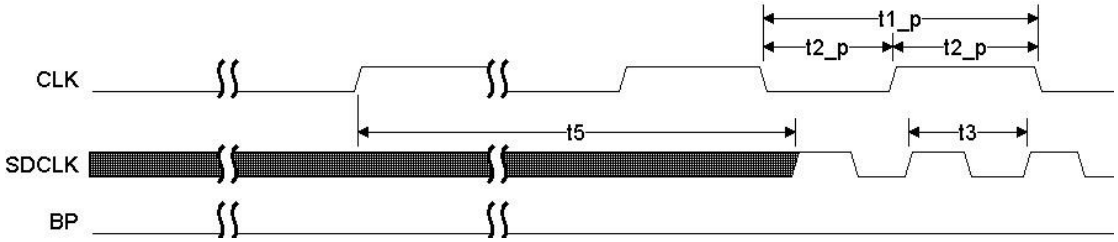


图 5-1-2　使用 PLL 情况下的时钟信号

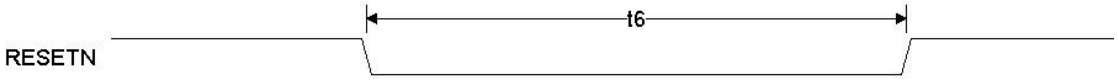


图 5-1-3 复位信号

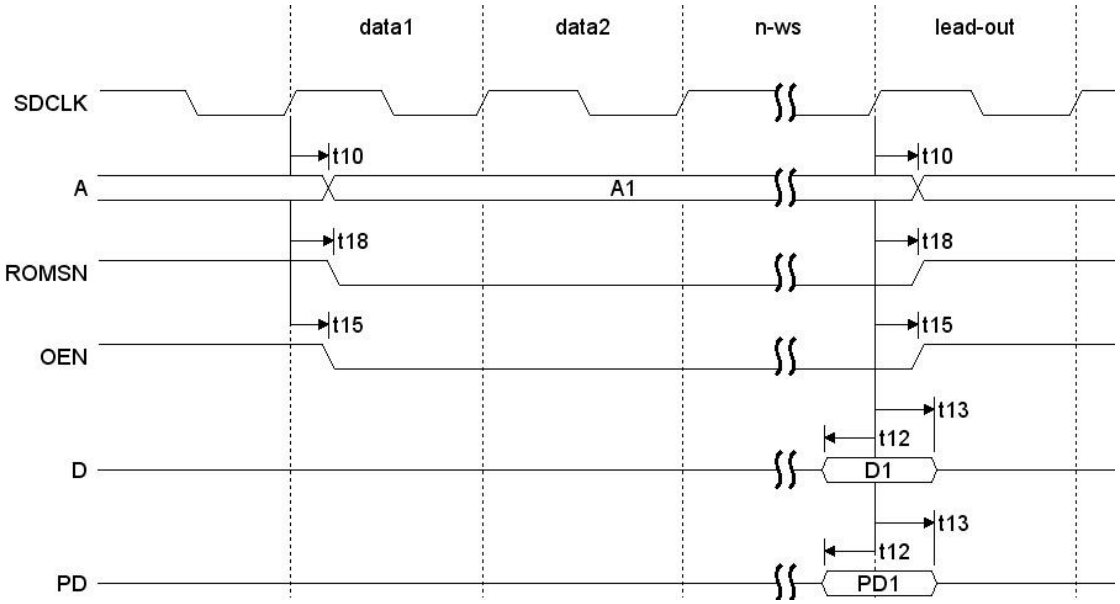


图 5-1-4　32 位 PROM 读时序

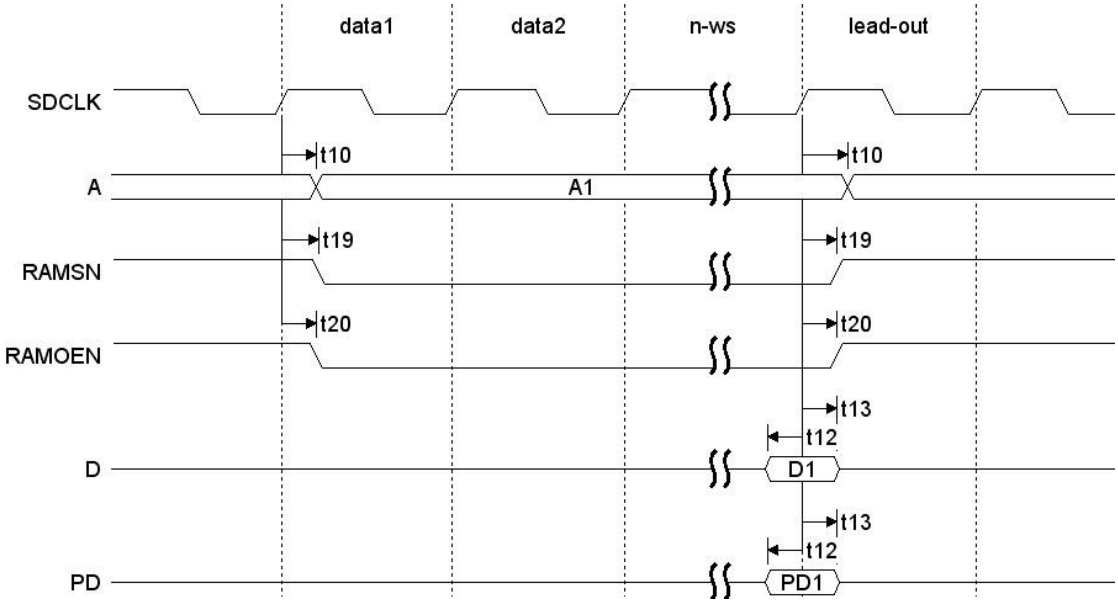


图 5-1-5　32 位 SRAM 读时序

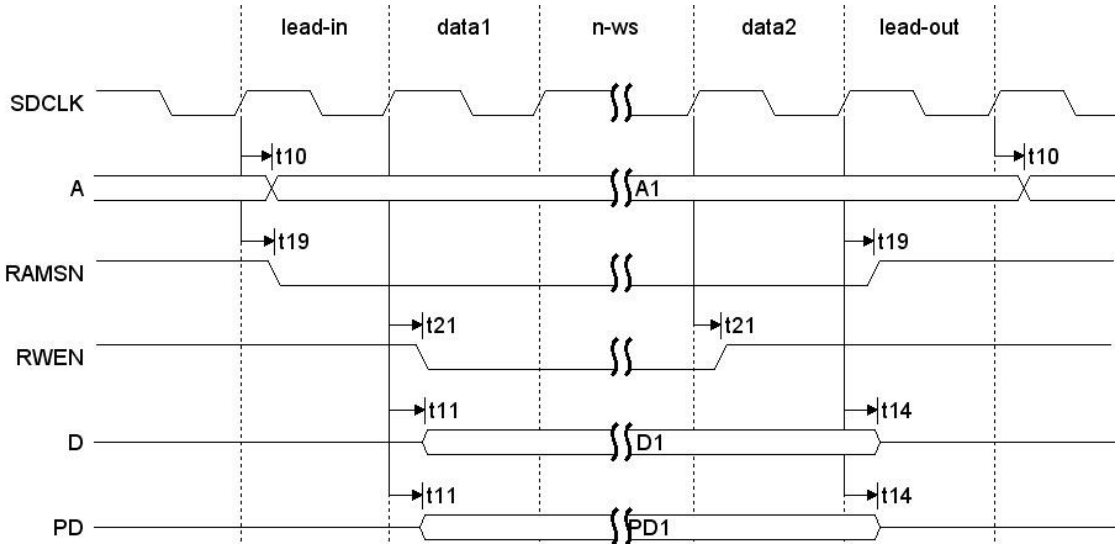


图 5-1-6　32 位 SRAM 写时序

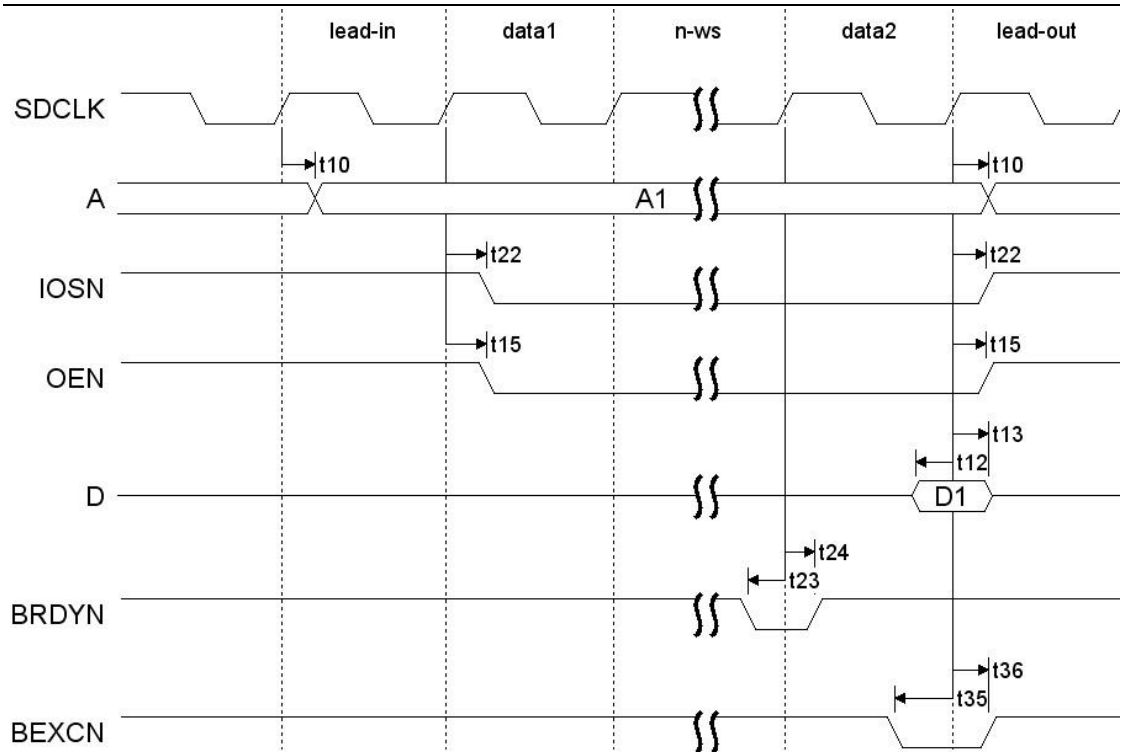


图 5-1-7　32 位 I/O 读时序

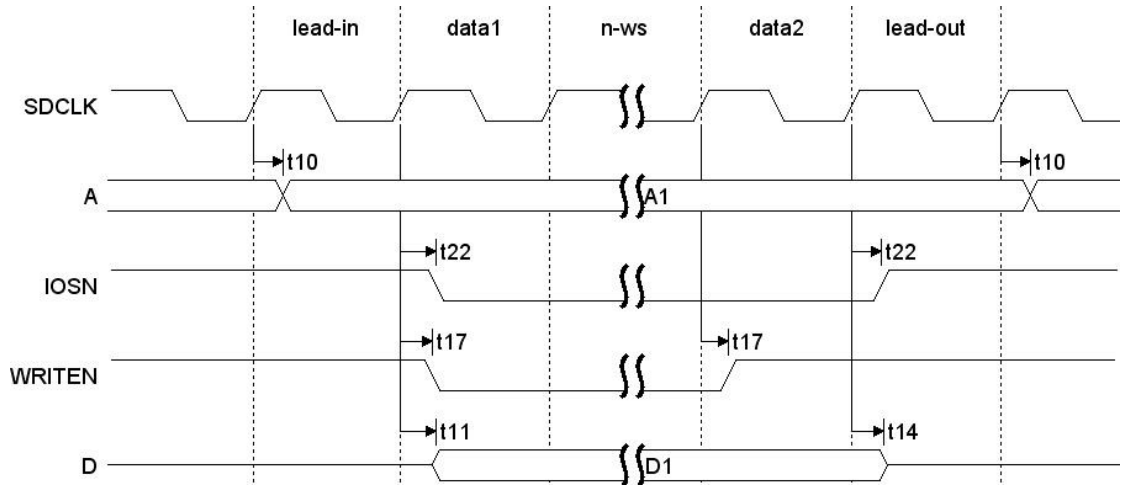


图 5-1-8　32 位 I/O 写时序

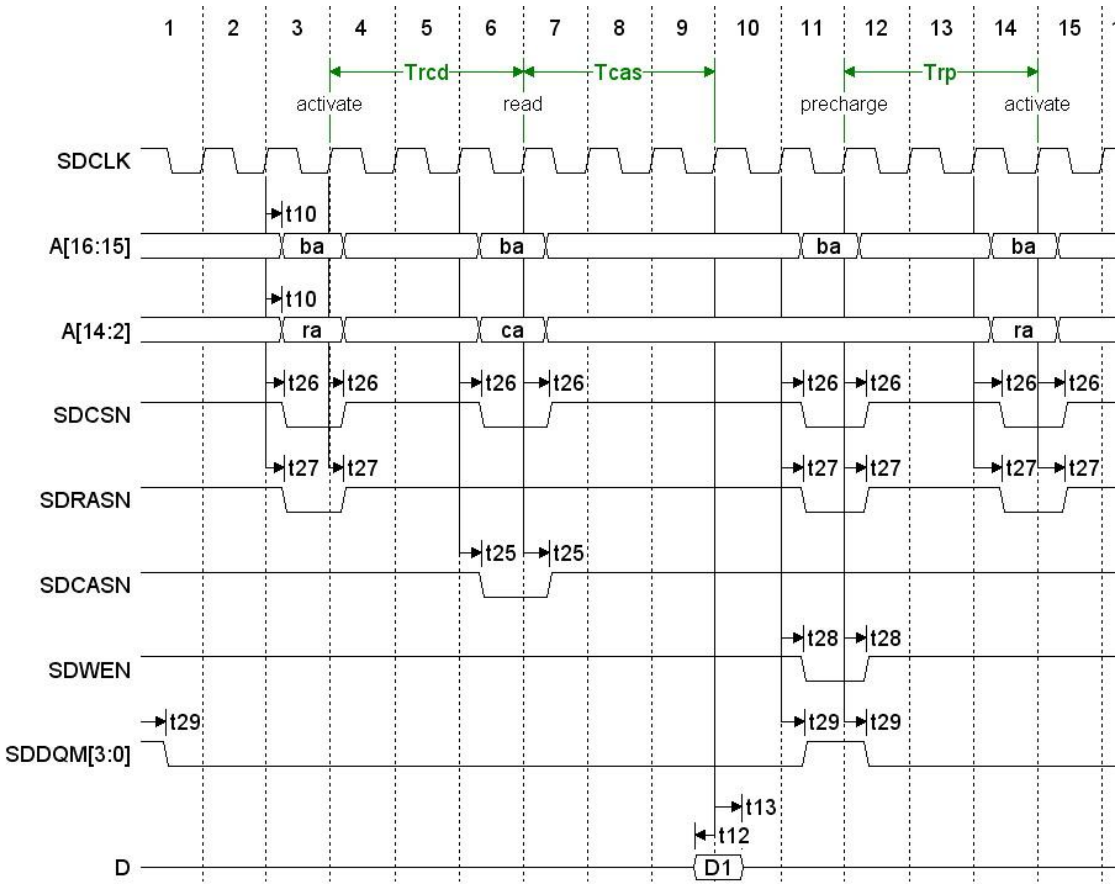


图 5-1-9　SDRAM 读时序

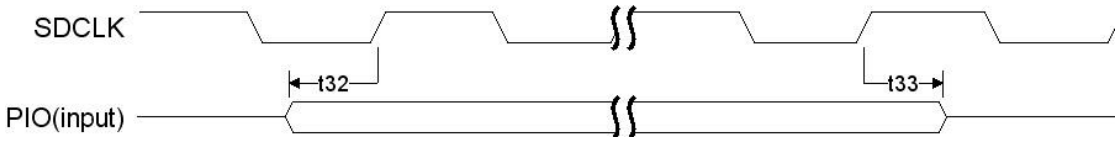


图 5-1-10　PIO 输入时序

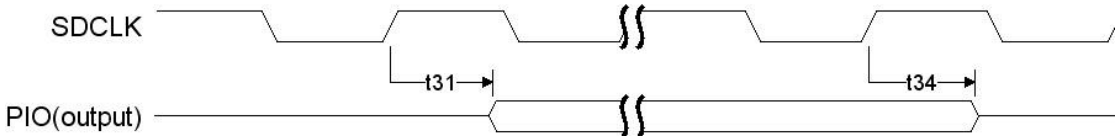


图 5-1-11　PIO 输出时序

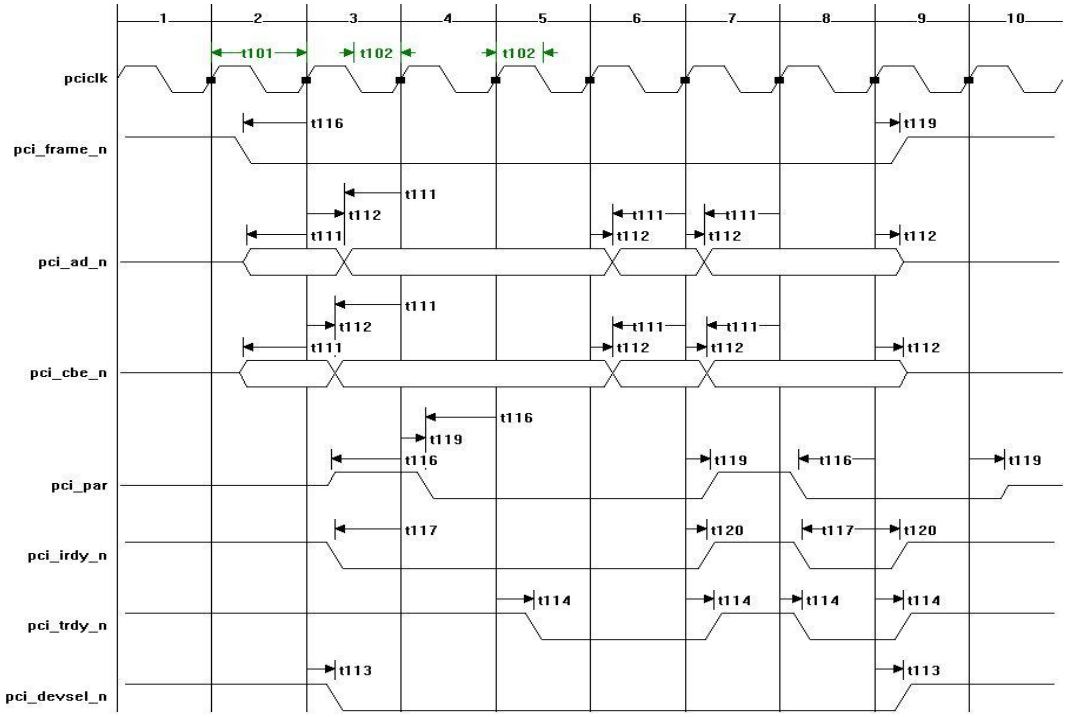


图 5-1-12　PCI 写时序

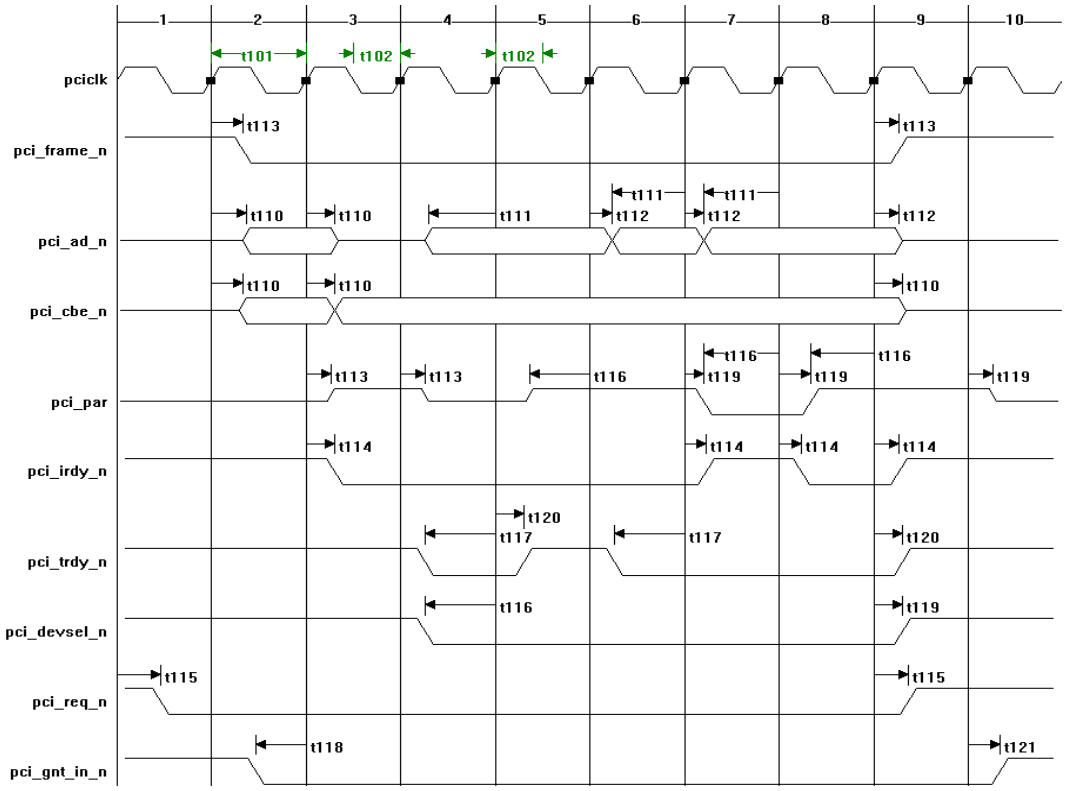


图 5-1-13　PCI 读时序

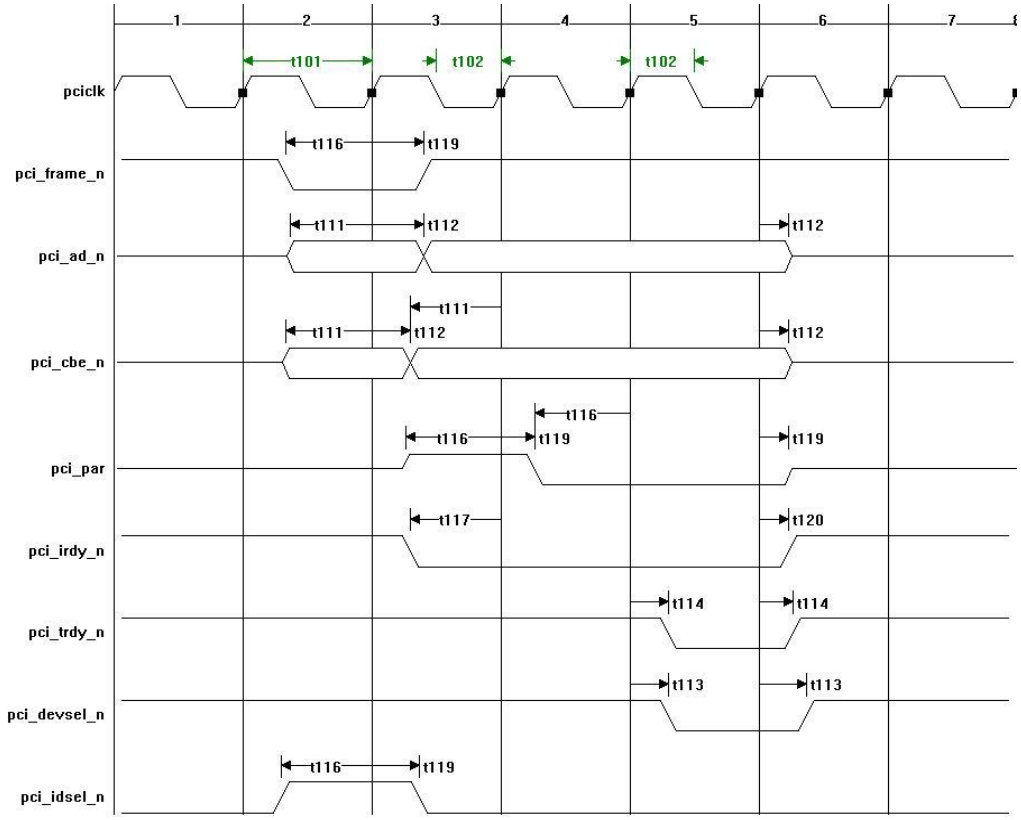


图 5-1-14 PCI 配置写时序

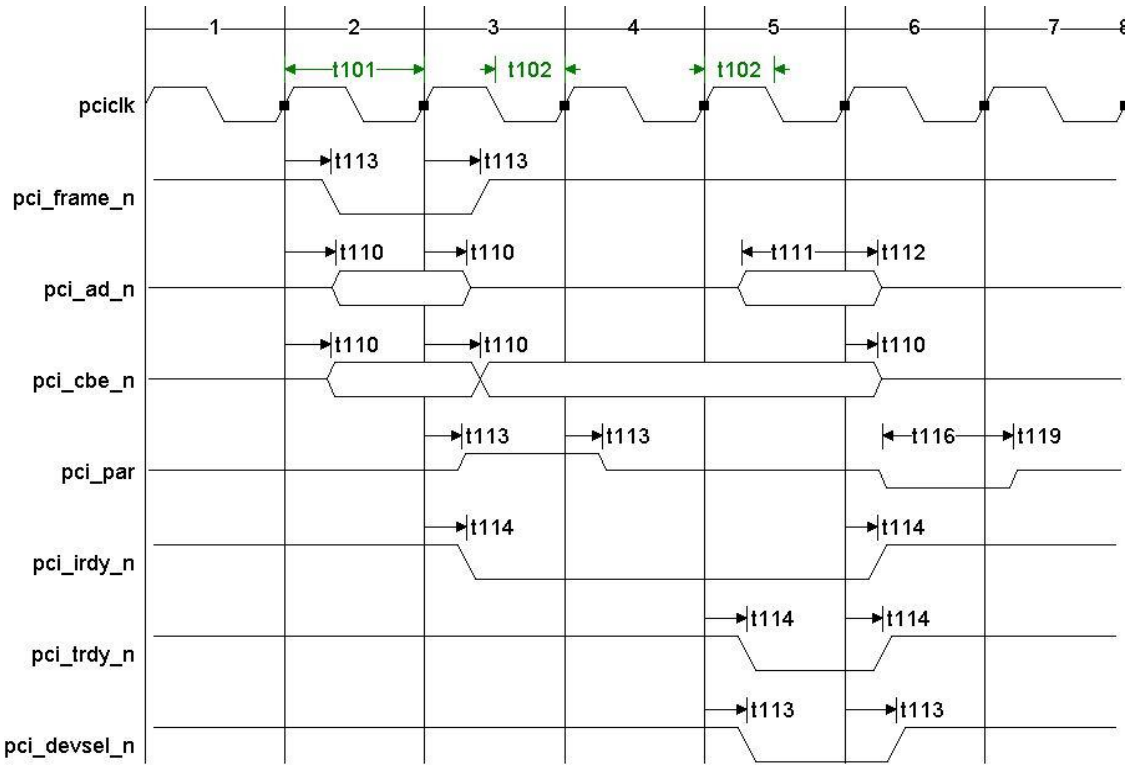


图 5-1-15　PCI 配置读时序

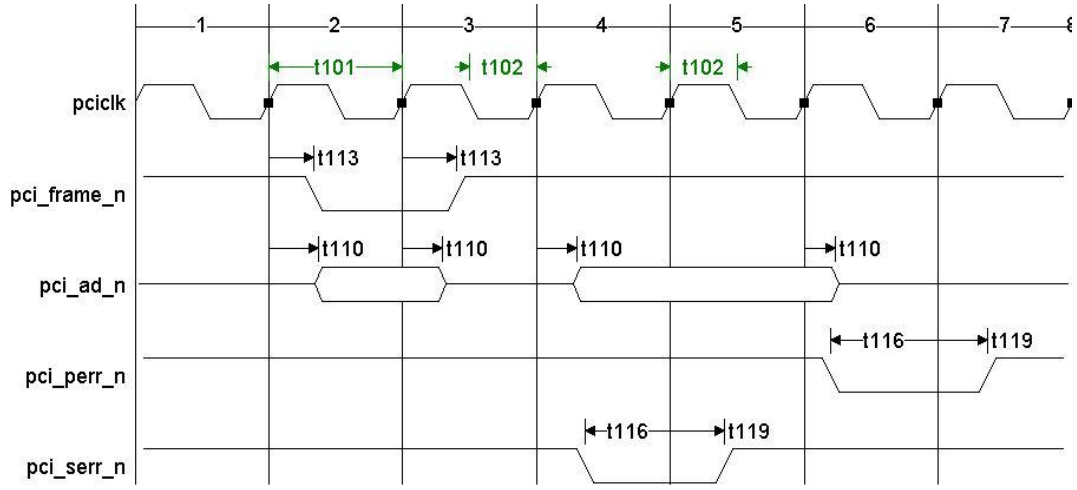


图 5-1-16　pci_perr_n 和 pci_serr_n 输入时序

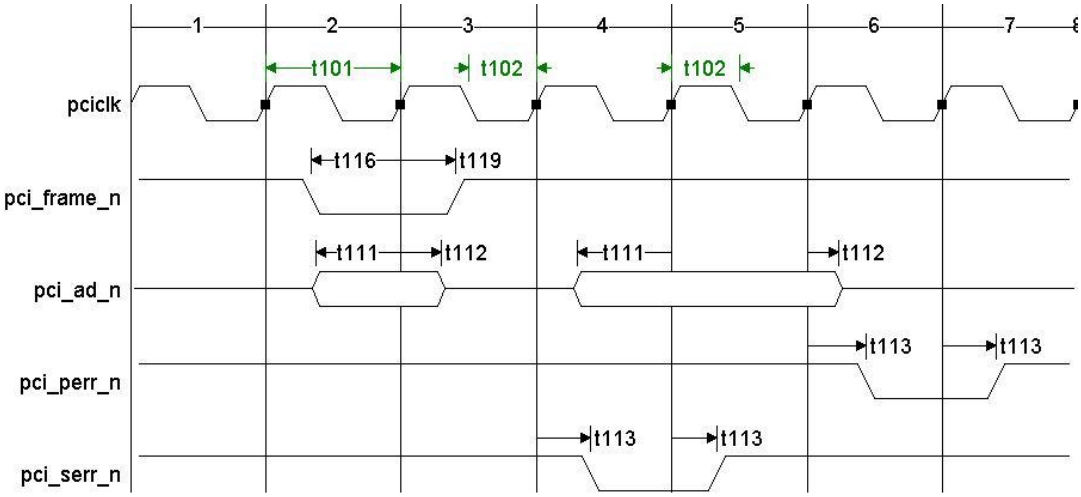


图 5-1-17　pci_perr_n 和 pci_serr_n 输出时序

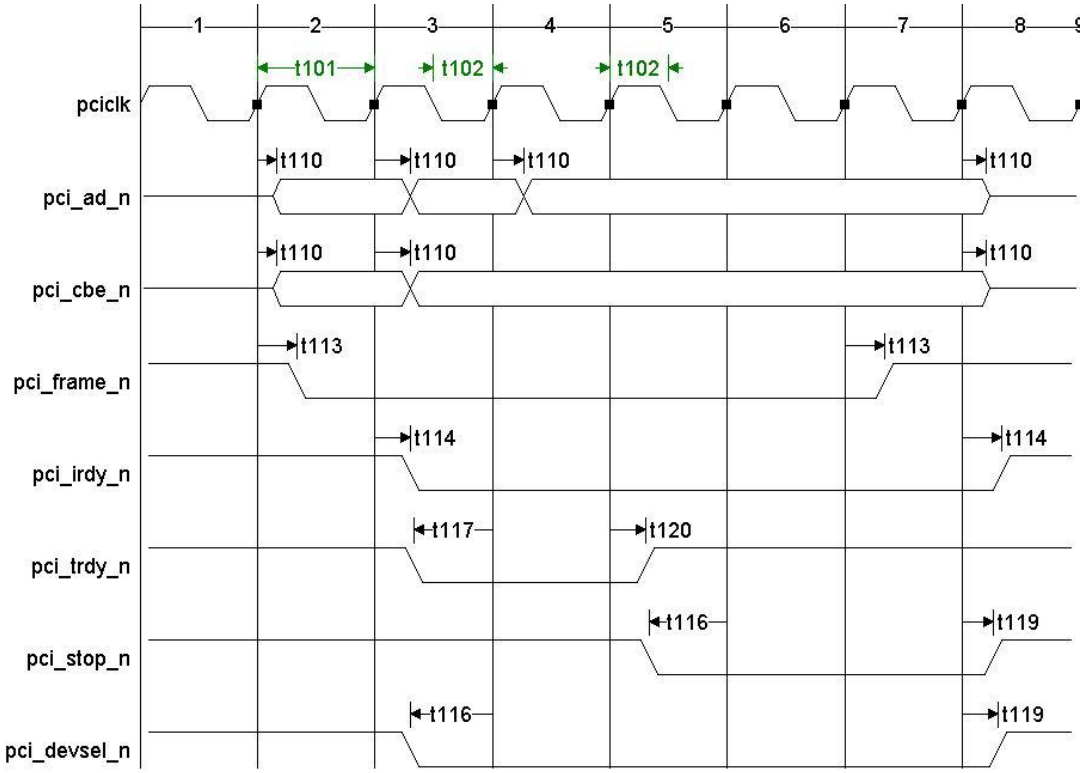


图 5-1-18　pci_stop_n 输入时序

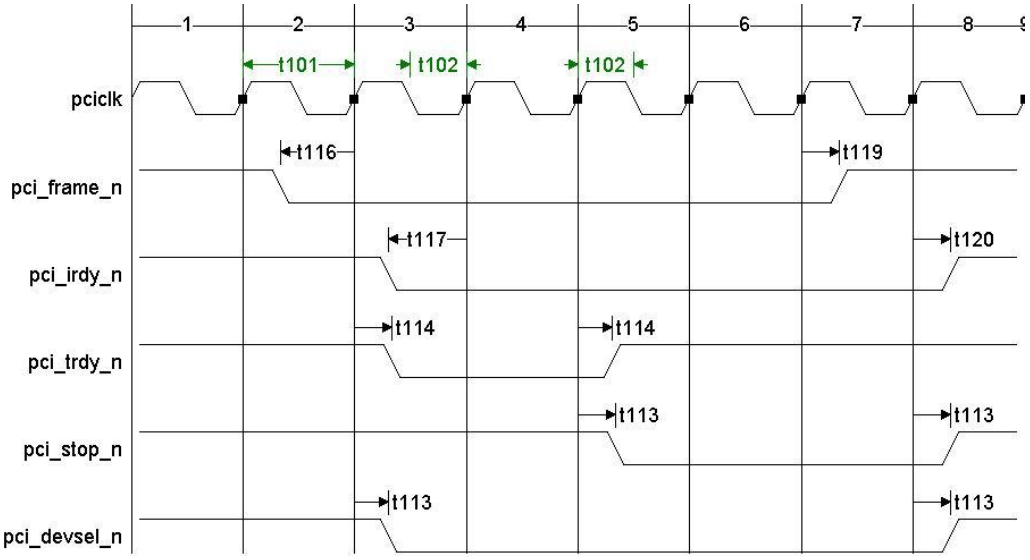


图 5-1-19　pci_stop_n 输出时序

六、 典型应用

选择 BM3803FMGRH 的典型功能，设计完成开发板，主要包括如下功能：

- 外部提供+5V 的电压输入；
- BM3803FMGRH 主频 100MHz；
- 板载 4MB FLASH；
- 板载 2MB SRAM；
- 板载 64MB SDRAM；
- 提供 3 个通用异步串行接口，1 个 DSU 调试端口；
- 提供 GPIO 扩展接口；
- 提供 1 路符合 PCI 2.3 规范的 PCI 扩展接口。

6.1 开发板说明

BM3803FMGRH 开发板外围硬件框图如图 6-1-1 所示。

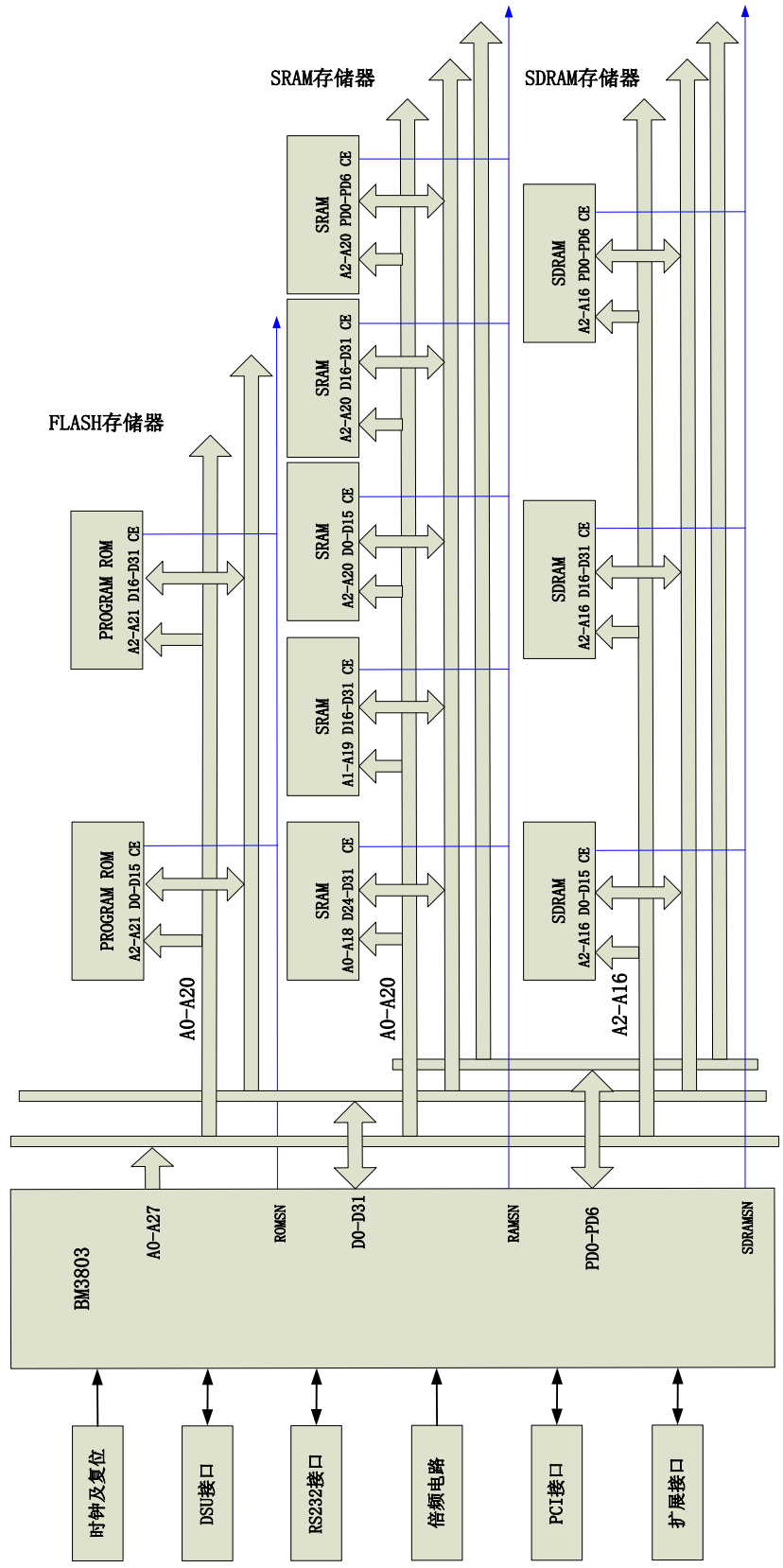


图 6-1-1 开发板外围硬件框图

6.2 开发板参考电路

6.2.1 电源模块

开发板采用 5V 电源供电，通过电源芯片分别输出模拟 3.3V（VCC33），数字 3.3V（VDD33），模拟 1.8V（VCC18），数字 1.8V（VDD18）电源电压。原理图如图 6-2-1 所示。

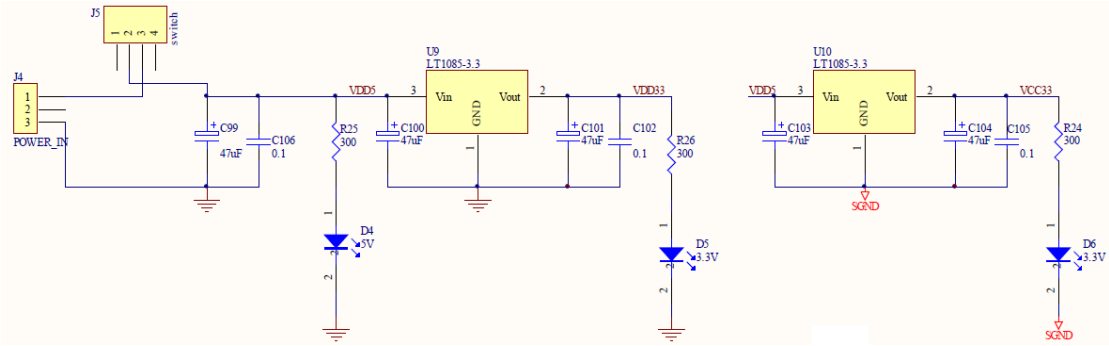


图 6-2-1-a 3.3V 电源原理图

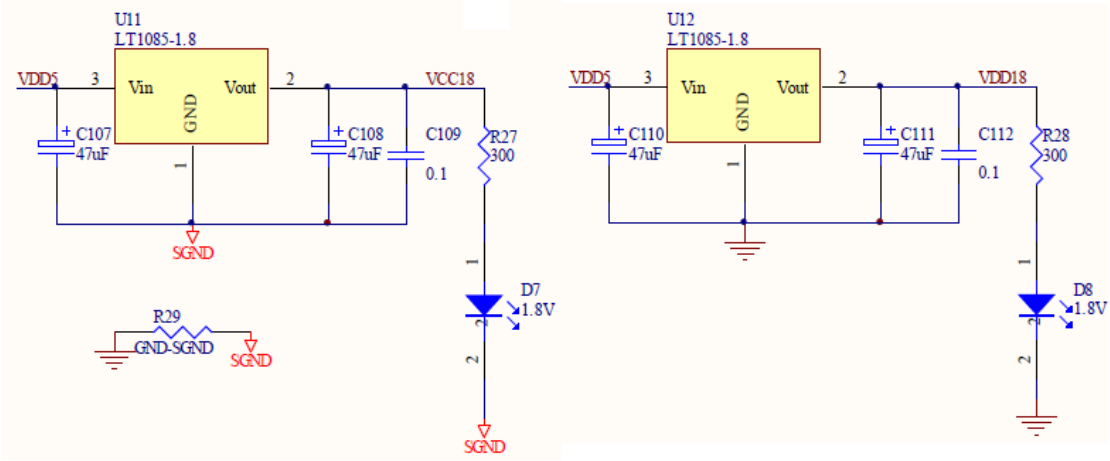


图 6-2-1-b 1.8V 电源原理图

图 6-2-1 电源原理图

6.2.2 时钟复位电路

开发板采用了 10MHz 晶振作为外部时钟输入，由于 BM3803FMGRH 内部集成了 pll，系统时钟由外部时钟输入和倍频系数 M 共同决定（其中 M 的范围在 1-15 之间，使用拨码开关

可以进行倍频选择)。复位电路采用 MAX708R 上电提供 200ms 低电平。原理图如图 6-2-2 所示。

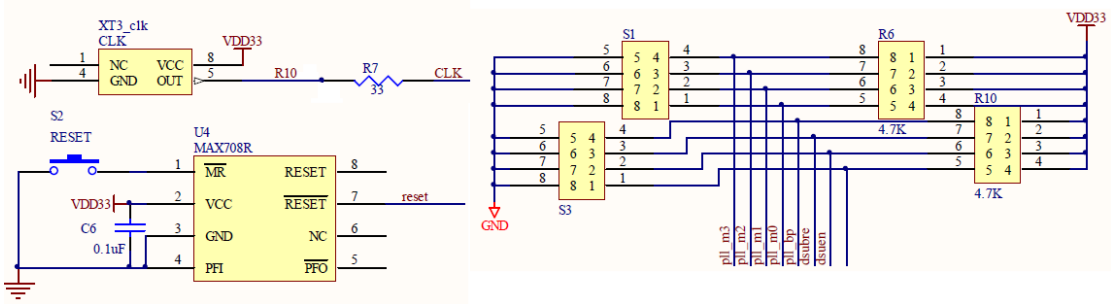


图 6-2-2 时钟复位原理图

时钟电路是否进行时钟倍频操作，取决于 pll 的工作模式，具体模式如下：

在 BYPASS=1 工作状态下（即 PLLBP=H 的条件下），倍频系数 M 无效：

$$f_{cpuclk} = f_{clk} = 10\text{MHz}$$

在 BYPASS=0 工作状态下（即 PLLBP=L 的条件下），倍频系数 M 有效：

$$f_{cpuclk} = f_{clk} * M$$

其中拨码开关用于倍频系数 M 的选择，如表 6-2-1 所示。

表 6-2-1 倍频系数

倍频系数 M	M[3]	M[2]	M[1]	M[0]
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0

11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

注：“1”代表连通 H，“0”代表连通 L；倍频系数 M 推荐 10，（额定频率 100MHz，不建议超频使用）

6.2.3 DSU 串口电路

DSU 串口电路采用 MAX3232 芯片，原理图如图 6-2-3 所示。

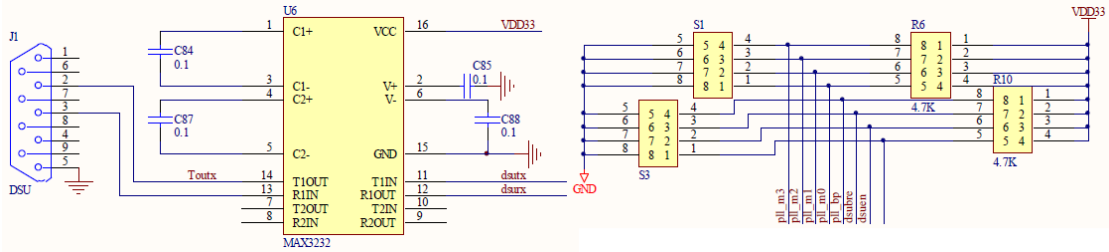


图 6-2-3 DSU 串口原理图

拨码开关的组态决定了 BM3803FMGRH 的两种工作状态，分别是：

◆ **运行模式**

将拨码开关的 dsuen 和 dsubre 都置于低电平，复位后进入运行模式，从地址 0x0 开始执行程序。

◆ **调试模式**

将拨码开关的 dsuen 和 dsubre 均置于高电平，复位后从地址 0x0 开始执行 4 条指令后进入调试模式。

6.2.4 测试管脚

BM3803FMGRH 开发板对测试管脚进行了处理，在工作模式（运行模式和调试模式）下不再使用，如表 6-2-2 和图 6-2-4 所示。

表 6-2-2 测试管脚处理方式说明

管脚	符号	类型	处理说明
T2	tck	I, PD	1KΩ电阻下拉
T4	tdi	I, PU	1KΩ电阻上拉
R1	tdo	O	悬空
P2	tms	I, PU	1KΩ电阻上拉
N1	trst	I, PU	1KΩ电阻下拉
C35	bist_mode	I, PD	1KΩ 电阻下拉
G31	scan_enable	I, PD	1KΩ 电阻下拉
E1	scan_mode	I, PD	1KΩ 电阻下拉

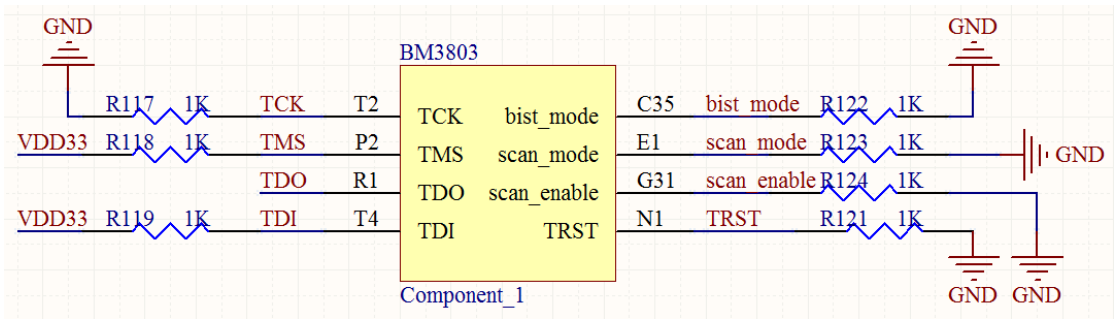


图 6-2-4 测试管脚处理方式

6.2.5 FLASH

PROM 的接口分为两个 Bank ，每个 Bank 最大空间为 256M。板载的 FLASH 采用 AM29LV160D(1M*16)FLASH。U200 和 U201 两片组成 32 位宽不带 EDAC 的 FLASH 系统，U202 组成 8 位宽的带 EDAC 功能的 FLASH 系统。当选用 ROM_CE0 时，FLASH 系统地址空间从 0x0 开始；当选用 ROM_CE1 时，FLASH 系统地址空间从 0x10000000 开始。原理图如图 6-2-5 所示。

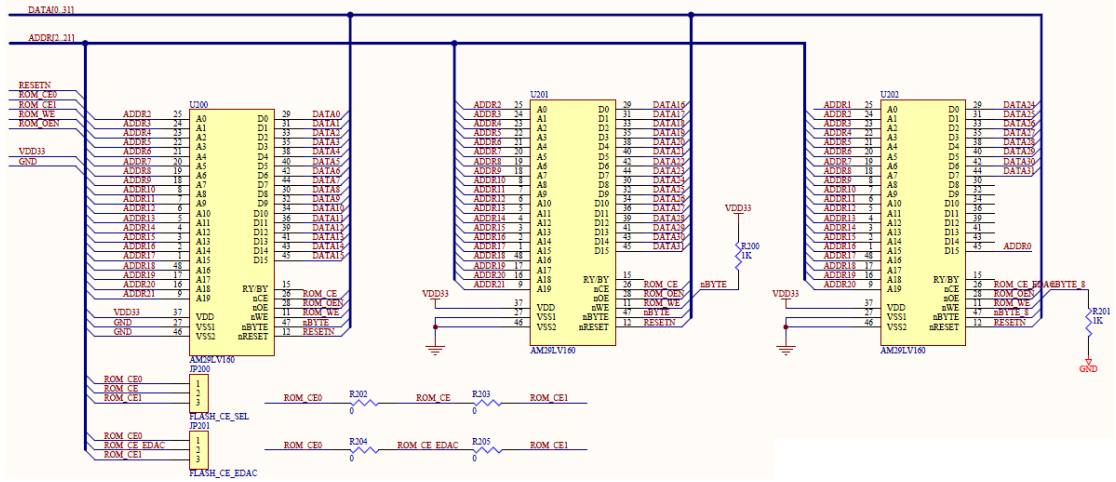


图 6-2-5 FLASH 系统原理图

6.2.6 SRAM

板载的三片型号为 CY7C1041BV33（256K*16）的 SRAM 组成 32 位宽带 EDAC 的 SRAM 系统， U13 和 U14 存放数据位，地址空间为：0x4000-0000 到 0x400F-FFFF，U16 存放校验位。SRAM 系统不支持写选通，只支持读改写，如图 6-2-6 所示。

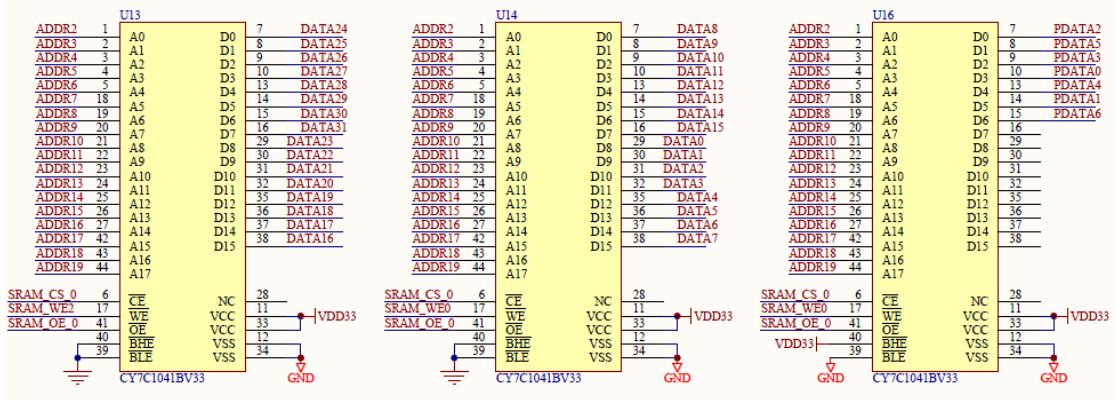


图 6-2-6 SRAM 系统原理图

6.2.7 SDRAM

SDRAM 分为两个 Bank ，每个 Bank 最大空间为 256M。板载的 SDRAM 采用三片型号为 HY57V561620(4Banks*4M*16)的 SDRAM 组成 32 位宽带 EDAC 的系统，U300 和 U301 存放数据位，地址空间为：0x6000-0000 到 0x63FF-FFFF，U302 存放校验位。SDRAM 系统原理图如图 6-2-7 所示。

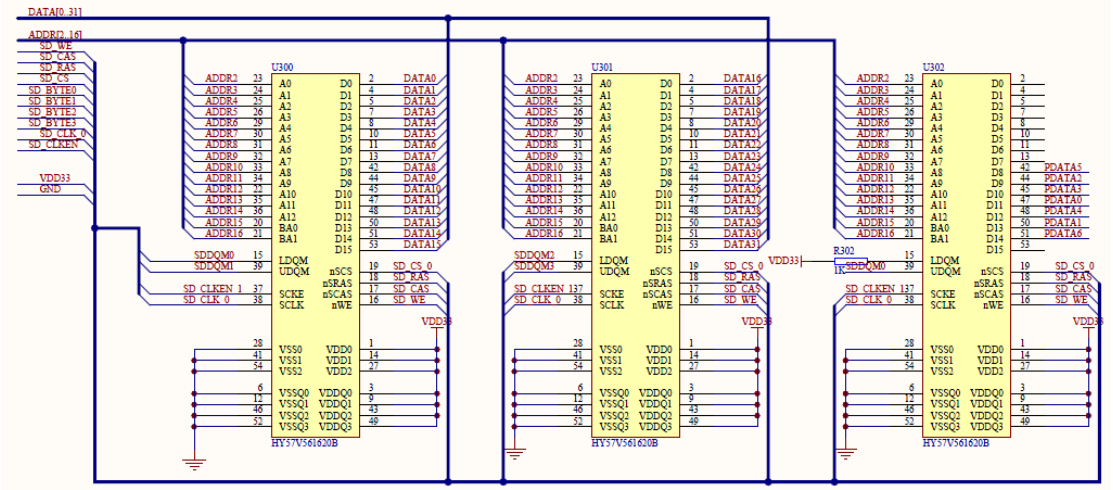


图 6-2-7 SDRAM 系统原理图

七、 应用注意事项

7.1 产品应用说明

1. 在产品使用过程中如有任何不明确之处，请及时反馈。
2. 当系统复位时，需先复位外围器件，然后复位本电路。
3. 电源上电顺序严格按照章节 7.2，禁止出现内核电源先于 I/O 电源上电的情况。
4. 利用外部复位引脚 resetn 进行复位时，需要在时钟有效后保持复位信号至少 10 个时钟周期。
5. 为保证 pll 稳定工作，外部时钟 clk 的选择必须同时满足以下三个条件：输入频率范围 2MHz-30MHz；占空比范围 45%-55%；倍频后的时钟范围 40MHz-100MHz。
6. 测试管脚 bist_mode、scan_mode、scan_enable 必须通过 1KΩ 电阻下拉。
7. 运行模式时，必须将 dsuen 和 dsubre 管脚置为低电平，dsurx 置为高电平。
8. PROM、SRAM、I/O 读访问时序中“lead_out”周期中的地址值，不保持上一个时钟周期中的地址值，而是上一个时钟周期中处理器内部总线地址值。
9. 当 SDRAM 和 SRAM 都配置使能，并且 SDRAM 的 bank 的 size 配置为 512MB 时，应当用处理器的管脚 sdcsn[1]连接 SDRAM 存储器的片选管脚。
10. SDRAM 存储器访问，在使能 EDAC 功能时，不支持内部 AHB 总线的 burst 传输模式。因此，当访问 SDRAM 存储器，并使能 EDAC 功能时，要避免内部 AHB 总线的 burst 传输模式：

    （1）关闭 Cache 控制寄存器（0x8000-0014）的 burst 取指。

    （2）避免 PCI 操作使用 burst 传输模式。
11. 辅助状态寄存器 ASR16 上电初始值是不定值，使用时必须先设置 ASR16 不使能造错，再读/写 Regfile 寄存器。严格禁止 ASR16 使能造错时，读/写 Regfile 寄存器。
12. 如果写保护要使用 segment 模式，即 BP 配置为“0”，只能使能一个写保护寄存器，另外一个写保护寄存器，不能同时使用 segment 模式。
13. LDDx 指令在 AHB 总线上发起的传输是 burst 传输模式。
14. PCI 模块操作不支持字节、半字操作，只允许字操作，操作地址必须字对齐。

15. power-down 模式启动前，必须先清除所有外部中断源，确保写 power-down 寄存器时无外部中断产生。

16. 电路采用 CMOS 工艺生产，使用时需注意静电防护。工作台应有可靠接地，并配有防静电设备，操作人员要佩戴防静电护腕。对电路进行测量前需先将所使用的测量探针接地，插拔电路时应使用专用工具进行操作，避免用手直接接触电路，以免造成污染或破坏。

17. 波峰焊工艺注意事项

（1）推荐采用合金成分为 37%Pb63%Sn 的焊料。

（2）器件在拾取和安装过程中应避免因外力造成的引线变形、损伤或表面镀层脱落。

（3）波峰焊前，应对器件进行预热，避免焊接过程中应力过大造成器件损坏。波峰焊时，器件实际达到的峰值温度建议不超过 220℃，锡炉温度在 230℃到 250℃之间。波峰焊温度曲线如下图 7-1-1 所示。

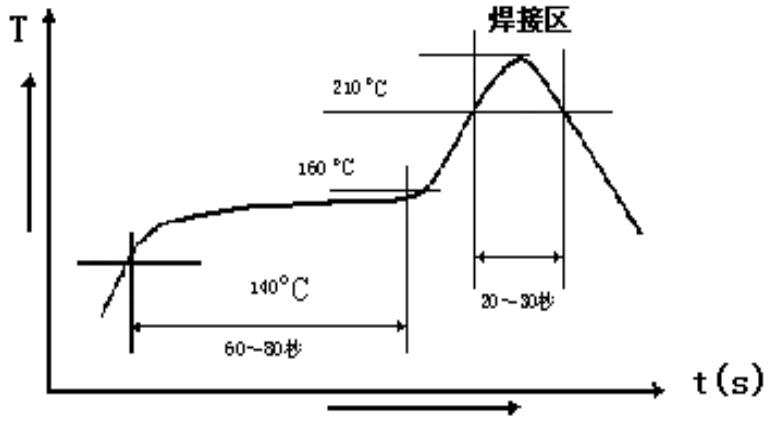


图 7-1-1 波峰焊温度曲线

18. 手工焊接注意事项

（1）器件在安装过程中应避免因外力造成引线发生变形损伤或表面镀层脱落。

（2）焊接时烙铁温度范围：240～300℃。

（3）单根引线焊接时间：5～10 秒，烙铁头温度越高焊接时间要越短。

7.2 对电源的要求

BM3803FMGRH 共有四路电源，分别是数字 I/O 电源、模拟 I/O 电源、数字内核电源、模拟内核电源。BM3803FMGRH 对电源的要求和推荐电路如下。

工作电压要求

数字 I/O 和模拟 I/O 工作电压是 3.30V±0.30V，数字内核和模拟内核工作电压是 1.80V±0.15V。

上电过冲电压要求

上电时数字 I/O 和模拟 I/O 电压过冲不得超过 4V，数字内核和模拟内核电压过冲不得超过 2V。

上电顺序要求

为保证 BM3803FMGRH 正常、安全启动，四路电源必须严格按照正确顺序上电，具体要求是：

1）数字 I/O 与模拟 I/O 同时上电（合称 I/O 电源）；

2）数字内核与模拟内核同时上电（合称内核电源）；

3）I/O 电源先于内核电源上电或者 I/O 电源与内核电源同时上电。

⚠注意 禁止上电过程中内核电压高于 I/O 电压超过 0.5V。

在电源设计时：上电顺序可设计成 I/O 电源先于内核电源上电，或 I/O 电源和内核电源同时上电 2 种情况：

一种是 I/O 电源先于内核电源上电，要求两电源之间的时间间隔（t）小于 2ms。如图 7-2-1 所示。

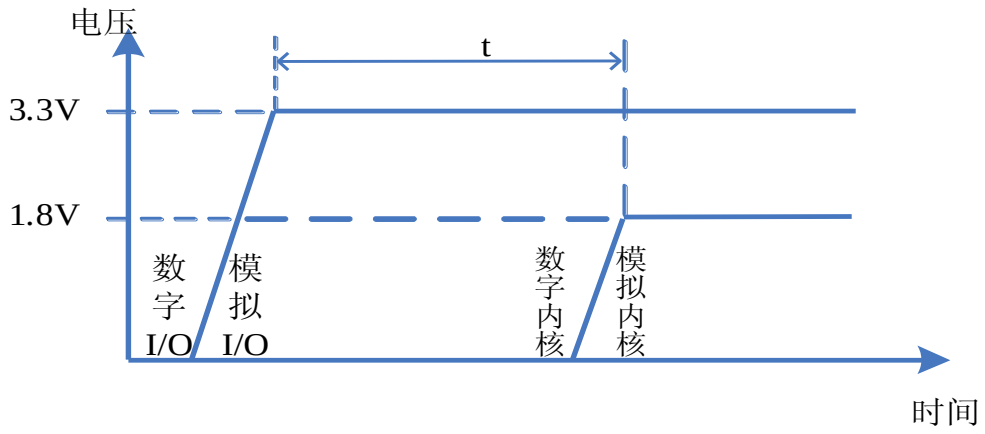


图 7-2-1 BM3803FMGRH 四路电源上电顺序

另一种是 I/O 电源和内核电源同时上电，要求上电过程中内核电源高于 I/O 电源的电压差（ΔV）不得超过 0.5V。如图 7-2-2 所示。

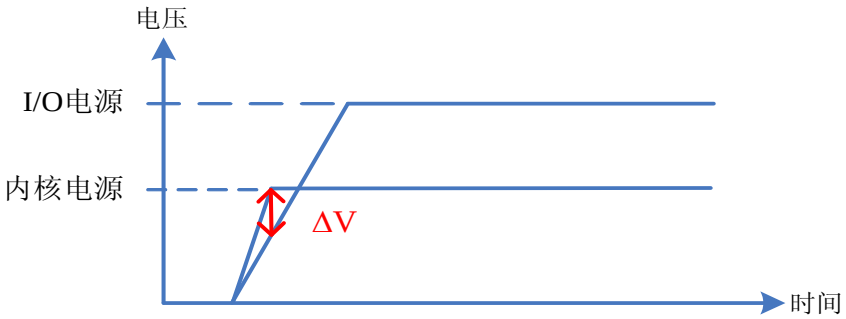


图 7-2-2 I/O 电源与内核电源同时上电

推荐设计

BM3803FMGRH 电源电路的推荐设计如下图所示，采用 DC-DC 电路提供 I/O 电源和 LDO 输入，LDO 电路提供内核电源。电路框图如图 7-2-3。

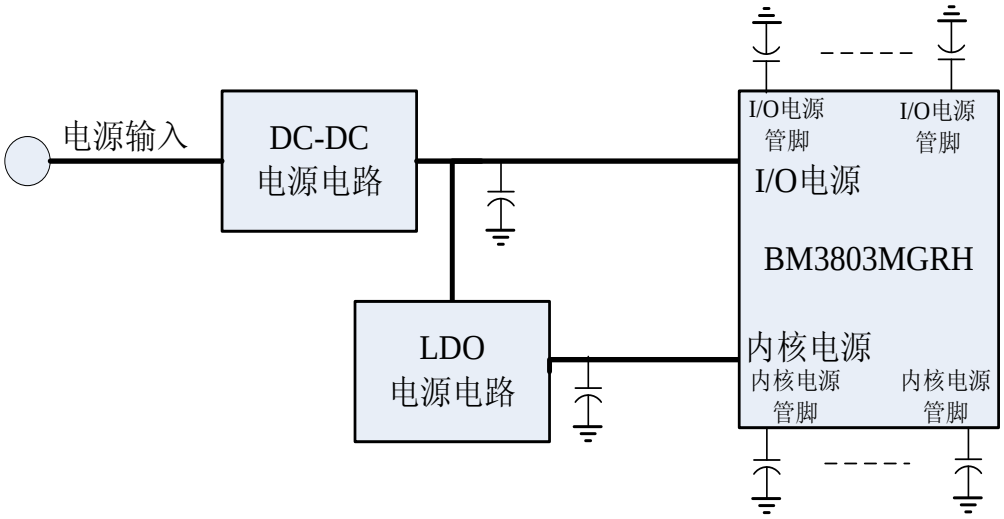


图 7-2-3 推荐电源电路框图

DC-DC 电源输出电流能力为 1.5A，LDO 输出电流能力为 1.0A。PCB 电路板上 I/O 电源和内核电源输出端放置合适的滤波电容，保证电源电路纹波符合电源要求；在每个 I/O 电源管脚和内核电源管脚尽可能近的位置上放置合适旁路电容（推荐 910pF 陶瓷电容），滤除开关噪声。

7.3 常见问题软件规避

7.3.1 浮点 IEEE-754 异常处理方法

7.3.1.1 问题概述

BM3803FMGRH 的 IEEE_754_excption 有两种处理方式：一种方式是在 FSR 中屏蔽 IEEE_754_excption 的五种异常，指令在发生异常时不响应 trap，直接置 FSR 中 *CEXC* 字段中相应的标志位，指令继续执行；另一种方式是在 FSR 中使能 IEEE_754_excption 的五种异常，指令在发生异常时响应 trap，指令被打断，不写执行结果，不改变 FSR 中 *CEXC* 字段中相应的标志位。

7.3.1.2 规避措施

为保证系统可靠应用，针对 BM3803FMGRH 浮点异常可选择的处理方法有两种：

1）在 FSR 中屏蔽 IEEE_754_excption 的五种异常，通过软件预先对浮点运算的数据进行相关判断，保证浮点运算不发生 IEEE_754_excption 异常，以除零异常为例，先对除数进行判断，如果为零就不进行浮点除法。

2）在 FSR 中使能 IEEE_754_excption 的五种异常的情况下，若需要置 FSR 中 *CEXC* 相应的标志位，可在 trap 处理程序中采用相应软件完成对 FSR 中 *CEXC* 字段的置位。

7.3.2 特定指令序列双精度浮点运算问题解决办法

7.3.2.1 问题概述

应用程序采用如下指令序列：

ld [%l1],%f2

nop（中断服务程序）

ld [%l2],%f3

faddd %f4,%f2,%f4

在 BM3803FMGRH 中运行时发现双浮点指令 faddd 计算结果不正确。根据分析，双精度浮点运算不正确的特定指令序列包括以下六种情况：

情况（1）：

LD [%rn], %fx+1

FPOPd(1) %fx, %fy, %fx

情况（2）：

LD [%rn], %fx+1

FPOPd(1) %fy, %fx, %fx

情况（3）：

LD [%rn], %fx+1

FPOPd(1) %fx, %fy, %fy

情况（4）：

LD [%rn], %fx+1

FPOPd(1) %fy, %fx, %fy

情况（5）：

LD [%rn], %fx+1

FPOPd(2) %fx, %fx, %fx

情况（6）：

LD [%rn], %fx+1

FSQRTd %fx, %fx

注：1. FPOPd 是 FADDd、FSUBd、FMULd、FDIVd 指令之一。

2. FPOPd 是 FADDd、FMULd 指令之一。

7.3.2.2 规避措施

为了避免该问题给系统使用带来影响，采用软件办法避免出问题的特定指令序列来解决该问题。软件修改的原理是通过调整指令顺序的方法，保证最终的程序中不出现存在该问题的所有指令序列。

根据编程方式的不同，解决该问题的方式如下：

采用高级语言编程时，通过二进制检查工具对编译链接后的文件（含操作系统程序）进行检查，找出可能出错的指令序列，将双精度浮点指令前的两条 ld 指令交换顺序（避免该两条 ld 指令被动态执行的代码打断）。如果仅有一条 ld 指令与双精度指令连续（或者两条

ld 指令之间已经存在静态代码），满足上述六种情况的，在双精度指令前选择合适的指令与这条 ld 指令交换。

（1）原出错指令流是：

```
ld [%l1],%f2
nop（中断服务程序）
ld [%l2],%f3
faddd %f4,%f2,%f4
```

改变指令流为：

```
ld [%l2],%f3
nop（中断服务程序）
ld [%l1],%f2
faddd %f4,%f2,%f4
```

采用汇编语言编程时，在两条 ld 指令与双精度浮点指令之间加一条 nop 指令。

（2）原出错指令流是：

```
ld [%l1],%f2
nop（中断服务程序）
ld [%l2],%f3
faddd %f4,%f2,%f4
```

改变指令流为：

```
ld [%l1],%f2
nop（中断服务程序）
ld [%l2],%f3
nop
faddd %f4,%f2,%f4
```

（3）原出错指令流是：

```
sethi  %hi(0x40001800), %g1
st  %o1, [ %fp + -16 ]
```

```
or  %g1, 0x230, %g1
sll  %o2, 3, %o2
ldd  [ %g1 + %o2 ], %f10
ld  [ %fp + -16 ], %f9
faddd  %f10, %f8, %f8
```

改变指令流为：

```
sethi  %hi(0x40001800), %g1
st  %o1, [ %fp + -16 ]
or  %g1, 0x230, %g1
sll  %o2, 3, %o2
ld  [ %fp + -16 ], %f9
ldd  [ %g1 + %o2 ], %f10
faddd  %f10, %f8, %f8
```

7.3.3 浮点异常陷阱返回地址问题解决办法

7.3.3.1 问题概述

运行双精度浮点指令并引发 fp_exception 陷阱时，当前窗口的%l1 和%l2 中都会保存 nPC 值，这一行为不符合 SPARC V8 手册中的定义：发生陷阱时，PC 和 nPC 分别保存到当前窗口的%l1 和%l2 中。

SPARC V8 处理器的陷阱发生在两种情况下：

1)陷阱发生在指令执行过程中,这种情况一般是因为正在执行的指令本身发生了错误，比如浮点除零。

2）陷阱发生在两条指令之间，这种情况一般是因为发生了外部中断，如串口收到新数据。

当 SPARC V8 处理器发生陷阱时，会自动把当前的 PC 和 nPC 值保存在当前窗口的%l1 和%l2 中，这两个值在从陷阱返回时用到。

从陷阱返回有两种方法：

方法 1、返回引发陷阱的指令，即重新执行这一条指令，此时程序写为：

```
jmp  %l1  ! return old PC
rett  %l2  ! return old nPC
```

方法 2、返回引发陷阱指令的下一条指令，此时程序写为：

```
jmp  %l2     ! return old nPC
rett  %l2 + 4  ! return old nPC + 4
```

选择何种返回方法依应用而定，与陷阱发生的原因不存在对应关系。

由于%l1 和%l2 都保存了 nPC，采用陷阱返回方法 1 时，不能实现重新执行引发陷阱的指令，此时，会连续执行两次引发陷阱指令后的那条指令，影响系统程序；若采用陷阱返回方法 2，不影响系统程序。

7.3.3.2 规避措施

为了避免该问题给系统使用带来影响，采用软件办法来解决该问题。

采用陷阱返回方法 1 时，在 fp_exception 陷阱处理程序中增加一条减法指令，通过%l2 保存的 nPC 减 4 生成 PC 存入%l1，即：

引发陷阱的程序：

```
0x4000100：fdivd %f0, %f2, %f4  （发生除零陷阱）
0x4000104：nop
```

陷阱处理程序：

```
sub %l2, 4, %l1
```

此时，%l1 = 0x4000100；%l2 = 0x4000104，与正确情况相符。

之后可以采用任意陷阱返回方法进行处理。

7.3.4 Ticc 指令清除中断问题解决办法

7.3.4.1 问题概述

运行指令“Ticc 0x1n”或者“Ticc 0x5n”时（n=[1:F]），若指令产生软件陷阱0x1n或者0x5n，即**陷阱基址寄存器TBR**的*TT*字段是0x9n或者0xDn，BM3803FMGRH会错误地清除中断0xn，即**陷阱基址寄存器TBR**的*TT*字段是0x1n。

Ticc 指令，即整数条件码陷阱指令，包括 15 条汇编指令：TA、TNE、TE、TG、TLE、TGE、TL、TGU、TLEU、TCC、TCS、TPOS、TNEG、TVC、TVS。

Ticc指令产生软件陷阱0x1n或者0x5n时，产生影响具体如下：

（1）对**中断请求寄存器**（地址0x8000-0094）和**中断强制寄存器**（地址0x8000-0098）：

若中断请求位0xn是1，并且中断强制位0xn是0，则清零中断请求位0xn；

若中断请求位0xn是0，并且中断强制位0xn是1，则清零中断强制位0xn；

若中断请求位0xn是1，并且中断强制位0xn是1，则清零中断强制位0xn。

（2）对**Cache控制寄存器**（地址0x8000-0014）：

若数据Cache使能（*DCS*字段是“11”），并且数据Cache中断冻结位使能（*DF*字段是“1”），则冻结数据Cache（改变*DCS*字段为“01”）；

若指令Cache使能（*ICS*字段是“11”），并且指令Cache中断冻结位使能（*IF*字段为“1”），则冻结指令Cache（改变*ICS*字段为“01”）。

7.3.4.2 规避措施

为了避免该问题给系统使用带来影响，用户应避免使用“Ticc 0x1n”和“Ticc 0x5n”指令（n=[1:F]），改为使用指令 Ticc 0xMn（M=0、2、3、4、6、7；n=[1:F]）。

7.4 编程说明

7.4.1 BM3803FMGRH 汇编编程注意事项

BM3803FMGRH 是基于 SPARC V8 体系结构的 32 位 RISC 嵌入式处理器，指令兼容《The SPARC Architecture Manual Version 8》。SPARC V8 共定义了 246 条指令，分为五种基本指令类型，分别为数据传输指令、算术/逻辑操作指令、控制传输指令、混合指令和浮点操作指令。

BM3803FMGRH 根据具体设计特点，实现了 SPARC V8 定义的 179 条指令，其中数据传输指令 21 条、算术/逻辑操作指令 35 条、控制传输指令 83 条、混合指令 17 条、浮点操作指令 23 条。如表 7-4-1 所示。

BM3803FMGRH 未实现的指令有 67 条，包括协处理器指令 41 条、MMU 相关指令 1 条、四精度浮点指令 14 条、单精度浮点相乘得双精度浮点指令 1 条、浮点队列指令 1 条、非 32 位交换空间加载/存储指令 9 条。如表 7-4-2 所示。

表 7-4-1 BM3803FMGRH 实现指令

指令类型	实现指令
数据传输指令	LDSB、LDSH、LDUB、LDUH、LD、LDD、STB、STH、ST、STD、LDSTUB、SWAP、LDF、LDDF、LDFSR、STF、STDF、STFSR、LDA、STA、SWAPA
算术/逻辑操作指令	AND、ANDcc、ANDN、ANDNcc、OR、ORcc、ORN、ORNcc、XOR、XORcc、XNOR、XNORcc、SLL、SRL、SRA、ADD、ADDcc、ADDX、ADDXcc、TADDcc、TADDccTV、SUB、SUBcc、SUBX、SUBXcc、TSUBcc、TSUBccTV、UMUL、SMUL、UMULcc、SMULcc、UDIV、SDIV、UDIVcc、SDIVcc
控制传输指令	BA、BN、BNE、BE、BG、BLE、BGE、BL、BGU、BLEU、BCC、BCS、BPOS、BNEG、BVC、BVS、BA,a、BN,a、BNE,a、BE,a、BG,a、BLE,a、BGE,a、BL,a、BGU,a、BLEU,a、BCC,a、BCS,a、BPOS,a、BNEG,a、BVC,a、BVS,a、FBA、FBN、FBU、FBG、FBUG、FBL、FBUL、FBLG、FBNE、FBE、FBUE、FBGE、FBUGE、FBLE、FBULE、FBO、FBA,a、FBN,a、FBU,a、FBG,a、FBUG,a、FBL,a、FBUL,a、FBLG,a、FBNE,a、FBE,a、FBUE,a、FBGE,a、FBUGE,a、FBLE,a、FBULE,a、FBO,a、CALL、JMPL、RETT、TA、TN、TNE、TE、TG、TLE、TGE、TL、TGU、TLEU、TCC、TCS、TPOS、TNEG、TVC、TVS
混合指令	RDY、RDASR、RDPSR、RDWIM、RDTBR、WRY、WRASR、WRPSR、WRWIM、WRTBR、FLUSH、SETHI、NOP、UNIMP、MULScc、SAVE、RESTORE
浮点操作指令	FMOVs、FNEGs、FABSs、FSQRTs、FSQRTd、

	FADDs、FADDd、FSUBs、FSUBd、FMULs、FMULd、FDIVs、FDIVd、FiTOs、FdTOs、FiTOd、FsTOd、FsTOi、FdTOi、FCMPs、FCMPd、FCMPEs、FCMPEd

表 7-4-2 BM3803FMGRH 未实现指令

指令类型	未实现指令
协处理器指令	CBA、CBN、CB3、CB2、CB23、CB1、CB13、CB12、CB123、CB0、CB03、CB02、CB023、CB01、CB013、CB012、CBA,a、CBN,a、CB3,a、CB2,a、CB23,a、CB1,a、CB13,a、CB12,a、CB123,a、CB0,a、CB03,a、CB02,a、CB023,a、CB01,a、CB013,a、CB012,a、LDC、LDDC、LDCSR、STC、STDC、STCSR、STDCQ、CPop1、CPop2
MMU 相关指令	STBAR
四精度浮点指令	FSQRTq、FADDq、FSUBq、FMULq、FDIVq、FdMULq、FqTOs、FqTOd、FiTOq、FsTOq、FdTOq、FqTOi、FCMPq、FCMPEq
单精度浮点相乘得双精度浮点指令	FsMULd
浮点队列指令	STDFQ
非 32 位交换空间存储/加载指令	LDSBA、LDSHA、LDUBA、LDUHA、LDDA、STBA、STHA、STDA、LDSTUBA

上述 BM3803FMGRH 实现的指令中，依据《The SPARC Architecture Manual Version 8》手册，对少数指令结合编译器功能进行设计简化，剩余指令遵守手册定义。

与 BM3803FMGRH 具体实现相关的指令分为两类，一类是《The SPARC Architecture Manual Version 8》手册中明确规定的与具体实现相关的指令，另一类是 BM3803FMGRH 根据实现难度，结合编译器功能进行设计简化的指令。下面对这些设计简化的指令进行详细说明。

7.4.1.1《The SPARC Architecture Manual Version 8》手册中明确规定的实现相关指令

（a）写状态寄存器指令是延迟指令，若用户软件在这些延迟指令后三指令周期内访问或隐含访问这些状态寄存器的指令，须（增加 3 条指令如 nop）满足该寄存器前一指令写入三指令后生效的要求，确保后续指令访问到的是已更新的状态寄存器。

一般的指令写法是：

```
mov reg _or_imm, %X (1)
nop
nop
nop
```

或者：

```
wr %g0, reg _or_imm, %X (1)
nop
nop
nop
```

注：(1) %X 是%psr、%wim、%tbr 寄存器其中之一。

（b）半字的访问必须在 2-byte 边界上对齐（aligned）；字的访问（包括取指令），必须在 4-byte 边界上对齐；双字的访问必须在 8-byte 边界上对齐。一个没有正确对齐的地址会让 load 或 store 指令产生一个 mem_address_not_aligned 陷阱。

7.4.1.2 BM3803FMGRH 中设计简化的指令

（a）mulscc 指令前禁止使用特定 ld 指令（ld 目的寄存器与 mulscc 的 rs1 寄存器相同）。

非法指令序列：

```
ld [address], regrs1
mulscc regrs1, regrs2_or_imm, regrd
```

若需要使用上述指令，必须在两指令之间插入 2 个或 2 个以上 nop 指令。

正确的指令序列是：

```
ld [address], regrs1
nop
……
nop
mulscc regrs1, regrs2_or_imm, regrd
```

（b）当除数是-1 时，被除数不能为负数最小值（-2^31）。如下指令序列：

```
div %o0, %o1, %o0
或 sdivcc %o0, %o1, %o0
```

若%o1 = -1，则%o0 不能赋值为-2^31。

除法运算中，首先对被除数和除数进行判断，若除数是-1，且被除数是负数最小值，则直接对结果赋值 0x7fffffff。

（c）jmpl 指令的操作数必须 4 字节对齐。

如下指令所示：

```
jmpl  regrs1+regrs2_or_imm
```

jmpl 的跳转地址由 regrs1+ regrs2 或 regrs1+imm 得出，必须保证 regrs1、regrs2、imm 的值是 4 的倍数。

7.4.2 BM3803FMGRH 软件示例

电路软件示例是针对用户的普通应用，根据电路特征所给出的应用代码片段，描述了主要功能模块的使用方法和操作手段，作为电路用户手册的补充，可作为基本操作的参考。

BM3803FMGRH 为核心构建的单板计算机系统，软件开发应根据 SPARC V8 手册、BM3803FMGRH 电路用户手册要求进行；在实际应用中，可根据需要修改使用本示例中的代码片段。

7.4.2.1 片上寄存器访问

片上寄存器是片上模块（包括外部存储器控制、中断控制、I/O、定时器、串口等）的控制寄存器的集合，其在片上地址空间中的编址从 0x80000000 开始。

片上寄存器的访问，类似于对统一编址的存储器的访问，如向 Cache 控制寄存器地址

0x80000014 写十六进制数据 0x81000f 打开 Cache 功能模块(缓存、指令 burst、数据 snoop)，c 语言程序的访问方式有多种：

直接使用地址

```
*(unsigned int *)0x80000014 = 0x81000f;
```

使用宏引用来访问片上寄存器

```
#define CACHE_CFG 0x80000014
*(unsigned int *)CACHE_CFG = 0x81000f;
```

或

```
#define CACHE_CFG *(volatile unsigned int *)0x80000014
CACHE_CFG = 0x81000f;
```

使用结构体指针来访问

对片上寄存器（全部或部分）定义相应的数据结构（结构体），

```
struct lregs{
    volatile unsigned int memcfg1;
    volatile unsigned int memcfg2;
    volatile unsigned int memcfg3;
    volatile unsigned int ahbfailaddr;
    volatile unsigned int ahbstatus;
    volatile unsigned int cachectrl;    /*offset=0x14*/
    …       /*…其它…*/
};
```

定义指向结构体的指针 regs（指向 0x80000000 开始的寄存器区）

```
struct lregs *regs = (struct lregs*)0x80000000;
```

如 Cache 打开可以用：

```
regs->cachectrl = 0x81000f;
```

将变量定义为 volatile 类型，是说这个变量可能会被意想不到语句改变；这样，编译器就不会去假设这个变量的值。精确地说就是，优化器在用到这个变量时必须每次都小心地

重新读取这个变量的值，而不是使用保存在寄存器里的备份，这样可以避免错误地引用寄存器的值，这在读取的时候更可靠一些。

7.4.2.2 外部存储器控制器

外部存储器配置寄存器，用来定义外部存储器控制器在控制各种不同类型外部存储器（包括 PROM、SRAM）时控制的参数（设备是否可用、总线宽度、读写等待周期、时序规定、存储器大小）等。

在实际配置时，PROM 控制参数在系统上电时硬件配置应完全满足对 PROM 空间 BANK 0 的访问，设置或修改对任一区域的控制器参数最好在访问此区域之前进行；且不应对别的区域的访问造成不良影响：如修改 IO 时影响了 PROM。

例如配置 SRAM 存储器控制器（bank 大小为 2M 字节、16 位、读写等待为 5 个周期，读改写使能），程序如下：

```
#define MEM_CFG2 *(volatile unsigned int *)0x80000004
MEM_CFG2     =    0x00029055; /*2MB、16 位、5 等待周期，读改写*/
```

7.4.2.3 Cache 控制

BM3803FMGRH 有 4 路指令 Cache（共 32KB），2 路数据 Cache（共 16KB），都采用了多路组相联（multi-set-Caches）技术。打开 Cache 缓存功能及指令 Cache 的突发取指功能可以提高程序性能。

指令 Cache 没有 snoop 功能，程序中修改指令内容应该增加 flush 指令以确保外部存储器和 Cache 的一致性。

汇编指令 flush

C 语言语句： asm(“flush”);

在 flush 之后，等待 256 个时钟周期，可达到存储体和 Cache 的一致。使用插入 nop 指令的方法，或者执行一段循环等待指令，期间避免访问被修改的相关代码即可。

7.4.2.4 中断和陷阱

BM3803FMGRH 支持中断和陷阱两种异常，陷阱是由于特定指令相关的硬件引起，在引起

异常产生的指令运行期间发生；BM3803FMGRH 的外部事件可以产生中断类型的异常，中断的产生不与任何特定的指令相关，并在指令执行期间发生。

当陷阱发生时，与之唯一对应的识别码写入 TBR 的 8 位 *TT* 字段，处理器自动保存 PC 到%l1，nPC 到%l2，程序以 TBR 的值为目的地址进行跳转到陷阱向量表，然后跳转到陷阱处理程序。由于 TBR 的低四位为 0，因此陷阱向量表的每一入口包含陷阱处理程序的前 4 条指令（16 字节），陷阱向量表共定义了 256 个陷阱，占用 4KB 地址。陷阱向量表定义如下：

```
sethi %hi(_trap_table), %g1; /*陷阱向量表首地址赋于 TBR*/
mov   %g1, %tbr;
nop; nop; nop;
/*绑定陷阱向量表*/
#define TRAP(H)    mov %psr,%l0; sethi %hi(H), %l4; jmp %l4+%lo(H); nop;
#define TRAP_ENTRY(H)   mov %psr,%l0; sethi %hi(H), %l4; jmp %l4+%lo(H);
mov %tbr, %l3;

TRAP(_window_overflow);         ! 05 window_overflow
…
TRAP_ENTRY(_ISR_Handler)          ! tt=0x11 level 1
…
/*陷阱处理程序*/
_window_overflow:
…
jmp  %l1                    ! Re-execute save.
rett %l2
```

当发生浮点异常陷阱（tt=0x08）时，nPC 保存在寄存器%l2 中，对此寄存器中保存的数值减四可以获得引起浮点异常陷阱指令的 PC 值。同时，当发生 IEEE-754 型浮点异常时，由于异常打断了正常的处理过程，浮点异常状态位 *CEXC* 字段不更新；若希望获得引起浮点异常陷阱的具体原因，可关闭%fsr 中 *TEM* 字段并重新执行引起陷阱的指令，从而更新 *CEXC* 字

段，得到具体原因。

BM3803FMGRH 可以处理中断；*IPEND* 显示了当前等待处理的中断状态；写**中断强制寄存器**可以人为触发中断，而写**中断清除寄存器**可以人为清除中断 *IPEND*；BM3803FMGRH 响应中断时相应中断标识会被自动清除。

7.4.2.5 定时器

BM3803FMGRH 具有定时器，定时器共用一个预分频器。预分频器每个时钟周期减一，当产生下溢时，由预分频重载寄存器自动重新装载，同时为定时器产生一个计数 TICK 信号，所以有效的分频值等于预分频重载寄存器值加 1。定时器计数值在每个 TICK 信号到来时减一，当产生下溢时，产生相应的中断，然后由各自的重载寄存器自动重新装载，有效的 TICK 等于重载寄存器值加 1。所以每次中断时间间隔（以定时器 1 为例，且两个初始值都等于重载值）为：

（预分频重载值+1）*（定时器重载值+1）/ 处理器频率 （单位秒）

如 CPU 主频 100MHz，定时器 10ms 中断可设置为：

```
#define CPU_FREQ 100000000        /* 处理器主频 100MHz */
#define TIMER_SCALER_CNT          *(volatile unsigned int *)0x80000060
#define TIMER_SCALER_LOAD    *(volatile unsigned int *)0x80000064
#define TIMER0_CNT       *(volatile unsigned int *)0x80000040
#define TIMER0_LOAD      *(volatile unsigned int *)0x80000044
#define TIMER0_CTRL      *(volatile unsigned int *)0x80000048

TIMER_SCALER_LOAD = CPU_FREQ/1000000 -1;/*预分频重载,每 us 溢出*/
TIMER_SCALER_CNT  = CPU_FREQ/1000000 -1;/*预分频计数初值*/
TIMER0_LOAD       = 10000 -1;/*定时器重载，每 10000us*/
TIMER0_CNT        = 10000 -1;/*定时器计数初值*/
TIMER0_CTRL       = 0x7; /*开始定时，自动重载，允许定时器发出中断*/
```

7.4.2.6 串口

BM3803FMGRH 提供了 3 个串口（支持全双工发送和接收数据）。波特率由分频器对系统时钟分频产生，设置串口 1 的参数示例如下：

```
#define BAUD_RATE 115200
#define UART_SCALER  ((((CPU_FREQ*10) / (8 * BAUD_RATE))-5)/10)
#define UART1_SCALER_SET     *(volatile unsigned int *)0x8000007C
#define UART1_STATUS    *(volatile unsigned int *)0x80000074
#define UART1_CTRL  *(volatile unsigned int *)0x80000078
#define GPIO_LOW_DIR    *(volatile unsigned int *)0x800000A4

UART1_SCALER_SET  = UART_SCALER ;//115200bps
UART1_STATUS    = 0; /*清除状态*/
UART1_CTRL  = 3; /*收发使能*/
GPIO_LOW_DIR    = 0x00008000;   /*串口 1 发送、接收需要设置复用 GPIO 的方向，且不考虑其它 GPIO 使用*/
```

7.4.2.7 GPIO

GPIO 作为普通 IO 使用时，其**方向寄存器**和**数据寄存器**要配合使用，先由**方向寄存器**设置方向，再由**数据寄存器**发送或读入数据，这些均与中断设置无关。

如 GPIO 中，高 16 位（31-16）设置为输出，低 16 位（15-0）设置为输入，如下：

```
#define GPIO _DIR       *(volatile unsigned int *)0x800000A4
#define GPIO _DATA      *(volatile unsigned int *)0x800000A0
GPIO _DIR = 0xffff0000;
```

此时，从 GPIO15-GPIO0 读入的数据：

```
unsigned int i = GPIO _DATA;    /*使用时取其低 16 位，即低 16 位值等于当前给 GPIO15-GPIO0 管脚的赋值，管脚接高电平则值为 1，接低电平则值为 0*/
```

向数据寄存器中写数，会从 GPIO31-GPIO16 输出数据，但不影响从 GPIO15-GPIO0 读入

的数据。

GPIO _DATA = 0x5555AAAA;　　/*高 16 位输出，此时高 16 位 GPIO _DATA [31:16]的值为 0x5555，而低 16 位 GPIO _DATA [15:0]的值保持为读入的数据不变*/

7.4.2.8 PCI

PCI 进行读写操作时一般采用 DMA 操作，需要配置 PCI 起始地址、AHB 起始地址以及 DMA 读写控制。

DMA 写操作示例：

*(unsigned int *)0xc0000000 = 0xc0000000; // DMA 写传输 PCI 总线起始地址

*(unsigned int *)0xc0000004 = 0x40000000; // DMA 写传输 AHB 总线起始地址

*(unsigned int *)0xc0000008 = 0x10071;　　//写 DMA 的空间大小和控制字

发起将 AHB 地址 0x4000-0000 处 0x100 字节写往 PCI 地址 0xC000-0000 处的 DMA Memory Write 操作。

DMA 读操作示例：

*(unsigned int *)0xc0000020 = 0x40000000; // DMA 读传输 PCI 总线起始地址

*(unsigned int *)0xc0000024 = 0x40001000; // DMA 读传输 AHB 总线起始地址

*(unsigned int *)0xc0000028 = 0x100a1;　　//读 DMA 的空间大小和控制字

发起将槽位地址在 0x4000-0000 处的 PCI 设备配置空间中 0x100 字节的内容读入 AHB 地址 0x4000-1000 处的 DMA Memory Read 操作。

BM3803FMGRH PCI Host 设备通过配置 DMA 进行 PCI 配置时，应尽量使用单字 DMA 读写。配置 DMA 读写时 PCI 地址的确定由 PCI 总线底板的槽位及选择关系决定。

7.5 产品防护

7.5.1 电装及防护措施

器件应采取防静电措施进行操作。推荐下列操作措施：

a) 器件应在防静电的工作台上操作；

b) 试验设备和器具应接地；

c) 不能直接用手触摸器件引线，应佩戴防静电指套和腕带；

d) 器件应存放在防静电材料制成的容器中；

e) 生产、测试、使用及流转过程工作区域内应避免使用能引起静电的塑料、橡胶或丝织物；

f) 相对湿度应尽可能保持在 30%~70%。

7.5.2 包装

器件包装应至少满足以下要求：

a) 由无腐蚀的材料制成；

b) 具有足够的强度，能够经得起搬运过程中的震动和冲击；

c) 用抗静电材料涂敷过或浸渍过，具备足够的抗静电能力；

d) 能够牢固的把所装器件支撑在一定的位置；

e) 能保持器件引线不发生变形；

f) 没有锋利的棱角；

g) 能安全容易的移动、检查和替换器件；

h) 一般不使用聚氯乙稀、氯丁橡胶、乙烯树脂和聚硫化物等材料，也不允许使用有硫、盐、酸、碱等腐蚀成分的材料，使用具有低放气指数、低尘粒脱落的材料制造为宜。

7.5.3 运输和贮存

器件在运输和贮存过程中，至少应满足以下要求：

a) 运输：在避免雨、雪直接影响的条件下，装有产品的包装箱可以用任何运输工具运输。但不能和带有酸性、碱性和其它腐蚀性物体堆放在一起。

b) 贮存：包装好的产品应贮存在环境温度为 16℃~28℃，相对湿度不大于 30%~70%，周围没有酸、碱或其它腐蚀性气体且通风良好的库房里。

八、 用户关注产品信息

8.1 产品鉴定信息

鉴定产品批次		1312
鉴定执行标准	总规范名称及编号	半导体集成电路总规范 GJB 597A-1996
	详细规范名称及编号	《半导体集成电路 BM3803FMGRH 型 SPARC V8 辐照加固微处理器详细规范》Q/Zt 20273-2013
	附加技术条件	无
	质量等级	GJB597A-1996 B 级
鉴定情况	鉴定试验日期	2013 年 6 月 8 日~2013 年 8 月 1 日
	鉴定试验机构	军用电子元器件北京第一检测中心

8.2 产品标识

产品标识按下图 8-2-1 所示。“CC”为用户标识，表示质量等级 CAST C 级，“BM3803FMGRH”为电路型号，“XXXX”为生产批号，“001”为电路序号，“ΔΔ”为静电标识。

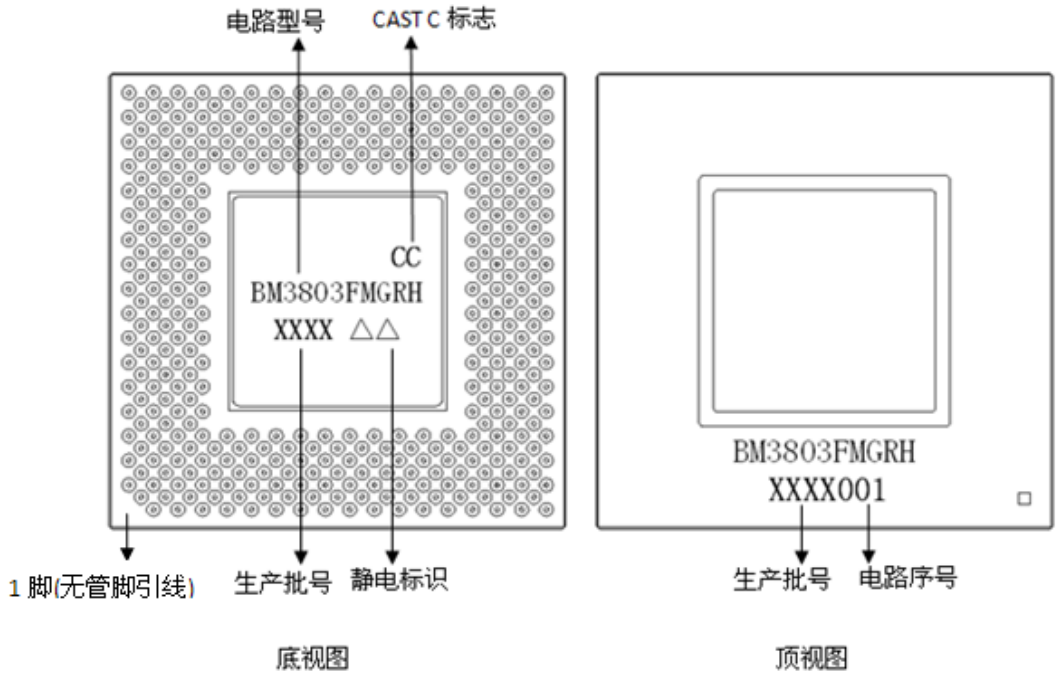


图 8-2-1 器件标识示意图

8.3 研制生产单位联系方式

通信地址：北京市丰台区东高地四营门北路 2 号

邮政编码：100076

技术联系人：SoC 部　刘亚丽　010-67968115-8215

市场联系人：市场二部　010-67968115-6313

传真：010-68757706

网址：www.bmti.com.cn

附录 1 警示页

1、 在产品使用过程中如有任何不明确之处，请及时反馈。

2、 若产品出现问题需要解焊，请提前经过产品技术人员确认再行解焊。

3、 不使用 JTAG 功能时，trst 管脚必须通过 1K 电阻下拉。

4、 电源上电顺序：先 I/O 电压 3.3V 上电，后核电压 1.8V 上电，禁止出现内核电源先于 I/O 电源上电的情况。

5、 输入管脚不得悬空，当输入管脚不使用时，需根据管脚功能上拉或下拉，避免悬空。

6、 输出管脚不使用时，禁止与其他管脚连接，包括电源和地。

7、 程序编写时禁止访问无响应地址，PCI 功能不使用时，无响应地址范围：0xA0000000~0xFFFFFFFF；PCI 功能使用时，无响应地址范围：0xA0000000~0xBFFFFFFF。

附录 2 寄存器列表

2.1 外设寄存器列表

2.1.1 0x80000000 地址上的寄存器

地址	寄存器描述	地址	寄存器描述
0x8000-0000	外部存储器配置寄存器 1	0x8000-0088	UART2 控制寄存器
0x8000-0004	外部存储器配置寄存器 2	0x8000-008C	UART2 分频器寄存器
0x8000-0008	外部存储器配置寄存器 3	0x8000-0090	中断级别/屏蔽控制寄存器
0x8000-000C	AHB 失效地址寄存器	0x8000-0094	中断请求寄存器
0x8000-0010	AHB 状态寄存器	0x8000-0098	中断强制寄存器
0x8000-0014	Cache 控制寄存器	0x8000-009C	中断清除寄存器
0x8000-0018	Power-down 寄存器	0x8000-00A0	GPIO 数据寄存器
0x8000-001C	写保护寄存器 1	0x8000-00A4	GPIO 方向寄存器
0x8000-0020	写保护寄存器 2	0x8000-00A8	GPIO 中断配置寄存器
0x8000-0024	产品配置寄存器	0x8000-00C4	DSU 串口状态寄存器
0x8000-0040	定时器 timer0 计数寄存器	0x8000-00C8	DSU 串口控制寄存器
0x8000-0044	定时器 timer0 重载寄存器	0x8000-00CC	DSU 串口波特率重载寄存器
0x8000-0048	定时器 timer0 控制寄存器	0x8000-00D0	PCI 复位控制寄存器
0x8000-004C	看门狗寄存器	0x8000-00E0	UART3 数据寄存器
0x8000-0050	定时器 timer1 计数寄存器	0x8000-00E4	UART3 状态寄存器
0x8000-0054	定时器 timer1 重载寄存器	0x8000-00E8	UART3 控制寄存器
0x8000-0058	定时器 timer1 控制寄存器	0x8000-00EC	UART3 分频器寄存器
0x8000-0060	预分频器计数寄存器	0x8000-0100	外部存储器容错配置寄存器 1
0x8000-0064	预分频器重载寄存器	0x8000-0104	外部存储器容错配置寄存器 2
0x8000-0070	UART1 数据寄存器	0x8000-0108	外部存储器容错配置寄存器 3
0x8000-0074	UART1 状态寄存器	0x8000-010C	外部存储器容错配置寄存器 4
0x8000-0078	UART1 控制寄存器	0x8000-0110	Cache 容错控制寄存器
0x8000-007C	UART1 分频器寄存器	0x8000-0114	Cache 造错寄存器
0x8000-0080	UART2 数据寄存器	0x8000-0118	Cache 错误计数寄存器
0x8000-0084	UART2 状态寄存器	0x8000-011C	保留

2.1.2 0x90000000 地址上的寄存器

地址	寄存器描述	地址	寄存器描述
0x9000-0000	DSU 控制寄存器	0x9000-0014	AHB 屏蔽地址寄存器 1
0x9000-0004	Trace buffer 控制寄存器	0x9000-0018	AHB 断点寄存器 2
0x9000-0008	Time tag counter	0x9000-001C	AHB 屏蔽地址寄存器 2
0x9000-0010	AHB 断点寄存器 1		

2.1.3 0xC0000000 地址上的寄存器

地址	寄存器描述	地址	寄存器描述
0xC000-0000	DMA 写操作 PCI 总线起始地址寄存器	0xC000-0074	PCI-AHB 窗口可预取范围控制寄存器
0xC000-0004	DMA 写操作 AHB 总线起始地址寄存器	0xC000-0078	PCI-AHB 窗口丢弃定时寄存器
0xC000-0008	DMA 写操作控制寄存器	0xC000-007C	AHB-PCI 窗口的丢弃定时寄存器
0xC000-0020	DMA 读操作 PCI 总线起始地址寄存器	0xC000-0080	PCI 控制寄存器
0xC000-0024	DMA 读操作 AHB 总线起始地址寄存器	0xC000-0084	PCI 设备和供应商标识寄存器
0xC000-0028	DMA 读传输控制寄存器	0xC000-0088	PCI 子系统设备和子系统供应商配置寄存器
0xC000-0040	CPU 中断掩码寄存器	0xC000-008C	PCI 分类代码和版本配置寄存器
0xC000-0044	CPU 中断状态寄存器	0xC000-0090	PCI 仲裁状态寄存器
0xC000-0048	CPU 中断命令寄存器	0xC000-0094	PCI-AHB 窗口不可预取空间大小配置寄存器
0xC000-004C	CPU 状态及版本寄存器	0xC000-0098	PCI-AHB 窗口可预取空间大小配置寄存器
0xC000-0070	不可预取范围 PCI-AHB 窗口范围控制寄存器		

2.1.4 PCI 配置空间的寄存器

地址	寄存器描述	地址	寄存器描述
PCI 配置空间 0x00	PCI 设备和供应商标识寄存器	PCI 配置空间 0x28	PCI Cardbus CIS Pointer 寄存器
PCI 配置空间 0x04	PCI 状态和命令寄存器	PCI 配置空间 0x2C	PCI 子系统设备和子系统供应商配置寄存器
PCI 配置空间 0x08	PCI 分类代码和版本配置寄存器	PCI 配置空间 0x30	PCI 扩展 ROM 基地址寄存器
PCI 配置空间 0x0C	PCI Latency Timer 寄存器	PCI 配置空间 0x34	PCI 能力指针寄存器
PCI 配置空间 0x10	PCI 基地址寄存器 0	PCI 配置空间 0x38	PCI 保留寄存器
PCI 配置空间 0x14	PCI 基地址寄存器 1	PCI 配置空间 0x3C	PCI 中断线及中断引脚寄存器
PCI 配置空间 0x18	PCI 基地址寄存器 2	PCI 配置空间 0x80	PCI 中断掩码寄存器
PCI 配置空间 0x1C	PCI 基地址寄存器 3	PCI 配置空间 0x84	PCI 中断状态寄存器
PCI 配置空间 0x20	PCI 基地址寄存器 4	PCI 配置空间 0x88	PCI 中断命令寄存器
PCI 配置空间 0x24	PCI 基地址寄存器 5	PCI 配置空间 0x8C	PCI 状态及版本信息寄存器

附录 3 寄存器描述

3.1 IU 寄存器

BM3803FMGRH 的 IU 寄存器包括窗口寄存器和全局寄存器，实现了 8 个窗口，每个窗口寄存器由输入寄存器（in register）、本地寄存器（local register）和输出寄存器（out register）组成。输入、输出寄存器主要用于向子程序传递参数、接收子程序运行结果和跟踪存储器堆栈。当程序调用时，调用程序（caller）的输出寄存器成为被调程序（callee）的输入寄存器。本地寄存器用于保存自动类型的变量（如 C 语言中的 auto 型变量）和大部分临时值。

与输入、本地和输出寄存器不同，全局寄存器不属于任何寄存器窗口。全局寄存器由 8 个全局范围的寄存器组成，类似传统处理器结构中的寄存器。全局寄存器%g0 写无效，读出恒为零。其余全局寄存器（%g1~%g7）可以用于保存临时变量、全局变量或全局指针（用户变量或作为程序执行环境一部分保存的值）。

3.1.1 全局寄存器

%g0 ： 0x9002-0280；

%g1 ： 0x9002-0284；

%g2 ： 0x9002-0288；

%g3 ： 0x9002-028C；

%g4 ： 0x9002-0290；

%g5 ： 0x9002-0294；

%g6 ： 0x9002-0298；

%g7 ： 0x9002-029C。

3.1.2 窗口寄存器

1. 窗口 0 寄存器

cwp = 0

%o0：0x9002-0020；%l0：0x9002-0040；%i0：0x9002-0060；

%o1：0x9002-0024；%l1：0x9002-0044；%i1：0x9002-0064；

%o2：0x9002-0028；%l2：0x9002-0048；%i2：0x9002-0068；

%o3：0x9002-002C；%l3：0x9002-004C；%i3：0x9002-006C；

%o4：0x9002-0030；%l4：0x9002-0050；%i4：0x9002-0070；

%o5：0x9002-0034；%l5：0x9002-0054；%i5：0x9002-0074；

%o6：0x9002-0038；%l6：0x9002-0058；%i6：0x9002-0078；

%o7：0x9002-003C；%l7：0x9002-005C；%i7：0x9002-007C。

2. 窗口 1 寄存器

cwp = 1

%o0 ：0x9002-0060；%l0：0x9002-0080；%i0：0x9002-00A0；

%o1 ：0x9002-0064；%l1：0x9002-0084；%i1：0x9002-00A4；

%o2 ：0x9002-0068；%l2：0x9002-0088；%i2：0x9002-00A8；

%o3 ：0x9002-006C；%l3：0x9002-008C；%i3：0x9002-00AC；

%o4 ：0x9002-0070；%l4：0x9002-0090；%i4：0x9002-00B0；

%o5 ：0x9002-0074；%l5：0x9002-0094；%i5：0x9002-00B4；

%o6 ：0x9002-0078；%l6：0x9002-0098；%i6：0x9002-00B8；

%o7 ：0x9002-007C；%l7：0x9002-009C；%i7：0x9002-00BC。

3. 窗口 2 寄存器

cwp = 2

%o0 ：0x9002-00A0；%l0：0x9002-00C0；%i0：0x9002-00E0

%o1 ：0x9002-00A4；%l1：0x9002-00C4；%i1：0x9002-00E4；

%o2 ：0x9002-00A8；%l2：0x9002-00C8；%i2：0x9002-00E8；

%o3 ：0x9002-00AC；%l3：0x9002-00CC；%i3：0x9002-00EC；

%o4 ：0x9002-00B0；%l4：0x9002-00D0；%i4：0x9002-00F0；

%o5 ：0x9002-00B4；%l5：0x9002-00D4；%i5：0x9002-00F4；

%o6 ： 0x9002-00B8；%l6：0x9002-00D8；%i6：0x9002-00F8；

%o7 ： 0x9002-00BC；%l7：0x9002-00DC；%i7：0x9002-00FC。

4. 窗口 3 寄存器

cwp = 3

%o0 ： 0x9002-00E0；%l0 ： 0x9002-0100；%i0：0x9002-0120；

%o1 ： 0x9002-00E4；%l1 ： 0x9002-0104；%i1：0x9002-0124；

%o2 ： 0x9002-00E8；%l2 ： 0x9002-0108；%i2：0x9002-0128；

%o3 ： 0x9002-00EC；%l3：0x9002-010C；%i3：0x9002-012C；

%o4：0x9002-00F0；%l4 ： 0x9002-0110；%i4：0x9002-0130；

%o5 ： 0x9002-00F4；%l5 ： 0x9002-0114；%i5：0x9002-0134；

%o6 ： 0x9002-00F8；%l6 ： 0x9002-0118；%i6：0x9002-0138；

%o7：0x9002-00FC；%l7 ： 0x9002-011C；%i7：0x9002-013C。

5. 窗口 4 寄存器

cwp = 4

%o0 ： 0x9002-0120；%l0：0x9002-0140；%i0：0x9002-0160；

%o1 ： 0x9002-0124；%l1：0x9002-0144；%i1：0x9002-0164；

%o2 ： 0x9002-0128；%l2：0x9002-0148；%i2：0x9002-0168；

%o3 ： 0x9002-012C；%l3：0x9002-014C；%i3：0x9002-016C；

%o4 ： 0x9002-0130；%l4：0x9002-0150；%i4：0x9002-0170；

%o5 ： 0x9002-0134；%l5：0x9002-0154；%i5：0x9002-0174；

%o6 ： 0x9002-0138；%l6：0x9002-0158；%i6：0x9002-0178；

%o7 ： 0x9002-013C；%l7：0x9002-015C；%i7：0x9002-017C。

6. 窗口 5 寄存器

cwp = 5

%o0 ： 0x9002-0160；%l0：0x9002-0180；%i0：0x9002-01A0；

%o1 ： 0x9002-0164；%l1：0x9002-0184；%i1：0x9002-01A4；

%o2 ： 0x9002-0168；%l2：0x9002-0188；%i2：0x9002-01A8；

%o3 ： 0x9002-016C；%l3：0x9002-018C；%i3：0x9002-01AC；

%o4 ： 0x9002-0170；%l4：0x9002-0190；%i4：0x9002-01B0；

%o5 ： 0x9002-0174；%l5：0x9002-0194；%i5：0x9002-01B4；

%o6 ： 0x9002-0178；%l6：0x9002-0198；%i6：0x9002-01B8；

%o7 ： 0x9002-017C；%l7：0x9002-019C；%i7：0x9002-01BC。

7. 窗口 6 寄存器

cwp = 6

%o0 ： 0x9002-01A0；%l0：0x9002-01C0；%i0：0x9002-01E0；

%o1 ： 0x9002-01A4；%l1：0x9002-01C4；%i1：0x9002-01E4；

%o2 ： 0x9002-01A8；%l2：0x9002-01C8；%i2：0x9002-01E8；

%o3 ： 0x9002-01AC；%l3：0x9002-01CC；%i3：0x9002-01EC；

%o4 ： 0x9002-01B0；%l4：0x9002-01D0；%i4：0x9002-01F0；

%o5 ： 0x9002-01B4；%l5：0x9002-01D4；%i5：0x9002-01F4；

%o6 ： 0x9002-01B8；%l6：0x9002-01D8；%i6：0x9002-01F8；

%o7 ： 0x9002-01BC；%l7：0x9002-01DC；%i7：0x9002-01FC。

8. 窗口 7 寄存器

cwp = 7

%o0 ： 0x9002-01E0；%l0 ： 0x9002-0000；%i0：0x9002-0020；

%o1 ： 0x9002-01E4；%l1 ： 0x9002-0004；%i1：0x9002-0024；

%o2 ： 0x9002-01E8；%l2 ： 0x9002-0008；%i2：0x9002-0028；

%o3 ： 0x9002-01EC；%l3：0x9002-000C；%i3：0x9002-002C；

%o4：0x9002-01F0；%l4 ： 0x9002-0010；%i4：0x9002-0030；

%o5 ： 0x9002-01F4；%l5 ： 0x9002-0014；%i5：0x9002-0034；

%o6 ： 0x9002-01F8；%l6 ： 0x9002-0018；%i6：0x9002-0038；

%o7 ： 0x9002-01FC；%l7 ： 0x9002-001C；%i7：0x9002-003C。

3.2 浮点寄存器

BM3803FMGRH 有 32 个浮点寄存器%f0~%f31，硬件复位后均是不定值，初始化程序必须对其赋值。与 IU 寄存器类似，浮点寄存器必须通过软件管理。编译器使用浮点寄存器保存用户变量、编译器临时变量和运算中的浮点数值。

浮点寄存器地址如下：

%f0：0x9002-0200；%f8：0x9002-0220；

%f1：0x9002-0204；%f9：0x9002-0224；

%f2：0x9002-0208；%f10：0x9002-0228；

%f3：0x9002-020C；%f11：0x9002-022C；

%f4：0x9002-0210；%f12：0x9002-0230；

%f5：0x9002-0214；%f13：0x9002-0234；

%f6：0x9002-0218；%f14：0x9002-0238；

%f7：0x9002-021C；%f15：0x9002-023C；

%f16：0x9002-0240；%f24：0x9002-0260；

%f17：0x9002-0244；%f25：0x9002-0264；

%f18：0x9002-0248；%f26：0x9002-0268；

%f19：0x9002-024C；%f27：0x9002-026C；

%f20：0x9002-0250；%f28：0x9002-0270；

%f21：0x9002-0254；%f29：0x9002-0274；

%f22：0x9002-0258；%f30：0x9002-0278；

%f23：0x9002-025C；%f31：0x9002-027C。

3.3 状态寄存器

IU 的状态寄存器中，PC 和 nPC 硬件复位值分别是 0x0 和 0x4，在用户软件/程序中不可见，但可以进行调试，其余的 15 个状态寄存器硬件复位值均是不定值，初始化程序需要对

其赋初值。依据《The SPARC Architecture Manual Version 8》手册的规定，对于状态寄存器的写操作指令是延迟指令。BM3803FMGRH 根据具体实现，对于状态寄存器 PSR、WIM、TBR、ASR16、ASR17 的写操作指令是三指令延迟指令，若用户软件在这些指令后三指令周期内访问或隐含访问这些状态寄存器的指令，须（增加指令如 nop）满足该寄存器前一指令写入三指令后生效的要求，确保后续指令访问到的是已更新的状态寄存器。

1. 处理器状态寄存器（PSR）

建议软件初始化值：0x080210c0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMPL				VER				N	Z	V	C	保留						EC	EF	PIL				S	PS	ET	保留		CWP		
r				r				r/w				r						r	r/w	r/w				r/w			r		r/w		
0				1	0	0	0	xxxx				0	0	1	0	0	0	0	x	xxxx				1	x	0	00		xxx		

- IMPL[31:28]：处理器实现的等级，只读，其值为 4’b0000。
- VER[27:24]：版本号，其值为 4’b1000。
- ICC[23:20]：整数单元的条件码，被以字母 CC 结尾的指令操作。N，表示 ALU 的逻辑运算单元的 32 位结果是否为负数，1 表示结果是负数；Z，表示运算结果是否是零，1 表示结果是零；V，表示运算结果是否超出 32 位数的范围，1 表示结果溢出；C，表示运算的结果是否引起进位，1 表示结果有进位。
- EC[13]：表示是否使能协处理器，1 表示使能。
- EF [12]：表示是否使能浮点处理器，1 表示使能。
- PIL[11:8]：中断级别，处理器只能处理高于 PIL 级别的中断。
- S[7]：指定处理器处于用户态还是管理态，1 表示处于管理态。
- PS[6]：保留最近一次中断发生时 S 位中的值。
- ET[5]：当 ET=0 时禁止陷阱，当 ET=1 时表示使能陷阱。
- CWP[2:0]：当前窗口指针，可以识别当前寄存器窗口。

位号	助记符	描述
31:28	IMPL	规范实现标识，定义为4'b000
27:24	VER	处理器版本号，定义为8，4'b1000
23	N	整数单元的条件码，表示运算结果是否是负数 1 = 负数 0 = 非负数
22	Z	整数单元的条件码，表示运算结果是否是零 1 = 零 0 = 非零
21	V	整数单元的条件码，表示运算结果是否超出32位数的范围 1 = 溢出 0 = 不溢出
20	C	整数单元的条件码，表示运算结果是否引起进位 1 = 进位 0 = 不进位
13	EC	表示是否使能协处理器 1 = 使能 0 = 禁能
12	EF	表示是否使能浮点处理器 1 = 使能 0 = 禁能
11:8	PIL	中断级别，处理器只能处理高于PIL级别的中断

7	S	表示 处理器处于管理模式还是用户模式 1 = 管理模式 0 = 用户模式
6	PS	保存 最近一次陷阱发生时S 位的值
5	ET	表示陷阱是否使能 1 = 陷阱使能 0 = 陷阱禁用
4:0	CWP	当前窗口指针

2. 窗口无效屏蔽寄存器（WIM）

建议软件初始化值：0x2。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																								Windows							
																								7	6	5	4	3	2	1	0
r																								r/w							
0																								x							

BM3803FMGRH 具有 8 个寄存器窗口，WIM 会随着程序的运行而变化。WIM[n]=0 表示窗口 n 有效，WIM[n]=1 表示窗口 n 无效。

位号	助记符	描述
0 < n < 7	Windows[n]	表示窗口是否有效 0 = 有效 1 = 无效

3. 异常基址标志寄存器（TBR）

建议软件初始化值：用户软件自定义。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBA																				TT								ZERO			
r/w																				r								r			
x																				x								0000			

- TBA[31:12]：陷阱的基地址，这个基址由软件设置，包含高位的 20 位有效位和陷阱地址表，通过 WRTBR 指令实现对 TBA 字段的读写。
- TT[11:4]：陷阱类型字段，当陷阱发生时硬件对这 8 位进行写入，保持值不变，直到下一个陷阱发生。
- ZERO：全零。

位号	助记符	描述
31:12	TBA	陷阱的基地址 由软件设置，包含陷阱表基地址的高20位
11:4	TT	陷阱类型字段 陷阱发生由硬件写入，并保存到下一个陷阱发生
3:0	ZERO	全零

4. 乘法/除法寄存器（Y）

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Y																															
r/w																															
x																															

32 位的 Y 寄存器在乘法运算时，其内容为整型乘积的高 32 位；在除法运算时，其内容为整型被除数的高 32 位。

5. 程序计数器（PC）

建议软件初始化值：硬件复位值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Y																															
r/w																															
0																															

PC 为 32 位寄存器，保存 IU 正在执行指令的地址。

6. 程序计数器（nPC）

建议软件初始化值：硬件复位值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Y																															
r/w																															
0x4																															

nPC 为 32 位寄存器，保存着将要被执行的下一条指令的地址。

7. Regfile EDAC 辅助状态寄存器 ASR16

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB			R						TCB							CNT															
r/w			r						r/w							r															
xxx			0						x							x															

- [31:29]：EDAC 的控制位，其值含义如下：

  “000/001/011/100/110/111”，“错误”计数。

  “010”，“错误”计数清零。

“101”，打开造错功能，包括校验位造错和数据位造错。

- [28:23]：保留位。
- [22:16]：校验位造错，可读写。
- [15:0]：记录发生一位错误的次数，只读。

位号	名称	描述
31:29	CB	000或001或011或100或110或111 = 错误计数 010 = 错误计数清零 101 = 造错使能，即对及数据位和校验位造错
22:16	TCB	对校验位造错,1表示造错，0表示不造错
15:0	CNT	记录发生一位错的次数

8. Regfile EDAC 造错控制寄存器 ASR17

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCB																															
r/w																															
x																															

- [31:0]：数据位造错，可读写。

位号	助记符	描述
31：0	DCB	数据位造错 1表示造错 0表示不造错

BM3803FMGRH 的寄存器堆采用“纠一检二”的 EDAC 机制，即纠正一位错，检查两位错。ASR16 和 ASR17 控制 Regfile 的 EDAC 功能，硬件复位时，硬件不对 ASR16 造错控制位[31:29]、计数位[15:0]清零，也不对 ASR17 赋值。当设置 ASR16[31:29]=101 即造错功能使能时，则电路进入造错功能模式，此时写入到 Regfile 寄存器的值，是写入值与造错辅助寄存器 ASR17 值异或的结果，可能造成写入到 Regfile 寄存器的值与预期值不符，再读取 Regfile 寄存器时，检测到 Regfile 寄存器的 32 位数据和 7 位校验位不匹配；如果检测到两位错，则进入

TT=0x20 陷阱，如果有多位错，则可能引发 TT=0x07 陷阱或者 TT=0x09 陷阱或者运算、取数据错误。

在对 Regfile 寄存器赋值前，必须首先保证 ASR16 不使能造错，否则，会造成 Regfile EDAC 错。

9. 观察点寄存器 ASR24

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WADDR																														保留	IF
r/w																														r	r/w
x																														0	0

位号	助记符	描述
31:2	WADDR	表示观察点的地址范围
0	IF	取值使能 1 = 使能 0 = 禁能

10. 观察点寄存器 ASR26

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WADDR																														保留	IF
r/w																														r	r/w
x																														0	0

位号	助记符	描述
31:2	WADDR	表示观察点的地址范围
0	IF	取值使能 1 = 使能 0 = 禁能

11. 观察点寄存器 ASR28

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WADDR																														保留	IF
r/w																														r	r/w
x																														0	0

位号	助记符	描述
31:2	WADDR	表示观察点的地址范围
0	IF	取值使能　1 = 使能　0 = 禁能

12. 观察点寄存器 ASR30

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WADDR																														保留	IF
r/w																														r	r/w
x																														0	0

位号	助记符	描述
31:2	WADDR	表示观察点的地址范围
0	IF	取值使能　1 = 使能　0 = 禁能

13. 观察点寄存器 ASR25

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMASK																														DL	DS
r/w																														r/w	r/w
x																														0	0

位号	助记符	描述
31:2	WMASK	表示与观察点的比较位 1 = 比较使能 0 = 比较禁能
1	DL	表示是否使能加载数据命中 1 = 使能加载 0 = 禁能加载
0	DS	表示是否使能存储数据命中 1 = 使能 0 = 禁能

14. 观察点寄存器 ASR27

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMASK																														DL	DS
r/w																														r/w	r/w
x																														0	0

位号	助记符	描述
31:2	WMASK	表示与观察点的比较位 1 = 比较使能 0 = 比较禁能
1	DL	表示是否使能加载数据命中 1 = 使能加载 0 = 禁能加载
0	DS	表示是否使能存储数据命中 1 = 使能 0 = 禁能

15. 观察点寄存器 ASR29

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMASK																														DL	DS
r/w																														r/w	r/w
x																														0	0

位号	助记符	描述
31:2	WMASK	表示与观察点的比较位 1 = 比较使能 0 = 比较禁能
1	DL	表示是否使能加载数据命中 1 = 使能加载 0 = 禁能加载
0	DS	表示是否使能存储数据命中 1 = 使能 0 = 禁能

16. 观察点寄存器 ASR31

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMASK																														DL	DS
r/w																														r/w	r/w
x																														0	0

位号	助记符	描述
31:2	WMASK	表示与观察点的比较位 1 = 比较使能 0 = 比较禁能
1	DL	表示是否使能加载数据命中 1 = 使能加载 0 = 禁能加载
0	DS	表示是否使能存储数据命中 1 = 使能 0 = 禁能

17. 浮点状态寄存器（FSR）

建议软件初始化值：0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD		保留		TEM					NS	保留		VER			FTT			保留		FCC		AEXC					CEXC				
r/w		r		r/w					r	r		r			r			r		r/w		r/w					r/w				
xx		00		xxxxx					0	00		000			xxx			00		00		xxxxx					xxxxx				

- RD[31:30]：舍入方向，根据 ANSI/IEEE 的 754-1985 标准选择 FPU 浮点算术操作时的舍入方向。

RD Round Toward:
0 Nearest(even, if tie) 1 0 2 +∞ 3 -∞

- TEM[27:23]：异常使能标志，使能由 FPops 引起的异常。这些值同 CEXC（当前异常域值）相与以决定是否对 IU 产生浮点异常。所有使能域的命名与 CEXC 中一致，“0”不使能异常，“1”使能异常。
- NS[22]：FPU 实现标志，使 FPU 按其设计定义产生所要实现的结果。
- VER[19:17]：FPU 设计实现版本，只读，其值为 3b’000。
- FTT[16:14]：浮点异常类型位，识别浮点异常类型，在指令 STFSR 或者另一条 FPop 指令执行之前，FTT 域值表示当前浮点异常类型。
- FCC[11:10]：包含浮点条件码内容。浮点比较指令[FCMP，FCMPE]用于更新 FCC。指令 STFSR 和 LDFSR 用于读写 FCC。指令 FBfcc 基于 FCC 的值进行分支控制。
- AEXC[9:5]： 通过 TEM 不使能 fp_exception 异常时，保存 IEEE_754 浮点异常。一条浮点操作指令完成之后，将 TEM 和 CEXC 两个字段进行逻辑与操作，如果结果非零，将会产生一个 fp_exception 异常陷阱；否则，新的 CEXC 字段与 AEXC 字段进行或操作后将结果保留到 AEXC 字段中。这样，当中断被屏蔽的时候，异常就会保存在 AEXC 中。
- CEXC[4:0]：表示最近执行的 Fpop 指令引起的一个或多个 IEEE 浮点异常。异常不发生时，其对应位被清零。

位号	助记符	描述
31:30	RD	舍入方向：00-就近舍入；01-向0舍入；10-向正无穷舍入；11-向负无穷舍入。
27:23	TEM	异常使能掩码 1 = 异常使能 0 = 异常禁能
22	NS	FPU 实现标志位
19:17	VER	FPU 设计实现版本
16:14	FTT	浮点异常类型
11:10	FCC	浮点条件代码位
9:5	AEXEC	TEM不使能fp_exception异常时，保存IEEE_754浮点异常。
4:0	CEXEC	表示最近执行的Fpop指令引起的一个或多个IEEE浮点异常

FSR 的浮点陷阱类型（FTT）字段：

FTT	陷阱类型	描述
0	None	无陷阱
1	IEEE_754 exception	表示产生一个符合IEEE 754－1985标准的陷阱
4	Sequence_error	表示FPU中三种异常错误条件之一

FSR 的浮点条件代码（FCC）：

FCC	描述
0	f_{rs1} = f_{rs2}
1	f_{rs1} < f_{rs2}
2	f_{rs1} > f_{rs2}
3	f_{rs1} ? f_{rs2} （无序） 如果f_{rs1} 或者 f_{rs2} 是signaling NaN或quiet NaN，表示两者无法比较
f_{rs1} 和 f_{rs2} 指存放在f寄存器中单精度或双精度浮点数的值。 注意：当FCMP或者FCMPE指令产生IEEE_exception陷阱时FCC并不改变。	

浮点数异常字段：

AEXC	CEXC	名称	描述
NVA	NVC	Invalid	表示将要执行操作的操作数是否为无效操作数 1 = 无效操作数 0 = 合法操作数
OFA	OFC	Overflow	表示在数据格式中舍入后的结果是否会上溢 1 = 上溢 0 = 未上溢
UFA	UFC	Underflow	表示在制定的数据格式中舍入的结果是否会下溢 1 = 下溢 0 = 不下溢
DZA	DZC	Division-by-zero	1=被零除，0=未被零除 0÷0不设置dzc位
NXA	NXC	Inexact	与无限精确的正确结果相比，表示舍入结果是否可认为是精确结果 1 = 不精确 0 = 精确

3.4 系统寄存器

1. 产品配置寄存器 （PCR）

建议软件初始化值：只读不可设初值。

Address = 0x8000-0024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMU	DSU	SDRCTRL	WTPNB			IMAC	NWINDOWS					ICSZ			ILSZ		DCSZ			DLSZ		DIVINST	MULINST	WDOG	MEMSTAT	FPU		PCI		WPRT	
r	r	r	r			r	r					r			r		r			r		r	r	r	r	r		r		r	
0	1	1	1	0	0	1	0	0	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0	1

- MMU[31]：‘0’，表明没有 MMU。
- DSU[30]：‘1’表明具有 DSU。
- STRCTRL[29]：‘1’表明存在 SDRAM 功能。
- WTPNB[28:26]：“100” 4 个 watchpoint。
- IMAC[25]：‘1’支持 MAC 指令。
- NWINDOWS[24:20]：“00111”具有 8 寄存器窗口数目。
- ICSZ[19:17]：“011”指令 Cache 的大小。
- ILSZ[16:15]：“11” 指令 Cache 的 LINE 的大小。
- DCSZ[14:12]：“011”数据 Cache 的大小。
- DLSZ[11:10]：“10”数据 Cache 的 LINE 的大小。
- DIVINST[9]：“1”，有硬件除法器。
- MULINST[8]：“1”，有硬件乘法器。
- WDOG[7]：“1”，有看门狗。
- MEMSTAT[6]：“1”，有总线状态寄存器。
- FPU[5:4]：“01”浮点处理器类型。
- PCI[3:2]：“11”，PCI 核的类型。
- WPRT [1:0]：WPROTEN 存在与否的标识。初始值是“01”。

位号	助记符	描述
31	MMU	0 = 表示没有MMU
30	DSU	1 = 表示具有DSU
29	SDRCTRL	1 = 表示存在SDRAM 功能
28:26	WTPNB	100 = 表示具有4 个 watchpoint
25	IMAC	1 = 表示支持MAC指令
24:20	NWINDOWS	00111 = 具有8 个寄存器窗口数目
19:17	ICSZ	011 = 表示指令Cache的大小
16:15	ILSZ	11 = 表示指令Cache的LINE大小
14:12	DCSZ	011 = 表示数据Cache的大小
11:10	DLSZ	10 = 数据Cache的LINE大小
9	DIVINST	1 = 包含硬件除法器
8	MULINST	1 = 包含有硬件乘法器
7	WDOG	1 = 包含有看门狗
6	MEMSTAT	1 = 包含有总线状态寄存器
5:4	FPU	01 = 表示浮点处理器类型
3:2	PCI	11 = 表示PCI的类型
1:0	WPRT	表示WPROTEN存在与否， 初始值为‘01’

2. AHB 失效地址寄存器

建议软件初始化值：只读不可设初值。

Address = 0x8000-000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FA																															
r																															
x																															

FA[31:0]：记录当访问发生错误时的总线地址。

位号	助记符	描述
31:0	FA	失效地址位 表示访问发生错误时的总线地址

3. AHB 状态寄存器

建议软件初始化值：硬件复位值。

Address = 0x8000-0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																							NE	RW	HMAST			HSIZE			
r																							r/w	r	r			r			
0																							0	x	xxxx			xxx			

- [31:9]：保留。
- [8]: 错误状态，“1”为有错误，“0”为无错误。当有错误发生后，需要软件清零。
- [7]: 读写访问，“1”为读，“0”为写。
- [6:3]: 记录访问失效时的 HMASER，即总线主设备的值。
- [2:0]: HSIZE 传输大小，“000”表示字节，“001”表示半字，“010”表示字，“011”表示双字。

位号	助记符	描述
8	NE	错误状态，“1”为有错误，“0”为无错误。
7	RW	读写访问，“1”为读，“0”为写。
6:3	HMAST	记录访问失效时HMAST的值，即总线主设备值
2:0	HISZE	HSIZE传输大小，“000”表示字节，“001”表示半字， “010”表示字，“011”表示双字

3.5 Cache 寄存器

1. Cache 控制寄存器

Address = 0x8000-0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRP		IRP		ISTS		DSTS		DS	FD	FI	R				IB	IP	DP	R								DF	IF	DCS		ICS	
r		r		r		r		r/w	w	w	r				r/w	r	r	r								r/w	r/w	r/w		r/w	
11		11		11		01		0	0	0	0000				0	0	0	0								x	x	00		00	

位号	助记符	描述
31:30	DRP	数据 Cache 的替换策略。
29:28	IRP	指令 Cache 的替换策略。
27:26	ISTS	指令 Cache 的相联数。
25:24	DSTS	数据 Cache 的相联数。
23	DS	数据 Cache 的 Snoop 使能位侦，使能时，若其他主设备对可 Cache 存

		储区域进行写操作，Cache 中相应数据会无效。
22	FD	数据 Cache 刷新使能位，使能时将 flush 数据 Cache。1 表示使能，0 表示禁能。
21	FI	指令 Cache 刷新使能位，使能时将 flush 指令 Cache。1 表示使能，0 表示禁能。
20：17	R	保留位，读出为 0。
16	IB	指令突发取值使能位，1 表示使能，0 表示禁能。
15	IP	指令 Cache 正在刷新指示位，1 表示指令 Cache 正在刷新，0 表示指令 Cache 刷新完成或未刷新。
14	DP	数据 Cache 正在刷新指示位，1 表示数据 Cache 正在刷新，0 表示数据 Cache 刷新完成或未刷新。
13：6	R	保留位，读出为 0。
5	DF	数据 Cache 中断冻结控制位，使能时数据 Cache 将在发生中断时自动冻结。1 表示使能，0 表示禁能。
4	IF	指令 Cache 中断冻结控制位，使能时指令 Cache 将在发生中断时自动冻结。1 表示使能，0 表示禁能。
3:2	DCS	数据 Cache 状态位，‘01’：数据 Cache 冻结，‘11’：数据 Cache 使能，‘x0’：数据 Cache 禁止。
1：0	ICS	指令 Cache 状态位，‘01’：指令 Cache 冻结，‘11’：指令 Cache 使能，‘x0’：指令 Cache 禁止。

2. Cache 容错控制寄存器 1

Address = 0x8000-0110

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VER			R															CMEEN			CNTRST	CPSEL		R				EPOS			
r			r															r/w			r/w	r/w		r				r/w			
001			0															000			0	00		0000				0000			

位号	助记符	描述
31:29	VER	Cache 版本号寄存器，只读，其值为 3’b001。
28:14	R	保留位，读出为 0。
13:11	CMEEN	Cache 造错使能标志位。"011"表示造错开启，其它值表示不造错。
10	CNTRST	Cache 校验计数器清零标志。置为’1’时，清除所有错误计数器。
9:8	CPSEL	Cache 造错目标区域选择位。"00"表示指令 Cache 的 data 区；"01"表示指令 Cache 的 tag 区；"10"表示数据 Cache 的 data 区；"11"表示数据

		Cache 的 tag 区。
7:4	R	保留位，读出为 0。
3:0	EPOS	Cache 造错位的高 4 位。当目标区域选择为 data 区，[3:0]对应数据的 4 个校验位；当目标区域选择为 tag 区，[3:0]对应 tag 的 4 个校验位。

3. Cache 容错控制寄存器 2

Address = 0x8000-0114

当 Cache 造错使能，且目标区域为 data 区时：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPOS																															
r/w																															
0																															

位号	助记符	描述
31:0	EPOS	Cache 造错位的低 32 位，EPOS[31:0]分别对应数据位的 data[31:0]。

当 Cache 造错使能，且目标区域为数据 Cache 的 tag 区时：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	TAG																			VALID				R							
r/w	r/w																			r/w				r							
0	0																			0				0							

位号	助记符	描述
31	LOCK	Cache 造错的第 31 位，对应数据 Cache 的 lock 位。
30:12	TAG	Cache 造错的第 30-12 位，对应数据 Cache 的 tag 位。
11:8	VALID	Cache 造错的第 11-8 位，对应数据 Cache 的 valid 位。
7:0	R	保留位，读出为 0。

当 Cache 造错使能，且目标区域为指令 Cache 的 tag 区时：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	TAG																			VALID								R			
r/w	r/w																			r/w								r			
0	0																			0								0			

位号	助记符	描述
31	LOCK	Cache 造错的第 31 位，对应指令 Cache 的 lock 位。
30:12	TAG	Cache 造错的第 30-12 位，对应指令 Cache 的 tag 位。
11:4	VALID	Cache 造错的第 11-4 位，对应指令 Cache 的 valid 位。
3:0	R	保留位，读出为 0。

4. Cache 容错控制寄存器 3

Address = 0x8000-0118

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTAGCNT						DDATACNT										ITAGCNT				IDATACNT											
r						r										r				r											
0						0										0				0											

位号	助记符	描述
31:26	DTAGCNT	数据 Cache 的 tag 区错误计数。
25:16	DDATACNT	数据 Cache 的 data 区错误计数。
15:10	ITAGCNT	指令 Cache 的 tag 区错误计数。
9: 0	IDATACNT	指令 Cache 的 data 区错误计数。

3.6 外部存储器接口寄存器

1. 外部存储器配置寄存器 1（MCFG1）

建议软件初始化值：0x00f1f21f（必须根据板级实际情况赋值）。

Address = 0x8000-0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			IOWDH		BRDY	BEXC	保留	IOWS				IOEN	保留		PRWWS					PRWEN	保留		PRWDH		保留				PRRWS			
r			r/w		r/w	r/w	r	r/w				r/w	r		r/w					r/w	r		r/w		r				r/w			
000			xx		0	0	0	1111				0	00		11111					0	0		xx		000				11111			

- [31:29]：保留。
- [28:27]：I/O 设备总线宽度（”00”=8，“01”=16，“1x”=32）。
- [26]：总线准备好(BRDYN) 使能，用于 I/O 设备。“1”为使能，“0”为不使能。
- [25]：总线错误(BEXCN)使能，用于 PROM、I/O 和 SRAM，“1”为使能，“0”为不使能。
- [24]：保留。
- [23:20]：I/O 设备读写等待周期数。初始值为 15。(“0000”=0,“0001”=1, “0010”=2,…, “1111”=15)。
- [19]：I/O 设备读写访问使能，“1”为使能，“0”为不使能。
- [18:17]：保留。
- [16:12]：PROM 写等待周期数。初始值为 31。(“00000”=0, “00001”=1, “00010”=2,…,“11111”=31)。
- [11]：PROM 写使能，“1”为使能，“0”为不使能。
- [10]：保留。
- [9:8]：PROM 数据宽度(“00”=8,“01”=16, “1x”=32)。复位初始值由 GPIO 的第 1、0 两位管脚配置。
- [7:5]：保留。
- [4:0]：PROM 读等待周期数。初始值为 31。(“00000”=0, “00001”=1, “00010”=2,…, “11111”=31)。

位号	助记符	描述
31:29 24，18:17 10，7:5	R	保留
28:27	IOWDH	I/O设备总线宽度（“00”=8，“01”=16，“1x”=32）
26	BRDY	总线准备好(BRDYN) 使能，用于I/O设备。“1”为使能，“0”为不使能。
25	BEXC	总线错误(BEXCN)使能，用于PROM、I/O和SRAM。“1”为使能，“0”为不使能。
23:20	IOWS	I/O设备读写等待周期数。初始值为15。(“0000”=0,“0001”=1，“0010”=2,…，“1111”=15)
19	IOEN	I/O设备读写访问使能。“1”为使能，“0”为不使能。
16:12	PRWWS	PROM写等待周期数。初始值为31。(“00000”=0,“00001”=1,“00010”=2,…,“11111”=31)
11	PRWEN	PROM写使能。“1”为使能，“0”为不使能。
9:8	PRWDH	PROM 数据宽度(“00”=8,“01”=16，“1x”=32)。复位初始值由GPIO的第1、0两位管脚配置。
4:0	PRRWS	PROM读等待周期数。初始值为31。(“00000”=0,“00001”=1,“00010”=2,…,“11111”=31)。

2. 外部存储器配置寄存器 2（MCFG2）

建议软件初始化值：0x7c47806f（必须根据板级实际情况赋值）

Address = 0x8000-0004

31	30	29 28 27	26	25 24 23	22 21	20 19	18 17 16 15	14	13	12 11 10 9	8	7	6	5 4	3 2 1 0
SDREF	TRP	TRFC	SDRCAS	SDRBS	SDRCLS	SDRCMD	SRWWS	SDREN	SRDIS	SRBS	I I	BRDY	RMW	SRWDH	SRRWS
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r	r/w	r/w	r/w	r/w
0	1	1 1 1	1	0 0 0	1 0	0 0	1 1 1 1	0	0	x x x x	0	x	x x	x x	1 1 1 1

- [31]：SDRAM 刷新使能，“1”为使能，“0”为不使能。
- [30]：SDRAM T_{RP} 时间。“0”表示 TRP 为 2 个时钟周期，“1”表示 TRP 为 3 个时钟周期。
- [29:27]：SDRAM T_{RFC} 时间。 T_{RFC} 为（设置数值＋3）个系统时钟周期。
- [26]：SDRAM CAS 延时。“0”表示 CAS 延时为 2 个时钟周期，“1”表示 CAS 延时为 3 个时钟周期。
- [25:23]：SDRAM BANK 的大小（“000”=4Mbyte,“001”=8Mbyte, “010”=16Mbyte .... “111”=512 Mbyte）。
- [22:21]：SDRAM Column 的大小。(“00” =256，“01” =512，“10” =1024，当[22:21]= “11”并且[25:23]= “111”时，为 4096，
  当[22:21]= “11”并且[25:23]!= “111”时，为 2048。)
- [20:19]：SDRAM 命令。当写入非 0 数值时，将产生 SDRAM 命令。

“01”=PRECHARGE, “10”=AUTO-REFRESH, “11”=LOAD-COMMAND-REGISTER.命令执行后，此数值清零。

- [18:15]：SRAM 写等待周期数，初始值为 15。(“0000”=0,“0001”=1,“0010”=2,…, “1111”=15)。
- [14]：SDRAM 使能。“1”为使能，“0”为不使能。
- [13]：SRAM 不使能。“1”为不使能，“0”为使能。
- [12:9]：SRAM bank 大小。定义每个 SRAM 大小。(“0000”=8 Kbyte, “0001”=16 Kbyte , … ,“1111”=256 Mbyte)。
- [8]：保留。
- [7]：总线准备好(BRDYN) 使能，用于第 5 bank 的 SRAM。“1”为使能，“0” 为不使能。
- [6]：SRAM “读改写”方式使能，“1”为使能，则 SRAM 写操作使用“读改写”方式，“0”为不使能，则 SRAM 写操作使用“写选通”方式。
- [5:4]：SRAM 数据总线宽度(“00”=8,“01”=16, “1x”=32)。
- [3:0]：SRAM 读等待周期数，初始值为 15。(“0000”=0, “0001”=1, “0010”=2,…,“1111”=15)。

位号	助记符	描述
31	SDREF	SDRAM 刷新使能，“1”为使能，“0”为不使能。
30	TRP	SDRAM TRP时间。“0”表示TRP为2个时钟周期，“1”表示TRP为3个时钟周期。
29:27	TRFC	SDRAM TRFC 时间。 TRFC为（设置数值+3）个系统时钟周期。
26	SDRCAS	SDRAM CAS 延时。“0”表示CAS 延时为2个时钟周期，“1”表示CAS 延时为3个时钟周期。
25:23	SDRBS	SDRAM BANK的大小 “000”=4 Mbyte,“001”=8 Mbyte, “010”=16 Mbyte .... “111”=512 Mbyte
22:21	SDRCLS	SDRAM Column的大小。 “00”=256, “01”=512, “10”=1024, 当[22:21]=“11”并且[25:23]=“111”时，为4096 当[22:21]=“11”并且[25:23]/=“111”时，为2048。
20:19	SDRCMD	SDRAM 命令。当写入非0数值时，将产生SDRAM命令。 “01”=PRECHARGE,“10”=AUTO-REFRESH,“11”=LOAD-COMMAND-REGISTER.命令执行后，此数值清零。
18:15	SRWWS	SRAM写等待周期数，初始值为15。(“0000”=0,“0001”=1,“0010”=2,…, “1111”=15)。
14	SDREN	SDRAM使能 1 = 使能 0 =禁能
13	SRDIS	SRAM禁能 1 = 禁能 0 =使能
12:9	SRBS	SRAM bank 大小。定义每个SRAM大小。 “0000”=8 Kbyte, “0001”=16 Kbyte , … ,“1111”=256 Mbyte
7	BRDY	总线准备好使能，用于第5 bank的SRAM。 “1”为使能，“0”为不使能。
6	RMW	SRAM读改写方式使能，“1”为使能，使用“读改写”方式，“0”为不使能，则使用“写选通”方式。
5:4	SRWDH	SRAM数据总线宽度 ‘00’=8 ‘01’=16 ‘1x’=32
3:0	SRRWS	SRAM读等待周期数，初始值为15 ‘0000’=0 ‘0001’=1 ‘0010’=2 ...‘1111’=15

3. 外部存储器配置寄存器 3（MCFG3）

建议软件初始化值：0x0（必须根据板级实际情况赋值）

Address = 0x8000-0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					SDRRV															保留											
r					r/w															r											
0					x															0											

- [31:27]：保留。
- [26:12]：SDRAM 刷新计数器重载值。计算 AUTO-REFRESH 命令间隔时间公式为：tREFRESH = ((reload value) + 1) / SYSCLK。
- [11:0]：保留。

位号	助记符	描述
26:12	SDRRV	SDRAM刷新计数器重载值 计算AUTO-REFRESH命令间隔时间公式为：$T_{REFRESH}$ = ((reload value) + 1) / SYSCLK

4. 外部存储器容错配置寄存器 1 （MECFG1）

建议软件初始化赋值：0x00400300（必须根据板级实际情况赋值）

Address = 0x8000-0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SR1EEN	SR2EEN	SR3EEN	SR4EEN	SR5EEN	ROMEEN	保留			EWB2	ERB	EWB1	SDEEN	EWB0	PRBS				保留				TCBAREA		保留	TCB							
r/w	r/w	r/w	r/w	r/w	r/w	r			r/w	r/w	r/w	r/w	r/w	r/w				r				r/w		r	r/w						
0	0	0	0	0	0	000			1	0	0	0	0	0000				0000				11		0	xxx xxxx						

- [31]：SRAM bank1 EDAC 使能，“1”为使能，“0”为不使能。
- [30]：SRAM bank2 EDAC 使能，“1”为使能，“0”为不使能。
- [29]：SRAM bank3 EDAC 使能，“1”为使能，“0”为不使能。
- [28]：SRAM bank4 EDAC 使能，“1”为使能，“0”为不使能。
- [27]：SRAM bank5 EDAC 使能，“1”为使能，“0”为不使能。
- [26]：PROM EDAC 使能，高有效。此字段的初值在上电复位时由 GPIO 的第 2 位管脚配置。
- [25:23]：保留。
- [22]：EDAC 写旁路（EWB2），EWB2、1、0=“011”时有效。
- [21]：EDAC 读旁路(ERB)，“1”为使能，“0”为不使能。
- [20]：EDAC 写旁路（EWB1），EWB2、1、0=“011”时有效。
- [19]：SDRAM EDAC 使能，“1”为使能，“0”为不使能。
- [18]：EDAC 写旁路(EWB0)，EWB2、1、0=“011”时有效。
- [17:14]：PROM bank size。定义每个 PROM 大小。 (“0000”=8 Kbyte, “0001”=16 Kbyte , … ,“1111”=256 Mbyte)。用于 8 位 PROM 的 EDAC。此字段的初值在上电复位时由外部管脚 ROMBSD[3:0]配置。
- [13:10]：保留。

- [9:8]：TCB area。将每个 bank 的 SRAM 和每个 bank 的 SDRAM 平均分成 4 份区域，读取此字段指向的区域中的数据，其对应的校验位，在 *ERB* 使能时存入 *TCB* 字段。第 5 片选的 SRAM 不受 *TCBarea* 字段控制。RPOM 不受 *TCBarea* 字段控制。
- [7]：保留。
- [6:0]：Test check bits(TCB)。用于测试校验，如果 MECFG1 中 ERB 被使能，且对于 8、32 位数据宽度的 PROM 和 SRAM，以及 32 位数据宽度的 SDRAM，那么在读数据时，根据 TCBarea 字段，读得的正常的校验位存入此字段。

位号	助记符	描述
31	SR1EEN	SRAM BANK1 EDAC 使能 1 = 使能 0 = 禁能
30	SR2EEN	SRAM BANK2 EDAC 使能 1 = 使能 0 = 禁能
29	SR3EEN	SRAM BANK3 EDAC 使能 1 = 使能 0 = 禁能
28	SR4EEN	SRAM BANK4 EDAC 使能 1 = 使能 0 = 禁能
27	SR5EEN	SRAM BANK5 EDAC 使能 1 = 使能 0 = 禁能
26	ROMEEN	Prom EDAC使能，1 = 使能 0 = 禁能 此字段的初值在上电复位时由GPIO的第2位配置。
22	EWB2	EDAC写旁路 EWB2、1、0 = “011”有效
21	ERB	EDAC读旁路使能， 1 = 使能 0 = 禁能
20	EWB1	EDAC写旁路使能，EWB2、1、0 = “011”有效
19	SDEEN	SDRAM EDAC使能，1 = 使能 0 = 禁能
18	EWB0	EDAC写旁路，EWB2、1、0 = “011”有效
17:14	PRBS	PROM bank size。定义每个PROM大小。 “0000”=8 Kbyte, “0001”=16 Kbyte , … , “1111”=256 Mbyte 用于8位prom的EDAC。此字段的初值在上电复位时由外部管脚ROMBSD [3:0]配置。
9:8	TCBAREA	TCB area。将每个bank的SRAM和每个bank的SDRAM平均分成4份区域，读取此字段指向的区域中的数据，其对应的校验位，在ERB使能时存入TCB字段。第5片选的SRAM不受TCBarea字段控制。RPOM不受TCBarea字段控制。
6:0	TCB	Test check bits(TCB)。用于测试校验，如果MECFG1中ERB被使能，且对于8、32位数据宽度的PROM和SRAM，以及32位数据宽度的SDRAM，那么在读数据时，根据TCBarea字段，读得的正常的校验位存入此字段。

5. 外部存储器容错配置寄存器 2 （MECFG2）

建议软件初始化值：硬件复位值。

Address = 0x8000-0104

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR																															
r/w																															
0																															

- [31:0]DR：数据位翻转。“1”为翻转，“0”为不翻转。

MECFG2 用于数据位造错。当 MECFG1 中 EWB 设置有效时，对于 32 位数据宽度 SRAM 和 SDRAM 整个字的写操作，则写入数据位按 MECFG2 来翻转。

位号	助记符	描述
31:0	DR	Data Reversal数据位翻转使能位 1 = 使能 0 = 禁能

6. 外部存储器容错配置寄存器 3 （MECFG3）

建议软件初始化值：硬件复位值。

Address = 0x8000-0108

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																									PR						
r																									r/w						
0																									0						

- [6:0]PR：校验位翻转。“1”为翻转，“0”为不翻转。

MECFG3 用于校验位造错。当 MECFG1 中 EWB 设置有效时，对于 8、32 位数据宽度 SRAM 和 SDRAM 整个字的写操作，则写入校验位按 MECFG3 来翻转。

位号	助记符	描述
6:0	PR	Pardata Reversal校验位翻转使能位 1 = 使能 0 = 禁能

7. 外部存储器容错配置寄存器 4 （MECFG4）

建议软件初始化值：只读不可设初值。

Address = 0x8000-010C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCTRL																															
r																															
0																														1	1

位号	助记符	描述
31:0	MCTRL	版本号

3.7 写保护寄存器

1. 写保护寄存器 1（WPR1）

建议软件初始化值：0x0。

Address = 0x8000-001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	BP	TAG															MASK														
r/w	r/w	r/w															r/w														
x	x	x															x														

- [31]：写保护使能，“1”为使能，“0”为不使能。
- [30]：block 保护模式，“1”为 block 保护模式，“0”为 segment 保护模式。
- [29:15]：比较地址。
- [14:0]：掩码地址。

位号	助记符	描述
31	EN	写保护使能，“1”为使能，“0”为不使能。
30	BP	block保护模式，“1”为block保护模式，“0”为segment保护模式。
29:15	TAG	比较地址。
14:0	MASK	掩码地址。

2. 写保护寄存器 2（WPR2）

建议软件初始化值：0x0。

Address = 0x8000-0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	BP	TAG															MASK														
r/w	r/w	r/w															r/w														
x	x	x															x														

- [31]：写保护使能，“1”为使能，“0”为不使能。
- [30]：block 保护模式，“1”为 block 保护模式，“0”为 segment 保护模式。
- [29:15]：比较地址。
- [14:0]：掩码地址。

位号	助记符	描述
31	EN	写保护使能，“1”为使能，“0”为不使能。
30	BP	block保护模式，“1”为block保护模式，“0”为segment保护模式。
29:15	TAG	比较地址。
14:0	MASK	掩码地址。

3.8 定时器和看门狗寄存器

1. 定时器 0 计数寄存器

建议软件初始化值：0xffffff（必须根据板级实际情况赋值）。

Address = 0x8000-0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TIMERCNT0																							
r								r/w																							
0								x																							

位号	助记符	描述
23:0	TIMERCNT0	定时器 0 当前计数值，预分频器下溢时，该值减 1。可读可写，复位初值为 X

2. 定时器 0 重载寄存器

建议软件初始化值：0xffffff（必须根据板级实际情况赋值）。

Address = 0x8000-0044

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TIMERLOAD0																							
r								r/w																							
0								x																							

位号	助记符	描述
23:0	TIMERLOAD0	定时器 0 重新加载值。如果在控制寄存器中设置了 RL0 位，那么在定时器溢出后，将自动加载该值。可读可写，复位初值为 X

3. 定时器 0 控制寄存器

建议软件初始化值：0x0。

Address = 0x8000-0048

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																											WTDEN	保留	LD0	RL0	EN0
r																											r/w	r	r/w	r/w	r/w
0																											x	0	x	x	0

位号	助记符	描述
4	WTDEN	看门狗使能位。置 1 时使能看门狗，置 0 时则关闭看门狗。可读可写，复位初值为 X
2	LD0	定时器 0 重载位。此位置“1”时，定时器 0 计数寄存器将加载定时器 0 重载寄存器的值，并继续计数。在该位为“1”期间，定时器的溢出将不会产生中断。该位在设置“1”后的下一个周期将自动清“0”。复位初值为 X
1	RL0	此位的值决定了定时器在溢出后是继续进行还是停止。如果该位为“1”，那么在定时器 0 溢出后，将重新装载重载寄存器中的重载值，继续下一次定时；如果该位为“0”，在溢出后，定时器

		0 将停止。复位初值为 X
0	EN0	定时器 0 使能位。此位为“1”时，定时器使能。复位初值为“0”

4. 看门狗计数值寄存器

建议软件初始化值：硬件复位值。

Address = 0x8000-004C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								WDCNT																							
r								r/w																							
0								FFFFFF																							

位号	助记符	描述
23:0	WDCNT	看门狗当前计数值，预分频器下溢时，该值减 1。该位可读可写。复位初值为全“1”

5. 定时器 1 计数寄存器

建议软件初始化值：0xffffff（必须根据板级实际情况赋值）。

Address = 0x8000-0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TIMERCNT1																							
r								r/w																							
0								x																							

位号	助记符	描述
23:0	TIMERCNT1	定时器 1 当前计数值，预分频器下溢时，该值减 1。可读可写，复位初值为 X

6. 定时器 1 重载寄存器

建议软件初始化值：0xffffff（必须根据板级实际情况赋值）。

Address = 0x8000-0054

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TIMERLOAD1																							
r								r/w																							
0								x																							

位号	助记符	描述
23:0	TIMERLOAD1	定时器 1 重新加载值。如果在控制寄存器中设置了 RL1 位，那么在定时器溢出后，将自动加载该值。可读可写，复位初值为 X

7. 定时器 1 控制寄存器

建议软件初始化值：0x0。

Address = 0x8000-0058

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																													LD1	RL1	EN1
r																													r/w	r/w	r/w
0																													x	x	0

位号	助记符	描述
2	LD1	定时器 1 重载位。此位置“1”时，定时器 1 计数寄存器将加载定时器 1 重载寄存器的值，并继续计数。在该位为“1”期间，定时器的溢出将不会产生中断。该位在设置“1”后的下一个周期将自动清“0”。复位初值为 X
1	RL1	此位的值决定了定时器在溢出后是继续进行还是停止。如果该位为“1”，那么在定时器 1 溢出后，将重新装载重载寄存器中的重载值，继续下一次定时；如果该位为“0”，在溢出后，定时器 1 将停止。复位初值为 X
0	EN1	定时器 1 使能位。此位为“1”时，定时器使能。复位初值为“0”

8. 定时器预分频计数寄存器

建议软件初始化值：0x0。

Address = 0x8000-0060

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																						SCALERCNT									
r																						r/w									
0																						0x0									

位号	助记符	描述
9:0	SCALERCNT	预分频当前计数值位。该值每个时钟周期减“1”，在溢出后，自动重新加载预分频重载值寄存器中的重载值。有效的预分频的周期为（SCALERCNT+1）个系统时钟周期。该位可读可写，复位初值为全‘0’。

9. 定时器预分频重载寄存器

建议软件初始化值：0x3ff。

Address = 0x8000-0064

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																						SCALERLOAD									
r																						r/w									
0																						0x3FF									

位号	助记符	描述
9:0	SCALERLOAD	预分频重载值。当预分频计数器溢出后，自动加载该值。可读可写。复位初值为‘0x3ff’。

3.9 UART 寄存器

1. UART1 数据寄存器

建议软件初始化值：0x0。

Address = 0x8000-0070

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																								RTD							
r																								r/w							
0																								x							

位号	助记符	描述
7:0	RTD	接收保持寄存器（读访问）/发送保持寄存器（写访问）

2. UART1 状态寄存器

建议软件初始化值：硬件复位值。

Address = 0x8000-0074

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																									FE	PE	OV	BR	TH	TS	DR
r																									r/w	r/w	r/w	r/w	r	r	r
0																									0	0	0	0	1	1	0

位号	助记符	描述
6	FE	‘1’表示检测到帧错误
5	PE	‘1’表示检测到奇偶校验错误
4	OV	‘1’表示由于数据溢出而丢失一个或更多的字符
3	BR	‘1’表示接收到 BREAK
2	TH	‘1’表示发送保持寄存器为空
1	TS	‘1’表示发送移位寄存器为空
0	DR	‘1’表示可以在接收保持寄存器中获取新数据

3. UART1 控制寄存器

建议软件初始化值：0x0。

Address = 0x8000-0078

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																							EC	LB	FL	PE	PS	TI	RI	TE	RE
r																							r/fw	r/fw	r/fw	r/fw	r/fw	r/fw	r/fw	r/fw	r/fw
0																							0	0	0	x	x	x	x	0	0

位号	助记符	描述
8	EC	外部时钟使能，若为‘1’，UART 分频脉冲将从 PIO[3]管脚获取
7	LB	回环使能，若为‘1’，将进入回环模式
6	FL	流控使能，若为‘1’，可以使用 ctsn/rtsn 进行流控制
5	PE	奇偶校验使能，若为‘1’，允许进行奇偶校验
4	PS	奇偶校验选择，0 偶校验，1 奇校验
3	TI	发送中断使能，若为‘1’，允许产生发送中断
2	RI	接收中断使能，若为‘1’，允许产生接收中断
1	TE	发送使能，若为‘1’，可以发送数据
0	RE	接收使能，若为‘1’，可以接收数据

4. UART1 分频器重载寄存器

建议软件初始化值：0x0。

Address = 0x8000-007C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																				**SRV**											
r																				**r/w**											
0																				**x**											

位号	助记符	描述
11:0	SRV	分频器重载值。 重载值 = （（系统时钟*10）/（波特率*8）-5）/10。

5. UART2 数据寄存器

建议软件初始化值：0x0。

Address = 0x8000-0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																								RTD							
r																								r/w							
0																								x							

位号	助记符	描述
7:0	RTD	接收保持寄存器（读访问）/发送保持寄存器（写访问）

6. UART2 状态寄存器

建议软件初始化值：硬件复位值。

Address = 0x8000-0084

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																									FE	PE	OV	BR	TH	TS	DR
r																									r/w	r/w	r/w	r/w	r	r	r
0																									0	0	0	0	1	1	0

位号	助记符	描述
6	FE	‘1’表示检测到帧错误
5	PE	‘1’表示检测到奇偶校验错误
4	OV	‘1’表示由于数据溢出而丢失一个或更多的字符
3	BR	‘1’表示接收到 BREAK
2	TH	‘1’表示发送保持寄存器为空
1	TS	‘1’表示发送移位寄存器为空
0	DR	‘1’表示可以在接收保持寄存器中获取新数据

7. UART2 控制寄存器

建议软件初始化值：0x0。

Address = 0x8000-0088

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																							EC	LB	FL	PE	PS	TI	RI	TE	RE
r																							r/tw	r/tw	r/tw	r/tw	r/tw	r/tw	r/tw	r/tw	r/tw
0																							0	0	0	x	x	x	x	0	0

位号	助记符	描述
8	EC	外部时钟使能，若为‘1’，UART 分频脉冲将从 PIO[3]管脚获取
7	LB	回环使能，若为‘1’，将进入回环模式
6	FL	流控使能，若为‘1’，可以使用 ctsn/rtsn 进行流控制
5	PE	奇偶校验使能，若为‘1’，允许进行奇偶校验
4	PS	奇偶校验选择，0 偶校验，1 奇校验
3	TI	发送中断使能，若为‘1’，允许产生发送中断
2	RI	接收中断使能，若为‘1’，允许产生接收中断
1	TE	发送使能，若为‘1’，可以发送数据
0	RE	接收使能，若为‘1’，可以接收数据

8. UART2 分频器重载寄存器

建议软件初始化值：0x0。

Address = 0x8000-008C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																				**SRV**											
r																				**r/w**											
0																				**x**											

位号	助记符	描述
11:0	SRV	分频器重载值。 重载值= （（系统时钟*10）/（波特率*8）-5）/10。

9. UART3 数据寄存器

建议软件初始化值：0x0。

Address = 0x8000-00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																								RTD							
r																								r/w							
0																								x							

位号	助记符	描述
7:0	RTD	接收保持寄存器（读访问）/发送保持寄存器（写访问）

10. UART3 状态寄存器

建议软件初始化值：硬件复位值。

Address = 0x8000-00E4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																									FE	PE	OV	BR	TH	TS	DR
r																									r/w	r/w	r/w	r/w	r	r	r
0																									0	0	0	0	1	1	0

位号	助记符	描述
6	FE	‘1’表示检测到帧错误
5	PE	‘1’表示检测到奇偶校验错误
4	OV	‘1’表示由于数据溢出而丢失一个或更多的字符
3	BR	‘1’表示接收到 BREAK
2	TH	‘1’表示发送保持寄存器为空
1	TS	‘1’表示发送移位寄存器为空
0	DR	‘1’表示可以在接收保持寄存器中获取新数据

11. UART3 控制寄存器

建议软件初始化值：0x0。

Address = 0x8000-00E8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																							EC	LB	FL	PE	PS	TI	RI	TE	RE
r																							r/tw	r/tw	r/tw	r/tw	r/tw	r/tw	r/tw	r/tw	r/tw
0																							0	0	0	x	x	x	x	0	0

位号	助记符	描述
8	EC	外部时钟使能，若为‘1’，UART 分频脉冲将从 PIO[3]管脚获取
7	LB	回环使能，若为‘1’，将进入回环模式
6	FL	流控使能，若为‘1’，可以使用 ctsn/rtsn 进行流控制
5	PE	奇偶校验使能，若为‘1’，允许进行奇偶校验
4	PS	奇偶校验选择，0 偶校验，1 奇校验
3	TI	发送中断使能，若为‘1’，允许产生发送中断
2	RI	接收中断使能，若为‘1’，允许产生接收中断
1	TE	发送使能，若为‘1’，可以发送数据
0	RE	接收使能，若为‘1’，可以接收数据

12. UART3 分频器重载寄存器

建议软件初始化值：0x0。

Address = 0x8000-00EC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																				**SRV**											
r																				**r/w**											
0																				**x**											

位号	助记符	描述
11:0	SRV	分频器重载值。

		重载值 = （（系统时钟*10）/（波特率*8）-5）/10。

3.10 中断寄存器

1. 中断级别/屏蔽寄存器

建议初始值：0x0 。

Address = 0x8000-0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILEVEL															保留	IMASK															保留
r/w															r	r/w															r
x															x	0															x

位号	助记符	描述
31:17	LEVEL[15:1]	可读写寄存器，初值不定，中断优先级的 15 位对应 15 路中断，每位可配置为 1 或 0，指示中断所属的优先级，级别为 1 的中断，高于所有级别为 0 的中断；同级别的中断，依据中断号排列优先级别，中断号越大，优先级越高。复位初值不定，需要初始化。
15:1	IMASK[15:1]	可读写寄存器，初值为 0，中断屏蔽位[15:1]，若某位为 1，表示中断可响应，0 表示中断被屏蔽。中断屏蔽位复位初值位全 0，即使对应的中断被触发，也不会被响应。

2. 中断请求寄存器

建议初始值：0x0。

Address = 0x8000-0094

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																IPEND															保留
r																r/w															r
x																x															x

位号	助记符	描述
15:1	IPEND[15:1]	可读写寄存器，初值不定，若相应中断号的请求位是 1，表明存在一个对应的中断请求。当中断请求被处理后，相应的中断位会被清 0。

3. 中断强制寄存器

建议初始值：0x0。

Address = 0x8000-0098

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																IFORCE															保留
r																r/w															r
x																x															x

位号	助记符	描述
15:1	IFORCE[15:1]	可读写寄存器，初值不定，将相应中断位写 1，强制产生该位中断。当中断请求被处理后，相应的中断位会被清 0。

4. 中断清除寄存器

建议初始值：0xffff。

Address = 0x8000-009C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																ICLEAR															保留
r																w															r
x																x															x

位号	助记符	描述
15:1	ICLEAR[15:1]	中断清除域，只写，置 1 将清除中断请求寄存器中的相应位，置 0 无效，读出无效。

3.11 GPIO 寄存器

1. GPIO 数据寄存器

Address = 0x8000-00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEDDAT								LOWDAT								IODATA															
r/w								r/w								r/w															
x								x								x															

位号	助记符	描述
31:24	*MEDDAT*	与数据总线D[15:8]连接
23:16	*LOWDAT*	与数据总线D[7:0]连接
15:0	*IODATA*	I/O 端口数据

2. GPIO 方向寄存器

Address = 0x8000-00A4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														MEDDIR	LOWDIR	IODIR[15：0]															
r														r/w	r/w	r/w															
0														0	0	0															

位号	助记符	描述
17	*MEDDIR*	表示D[15:8]的方向 1 = 输出 0 = 输入
16	*LOWDIR*	表示D[7:0]的方向 1 = 输出 0 = 输入
15:0	*IODIR*	I/O端口[15:0]的方向 1 = 输出 0 = 输入

3. GPIO 中断控制寄存器

Address = 0x8000-00A8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	LE	PL	ISEL					EN	LE	PL	ISEL					EN	LE	PL	ISEL					EN	LE	PL	ISEL				
r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w				
0	x	x	x					0	x	x	x					0	x	x	x					0	x	x	x				

位号	助记符	描述
31 23 15 7	*EN*	使能信号 1 = 相应中断使能 0 = 相应中断屏蔽
30 22 14 6	*LE*	触发方式 1 = 边沿触发 0 = 电平触发
29 21 13 5	*PL*	极性 1 = 相应中断上升沿有效 0 = 相应中断下降沿有效
28...24 20...16 12...8 4...0	*ISEL*	I/O端口选择 确定哪个I/O端口（31：0）产生GPIO中断n

3.12 PCI 寄存器

1. PCI 总线复位寄存器（PCI_RESET）

建议初始值：不用赋初值。

Address = 0x8000-00D0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																												RESET			
r																												w			
x																												xxxx			

- [31:4]：保留。
- RESET[3:0]：PCI 总线软件复位寄存器。在主桥模式下，写 0x5 产生 PCI 软件复位，复位 PCI 接口 PCI 时域的寄存器及 PCI 总线上的设备，写其他值无效；在从桥模式下，该寄存器无效。

位号	助记符	描述
3:0	RESET	PCI接口复位，写0x05产生PCI总线复位。

2. DMA 写操作 PCI 总线起始地址寄存器（WDMA_PCI_ADDR）

建议初始值：硬件复位值。

Address = 0xC000-0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
r/w																															
0																															

- ADDR[31:0]：DMA 写操作在 PCI 总线上起始地址，在 DMA 写操作过程中，DMA 控制器将数据写入该寄存器配置的地址，随数据传输该寄存器值自动增加。建议使用时，寄存器后两位(ADDR[1:0]）设为 0。

位号	助记符	描述
31:0	ADDR	DMA写操作在PCI总线上起始地址，在DMA写操作过程中，此寄存器值随数据传输自动增加。建议使用时，寄存器后两位（ADDR（1:0））设为0。

3. DMA 写操作 AHB 总线起始地址寄存器（WDMA_AHB_ADDR）

建议初始值：硬件复位值 。

Address = 0xC000-0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
r/w																															
0																															

- ADDR[31:0]：DMA 写操作在 AHB 总线上起始地址，在 DMA 写操作过程中，DMA 控制器从该寄存器配置的地址读取数据，随着数据传输该寄存器值自动增加。建议使用时，寄存器后两位（ADDR[1:0]）设为 0。

位号	助记符	描述
31:0	ADDR	DMA写操作在AHB总线上起始地址，在DMA写操作过程中，随着数据传输该寄存器值自动增加。建议使用时，寄存器后两位（ADDR[1:0]）设为0。

4. DMA 写操作控制寄存器（WDMA_CONTROL）

建议初始值：硬件复位值。

Address = 0xC000-0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSFER SIZE																								COMMAND				PCI INTERRUPT	保留	STOP TRANSFER	START TRANSFER
r/w																								r/w				r/w	r	r/w	r/w
0																								0000				0	0	0	0

- TRANSFER SIZE[31:8]：DMA 写传操作时传递的字节数，范围为 1-0xFFFFFF 字节。建议进行 DMA 操作时，以整字（4 个字节）倍数长度进行数据传输。
- COMMAND[7:4]：DMA 写操作在 PCI 总线上传送命令。按正文表 4-9-4 写 DMA 设置 PCI 命令，设置其他 PCI 命令无效。
- PCI INTERRUPT[3]：在从桥模式下，该寄存器配置为 1，在 DMA 写操作正常完成时产生中断请求，DMA 写传输异常退出时不产生中断请求；该寄存器配置为 0，在 DMA 写操作正常完成时不发出中断请求。在主桥模式下，该寄存器无效。
- STOP TRANSFER[1]：DMA 写操作停止位，写 1 停止当前的 DMA 写传输，写 0 无效。该寄存器写 1 后自动清 0，即读出值为 0。
- START TRANSFER[0]：DMA 写操作起动位，写 1 发起 DMA 写操作，写 0 无效。DMA 传输完成后此位自动清 0，因此可通过此位判断 DMA 写传输是否完成。

位号	助记符	描述
31:8	TRANSFER SIZE	DMA写操作时传递的字节数，范围为1-0xFFFFFF字节。建议进行DMA操作时，以整字（4个字节）倍数长度进行数据传输。
7:4	COMMAND	DMA写操作时在PCI总线传送命令。按表A-61写DMA中设置PCI命令，设置其他PCI命令无效。
3	PCI INTERRUPT	从桥模式下，配置为1，DMA写操作正常完成时产生中断请求，DMA异常退出不产生中断请求。配置为0时，DMA写操作正常完成时不产生中断请求。在主桥模式下，该寄存器无效。
1	STOP TRANSFER	DMA传输停止位，写1停止当前的DMA写传输，写0无效。该寄存器写1后自动清0，即读出值为0。
0	START TRANSFER	DMA写操作起动位，写1发起DMA写操作，写0无效，DMA传输完成后此位自动清0，因此可通过此位判断DMA写传输是否完成。

5. DMA 读操作 PCI 总线起始地址寄存器（RDMA_PCI_ADDR）

建议初始值：硬件复位值。

Address = 0xC000-0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
r/w																															
0																															

- ADDR[31:0]：DMA 读操作在 PCI 总线上起始地址，在 DMA 读操作过程中，DMA 控制器从该寄存器配置的地址读出数据，随数据传输该寄存器值自动增加。建议使用时，寄存器后两位（ADDR[1:0]）设为 0。

位号	助记符	描述
31:0	ADDR	DMA读操作在AHB总线上的起始地址，在DMA读操作过程中，此寄存器值随着数据传输自动增加。建议使用时，寄存器后两位(ADDR(1:0))设为0。

6. DMA 读操作 AHB 总线起始地址寄存器（RDMA_AHB_ADDR）

建议初始值：硬件复位值。

Address = 0xC000-0024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															
r/w																															
0																															

- ADDR[31:0]：DMA 读操作在 AHB 总线上起始地址，在 DMA 读操作过程中，DMA 控制器将数据写入该寄存器配置的地址，随着数据传输该寄存器值自动增加。建议使用时，寄存器后两位（ADDR[1:0]）设为 0。

位号	助记符	描述
31:0	ADDR	DMA读操作在AHB总线上起始地址，在DMA读操作过程中，此寄存器值随着数据传输自动增加。建议使用时，寄存器后两位（ADDR（1:0））设为0。

7. DMA 读传输控制寄存器（RDMA_CONTROL）

建议初始值：硬件复位值。

Address = 0xC000-0028

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSFER SIZE																								COMMAND				PCI INTERRUPT	保留	STOP TRANSFER	START TRANSFER
r/w																								r/w				r/w	r	r/w	r/w
0																								0000				0	0	0	0

- TRANSFER SIZE[31:8]：DMA 读操作时传递的字节数，范围为 1-0xFFFFFF 字节。建议进行 DMA 操作时，以整字（4 个字节）倍数长度进行数据传输。
- COMMAND[7:4]： DMA 读操作时在 PCI 总线上传送命令。按正文表 4-9-4 读 DMA 设置 PCI 命令，设置其他 PCI 命令无效。
- PCI INTERRUPT[3]：从桥模式下，该寄存器配置为 1，在 DMA 读操作正常完成时产生中断请求，DMA 读操作异常退出时不产生中断请求；该寄存器配置为 0，在 DMA 写操作正常完成时不发出中断请求。在主桥模式下，该寄存器无效。
- STOP TRANSFER[1]：DMA 读传输停止位，写 1 停止当前的 DMA 读传输，写 0 无效。该寄存器写 1 后自动清 0，即读出值为 0。
- START TRANSFER[0]： DMA 读操作起动位，写 1 发起 DMA 读传输，写 0 无效，DMA 传输完成后此位自动清 0，因此可通过此位判断 DMA 读传输是否完成。

位号	助记符	描述
31:8	TRANSFER SIZE	DMA读操作时传递的字节数，范围为1-0xFFFFFF字节。建议进行DMA操作时，以整字（4个字节）倍数长度进行数据传输。
7:4	COMMAND	DMA读操作时在PCI总线传送命令。按表A-61读DMA中设置PCI命令，设置其他PCI命令无效。
3	PCI INTERRUPT	从桥模式下，配置为1，DMA读操作正常完成时产生中断请求，DMA异常退出不产生中断请求。配置为0时，DMA读操作正常完成时不产生中断请求。在主桥模式下，该寄存器无效。
1	STOP TRANSFER	DMA传输停止位，写1停止当前的DMA读传输，写0无效。该寄存器写1后自动清0，即读出值为0。
0	START TRANSFER	DMA读操作起动位，写1发起DMA读传输，写0无效，DMA传输完成后此位自动清0，因此可通过此位判断DMA读传输是否完成。

8. CPU 中断掩码寄存器(CPU_IMASK)

建议初始值：硬件复位值。

Address = 0xC000-0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PCI SYSTEM ERROR	PCI INTERRUPT				保留				PCI DOORBELL				保留	AHB- PCI WINDOW			READ DMA				保留	PCI- AHB WINDOW			WRITE DMA			
																	DISCARD	FETCH ERROR	POST ERROR	AHB ERROR	PARITY ERROR	ABORT	END		DISCARD	FETCH ERROR	POST ERROR	AHB ERROR	PARITY ERROR	ABORT	END
r/w			r/w	r\w				r/w				r/w				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w
000			0	0000				0000				0000				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- 保留位[31:30]：在主桥模式下，该寄存器可读可写；在从桥模式下，该寄存器只读，读出值为 0。
- 保留位[29]：在从桥模式下，该寄存器可读可写；在主桥模式下，该寄存器只读，读出值为 0。
- PCI SYSTEM ERROR[28]，PCI 系统错误，表示在 PCI 总线传输地址周期出现奇偶校验错，1=中断使能，0=中断屏蔽。在主桥模式下，该寄存器可读可写。在从桥模式下，该寄存器只读，读出值为 0。
- PCI INTERRUPT[27:24]：PCI 中断，外部 PCI 设备通过中断管脚（pci_inta_n，pci_intb_n，pci_intc_n，pci_intd_n）向处理器发起中断请求，1=中断使能，0=中断屏蔽。在主桥模式下，该寄存器可读可写；在从桥模式下，该寄存器无效，读出值为 0。
- PCI DOORBELL[19:16]：外部 PCI 设备通过 PCI DOORBELL 位向处理器发起中断请求，1=中断使能，0=中断屏蔽。在从桥模式下，该寄存器可读可写；在主桥模式下，该寄存器无效，读出值为 0。
- AHB-PCI WINDOW DISCARD[14]：AHB-PCI 窗口丢弃，表示处理器通过 AHB-PCI 窗口发起读操作，超过 AHB-PCI 窗口定时寄存器（AHBPCI_TIMER）设置的时钟周期个数，操作仍未完成，此次操作被丢弃。1=中断使能，0=中断屏蔽。
- AHB-PCI WINDOW FETCH ERROR[13]：AHB-PCI 窗口取值错误，表示通过 AHB-PCI 窗口传输数据时，从外部 PCI 设备读取数据时发生错误。1=中断使能，0=中断屏蔽。
- AHB-PCI WINDOW POST ERROR[12]：AHB-PCI 窗口存储错误，表示通过 AHB-PCI 窗口传输数据时，数据写入外部 PCI 设备时发生错误。1=中断使能，0=中断屏蔽。
- READ DMA AHB ERROR[11]：DMA 读操作 AHB 错误，表示 DMA 控制器向 AHB 地址写

入数据时发生错误。1=中断使能，0=中断屏蔽。

- READ DMA PARITY ERROR[10]：DMA 读操作数据奇偶错，表示 DMA 读传输过程中，从外部 PCI 总线设备读取数据时，发生了奇偶校验错。1=中断使能，0=中断屏蔽。
- READ DMA ABORT[9]：DMA 读操作异常终止，表示在 DMA 读传输过程中，PCI 总线上发生了目标设备终止或主设备终止。1=中断使能，0=中断屏蔽。
- READ DMA END[8]：DMA 读操作结束，表示 DMA 读操作正常完成或当前正在进行的 DMA 读操作被停止。1=中断使能，0=中断屏蔽。
- PCI-AHB WINDOW DISCARD[6]：PCI-AHB 窗口丢弃，表示外部 PCI 总线设备通过 PCI –AHB 窗口发起读操作，超过 PCI-AHB 窗口定时寄存器[PCIAHB_TIMER）设置的时钟周期个数，操作仍未完成，此次操作被丢弃。1=中断使能，0=中断屏蔽。
- PCI-AHB WINDOW FETCH ERROR[5]：PCI-AHB 窗口取值错误，表示通过 PCI-AHB 窗口传输数据时，从 AHB 地址空间读取数据时发生错误。1=中断使能，0=中断屏蔽。
- PCI-AHB WINDOW POST ERROR[4]：PCI-AHB 窗口存储错误，表示通过 PCI-AHB 窗口传输数据时，数据写入 AHB 地址空间时发生错误。1=中断使能，0=中断屏蔽。
- WRITE DMA AHB ERROR[3]：DMA 写操作 AHB 错误，表示 DMA 控制器从 AHB 地址空间读数据时发生错误。1=中断使能，0=中断屏蔽。
- WRITE DMA PARITY ERROR[2]：DMA 写操作数据奇偶错，表示 DMA 写传输过程中，向外部 PCI 总线设备写入数据时，发生了奇偶校验错。1=中断使能，0=中断屏蔽。
- WRITE DMA ABORT[1]：DMA 写操作异常终止，表示在 DMA 写传输过程中，PCI 总线上发生了目标设备终止或主设备终止。1=中断使能，0=中断屏蔽。
- WRITE DMA END[0]：DMA 写操作结束，表示 DMA 写操作正常结束或当前正在进行的 DMA 写操作被停止。1=中断使能，0=中断屏蔽。

位号	助记符	描述
31:30	保留	在主桥模式下，该寄存器可读可写；在从桥模式下，该寄存器只读，读出值为0。
29	保留	在从桥模式下，该寄存器可读可写；在主桥模式下，该寄存器只读，读出值为0。
28	PCI SYSTEM ERROR	PCI系统错误，1=中断使能，0=中断屏蔽。在主桥模式下，该寄存器可读可写。在从桥模式下，该寄存器无效，只读，读出值为0。
27:24	PCI INTERRUPT	PCI中断，PCI设备向处理器发起中断请求，1=中断使能，0=中断屏蔽。在主桥模式下，该寄存器可读可写；在从桥模式下，该寄存器无效，只读，读出值为0。
19:16	PCI DOORBELL	PCI设备通过PCI DOORBELL位向处理器发起中断请求，1=中断使能，0=中断屏蔽。在从桥模式下，该寄存器可读可写；在主桥模式下，该寄存器无效，只读，读出值为0。
14	AHB-PCI WINDOW DISCARD	AHB-PCI窗口丢弃，1=中断使能，0=中断屏蔽
13	AHB-PCI WINDOW FETCH ERROR	AHB-PCI窗口取值错误，1=中断使能，0=中断屏蔽
12	AHB-PCI WINDOW POST ERROR	AHB-PCI窗口存储错误，1=中断使能，0=中断屏蔽
11	READ DMA AHB ERROR	DMA读操作AHB错误，1=中断使能，0=中断屏蔽
10	READ DMA PARITY ERROR	DMA读操作数据奇偶校验错误位，1=中断使能，0=中断屏蔽
9	READ DMA ABORT	DMA读操作异常终止，1=中断使能，0=中断屏蔽
8	READ DMA END	DMA读操作结束，1=中断使能，0=中断屏蔽
6	PCI-AHB WINDOW DISCARD	PCI-ABH窗口丢弃，1=中断使能，0=中断屏蔽
5	PCI-AHB WINDOW FETCH ERROR	PCI-AHB窗口取值错误，1=中断使能，0=中断屏蔽
4	PCI-AHB WINDOW POST ERROR	PCI-AHB窗口存储错误，1=中断使能，0=中断屏蔽
3	WRITE DMA AHB ERROR	DMA写操作AHB错误，1=中断使能，0=中断屏蔽
2	WRITE DMA PARITY ERROR	DMA写操作数据奇偶校验错误，1=中断使能，0=中断屏蔽
1	WRITE DMA ABORT	DMA写操作异常终止，1=中断使能，0=中断屏蔽
0	WRITE DMA END	DMA写操作结束，1=中断使能，0=中断屏蔽

9. CPU 中断状态寄存器(CPU_ISTATUS)

建议初始值：硬件复位值。

Address = 0xC000-0044

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PCI SYSTEM ERROR	PCI INTERRUPT				保留				PCI DOORBELL				保留	AHB-PCI WINDOW			READ DMA				保留	PCI-AHB WINDOW			WRITE DMA			
																	DISCARD	FETCH ERROR	POST ERROR	AHB ERROR	PARITY ERROR	ABORT	END		DISCARD	FETCH ERROR	POST ERROR	AHB ERROR	PARITY ERROR	ABORT	END
r			r/w					r				r/w				r	r/w							r	r/w						
000			000000					0000				0000				0	0							0	0						

位号	助记符	描述
28	PCI SYSTEM ERROR	在主桥模式下，1表示PCI系统错误，在PCI总线传输地址周期出现奇偶错。写1请0，写0无效。在从桥模式下，该寄存器无效，读出值为0。
27:24	PCI INTERRUPT	在主桥模式下，1表示相应PCI总线设备发起的处理器中断有效。写1相应位清0，写0无效。在从桥模式下，该寄存器无效，读出值为0。
19:16	PCI DOORBELL	在从桥模式下，1表示PCI DOORBELL相应位有效。写1相应位清0，写0无效。在主桥模式下，该寄存器无效，读出值为0。
14	AHB-PCI WINDOW DISCARD	1表示AHB-PCI窗口丢弃。写1清0，写0无效。
13	AHB-PCI WINDOW FETCH ERROR	1表示AHB-PCI窗口取值错误。写1清0，写0无效
12	AHB-PCI WINDOW POST ERROR	1表示AHB-PCI窗口存储错误。写1清0，写0无效
11	READ DMA AHB ERROR	1表示DMA读操作AHB错误。写1清0，写0无效
10	READ DMA PARITY ERROR	1表示DMA读操作数据奇偶校验错误。写1清0，写0无效
9	READ DMA ABORT	1表示DMA读操作异常终止。写1清0，写0无效
8	READ DMA END	1表示DMA读操作结束。写1清0，写0无效
6	PCI-AHB WINDOW DISCARD	1表示PCI-ABH窗口丢弃。写1清0，写0无效
5	PCI-AHB WINDOW FETCH ERROR	1表示PCI-AHB窗口取值错误。写1清0，写0无效
4	PCI-AHB WINDOW POST ERROR	1表示PCI-AHB窗口存储错误。写1清0，写0无效
3	WRITE DMA AHB ERROR	1表示DMA写操作AHB错误。写1清0，写0无效
2	WRITE DMA PARITY ERROR	1表示DMA写操作数据奇偶校验错误。写1清0，写0无效
1	WRITE DMA ABORT	1表示DMA写异常终止。写1清0，写0无效
0	WRITE DMA END	1表示DMA写结束。写1清0，写0无效

10. CPU 中断命令寄存器(CPU_ICMD)

建议初始值：硬件复位值。

Address = 0xC000-0048

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CPU DOORBELL				保留																			
r								w				r																			
0								0000				0																			

- CPU DOORBELL[23:20]：在从桥模式下，CPU 中断命令寄存器，用于人为发出一些 PCI 中断。写 1 使能，写 0 无效，读出值为 0。在主桥模式下，该寄存器无效。该寄存器与 PCI 中断掩码寄存器的[23:20]位一一对应，即 20 位对应 20 位、21 位对应 21 位、22 位对应 22 位、23 位对应 23 位。当配置 PCI 中断掩码寄存器[23:20]的某位为 1，CPU 向该寄存器的相应位写 1，则 BM3803FMGRH 通过管脚 pci_inta_n 向 PCI 总线发

出中断请求。

位号	助记符	描述
23:20	CPU DOORBELL	在从桥模式下，CPU中断命令寄存器，用于人为发出一些PCI中断，该寄存器写1使能性应位，写0无效，读出值为0。在主桥模式下，该寄存器无效，读出值为0。

11. CPU 状态及版本寄存器（CPU_VERSION）

建议初始值：硬件复位值。

Address = 0xC000-004C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																TYPE				VERSION											
r																r				r											
0																xxxx				0x380											

位号	助记符	描述
15:12	TYPE	表示PCI接口模式 1 = 32位从桥模式 2 = 32位主桥模式 管脚pci_host接低电平，为2；接高电平，为1
11:0	VERSION	PCI桥版本,由IC厂商定义

12. 不可预取范围 PCI-AHB 窗口范围控制寄存器（PCIAHB_ADDR_NP）

建议初始值：硬件复位值。

Address = 0xC000-0070

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															EN
r/w																								r							r/w
0x800000																								0							1

- ADDR[31:1]：PCI 设备通过 PCI-AHB 窗口访问处理器地址空间的基地址。在主桥和从桥模式下，该寄存器功能不同。在主桥模式下，PCI-AHB 窗口采用扁平的寻址方式，

PCI 总线读写操作的地址直接转化成内部 AHB 总线操作的地址。该寄存器定义了不可预取存储器空间（相当于定义了 PCI 基地址寄存器 3（BAR3），外部 PCI 设备通过访问该寄存器定义的地址空间，就实现了对处理器 AJ 地址空间的访问。在从桥模式下，PCI 设备通过 PCI-AHB 窗口访问处理器地址空间，地址需要进行变换。PCI 基地址寄存器 3（BAR3）由外部 PCI 主机系统进行配置，处理器通过该寄存器设置转换后的 AHB 地址，PCI-AHB 地址空间转换如下：AHB 地址 = PCIAHB_ADDR_NP + (PCI 操作地址－PCI 基地址 3（BAR3）)外部 PCI 设备访问 PCI 基地址空间 BAR3，就实现了对 AHB_ADDR_NP 定义的 AHB 址空间的访问。

- EN[0]:PCI-AHB 地址空间使能位。1=使能 PCI 总线操作到 AHB 操作的转换，0=PCI 总线操作以目标异常终止（Target Abort）结束。

位号	助记符	描述
31:8	AHB BASE ADDRESS	在从桥模式下,PCI总线操作转换地址;主桥模式下，PCI-AHB窗口基地址 此地址必须32bit对齐，可以覆盖多个AHB设备的地址空间。
0	WINDOWS ENABLE	PCI-AHB地址空间使能位 1 = 使能PCI总线操作转换 0 = 所有PCI操作以Targe-abort结束

13. PCI-AHB 窗口可预取范围控制寄存器（PCIAHB_ADDR_PF）

建议初始值：硬件复位值。

Address = 0xC000-0074

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															EN
r/w																								r							r/w
0x400000																								0							1

- ADDR[31:1]：PCI 设备通过 PCI-AHB 窗口访问处理器地址空间的基地址。在主桥和从桥模式下，该寄存器功能不同。在主桥模式下，PCI-AHB 窗口采用扁平的寻址方式，PCI 总线读写操作的地址直接转化成内部 AHB 总线操作的地址。该寄存器定义了可预取存储器空间（相当于定义了 PCI 基地址寄存器 4（BAR4），外部 PCI 设备通过访问该寄存器定义的地址空间，就实现了对处理器 AHB 地址空间的访问。在从桥模式下，

PCI 设备通过 PCI-AHB 窗口访问处理器地址空间，地址需要进行变换。PCI 基地址寄存器 4（BAR4）由外部 PCI 主机系统进行配置，处理器通过该寄存器设置转换后的 AHB 地址，PCI-AHB 地址空间转换如下：AHB 地址 = PCIAHB_ADDR_NP + (PCI 操作地址－PCI 基地址 4（BAR4）)外部 PCI 设备访问 PCI 基地址空间 BAR4，就实现了对 AHB_ADDR_PF 定义的 AHB 地址空间的访问。

- EN[0]:PCI-AHB 地址空间使能位。1=使能 PCI 总线操作到 AHB 操作的转换，0=PCI 总线操作以目标异常终止（Target Abort）结束。

位号	助记符	描述
31:8	AHB BASE ADDRESS	在从桥模式下,PCI总线操作转换地址;主桥模式下，PCI-AHB窗口基地址 此地址必须32bit对齐，可以覆盖多个AHB设备的地址空间。
0	WINDOWS ENABLE	PCI-AHB地址空间使能位 1 = 使能PCI总线操作转换 0 = 所有PCI操作以Targe-abort结束

14. PCI-AHB 窗口丢弃定时寄存器（PCIAHB_TIMER）

建议初始值：硬件复位值。

Address = 0xC000-0078

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																				DISCARD TIMER											
r																				r/w											
0																				0x100											

位号	助记符	描述
11:0	DISCARD TIMER	清空交易定时位，防止PCI总线锁死。PCI总线主设备通过PCI-AHB窗口发起读操作，超过该寄存器定义的PCI时钟周期个数操作仍未完成，此次操作丢弃。

15. AHB-PCI 窗口的丢弃定时寄存器（AHBPCI_TIMER）

建议初始值：硬件复位值。

Address = 0xC000-007C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																				DISCARD TIMER											
r																				r/w											
0																				0x100											

位号	助记符	描述
11:0	DISCARD TIMER	清空交易定时位，防止PCI总线锁死。处理器通过通过AHB-PCI窗口读PCI总线上的设备，超过该寄存器定义的系统时钟周期个数操作仍未完成，此次操作丢弃。

16. PCI 控制寄存器

建议初始值：硬件复位值。

Address = 0xC000-0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																													MasterEN	MemoryEN	BridgePCIReady
r																													r	r	r
0																													x	x	x

- [31:3]：保留。
- MasterEN[2]：主设备能力状态位，1=具备主设备能力，0=不具备。该位为 1，表示处理器能够作为主设备对 PCI 总线上的目标设备发起读写操作。在主桥模式下，该位一直为 1；在从桥模式下，该位复位后值为 0，通过 PCI 状态和命令寄存器[2]位进行配置，PCI 状态和命令寄存器[2]位配置为 1，则此位也为 1。
- MemoryEN[1]：存储器地址空间使能位，1=使能，0=禁止。主桥模式下，该位总置 1；从桥模式下，该位复位后为 0，通过 PCI 状态和命令寄存器[1]位进行配置，PCI 状态和命令寄存器[1]位配置为 1，则此位也为 1。
- Bridge PCI Ready[0]: 在主桥模式下，系统复位后此位为 0，225 个 PCI 时钟周期之后置为 1，处理器进行 PCI 总线配置访问之前必须等此位为 1；在从桥模式下，该位复位后一直为 1。

位号	助记符	描述
2	MasterEN	主设备能力使能，1=使能，0=禁止。主桥模式下，该位总置1；从桥模式下，该位复位后为0，通过PCI状态和命令寄存器（2）位，实现对此位进行配置。
1	MemoryEN	存储器地址空间使能，1=使能，0=禁止。主桥模式下，该位总置1； 从桥模式下，该位复位后为0，通过PCI状态和命令寄存器（1）位，实现对此位进行配置。
0	Bridge PCI Ready	在主桥模式下，系统复位后此位为0，2^{25}个PCI时钟周期之后置为1，处理器进行PCI总线配置访问之前必须等此位为1。从桥模式下，该位复位后一直为1。

17. PCI 设备和供应商标识寄存器（PCI_DV）

建议初始值：硬件复位值 。

Address = 0xC000-0084

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DV																															
r/w																															
0x55551556																															

位号	助记符	描述
31:0	DV	PCI设备和供应商信息位

18. PCI 子系统设备和子系统供应商配置寄存器（PCI_SUB）

建议初始值：硬件复位值。

Address = 0xC000-0088

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBSYSTEM ID																SUBSYSTEM VENDOR ID															
r/w																r/w															
0																0															

位号	助记符	描述
31:16	SUBSYSTEM ID	PCI子系统设备信息位
15:0	SUBSYSTEM VENDOR ID	PCI子系统供应商信息位

19. PCI 分类代码和版本配置寄存器（PCI_CREV）

建议初始值：硬件复位值。

Address = 0xC000-008C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLASS CODE																								REVISION ID							
r/w																								r/w							
0x040000																								0							

位号	助记符	描述
31: 8	CLASS CODE	PCI分类代码
7: 0	REVISION ID	版本信息位

20. PCI 仲裁状态寄存器（PCI_BROKEN）

建议初始值：硬件复位值。

Address = 0xC000-0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																								STATUS							
r																								x							
0																								0							

位号	助记符	描述
7:0	STATUS	在主桥模式下，该寄存器可读可写，表示PCI总线上Master设备的状态。1=表示相应的PCI Master设备被禁止，表明严重的系统错误。写1清0，读出值为0，写0无效。 在从桥模式下，该寄存器无效，只读，读出值为0。

21. PCI-AHB 窗口不可预取空间大小配置寄存器（PCIAHB_SIZ_NP）

建议初始值：硬件复位值。

Address = 0xC000-0094

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIZ																															
r/w																															
0xFFFC00																								0							

位号	助记符	描述
31:0	SIZ	设置不可预取空间大小。 低8位恒为0，最小范围为0x100字节。设定空间大小时，低位应设置成全0，高位设置成全1，如0的个数为n，则空间大小为2^n字节。

22. PCI-AHB 窗口可预取空间大小配置寄存器（PCIAHB_SIZ_PF）

建议初始值：硬件复位值。

Address = 0xC000-0098

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIZ																															
r/w																															
0xFFFC00																								0							

位号	助记符	描述
31:0	SIZ	表示可预取空间大小，低8位恒为0，最小范围为0x100字节。设定空间大小时，低位应设置成全0，高位设置成全1，如0的个数为n，则空间大小为2^n字节。

23. PCI 设备和供应商标识寄存器（PCI_DV）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEVICE ID																VENDOR ID															
r																r															
0x5555																0x1556															

位号	助记符	描述
31:16	DEVICE ID	供应商标识。该寄存器值由寄存器PCI设备和供应商标识寄存器（0xC000-0084）设定。
15:0	VENDOR ID	设备标识该寄存器值由寄存器PCI设备和供应商标识寄存器（0xC000-0084）设定。

24. PCI 状态和命令寄存器（PCI_SCR）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STATUS																COMMAND															
DETECTED PARITY ERROR	SIGNALED SYSTEM ERROR	RECEIVED MASTER ABORT	RECEIVED TARGET ABORT	SINGALED TARGET ABORT	DEVSEL TIMING		MASTER DATA PARITY ERROR	FAST BACK-TO-BACK CAPABLE	保留	保留	保留	INTERRUPT STATUS	保留			保留					INTERRUPT DISABLE	保留	SERR ENABLE	保留	PARITY ERROR RESPONSE	保留	保留	保留	BUS MASTER	MEMORY SPACE	保留
r	r	r	r	r	r		r	r	r	r	r	r	r			r					r/w	r/w	r/w	r	r/w	r	r	r	r/w	r/w	r/w
0	0	0	0	0	01		0	0	0	1	0	0	000			00000					0	0	0	0	0	0	0	0	0	0	0

- DETECTED PARITY ERROR[31]：奇偶校验错。当设备监测到奇偶校验错时，将此位置 1。1=监测到奇偶校验错，0=传输正常。该位写 1 清 0，写 0 无效。
- SINGALED SYSTEM ERROR[30]：发出系统错误。当设备驱动 SERR 有效时，将此位置 1。1=出现系统错误，0=传输正常。该位写 1 清 0，写 0 无效。
- RECEIVED MASTER ABORT[29]：收到主设备异常终止。当总线主设备发起的交易以主设备异常终止结束，将此位置 1。1=主设备异常终止，0=传输正常该位写 1 清 0，写 0 无效。
- RECEIVED TARGET ABORT[28]：收到目标设备异常终止，总线主设备发起总线传输被目标设备异常终止时，总线主设备必须将此位置 1。1=收到目标异常终止，0=传输正常。该位写 1 清 0，写 0 无效。
- SINGALED TARGET ABORT[27]：目标设备异常终止。目标设备异常终止一个 PCI 总线传输，目标设备必须将此位置为 1。1=目标异常终止，0=传输正常。该位写 1 清 0，写 0 无效。
- DEVESEL TIMING[26:25]：设备选择定时，表示作为目标设备有效 DEVSEL 的速度。该寄存器只读，读出值为 0x1，表示中速。
- MASTER DATA PARITY ERROR[24）：主设备奇偶校验错。1=总线出现奇偶校验错，0=未出现奇偶校验错。该位写 1 后值为 0，写 0 对值无影响。满足下面三个条件该寄存器置 1：

  a） **桥作为总线主设备，是 PCI 总线访问的发起者。**

  b） **命令寄存器 PCI_SCR bit[6] PARITY ERROR RESPONSE 设置为 1。**

c） 通过自行设置错 PERR（在读操作的时候）有效或者监测到目标设备设置（在写的时候）PERR 有效。

- FAST BACK-TO-BACK ENABLE[23]：目标设备应答背靠背传输的能力。1=具备，0=不具备。
- INTERRUPT STATUS[19)：中断状态。1=PCI 中断信号有效，0=PCI 中断信号无效。
- INTERRUPT DISABLE[10]：中断输出禁止位。1=禁止输出 PCI 中断信号，0=允许 PCI 中断信号输出。
- SERR ENABLE[8]：SERR 信号驱动使能。1=允许驱动 SERR 信号线，0=不能驱动。
- PARITY ERROR RESPONSE[6]：奇偶校验错误响应。1=使能奇偶校验检查，0=忽略奇偶错误。
- BUS MASTER[2]：总线主控能力。1=允许作为主设备操作 PCI 总线，0=禁止作为主设备操作 PCI 总线。
- MEMEORY SPACE[1]：存储器空间访问应答。1=允许访问，0=禁止访问。

位号	助记符	描述
31	DETECTED PARITY ERROR	奇偶校验错。当设备监测到奇偶校验错时，将此位置1。1=监测到奇偶校验错，0=传输正常。该位写1清0，写0无效。
30	SINGALED SYSTEM ERROR	发出系统错误。当设备驱动SERR有效时，将此位置1。1=出现系统错误，0=传输正常。该位写1清0，写0无效。
29	RECEIVED MASTER ABORT	主设备异常终止。当总线主设备发起的交易以主设备异常终止结束，将此位置1。1=主设备异常终止，0=传输正常。该位写1清0，写0无效。
28	RECEIVED TARGET ABORT	收到目标异常终止，总线主设备发起总线传输被目标设备异常终止时，总线主设备必须将此位置1。1=收到目标异常终止，0=传输正常。该位写1清0，写0无效。
27	SINGALED TARGET ABORT	目标异常终止。目标设备异常终止一个PCI总线传输，目标设备必须将此位置为1。1=目标异常终止，0=传输正常。该位写1清0，写0无效。
26:25	DEVESEL TIMING	设备选择定时，表示作为目标设备有效DEVSEL的速度。该寄存器只读，读出值为0x1，表示中速。
24	MASTER DATA PARITY ERROR	主设备奇偶校验错。1=总线出现奇偶校验错，0=未出现奇偶校验错。该位写1清0，写0无效。
23	FAST BACK-TO-BACK ENABLE	目标设备应答背靠背传输的能力。1=具备，0=不具备。
19	INTERRUPT STATUS	中断状态。1=PCI中断信号有效，0=PCI中断信号无效。
10	INTERRUPT DISABLE	中断输出禁止位。1=禁止输出PCI中断信号，0=允许PCI中断信号输出。
8	SERR ENABLE	SERR信号驱动使能。1=允许驱动SERR信号线，0=不能驱动。
6	PARITY ERROR RESPONSE	奇偶校验错误响应。1=使能奇偶校验检查0=忽略奇偶错误。
2	BUS MASTER	总线主控能力。1=允许作为主设备操作PCI总线，0=禁止作为主设备操作PCI总线。
1	MEMEORY SPACE	存储器空间访问应答。1=允许访问，0=禁止访问。

25. PCI 分类代码和版本配置寄存器（PCI_CREV）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Class Code																								Revision ID							
r																								r							
0x040000																								0							

位号	助记符	描述
31: 8	CLASS CODE	PCI分类代码。该寄存器值由地址为0xC000_008C的寄存器设定
7: 0	REVISION ID	版本信息位。该寄存器值由地址为0xC000_008C的寄存器设定

26. PCI Latency Timer 寄存器（PCI_LT）

建议初始值：0xFC00。

Address = PCI 配置空间 0x0C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																LATENCY TIMER								保留							
r																r/w						r		r							
0																0						0		0							

位号	助记符	描述
15:8	LATENCY TIMER	该寄存器定义了总线主设备一直保有总线控制权的PCI时钟周期个数，。

27. PCI 基地址寄存器 0（BAR0）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																															
r																															
0																															

28. PCI 基地址寄存器 1（BAR1）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x14

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																															
r																															
0																															

29. PCI 基地址寄存器 2（BAR2）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																															
r																															
0																															

30. PCI 基地址寄存器 3（BAR3）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x1C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE ADDRESS																												PF	TYPE		MSI
r/w																								r				r	r		r
0																								0000				0	0		0

- BASE ADDRESS[31:4]：存储器基地址空间。外部 PCI 设备访问处理器地址空间的基地址，空间大小由寄存器 PCI-AHB 窗口不可预取范围屏蔽寄存器（PCIAHB_SIZ_NP）设定。
- PF[3]：只读，读出值为 0，表明该地址空间不支持预存取。
- TYPE[2:1]：只读，读出值为 00，表明支持 32 位寻址。
- MSI[0]：只读，读出值为 0，表明地址空间为存储器空间。

位号	助记符	描述
31:4	BASE ADDRESS	存储器基地址空间。外部PCI设备访问处理器地址空间的基地址，空间大小由寄存器PCI-AHB窗口不可预取范围屏蔽寄存器（PCIAHB_SIZ_NP）设定。
3	PF	只读，读出值为0，表明该地址空间不支持预存取。
2:1	TYPE	只读，读出值为0，表明支持32位寻址。
0	MSI	只读，读出值为0，表明地址空间为存储器空间。

31. PCI 基地址寄存器 4（BAR4）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x20

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE ADDRESS																												PF	TYPE		MSI
r/w																								r				r	r		r
0																								0000				1	0		0

- BASE ADDRESS[31:4]：存储器基地址空间。外部 PCI 设备访问处理器地址空间的基地址，空间大小由寄存器 PCI-AHB 窗口可预取范围屏蔽寄存器（PCIAHB_SIZ_PF）设定。
- PF[3]：只读，读出值为 1，表明该地址空间支持预存取。
- TYPE[2:1]：只读，读出值为 00，表明支持 32 位寻址。
- MSI[0]：只读，读出值为 0，表明地址空间为存储器空间。

位号	助记符	描述
31:4	BASE ADDRESS	存储器基地址空间。外部PCI设备访问处理器地址空间的基地址，空间大小由寄存器PCI-AHB窗口可预取范围屏蔽寄存器（PCIAHB_SIZ_PF）设定。
3	PF	只读，读出值为0，表明该地址空间支持预存取。
2:1	TYPE	只读，读出值为0，表明支持32位寻址。
0	MSI	只读，读出值为0，表明地址空间为存储器空间。

32. PCI 基地址寄存器 5（BAR5）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x24

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE ADDRESS																												PF	TYPE		MSI
r/w																								r				r	r		r
0																								0000				0	0		0

- BASE ADDRESS[31:4]：配置寄存器基地址空间。外部 PCI 设备可通过该地址空间访问片内 PCI 侧寄存器，空间大小为 256 字节。
- PF[3]：只读，读出值为 0，表明该地址空间不支持预存取。
- TYPE[2:1]：只读，读出值为 00，表明支持 32 位寻址。
- MSI[0]：只读，读出值为 0，表明地址空间为存储器空间。

位号	助记符	描述
31:4	BASE ADDRESS	配置寄存器基地址空间。外部PCI设备可通过该地址空间访问片内PCI侧寄存器，空间大小为256字节。
3	PF	只读，读出值为0，表明该地址空间不支持预存取。
2:1	TYPE	只读，读出值为0，表明支持32位寻址。
0	MSI	只读，读出值为0，表明地址空间为存储器空间。

33. PCI Cardbus CIS Pointer 寄存器

建议初始值：硬件复位值。

Address = PCI 配置空间 0x28

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																															
r																															
0																															

34. PCI 子系统设备和子系统供应商配置寄存器（PCI_SUB）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x2C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBSYSTEM ID																SUBSYSTEM VENDOR ID															
r																r															
0																0															

位号	助记符	描述
31:16	SUBSYSTEM ID	PCI子系统设备信息位，该寄存器值由地址0xC000-0088的寄存器设定。
15:0	SUBSYSTEM VENDOR ID	PCI子系统供应商信息位，该寄存器值由地址0xC000-0088的寄存器设定。

35. PCI 扩展 ROM 基地址寄存器

建议初始值：硬件复位值。

Address = PCI 配置空间 0x30

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																															
r																															
0																															

36. PCI 能力指针寄存器

建议初始值：硬件复位值。

Address = PCI 配置空间 0x34

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																															
r																															
0																								0x78							

37. PCI 保留寄存器

建议初始值：硬件复位值。

Address = PCI 配置空间 0x38

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																															
r																															
0																															

38. PCI 中断线及中断引脚寄存器(PCI_INTLP)

建议初始值：硬件复位值。

Address = PCI 配置空间 0x3C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																INTERRUPT PIN								INTERRUPT LINE							
r																r								r/w							
0																0x1								0							

位号	助记符	描述
15:8	INTERRUPT PIN	中断管脚寄存器。0x1表示设备使用pci_inta_n作为中断输出管脚
7:0	INTERRUPT LINE	中断线寄存器，中断信号pci_inta_n连接到主机的中断控制器哪个管脚上。

39. PCI 中断掩码寄存器（PCI_IMASK）

建议初始值：硬件复位值。

Address = PCI 配置空间 0x80

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CPU DOORBELL				保留											READ DMA END	保留							WRITE DMA END
r								r/w				r											r/w	r							r/w
0								0				0											0	0							0

位号	助记符	描述
23:20	CPU DOORBELL	在从桥模式下，CPU DOORBELL位中断使能位。1=中断使能，0=禁止中断。
8	READ DMA END	在从桥模式下，DMA读操作结束中断使能位。1=中断使能，0=中断禁止。
0	WRITE DMA END	在从桥模式下，DMA写操作结束使能位。1=中断使能，0=中断禁止。

40. PCI 中断状态寄存器(PCI_ISTATUS)

建议初始值：硬件复位值。

Address = PCI 配置空间 0x84

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CPU DOORBELL				保留											READ DMA END	保留							WRITE DMA END
r								r/w				r											r/w	r							r/w
0								0000				0											0	0							0

位号	助记符	描述
23:20	CPU PC DOORBELL	在从桥模式下，1表示对应的CPU DOORBELL位有效。写1则相应位清0,写0无效。
8	READ DMA END	在从桥模式下，1表示DMA读操作结束。写1清0，写0无效。
0	WRITE DMA END	在从桥模式下，1表示DMA写操作结束。写1清0，写0无效。

41. PCI 中断命令寄存器(PCI_ICMD)

建议初始值：硬件复位值。

Address = PCI 配置空间 0x88

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												PCI DOORBELL				保留																
r												w				r															
0												0000				0															

位号	助记符	描述
19：16	PCI DOORBELL	在从桥模式下，PCI DOORBELL位。该字段只写，读出的值为0，写1使相应位有效，发出中断请求。

42. PCI 状态及版本信息寄存器(PCI_VERSION)

建议初始值：硬件复位值。

Address = PCI 配置空间 0x8C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																TYPE				VERSION											
r																r				r											
0																xxxx				0x380											

位号	助记符	描述
15:12	TYPE	表示PCI接口模式 1 = 32位从桥模式 2 = 32位主桥模式 管脚pci_host接低电平，为2；接高电平，为1
11:0	VERSION	PCI桥版本,由IC厂商定义

3.13 DSU 寄存器

1. DSU 控制寄存器

Address = 0x9000-0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			DCNT									RE	DR	LR	SS	PE	EE	EB	DM	DE	BZ	BX	BB	BN	BS	BW	BE	FT	BT	DM	TE
r			r/w									r/w	r/w	r/w	r/w	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
0			x									0	0	0	0	0	x	x	0	0	x	x	0	x	0	x	x	0	0	x	0

位号	助记符	描述
31:29	R	保留位，读出为 0
28:20	DCNT	AHB Trace延时计数器，延时冻结Trace功能的计数值，与 *DE* 位配合使用
19	RE	错误模式恢复位，置为‘1’时将处理器从错误模式中复位。1=复位，0=不复位
18	DR	调试模式响应位。置为‘1’时处理器进入调试模式时从DSU串口输出高电平响应，1=输出响应，0=不输出响应
17	LR	调试模式下DSU读/写响应位控制。置为‘1’时，在DSU读/写命令时从DSU串口输出高电平响应。1=输出响应，0=不输出响应
16	SS	单步执行控制位。置为‘1’时，处理器执行一条指令后将重新返回调试状态1=单步执行，0=不进行单步执行
15	PE	处理器错误模式状态位。1=错误模式，0=正常模式
14	EE	外部DSUEN信号指示位。
13	EB	外部DSUBRE信号指示位。
12	DM	调试模式指示位。1=调试模式，0=非调试模式
11	DE	延时计数模式使能位。置为‘1’时，在DSU断点命中时延时计数器开始减1计数，计数值为0时AHB Trace功能才会冻结，否则立即冻结。1=使能，0=不使能
10	BZ	处理器能被挂起的条件位，置位为1，则表示特定陷阱才能引起处理器进入调试模式，此特殊陷阱为不包括特权指令、FPU禁止、窗口溢出、异步中断、ticc陷阱的陷阱。清0，则上述条件不能挂起
9	BX	处理器能被挂起的条件位，置位为1，则表示任意陷阱都能引起处理器进入调试模式。清0，则上述条件不能挂起
8	BB	处理器能被挂起的条件位，置位为1，则在DSU断点命中时将强制处理器进入调试模式。清0，则上述条件不能挂起
7	BN	处理器能被挂起的条件位，置位为1，则立即进入调试模式，清0则恢复执行模式。
6	BS	处理器能被挂起的条件位，置位为1，则在执行中断指令（ta 1指令）时进入调试模式。清0，则上述条件不能挂起
5	BW	处理器能被挂起的条件位，置位为1，则在IU观测点命中时进入调试模式（trap 0xb）。清0，则上述条件不能挂起
4	BE	处理器能被挂起的条件位，置位为1，则当处理器进入错误状态时进入调试模式（trap in trap）。清0，则上述条件不能挂起
3	FT	定时器时钟冻结位，置位为1，则在调试模式下定时器处于挂起状态，从而应用程序的时间能保持不变。
2	BT	处理器能被挂起的条件位，置位为1，则在trace延时功能使能条件下trace功能冻结时才会进入调试模式。清0，则上述条件不能挂起
1	DM	总线追踪或指令追踪设置位，置位为1，则在混合追踪模式下进行AHB总线功能延时冻结，清0则只进行指令追踪功能延时冻结。
0	TE	Trace功能使能位，1=使能trace功能，0=不使能trace功能

2. Trace buffer 控制寄存器

Address = 0x9000-00004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						TA	TI	R			AHB INDEX									R			INST INDEX								
r						r/w	r/w	r			r/w									r			r/w								
0						x	x	0			x									0			x								

位号	助记符	描述
31:26	R	保留位，读出为0。
25	TA	AHB总线追踪使能位。1=总线追踪使能，0=不使能总线追踪。
24	TI	指令追踪使能位。1=指令追踪使能，0=不使能。
23:21	R	保留位，读出为0。
20:12	AHB INDEX	当前所追踪总线信息存放位置索引。
11:9	R	保留位，读出为0。
8:0	INST INDEX	当前所追踪指令信息存放位置索引。

3. DSU Trace 时间标志寄存器

Address = 0x9000-0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DSU TIME TAG VALUE																														
r	r/w																														
0	0																														

位号	助记符	描述
31:30	R	保留位，读出为0
29:0	DSU TIME TAG VALUE	DSU追踪信息时间标志计数值。

4. AHB 断点寄存器 1

Address = 0x9000-0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BADD																														R	EN
r/w																														r	w
x																														0	0

位号	助记符	描述
31:2	BADD	断点地址位。
1	R	保留位，读出为0。
0	EN	指令执行使能比较位，1=指令执行时使能比较PC值，0=不比较PC值。

5. AHB 屏蔽地址寄存器 1

Address = 0x9000-0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMA																														LD	ST
r/w																														r/w	r/w
x																														0	0

位号	助记符	描述
31:2	BMA	屏蔽地址位。与DSU断电寄存器配合使用，如果某位为0，则在判断断点命中时忽略DSU断点寄存器的相应位，表明该位不同也能命中。
1	LD	置为‘1’时，则读数据时进行地址比较，否则不进行地址比较。1=使能比较，0=不使能比较
0	ST	置为‘1’时，则写数据时进行地址比较，否则不进行地址比较。1=使能比较，0=不使能比较

6. AHB 断点寄存器 2

Address = 0x9000-0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BADD																														R	EN
r/w																														r	w
x																														0	0

位号	助记符	描述
31:2	BADD	断点地址位。
1	R	保留位，读出为0。
0	EN	指令执行使能比较位，1=指令执行时使能比较PC值，0=不比较PC值。

7. AHB 屏蔽地址寄存器 2

Address = 0x9000-001c

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMA																														LD	ST
r/w																														r/w	r/w
x																														0	0

位号	助记符	描述
31:2	BMA	屏蔽地址位。与DSU断电寄存器配合使用，如果某位为0，则在判断断点命中时忽略DSU断点寄存器的相应位，表明该位不同也能命中。
1	LD	置为‘1’时，则读数据时进行地址比较，否则不进行地址比较。1=使能比较，0=不使能比较
0	ST	置为‘1’时，则写数据时进行地址比较，否则不进行地址比较。1=使能比较，0=不使能比较

8. DSU 串口状态寄存器

Address = 0x8000-00c4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																									FE	R	OV	BR	TH	TS	DR
r																									r/w	r	r/w	r/w	r	r	r
0																									0	0	0	0	1	1	0

位号	助记符	描述
31:7	R	保留位，读出为0。
6	FE	数据帧错误指示位。1=表示数据帧错误,表明检测到帧错误， 0=没有检测到错误。
5	R	保留位，读出为0。
4	OV	数据溢出指示位。1=表示溢出，一个或多个字符丢失，0=没有溢出。
3	BR	Break标志位，1=收到break，0=没有接收到break。
2	TH	发送保持寄存器空标志位，1=为空， 0=不为空。
1	TS	发送移位寄存器空标志位，1=为空， 0=不为空。
0	DR	表示已收到数据但尚未被AHB主设备读出，1=收到但未读出，0=没有收到。

9. DSU 串口控制寄存器

Address = 0x8000-00c8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																														BL	EN
r																														r/w	r/w
0																														0	0

位号	助记符	描述
31:2	R	保留位，读出位0。
1	BL	波特率锁定指示位。1=锁定，0=没有锁定。
0	EN	接收发送使能位。1=使能，允许接收和发送；0=不使能。

10. DSU 串口波特率重载寄存器

Address = 0x8000-00cc

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														SRV																	
r														r/w																	
0														3FFFF																	

位号	助记符	描述
31 : 18	R	**保留位，读出为0。**
17 : 0	SRV	**波特率重载值。重载值 = （（系统时钟*10）/（波特率*8） - 5）/10。**

3.14 寄存器堆地址列表

3.14.1 全局寄存器

%g0 0x9002-0280

%g1 0x9002-0284

%g2 0x9002-0288

%g3 0x9002-028C

%g4 0x9002-0290

%g5 0x9002-0294

%g6 0x9002-0298

%g7	0x9002-029C

3.14.2 窗口寄存器

cwp = 0

%o0	0x9002-0020	%l0	0x9002-0040	%i0	0x9002-0060
%o1	0x9002-0024	%l1	0x9002-0044	%i1	0x9002-0064
%o2	0x9002-0028	%l2	0x9002-0048	%i2	0x9002-0068
%o3	0x9002-002C	%l3	0x9002-004C	%i3	0x9002-006C
%o4	0x9002-0030	%l4	0x9002-0050	%i4	0x9002-0070
%o5	0x9002-0034	%l5	0x9002-0054	%i5	0x9002-0074
%o6	0x9002-0038	%l6	0x9002-0058	%i6	0x9002-0078
%o7	0x9002-003C	%l7	0x9002-005C	%i7	0x9002-007C

cwp = 1

%o0	0x9002-0060	%l0	0x9002-0080	%i0	0x9002-00A0
%o1	0x9002-0064	%l1	0x9002-0084	%i1	0x9002-00A4
%o2	0x9002-0068	%l2	0x9002-0088	%i2	0x9002-00A8
%o3	0x9002-006C	%l3	0x9002-008C	%i3	0x9002-00AC
%o4	0x9002-0070	%l4	0x9002-0090	%i4	0x9002-00B0
%o5	0x9002-0074	%l5	0x9002-0094	%i5	0x9002-00B4
%o6	0x9002-0078	%l6	0x9002-0098	%i6	0x9002-00B8
%o7	0x9002-007C	%l7	0x9002-009C	%i7	0x9002-00BC

cwp = 2

%o0	0x9002-00A0	%l0	0x9002-00C0	%i0	0x9002-00E0
%o1	0x9002-00A4	%l1	0x9002-00C4	%i1	0x9002-00E4
%o2	0x9002-00A8	%l2	0x9002-00C8	%i2	0x9002-00E8
%o3	0x9002-00AC	%l3	0x9002-00CC	%i3	0x9002-00EC
%o4	0x9002-00B0	%l4	0x9002-00D0	%i4	0x9002-00F0
%o5	0x9002-00B4	%l5	0x9002-00D4	%i5	0x9002-00F4
%o6	0x9002-00B8	%l6	0x9002-00D8	%i6	0x9002-00F8
%o7	0x9002-00BC	%l7	0x9002-00DC	%i7	0x9002-00FC

cwp = 3

%o0	0x9002-00E0	%l0	0x9002-0100	%i0	0x9002-0120
%o1	0x9002-00E4	%l1	0x9002-0104	%i1	0x9002-0124
%o2	0x9002-00E8	%l2	0x9002-0108	%i2	0x9002-0128
%o3	0x9002-00EC	%l3	0x9002-010C	%i3	0x9002-012C
%o4	0x9002-00F0	%l4	0x9002-0110	%i4	0x9002-0130

%o5	0x9002-00F4	%l5	0x9002-0114	%i5	0x9002-0134
%o6	0x9002-00F8	%l6	0x9002-0118	%i6	0x9002-0138
%o7	0x9002-00FC	%l7	0x9002-011C	%i7	0x9002-013C

cwp = 4

%o0	0x9002-0120	%l0	0x9002-0140	%i0	0x9002-0160
%o1	0x9002-0124	%l1	0x9002-0144	%i1	0x9002-0164
%o2	0x9002-0128	%l2	0x9002-0148	%i2	0x9002-0168
%o3	0x9002-012C	%l3	0x9002-014C	%i3	0x9002-016C
%o4	0x9002-0130	%l4	0x9002-0150	%i4	0x9002-0170
%o5	0x9002-0134	%l5	0x9002-0154	%i5	0x9002-0174
%o6	0x9002-0138	%l6	0x9002-0158	%i6	0x9002-0178
%o7	0x9002-013C	%l7	0x9002-015C	%i7	0x9002-017C

cwp = 5

%o0	0x9002-0160	%l0	0x9002-0180	%i0	0x9002-01A0
%o1	0x9002-0164	%l1	0x9002-0184	%i1	0x9002-01A4
%o2	0x9002-0168	%l2	0x9002-0188	%i2	0x9002-01A8
%o3	0x9002-016C	%l3	0x9002-018C	%i3	0x9002-01AC
%o4	0x9002-0170	%l4	0x9002-0190	%i4	0x9002-01B0
%o5	0x9002-0174	%l5	0x9002-0194	%i5	0x9002-01B4
%o6	0x9002-0178	%l6	0x9002-0198	%i6	0x9002-01B8
%o7	0x9002-017C	%l7	0x9002-019C	%i7	0x9002-01BC

cwp = 6

%o0	0x9002-01A0	%l0	0x9002-01C0	%i0	0x9002-01E0
%o1	0x9002-01A4	%l1	0x9002-01C4	%i1	0x9002-01E4
%o2	0x9002-01A8	%l2	0x9002-01C8	%i2	0x9002-01E8
%o3	0x9002-01AC	%l3	0x9002-01CC	%i3	0x9002-01EC
%o4	0x9002-01B0	%l4	0x9002-01D0	%i4	0x9002-01F0
%o5	0x9002-01B4	%l5	0x9002-01D4	%i5	0x9002-01F4
%o6	0x9002-01B8	%l6	0x9002-01D8	%i6	0x9002-01F8
%o7	0x9002-01BC	%l7	0x9002-01DC	%i7	0x9002-01FC

cwp = 7

%o0	0x9002-01E0	%l0	0x9002-0000	%i0	0x9002-0020
%o1	0x9002-01E4	%l1	0x9002-0004	%i1	0x9002-0024
%o2	0x9002-01E8	%l2	0x9002-0008	%i2	0x9002-0028
%o3	0x9002-01EC	%l3	0x9002-000C	%i3	0x9002-002C
%o4	0x9002-01F0	%l4	0x9002-0010	%i4	0x9002-0030
%o5	0x9002-01F4	%l5	0x9002-0014	%i5	0x9002-0034

%o6	0x9002-01F8	%l6	0x9002-0018	%i6	0x9002-0038
%o7	0x9002-01FC	%l7	0x9002-001C	%i7	0x9002-003C

3.14.3 浮点寄存器

%f0	0x9002-0200	%f8	0x9002-0220
%f1	0x9002-0204	%f9	0x9002-0224
%f2	0x9002-0208	%f10	0x9002-0228
%f3	0x9002-020C	%f11	0x9002-022C
%f4	0x9002-0210	%f12	0x9002-0230
%f5	0x9002-0214	%f13	0x9002-0234
%f6	0x9002-0218	%f14	0x9002-0238
%f7	0x9002-021C	%f15	0x9002-023C
%f16	0x9002-0240	%f24	0x9002-0260
%f17	0x9002-0244	%f25	0x9002-0264
%f18	0x9002-0248	%f26	0x9002-0268
%f19	0x9002-024C	%f27	0x9002-026C
%f20	0x9002-0250	%f28	0x9002-0270
%f21	0x9002-0254	%f29	0x9002-0274
%f22	0x9002-0258	%f30	0x9002-0278
%f23	0x9002-025C	%f31	0x9002-027C

3.15 寄存器使用注意事项

- ✓ 不使用造错功能（含 REGFILE、Cache、外部存储器）时，应确保相关寄存器处于非造错模式；
- ✓ IU 窗口寄存器及浮点运算寄存器，在读访问之前必须先写，以避免读时出现 EDAC 校验错误；
- ✓ 使用“写状态寄存器”指令修改 PSR、WIM、TBR、ASR16、ASR17 寄存器，在三指令周期内需要访问或隐含访问这些寄存器时，须（增加指令如 nop）满足该寄存器前一指令写入三指令后生效的要求，确保后续指令访问到的是已更新的状态寄存器；
- ✓ 在进行浮点运算之前，应先初始化%fsr 寄存器；
- ✓ 在可能触发 trap 之前，必须先初始化 TBR 寄存器及建立相关 trap 的处理过程，以及将%psr.ET 置为 1；
- ✓ 在进行与窗口有关的操作前，必须先初始化 WIM 和 PSR 寄存器的 CWP 字段；

- ✓ 在使用外部存储器之前，必须确保相关的外部存储器配置寄存器已初始化；
- ✓ 在使用 Cache 前，必须先在关 Cache 时刷新 Cache，完毕后使能相关 Cache 控制；
- ✓ 在使用外部设备前，应先清除其上电后寄存器的不定状态，再按应用要求初始化；
- ✓ 在程序调用时，使用堆栈之前，应先初始化%sp 寄存器；
- ✓ 在初始化过程中（除了跳转延迟槽后）增加 nop 指令，对程序没有功能影响。

附录 4 寄存器初始化

4.1 寄存器初始化简介

BM3803FMGRH 实现了内核寄存器和片上寄存器，根据寄存器的实现和应用要求，相关寄存器上电复位后应软件初始化，为用户程序设置一个固定的 CPU 初始环境，确保应用程序不发生异常。

初始化寄存器的基本要求是，在写 REGFILE 寄存器（含窗口寄存器、浮点数据寄存器）之前应先初始化 ASR16 寄存器。

根据寄存器功能，可把寄存器分为三部分：内核寄存器、片上寄存器、用户应用程序相关寄存器。依据使用习惯，寄存器初始化过程可按此三部分顺次初始化（实际对配置顺序不做强制要求，以满足应用要求为目标）。流程示意图如图A4-1-1所示。

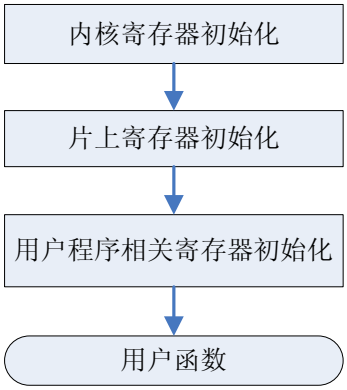


图 A4-1-1 启动程序初始化流程

4.2 推荐寄存器初始化过程

4.2.1 内核寄存器初始化

内核寄存器包括IU及FPU相关的寄存器以及辅助寄存器。如窗口寄存器、浮点数据寄存器、PSR、WIM、TBR、FSR、Y寄存器、ASR16、ASR17、ASR24-ASR31寄存器。初始化示意流程图如下所示。

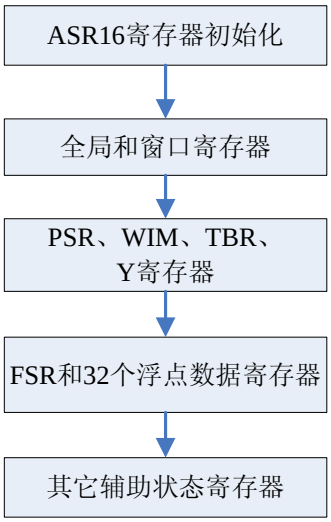


图 A4-2-1 内核寄存器初始化流程

首先需要初始化的寄存器为ASR16，如果ASR16寄存器没有初始化，则可能会导致上电后使处理器处在REGFILE造错模式。

4.2.1.1 辅助状态寄存器 ASR16

CPU 上电复位后写内核寄存器前首先需要对 ASR16 进行初始化，即关闭造错功能，初始化值推荐为 0x0（确保高 3 位不处于造错模式），示例代码如下所示（叹号!之后的语句为本行注释）：

```
mov %g0, %asr16          ! %asr16清零
nop;nop;nop;
```

4.2.1.2 IU 相关的寄存器

（1）全局和窗口寄存器

在寄存器堆中的寄存器（包括：全局寄存器%g1-%g7，寄存器窗口中的局部寄存器%l0-%l7、输出寄存器%o0-%07（与%i0-%i7 在交叠窗口中物理实现为同一寄存器），32 个浮点寄存器（%f0-%f31），由于采用了 EDAC 设计，要求在读之前确保其 32 位数据和 7 位校验码匹配，读访问之前完成初始化即可。启动时软件初始化这些寄存器，可以确保应用程序未写先读发生异常。

这些窗口寄存器可以全部设置为 0x0。初始化时当前窗口指向窗口 0，trap 禁止并设置

处理器处于管理态（**处理器状态寄存器** PSR 的初值设置为 0x10c0），然后将全局寄存器%g1~%g7 清零，旋转窗口，依次把每个窗口的 local 寄存器、input/output 寄存器清零（可以初始化为其它值，清零更简单），示例如下：

```
        set 0x10c0,%g1
        mov %g1,%psr
        nop
        nop
        nop
        mov %g0, %wim  !WIM设置为不存在无效窗口，避免旋转窗口时上下溢
        nop      !三周期内指令不涉及WIM，此处3条nop可省
        nop      !本附录中的例子并不是所有状态寄存器后都增加了三周期
        nop
        mov %g0, %g1
        mov %g0, %g2
        mov %g0, %g3
        mov %g0, %g4
        mov %g0, %g5
        mov %g0, %g6
        mov %g0, %g7
        ! init the local/in/out registers
        mov 0x8, %g1      !寄存器窗口为8个
    1:  mov %g0, %l0      !设置标号1
        mov %g0, %l1
        mov %g0, %l2
        mov %g0, %l3
        mov %g0, %l4
        mov %g0, %l5
```

```
mov %g0, %l6
mov %g0, %l7
mov %g0, %i0
mov %g0, %i1
mov %g0, %i2
mov %g0, %i3
mov %g0, %i4
mov %g0, %i5
mov %g0, %i6
mov %g0, %i7
subcc %g1,1,%g1
save      !旋转窗口
bne 1b    ! %g1不为0（以%psr.icc.Z判断），则回跳到标号1处
nop
```

（2）其它 IU 寄存器

初始化窗口寄存器时,PSR 寄存器的值设定为 0x10c0(使能浮点单元、关闭 trap 使能、设置处理器处于管理态、当前窗口指针指向 0 窗口)，即在“全局和窗口寄存器初始化”一节的设置值即满足要求。

窗口无效屏蔽寄存器（WIM)，由于%psr.CWP 已设置为 0，也就是当前窗口指向 0 窗口，为了最大化利用窗口寄存器，一般把 WIM 设置为窗口 1，即 WIM 设置为 2，示例代码如下：

```
set 2, %g1
mov %g1, %wim
nop
nop
nop
```

异常基址寄存器（TBR）定义了响应 trap/中断时系统的跳转目标的基地址，必须是一个 4K 字节地址对齐的值，一般是 trap 表的标号。示例如下：

```
sethi %hi(_trap_table), %g1     !或者用绝对值，如set 0x40000000,%g1
mov %g1, %tbr
nop
nop
nop
```

乘法/除法寄存器（Y）的初始值可以初始化为 0：

```
wr %g0,0,%y
nop
nop
nop
```

4.2.1.3 FPU 相关的寄存器

（1）浮点状态寄存器

在使用浮点指令之前必须先使能浮点处理单元（一般在 PSR 中已经将其使能），之后配置**浮点状态寄存器** FSR。该寄存器初始值必需从存储器中载入，如先定义一个 WORD 型（32 位）变量（初始值设置为 0），之后使用该变量初始化 FSR。参考代码如下：

```
.data  !该变量数据段，如果不在数据段，也可以使用
.global _fsrinit   !声明变量，名字叫_fsrinit
.align 4          !该变量存放的地址是四字节对齐
_fsrinit:
.word 0x0     !变量的值，也是 fsr 寄存器的初值

set   _fsrinit,%g1  ! _fsrinit 变量存放的地址
 ld    [%g1] , %fsr
 nop
 nop
 nop
```

（2）浮点数据寄存器

%f0~%f31 寄存器可以从存储器中初始化，如先定义一个 WORD 型变量(可设初始值为 0x0)，之后使用该变量初始化。参考代码为：

```
.global  _fpdata     ! 定义变量_fpdata
.align 4      !四字节对齐
_fpdata:
.word 0     !浮点寄存器的值

set  _fpdata,%g1
ld  [%g1], %f0  !其它 f1-f31 也如此，作为目的寄存器进行初始化或者将_fpdata 定
```

义为 8 字节对齐数据，使用 ldd 指令一次初始化连续两个浮点数据寄存器。

```
.align 8  !八字节对齐
_fpdata:
.word 0,0 !浮点寄存器的值，占两个 word
set     _fpdata,%g1
ldd [%g1], %f0  !将 f0,f1 初始化为 0
ldd [%g1], %f2  !将 f2,f3 初始化为 0，其它用 f4-f30 的寄存器如此初始化
```

当初始化完%f0 寄存器后，可以使用浮点操作指令，初始化其它浮点寄存器，由于不访问数据存储器，速度稍快，如：

```
fmovs %f0,%f1  !浮点数据寄存器%f1 作为目的寄存器完成%f1 初始化，但在浮点协处
```

理器被使能之前，不能使用浮点指令(初始化浮点寄存器)。

4.2.1.4 其他辅助状态寄存器

辅助寄存器中ASR17、ASR24、ASR26、ASR28、ASR30和ASR25、ASR27、ASR29、ASR31寄存器，一般设置为0，无配置顺序要求，参考代码如下：

```
mov %g0,%asr17
mov %g0,%asr24
mov %g0,%asr26
mov %g0,%asr28
```

```
mov %g0,%asr30
mov %g0,%asr25
mov %g0,%asr27
mov %g0,%asr29
mov %g0,%asr31
```

4.2.2 片上寄存器初始化

片上寄存器包括外部存储器及容错配置寄存器、Cache 控制及容错寄存器、中断控制寄存器、GPIO、PCI 相关寄存器、定时器、串口以及 DSU 寄存器。这些寄存器可根据应用需要初始化，流程示意图如图 A4-2-2 所示。

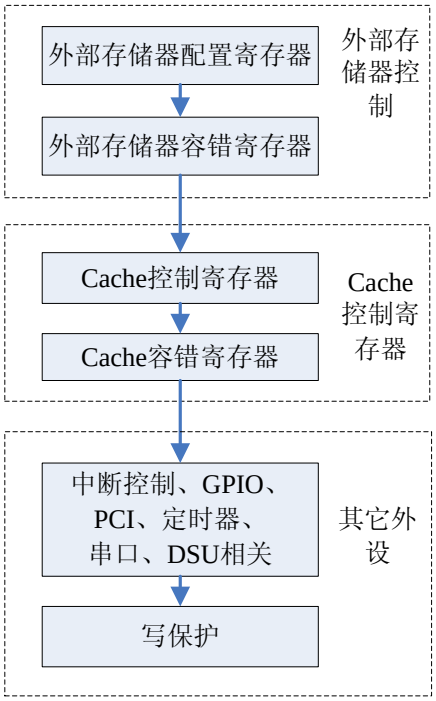


图 A4-2-2 片上寄存器初始化流程

1、外部存储器配置寄存器

外部存储器接口配置通过**外部存储器配置寄存器** MCFG1、MCFG2、MCFG3 来实现，初始值需要根据外部存储器系统设计来初始化，是与硬件相关的。如果硬件设计不包括 SDRAM 存储器，则可以不设置 MCFG3。

外部存储器容错配置寄存器实现对外部存储器的纠检错功能的控制，包括外部存储器容

错配置寄存器 MECFG1、MECFG2、MECFG3；在不造错的模式下，后三个寄存器不需要配置。在 RAM 空间 EDAC 使能时，需要初始化 MECFG1（0x80000100）。

如RAM存储器系统设计为无SDRAM，32位SRAM置于bank 0（另有7位校验SRAM），bank大小为2M字节，读、写等待周期分别设置为3个CPU周期，且该BANK使能EDAC。则需要初始化MCFG2和MECFG1。参考代码为：

```
set    0x80000000, %g1
set    0x00019063, %g2    !使能 EDAC 时，可不明确指出使用读改写模式
st      %g2, [%g1+0x4]
set    0x80400000,%g2    !设置为 0x80000000 也满足应用要求，
                          !仅考虑 sram，不应影响 SDRAM 和 PROM 的 EDAC
st     %g2, [%g1+0x100]
```

上电后 mcfg1 可以不重新初始化，除非用到 IO 空间或修改 ROM 空间的读写等待周期；或者程序跳转到其它存储器空间后修改 rom 空间的其它属性。

2、Cache 控制寄存器

Cache 相关寄存器包括 **Cache 控制寄存器**（CCR）和 **Cache 容错寄存器**。初始化时可根据实际需要选择使能 Cache 或禁止 Cache，使能的参考代码为：

```
set    0x80000000, %g1
set    0x0000000f , %g2       ! 使能指令和数据 Cache
st     %g2, [%g1+0x14]
```

如果要禁止 Cache，直接向 CCR 寄存器写 0 即可。

```
set    0x80000000, %g1
st      %g0, [%g1+0x14]
```

Cache 刷新操作在使能 Cache 前进行。

Cache 容错控制寄存器 1、2、3 的上电默认值即可正常使用，不需要初始化。

3、中断控制寄存器

中断控制主要包括中断相关的四个寄存器，在初始化时需要清除**中断请求寄存器**，清除**中断强制寄存器**，并屏蔽全部中断。初始化参考代码为：

```
set    0x80000000, %g1
st     %g0, [%g1+0x90]  ! level 和 mask 均置 0
st     %g0, [%g1+0x98]  !清除强制中断寄存器的上电不定态
set    0xffff,%g2
st     %g2, [%g1+0x9c]   !清除中断请求寄存器的上电不定态
```

4、写保护寄存器

写保护功能对 ram 空间的意外写操作进行保护，支持两个写保护寄存器分别对不同的地址区域进行写保护。相关寄存器为**写保护寄存器** 1 和 2;初始化为禁止写保护的参考代码为：

```
set    0x80000000, %g1
st     %g0, [%g1+0x1c]
st     %g0, [%g1+0x20]
```

5、其它控制寄存器

GPIO 方向寄存器在上电复位后为输入，**GPIO 中断配置寄存器**的复位初始值也是不产生中断的设置，使用时根据需要初始化。

三个通用串口中，复位后控制寄存器值为发送/接收不使能，不使用可不进行初始化。

定时器和看门狗上电后是不使能的，不使用可以不进行初始化。

系统寄存器是只读的，不需要初始化。

PCI 模块上电复位后，与 CPU 中断相关的三个寄存器和两个 DMA 读写控制寄存器复位值均为 0，不使用可不进行初始化。

DSU 在运行模式不需要使用，禁止 DSU 串口发送接收（调试程序时不能禁止）参考代码如下：

```
set    0x80000000, %g1
st     %g0, [%g1+0xc8]
```

4.2.3 用户程序相关寄存器初始化

完成处理器内核寄存器和片上寄存器的初始化后，可以进行应用程序相关寄存器的初始化，主要是指定栈的栈底，并使能中断和TRAP。示例如图A4-2-3所示。

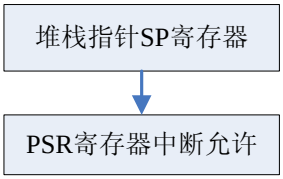


图 A4-2-3 用户程序相关寄存器初始化

根据应用情况设置堆栈栈底指针，在本示例中把堆栈指针设置为SRAM存储器的高端位置，如SRAM空间大小为1MB，初始化参考代码如下所示。

```
set   0x400FFF00, %g1    !预留了0x100字节在存储器高地址处
mov   %g1, %sp
```

然后使能系统的TRAP允许并向用户主函数如main跳转。

```
set    0x10e0,%g1,
mov   %g1,%psr
nop
nop
nop
call main   !向用户函数如main函数跳转
nop
nop
```

片上外设相关的寄存器，在使用时应根据应用要求进行相应设置。在使用中断时，一般先初始化中断源设备的相关寄存器，再设置中断控制器相关的寄存器。

附录 5 软件开发规范

5.1 通用编程规范

5.1.1 寄存器窗口与函数调用

5.1.1.1 函数调用

函数调用的一般原则：

1．调用函数通过 CALL 指令使 PC 指针分支到被调函数的入口地址。

2．被调函数使用 SAVE 指令获得新的寄存器窗口，并取得新的堆栈空间。

3．被调函数使用“RET； RESTORE”指令组合返回调用函数。

例 1：调用函数：main（）；被调函数：printf（）

```
main：
    call printf                 ！调用 printf（）
    nop                         ！延迟一条指令
    nop                         ！位置
    …
printf:
        save %sp, -1024, %sp        ！获得新寄存器窗口
                                    ！和新堆栈空间
        …
        ret
        restore                     ！被调函数返回
                                    ！main（）位置
```

5.1.1.2 参数传递

参数传递的一般规则：

1．调用函数将参数写入当前寄存器窗口的输出寄存器中（%o0~%o5）。如果参数多于 6 个，则从第 7 个参数开始写入堆栈。

2．调用函数使用 CALL 指令调用被调函数。

3．被调函数使用 SAVE 获得新寄存器窗口和堆栈后，从新寄存器窗口的输入寄存器（%i0~%i5）取得参数。如果参数多于 6 个，则从调用函数的堆栈取得后面的参数。

4．被调函数将返回值放入输入寄存器。返回调用函数后，调用函数从输出寄存器中取得返回值。

例 2：

```
int reg_window(int a1,int a2,int a3,int a4, int a5, int a6, int a7,int a8)
{
    int b;
    b=a1+a2+a3+a4+a5+a6+a7+a8;
    return b;
}
main()
{
    int c1=1,c2=2,c3=3,c4=4,c5=5,c6=6,c7=7,c8=8;
    int sum = 0;

    sum = reg_window(c1,c2,c3,c4,c5,c6,c7,c8);
    sum = sum + 1;
}
```

以上两函数编译后生成：

```
<main>:
 save   %sp, -152, %sp
 mov    1, %o0
 st     %o0, [ %fp + -12 ]
 mov    2, %o0
```

```
st    %o0, [ %fp + -16 ]
mov    3, %o0
st    %o0, [ %fp + -20 ]
mov    4, %o0
st    %o0, [ %fp + -24 ]
mov    5, %o0
st    %o0, [ %fp + -28 ]
mov    6, %o0
st    %o0, [ %fp + -32 ]
mov    7, %o0
st    %o0, [ %fp + -36 ]
mov    8, %o0
st    %o0, [ %fp + -40 ]
clr    [ %fp + -44 ]
ld    [ %fp + -36 ], %o0
st    %o0, [ %sp + 0x5c ]     ！参数 c7，堆栈传递
ld    [ %fp + -40 ], %o0
st    %o0, [ %sp + 0x60 ]     ！参数 c8，堆栈传递
ld    [ %fp + -12 ], %o0      ！参数 c1
ld    [ %fp + -16 ], %o1      ！参数 c2
ld    [ %fp + -20 ], %o2      ！参数 c3
ld    [ %fp + -24 ], %o3      ！参数 c4
ld    [ %fp + -28 ], %o4      ！参数 c5
ld    [ %fp + -32 ], %o5      ！参数 c6
call    40001d30 <reg_window>
nop
st    %o0, [ %fp + -44 ] ！从输出寄存器
```

```
！取得返回值 b→sum
 ld    [ %fp + -44 ], %o0
 add    %o0, 1, %o1
 st    %o1, [ %fp + -44 ]
 ret
 restore


<reg_window>:
 save    %sp, -112, %sp
 st    %i0, [ %fp + 0x44 ]     ！输入参数，c1→a1
 st    %i1, [ %fp + 0x48 ]     ！输入参数，c2→a2
 st    %i2, [ %fp + 0x4c ]     ！输入参数，c3→a3
 st    %i3, [ %fp + 0x50 ]     ！输入参数，c4→a4
 st    %i4, [ %fp + 0x54 ]     ！输入参数，c5→a5
 st    %i5, [ %fp + 0x58 ]     ！输入参数，c6→a6
 ld    [ %fp + 0x44 ], %o0
 ld    [ %fp + 0x48 ], %o1
 add    %o0, %o1, %o0
 ld    [ %fp + 0x4c ], %o1
 add    %o0, %o1, %o0
 ld    [ %fp + 0x50 ], %o1
 add    %o0, %o1, %o0
 ld    [ %fp + 0x54 ], %o1
 add    %o0, %o1, %o0
 ld    [ %fp + 0x58 ], %o1
 add    %o0, %o1, %o0
 ld    [ %fp + 0x5c ], %o1     ！输入参数，c7→a7
```

```
add    %o0, %o1, %o0
ld    [ %fp + 0x60 ], %o1     ！输入参数，c8→a8
add    %o0, %o1, %o0
st    %o0, [ %fp + -12 ]
ld    [ %fp + -12 ], %o0
mov    %o0, %i0     ！返回值 b
b    40001d9c <reg_window+0x6c>
nop
ret
restore
```

5.1.1.3 C 语言、汇编语言函数混合调用

C 语言与汇编语言函数相互调用时同样遵循上述参数传递原则。不同的是，需要手工确定参数地址和顺序。

例 3，C 程序调用汇编语言函数：

使用汇编语言定义数据向左位移函数 asm_shift_function(p1,p2); //p1=被位移的数据；p2=位移量。例中使用 asm_shift_function()将 1 左移 2 位。代码如下：

```
asm("
asm_shift_function:
save %sp, -112, %sp
sll %i0, %i1, %i0     ！从输入寄存器中取得输入参数
ret
restore
 ");

main()
{
      int result = 0;
```

```
        result = asm_shift_function(1,2); // 输入参数
    }
```

汇编后生成：

```
<asm_shift_function>:  ！编译器不修改汇编语言编写的代码
save  %sp, -112, %sp
sll  %i0, %i1, %i0
ret
restore

<main>:
save  %sp, -152, %sp
clr  [ %fp + -48 ]
mov  1, %o0    ！按照 C 代码顺序将参数放入输出寄存器
mov  2, %o1    ！按照 C 代码顺序将参数放入输出寄存器
call  40001da4 <asm_shift_function>
nop
st  %o0, [ %fp + -48 ] ！从输出寄存器中取得函数返回值
ret
restore
```

5.1.2 堆栈

SPARC V8 标准定义的程序堆栈由堆指针（%fp）和栈指针（%sp）组成，由高地址向低地址生长。

堆栈使用方法如图 A5-1-1 所示。

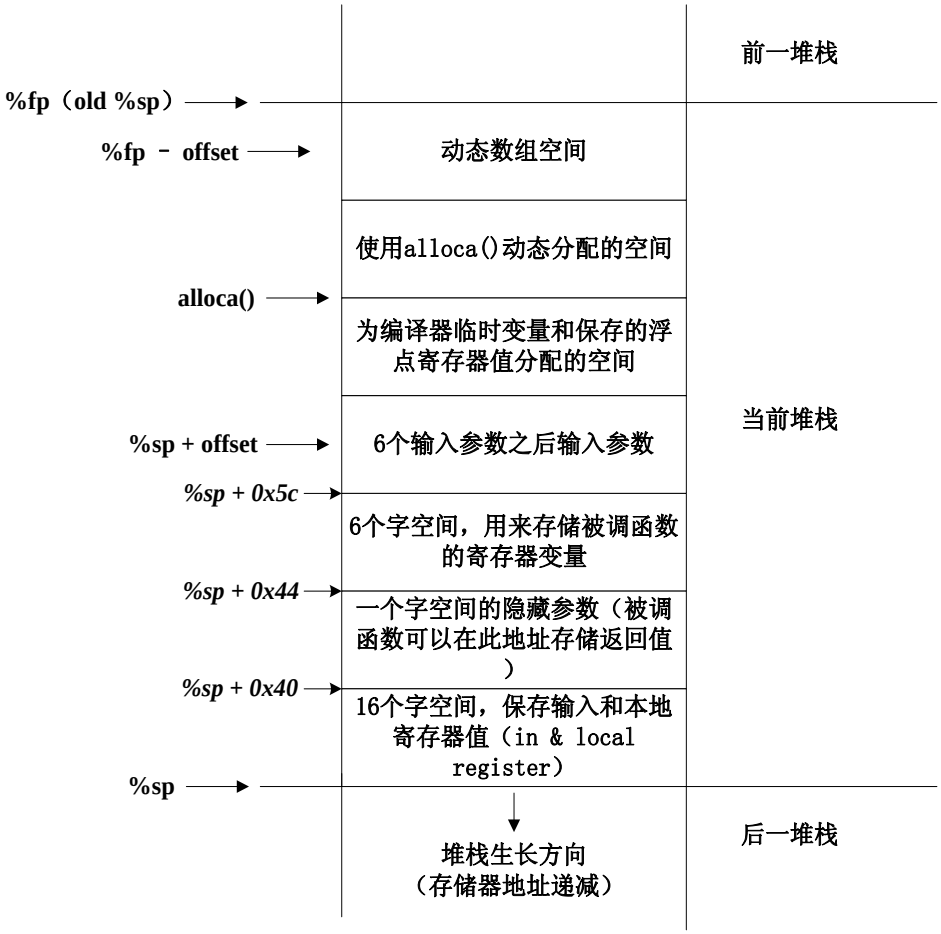


图 A5-1-1 堆栈使用方法

每次分配一个寄存器窗口，同时也分配一段存储器空间作为堆栈。由于堆栈向上生长，因此通常程序段首地址在 ROM 或 RAM 的地址低端，堆栈段首地址在 RAM 的地址高端，以达到 RAM 利用率最高。

5.1.3 陷阱与中断

BM3803FMGRH 支持异常处理机制，中断陷阱一般称为中断，其他陷阱一般称为陷阱。与中断和陷阱相关的配置寄存器及相关设置如下：

1、处理器状态寄存器 PSR

ET：陷阱和中断使能控制

PIL：设置处理器响应中断的级别

2、浮点状态寄存器 FSR

TEM：浮点陷阱使能掩码

FTT：浮点陷阱类型

AEXC：已产生浮点异常

CEXC：当前浮点异常

3、陷阱基址寄存器 TBR

4、中断控制寄存器组

中断级别/屏蔽控制寄存器（0x8000-0090）

中断请求寄存器（0x8000-0094）

中断强制寄存器（0x8000-0098）

中断清除寄存器（0x8000-009C）

5、陷阱/中断向量表（即中断/陷阱程序入口）

5.1.3.1 陷阱控制

1、*ET* 和 *PIL* 控制

陷阱和中断要正常响应，PSR 中的 *ET* 位必须为 1。

ET=1，如果发生陷阱，处理器按照陷阱处理程序响应陷阱。若同时产生多个陷阱，IU 根据陷阱优先级区分陷阱响应次序，只响应最高优先级的陷阱，其他同时产生的较低优先级陷阱被忽略。*ET*=0，如果发生陷阱，处理器进入错误模式。

ET=1，如果发生中断，处理器比较 PSR 中的 *PIL* 字段和**中断级别/屏蔽控制寄存器**中的 *ILEVEL* 和 *IMASK* 字段，如果中断请求的优先级别 *ILEVEL* 和中断序号最高，且中断序号大于 *PIL* 字段值，且**中断级别/屏蔽控制寄存器**中的 *IMASK* 字段为 1，即未屏蔽此中断，则处理器响应该中断。当处理最高优先级中断时，低优先级中断请求保持在**中断请求寄存器**中。*ET*=0，所有中断被屏蔽。

2、*TEM* 控制

如果浮点陷阱发生时，相应**浮点状态寄存器** FSR 中的 *TEM* 字段置 1，则使能浮点陷阱，此时若 *ET*=1，则处理器按照浮点陷阱处理程序响应浮点陷阱，且引起陷阱的指令的目的寄存器、FSR 的 *FCC* 和 *AEXC* 字段保持不变。如果 *TEM* 字段置 0，则屏蔽浮点陷阱，此时，产生的陷阱会被记录在 FSR 的 *AEXC* 字段中，且引起陷阱的指令的目的寄存器、FSR 的 *FCC* 字

段被更新。

5.1.3.2 陷阱识别

陷阱基址寄存器 TBR 中的陷阱基址字段 *TBA* 保存陷阱向量表基地址的高 20 位。

当陷阱发生时，与之唯一对应的识别码写入 TBR 的 8 位 *TT* 字段，程序以 TBR 的值为目的地址进行跳转。由于 TBR 的低四位为 0，因此陷阱向量表的每一入口包含陷阱处理程序的前 4 条指令（16 字节）。

5.1.3.3 陷阱处理

如果 *ET*=1，陷阱响应时处理器出现以下改变：

- 禁止陷阱：ET←0。
- 保存当前用户/管理模式：PS←S。
- 转换为管理模式：S←1。
- 进入一个新的寄存器窗口：CWP←((CWP-1) modulo 8)[注意：不检查窗口是否溢出]。
- 在新窗口的本地寄存器中保存当前 PC 和 nPC 值：%l1←PC，%l2←nPC。
- 除了复位陷阱和错误模式，TBR 的 tt 字段写入陷阱或中断类型。
- 若陷阱是复位陷阱，则程序分支到地址 0x0：PC←0，nPC←4；若陷阱不是复位陷阱，则程序分支到陷阱向量表：PC←TBR，nPC←TBR+4。

如果 ET=0：

- 发生陷阱，则处理器进入错误模式。
- 发生中断，则处理器忽略该中断。

5.1.3.4 陷阱与中断的区别

- 触发位置

  陷阱发生在符合陷阱触发条件的指令运行期间；

  中断发生在一条指令运行完毕之后。

- 控制方式

陷阱只受 *ET* 或 *ET+TEM* 自动控制；

中断不但受 *ET* 控制，而且受 *PIL* 和四个中断控制寄存器控制。

5.1.4 Cache 刷新

BM3803FMGRH 可以通过指令刷新 Cache，刷新完成后，Cache 的 tag 位全部清零。

5.1.4.1 刷新方法

- **指令 Cache 刷新（以下三种方法均能实现）**

1、执行 flush 指令；

2、**Cache 控制寄存器**的 21 位写 1；

3、执行 asi=0x5 的 sta 指令（写任意地址）。

- **数据 Cache 刷新（以下三种方法均能实现）**

1、执行 flush 指令；

2、**Cache 控制寄存器**的 22 位写 1；

3、执行 asi=0x6 的 sta 指令（写任意地址）。

- **刷新状态判断**

Cache 控制寄存器的 14、15 位为 1 分别表示数据 Cache、指令 Cache 正在刷新，为 0 表示刷新结束或没有刷新动作。

5.1.4.2 刷新步骤

1、关闭 Cache；

2、刷新 Cache；

3、判断 Cache 刷新状态，等待刷新结束；

4、打开 Cache。

5.2 BM3803FMGRH 软件容错设计

BM3803FMGRH 片内 Regfile、Cache 和外部存储器控制器三部分包含系统级容错能力。本节介绍这三个模块容错的使用方法。

5.2.1 Regfile EDAC

5.2.1.1 工作过程

BM3803FMGRH 设计中对 Regfile 数据采用 EDAC 技术进行保护，可以纠正一位错，检测两位错及某些多位错。

Regfile 由 32 位数据和 7 位校验码组成。向 Regfile 写数据时，根据写入数据计算生成校验位，将数据和校验位一同写入 Regfile。从 Regfile 读取数据时，如果检测到可以纠正的错误，流水线断流，纠正错误，将正确的数据写入发生错误的寄存器，之后，重启流水线，从断流位置重新运行程序；如果检测到不可纠正的错误，引起 0x20 陷阱。

5.2.1.2 寄存器说明

控制 Regfile EDAC 功能的寄存器有两个，分别是 ASR16 和 ASR17。详细说明如下所示。

ASR16：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB			R						TCB							CNT															
r/w			r						r/w							r															
x	x	x	0						x							x															

位号	名称	描述
31:29	CB	000或001或011或100或110或111 = 错误计数 010 = 错误计数清零 101 = 造错使能，即对及数据位和校验位造错
22:16	TCB	对7位校验位造错，某位置1表示对该位造错，置0表示不造错
15:0	CNT	记录发生一位错的次数

ASR17：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCB																															
r/w																															
x																															

位号	助记符	描述
31:0	DCB	对32位数据位造错，某位置1表示对该位造错，置0表示不造错。

基本操作步骤：

◆ 纠、检错

1、初始化开始处配置 ASR16 和 ASR17，清“校验位造错”和“数据位造错”字段。

2、根据需要使能或禁能“错误计数”。若希望记录一位错发生次数，则首先清错误计数，再使能错误计数。

◆ 造错

1、初始化开始处配置 ASR16 和 ASR17，清“校验位造错”和“数据位造错”字段。

2、配置“校验位造错”和“数据位造错”字段，在希望造错的相应位置写“1”。

3、ASR16“造错控制”字段使能造错。

4、向希望造错的寄存器写入正确数据。此时，该寄存器中保存的是已经造好错误的数据。

5、关闭造错功能。

⚠注意

1、ASR16 上电初始值是不定值，使用时必须先设置 ASR16 不使能造错，再读/写 Regfile 寄存器。严格禁止 ASR16 使能造错时，读/写 Regfile 寄存器。

2、Regfile EDAC 功能不能被关闭。

3、引导程序起始处关闭 Regfile 造错机制。

4、不能对已经造错的程序段在实际运行前进行读取操作，而且造错的程序段不能调试。

5、对数据位和校验位最多共造 2 位错，如果造错多于 2 位，则很可能引起程序错误。

5.2.2 Cache 奇偶校验

5.2.2.1 工作过程

BM3803FMGRH 设计中对 Cache 中数据和 TAG 采用奇偶校验技术进行保护，可以检测奇位错。

指令和数据 Cache 的数据和 TAG 由数据位和校验位组成。向 Cache 写数据时，根据写入数据计算生成校验位，将数据和校验位一同写入 Cache 存储体。从 Cache 读取数据时，如果检测到错误，则直接从外部存储器中该地址处读取数据送往 IU，并把正确的数据写入 Cache。

5.2.2.2 寄存器说明

控制 Cache 奇偶校验功能的寄存器有三个，详细说明如下所示。

（1）Cache 容错控制寄存器（CCR1）

Address = 0x80000110

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VER			保留															CMEEN			CNTRST	CPSEL		保留				EPOS			
r			r															r/w			r/w	r/w		r				r/w			
0	0	1	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0

位号	助记符	描述
31:29	VER	Cache寄存器版本号，定义为001
13:11	CMEEN	Cache造错使能标志位 ‘011’=造错开启 否则造错无效
10	CNTRST	Cache校验计数器清零标志位 1 =清零所有错误计数器
9:8	CPSEL	Cache造错目标区选择位 ‘00’=指令Cache的data区 ‘01’=指令Cache的tag区 ‘10’=数据Cache的data区 ‘11’=数据Cache的tag区
3:0	EPOS	Cache造错位置的高4位 若选择为data区，对应于数据的4个校验位；若选择为tag区，对应于tag的4个校验位

（2）Cache 造错寄存器（CCR2）

Address = 0x80000114

当 Cache 造错使能，且目标区域为 Data 区时：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPOS																															
r/w																															
0																															

位号	助记符	描述
31:0	EPOS	造错位置标识低32位

当 Cache 造错使能，且目标区域为 Tag 区时：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	TAG																			VALID				保留							
r/w	r/w																			r/w				r							
0	0																			0	0	0	0	0							

位号	助记符	描述
31	LOCK	Cache LOCK位
30:12	TAG	Cache TAG区域数据
11:8	VALID	无效位

（3）Cache 错误计数寄存器（CCR3）

Address = 0x80000118

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTAGCNT						DDATACNT										ITAGCNT						IDATACNT									
r						r										r						r									
0						0										0						0									

位号	助记符	描述
31:26	DTAGCNT	DCache Tag 错误计数器
25:16	DDATACNT	DCache Data错误计数器
15:10	ITAGCNT	ICache Tag 错误计数器
9:0	IDATACNT	ICache Data错误计数器

基本操作步骤：

◆ 检错

1、对 Cache 造错寄存器 CCR2 和 Cache 错误计数寄存器 CCR3 清零。

2、对 Cache 容错控制寄存器 CCR1 写“0x0”，关闭造错。

◆ 造错

1. 关闭 Cache。
2. 配置 Cache 容错寄存器 CCR1：

（1）配置 *CMEEN*(Cache Make Error Enable)位为"011"：开启 Cache 造错功能；

（2）配置 *CPSEL*(Cache Parity Select)位，选择需要造错的目标区域："00"表示目标区域为指令 Cache 的 data 区，"01"表示目标区域为指令 Cache 的 tag 区，"10"表示目标区域为数据 Cache 的 data 区，"11"表示目标区域为数据 Cache 的 tag 区。

3. 根据造错需求配置 Cache 造错寄存器 CCR2 和 Cache 容错控制寄存器 CCR1 的 EPOS(Error Position)位。
4. 使能 Cache。
5. 从外部存储器中读取数据，造错后的数据自动加载到 Cache 中，完成造错。
6. 关闭 Cache 造错功能。

⚠注意

1、Cache 奇偶校验功能总是使能。

2、造错时请通过一般方式（LOAD 和 STORE）操作 Cache，不要使用 ASI 指令造错。

3、造错时请仔细考虑有多少条指令会被缓存进 Cache，特别是指令 Cache。由于读取指令的连续性，往往难于对单独某一条指令进行造错。

5.2.3 外部存储器控制器 EDAC

5.2.3.1 概述

BM3803FMGRH 处理器支持外部存储器纠错检错功能，能够纠正一位错，检测两位错误及某些多位错。支持 EDAC 的外部存储器类型如表 A5-2-1 所示。

表 A5-2-1 支持 EDAC 的外部存储器类型

存储器类型	片选个数	数据位宽	支持 EDAC 功能
prom	2	32	是
		16	否
		8	是
i/o	1	32、16、8	否
sram	5	32	是
		16	否
		8	是
sdram	2	32	是

表 A5-2-2 32 位宽和 8 位宽 prom 检错纠错配置类型

检错纠错配置	对错误的处理	
	对单位错的处理	对二位错的处理
不检错，不纠错	不检错，不纠错	
既检错，又纠错	检错并纠错	检错，不纠错

表 A5-2-3 32 位宽 sram、8 位宽 sram、32 位宽 sdram 检错纠错配置类型

检错纠错配置	对错误的处理	
	对单位错的处理	对二位错的处理
不检错，不纠错	不检错，不纠错	
只检错，不纠错	检错，不纠错	
既检错，又纠错	检错并纠错	检错，不纠错

控制外部存储器 EDAC 功能的寄存器有三个：MECFG1、MECFG2、MECFG3，详见附录 3.6。

5.2.3.2 RAM 纠错检错配置说明

表 A5-2-4 MECFG1 中 RAM 纠错检错配置说明表

配置名称	EDAC 使能	EWB 使能	ERB 使能	能否检错	能否纠错	RAM 类型	写入正确的或造错的数据位和校验位	能否读出校验位

							数据位	校验位	
mecfg1_1	有效	无效	无效	能	能	a	正确	正确	不能
						b	正确	正确	不能
mecfg1_2	有效	无效	有效	否	否	a	正确	正确	能
						b	正确	正确	能
mecfg1_3	有效	有效	无效	能	否	a	正确(造错)	正确(造错)	不能
						b	正确	正确(造错)	不能
mecfg1_4	有效	有效	有效	否	否	a	正确(造错)	正确(造错)	能
						b	正确	正确(造错)	能
mecfg1_5	无效	无效	无效	否	否	a	正确	正确	不能
						b	正确	不写校验位	不能
mecfg1_6	无效	无效	有效	否	否	a	正确	正确	能
						b	正确	不写校验位	能
mecfg1_7	无效	有效	无效	否	否	a	正确(造错)	正确(造错)	不能
						b	正确	正确(造错)	不能
mecfg1_8	无效	有效	有效	否	否	a	正确(造错)	正确(造错)	能
						b	正确	正确(造错)	能

注：(1)“造错”是指在进行字的写操作时，可以通过 mecfg2 或 mecfg3 自动造错。

(2)a 表示 32 位数据宽度 SRAM 和 SDRAM；b 表示 8 位数据宽度 SRAM。

mecfg1_1 是对 RAM 既检错又纠错配置。

mecfg1_3 是对 RAM 只检错不纠错配置。

⚠注意 当 mecfg1 配置为 mecfg1_3 时，如果 mecfg2 和 mecfg3 配置值为非零，在进行字的写操作时，会对数据位和校验位造错，因此，在 mecfg1 配置为 mecfg1_3 时，要提前配置 mecfg2 和 mecfg3 为全零。

5.2.3.3 RAM 纠错检错配置基本操作过程

5.2.3.3.1 RAM 既检错又纠错配置基本操作过程

1、mecfg2、3 清零。

2、配置 mecfg1 为 mecfg1_1。

5.2.3.3.2 RAM 只检错不纠错配置基本操作过程

1、mecfg2、3 清零。

2、配置 mecfg1 为 mecfg1_3。

5.2.3.3.3 32 位数据宽度 SRAM 和 SDRAM 造错基本操作过程

1、根据造错的需要来配置 mecfg2 和 mecfg3，准备对数据位或校验位中的某位翻转。

2、配置 mecfg1 成 mecfg1_7。

3、向外部存储器进行字的写操作，则写入的数据位和校验位将按照 mecfg2 和 mecfg3 中值为“1”的位翻转。

5.2.3.3.4 8 位数据宽度 SRAM 造错基本操作过程

1、根据造错的需要来配置 mecfg3，准备对校验位中的某位翻转，mecfg2 清零。

2、配置 mecfg1 成 mecfg1_7。

3、向外部存储器进行字的写操作，则写入的数据对应的校验位将按照 mecfg3 中值为“1”的位翻转。

4、配置 mecfg1 为 mecfg1_5。

5、配置 mecfg3 清零。

6、向外部存储器写入错误的数据。

⚠注意

1、对 32 位宽 PROM 空间应用要考虑到校验位管脚只有 7 位。8 位宽 PROM 空间不支持写校验位。

2、只能读出一个 bank 的四分之一空间里的校验位。特别注意：读取校验位的地址和程序运行的地址应处在同一 bank 的不同四分之一中，否则校验码会被连续读出的指令的校验码冲掉。

3、发生 trap 0x01、trap 0x09、trap 0x2b 时，同时产生 1 号中断，而且将错误状态保存在 **AHB 失效地址寄存器**（0x8000-000c）和 **AHB 状态寄存器**（0x8000-0010）中。在错误发生后，需要软件将 **AHB 状态寄存器**（0x8000-0010）中的错误状态位清除，否则在下一次

发生错误时，不能正常保存。

5.3 BM3803FMGRH 中断处理

BM3803FMGRH 处理器可处理多个外部中断（包括 TT=0x11 至 0x1F），硬件不支持嵌套，若应用程序需要实现中断嵌套，则必须手动修改中断处理程序的%psr.PIL 位和**中断级别/屏蔽寄存器**的 *IMASK* 位，软件实现中断嵌套，并且根据中断 LEVEL 分级与否，中断嵌套的软件实现不同。

5.3.1 中断处理流程

当满足中断响应的条件时，IU 响应中断请求，进行中断处理，硬件自动完成一些操作，如：窗口旋转 CWP=((CWP-1)%8)，%psr.ET=0（禁止其它中断），%psr.PS←%psr.S，%l1←PC，%l2←nPC，%TBR 变化，系统跳转等。

软件进行中断处理时，可直接使用当前新窗口的局部寄存器，其它寄存器在未保存之前最好不要更改，主要的步骤包括：

记录中断时现场的部分信息（%psr 的值，%tbr 的值）。

判断中断时窗口是否溢出：由于寄存器窗口的特殊结构，中断时系统自动将窗口 cwp-1，用户需要先判断当前窗口是否溢出，如溢出则需要先保存窗口。

保存中断现场：保存 CPU 内核的寄存器：状态寄存器，IU 中的全局寄存器、输入寄存器，以及 FPU 相关的寄存器（可根据需要保存），以及应用程序认为需要保存的信息等。

保存原中断设置，更新当前中断服务时的中断设置：根据中断的优先级（*LEVEL* 和序号），通过中断控制器和%psr.PIL 设置更新，实现中断嵌套和优先级功能。

打开中断处理使能位：%psr.ET=1，允许用户中断处理中可以发生其它 trap（包括外部中断）。

跳转到用户中断处理程序：将当前的中断号传递给用户中断处理程序，并跳转到用户处理程序（一般采用 C 语言完成）。

运行用户处理程序（可处理一个或多个中断源的中断）。

从用户中断处理程序返回，恢复现场：设置%psr.ET=0（禁止中断），进行现场（原中断设置、IU 寄存器、浮点现场等、应用程序信息）的恢复。

判断窗口是否溢出（与前面的保存窗口相对应的恢复窗口处理）。

返回到被中断的代码。

中断处理流程示意图如图 A5-3-1 所示。

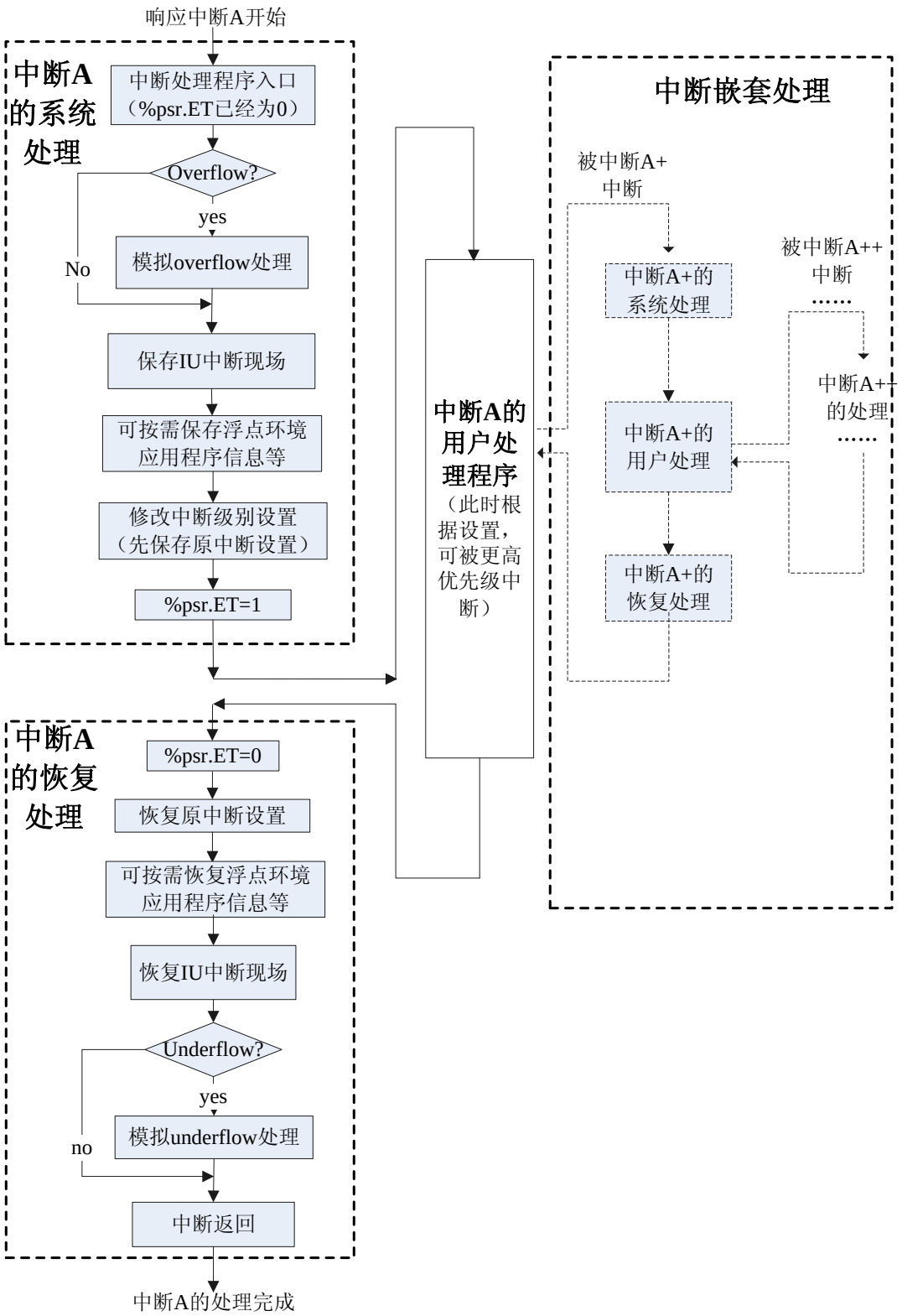


图 A5-3-1 中断处理流程示意图

中断不嵌套，就是中断嵌套处理中，去除“中断嵌套处理部分”，由“中断 A 的系统处理”到“中断 A 的用户处理程序”再到“中断 A 的恢复处理”。

5.3.2 中断处理中的中断级别设置

中断处理程序中，在准备进入当前中断的用户处理函数，使能%psr.ET 之前，需要对系统可中断级别做出设置。

当中断 LEVEL 不分级时，中断程序可嵌套和不可嵌套两种情况如下：

- 中断程序可嵌套（低优先级中断处理可被高优先级中断打断/抢占）时，把%psr.PIL 字段设置成当前中断号，比当前中断优先级高的中断可以打断/抢占当前的处理，低于或等于当前优先级的中断暂不响应，服务完该中断后再响应低优先级中断；
- 中断程序不可嵌套时，把%psr.PIL 字段设置成 0xF 即可，这样其它中断不可打断当前中断。

当中断 LEVEL 分 0、1 两级时，中断请求的数量影响了嵌套判断过程。当只有一个中断请求时，处理器根据当前中断处理程序的%psr.PIL 值判断是否可以嵌套；当有两个或两个以上中断请求时，中断控制器先根据中断 LEVEL 值和中断序号，送出级别最高的一个中断，然后处理器根据当前中断处理程序的%psr.PIL 设置值判断此中断是否可以嵌套。此时，若使用%psr.PIL 屏蔽机制实现中断嵌套，可能会导致当前中断 N（LEVEL=1）的处理程序被那些 LEVEL=0 级，但中断序号比 N 高的中断打断/抢占；也可能导致当前正在处理的 LEVEL=0 级的中断 N，屏蔽那些 LEVEL=1 但序号比 N 小的中断。

当中断 LEVEL 分 0、1 两级时，中断程序可嵌套和不可嵌套两种情况如下：

- **中断程序可嵌套（低优先级中断处理可被高优先级中断打断/抢占）时，采用更新中断屏蔽寄存器的方法实现中断嵌套，即在处理当前中断时，%psr.PIL 固定设置为 0，新设置的中断屏蔽寄存器屏蔽比当前中断优先级低（LEVEL 相等但序号比当前中断小，以及 LEVEL 比当前中断 LEVEL 低）的中断源。原中断屏蔽设置值保存在当前现场中，等到用户中断程序处理完毕后恢复即可。**
- **中断程序不可嵌套时，把%psr.PIL 字段设置成 0xF 即可，这样其它中断**

不可打断当前中断。

5.4 引发 Trap0x2b 陷阱条件

1. 当**外部存储器配置寄存器** MCFG1(0x8000-0000)的[11]位（PROM write enable）配置为“0”时，表示 PROM 写操作不使能，如果出现对 PROM 进行写操作，则会产生 trap0x2b 陷阱。

2. 当**写保护寄存器**配置成写保护模式时，即**写保护寄存器** WPR1(0x8000-001C)、WPR2(0x8000-0020)的[31]位（写保护使能）配置为“1”时，如果出现对写保护地址进行写操作，则会产生 trap0x2b 陷阱。

3. 当**外部存储器容错配置寄存器**配置为 EDAC 使能时，即**外部存储器容错配置寄存器** MECFG1 (0x8000-0100)的[31]位（SRAM bank1 EDAC 使能）、[30]位（SRAM bank2 EDAC 使能）、[29]位（SRAM bank3 EDAC 使能）、[28]位（SRAM bank4 EDAC 使能）、[27]位（SRAM bank5 EDAC 使能）、[26]位（PROM EDAC 使能）、[19]位（SDRAM EDAC 使能）配置为“1”时，如果外部存储器访问地址中的数据有错误(在纠错模式下有 2 位错误，在检错模式下有 1 位或 2 为错误)，在字节写或半字写操作情况，会产生 trap0x2b 陷阱。

4. 当配置**外部存储器配置寄存器** MCFG1(0x8000-0000)的[25]位 bus error enable 为“1”时，表示总线错误使能，在对外部存储器进行写操作时，如果系统输入管脚 bexcn 为“0”时，会产生 trap0x2b 陷阱。

5.5 引发 Trap0x09 陷阱条件

1. 当**外部存储器容错配置寄存器**配置为 EDAC 使能时，即**外部存储器容错配置寄存器** MECFG1 (0x80000100)的[31:27]位（各位分别是 SRAM bank1-bank5 的 EDAC 使能）、[26]位（PROM EDAC 使能）、[19]位（SDRAM EDAC 使能）配置为“1”时，如果外部存储器访问地址中的数据有错误(在纠错模式下有 2 位错误，在检错模式下有 1 位或 2 位错误)，在读操作情况，会产生 Trap0x09 陷阱。

2. 当配置**外部存储器配置寄存器** MCFG1(0x80000000)的[25]位 bus error enable 为“1”时，表示总线错误使能。在对外部存储器进行读操作时，如果系统输入管脚 bexcn 为“0”时，会产生 Trap0x09 陷阱。

3. 当PCI配置为主模式且PCI总线上没有PCI从设备时，对AHB-PCI地址空间进行读操作，即 AHB-PCI 不可预取地址空间、AHB-PCI 可预取地址空间、AHB-PCI I/O 地址空间，会产生 trap09 陷阱。

附录 6 SPARC V8 介绍

6.1 SPARC V8 概述

SPARC 是一种精简指令集（RISC）指令集架构。根据性能、价格等不同需求，SPARC 体系架构有不同的实现方式。SPARC 的设计目标是优化编译器、简化流水线的硬件实现。SPARC 架构产品的优势在于执行速度和开发周期。SPARC 主要包括下面几个特性：

- 32 位线性地址空间，理论上可寻址 4GB 空间；
- 简洁的指令格式：均为 32 位宽，在内存中以 32 位对齐存储，诸多的指令只有三种存储格式，不同的存储格式其操作码和寄存器地址的存放位置固定；
- 只有 load 和 store 命令可以访问内存；
- 少数几种寻址模式： 寄存器+寄存器、寄存器+立即数；
- 三寄存器寻址： 一般都是将两个寄存器中的内容作相应运算后将结果放到第三个寄存器中；
- 一个大的窗口寄存器文件：在任一时刻，每个程序可以访问 8 个全局整数寄存器和一个由 24 个寄存器组成的寄存器窗口。所谓的窗口寄存器可以理解为存储着进程参数、局部变量和返回地址的存储区；
- 独立的浮点寄存器文件：可由软件设置成单（32 位）、双（64 位）、四（128 位）三种不同精度；
- 延迟控制转移特性：指示 CPU 延迟一条指令执行或跳过下一条指令，由控制转移指令的“annul”位决定执行还是跳过；
- 快速的陷阱处理：陷阱的处理存放在向量表中，陷阱处理时将旋转到下一个新的寄存器窗口。

6.2 数据类型

SPARC 架构可以识别三种基本的数据类型（大端存储模式）：

有符号整数：8、16、32、64 位长（最高位为符号位）。

无符号整数：8、16、32、64 位长。

浮点数：32、64、128 位长。

整数和浮点数的存储格式分别如图 A6-1-1 和图 A6-1-2 所示。

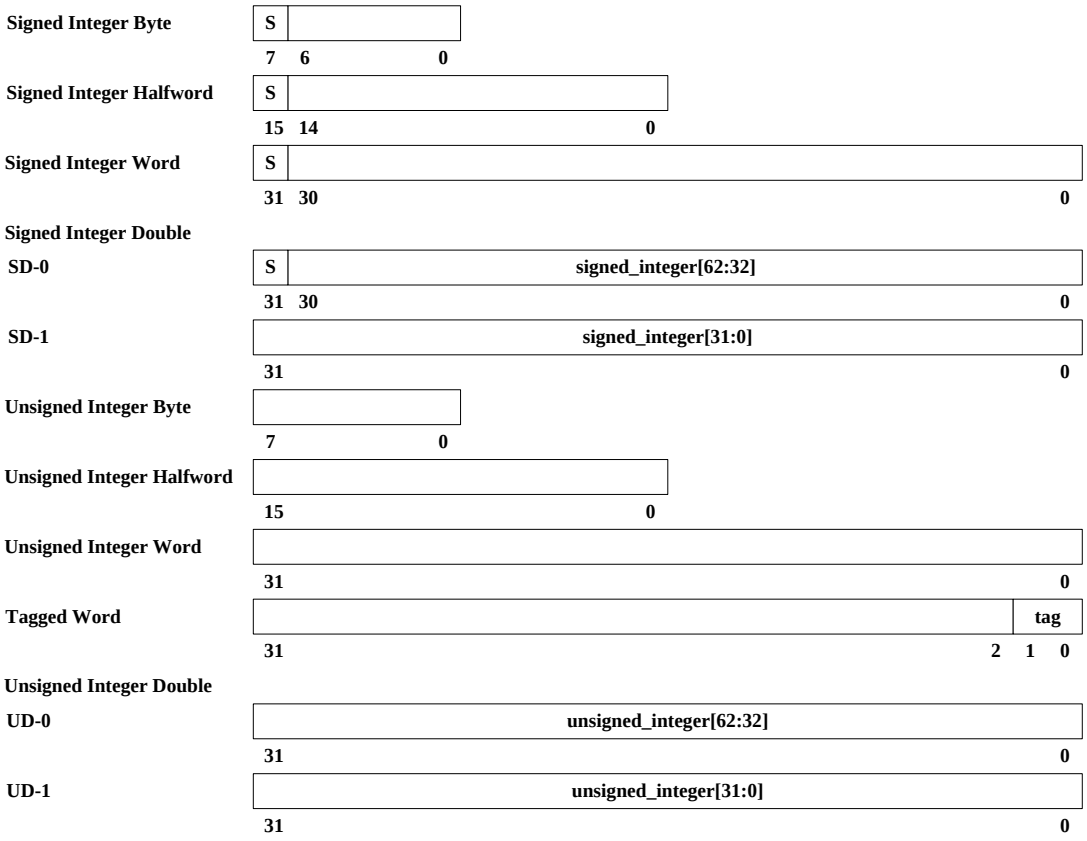


图 A6-1-1 整数存储格式

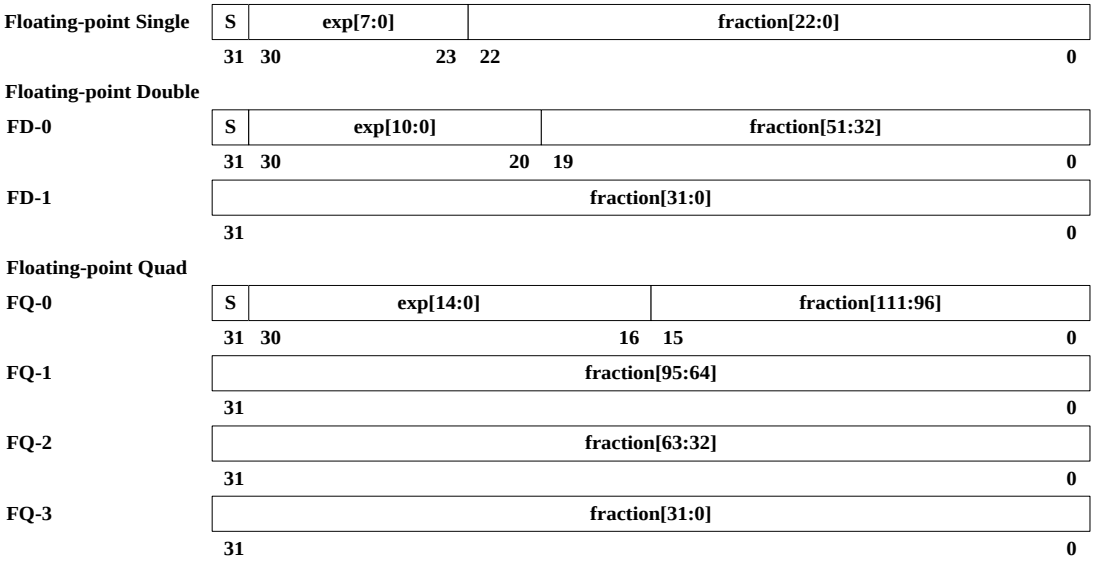


图 A6-1-2 浮点数存储格式

6.3 寄存器

处理器的寄存器包括两种类型：通用寄存器（或叫工作区数据寄存器）和控制/状态寄存器。

通用寄存器：IU 和 FPU 的通用寄存器分别叫 r 寄存器和 f 寄存器。

控制状态寄存器：IU 的控制状态寄存器包括处理器状态寄存器 PSR、窗口无效屏蔽寄存器 WIM、陷阱基址寄存器 TBR、乘/除寄存器 Y、程序计数器 PC/nPC 和辅助状态寄存器 ASR。

FPU 的控制/状态寄存器是**浮点处理器状态寄存器 FSR**。

6.3.1 IU 寄存器

IU 包括 40 到 520 个通用 32 位 r 寄存器，这些 r 寄存器被分成 8 个全局寄存器，一系列由 16 个寄存器组成的寄存器组，这 16 个寄存器又可分成 8 个输入寄存器和 8 个局部寄存器。

Windowed Register Address	*r* Register Address
in[0] – in[7]	r[24] – r[31]
local[0] – local[7]	r[16] – r[23]
out[0] – out[7]	r[8] – r[15]
global[0] – global[7]	r[0] – r[7]

图 A6-3-3 窗口编址

窗口 r 寄存器：任一时刻，一条指令可以访问 8 个全局寄存器和 24 个窗口寄存器，每个寄存器窗口由 8 个输入寄存器、8 个局部寄存器和共享相邻寄存器组的 8 个输入寄存器（作为本窗口的输出寄存器）组成。当前窗口由 CWP 指针指定。

窗口的重叠：窗口的重叠指定是当前的窗口可以访问 CWP+1 窗口的输出寄存器，作为本窗口的输入寄存器；当前窗口的输出寄存器可以作为 CWP-1 窗口的输入寄存器，每个窗口的 8 个 local 寄存器是唯一的。窗口按一个环顺序排列，也就是说最后一个窗口指针加 1 指向 0 窗口。另外 save 命令将使 CWP 减 1，restore 命令将使 CWP 加 1。

特殊 r 寄存器：有四个特殊的 r 寄存的用法是固定的：（1）如果 r[0]作为源操作数或是作为 store 的目的地址，那么 r[0]将被当作 0 来读写，如果 r[0]作为目的操作数，那么相应的操作将被忽略。（2）call 指令将把自身的地址写进 r[15]寄存器。（3）陷阱发生的时

候：PC 和 nPC 中的内容将会自动复制到 r[17]和 r[18]当中。

6.3.2 FPU 寄存器

FPU 总共包含 32 个 32 位浮点 f 寄存器，标号从 f[0]到 f[31]，f 寄存器可以通过浮点数操作指令访问。与 r 寄存器不同的是这 32 个 f 寄存器都可以访问。另外，一个 f 寄存器可以存放 1 个单精度浮点数，对于双精度和四精度的浮点数的存放，则分别需要两个和四个相邻的浮点寄存器。

6.3.3 寄存器推荐使用方法汇总

表 A6-3-1 寄存器推荐使用方法汇总表

in	%i7	(%r31)	返回地址 - 8
	%fp，%i6	(%r30)	堆指针
	%i5	(%r29)	输入参数 6
	%i4	(%r28)	输入参数 5
	%i3	(%r27)	输入参数 4
	%i2	(%r26)	输入参数 3
	%i1	(%r25)	输入参数 2
	%i0	(%r24)	输入参数 1 / 向调用函数返回的值
local	%l7	(%r23)	本地变量 7
	%l6	(%r22)	本地变量 6
	%l5	(%r21)	本地变量 5
	%l4	(%r20)	本地变量 4
	%l3	(%r19)	本地变量 3
	%l2	(%r18)	本地变量 2
	%l1	(%r17)	本地变量 1
	%l0	(%r16)	本地变量 0
out	%o7	(%r15)	临时值 / CALL 指令的地址
	%sp，%o6	(%r14)	栈指针
	%o5	(%r13)	输出参数 6
	%o4	(%r12)	输出参数 5
	%o3	(%r11)	输出参数 4
	%o2	(%r10)	输出参数 3
	%o1	(%r9)	输出参数 2

	%o0	(%r8)	输出参数 1 / 从被调函数返回的值
global	%g7	(%r7)	全局变量 7 (SPARC ABI：保留)
	%g6	(%r6)	全局变量 6 (SPARC ABI：保留)
	%g5	(%r5)	全局变量 5 (SPARC ABI：保留)
	%g4	(%r4)	全局变量 4 (SPARC ABI：全局reg型变量)
	%g3	(%r3)	全局变量 3 (SPARC ABI：全局reg型变量)
	%g2	(%r2)	全局变量 2 (SPARC ABI：全局reg型变量)
	%g1	(%r1)	临时值
	%g0	(%r0)	0
state	%y		Y寄存器 (用于乘/除法)
	(*icc* field of %psr)		整数状态位
	(*fcc* field of %fsr)		浮点数状态位
floating point	%f31		浮点值
	:		:
	:		:
	%f0		浮点值

6.4 指令

指令由 CPU 从内存中读取并执行，PC 和 nPC 分别提供当前指令的地址和下一条指令的地址。指令被编码成三种格式存储，按功能可以分成 6 类，有 72 条基本的指令操作。

6.4.1 指令格式

SPARC 架构指令存储的格式可以分成如下三种，各位的含义如下图 A6-4-1 所标。

Format 1(op=1): CALL

op	disp30
31	29 0

Format 2(op=0): SETHI&Branches(Bicc,FBfcc,CBccc)

op	rd	op2	imm22
op	a cond	op2	disp22
31	29 28	24	21 0

Format 3(op=2 or 3): Remaining instructions

op	rd	op3	rs1	i=0	asi	rs2
op	rd	op3	rs1	i=1	simm13	
op	rd	op3	rs1	opf		rs2
31	29	24	18	13 12	4	0

图 A6-4-1 指令存储位含义

6.4.2 指令分类

SPARC 指令集可以分成 6 类，分别为：

1）加载/存储指令 load/store：

唯一可以访问内存的指令。加载/存储指令利用两个 r 寄存器或是一个 r 寄存器和一个符号扩展的 13 位立即数来计算字节对齐的 32 位内存地址。

2）整数算术指令：

该类指令通常都是三寄存器地址指令，即对两个寄存器的数据进行运算。然后将结果存放到第三个寄存器中。源操作数一个恒为 r 寄存器的数据，另一个源操作数依赖于指令的第 i 位，可能为 r 寄存器的数据（i 位为 0）也可能为符号扩展的 13 位立即数（i 位为 1）。

3）控制传输指令

该类指令修改 nPC 的值，也就是改变程序的执行路线。有五类基本的控制转移命令类型：

条件转移 Bicc、FBfcc

调用 call and link

跳转 jump and link

陷阱返回 RETI

陷阱 Ticc

4）读写控制寄存器指令

该类指令访问程序可见的状态寄存器，将状态寄存器的值读入 r 寄存器或是将 r 寄存器的值写入状态寄存器。对于读/写辅助状态寄存器的指令，如果访问的寄存器是特权寄存器，那么该条指令将是特权指令。

5）浮点数操作指令

除了浮点数比较指令（修改 FSR 的 fcc 位）和浮点数交换命令（利用一个源操作数和一个目标操作数），其他的浮点操作数指令基本也都是三操作数指令。 所有的浮点操作数指令（Fpop）都清除 *FTT* 位、置位 *CEXC*。

6）协处理器操作

协处理器操作指令由附加的协处理器执行。

6.5 陷阱

6.5.1 精确陷阱

精确陷阱由特定的指令产生，在程序状态被该指令修改之前由 CPU 捕获，当精确陷阱发生的时候，有几个条件或是状态必须要保证和保持：

a）PC 指向产生该陷阱的指令，nPC 指向捕获该陷阱后 CPU 将要执行的指令地址；

b）引起陷阱的指令之前的一条指令将完全执行；

c）产生该陷阱的指令的后面一条指令不执行。

6.5.2 中断陷阱

中断陷阱受 PSR 的处理器中断优先级 *PIL* 位和陷阱使能 *ET* 位的联合控制。中断陷阱的产生可能是以下的原因造成：

a）外部中断请求；

b）与之前指令无关的异常；

c）先前执行指令引起的异常。

6.5.3 延迟陷阱

延迟陷阱也是由特定的指令产生的，与精确陷阱不同的是，延迟陷阱的发生可能在程序可见的状态改变之后。

6.5.4 陷阱控制

1. ET 和 PIL 控制

陷阱要正常工作，PSR 中的 ET 位必须为 1。

当 ET=1 时，IU 根据陷阱表区分陷阱的优先次序，只响应最高优先级的陷阱。对中断响应来说，IU 比较中断请求等级与 PSR 中 PIL 字段。如果中断请求等级大于或等于 PIL 字段值，且未屏蔽此中断，则处理器响应该中断请求。当处理最高优先级中断时，低优先级中断请求保持在中断请求寄存器中。

当 ET=0 时，如果发生陷阱，处理器进入错误模式。

2. TEM 控制

如果 TEM 字段的特定位置 1，与之对应的浮点异常发生时，允许引起浮点异常陷阱（fp_exception）。

如果 TEM 字段的的特定位置 0，不允许引起浮点异常陷阱。此时，产生的异常会被记录在 FSR 的“产生浮点异常”字段中（AEXC）。

如果没有屏蔽浮点异常陷阱，则引起浮点陷阱时，被中断执行的指令的目的寄存器、FCC 和 AEXC 字段保持不变。如果浮点异常被屏蔽，引起陷阱的指令的目的寄存器的 FCC 和 AEXC 字段被更新。

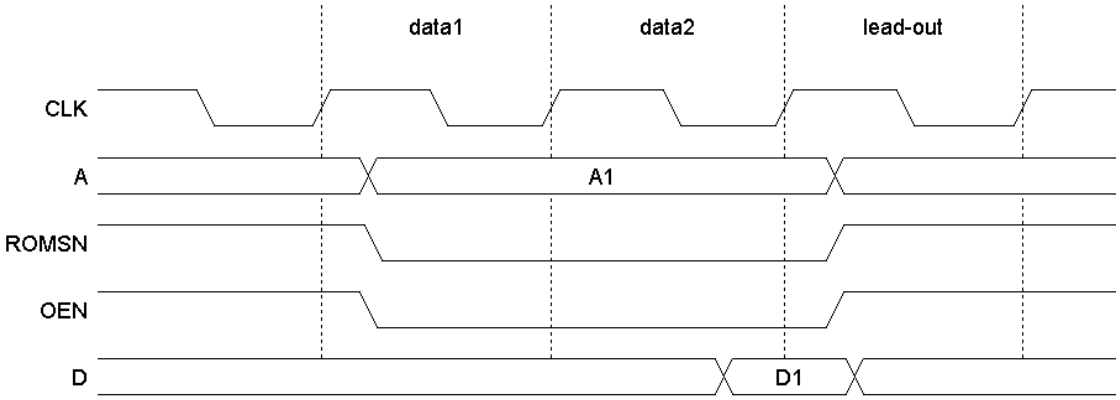
6.5.5 陷阱识别

陷阱基地址寄存器 TBR 中的“陷阱基地址”字段 *TBA* 保存陷阱向量表基地址的高 20 位。

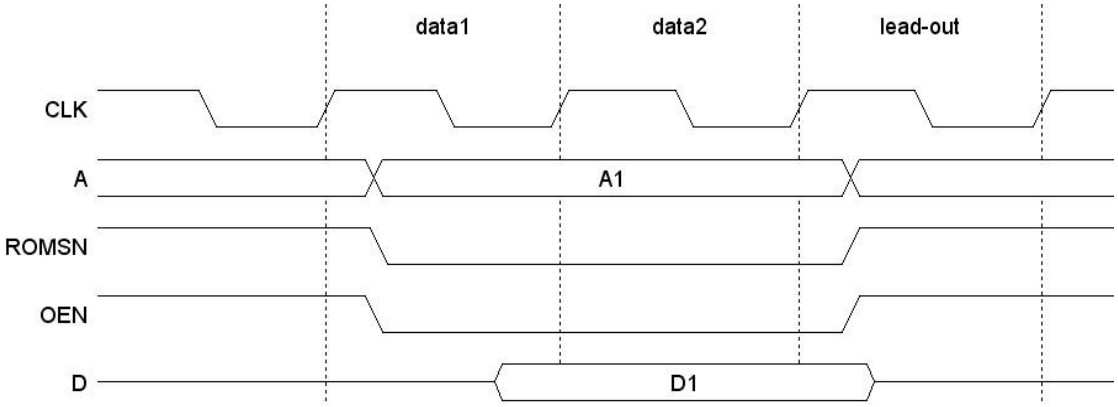
当陷阱发生时，与之唯一对应的识别码写入 TBR 的 8 位 *TT* 字段，程序以 TBR 的值为目的地址进行跳转。因为 TBR 的低四位为 0，因此陷阱向量表的每一入口包含中断处理程序的前 4 条指令（16 个字节）。

附录 7 PROM、SRAM、IO、SDRAM 访问时序

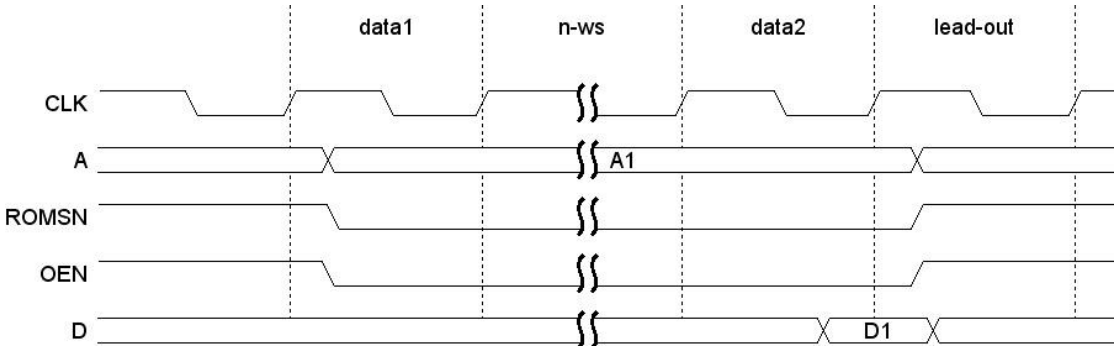
1. PROM 读时序（EDAC 不使能，0 等待周期）



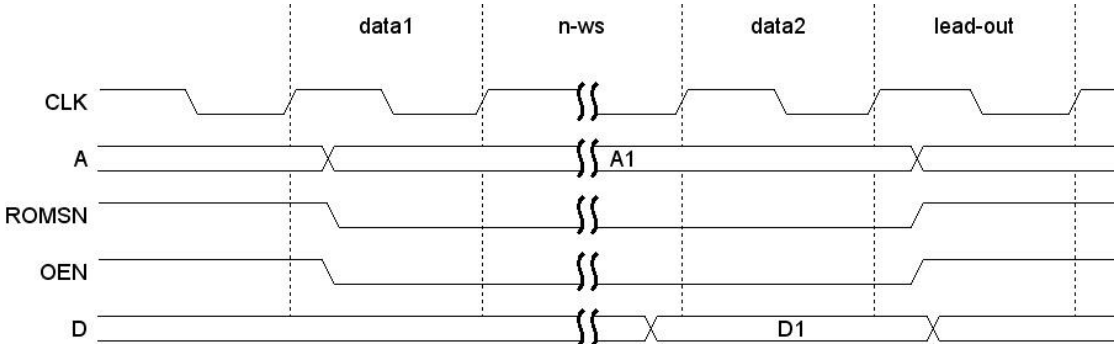
2. PROM 读时序（EDAC 使能，0 等待周期）



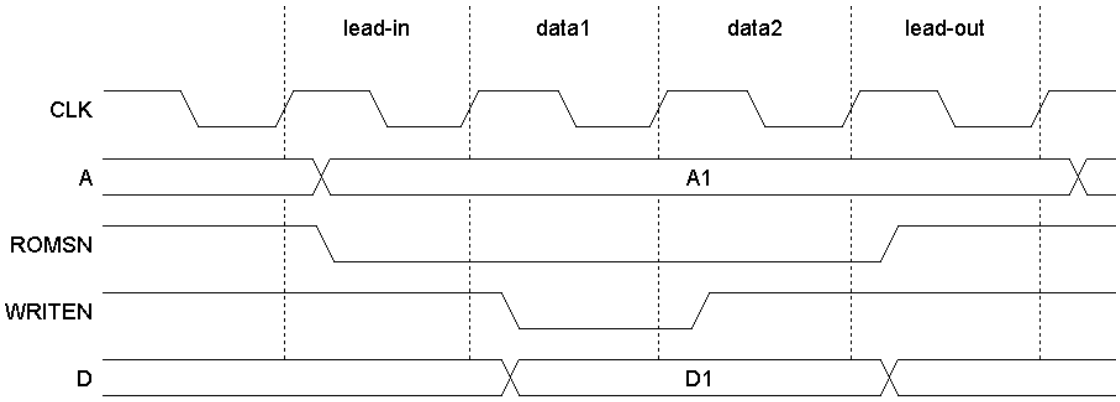
3. PROM 读时序（EDAC 不使能，n 个等待周期）



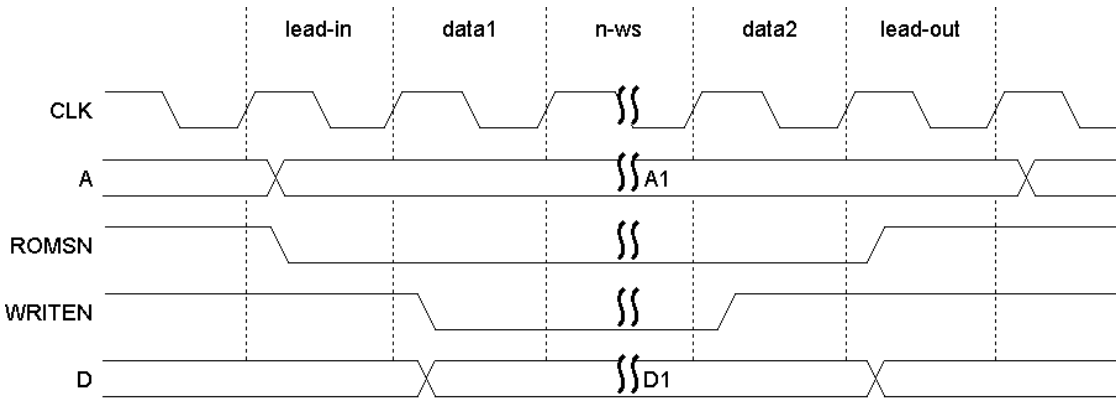
4. PROM 读时序（EDAC 使能，n 个等待周期）



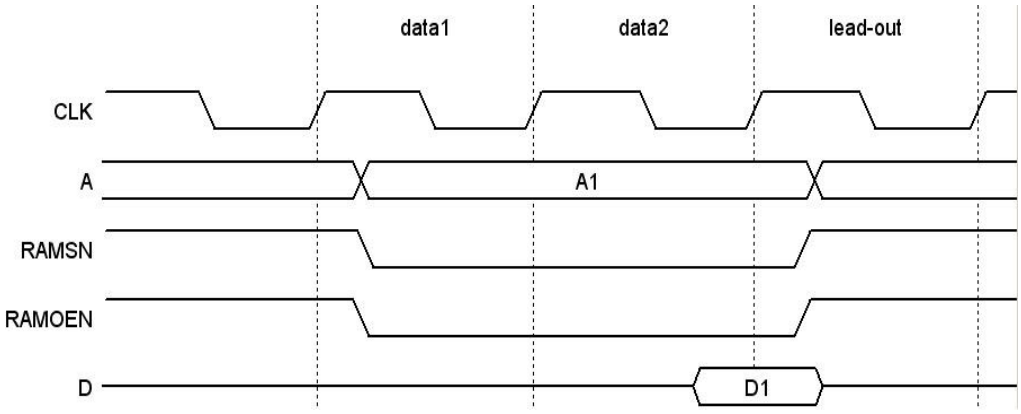
5. PROM 写时序（0 个等待周期）



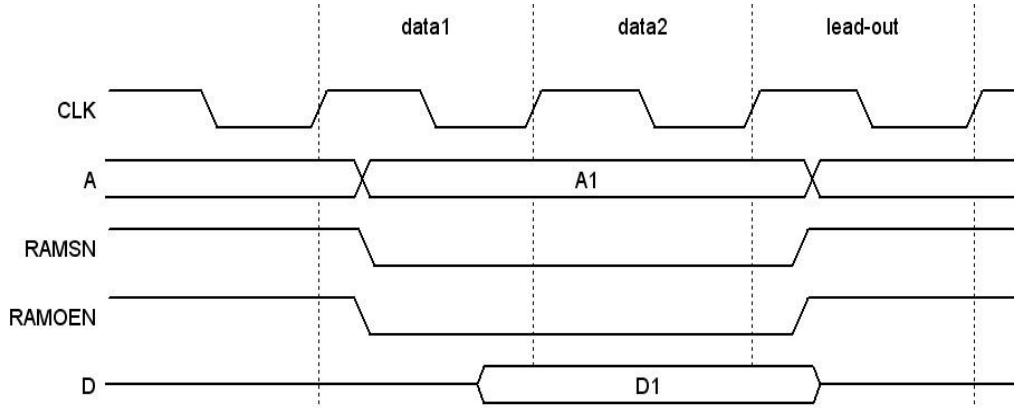
6. PROM 写时序（n 个等待周期）



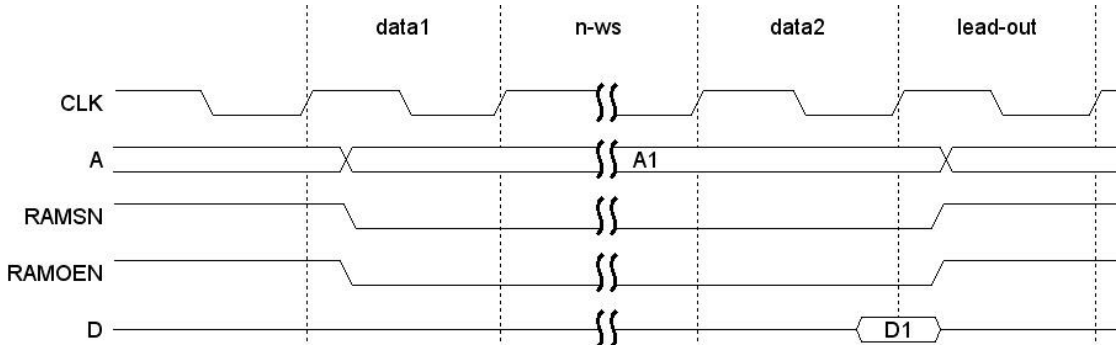
7. SRAM 读访问时序（EDAC 不使能，0 等待周期）



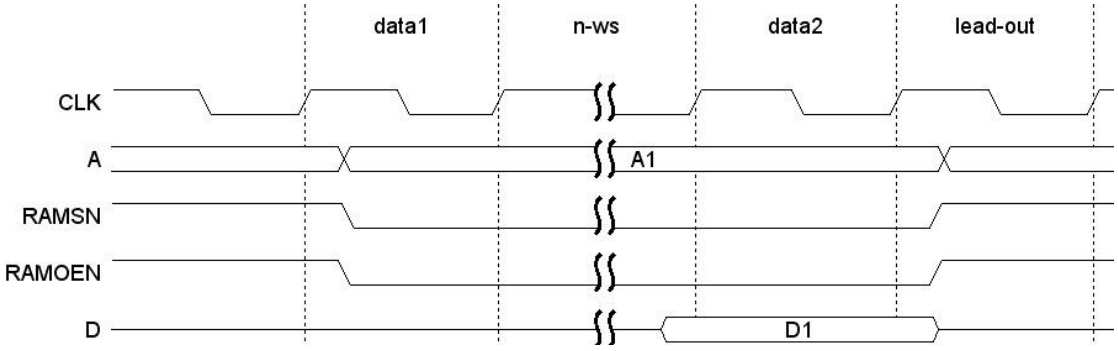
8. SRAM 读访问时序（EDAC 使能，0 等待周期）



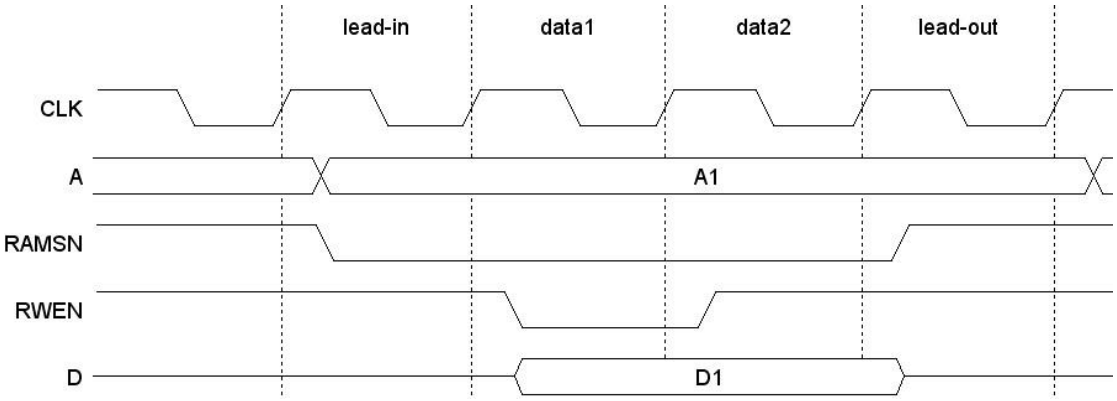
9. SRAM 读访问时序（EDAC 不使能，n 等待周期）



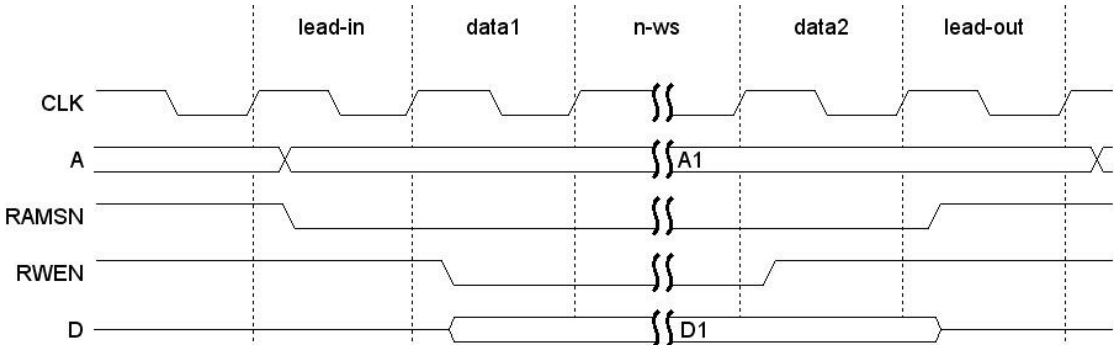
10. SRAM 读访问时序（EDAC 使能，n 等待周期）



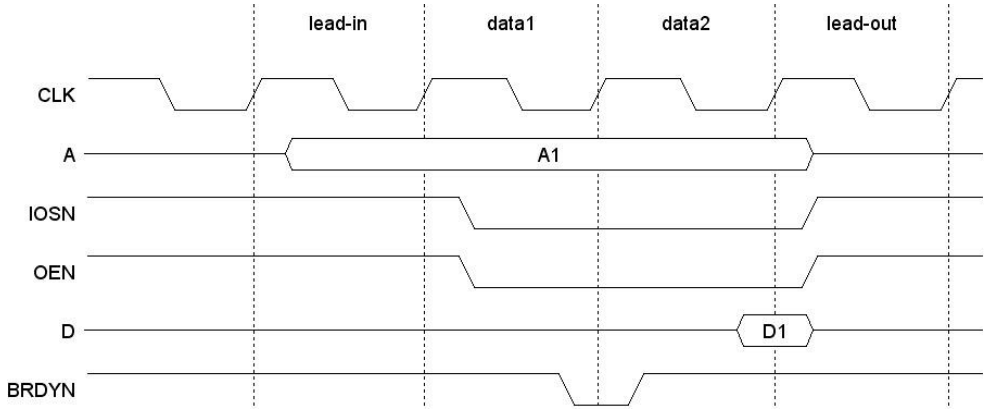
11. SRAM 写访问时序（0 等待周期）



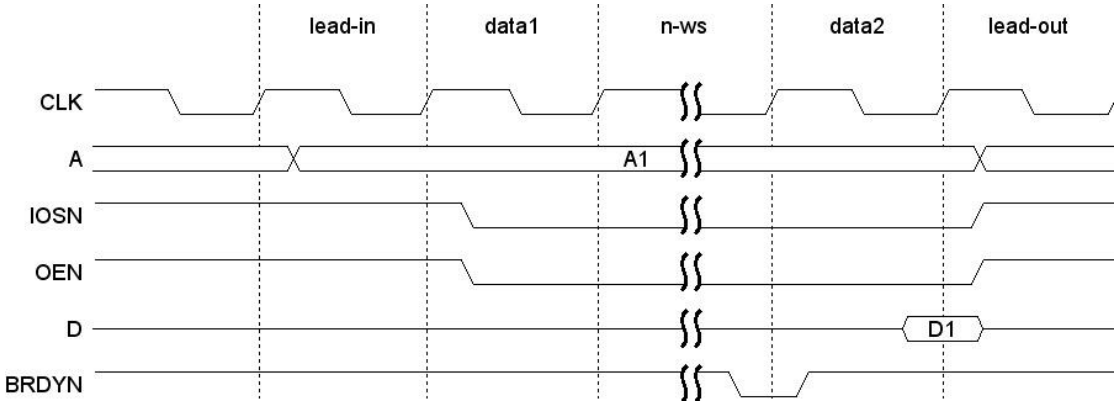
12. SRAM 写访问时序（n 等待周期）



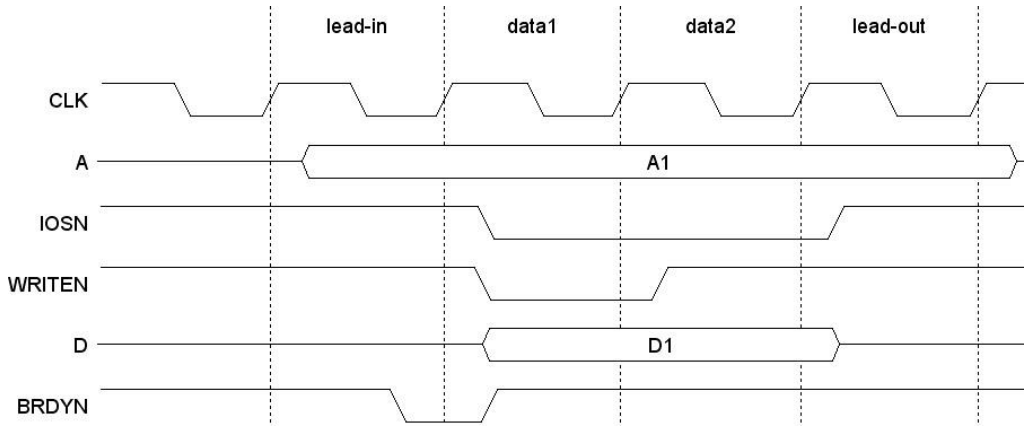
13. I/O 读访问时序（0 等待周期）



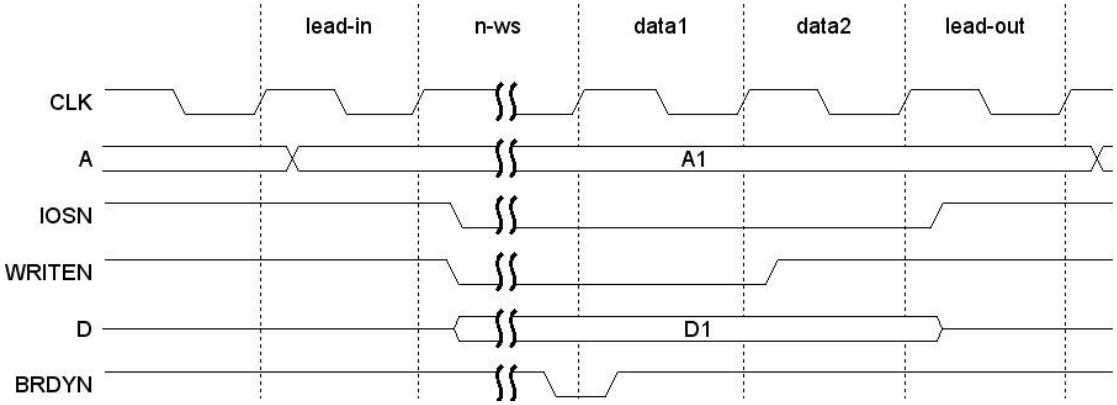
14. I/O 读访问时序（n 等待周期）



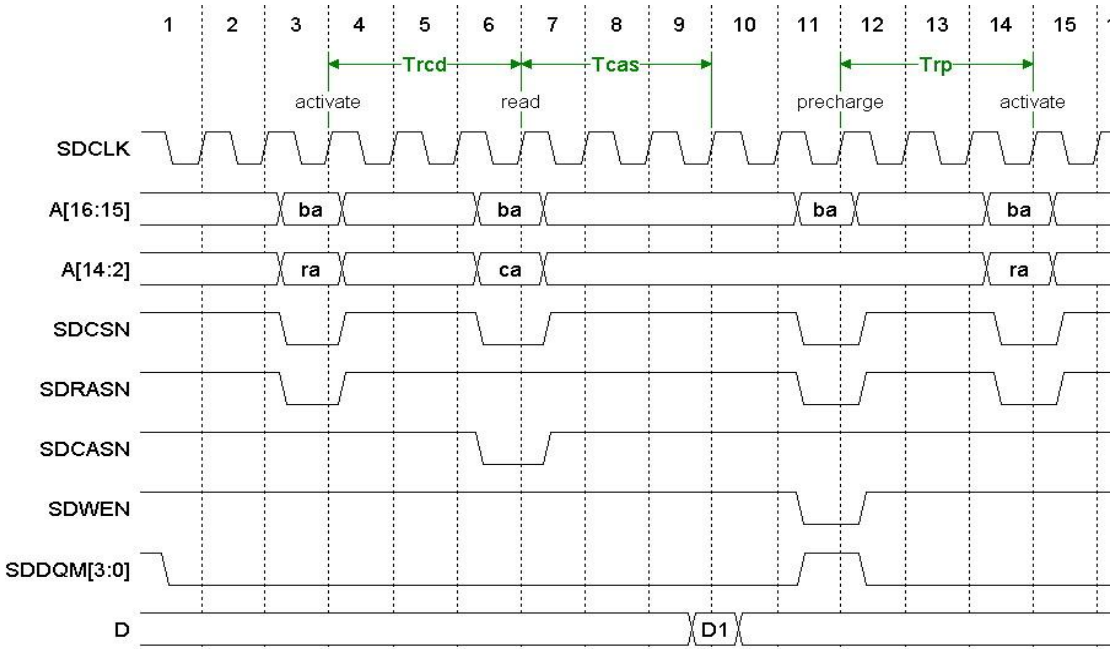
15. I/O 写访问时序（0 等待周期）



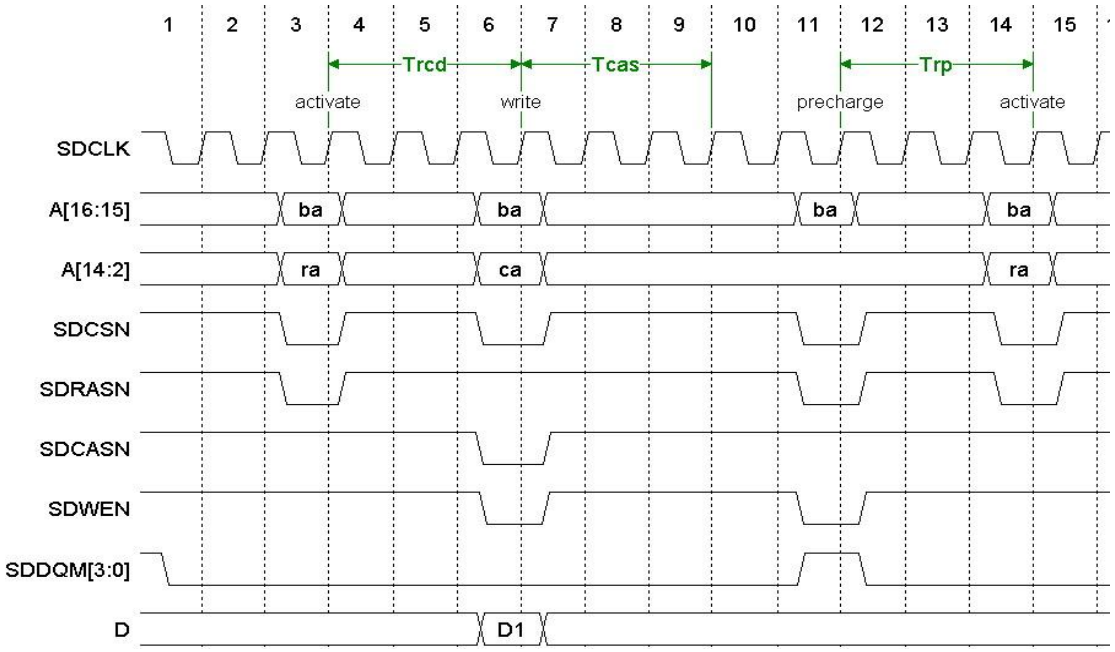
16. I/O 写访问时序（n 等待周期）



17. SDRAM 读时序（Burst length = 1; CL = 3 ）



18. SDRAM 写时序（Burst length = 1; CL = 3 ）


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Trcd
Tcas
Trp
activate
write
precharge
activate
SDCLK
A[16:15]
ba
ba
ba
ba
A[14:2]
ra
ca
ra
SDCSN
SDRASN
SDCASN
SDWEN
SDDQM[3:0]
D
D1

附录 8 与 AT697E 差异性

序号	描述	AT697E	BM3803FMGRH	备注
1	Timer0 和 Timer1 输出管脚	无	有	BM3803FMGRH 的 Timer0 和 Timer1 管脚可以输出一个时钟周期的高电平脉冲。
2	校验位管脚	[7:0]	[6:0]	AT697E 的[7]位用于使能 flash 在线编程。
3	UART	2 个	3 个	
4	8 位 PROM 在启动时是否具有 EDAC 功能	否	是	BM3803FMGRH 的 8 位 PROM 的 EDAC 功能更完备。
5	PLL	固定 4 倍频	1~15 倍频可选	BM3803FMGRH 的 PLL 的倍频更灵活。
		Clk 输入范围 20~25MHz	Clk 输入范围 2~30MHz	BM3803FMGRH 的 PLL 输入时钟的范围更宽。
6	mbist_mode、scan_mode、scan_enable 三个管脚	无	有	BM3803FMGRH 在封装后可进行 mbist 和 scan 测试，保证可靠性和质量。AT697E 在封装后无法进行 mbist 和 scan 测试。
7	符合 PCI 协议	PCI_2.2	PCI_2.3	
8	看门狗使能位	无	有	AT697E 固定使能，BM3803FMGRH 软件控制是否使能。
9	对 regfile 的 EDAC 功能进行造错和诊断	使用 ASR16 寄存器	使用 ASR16 和 ASR17 寄存器	BM3803FMGRH 的 regfile 的造错使能位为 3 位，AT697E 的 regfile 的造错使能位为 1 位，BM3803FMGRH 的 regfile 的造错机制更可靠。
10	Cache 容错配置	容错配置包含在 Cache 的基本配置寄存器中	有 3 个专门的 Cache 容错寄存器	38BM3803FMGRH03 能够对 Cache 的 data 区、tag 区、奇偶校验位都可以造错；AT697E 只对奇偶校验位造错。
	Cache 校验位	校验位有 2 位	校验位有 4 位	BM3803FMGRH 的 Cache

				容错能力更强。
11	外部存储器配置寄存器	PROM 的读写等待周期数为 0，2，4，……，30	PROM 的读写等待周期数为 0，1，2，……，31	BM3803FMGRH 的 PROM 的读写等待周期的设置更精细。
		sram 的读写等待周期数为 0，1，2，3	sram 的读写等待周期数为 0，1，2，……，15	BM3803FMGRH 的 sram 的读写等待周期的设置更宽，最大为 15，而 697 最大为 3。
12	外部存储器容错配置寄存器	容错配置包含在外部存储器配置寄存器 3 中	有 3 个专门的外部存储器容错配置寄存器	BM3803FMGRH 可更灵活的对外部存储器控制器的 EDAC 进行配置，更便于应用。
13	封装	MCGA349	CPGA391	
14	抗辐射能力	抗总剂量≥60kRads (Si)；单粒子锁定（SEL）阈值≥70Mev cm^2/mg；单粒子错误率 GEO 轨道优于 1×10^{-5} 次/天•器件	抗总剂量≥100kRads (Si)；单粒子锁定（SEL）阈值≥75Mev cm^2/mg；单粒子错误率 GEO 轨道优于 8×10^{-5} 次/天•器件	

附录 9 PCI 不使用时 PCI 管脚接法说明

BM3803FMGRH 的 PCI 支持主从两种模式，为确保 PCI 功能模块不对 BM3803FMGRH 其余功能正常使用及长期可靠性造成影响，PCI 模块在不使用时，必须保证 PCI 所有输入都上拉或者下拉，所有双向 IO 都固定为输入状态，避免管脚悬空产生多余功耗和干扰，避免对长期使用的可靠性产生影响。不使用 PCI 功能时相关管脚连接要求如表 A9-1 和表 A9-2 所示。

表 A9-1 不使用 PCI 功能（从桥模式）相关管脚连接要求

管脚名	方向	接法
pci_arb_req[6:0]	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源），PCI 功能使用时，低有效。
pci_idsel	I	下拉（通过 4.7k 欧电阻接 io 地），PCI 功能使用时，高有效。
pci_host	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源），把 PCI 接口置于从模式。
pci_66	I	下拉（通过 4.7k 欧电阻接 io 地）(测试管脚)
pci_inta_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源），PCI 功能使用时，低有效。
pci_intb_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源），PCI 功能使用时，低有效。
pci_intc_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源），PCI 功能使用时，低有效。
pci_intd_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源），PCI 功能使用时，低有效。
pci_gnt	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源），PCI 功能使用时，低有效。
sim_fast_reset	I	下拉（通过 4.7k 欧电阻接 io 地）(测试管脚)

pci_clk	I	下拉（通过 4.7k 欧电阻接 io 地）
pci_frame	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_irdy	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_devsel	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_stop	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_trdy	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_perr	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_serr	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_ad(31:0)	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)
pci_cbe(3:0)	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)
pci_par	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)
pci_req	0	浮空
pci_arb_gnt_n(6:0)	0	浮空
pci_rst	I	下拉（通过 4.7k 欧电阻接 io 地），PCI 功能使用时，低有效。
testin0	I	下拉（通过 4.7k 欧电阻接 io 地）
testin1	I	下拉（通过 4.7k 欧电阻接 io 地）
testout0	0	浮空
testout1	I/O	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)

注：表 A9-1 的管脚方向只适用于在 PCI 功能不使用，且配置为从模式、pci_rst 置为有效（下拉）时。

表 A9-2 不使用 PCI 功能（主桥模式）相关管脚连接要求

管脚名	方向	接法
pci_arb_req[6:0]	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_idsel	I	下拉（通过 4.7k 欧电阻接 io 地)，PCI 功能使用时，高有效。
pci_host	I	下拉（通过 4.7k 欧电阻接 io 地)，把 PCI 接口置于主模式。
pci_66	I	下拉（通过 4.7k 欧电阻接 io 地)，测试管脚。
pci_inta_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_intb_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_intc_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_intd_n	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_gnt	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
sim_fast_reset	I	下拉（通过 4.7k 欧电阻接 io 地)，测试管脚。
pci_clk	I	接高或低
pci_frame	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_irdy	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。

pci_devsel	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_stop	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_trdy	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_perr	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_serr	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)，PCI 功能使用时，低有效。
pci_ad(31:0)	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)
pci_cbe(3:0)	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)
pci_par	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)
pci_req	0	浮空
pci_arb_gnt_n(6:0)	0	浮空
pci_rst	0	浮空
testin0	I	下拉（通过 4.7k 欧电阻接 io 地）
testin1	I	下拉（通过 4.7k 欧电阻接 io 地）
testout0	0	浮空
testout1	I	上拉(通过 4.7k 欧电阻接 3.3V I/O 电源)

注：表 A9-2 的管脚方向只适用于在 PCI 功能不使用，且配置为主模式时。