

S3C2440A

32 位 CMOS RISC

微控制器

用户手册

1

产品概述

引言

此用户手册描述的是三星公司的 16/32 位精简指令集 (**RISC**) 微处理器 S3C2440A。三星公司的 S3C2440A 为手持设备和普通应用提供了低功耗和高性能的小型芯片微控制器的解决方案。为了降低整体系统成本 ,S3C2440A 还提供了以下丰富的内部设备。

S3C2440A 基于 ARM920T 核心 , 0.13 μ m 的 CMOS 标准宏单元和存储器单元。低功耗 , 简单 , 精致 , 且全静态设计特别适合于对成本和功率敏感型的应用。它采用了新的总线架构如先进微控制总线构架 (**AMBA**)。

S3C2440A 的突出特点是其处理器核心 , 是一个由 Advanced RISC Machines (**ARM**) 公司设计的 16/32 位 ARM920T 的 RISC 处理器。ARM920T 实现了 MMU , AMBA 总线和哈佛结构高速缓冲体系结构。这一结构具有独立的 16KB 指令高速缓存和 16KB 数据高速缓存。每个都是由具有 8 字长的行(*line*)组成。

通过提供一套完整的通用系统外设 , S3C2440A 减少整体系统成本和无需配置额外的组件。综合对芯片的功能描述,本手册将介绍 S3C2440A 集成的以下片上功能 :

- 1.2V 内核供电, 1.8V/2.5V/3.3V 储存器供电, 3.3V 外部 I/O 供电 具备 16KB 的指令缓存和 16KB 的数据缓存和 MMU 的微处理器
- 外部存储控制器 (SDRAM 控制和片选逻辑)
- LCD 控制器 (最大支持 4K 色 STN 和 256K 色 TFT) 提供 1 通道 LCD 专用 DMA
- 4 通道 DMA 并有外部请求引脚
- 3 通道 UART (IrDA1.0, 64 字节发送 FIFO 和 64 字节接收 FIFO)
- 2 通道 SPI
- 1 通道 IIC 总线接口 (支持多主机)
- 1 通道 IIS 总线音频编码器接口
- AC'97 编解码器接口
- 兼容 SD 主接口协议 1.0 版和 MMC 卡协议 2.11 兼容版
- 2 通道 USB 主机/1 通道 USB 设备 (1.1 版)
- 4 通道 PWM 定时器和 1 通道内部定时器/看门狗定时器
- 8 通道 10 位 ADC 和触摸屏接口
- 具有日历功能的 RTC
- 摄像头接口 (最大支持 4096×4096 像素输入 ; 2048×2048 像素输入支持缩放)
- 130 个通用 I/O 口和 24 通道外部中断源
- 具有普通 , 慢速 , 空闲和掉电模式
- 具有 PLL 片上时钟发生器

特性

体系结构

- 手持设备的完整系统和普通嵌入式应用
- 16/32 位 RISC 体系架构和 ARM920T CPU 核心的强大的指令集
- 增强型 ARM 架构 MMU 以支持 WinCE , EPOC 32 和 Linux
- 指令高速缓存 , 数据高速缓存 , 写缓冲和物理地址 TAG RAM 以减少执行主存储器带宽和延迟性能的影响
- ARM920T CPU 核支持 ARM 调试架构
- 内部先进微控制器总线架构 (AMBA)(AMBA2.0 , AHB/APB)

系统管理

- 支持大/小端
- 地址空间 : 每 Bank 128M 字节 (总共 1G 字节)
- 支持可编程的每 Bank 8/16/32 位数据总线宽度
- BANK0 到 BANK6 固定 Bank 的起始地址
- BANK7 具有可编程 Bank 起始地址和大小
- 8 个存储器 Bank :
- 六个存储器 Bank 为 ROM , SRAM 和其它
- 两个存储器 Bank 为 ROM/SRAM/ SDRAM
- 所有存储器具备完整可编程访问周期
- 支持外部等待信号来扩展总线周期
- 支持 SDRAM 掉电时自刷新模式
- 支持从各种类型 ROM 启动 (NOR/NAND Flash , EEPROM 或其它)

NAND Flash 启动引导 (*BootLoader*)

- 支持从 NAND Flash 启动
- 4KB 的启动内部缓冲区
- 支持启动后 NAND flash 作为存储器
- 支持先进 NAND Flash

高速缓存存储器

- 64 路指令缓存 (16KB) 和数据缓存 (16KB) 的组相联高速缓存
- 每行 8 字长度 , 其中含一个有效位和两个 dirty 位
- 伪随机或循环 robin 置换算法
- 执行直写或回写高速缓存刷新主存储器
- 写缓冲区可以保存 16 字的数据和 4 个地址

时钟和电源管理

- 片上 MPLL 和 UPLL :
 - UPLL 产生时钟运作 USB 主机/设备
 - MPLL 产生时钟运作 1.3V 下最高 400MHz 的 MCU
- 用软件可以有选择的提供时钟给各功能模块
- 电源模式 : 普通、慢速、空闲和睡眠模式
 - 普通模式 : 正常运行模式
 - 慢速模式 : 无 PLL 的低频率时钟
 - 空闲模式 : 只停止 CPU 的时钟
 - 睡眠模式 : 关闭包括所有外设的核心电源
- EINT[15:0] 或 RTC 闹钟中断触发从睡眠模式中唤醒

中断控制器

- 60 个中断源(1 个看门狗 ,5 个定时器 ,9 个 UART , 24 个外部中断 ,4 个 DMA ,2 个 RTC ,2 个 ADC , 1 个 IIC ,2 个 SPI ,1 个 SDI ,2 个 USB ,1 个 LCD , 1 个电池故障 ,1 个 NAND ,2 个摄像头 ,1 个 AC'97)
- 外部中断源中电平/边沿模式
- 可编程边沿和电平的极性
- 支持快速中断请求 (FIQ) 给非常紧急的中断请求

脉宽调制 (PWM) 定时器

- 4 通道 16 位具有 PWM 功能的定时器 ,1 通道 16 位基于 DMA 或基于中断运行的内部定时器
- 可编程的占空比 , 频率和极性
- 能产生死区
- 支持外部时钟源。

特性 (续)

RCT (实时时钟)

- 完整时钟特性：毫秒、秒、分、时、星期、日、月和年
- 工作在 32.768KHz 时钟频率
- 闹钟中断
- 时钟节拍中断

通用输入/输出端口

- 24 个外部中断端口
- 130 个复用输入/输出端口

DMA 控制器

- 4 通道 DMA 控制器
- 支持存储器到存储器，IO 口到存储器，存储器到 IO 口和 IO 口到 IO 口的传输
- 采用触发传输模式来提高传输速率

UART

- 3 通道基于 DMA 或基于中断运行的 UART
- 支持 5 位、6 位、7 位、或 8 位串行数据发送/接收
- 支持 UART 运行在外部时钟 (UEXTCLK)
- 可编程波特率
- 支持 IrDA 1.0
- 测试用回环模式
- 每个通道都包含内部 64 位发送 FIFO 和 64 位接收 FIFO

A/D 转换器和触屏接口

- 8 通道多路复用 ADC
- 最高 500KSPS 和 10 位分辨率
- 内置 FET 给线性触屏接口

IIC 总线接口

- 1 通道多主机 IIC 总线
- 串行，8 位，可在标准模式 100Kbit/s 下或快速模式 400Kbit/s 下进行双向数据传输

LCD 控制器 STN LCD 显示特性

- 支持 3 种类型 STN LCD 面板：4 位双扫描，4 位单扫描和 8 位单扫描显示类型
- 支持单色模式，4 阶灰度，16 阶灰度，256 色和 4096 色的 STN LCD
- 支持多种屏幕尺寸
 - 实际屏幕尺寸典型值：640x480 , 320x240 , 160x160 和其它
 - 最大帧缓冲区大小为 4M 字节
 - 256 色模式下最大实际屏幕尺寸：4096x1024 , 2048x2048 , 1024x4096 和其它

TFT (薄膜晶体管) 彩色显示特性

- 支持彩色 TFT 的 1、2、4 或 8 bpp(位/像素) 调色显示
- 支持彩色 TFT 的 16 , 24 bpp 非调色真彩显示
- 支持在 24 bpp 模式下最大 16M 色的 TFT
- 内嵌 LPC3600 时序控制器，支持 LTS350Q1-PD1/2 (三星 3.5 吋竖屏/256K 色/反光型 a-Si TFT LCD)
- 内嵌 LCC3600 时序控制器，支持 LTS350Q1-PE1/2 (三星 3.5 吋竖屏/256K 色/半透型 a-Si TFT LCD)
- 支持多种屏幕尺寸
 - 实际屏幕尺寸典型值：640x480 , 320x240 , 160x160 和其它
 - 最大帧缓冲区大小为 4M 字节
 - 64K 色模式下最大实际屏幕尺寸：2048x1024 和其它

看门狗定时器

- 16 位看门狗定时器
- 中断请求或系统复位超时

特性 (续)

IIS 总线接口

- 1 通道 IIS 总线 , 运行在基于 DMA 音频接口
- 串行 , 8/16 位每通道数据传输
- 发送/接收具备 128 字节(64 字节+64 字节)FIFO
- 支持 IIS 格式和 MSB-justified 数据格式

AC'97 音频编解码器接口

- 支持 16 位采样
- 1 通道立体声 PCM 输入 , 1 通道立体声 PCM 输出和 1 通道 MIC 输入

USB 主机 (Host)

- 2 个 USB 主机端口
- 遵从 OHCI Rev. 1.0
- 兼容 USB 规格 1.1 版本

USB 设备 (Device)

- 1 个 USB 设备端口
- 5 个 USB 设备端点
- 兼容 USB 规格 1.1 版本

SD 主机接口

- 正常 , 中断和 DMA 数据传输模式 (可按字节 , 半字 , 字传输)
- 支持 DMA burst4 访问 (只支持字传输)
- 兼容 SD 记忆卡协议 1.0 版本
- 兼容 SDIO 卡协议 1.0 版本
- 发送/接收具备 64 字节 FIFO
- 兼容 MMC 卡协议 2.11 版本

SPI 接口

- 兼容 2 通道 SPI 接口协议 2.11 版本
- 发送/接收具备 2 个 8 位移位寄存器
- 基于 DMA 或基于中断运行

摄像头接口

- 支持 ITU-R BT 601/656 8 位模式
- 发送/接收具备 2 个 8 位移位寄存器
- 基于 DMA 或基于中断运行
- DZI (数字放大) 能力
- 可编程视频同步信号极性
- 最大支持 4096 × 4096 像素输入 (2048 × 2048 像素输入时支持缩放)
- 图像镜像和旋转 (X 轴镜像 , Y 轴镜像和 180 ° 旋转)
- 格式化摄像头输出 (RGB 16/24 位和 YCbCr 4:2:0/4:2:2 格式)

工作电压范围

- 核心电压 : 300MHz 下 1.20V
400MHz 下 1.30V
- 存储器电压 : 1.8V/2.5V/3.0V/3.3V
- I/O 口电压 : 3.3V

工作频率

- Fclk 最高 400MHz
- Hclk 最高 136MHz
- Pclk 最高 68MHz

封装

- 289-FBGA

方框图

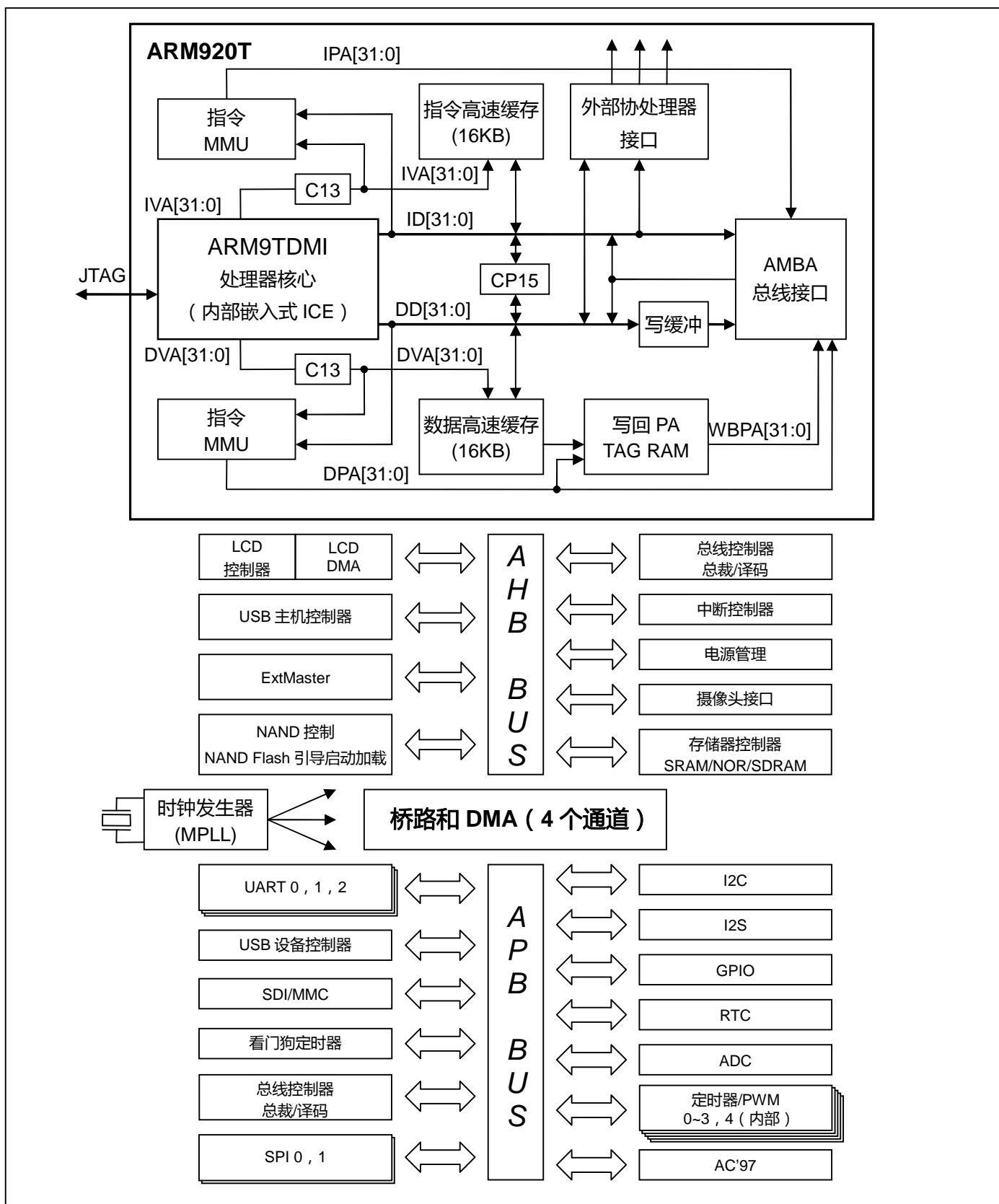


图 1-1. S3C2440A 方框图

引脚分配

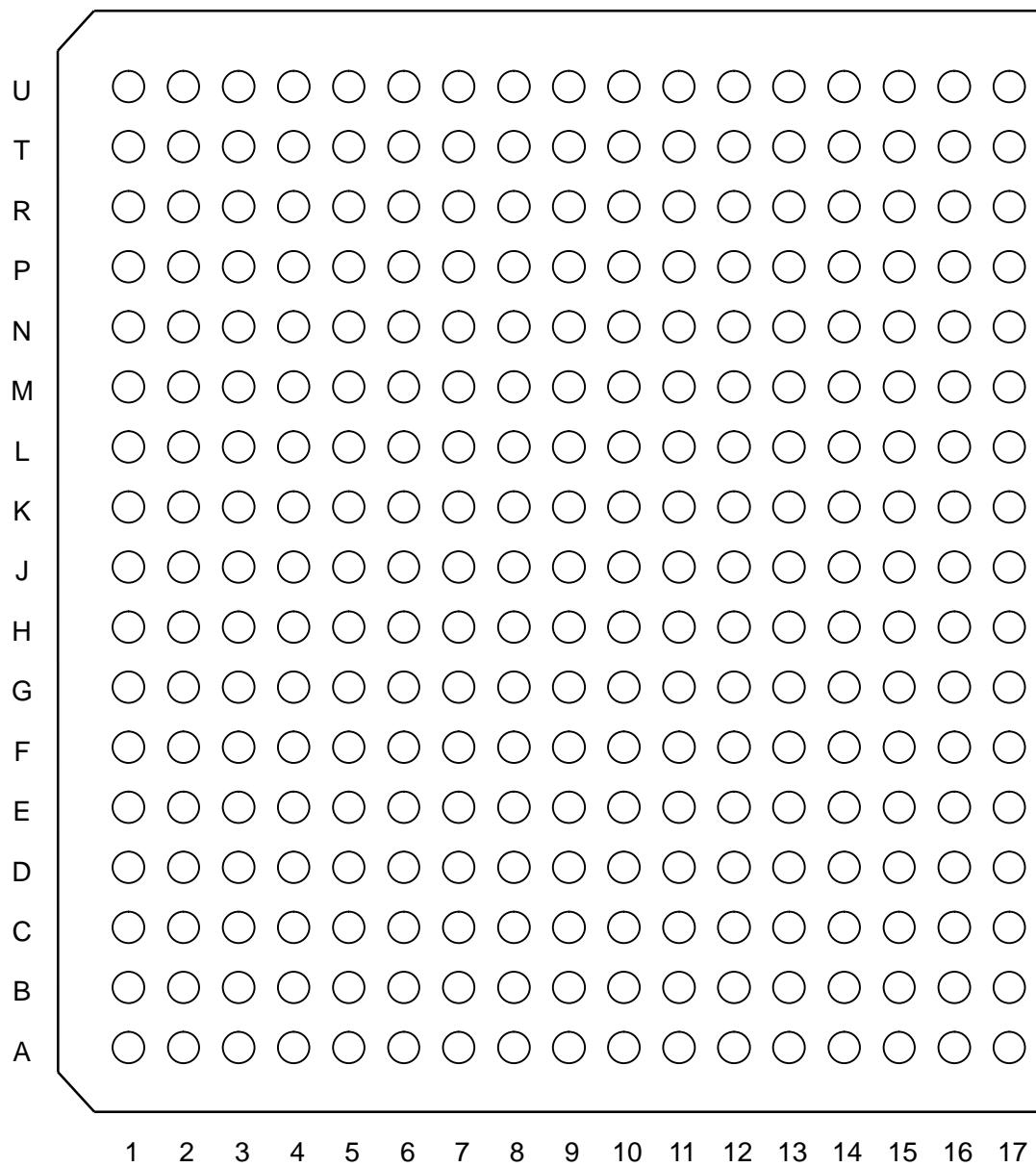


图 1-2. S3C2440A 引脚分配 (289-FBGA)

表 1-1. 289 管脚 FBGA 引脚分配-管脚号顺序(1/2)

管脚号	管脚名称	管脚号	管脚名称	管脚号	管脚名称
A1	VDDi	D1	ALE/GPA18	G1	VSSOP
A2	SCKE	D2	nGCS6	G2	CAMHREF/GPJ10
A3	VSSI	D3	nGCS4/GPA15	G3	CAMDATA1/GPJ1
A4	VSSI	D4	nBE0	G4	VDDalive
A5	VSSMOP	D5	nBE2	G5	CAMPCLK/GPJ8
A6	VDDi	D6	nSCAS	G6	FRnB
A7	VSSMOP	D7	ADDR7	G7	CAMVSYNC/GPJ9
A8	ADDR10	D8	ADDR5	G8	ADDR8
A9	VDDMOP	D9	ADDR16/GPA1	G9	ADDR17/GPA2
A10	VDDi	D10	ADDR20/GPA5	G10	ADDR25/GPA10
A11	VSSMOP	D11	ADDR26/GPA11	G11	DATA28
A12	VSSI	D12	DATA0	G12	DATA25
A13	DATA3	D13	DATA8	G13	DATA23
A14	DATA7	D14	DATA14	G14	XTIpll
A15	VSSMOP	D15	DATA12	G15	XTOpll
A16	VDDi	D16	VSSMOP	G16	DATA29
A17	DATA10	D17	VSSMOP	G17	VSSI
B1	VSSMOP	E1	nFRE/GPA20	H1	VSSiarm
B2	nGCS1/GPA12	E2	VSSMOP	H2	CAMDATA7/GPJ7
B3	SCLK1	E3	nGCS7	H3	CAMDATA4/GPJ4
B4	SCLK0	E4	nWAIT	H4	CAMDATA3/GPJ3
B5	nBE1	E5	nBE3	H5	CAMDATA2/GPJ2
B6	VDDMOP	E6	nWE	H6	CAMDATA0/GPJ0
B7	ADDR2	E7	ADDR1	H7	CAMDATA5/GPJ5
B8	ADDR9	E8	ADDR6	H8	ADDR13
B9	ADDR12	E9	ADDR14	H9	ADDR19/GPA4
B10	VSSI	E10	ADDR23/GPA8	H10	ADDR22/GPA7
B11	VDDi	E11	DATA2	H11	VSSOP
B12	VDDMOP	E12	DATA20	H12	EXTCLK
B13	VSSMOP	E13	DATA19	H13	DATA30
B14	VDDMOP	E14	DATA18	H14	N_BATT_FLT
B15	DATA9	E15	DATA17	H15	nTRST
B16	VDDMOP	E16	DATA21	H16	nRESET
B17	DATA15	E17	DATA24	H17	TDI
C1	VDDMOP	F1	VDDi	J1	VDDOP
C2	nGCS5/GPA16	F2	VSSI	J2	VDDiarm
C3	nGCS2/GPA13	F3	nFWE/GPA19	J3	CAMCLKOUT/GPJ11
C4	nGCS3/GPA14	F4	nFCE/GPA22	J4	CAMRESET/GPJ12
C5	nOE	F5	CLE/GPA17	J5	TOUT1/GPB1
C6	nSRAS	F6	nGCS0	J6	TOUT0/GPB0
C7	ADDR4	F7	ADDR0/GPA0	J7	TOUT2/GPB2
C8	ADDR11	F8	ADDR3	J8	CAMDATA6/GPJ6
C9	ADDR15	F9	ADDR18/GPA3	J9	SDDAT3/GPE10
C10	ADDR21/GPA6	F10	DATA4	J10	EINT10/nSS0/GPG2
C11	ADDR24/GPA9	F11	DATA5	J11	TXD2/nRTS1/GPH6
C12	DATA1	F12	DATA27	J12	PWREN
C13	DATA6	F13	DATA31	J13	TCK
C14	DATA11	F14	DATA26	J14	TMS
C15	DATA13	F15	DATA22	J15	RXD2/nCTS1/GPH7
C16	DATA16	F16	VDDi	J16	TDO
C17	VSSI	F17	VDDMOP	J17	VDDalive

表 1-1. 289 管脚 FBGA 引脚分配-管脚号顺序(2/2) (续)

管脚号	管脚名称	管脚号	管脚名称	管脚号	管脚名称
K1	VSSiarm	N1	VSSOP	T1	VSSiarm
K2	nXBACK/GPB5	N2	VD0/GPC8	T2	VSSiarm
K3	TOUT3/GPB3	N3	VD4/GPC12	T3	VDDOP
K4	TCLK0/GPB4	N4	VD2/GPC10	T4	VD17/SPI MOSI1/GPD9
K5	nXDREQ1/GPB8	N5	VD10/GPD2	T5	VD19/GPD11
K6	nXDREQ0/GPB10	N6	VD15/GPD7	T6	VDDiarm
K7	nXDACK1/GPB7	N7	VD22/nSS1/GPD14	T7	CDCLK/AC_nRESET
K8	SDCMD/GPE6	N8	SDCLK/GPE5	T8	VDDiarm
K9	SPI MOSI0/GPE11	N9	EINT8/GPG0	T9	EINT9/GPG1
K10	EINT13/SPI MOSI1/GPG5	N10	EINT18/nCTS1/GPG10	T10	EINT16/GPG8
K11	nCTS0/GPH0	N11	DP0	T11	EINT21/GPG13
K12	VDDOP	N12	DN1/PDN0	T12	VDDOP
K13	TXD0/GPH2	N13	nRSTOUT/GPA21	T13	OM3
K14	RXD0/GPH3	N14	MPLL CAP	T14	VSSA_ADC
K15	UEXTCLK/GPH8	N15	VDD_RTC	T15	OM0
K16	TXD1/GPH4	N16	VDDA_MPLL	T16	YM/AIN4
K17	RXD1/GPH5	N17	EINT0/GPF0	T17	YP/AIN5
L1	LEND/GPC0	P1	LCD_LPCREVB/GPC7	U1	VDDiarm
L2	VDDiarm	P2	VD5/GPC13	U2	VDDiarm
L3	nXDACK0/GPB9	P3	VD7/GPC15	U3	VSSOP
L4	VCLK/GPC1	P4	VD12/GPD4	U4	VSSiarm
L5	nXBREQ/GPB6	P5	VD14/GPD6	U5	VD23/nSS0/GPD15
L6	VD1/GPC9	P6	VD20/GPD12	U6	I2SSDO/AC_SDATA_OUT
L7	VFRAME/GPC3	P7	I2SLRCK/AC_SYNC	U7	VSSiarm
L8	I2SSDI/AC_SDATA_IN	P8	SDDAT2/GPE9	U8	IIC SCL/GPE14
L9	SPI CLK0/GPE13	P9	SPI MOSI0/GPE12	U9	VSSOP
L10	EINT15/SPI CLK1/GPG7	P10	CLKOUT1/GPH10	U10	VSSiarm
L11	EINT22/GPG14	P11	EINT12/LCD_PWREN/GPG4	U11	VDDi
L12	Xtortc	P12	DN0	U12	EINT19/TCLK1/GPG11
L13	EINT2/GPF2	P13	OM2	U13	EINT23/GPG15
L14	EINT5/GPF5	P14	VDDA_ADC	U14	DP1/PDP0
L15	EINT6/GPF6	P15	AIN3	U15	VSSOP
L16	EINT7/GPF7	P16	XP/AIN7	U16	Vref
L17	nRTS0/GPH1	P17	UPLL CAP	U17	AIN1
M1	VLIN/E/GPC2	R1	VD3/GPC11		
M2	LCD_LPCREV/GPC6	R2	VD8/GPD0		
M3	LCD_LPCOE/GPC5	R3	VD11/GPD3		
M4	VM/GPC4	R4	VD13/GPD5		
M5	VD9/GPD1	R5	VD18/SPI CLK1/GPD10		
M6	VD6/GPC14	R6	VD21 /GPD13		
M7	VD16/SPI MOSI1/GPD8	R7	I2SSCLK/AC_BIT_CLK		
M8	SDDAT1/GPE8	R8	SDDAT0/GPE7		
M9	IIC SDA/GPE15	R9	CLKOUT0/GPH9		
M10	EINT20/GPG12	R10	EINT11/nSS1/GPG3		
M11	EINT17/nRTS1/GPG9	R11	EINT14/SPI MOSI1/GPG6		
M12	VSSA_UPLL	R12	NCON		
M13	VDDA_UPLL	R13	OM1		
M14	Xtirtc	R14	AIN0		
M15	EINT3/GPF3	R15	AIN2		
M16	EINT1/GPF1	R16	XM/AIN6		
M17	EINT4/GPF4	R17	VSSA_MPLL		

表 1-2. S3C2440A 289 管脚 FBGA 引脚分配(1/6)

管脚号	管脚名称	默认功能	I/O 口状态 @BUS REQ	I/O 口状态 @睡眠模式	I/O 口状态 @nRESET	I/O 口 类型
F7	ADDR0/GPA0	ADDR0	Hi-z/-	O(L)/-	O(L)	t10s
E7	ADDR1	ADDR1	Hi-z	O(L)	O(L)	t10s
B7	ADDR2	ADDR2	Hi-z	O(L)	O(L)	t10s
F8	ADDR3	ADDR3	Hi-z	O(L)	O(L)	t10s
C7	ADDR4	ADDR4	Hi-z	O(L)	O(L)	t10s
D8	ADDR5	ADDR5	Hi-z	O(L)	O(L)	t10s
E8	ADDR6	ADDR6	Hi-z	O(L)	O(L)	t10s
D7	ADDR7	ADDR7	Hi-z	O(L)	O(L)	t10s
G8	ADDR8	ADDR8	Hi-z	O(L)	O(L)	t10s
B8	ADDR9	ADDR9	Hi-z	O(L)	O(L)	t10s
A8	ADDR10	ADDR10	Hi-z	O(L)	O(L)	t10s
C8	ADDR11	ADDR11	Hi-z	O(L)	O(L)	t10s
B9	ADDR12	ADDR12	Hi-z	O(L)	O(L)	t10s
H8	ADDR13	ADDR13	Hi-z	O(L)	O(L)	t10s
E9	ADDR14	ADDR14	Hi-z	O(L)	O(L)	t10s
C9	ADDR15	ADDR15	Hi-z	O(L)	O(L)	t10s
D9	ADDR16/GPA1	ADDR16	Hi-z/-	O(L)	O(L)	t10s
G9	ADDR17/GPA2	ADDR17	Hi-z/-	O(L)	O(L)	t10s
F9	ADDR18/GPA3	ADDR18	Hi-z/-	O(L)	O(L)	t10s
H9	ADDR19/GPA4	ADDR19	Hi-z/-	O(L)	O(L)	t10s
D10	ADDR20/GPA5	ADDR20	Hi-z/-	O(L)	O(L)	t10s
C10	ADDR21/GPA6	ADDR21	Hi-z/-	O(L)	O(L)	t10s
H10	ADDR22/GPA7	ADDR22	Hi-z/-	O(L)	O(L)	t10s
E10	ADDR23/GPA8	ADDR23	Hi-z/-	O(L)	O(L)	t10s
C11	ADDR24/GPA9	ADDR24	Hi-z/-	O(L)	O(L)	t10s
G10	ADDR25/GPA10	ADDR25	Hi-z/-	O(L)	O(L)	t10s
D11	ADDR26/GPA11	ADDR26	Hi-z/-	O(L)	O(L)	t10s
R14	AIN0	AIN0	-	-	AI	r10
U17	AIN1	AIN1	-	-	AI	r10
R15	AIN2	AIN2	-	-	AI	r10
P15	AIN3	AIN3	-	-	AI	r10
T16	YM/AIN4	AIN4	-/-	-/-	AI	r10
T17	YP/AIN5	YP	-/-	-/-	AI	r10
R16	XM/AIN6	AIN6	-/-	-/-	AI	r10
P16	XP/AIN7	XP	-/-	-/-	AI	r10
H6	CAMDATA0/GPJ0	GPJ0	-/-	Hi-z/-		t8
G3	CAMDATA1/GPJ1	GPJ1	-/-	Hi-z/-		t8
H5	CAMDATA2/GPJ2	GPJ2	-/-	Hi-z/-		t8
H4	CAMDATA3/GPJ3	GPJ3	-/-	Hi-z/-		t8
H3	CAMDATA4/GPJ4	GPJ4	-/-	Hi-z/-		t8
H7	CAMDATA5/GPJ5	GPJ5	-/-	Hi-z/-		t8
J8	CAMDATA6/GPJ6	GPJ6	-/-	Hi-z/-		t8
H2	CAMDATA7/GPJ7	GPJ7	-/-	Hi-z/-		t8
G5	CAMPCLK/GPJ8	GPJ8	-/-	Hi-z/-		t8
G7	CAMVSYNC/GPJ9	GPJ9	-/-	Hi-z/-		t8
G2	CAMHREF/GPJ10	GPJ10	-/-	Hi-z/-		t8
J3	CAMCLKOUT/GPJ11	GPJ11	-/-	O(L)/-		t8
J4	CAMRESET/GPJ12	GPJ12	-/-	O(L)/-		t8
D12	DATA0	DATA0	Hi-z	Hi-z,O(L)		b12s
C12	DATA1	DATA1	Hi-z	Hi-z,O(L)		b12s
E11	DATA2	DATA2	Hi-z	Hi-z,O(L)		b12s
A13	DATA3	DATA3	Hi-z	Hi-z,O(L)		b12s
F10	DATA4	DATA4	Hi-z	Hi-z,O(L)		b12s

表 1-2. S3C2440A 289 管脚 FBGA 引脚分配(2/6) (续)

管脚号	管脚名称	默认功能	I/O 口状态 @BUS REQ	I/O 口状态 @睡眠模式	I/O 口状态 @nRESET	I/O 口 类型
F11	DATA5	DATA5	Hi-z	Hi-z,O(L)	—	b12s
C13	DATA6	DATA6	Hi-z	Hi-z,O(L)	—	b12s
A14	DATA7	DATA7	Hi-z	Hi-z,O(L)	—	b12s
D13	DATA8	DATA8	Hi-z	Hi-z,O(L)	—	b12s
B15	DATA9	DATA9	Hi-z	Hi-z,O(L)	—	b12s
A17	DATA10	DATA10	Hi-z	Hi-z,O(L)	—	b12s
C14	DATA11	DATA11	Hi-z	Hi-z,O(L)	—	b12s
D15	DATA12	DATA12	Hi-z	Hi-z,O(L)	—	b12s
C15	DATA13	DATA13	Hi-z	Hi-z,O(L)	—	b12s
D14	DATA14	DATA14	Hi-z	Hi-z,O(L)	—	b12s
B17	DATA15	DATA15	Hi-z	Hi-z,O(L)	—	b12s
C16	DATA16	DATA16	Hi-z	Hi-z,O(L)	—	b12s
E15	DATA17	DATA17	Hi-z	Hi-z,O(L)	—	b12s
E14	DATA18	DATA18	Hi-z	Hi-z,O(L)	—	b12s
E13	DATA19	DATA19	Hi-z	Hi-z,O(L)	—	b12s
E12	DATA20	DATA20	Hi-z	Hi-z,O(L)	—	b12s
E16	DATA21	DATA21	Hi-z	Hi-z,O(L)	—	b12s
F15	DATA22	DATA22	Hi-z	Hi-z,O(L)	—	b12s
G13	DATA23	DATA23	Hi-z	Hi-z,O(L)	—	b12s
E17	DATA24	DATA24	Hi-z	Hi-z,O(L)	—	b12s
G12	DATA25	DATA25	Hi-z	Hi-z,O(L)	—	b12s
F14	DATA26	DATA26	Hi-z	Hi-z,O(L)	—	b12s
F12	DATA27	DATA27	Hi-z	Hi-z,O(L)	—	b12s
G11	DATA28	DATA28	Hi-z	Hi-z,O(L)	—	b12s
G16	DATA29	DATA29	Hi-z	Hi-z,O(L)	—	b12s
H13	DATA30	DATA30	Hi-z	Hi-z,O(L)	—	b12s
F13	DATA31	DATA31	Hi-z	Hi-z,O(L)	—	b12s
P12	DN0	DN0	—	—	AI	us
N11	DP0	DP0	—	—	AI	us
N12	DN1/PDN0	DN1	—/—	—	AI	us
U14	DP1/PDP0	DP1	—/—	—	AI	us
N17	EINT0/GPF0	GPF0	—/—	Hi-z/—	—	t8
M16	EINT1/GPF1	GPF1	—/—	Hi-z/—	—	t8
L13	EINT2/GPF2	GPF2	—/—	Hi-z/—	—	t8
M15	EINT3/GPF3	GPF3	—/—	Hi-z/—	—	t8
M17	EINT4/GPF4	GPF4	—/—	Hi-z/—	—	t8
L14	EINT5/GPF5	GPF5	—/—	Hi-z/—	—	t8
L15	EINT6/GPF6	GPF6	—/—	Hi-z/—	—	t8
L16	EINT7/GPF7	GPF7	—/—	Hi-z/—	—	t8
N9	EINT8/GPG0	GPG0	—/—	Hi-z/—	—	t8
T9	EINT9/GPG1	GPG1	—/—	Hi-z/—	—	t8
J10	EINT10/nSS0/GPG2	GPG2	—/—/—	Hi-z/Hi-z/—	—	t8
R10	EINT11/nSS1/GPG3	GPG3	—/—/—	Hi-z/Hi-z/—	—	t8
P11	EINT12/LCD_PWREN/GPG4	GPG4	—/—/—	Hi-z/O(L)/—	—	t8
K10	EINT13/SPIMISO1/GPG5	GPG5	—/—/—	Hi-z/Hi-z/—	—	t8
R11	EINT14/SPIMOSI1/GPG6	GPG6	—/—/—	Hi-z/Hi-z/—	—	t8
L10	EINT15/SPICLK1/GPG7	GPG7	—/—/—	Hi-z/Hi-z/—	—	t8
T10	EINT16/GPG8	GPG8	—/—	Hi-z/—	—	t8
M11	EINT17/nRTS1/GPG9	GPG9	—/—/—	Hi-z/O(H)/—	—	t8
N10	EINT18/nCTS1/GPG10	GPG10	—/—/—	Hi-z/Hi-z/—	—	t8
U12	EINT19/TCLK1/GPG11	GPG11	—/—/—	Hi-z/Hi-z/—	—	t12
M10	EINT20/GPG12	GPG12	—/—	Hi-z/—	—	t12
T11	EINT21/GPG13	GPG13	—/—	Hi-z/—	—	

表 1-2. S3C2440A 289 管脚 FBGA 引脚分配(3/6) (续)

管脚号	管脚名称	默认功能	I/O 口状态 @BUS REQ	I/O 口状态 @睡眠模式	I/O 口状态 @nRESET	I/O 口 类型
L11	EINT22/GPG14	GPG14	-/-	Hi-z/-	I	t12
U13	EINT23/GPG15	GPG15	-/-	Hi-z/-	I	t12
H12	EXTCLK	EXTCLK	-	-	AI	is
P17	UPLLCAP	UPLLCAP	-	-	AI	r50
N14	MPLLCP	MPLLCP	-	-	AI	r50
H14	nBATT_FLT	nBATT_FLT	-	-	I	is
D4	nBE0	nBE0	Hi-z	Hi-z,O(H)	O(H)	t10s
B5	nBE1	nBE1	Hi-z	Hi-z,O(H)	O(H)	t10s
D5	nBE2	nBE2	Hi-z	Hi-z,O(H)	O(H)	t10s
E5	nBE3	nBE3	Hi-z	Hi-z,O(H)	O(H)	t10s
R12	NCON	NCON	-	-	I	is
G6	FRnB	FRnB	-	Hi-z,O(L)	I	d2s
F3	nFWE/GPA19	GPA19	O(H)/-	Hi-z,O(H)/-	O(H)	t10s
E1	nFRE/GPA20	GPA20	O(H)/-	Hi-z,O(H)/-	O(H)	t10s
F4	nFCE/GPA22	GPA22	O(H)/-	Hi-z,O(H)/-	O(H)	t10s
F5	CLE/GPA17	GPA17	O(H)/-	Hi-z,O(H)/-	O(L)	t10s
D1	ALE/GPA18	GPA18	O(H)/-	Hi-z,O(H)/-	O(L)	t10s
N13	NRSTOUT/GPA21	GPA21	-/-	O(L)/-	O(L)	T8s
C5	nOE	nOE	Hi-z	Hi-z,O(H)	O(H)	t10s
H16	nRESET	nRESET	-	-	I	is
F6	nGCS0	nGCS0	Hi-z	Hi-z,O(H)	O(H)	t10s
B2	nGCS1/GPA12	GPA12	Hi-z/-	Hi-z,O(H)/-	O(H)	t10s
C3	nGCS2/GPA13	GPA13	Hi-z/-	Hi-z,O(H)/-	O(H)	t10s
C4	nGCS3/GPA14	GPA14	Hi-z/-	Hi-z,O(H)/-	O(H)	t10s
D3	nGCS4/GPA15	GPA15	Hi-z/-	Hi-z,O(H)/-	O(H)	t10s
C2	nGCS5/GPA16	GPA16	Hi-z/-	Hi-z,O(H)/-	O(H)	t10s
D2	nGCS6	nGCS6	Hi-z	Hi-z,O(H)	O(H)	t10s
E3	nGCS7	nGCS7	Hi-z	Hi-z,O(H)	O(H)	t10s
D6	nSCAS	nSCAS	Hi-z	Hi-z,O(H)	O(H)	t10s
C6	nSRAS	nSRAS	Hi-z	Hi-z,O(H)	O(H)	t10s
H15	nTRST	nTRST	I	-	I	is
E4	nWAIT	nWAIT	-	Hi-z,O(L)	I	d2s
E6	nWE	nWE	Hi-z	Hi-z,O(H)	O(H)	t10s
J6	TOUT0/GPB0	GPB0	-/-	O(L)/-	I	t8
J5	TOUT1/GPB1	GPB1	-/-	O(L)/-	I	t8
J7	TOUT2/GPB2	GPB2	-/-	O(L)/-	I	t8
K3	TOUT3/GPB3	GPB3	-/-	O(L)/-	I	t8
K4	TCLK0/GPB4	GPB4	-/-	-/-	I	t8
K2	nXBACK/GPB5	GPB5	-/-	O(H)/-	I	t8
L5	nXBREQ/GPB6	GPB6	-/-	-/-	I	t8
K7	nXDACK1/GPB7	GPB7	-/-	O(H)/-	I	t8
K5	nXDREQ1/GPB8	GPB8	-/-	-/-	I	t8
L3	nXDACK0/GPB9	GPB9	-/-	O(H)/-	I	t8
K6	nXDREQ0/GPB10	GPB10	-/-	-/-	I	t8
T15	OM0	OM0	-	-	I	is
R13	OM1	OM1	-	-	I	is
P13	OM2	OM2	-	-	I	is
T13	OM3	OM3	-	-	I	is
J12	PWREN	PWREN	O(H)	O(L)	O(H)	b8
K11	nCTS0/GPH0	GPH0	-/-	-/-	I	t8
L17	nRTS0/GPH1	GPH1	-/-	O(H)/-	I	t8
K13	TXD0/GPH2	GPH2	-/-	O(H)/-	I	t8
K14	RXD0/GPH3	GPH3	-/-	-/-	I	t8

表 1-2. S3C2440A 289 管脚 FBGA 引脚分配(4/6) (续)

管脚号	管脚名称	默认功能	I/O 口状态 @BUS REQ	I/O 口状态 @睡眠模式	I/O 口状态 @nRESET	I/O 口 类型
K16	TXD1/GPH4	GPH4	-/-	O(H)/-		t8
K17	RXD1/GPH5	GPH5	-/-	-/-		t8
J11	TXD2/nRTS1/GPH6	GPH6	-/-/-	O(H)/O(H)/-		t8
J15	RXD2/nCTS1/GPH7	GPH7	-/-/-	Hi-z/Hi-z/-		t8
K15	UEXTCLK/GPH8	GPH8	-/-	Hi-z/-		t8
R9	CLKOUT0/GPH9	GPH9	-/-	O(L)/-		t12
P10	CLKOUT1/GPH10	GPH10	-/-	O(L)/-		t12
A2	SCKE	SCKE	Hi-z	O(L)	O(H)	t10s
B4	SCLK0	SCLK0	Hi-z	O(L)	O(SCLK)	t12s
B3	SCLK1	SCLK1	Hi-z	O(L)	O(SCLK)	t12s
P7	I2SLRCK/AC_SYNC	GPE0	-/-	Hi-z/-		t8
R7	I2SSCLK/AC_BIT_CLK	GPE1	-/-	Hi-z/-		t8
T7	CDCLK/AC_nRESET	GPE2	-/-	Hi-z/-		t8
L8	CDCLK/AC_nRESET	GPE3	-/-/-	Hi-z/Hi-z/-		t8
U6	I2SSDO/AC_SDATA_OUT	GPE4	-/-/-	O(L)/Hi-z/-		t8
N8	SDCLK/GPE5	GPE5	-/-	O(L)/-		t8
K8	SDCMD/GPE6	GPE6	-/-	Hi-z/-		t8
R8	SDDAT0/GPE7	GPE7	-/-	Hi-z/-		t8
M8	SDDAT1/GPE8	GPE8	-/-	Hi-z/-		t8
P8	SDDAT2/GPE9	GPE9	-/-	Hi-z/-		t8
J9	SDDAT3/GPE10	GPE10	-/-	Hi-z/-		t8
K9	SPIMISO0/GPE11	GPE11	-/-	Hi-z/-		t8
P9	SPIMOSI0/GPE12	GPE12	-/-	Hi-z/-		t8
L9	SPICLK0/GPE13	GPE13	-/-	Hi-z/-		t8
U8	IICSCL/GPE14	GPE14	-/-	Hi-z/-		d8
M9	IICSDA/GPE15	GPE15	-/-	Hi-z/-		d8
J13	TCK	TCK		-		is
H17	TDI	TDI		-		is
J16	TDO	TDO	O	O	O	ot
J14	TMS	TMS		-		is
L1	LEND/GPC0	GPC0	-/-	O(L)/-		t8
L4	VCLK/GPC1	GPC1	-/-	O(L)/-		t8
M1	VLINE/GPC2	GPC2	-/-	O(L)/-		t8
L7	VFRAME/GPC3	GPC3	-/-	O(L)/-		t8
M4	VM/GPC4	GPC4	-/-	O(L)/-		t8
M3	LCD_LPCOE/GPC5	GPC5	-/-	O(L)/-		t8
M2	LCD_LPCREV/GPC6	GPC6	-/-	O(L)/-		t8
P1	LCD_LPCREVB/GPC7	GPC7	-/-	O(L)/-		t8
N2	VD0/GPC8	GPC8	-/-	O(L)/-		t8
L6	VD1/GPC9	GPC9	-/-	O(L)/-		t8
N4	VD2/GPC10	GPC10	-/-	O(L)/-		t8
R1	VD3/GPC11	GPC11	-/-	O(L)/-		t8
N3	VD4/GPC12	GPC12	-/-	O(L)/-		t8
P2	VD5/GPC13	GPC13	-/-	O(L)/-		t8
M6	VD6/GPC14	GPC14	-/-	O(L)/-		t8
P3	VD7/GPC15	GPC15	-/-	O(L)/-		t8
R2	VD8/GPD0	GPD0	-/-	O(L)/-		t8
M5	VD9/GPD1	GPD1	-/-	O(L)/-		t8
N5	VD10/GPD2	GPD2	-/-	O(L)/-		t8
R3	VD11/GPD3	GPD3	-/-	O(L)/-		t8
P4	VD12/GPD4	GPD4	-/-	O(L)/-		t8
R4	VD13/GPD5	GPD5	-/-/-	O(L)/O/-		t8
P5	VD14/GPD6	GPD6	-/-/-	O(L)/O/-		t8

表 1-2. S3C2440A 289 管脚 FBGA 引脚分配(5/6) (续)

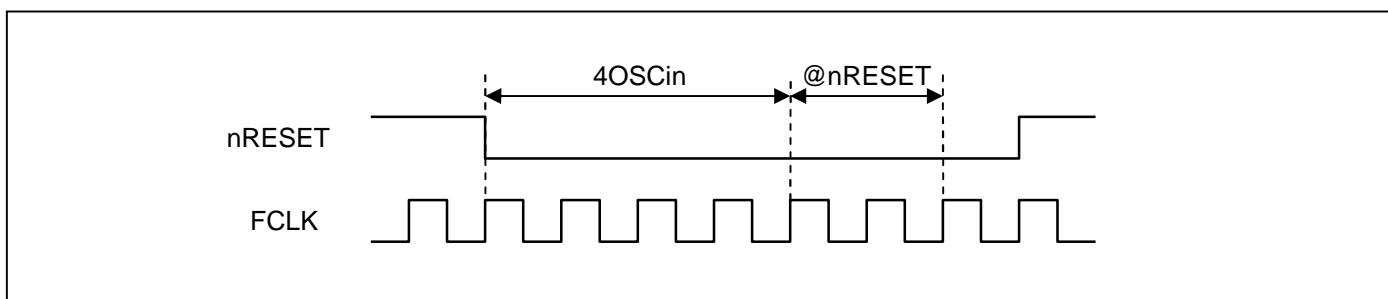
管脚号	管脚名称	默认功能	I/O 口状态 @BUS REQ	I/O 口状态 @睡眠模式	I/O 口状态 @nRESET	I/O 口 类型
N6	VD15/GPD7	GPD7	-/-/-	O(L)/O/-	I	t8
M7	VD16/SPIMISO1/GPD8	GPD8	-/-/-	O(L)/Hi-z/-	I	t8
T4	VD17/SPIMOSI1/GPD9	GPD9	-/-/-	O(L)/Hi-z/-	I	t8
R5	VD18/SPICLK1/GPD10	GPD10	-/-/-	O(L)/Hi-z/-	I	t8
T5	VD19//GPD11	GPD11	-/-/-	O(L)/Hi-z/-	I	t8
P6	VD20/ GPD12	GPD12	-/-/-	O(L)/Hi-z/-	I	t8
R6	VD21/ GPD13	GPD13	-/-/-	O(L)/Hi-z/-	I	t8
N7	VD22/nSS1/GPD14	GPD14	-/-/-	O(L)/Hi-z/-	I	t8
U5	VD23/nSS0/GPD15	GPD15	-/-/-	O(L)/Hi-z/-	I	t8
U16	Vref	Vref	-	-	AI	ia
G14	XTIpll	XTIpll	-	-	AI	m26
M14	Xtirtc	Xtirtc	-	-	AI	nc
G15	XTOppll	XTOppll	-	-	AO	m26
L12	Xtortc	Xtortc	-	-	AO	nc
N15	VDD_RTC	VDD_RTC	P	P	P	drtc
P14	VDDA_ADC	VDDA_ADC	P	P	P	d33th
N16	VDDA_MPLL	VDDA_MPLL	P	P	P	d12t
M13	VDDA_UPPLL	VDDA_UPPLL	P	P	P	d12t
G4	VDDalive	VDDalive	P	P	P	d12i
J17	VDDalive	VDDalive	P	P	P	d12i
A1	VDDi	VDDi	P	P	P	d12c
A10	VDDi	VDDi	P	P	P	d12c
A16	VDDi	VDDi	P	P	P	d12c
A6	VDDi	VDDi	P	P	P	d12c
B11	VDDi	VDDi	P	P	P	d12c
F1	VDDi	VDDi	P	P	P	d12c
F16	VDDi	VDDi	P	P	P	d12c
U11	VDDi	VDDi	P	P	P	d12c
L2	VDDiarm	VDDiarm	P	P	P	d12c
T6	VDDiarm	VDDiarm	P	P	P	d12c
T8	VDDiarm	VDDiarm	P	P	P	d12c
U1	VDDiarm	VDDiarm	P	P	P	d12c
J2	VDDiarm	VDDiarm	P	P	P	d12c
U2	VDDiarm	VDDiarm	P	P	P	d12c
A9	VDDMOP	VDDMOP	P	P	P	d33o
B12	VDDMOP	VDDMOP	P	P	P	d33o
B14	VDDMOP	VDDMOP	P	P	P	d33o
B16	VDDMOP	VDDMOP	P	P	P	d33o
B6	VDDMOP	VDDMOP	P	P	P	d33o
C1	VDDMOP	VDDMOP	P	P	P	d33o
F17	VDDMOP	VDDMOP	P	P	P	d33o
J1	VDDOP	VDDOP	P	P	P	d33o
T12	VDDOP	VDDOP	P	P	P	d33o
T3	VDDOP	VDDOP	P	P	P	d33o
K12	VDDOP	VDDOP	P	P	P	d33o
T14	VSSA_ADC	VSSA_ADC	P	P	P	sth
R17	VSSA_MPLL	VSSA_MPLL	P	P	P	st
M12	VSSA_UPPLL	VSSA_UPPLL	P	P	P	st
A12	VSSI	VSSI	P	P	P	si
A3	VSSI	VSSI	P	P	P	si
A4	VSSI	VSSI	P	P	P	si
B10	VSSI	VSSI	P	P	P	si
C17	VSSI	VSSI	P	P	P	si

表 1-2. S3C2440A 289 管脚 FBGA 引脚分配(6/6) (续)

管脚号	管脚名称	默认功能	I/O 口状态 @BUS REQ	I/O 口状态 @睡眠模式	I/O 口状态 @nRESET	I/O 口 类型
F2	VSSi	VSSi	P	P	P	si
G17	VSSi	VSSi	P	P	P	si
H1	VSSiarm	VSSiarm	P	P	P	si
K1	VSSiarm	VSSiarm	P	P	P	si
T1	VSSiarm	VSSiarm	P	P	P	si
T2	VSSiarm	VSSiarm	P	P	P	si
U10	VSSiarm	VSSiarm	P	P	P	si
U4	VSSiarm	VSSiarm	P	P	P	si
U7	VSSiarm	VSSiarm	P	P	P	si
A11	VSSMOP	VSSMOP	P	P	P	so
A15	VSSMOP	VSSMOP	P	P	P	so
A5	VSSMOP	VSSMOP	P	P	P	so
A7	VSSMOP	VSSMOP	P	P	P	so
B1	VSSMOP	VSSMOP	P	P	P	so
B13	VSSMOP	VSSMOP	P	P	P	so
D16	VSSMOP	VSSMOP	P	P	P	so
D17	VSSMOP	VSSMOP	P	P	P	so
E2	VSSMOP	VSSMOP	P	P	P	so
G1	VSSOP	VSSOP	P	P	P	so
N1	VSSOP	VSSOP	P	P	P	so
U15	VSSOP	VSSOP	P	P	P	so
U3	VSSOP	VSSOP	P	P	P	so
U9	VSSOP	VSSOP	P	P	P	so
H11	VSSOP	VSSOP	P	P	P	so

注释:

1. @BUS REQ 表示在外部总线该引脚状态，即总线被其它总线占用。
2. ' - ' 标记表明在总线请求模式下不变化引脚状态。.
3. Hi-z 或 Pre 意思为高阻态或之前状态并且由 MISCCR 寄存器的设置决定。
4. AI/AO 意思为模拟输入/模拟输出。
5. P, I 和 O 意思分别为电源，输入和输出。
6. I/O 口状态@nRESET 表示在 @nRESET 期间引脚状态，如下：



下表显示了 I/O 类型和描述

输入(I)/输出(O)类型	描述
d12i(vdd12ih)	1.2V V _{DD} 电源有效
d12c(vdd12ih_core), si(vssih=)	1.2V V _{DD} /V _{SS} 内部逻辑
d33o(vdd33oph), so(vssoph)	3.3V V _{DD} /V _{SS} 外部逻辑
d33th(vdd33th_abb),sth(vssbbh_abb)	3.3V V _{DD} /V _{SS} 模拟电路
d12t(vdd12t_abb), st(vssbb_abb)	1.2V V _{DD} /V _{SS} 模拟电路
drtc(vdd30th_RTC)	3.0V V _{DD} RCT 电源
t8(phbsu100ct8sm)	双向 pad , LVC MOS 施密特触发器 , 100kΩ 可控上拉电阻 , 三态 , I _O =8mA
is(phis)	输入向 pad , LVC MOS 施密特触发器电平
us(pbusb0)	USB 引脚
t10(phtot10cd)	承受 5V 电平输出引脚 , 三态
ot(phot8)	输出引脚 , 三态 , I _O =8mA
b8(phob8)	输出引脚 , I _O =8mA
t16(phot16sm)	输出引脚 , 三态 , 中等转换速率 , I _O =16mA
r10(phiar10_abb)	模拟输入引脚 , 含 10Ω 电阻
ia(phia_abb)	模拟输入引脚
gp(phgpad_option)	模拟引脚引脚
m26(phoscm26_2440a)	振荡器单元 , 使能和反馈电阻
t12(phbsu100ct12sm)	双向引脚 , LVC MOS 施密特触发器 , 100kΩ 可控上拉电阻 , 三态 , I _O =12mA
d8(phbsd8sm)	双向引脚 , LVC MOS 施密特触发器 , 开漏 , I _O =8mA
t10s(phtot10cd_10_2440a)	输出引脚 , LVC MOS , 三态 , 输出驱动强度控制 , I _O =4、6、8、10mA
b12s(phtbsu100ct12cd_12_2440a)	双向引脚 , LVC MOS 施密特触发器 , 100kΩ 可控上拉电阻 , 三态 , 输出驱动强度控制 , I _O =6、8、10、12mA
d2s(phtbsd2_2440a)	双向引脚 , LVC MOS 施密特触发器 , 开漏 , 忽略输出驱动强度
r50(phoar50_abb)	模拟输出引脚 , 50kΩ 电阻 , separated bulk-bias
t12s(phtot12cd_12_2440a)	输出 pad , LVC MOS , 三态 , 输出驱动强度控制 , I _O =6、8、10、12mA
nc(phnc)	未连接引脚

信号端描述

表 1-3. S3C2440A 信号端描述(1/6)

信号端	输入/输出	描述
总线控制		
OM[1:0]	I	OM[1:0]设置只用于生产时 S3C2440A 的测试模式；此外，它还决定了 nGCS0 的总线宽度。在 RESET 周期期间上拉/下拉电阻 确定其逻辑电平 00:Nand 启动 01:16 位 10:32 位 11:测试模式
ADDR[26:0]	O	ADDR[26:0] (地址总线)输出对应 bank 的存储器地址
DATA[31:0]	IO	DATA[31:0] (数据总线)当存储器读取时输入数据和存储器写入时输出数据；可编程总线宽度为 8/16/32 位
nGCS[7:0]	O	nGCS[7:0] (通用片选)当一个存储器的地址在每个 bank 中的寻址范围内被激活。可编程存取周期和 bank 大小
nWE	O	nWE (写使能) 表明当前总线周期为一个写周期
nOE	O	nOE (读使能) 表明当前总线周期为一个读周期
nXBREQ	I	nXBREQ(总线持有请求)允许另一个主机总线请求对本地总线的控制。返回激活表明同意控制总线权
nXBACK	O	nXBACK (总线持有应答)表明 S3C2440A 交出背地总线控制给其它主机总线
nWAIT	I	nWAIT 请求延长当前总线周期。只要 nWAIT 为低，不能完成当前总线周期
SDRAM/SRAM 存储器		
nSRAS	O	SDRAM 行地址选通
nSCAS	O	SDRAM 列地址选通
nSCS[1:0]	O	SDRAM 片选
DQM[3:0]	O	SDRAM 数据屏蔽
SCLK[1:0]	O	SDRAM 时钟
SCKE	O	SDRAM 时钟使能
nBE[3:0]	O	高位字节/低位字节使能(16 位 SRAM 情况时)
nWBE[3:0]	O	写字节使能
NAND Flash 存储器		
CLE	O	指令锁存使能
ALE	O	地址锁存使能
nFCE	O	Nand Flash 片选使能
nFRE	O	Nand Flash 读使能
nFWE	O	Nand Flash 写使能
NCON	I	Nand Flash 配置
FRnB	I	Nand Flash 就绪/忙
		*当 NAND Flash 控制器未启用，其必须被上拉。(3.3V)

表 1-3. S3C2440A 信号端描述(2/6) (续)

信号端	输入/输出	描述
LCD 控制单元		
VD[23:0]	O	STN/TFT/SEC TFT: LCD 数据总线
LCD_PWREN	O	STN/TFT/SEC TFT: LCD 面板电源使能控制信号
VCLK	O	STN/TFT: LCD 时钟信号
VFRAME	O	STN: LCD 帧信号
VLINE	O	STN: LCD 行 (line) 信号
VM	O	STN: VM 交替行和列电压的极性
VSYNC	O	TFT: 垂直同步信号
HSYNC	O	TFT: 水平同步信号
VDEN	O	TFT: 数据使能信号
LEND	O	TFT: 线路结束信号
STV	O	SEC TFT: SEC(三星电子公司)TFT LCD 面板控制信号
CPV	O	SEC TFT: SEC TFT LCD 面板控制信号
LCD_HCLK	O	SEC TFT: SEC TFT LCD 面板控制信号
TP	O	SEC TFT: SEC TFT LCD 面板控制信号
STH	O	SEC TFT: SEC TFT LCD 面板控制信号
LCD_LPCOE	O	SEC TFT: 特定 TFT LCD 时序控制信号
LCD_LPCREV	O	SEC TFT: 特定 TFT LCD 时序控制信号
LCD_LPCREVB	O	SEC TFT: 特定 TFT LCD 时序控制信号
摄像头接口		
CAMRESET	O	摄像头软件复位
CAMCLKOUT	O	摄像头的主机时钟
CAMPCLK	I	摄像头的像素时钟
CAMHREF	I	摄像头(输出)的水平同步信号
CAMVSYNC	I	摄像头(输出)的垂直同步信号
CAMDATA[7:0]	I	YCbCr 的像素数据
中断控制单元		
EINT[23:0]	I	外部中断请求
DMA		
nXDREQ[1:0]	I	外部 DMA 请求
nXDACK[1:0]	O	外部 DMA 应答
UART 异步串行接口		
RxD[2:0]	I	UART 接收数据输入
TxD[2:0]	O	UART 发送数据输出
nCTS[1:0]	I	UART 清除发送输入信号
nRTS[1:0]	O	UART 请求发送输出信号
UEXTCLK	I	UART 的外部时钟输入

表 1-3. S3C2440A 信号端描述(3/6) (续)

信号端	输入/输出	描述
ADC		
AIN[7:0]	AI	ADC 输入[7:0]；若未使用该引脚，将其设置为低电平（地）
Vref	AI	ADC Vref 基准
IIC 两线串行总线接口		
IICSDA	IO	IIC 总线数据线
IISDA	IO	IIC 总线时钟线
IIS 总线		
I2SLRCK	IO	IIS 总线通道时钟选择
I2SSDO	O	IIS 总线串行数据输出
I2SSDI	I	IIS 总线串行数据输入
I2SSCLK	IO	IIS 总线
CDCLK	O	编码系统时钟
AC'97 接口		
AC_SYNC	O	48kHz 固定频率同步采样
AC_BIT_CLK	IO	12.288MHz 串行数据时钟
AC_nRESET	O	AC'97 主机读/写复位
AC_SDATA_IN	I	串行，时分复用，AC'97 输入流
AC_SDATA_OUT	O	串行，时分复用，AC'97 输出流
触摸屏接口		
nXPON	O	正 X 轴开关控制信号
XMON	O	负 X 轴开关控制信号
nYPON	O	正 Y 轴开关控制信号
YMON	O	负 Y 轴开关控制信号
USB 主机		
DN[1:0]	IO	USB 主机的 DATA(-)；(需要 15KΩ 下拉电阻)
DP[1:0]	IO	USB 主机的 DATA(+)；(需要 15KΩ 下拉电阻)
USB 设备		
PDN0	IO	USB 外设的 DATA(-)；(睡眠模式需要 470KΩ 下拉电阻消耗电源)
PDP0	IO	USB 外设的 DATA(+)；(需要 470KΩ 上拉电阻)

表 1-3. S3C2440A 信号端描述(4/6) (续)

信号端	输入/输出	描述
SPI 同步串行接口		
SPIMISO[1:0]	IO	当 SPI 配置为主机时 SPIMISO 为主机数据输入线。 当 SPI 配置为从机时，该引脚颠倒其作用。
SPIMISO[1:0]	IO	当 SPI 配置为主机时 SPIMISO 为主机数据输入线。 当 SPI 配置为从机时，该引脚颠倒其作用。
SPICLK[1:0]	IO	SPI 时钟
nSS[1:0]	I	SPI 片选(只用于从机模式)
SD 接口		
SDDAT[3:0]	IO	SD 接收/发送数据
SDCMD	IO	SD 接应收应答/发送指令
SDCLK	O	SD 时钟
通用 I/O 端口		
GPn[129:0]	IO	通用输入/输出端口 (一些端口只能输出)
定时器/PWM		
TOUT[3:0]	O	定时器输出[3:0]
TCLK[1:0]	I	外部定时器时钟输入
JTAG 逻辑测试		
nTRST	I	nTRST (TAP 控制器复位) 启动时复位 TAP 控制器 若使用调试器，接 10K 的上拉电阻 若未使用调试器(拒绝 ICE)，nTRST 引脚必须由一个有效低电平脉冲发出(通常连接到 nRESET)。
TMS	I	TMS (TAP 控制器模式选择) TAP 控制器状态相关控制 TMS 引脚连接 10K 的上拉电阻
TCK	I	TCK (TAP 控制器时钟) 提供逻辑 JTAG 的时钟输入 TCK 引脚连接 10K 的上拉电阻
TDI	I	TDI (TAP 控制器数据输入) 是测试指令和数据的串行输入端 TDI 引脚连接 10K 的上拉电阻
TDO	O	TDO (TAP 控制器数据输出) 是测试指令和数据的串行输出端

表 1-3. S3C2440A 信号端描述(5/6) (续)

信号端	输入/输出	描述
复位，时钟和电源		
XTOpll	AO	内部振荡电路的晶体振荡器输出 当 OM[3:2] = 00b , XTIpII 用于 MPLL 时钟源和 UPLL 时钟源 当 OM[3:2] = 01b , XTIpII 只用于 MPLL 时钟源 当 OM[3:2] = 10b , XTIpII 只用于 UPLL 时钟源 若未启用，将其悬空
MPLLCP	AI	主时钟环路滤波电容
UPLLCP	AI	USB 时钟环路滤波电容
XTIrtc	AI	RTC 的 32kHz 晶振输入。若未启用，将其接高电平(3.3V)
XTOrtc	AO	RTC 的 32kHz 晶振输出。若未启用，将其悬空
CLKOUT[1:0]	O	时钟信号输出。MISCCR 寄存器的 CLKSEL 位配置时钟输出模式，可以为 MPLL 时钟，UPLL 时钟，FCLK，HCLK，PCLK
nRESET	ST	nRESET 停止任何进行中的操作，并将 S3C2440A 置为一个已知的复位状态。 对于复位，nRESET 必须在处理器电源稳定后至少保持 4 个 OSCin 低电平
nRSTOUT	O	用于外部设备复位控制 (nRSTOUT = nRESET & nWDTRST & SW_RESET)
PWREN	O	1.2V/1.3V 内核供电开关控制信号
nBATT_FLT	I	电池状态传感器 (低电池状态时并不唤醒睡眠模式)。若未启用，将其接高电平(3.3V)
OM[3:2]	I	OM[3:2]确定时钟模式选择 OM[3:2] = 00b , 晶振用于 MPLL 时钟源和 UPLL 时钟源 OM[3:2] = 01b , 晶振用于 MPLL 时钟源并且 EXTCLK 为 UPLL 时钟源 OM[3:2] = 10Bb , EXTCLK 用于 MPLL 时钟源并且晶振为 UPLL 时钟源 OM[3:2] = 11Bb , EXTCLK 用于 MPLL 时钟源和 UPLL 时钟源
EXTCLK	I	外部时钟源 当 OM[3:2] = 11b , EXTCLK 用于 MPLL 时钟源和 UPLL 时钟源 当 OM[3:2] = 10b , EXTCLK 只用于 MPLL 时钟源 当 OM[3:2] = 01b , EXTCLK 只用于 UPLL 时钟源 若未启用，将其接高电平(3.3V)
XTIpII	AI	内部振荡电路的晶体振荡器输入 当 OM[3:2] = 00b , XTIpII 用于 MPLL 时钟源和 UPLL 时钟源 当 OM[3:2] = 01b , XTIpII 只用于 MPLL 时钟源 当 OM[3:2] = 10b , XTIpII 只用于 UPLL 时钟源 若未启用，将 XTIpII 接高电平(3.3V)

表 1-3. S3C2440A 信号端描述(6/6) (续)

信号端	输入/输出	描述
电源		
VDDalive	P	复位模块和端口状态寄存器电源 VDD 无论正常模式或休眠模式，都应该始终供电
VDDiarm	P	ARM 内核的核心逻辑电源 VDD
VDDi	P	内部模块核心逻辑电源 VDD
VSSi/VSSiarm	P	核心逻辑地 VSS
VDDi_MPLL	P	MPLL 模拟和数字电源 VDD
VSSi_MPLL	P	MPLL 模拟和数字地 VSS.
VDDOP	P	I/O 端口电源 VDD (3.3V)
VDDMOP	P	存储器 I/O 口电源 VDD 3.3V : SCLK 最高到 135MHz 2.5V : SCLK 最高到 135MHz 1.8V : SCLK 最高到 93MHz
VSSOP	P	I/O 端口地 VSS
RTCVDD	P	RTC 电源 VDD (3.0V , 输入范围 : 1.8 ~ 3.6V) 若 RTC 未启用，该引脚必须适当的接电源
VDDi_UPLL	P	UPLL 模拟和数字电源 VDD
VSSi_UPLL	P	UPLL 模拟和数字地 VSS
VDDA_ADC	P	ADC 电源 VDD(3.3V)
VSSA_ADC	P	ADC 地 VSS

注释:

1. **I/O** 的意思为输入/输出。
2. **AI/AO** 的意思为模拟输入/模拟输出。.
3. **ST** 的意思为施密特触发器。
4. **P** 的意思为电源。

S3C2440A 特殊寄存器

表 1-4. S3C2440A 特殊寄存器(1/11)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
存储器控制器					
BWSCON	0x48000000				总线宽度和等待状态控制
BANKCON0	0x48000004				启动 ROM 控制
BANKCON1	0x48000008				BANK1 控制
BANKCON2	0x4800000C				BANK2 控制
BANKCON3	0x48000010				BANK3 控制
BANKCON4	0x48000014				BANK4 控制
BANKCON5	0x48000018				BANK5 控制
BANKCON6	0x4800001C				BANK6 控制
BANKCON7	0x48000020				BANK7 控制
REFRESH	0x48000024				DRAM/SDRAM 刷新控制
BANKSIZE	0x48000028				可变 Bank 大小
MRSRB6	0x4800002C				SDRAM BANK6 的模式寄存器设置
MRSRB7	0x48000030				SDRAM BANK7 的模式寄存器设置
USB 主机控制器					
HcRevision	0x49000000				控制和状态组
HcControl	0x49000004				
HcCommonStatus	0x49000008				
HcInterruptStatus	0x4900000C				
HcInterruptEnable	0x49000010				
HcInterruptDisable	0x49000014				
HcHCCA	0x49000018				
HcPeriodCuttentED	0x4900001C				
HcControlHeadED	0x49000020				
HcControlCurrentED	0x49000024				
HcBulkHeadED	0x49000028				内存指针组
HcBulkCurrentED	0x4900002C				
HcDoneHead	0x49000030				
HcRmInterval	0x49000034				
HcFmRemaining	0x49000038				
HcFmNumber	0x4900003C				
HcPeriodicStart	0x49000040				
HcLSThreshold	0x49000044				
HcRhDescriptorA	0x49000048				帧控制组
HcRhDescriptorB	0x4900004C				
HcRhStatus	0x49000050				
HcRhPortStatus1	0x49000054				
HcRhPortStatus2	0x49000058				逻辑根集线器组

表 1-4. S3C2440A 特殊寄存器(2/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能	
中断控制器						
SRCPND	0X4A000000	W	W	R/W	中断请求状态	
INTMOD	0X4A000004			W	中断模式控制	
INTMSK	0X4A000008			R/W	中断屏蔽控制	
PRIORITY	0X4A00000C			W	IRQ 优先级控制	
INTPND	0X4A000010			R/W	中断请求状态	
INTOFFSET	0X4A000014			R	中断请求源偏移	
SUBSRCPND	0X4A000018			R/W	次级源挂起	
INTSUBMSK	0X4A00001C			R/W	中断分支屏蔽	
DMA						
DISRC0	0x4B000000	W	R/W	DMA 0 初始源	DMA 0 初始源	
DISRCC0	0x4B000004				DMA 0 初始源控制	
DIDST0	0x4B000008				DMA 0 初始目标	
DIDSTC0	0x4B00000C				DMA 0 初始目标控制	
DCON0	0x4B000010				DMA 0 控制	
DSTAT0	0x4B000014		R	DMA 0 计数	DMA 0 计数	
DCSRC0	0x4B000018				DMA 0 当前源	
DCDST0	0x4B00001C				DMA 0 当前目标	
DMASKTRIG0	0x4B000020		R/W	DMA 0 屏蔽触发	DMA 0 屏蔽触发	
DISRC1	0x4B000040	W	R/W		DMA 1 初始源	
DISRCC1	0x4B000044				DMA 1 初始源控制	
DIDST1	0x4B000048				DMA 1 初始目标	
DIDSTC1	0x4B00004C				DMA 1 初始目标控制	
DCON1	0x4B000050				DMA 1 控制	
DSTAT1	0x4B000054		R	DMA 1 计数	DMA 1 计数	
DCSRC1	0x4B000058				DMA 1 当前源	
DCDST1	0x4B00005C				DMA 1 当前目标	
DMASKTRIG1	0x4B000060		R/W	DMA 1 屏蔽触发	DMA 1 屏蔽触发	
DISRC2	0x4B000080	W	R/W		DMA 2 初始源	
DISRCC2	0x4B000084				DMA 2 初始源控制	
DIDST2	0x4B000088				DMA 2 初始目标	
DIDSTC2	0x4B00008C				DMA 2 初始目标控制	
DCON2	0x4B000090				DMA 2 控制	
DSTAT2	0x4B000094		R	DMA 2 计数	DMA 2 计数	
DCSRC2	0x4B000098				DMA 2 当前源	
DCDST2	0x4B00009C				DMA 2 当前目标	
DMASKTRIG2	0x4B0000A0		R/W	DMA 0 屏蔽触发	DMA 0 屏蔽触发	

表 1-4. S3C2440A 特殊寄存器(3/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
DMA (续)					
DISRC3	0x4B0000C0	← W	R/W	DMA 3 初始源	
DISRCC3	0x4B0000C4			DMA 3 初始源控制	
DIDST3	0x4B0000C8			DMA 3 初始目标	
DIDSTC3	0x4B0000CC			DMA 3 初始目标控制	
DCON3	0x4B0000D0			DMA 3 控制	
DSTAT3	0x4B0000D4		R	DMA 3 计数	
DCSRC3	0x4B0000D8			DMA 3 当前源	
DCDST3	0x4B0000DC			DMA 3 当前目标	
DMASKTRIG3	0x4B0000E0		R/W	DMA 3 屏蔽触发	
时钟和电源管理					
LOCKTIME	0x4C000000	← W	R/W	PLL 锁定时间计数器	
MPLLCON	0x4C000004			MPLL 控制	
UPLLCON	0x4C000008			UPLL 控制	
CLKCON	0x4C00000C			时钟生成控制	
CLKSLOW	0x4C000010			慢时钟控制	
CLKDIVN	0x4C000014			时钟分频控制	
CAMDIVN	0x4C000018			摄像头时钟分频控制	
LCD 控制器					
LCDCON1	0X4D000000	← W	R/W	LCD 控制 1	
LCDCON2	0X4D000004			LCD 控制 2	
LCDCON3	0X4D000008			LCD 控制 3	
LCDCON4	0X4D00000C			LCD 控制 4	
LCDCON5	0X4D000010			LCD 控制 5	
LCDSADDR1	0X4D000014			STN/TFT : 帧缓冲区开始地址 1	
LCDSADDR2	0X4D000018			STN/TFT : 帧缓冲区开始地址 2	
LCDSADDR3	0X4D00001C			STN/TFT : 虚拟屏幕地址设置	
REDLUT	0X4D000020			STN : 红色查找表	
GREENLUT	0X4D000024			STN : 绿色查找表	
BLUELUT	0X4D000028			STN : 蓝色查找表	
DITHMODE	0X4D00004C			STN : 抖动模式	
TPAL	0X4D000050			TFT : 临时调色板	
LCDINTPND	0X4D000054			LCD 中断等待	
LCDSRCPND	0X4D000058			LCD 中断源	
LCDINTMSK	0X4D00005C			LCD 中断屏蔽	
TCONSEL	0X4D000060			TCON(LPC3600/LCC3600)控制	

表 1-4. S3C2440A 特殊寄存器(4/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
NAND Flash					
NFCNF	0x4E000000	W	R/W	NAND Flash 配制	
NFCONT	0x4E000004			NAND Flash 控制	
NFCMD	0x4E000008			NAND Flash 指令	
NFADDR	0x4E00000C			NAND Flash 地址	
NFDATA	0x4E000010			NAND Flash 数据	
NFMECC0	0x4E000014			NAND Flash 主区域 ECC0/1	
NFMECC1	0x4E000018			NAND Flash 主区域 ECC2/3	
NFSECC	0x4E00001C			NAND Flash 空闲区域 ECC	
NFSTAT	0x4E000020			NAND Flash 运行状态	
NFESTAT0	0x4E000024			NAND Flash I/O[7:0]ECC 状态	
NFESTAT1	0x4E000028			NAND Flash I/O[15:8]ECC 状态	
NFMECC0	0x4E00002C		R	NAND Flash 主区域 ECC0 状态	
NFMECC1	0x4E000030			NAND Flash 主区域 ECC1 状态	
NFSECC	0x4E000034			NAND Flash 空闲区域 ECC 状态	
NFSBLK	0x4E000038	W	R/W	NAND Flash 开始块地址	
NFEBLK	0x4E00003C			NAND Flash 结束块地址	
摄像头接口					
CISRCFMT	0x4F000000	W	R/W	输入源格式	
CIWDOFST	0x4F000004			窗口偏移寄存器	
CIGCTRL	0x4F000008			全局控制寄存器	
CICOYSA1	0x4F000018			编码 DMA Y 第一帧开始地址	
CICOYSA2	0x4F00001C			编码 DMA Y 第二帧开始地址	
CICOYSA3	0x4F000020			编码 DMA Y 第三帧开始地址	
CICOYSA4	0x4F000024			编码 DMA Y 第四帧开始地址	
CICOCBSA1	0x4F000028			编码 DMA Cb 第一帧开始地址	
CICOCBSA2	0x4F00002C			编码 DMA Cb 第二帧开始地址	
CICOCBSA3	0x4F000030			编码 DMA Cb 第三帧开始地址	
CICOCBSA4	0x4F000034			编码 DMA Cb 第四帧开始地址	
CICOCRSA1	0x4F000038			编码 DMA Cr 第一帧开始地址	
CICOCRSA2	0x4F00003C			编码 DMA Cr 第二帧开始地址	
CICOCRSA3	0x4F000040			编码 DMA Cr 第三帧开始地址	
CICOCRSA4	0x4F000044			编码 DMA Cr 第四帧开始地址	
CICOTRGFMT	0x4F000048			编码 DMA 目标图像格式	
CICOCTRL	0x4F00004C			编码 DMA 关系控制	
CICOSCPRERATIO	0x4F000050			编码预缩放比控制	
CICOSCPRDST	0x4F000054			编码预缩放目标格式	

表 1-4. S3C2440A 特殊寄存器(5/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
摄像头接口(续)					
CICOSCCTRL	0x4F000058	W	R/W	编码主缩放控制	
CICOTAREA	0x4F00005C			编码缩放目标区域	
CICOSTATUS	0x4F000064			编码通路状态	
CIPRCLRSA1	0x4F00006C			预览 DMA RGB 第一帧开始地址	
CIPRCLRSA2	0x4F000070			预览 DMA RGB 第二帧开始地址	
CIPRCLRSA3	0x4F000074			预览 DMA RGB 第三帧开始地址	
CIPRCLRSA4	0x4F000078			预览 DMA RGB 第四帧开始地址	
CIPRTRGFM	0x4F00007C			预览 DMA 目标图像格式	
CIPRCTRL	0x4F000080			预览 DMA 关系控制	
CIPRSCPFRATIO	0x4F000084			预览预缩放比控制	
CIPRSCPREDST	0x4F000088			预览预缩放目标格式	
CIPRSCCTRL	0x4F00008C			预览主缩放控制	
CIPRTAREA	0x4F000090			预览缩放目标区域	
CIPRSTATUS	0x4F000098			预览通路状态	
CIIMGCPT	0x4F0000A0			图像采集使能命令	
UART					
ULCON0	0x50000000	W	R/W	UART 0 线路控制	
UCON0	0x50000004			UART 0 控制	
UFCON0	0x50000008			UART 0 FIFO 控制	
UMCON0	0x5000000C			UART 0 Modem 控制	
UTRSTAT0	0x50000010		R	UART 0 Tx/Rx 状态	
UERSTAT0	0x50000014			UART 0 Rx 错误状态	
UFSTAT0	0x50000018			UART 0 FIFO 状态	
UMSTAT0	0x5000001C			UART 0 Modem 状态	
UTXH0	0x50000023	B	W	UART 0 发送保持	
URXH0	0x50000027		R	UART 0 接收缓冲器	
UBRDIV0	0x50000028	←	W	R/W	UART 0 波特率分频器
ULCON1	0x50004000	W	R/W	UART 1 线路控制	
UCON1	0x50004004			UART 1 控制	
UFCON1	0x50004008			UART 1 FIFO 控制	
UMCON1	0x5000400C			UART 1 Modem 控制	
UTRSTAT1	0x50004010		R	UART 1 Tx/Rx 状态	
UERSTAT1	0x50004014			UART 1 Rx 错误状态	
UFSTAT1	0x50004018			UART 1 FIFO 状态	
UMSTAT1	0x5000401C			UART 1 Modem 状态	
UTXH1	0x50004023	0x50004020	B	R	UART 1 发送保持

表 1-4. S3C2440A 特殊寄存器(6/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
UART (续)					
URXH1	0x50004027	0x50004024	B	R/W	UART 1 接收缓冲器
UBRDIV1	0x50004028		←	W	UART 1 波特率分频器
ULCON2	0x50008000		W	R/W	UART 2 线路控制
UCON2	0x50008004				UART 2 控制
UFCON2	0x50008008				UART 2 FIFO 控制
UMCON2	0x5000800C				UART 2 Modem 控制
UTRSTAT2	0x50008010			R	UART 2 Tx/Rx 状态
UERSTAT2	0x50008014				UART 2 Rx 错误状态
UFSTAT2	0x50008018				UART 2 FIFO 状态
UMSTAT2	0x5000801C				UART 2 Modem 状态
UTXH2	0x50008023	0x50008020	B	W	UART 2 发送保持
URXH2	0x50008027	0x50008024		R	UART 2 接收缓冲器
UBRDIV2	0x50008028	←	W	R/W	UART 2 波特率分频器
PWM 定时器					
TCFG0	0x51000000		W	R/W	定时器配制
TCFG1	0x51000004				定时器配制
TCON	0x51000008				定时器控制
TCNTB0	0x5100000C				定时器计数缓冲器 0
TCMPB0	0x51000010				定时器比较缓冲器 0
TCNTO0	0x51000014			R	定时器计数监视器 0
TCNTB1	0x51000018			R/W	定时器计数缓冲器 1
TCMPB1	0x5100001C				定时器比较缓冲器 1
TCNTO1	0x51000020			R	定时器计数监视器 1
TCNTB2	0x51000024			R/W	定时器计数缓冲器 2
TCMPB2	0x51000028				定时器比较缓冲器 2
TCNTO2	0x5100002C			R	定时器计数监视器 2
TCNTB3	0x51000030			R/W	定时器计数缓冲器 3
TCMPB3	0x51000034				定时器比较缓冲器 3
TCNTO3	0x51000038			R	定时器计数监视器 3
TCNTB4	0x5100003C			R/W	定时器计数缓冲器 4
TCNTO4	0x51000040		R		定时器计数监视器 4
USB 设备					
FUNC_ADDR_REG	0x52000143	0x52000140	B	R/W	工作地址
PWR_REG	0x52000147	0x52000144			电源管理
EP_INT_REG	0x5200014B	0x52000148			EP 中断等待和清除
USB_INT_REG	0x5200015B	0x52000158			USB 中断等待和清除

表 1-4. S3C2440A 特殊寄存器(7/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能	
USB 设备(续)						
EP_INT_EN_REG	0x5200015F	0x5200015C	B	R/W	中断使能	
USB_INT_EN_REG	0x5200016F	0x5200016C			中断使能	
FRAME_NUM1_REG	0x52000173	0x52000170		R	帧号低字节	
FRAME_NUM2_REG	0x52000177	0x52000174			帧号高字节	
INDEX_REG	0x5200017B	0x52000178	B	R/W	寄存器索引	
EP0_CSR	0x52000187	0x52000184			端点状态	
IN_CSR1_REG	0x52000187	0x52000184			入端点控制状态	
IN_CSR2_REG	0x5200018B	0x52000188			入端点控制状态	
MAXP_REG	0x52000183	0x52000180			端点最大包	
OUT_CSR1_REG	0x52000193	0x52000190			出端点控制状态	
OUT_CSR2_REG	0x52000197	0x52000194	B	R	出端点控制状态	
OUT_FIFO_CNT1_REG	0x5200019B	0x52000198			端点出写计数器	
OUT_FIFO_CNT2_REG	0x5200019F	0x5200019C			端点出写计数器	
EP0_FIFO	0x520001C3	0x520001C0		R/W	端点 0 FIFO	
EP1_FIFO	0x520001C7	0x520001C4	B		端点 1 FIFO	
EP2_FIFO	0x520001CB	0x520001C8			端点 2 FIFO	
EP3_FIFO	0x520001CF	0x520001CC			端点 3 FIFO	
EP4_FIFO	0x520001D3	0x520001D0			端点 4 FIFO	
EP1_DMA_CON	0x52000203	0x52000200			EP1 DMA 接口控制	
EP1_DMA_UNIT	0x52000207	0x52000204			EP1 DMA Tx 单元计数器	
EP1_DMA_FIFO	0x5200020B	0x52000208			EP1 DMA Tx FIFO 计数器	
EP1_DMA_TTC_L	0x5200020F	0x5200020C			EP1 DMA 总 Tx 计数器	
EP1_DMA_TTC_M	0x52000213	0x52000210			EP1 DMA 总 Tx 计数器	
EP1_DMA_TTC_H	0x52000217	0x52000214			EP1 DMA 总 Tx 计数器	
EP2_DMA_CON	0x5200021B	0x52000218			EP2 DMA 接口控制	
EP2_DMA_UNIT	0x5200021F	0x5200021C			EP2 DMA Tx 单元计数器	
EP2_DMA_FIFO	0x52000223	0x52000220			EP2DMA Tx FIFO 计数器	
EP2_DMA_TTC_L	0x52000227	0x52000224			EP2 DMA 总 Tx 计数器	
EP2_DMA_TTC_M	0x5200022B	0x52000228			EP2 DMA 总 Tx 计数器	
EP2_DMA_TTC_H	0x5200022F	0x5200022C			EP2 DMA 总 Tx 计数器	
EP3_DMA_CON	0x52000243	0x52000240			EP3 DMA 接口控制	
EP3_DMA_UNIT	0x52000247	0x52000244			EP3 DMA Tx 单元计数器	
EP3_DMA_FIFO	0x5200024B	0x52000248			EP3 DMA Tx FIFO 计数器	
EP3_DMA_TTC_L	0x5200024F	0x5200024C			EP3 DMA 总 Tx 计数器	
EP3_DMA_TTC_M	0x52000253	0x52000250			EP3 DMA 总 Tx 计数器	
EP3_DMA_TTC_H	0x52000257	0x52000254			EP3 DMA 总 Tx 计数器	

表 1-4. S3C2440A 特殊寄存器(8/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能	
USB 设备(续)						
EP4_DMA_CON	0x5200025B	0x52000258	B	R/W	EP4 DMA 接口控制	
EP4_DMA_UNIT	0x5200025F	0x5200025C			EP4 DMA Tx 单元计数器	
EP4_DMA_FIFO	0x52000263	0x52000260			EP4 DMA Tx FIFO 计数器	
EP4_DMA_TTC_L	0x52000267	0x52000264			EP4 DMA 总 Tx 计数器	
EP4_DMA_TTC_M	0x5200026B	0x52000268			EP4 DMA 总 Tx 计数器	
EP4_DMA_TTC_H	0x5200026F	0x5200026C			EP4 DMA 总 Tx 计数器	
看门狗定时器						
WTCON	0x53000000	0x53000000	←	W	R/W	看门狗定时器模式
WTDAT	0x53000004	0x53000004				看门狗定时器数据
WTCNT	0x53000008	0x53000008				看门狗定时器计数
IIC						
IICCON	0x54000000	0x54000000	←	W	R/W	IIC 控制
IICSTAT	0x54000004	0x54000004				IIC 状态
IICADD	0x54000008	0x54000008				IIC 地址
IICDS	0x5400000C	0x5400000C				IIC 数据变换
IICLC	0x54000010	0x54000010				IIC 多主机线控制
IIS						
IISCON	0x55000000,02	0x55000000	HW,W	R/W	IIS 控制	IIS 控制
IISMOD	0x55000004,06	0x55000004				IIS 模式
IISPSR	0x55000008,0A	0x55000008				IIS 预标定器
IISFCON	0x5500000C,0E	0x5500000C				IIS FIFO 控制
IISIFO	0x55000012	0x55000010				IIS FIFO 入口
输入/输出端口						
GPACON	0x56000000	0x56000000	←	W	R/W	端口 A 控制
GPADAT	0x56000004	0x56000004				端口 A 数据
GPBCON	0x56000010	0x56000010				端口 B 控制
GPBDAT	0x56000014	0x56000014				端口 B 数据
GPBUP	0x56000018	0x56000018				上拉控制 B
GPCCON	0x56000020	0x56000020				端口 C 控制
GPCDAT	0x56000024	0x56000024				端口 C 数据
GPCUP	0x56000028	0x56000028				上拉控制 C
GPDCON	0x56000030	0x56000030				端口 D 控制
GPDDA1T	0x56000034	0x56000034				端口 D 数据
GPDUP	0x56000038	0x56000038				上拉控制 D
GPECON	0x56000040	0x56000040				端口 E 控制
GPEDAT	0x56000044	0x56000044				端口 E 数据

表 1-4. S3C2440A 特殊寄存器(9/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
输入/输出端口(续)					
GPEUP	0x56000048				上拉控制 E
GPFCON	0x56000050				端口 F 控制
GPFDAT	0x56000054				端口 F 数据
GPFUP	0x56000058				上拉控制 F
PGPCON	0x56000060				端口 G 控制
PGPDAT	0x56000064				端口 G 数据
PGPUP	0x56000068				上拉控制 G
PGHCON	0x56000070				端口 H 控制
PGHDAT	0x56000074				端口 H 数据
PGHUP	0x56000078				上拉控制 H
GPJCON	0x560000D0			R/W	端口 J 控制
GPJDAT	0x560000D4				端口 J 数据
GPJUP	0x560000D8				上拉控制 J
MISCCR	0x56000080				杂项控制
DCLKCON	0x56000084				DCLK0/1 控制
EXTINT0	0x56000088				外部中断控制寄存器 0
EXTINT1	0x5600008C				外部中断控制寄存器 1
EXTINT2	0x56000090				外部中断控制寄存器 2
EINTFLT0	0x56000094				保留
EINTFLT1	0x56000098				保留
EINTFLT2	0x5600009C				外部中断滤波控制寄存器 2
EINTFLT3	0x560000A0				外部中断滤波控制寄存器 2
EINTMASK	0x560000A4				外部中断屏蔽
EINTPEND	0x560000A8				外部中断等待
GSTATUS0	0x560000AC			R	外部引脚状态
GSTATUS1	0x560000B0				片上 ID
GSTATUS2	0x560000B4				复位状态
GSTATUS3	0x560000B8			R/W	通告寄存器
GSTATUS4	0x560000BC				通告寄存器
MSLCON	0x560000CC				存储器睡眠控制寄存器

表 1-4. S3C2440A 特殊寄存器(10/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
RTC					
RTCCON	0x57000043	0x57000040	B R/W		RTC 控制
TICNT	0x57000047	0x57000044			节拍时间控制
RTCALM	0x57000053	0x57000050			RTC 闹铃控制
ALMSEC	0x57000057	0x57000054			闹铃秒
ALMMIN	0x5700005B	0x57000058			闹铃分
ALMHOUR	0x5700005F	0x5700005C			闹铃时
ALMDATE	0x57000063	0x57000060			闹铃天
ALMMON	0x57000067	0x57000064			闹铃月
ALMYEAR	0x5700006B	0x57000068			闹铃年
BCDSEC	0x57000073	0x57000070			BCD 秒
BCDMIN	0x57000077	0x57000074			BCD 分
BCDHOUR	0x5700007B	0x57000078			BCD 时
BCDDATE	0x5700007F	0x5700007C			BCD 天
BCDDAY	0x57000083	0x57000080			BCD 日期
BCDMON	0x57000087	0x57000084			BCD 月
BCDYEAR	0x5700008B	0x57000088			BCD 年
A/D 转换器					
ADCCON	0x58000000	W ←	R/W		ADC 控制
ADCTSC	0x58000004				ADC 触屏控制
ADCDLY	0x58000008				ADC 开始或间隔延时
ADCDAT0	0x5800000C		R		ADC 转换数据
ADCDAT1	0x58000010				ADC 转换数据
ADCUPDN	0x58000014		R/W		笔尖提起或点下中断状态
SPI					
SPCON0,1	0x59000000,20	W ←	R/W		SPI 控制
SPSTA0,1	0x59000004,24				SPI 状态
SPPIN0,1	0x59000008,28				SPI 引脚控制
SPPRE0,1	0x5900000C,2C		R/W		SPI 波特率预标定器
SPTDAT0,1	0x59000010,30				SPI Tx 数据
SPRDAT0,1	0x59000014,34		R		SPI Rx 数据

表 1-4. S3C2440A 特殊寄存器(11/11) (续)

寄存器名称	地址(大端)	地址(小端)	访问单位	读/写	功能
SD 接口					
SDICON	0x5A000000			R/W	SDI 控制
SDIPRE	0x5A000004			R/W	SDI 波特率预标定器
SDICARG	0x5A000008			R/W	SDI 指令参数
SDICCON	0x5A00000C			R/W	SDI 指令控制
SDICSTA	0x5A000010			R/C	SDI 指令状态
SDIRSP0	0x5A000014			R	SDI 回复
SDIRSP1	0x5A000018			R	SDI 回复
SDIRSP2	0x5A00001C			R	SDI 回复
SDIRSP3	0x5A000020			R	SDI 回复
SDIDTIMER	0x5A000024			R/W	SDI 数据/忙定时器
SDIBSIZE	0x5A000028			R/W	SDI 块大小
SDIDCON	0x5A00002C			R/W	SDI 数据控制
SDIDCNT	0x5A000030			R	SDI 数据逗留控制
SDIDSTA	0x5A000034				SDI 数据状态
SDIFSTA	0x5A000038			R	SDI FIFO 状态
SDIIMSK	0x5A00003C				SDI 中断屏蔽
SDIDAT	0x5A000043	0x5A000040	B	R/W	SDI 数据
AC'97 音频 CODEC 接口					
AC_GLBCTRL	0x5B000000			R/W	AC'97 整体控制寄存器
AC_GLBSTAT	0x5B000004			R	AC'97 整体状态寄存器
AC_CODEC_CMD	0x5B000008			R/W	AC'97 编解码指令寄存器
AC_CODEC_STAT	0x5B00000C			R	AC'97 编解码状态寄存器
AC_PCMADDR	0x5B000010			R	AC'97 PCM 输出/入通道 FIFO 寄存器
AC_MICADDR	0x5B000014			R	AC'97 麦克风输入通道 FIFO 地址寄存器
AC_PCMDATA	0x5B000018			R/W	AC'97 PCM 输出/入通道 FIFO 数据寄存器
AC_MICDATA	0x5B00001C			R/W	AC'97 麦克风输入通道 FIFO 数据寄存器

S3C2440A 特殊寄存器注意事项:

1. 小端模式下必须使用小端地址，大端模式下必须使用大端地址。
2. 使用推荐的访问单位访问特殊寄存器。
3. 大/小端模式下除 ADC、RTC 和 UART 寄存器外的所有寄存器都必须用以字（32 位）为单位进行读/写。
4. 请确保 ADC、RTC 和 UART 寄存器使用规定的访问单位和地址进行读/写。此外还必须考虑使用的端模式。
5. W : 32 位寄存器，必须使用 LDR/STR 或 int 指针类型 (int *) 访问。
HW : 16 位寄存器，必须使用 LDRH/STRH 或 short int 指针类型 (short int *) 访问。
B : 8 位寄存器，必须使用 LDRB/STRB 或 char 指针类型 (char *) 访问。

2 程序员模型

概述

S3C2440A 开发采用了由 **ARM**(Advanced RISC Machines)公司研发的先进的 ARM920T 核心。

处理器运行状态

从程序员的角度看，ARM920T 处于以下两种状态之一：

- **ARM 状态**：执行 32 位以字对齐的 ARM 指令。
- **Thumb 状态**：执行 16 位以半字对齐的 Thumb 指令。在此状态下，程序计数器 (PC) 使用位 1 来切换半字。

注释：

两种状态的切换并不影响处理器模式或寄存器内容。

状态切换

进入 Thumb 状态

执行一个 BX 指令可以实现进入到 Thumb 状态，操作数寄存器设置状态位(位[0])

如果处理器在 Thumb 状态进入发生异常 (如 IRQ、FIQ、UNDEF、ABORT、SWI 等)，异常处理返回时也将自动切换回 Thumb 状态。(异常都是在 ARM 状态中执行)

进入 ARM 状态

ARM 状态的进入可以通过下列方法：

- 执行 BX 指令，并且操作数寄存器清除状态位。
- 处理器发生异常 (如 IRQ, FIQ, RESET, UNDEF, ABORT, SWI 等)。此情况下，将程序计数器的内容复制到异常模式的链接寄存器中，并且异常处理将从异常向量地址开始。

存储格式

ARM920T 将存储器视为一个从 0 开始线性递增的字节集合。字节 0 到 3 保存第一个储存字，字节 4 到 7 保存第二个储存字，后面以此类推。ARM920T 能将存储格式为大端或小端的存储器按字处理。

大端 (*Big-Endian*) 格式

在大端格式中，字中最高有效位 MSB(*Most Significant Byte*) 存储在编号最低的字节中，最低有效位 LSB(*Least Significant Byte*) 存储在编号最高的字节中。因此存储器系统的字节 0 关联到数据行的 24 到 31。

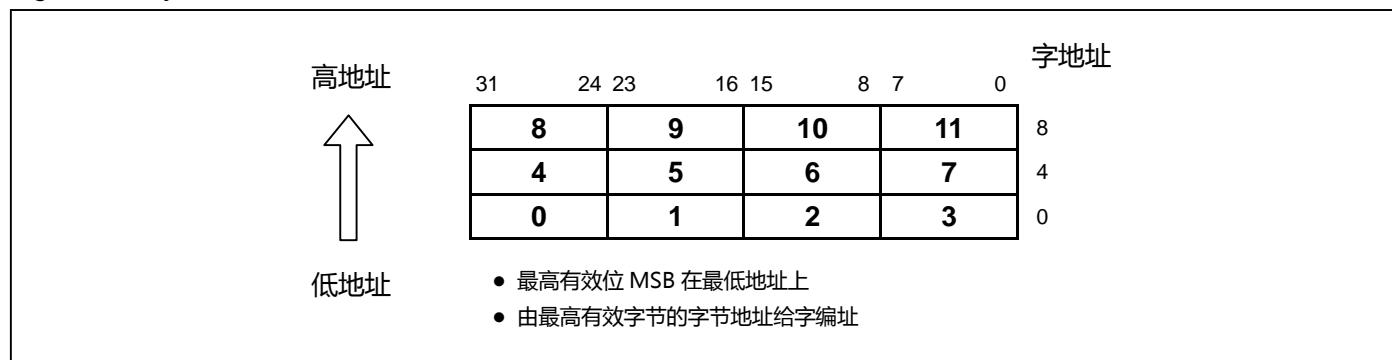


图 2-1. 字内字节的大端地址

小端 (*Little-Endian*) 格式

在小端格式中，字中编号最低的字节被认为是最低有效位 LSB，编号最高的字节为最高有效位 MSB。因此存储器系统的字节 0 关联到数据行的 0 到 7。

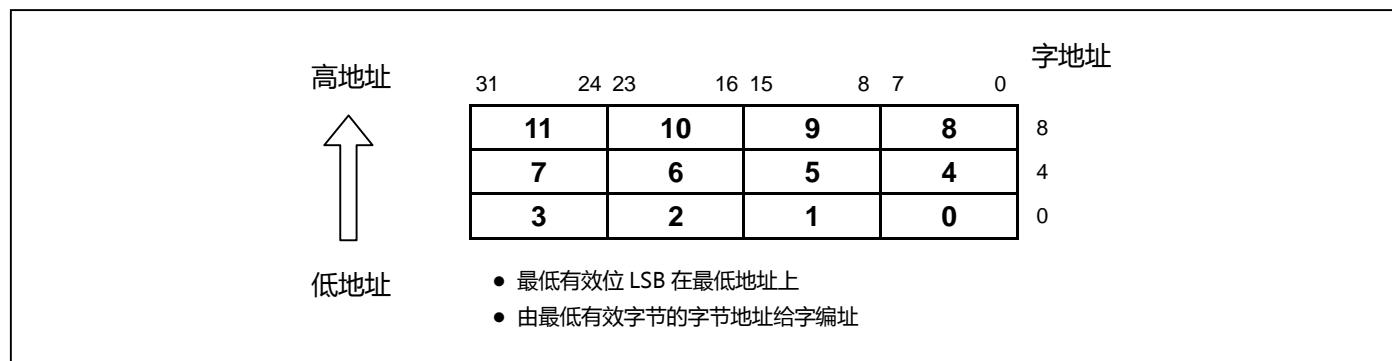


图 2-2. 字内字节的小端地址

指令长度

指令可以为 32 位长 (ARM 状态) 或 16 位长 (Thumb 状态)。

数据类型

ARM920T 支持字节 (8 位)、半字 (16 位) 和字 (32 位) 的数据类型。字必须按 4 字节对齐边界，半字必须按 2 字节对齐边界。

运行模式

ARM920T 支持 7 种运行模式：

- **用户 (usr)**: 正常 ARM 程序执行状态
- **快中断 (fiq)**: 为支持数据传输或通道处理设计
- **中断 (irq)**: 用于一般用途的中断处理
- **管理 (svc)**: 操作系统保护模式
- **中止 (abt)**: 数据或指令预取中止后进入
- **系统 (sys)**: 操作系统的特权用户模式
- **未定义 (und)**: 执行了一个未定义指令时进入

模式的改变可由软件控制，或者由外部中断或进入异常引起。大部分应用程序都将在用户模式执行。被称为特权模式的非用户模式，都将进入到中断服务或异常中去，或者访问受保护的资源。

内部寄存器

ARM920T 总共有 37 个寄存器，其中 31 通用 32 位寄存器和 6 个状态寄存器，但不能在同一时刻对所有的寄存器可见。处理器状态和运行模式决定了哪些寄存器对程序员可见。

ARM 状态时内部寄存器集

在 ARM 状态，16 个通用寄存器和一个状态寄存器在任意时刻都可见。在特权（非用户）模式下，将切换到指定模式的分组 (banked) 寄存器。图 2-3 显示了哪些寄存器在各模式下是可见的：分组寄存器被标记了阴影三角形。

ARM 状态时寄存器被设为包含 16 个直接可以访问的寄存器：R0 到 R15。除了 R15，其他全部寄存器都为通用寄存器，如可能用于保存数据或地址值。除此之外，还有第 17 个寄存器用于存储状态信息。

寄存器 14	此寄存器被用作子程序的链接寄存器。当执行分支和链接 (BL) 指令时该寄存器接收 R15 的备份。其余时间也可作为通用寄存器处理。相对应的分组寄存器 R14_svc , R14_irq , R14_fiq , R14_abt 和 R14_und 也类似用于中断或异常发生时 R15 的返回值，或者在中断或异常例程中执行了分支和链接中断。
寄存器 15	此寄存器存放着程序计数器(PC)。在 ARM 状态 ,R15 的位[1:0]都为 0 ,位[31:2]为 PC 值。在 Thumb 状态，只有位[0]为 0 ，位[31:1]为 PC 值。
寄存器 16	此寄存器为 CPSR (当前程序状态寄存器)。它包含条件码标志位和当前模式位。

FIQ 模式包含 7 个分组寄存器，分别映射到 R8-14 (R8_fiq-R14_fiq)。在 ARM 状态，有很多不需要保存寄存器的 FIQ 处理程序。用户，IRQ，管理中止和未定义模式都包含两个分组寄存器映射到 R13 和 R14，允许这些模式都包含私有堆栈指针和链接寄存器。



图 2-3. ARM 状态时寄存器组织

Thumb 状态时内部寄存器集

Thumb 状态时的寄存器集为 ARM 状态时的寄存器的分配的一个子集。程序员能够直接访问 8 个通用寄存器 R0 至 R7，还有程序计数器 (PC)，一个堆栈指针寄存器 (SP)，一个链接寄存器 (LR) 和 CPSR。各自特权模式还分别有分组堆栈指针，链接寄存器和进程保存状态寄存器(SPSR)。如图 2-4 所示。

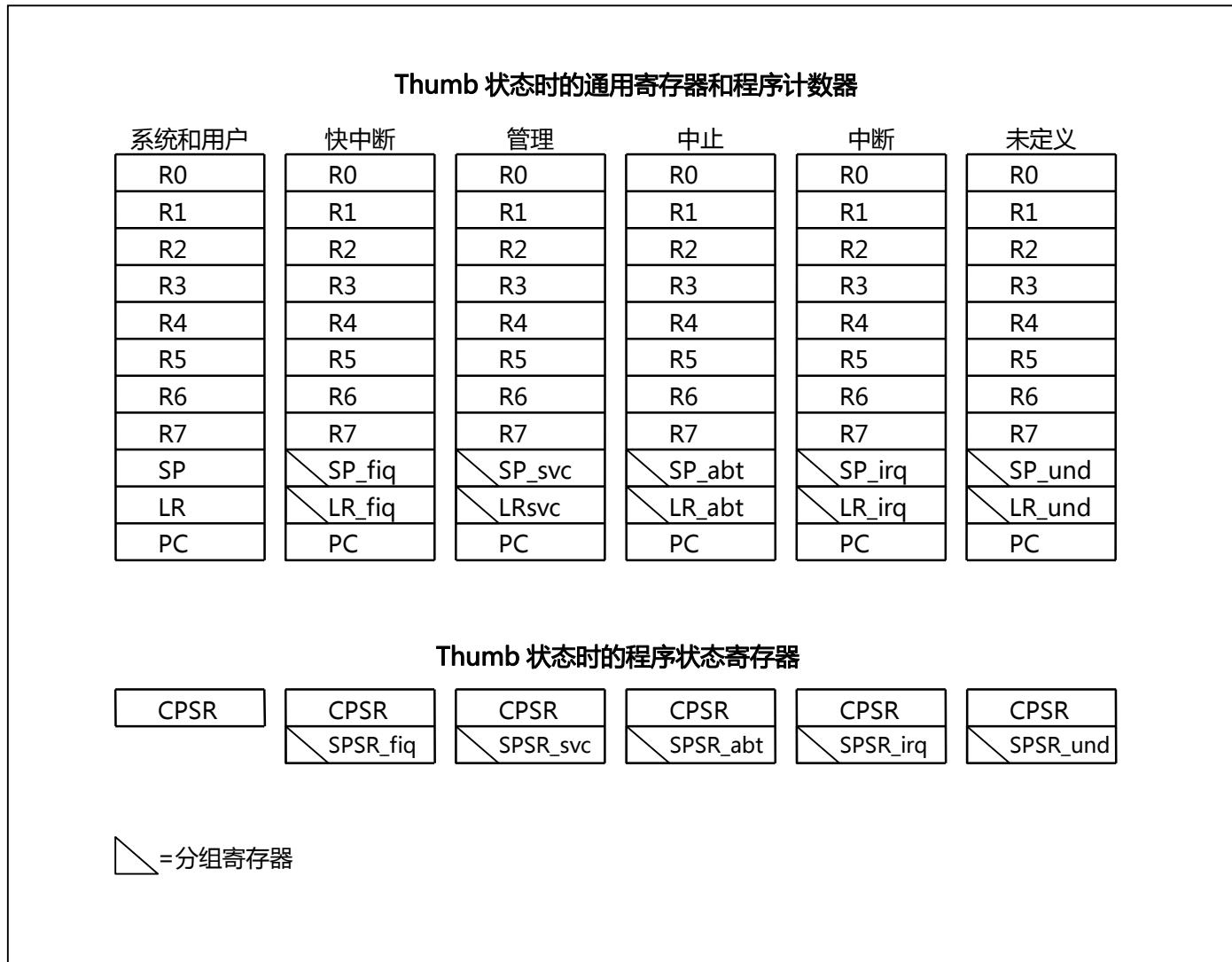


图 2-4. Thumb 状态时寄存器组织

ARM 状态的寄存器与 Thumb 状态的寄存器之间的关系

ARM 与 Thumb 状态寄存器之间有如下关系：

- Thumb 状态的 R0-R7 与 ARM 状态的 R0-R7 相同
- Thumb 状态的 CPSR 和 SPSR 与 ARM 状态的 CPSR 和 SPSR 相同
- Thumb 状态的 SP 映射到 ARM 状态的 R13
- Thumb 状态的 LR 映射到 ARM 状态的 R14
- Thumb 状态的程序计数器映射到 ARM 状态的程序计数器 (R15)

这些关系如图 2-5 所示。

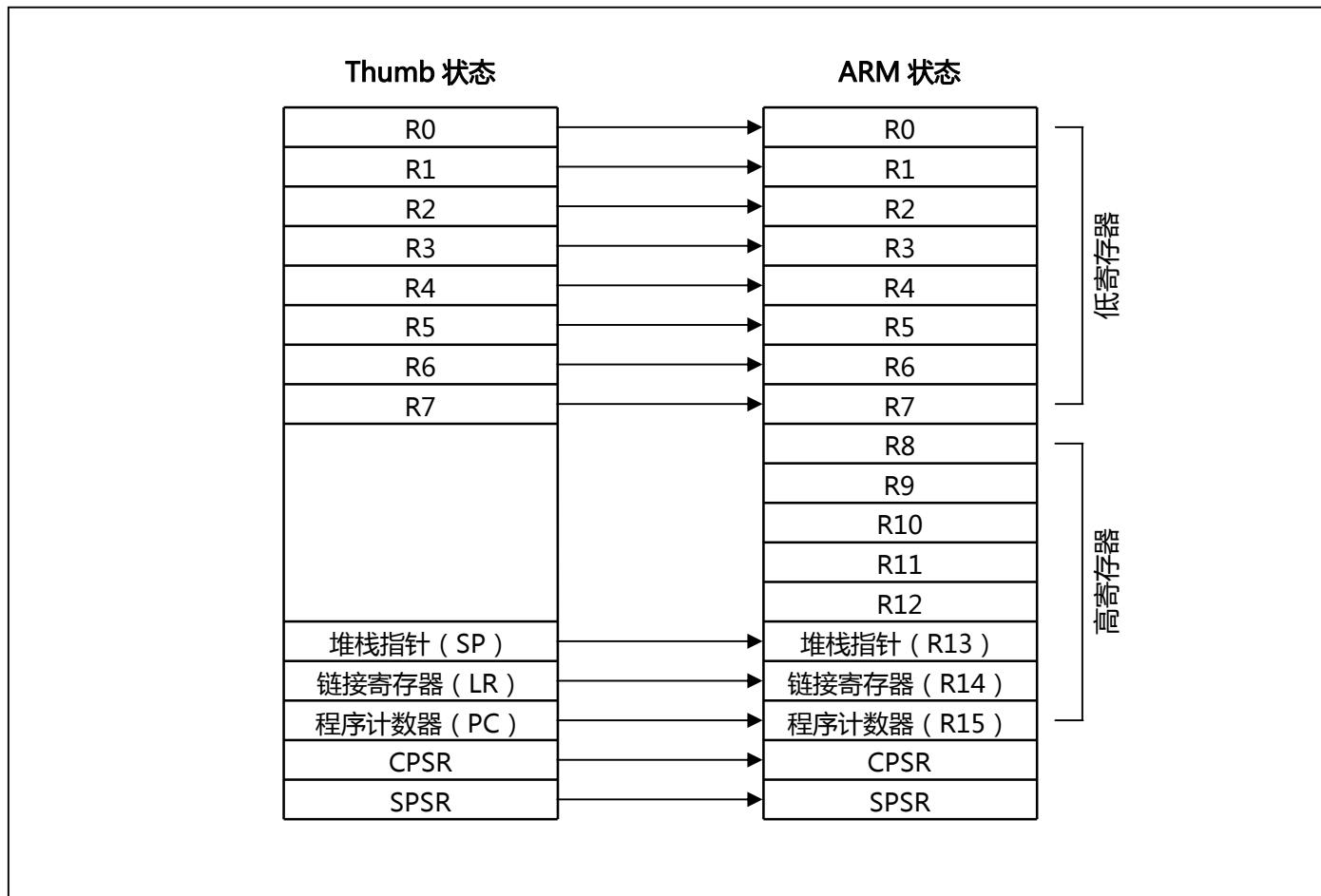


图 2-5. Thumb 状态的寄存器在 ARM 状态的寄存器上的映射

Thumb 状态访问高寄存器 (*Hi-Registers*)

在 Thumb 状态，寄存器 R8-R15 (“*Hi registers*”) 不是标准的寄存器集的一部分。然而，汇编语言程序员可以受限的对它们进行访问，可以将它们用于快速暂存。

使用 MOV 指令的特殊变量可以将一个值从 R0-R7 (“*Lo register*”) 范围内的寄存器传送到高寄存器或从高寄存器到第低寄存器。使用 CMP 和 ADD 指令也可以对高寄存器的值与寄存器的值进行比较以或相加。更多内容请参考表 3-34。

程序状态寄存器

ARM920T 包含了一个当前程序状态寄存器 (*Current Program Status Register-CPSR*) , 另外还有 5 个用于异常程序处理的程序状态保存寄存器 (*Saved Program Status Registers-SPSR*) 。这些寄存器的功能为 :

- 保存最近已处理的 ALU 操作的信息
- 控制中断的使能与禁止
- 设置处理器的运行模式

图 2-6 显示了各位的编排

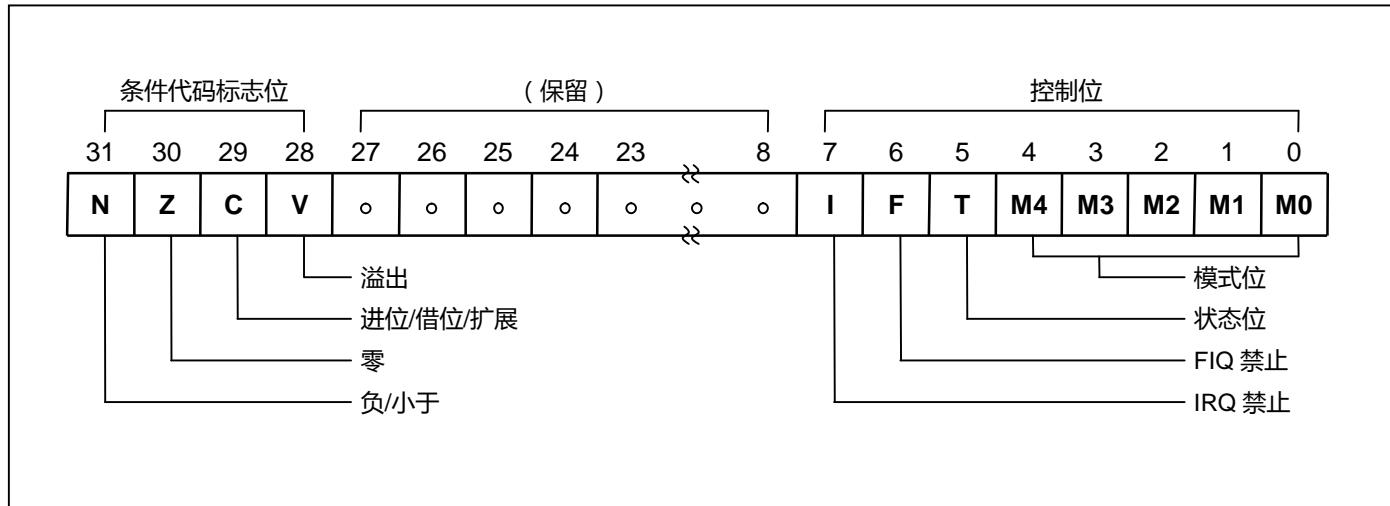


图 2-6. 程序状态寄存器格式

状态代码标志位

N , Z , C 和 V 位为状态代码标志位。算术或逻辑操作结果可能会改变这些位，并检验以决定是否应该执行某条指令。

在 ARM 状态，所有指令都可能为条件执行：详情见表 3-2。

在 Thumb 状态，只有分支指令才能条件执行：详情见表 3-46。

控制位

PSR (包含 I , F , T 和 M[4:0])的末端 8 位统称为控制位。当发生异常时将会改变这些位。如果处理器运行在特权模式，它们也可由软件控制。

T 位	该位反映了运行着的状态。当此位被置位，处理器的执行在 Thumb 状态下，否则在 ARM 状态。此位映射到 TBIT 外部信号端。 注意： 决不要试图用软件改变 CPSR 内的 TBIT 状态。如果发生这种情况，处理器将进入不可预测的状态！
中断禁止位	I 和 F 位为中断禁止位。如果置位，将分别禁止 IRQ 和 FIQ 中断。
模式位	M4 , M3 , M2 , M1 和 M0 位 (M[4:0]) 为模式位。这些位决定了处理器的运行模式，如表 2-1 所示。并不是所有的模式位组合都定义了一个有效的处理器模式。只有那些有明确描述的组合才能使用。用户必须注意是否有任何不合法的值被编程到模式位 M[4:0]，否则处理器将进入不可预测的状态！如果出现了，使用复位。
保留位	PSR 剩余的位保留。当改变一个 PSR 的标志位或控制位时，你必须保证没有改变这些未使用的位。同样你的程序不依赖于包含特定值的保留位，因为将来的处理器也许将它们设置为 1 或 0。

表 2-1. PSR 模式位值

M[4:0]	模式	可见的 Thumb 状态寄存器	可见的 ARM 状态寄存器
10000	用户	R7-R0, LR, SP PC, CPSR	R14-R0, PC, CPSR
10001	快中断	R7-R0, LR_fiq, SP_fiq PC, CPSR, SPSR_fiq	R7-R0, R14_fiq-R8_fiq, PC, CPSR, SPSR_fiq
10010	中断	R7-R0, LR_irq, SP_irq PC, CPSR, SPSR_irq	R12-R0, R14_irq, R13_irq, PC, CPSR, SPSR_irq
10011	管理	R7-R0, LR_svc, SP_svc, PC, CPSR, SPSR_svc	R12-R0, R14_svc, R13_svc, PC, CPSR, SPSR_svc
10111	中止	R7-R0, LR_abt, SP_abt, PC, CPSR, SPSR_abt	R12-R0, R14_abt, R13_abt, PC, CPSR, SPSR_abt
11011	未定义	R7-R0, LR_und, SP_und, PC, CPSR, SPSR_und	R12-R0, R14_und, R13_und, PC, CPSR
11111	系统	R7-R0, LR, SP PC, CPSR	R14-R0, PC, CPSR

异常

当正常程序流程被暂时停止发生异常，例如响应一个来自外设的中断。在处理异常前，必须保护当前的处理器状态，以便在完成处理程序后能恢复到原来的程序。

如果同时发生好几个异常，将会按照固定的顺序来分配，见 P2-13 的异常优先级。

进入异常行为

当处理一个异常时，ARM920T 将会进行以下活动：

1. 相应链接寄存器保存下条指令的地址。如果在 ARM 状态进入异常，下条指令的地址将会复制到链接寄存器（当前 PC+4 或 PC+8，由异常决定。详情见表 2-2）中。如果在 Thumb 状态进入异常，写入链接寄存器的值则为当前 PC 偏移一个值，这样异常返回后程序能从正确的位置恢复。这意味着异常处理不需要确定异常是从什么状态进入的。例如，在 SWI 的情况，无论是在 ARM 状态还是 Thumb 状态执行 SWI，**MOVS PC, R14_svc** 语句都将返回到下一条指令。
2. 复制 CPSR 的内容到相应 SPSR 中。
3. 根据异常类型强制将 CPSR 模式位设为某一个值。
4. 强制 PC 从相关异常向量处取下条指令。

通常也会置位中断禁止标志位，以防止不同的难处理的异常嵌套。

如果一个异常发生时处理器处于 Thumb 状态，当装载异常向量地址到 PC 时会自动切换到 ARM 状态。

离开异常行为

当异常结束，异常处理程序将会：

1. 将链接寄存器适当减去一个偏移量并放入到 PC 中。（偏移量由异常类型决定）
2. 复制 SPSR 的内容返回给 CPSR 中。
3. 如果在异常进入时置位了中断禁止标志位异常，清除中断禁止标志位。

注意：不需要在异常结束时切换回 Thumb 状态，因为在异常前会立刻保存 CPSR 中 T 位的值到 SPSR 中，并在退出异常时从 SPSR 恢复到 CPSR 中。

异常进入/退出总结

表 2-2 总结了进入异常时保存在相关 R14 中的 PC 值和被推荐的退出异常的指令。

表 2-2. 异常进入/退出

异常源	返回指令	先前状态		注释
		ARM R14_x	THUMB R14_x	
带链接分支指令 BL	MOV PC, R14	PC + 4	PC + 2	1
软件中断 SWI	MOVS PC, R14_svc	PC + 4	PC + 2	1
位定义指令 UDEF	MOVS PC, R14_und	PC + 4	PC + 2	1
快中断 FIQ	SUBS PC, R14_fiq, #4	PC + 4	PC + 4	2
中断 IRQ	SUBS PC, R14_irq, #4	PC + 4	PC + 4	2
预取中止 PABT	SUBS PC, R14_abt, #4	PC + 4	PC + 4	1
数据中止 DABT	SUBS PC, R14_abt, #8	PC + 8	PC + 8	3
复位 RESET	无	-	-	4

注释：

1. 此处 PC 为含预取中止的 BL/SWI/未定义指令取指令的地址。
2. 此处 PC 为由于 FIQ 或 IRQ 抢先而未执行的指令地址。
3. 此处 PC 为发生了数据中止的 Load 或 Store 指令的地址。
4. 当复位时保存在 R14_svc 的值将是不可预测的。

快中断 FIQ

FIQ (快中断请求) 异常是为支持数据传输或通道处理而设计的 , 在 ARM 状态拥有足够的私有寄存器来消除对寄存器保存的需求 (这样最小化了对上下文的切换的开销) 。

将 nFIQ 输入端拉低可以实现外部产生 FIQ 。由 ISYNC 输入信号端的状态决定是同步还是异步传输。当 ISYNC 为低电平 , 认为 nFIQ 和 nIRQ 为异步 , 中断前会引起同步周期延迟并影响处理器流程。

无论是从 ARM 还是 Thumb 状态进入的异常 , FIQ 处理时执行

SUBS PC,R14_fiq,#4

时 , 都应该避免中断。

如果设置 CPSR 的 F 标志位 , FIQ 将会被禁止 (但主义这不可能在用户模式中发生) 。如果 F 标志位为零 , ARM920T 将在每条指令末检测 FIQ 同步发生器的输出是否为低电平。

中断 IRQ

IRQ (中断请求) 异常是一个由 nIRQ 输入端的低电平产生的一个普通中断。 IRQ 的优先级低于 FIQ , 当进入了相关的 FIQ , 会屏蔽 IRQ 。除非是在特权(非用户)模式 , 其他任何时刻都禁止设置 CPSR 内的 I 位。

无论是从 ARM 还是 Thumb 状态进入的异常 , IRQ 处理时执行

SUBS PC,R14_fiq,#4

时 , 都应该避免中断。

中止

中止表示不能完成当前对存储器的访问。通过外部 ABORT 输入端指示的。 ARM920T 在存储器访问周期期间检测中止异常。

有两种类型的中止 :

- 预取中止 (*Prefetch Abort*) : 发生在指令预取期间
- 数据中止 (*Data Abort*) : 发生在数据访问期间

如果发生预取中止 , 将屏蔽预取指并为无效 , 但并不会立刻带来异常 , 直到指令到达流水线的执行阶段才发生。若未执行该指令 , 将不会发生中止 , 因为流水线发生了分支。

如果发生数据中止 , 由指令类型决定其行为 :

- 单一的数据转移指令 (LDR , STR) 回写到被修改的基址寄存器 : 中止处理程序必须意识到这点。
- 交换指令 (SWP) 执行失败 , 就如同没有被执行。
- 块数据转移指令 (LDM , STM) 完成。如果设置了回写 , 基址寄存器将被更新。如果指令会覆盖基址寄存器数据 (转移列表中包含基址) , 覆盖将会被阻止。表明了中止后所有寄存器的覆盖都会被阻止 , 特别是 R15 (通常是最最后转移的寄存器) 在一个被中止的 LDM 指令会被阻止覆盖。

中止机制使得分页虚拟存储器系统可以被实现。在这样一个系统中允许处理器产生任意地址。当无法获取某一地址上的数据时 , 内存管理单元 (*Memory Management Unit-MMU*) 将表明产生一个中止。中止处理程序必须紧接着找出中止原因 , 使得被请求的数据可用并重试被中止的指令。应用程序并不需要了解可以使用的内存总量 , 也不需要关心中止对其状态以任何方式被影响。

当确定中止原因后 , 处理程序应该不顾状态 (ARM 或 Thumb) 执行下列语句 :

SUBS PC,R14_abt,#4 ; 预取中止

SUBS PC,R14_abt,#8 ; 数据中止

这将恢复 PC 和 CPSR , 并重试被中止的指令。

软件中断 SWI

软件中断指令 (*Software Interrupt Instruction-SWI*) 用于进入管理模式，通常请求一个特定的管理功能。SWI 处理程序应当在状态 (ARM 或 Thumb) 中通过执行下列指令返回：

MOV PC,R14_svc

这将恢复 PC 和 CPSR，并返回到 SWI 之后的指令。

注意：nFIQ, nIRQ, ISYNC, LOCK, BIGEND 和 ABORT 引脚只存在于 ARM920T CPU 内核之中。

未定义指令

当 ARM920T 遇到不能处理的指令时，将产生未定义指令陷阱。这个机制可以用于软件仿真扩展 Thumb 指令集或 ARM 指令集。

仿真了失败指令后，陷阱处理程序应该执行在状态 (ARM 或 Thumb) 中执行下列指令：

MOVS PC,R14_und

这将恢复 CPSR，并返回到未定义指令之后的指令。

异常向量

下表显示了异常向量地址。

表 2-3. 异常向量

地址	异常	进入模式
0x00000000	复位	管理模式
0x00000004	未定义指令	未定义模式
0x00000008	软件中断	管理模式
0x0000000C	中止 (预取)	中止模式
0x00000010	中止 (数据)	中止模式
0x00000014	保留	保留
0x00000018	中断 IRQ	中断模式
0x0000001C	快中断 FIQ	快中断模式

异常优先级

当同时出现多个异常时，一个固定的优先级系统将确定它们的处理顺序：

高优先级：

1. 复位
2. 数据中止
3. 快中断 FIQ
4. 中断 IRQ
5. 预取中止

低优先级：

6. 未定义指令，软件中断

并不是所有的异常都可以在同一时刻发生！

未定义指令和软件中断异常互斥，因为它们分别对应于当前指令的特定(非重叠)译码。

如果发生数据中止的同时发生 FIQ 并且又使能了 FIQ (CPSR 的 F 标志位被清零)，ARM920T 将先进入数据中止处理程序，然后立即转向 FIQ 向量。当从 FIQ 正常返回将会引起数据中止处理程序恢复执行。为了确保传输错误不会在检测中被遗漏，必须将数据中止置为比 FIQ 更高的优先级。进入此异常的时间需要按加上最坏情况下 FIQ 的时间延迟来算。

中断延迟

当使能了 FIQ，其最坏情况的延迟包括：请求通过同步器的最长时间（**Tsyncmax**，如果为异步），加上完成最长指令的时间(**Tldm**，最长指令是装载包括 PC 在内所有的寄存器的 LDM 指令)，加上数据中止进入的时间(**Texc**)，再加上 FIQ 进入的时间 (**Tfiq**)。在该时间末 ARM920T 将会执行位于 0x1C 处的指令。

Tsyncmax 为 3 个才处理器周期，Tldm 为 20 个周期，Texc 为 3 个周期，Tfiq 为 2 个周期。因此总时间为 28 个处理器周期。在一个使用连续 20MHz 处理器时钟的系统中，其只过了 1.4 毫秒。最大 IRQ 延迟的计算与此类同，但还必须实际考虑，FIQ 具有更高的优先级并且能推迟任意长度的时间进入到 IRQ 处理程序。FIQ 或 IRQ 最小延迟包括请求通过同步器的最短时间 (**Tsyncmin**) 加上 Tfiq。此时为 4 个处理器周期。

复位

当 nRESET 信号端变为低电平时，ARM920T 将放弃正在处理的指令，然后从递增 (*incrementing*) 字地址继续取指令。

当 nRESET 信号端变为高电平时，ARM920T 将：

1. 复制当前 PC 和 CPSR 的值覆盖到 R14_svc 和 SPSR_svc 中。未定义保存的 PC 和 SPSR 的值
2. 强制将 M[4:0]设置为 10011 (管理模式)，置位 CPSR 中的 I 和 F 位并清除 CPSR 的 T 位。
3. 强制 PC 从 0x00 地址开始对下一条指令的取指。
4. 在 ARM 状态恢复执行。

3 ARM 指令集

指令集概述

此章描述了 ARM920T 核心的 ARM 指令集。

格式概述

ARM 指令集如下图所示。

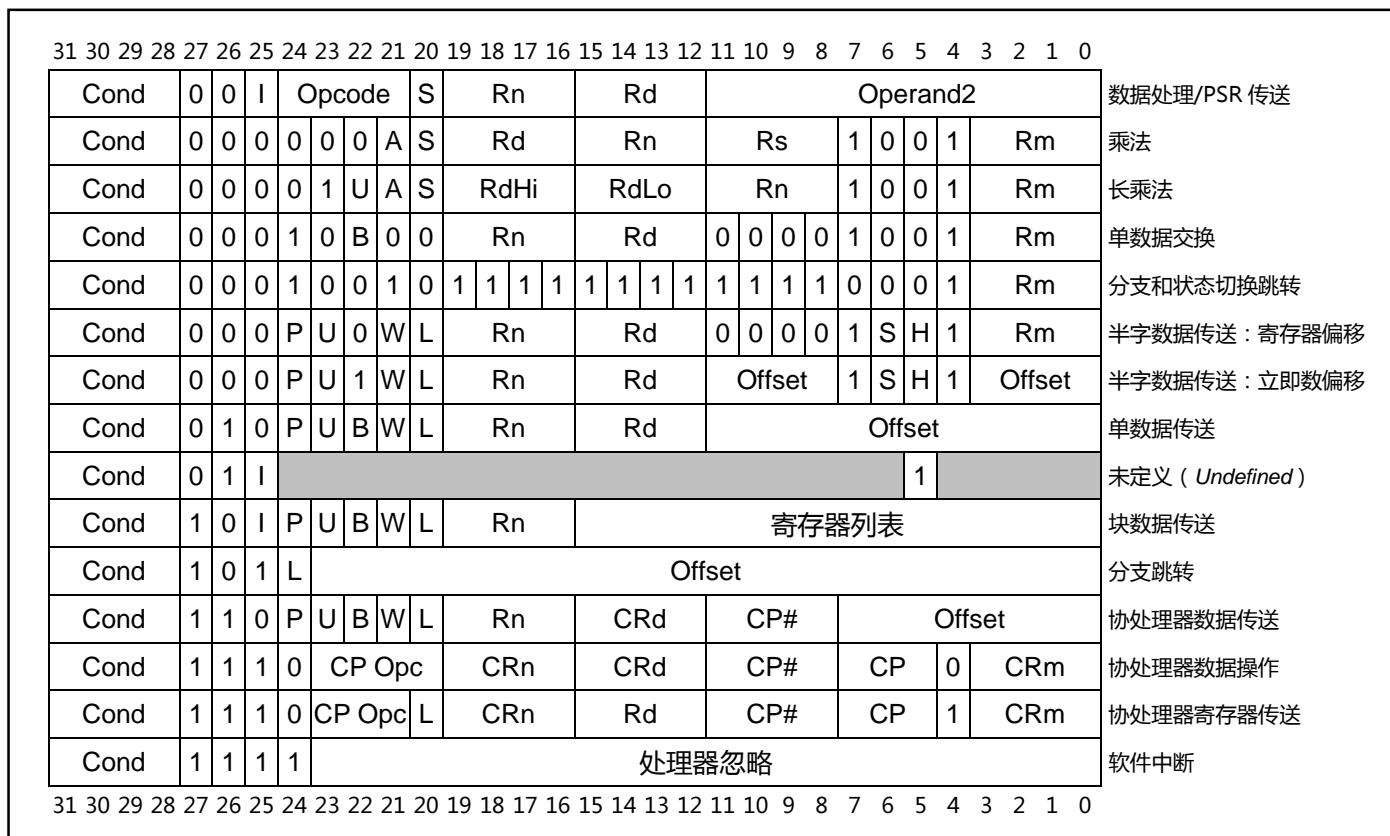


图 3-1. ARM 指令集格式

注意：

一些指令码未被定义，但并不一定会导致带来未定义指令陷阱 (*Undefined Instruction Trap*)，例如一个乘法指令的位[6]变为了 1。不要使用这些没有被定义的指令，在未来的 ARM 实施中可能会改变其功能。

指令概述

表 3-1. ARM 指令集

助记符	说明	操作
ADC	带进位加法	$Rd := Rn + Op2 + \text{进位}$
ADD	加法	$Rd := Rn + Op2$
AND	逻辑与	$Rd := Rn \text{ 逻辑与 } Op2$
B	分支跳转	$R15 := \text{地址}$
BIC	位清零	$Rd := Rn \text{ 逻辑与非 } Op2$
BL	带连接分支跳转	$R14 := R15, R15 := \text{地址}$
BX	分支和状态切换跳转	$R15 := Rn, T \text{ 位} := Rn[0]$
CDP	协处理器数据处理	(指定协处理器)
CMN	负数比较	CPSR 状态标志位 $:= Rn + Op2$
CMP	比较	CPSR 状态标志位 $:= Rn - Op2$
EOR	逻辑异或	$Rd := (Rn \text{ 逻辑与非 } Op2) \text{ 逻辑或 } (Op2 \text{ 逻辑与非 } Rn)$
LDC	加载字数据，从存储器到协处理器	协处理器加载
LDM	加载字数据，从存储器到多寄存器	栈操作 (Pop)
LDR	加载字数据，从存储器到寄存器	$Rd := (\text{地址})$
MCR	移动字数据，从 CPU 寄存器到协处理器寄存器	$cRn := rRn \{<\text{op}>cRm\}$
MLA	乘加	$Rd := (Rm \times Rs) + Rn$
MOV	移动寄存器中数据或常量	$Rd := Op2$
MRC	移动字器数，从协处理器寄存器到 CPU 寄存器	$Rn := cRn \{<\text{op}>cRm\}$
MRS	移动状态/标志位，从 PSR 到寄存器	$Rn := PSR$
MSR	移动状态/标志位，从寄存器数据到 PSR	$PSR := Rm$
MUL	乘法	$Rd := Rm \times Rs$
MVN	移动寄存器取反后的数据	$Rd := 0xFFFFFFFF \text{ 逻辑异或 } Op2$
ORR	逻辑或	$Rd := Rn \text{ 逻辑或 } Op2$
RSB	逆向减法	$Rd := Op2 - Rn$
RSC	带进位逆向减法	$Rd := Op2 - Rn - 1 + \text{进位}$
SBC	带进位减法	$Rd := Rn - Op2 - 1 + \text{进位}$
STC	存储字数据，从协处理器寄存器到存储器	地址 $:= CRn$
STM	多字数据存储	栈操作(Push)
STR	存储字数据，寄存器到存储器	<地址 $:= Rd$
SUB	减法	$Rd := Rn - Op2$
SWI	软件中断	操作系统调用
SWP	寄存器与存储器内容交换	$Rd := [Rn], [Rn] := Rm$
TEQ	按位测试相等	CPSR 标志位 $:= Rn \text{ 逻辑异或 } Op2$
TST	位测试	CPSR 标志位 $:= Rn \text{ 逻辑与 } Op2$

条件字段

在 ARM 状态，所有的指令都可以按照 CPSR 状态码和指令条件字段的状态来有条件地执行。此字段（位[31:28]）确定了在什么情况下哪一个指令被执行。如果 C, N, Z 和 V 标志位的状态符合字段的条件码，将执行指令，否则忽略不执行。

有 16 种可能的条件，每种表示为在指令助记符后附加两个字符后缀。例如，一个分支（汇编语言中的 B）跳转指令变成 BEQ 为“如果相等则分支跳转”，这意味着只有 Z 标志位被置位了才会执行分支跳转。

在实际应用当中，将会使用到 15 种不同的条件：如表 3-2 所列，保留第 16 种（1111），并且一定不要使用。

表 3-2. 条件码概述

码	助记符	标志位	含义
0000	EQ	Z 置位	相等
0001	NE	Z 清零	不相等
0010	CS	C 置位	无符号大于或等于
0011	CC	C 清零	无符号小雨
0100	MI	N 置位	负数
0101	PL	N 清零	正数或零
0110	VS	V 置位	溢出
0111	VC	V 清零	未溢出
1000	HI	C 置位并且 Z 清零	无符号大于
1001	LS	C 清零或 Z 置位	无符号小于或等于
1010	GE	N 等于 V	大于或等于
1011	LT	N 不等于 V	小于
1100	GT	Z 清零和 (N 等于 V) 逻辑与	大于
1101	LE	Z 置位和 (N 不等于 V) 逻辑或	小于或等于
1110	AL	(忽略)	始终

分支和状态切换跳转指令 (BX)

这条指令只在条件为真时执行。各种条件在表 3-2 中定义了。

这条指令执行了一个将一个普通寄存器 Rn 中的内容复制到程序计数器 PC 中的分支跳转。这个分支跳转导致清空流水线，并且由 Rn 指定的地址处重新填充流水线。这条指令同样的允许交换指令集。当执行该指令，Rn[0] 的值将决定是按 ARM 指令还是按 Thumb 指令来译码指令流。

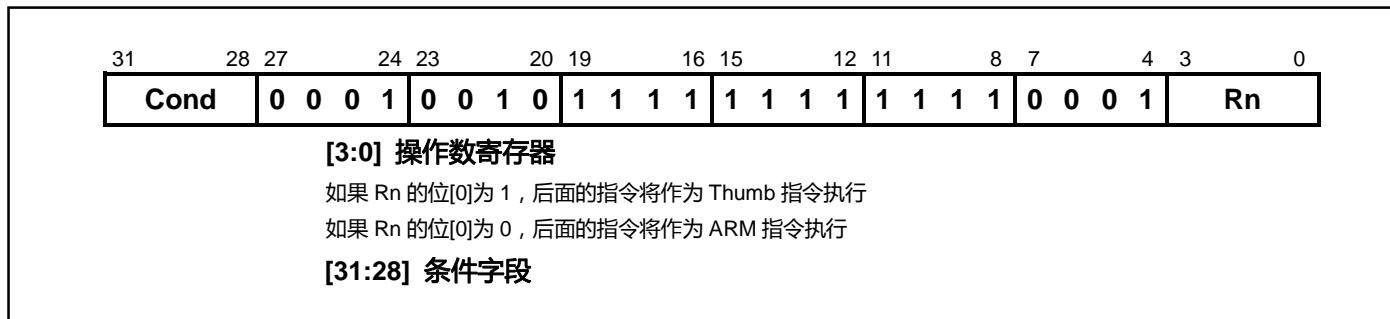


图 3-2. 分支和状态切换指令

指令周期时间

执行 BX 指令花费 $2S+1N$ 个周期，其中 S 和 N 分别是定义连续 (*sequential*) 周期 (S-cycle) 和非连续 (*non-sequential*) 周期 (N-cycle)。

汇编语法

BX – 分支和状态切换

BX{cond} Rn

{ cond } 两个字符条件助记符。见表 3-2

Rn 有效寄存器号的表达式

将 R15 当作操作数使用

如果 R15 被当成一个操作数，这种行为并没有被定义。

例子

```

ADR      R0,  Into_THUMB+1      ;引起分支跳转的目标地址，设置位[0]为高，从此时起它将进入 Thumb 状态
BX      R0                      ;分支跳转并且切换到 Thumb 状态
CODE16
Into_THUMB
ADR      R5,  Back_to_ARM      ;引起分支跳转的目标按字对齐地址，从此时起位[0]为低并将切换回 ARM 状态
BX      R5                      ;分支跳转并且切换回 ARM 状态
ALIGN
CODE32
Back_to_ARM

```

分支和带链接分支跳转指令 (B , BL)

这些指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如下图 3-3 所示。



图 3-3. 分支跳转指令

分支跳转指令包含了一个有符号补码的 24 位偏移量。该值左移两位后并将其有符号地扩展到 32 位 (有效位数为 25 位加 1 位符号位，即 $2^{25}=32M$)，再加入到 PC 中。因此该指令可以指定 $\pm 32M$ 字节的分支跳转。分支跳转偏移还必须考虑预取操作，因为 PC 超前于当前指令 2 字节 (8 字节)。

分支跳转如果超过了 $\pm 32M$ 字节，必须使用一个偏移量或绝对目标地址，将其事先装载到某一寄存器中。在这种情况下如果带链接分支跳转类型操作是必须的，PC 应当手动保存到 R14。

链接位

带链接分支跳转 (BL) 指令将原来的 PC 值写入到当前组的链接寄存器 (R14) 中。考虑到预取，还应调整写入到 R14 的 PC 的值，包含分支和链接跳转指令后的指令地址。注意未保存 CPSR，总是清除 PC 和 R14[1:0]。

从常规调用带链接分支转移指令返回，如果链接寄存器仍然有效则使用 MOV PC, R14 语句，或如果链接寄存器被保存到存在 Rn 中的堆栈指针中则使用 LDM Rn!, {...PC} 语句。

指令周期时间

分支跳转和带链接分支跳转指令耗时 $2S+1I$ 个指令周期，其中 S 和 I 分别是定义连续周期 (S-cycle) 和内部周期 (I-cycle)。

汇编语法

在 "{}" 内的项是可选的，在 "<>" 中的项是必须的。

B{L}{cond} <expression>

{L} 用于分支跳转指令请求带链接。如果不存在 L，指令将不会受影响 R14。

{cond} 两个字符条件助记符，如表 3-2 所示。如果不存在将会使用 AL (Always) 替代。

<expression> 目标地址。汇编计算偏移量。

例子

here	BAL	here	;汇编到 0xEFFFFFFE 处(注意 PC 偏移的影响)
	B	there	;总是默认条件执行
	CMP	R1, #0	;R1 和 0 比较，如果 R1 为 0 则分支跳转到 fred，否则继续执行
	BEQ	fred	;继续到下一指令
	BL	sub+ROM	;从计算后地址调用子程序
	ADDS	R1, #1	;R1 寄存器加 1，结果影响 CPSR 标志位
	BLCC	sub	;如果清零了 C 标志位调用子程序，将会如此下去，除非 R1 中存放着 0xFFFFFFFF

数据处理指令

数据处理指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-4 所示。

该指令使用一个或两个操作数执行算术或逻辑运算，并产生一个结果。第一个操作数通常是一个寄存器 (Rn)。

根据指令的 I 位的值，第二个操作数可能是一个移位 (shifted) 寄存器 (Rm)，或者是循环移位的 8 位立即数 (Imm)。按照指令中的 S 位的值，CPSR 中条件码会根据指令结果而可能被保持或更新。

某些操作数 (TST, TEQ, CMP, CMN) 不能将结果写入到 Rd。它们只用于执行测试和在结果中设置条件码，通常还包括设置 S 位。指令和其结果在表 3-3 中列出。

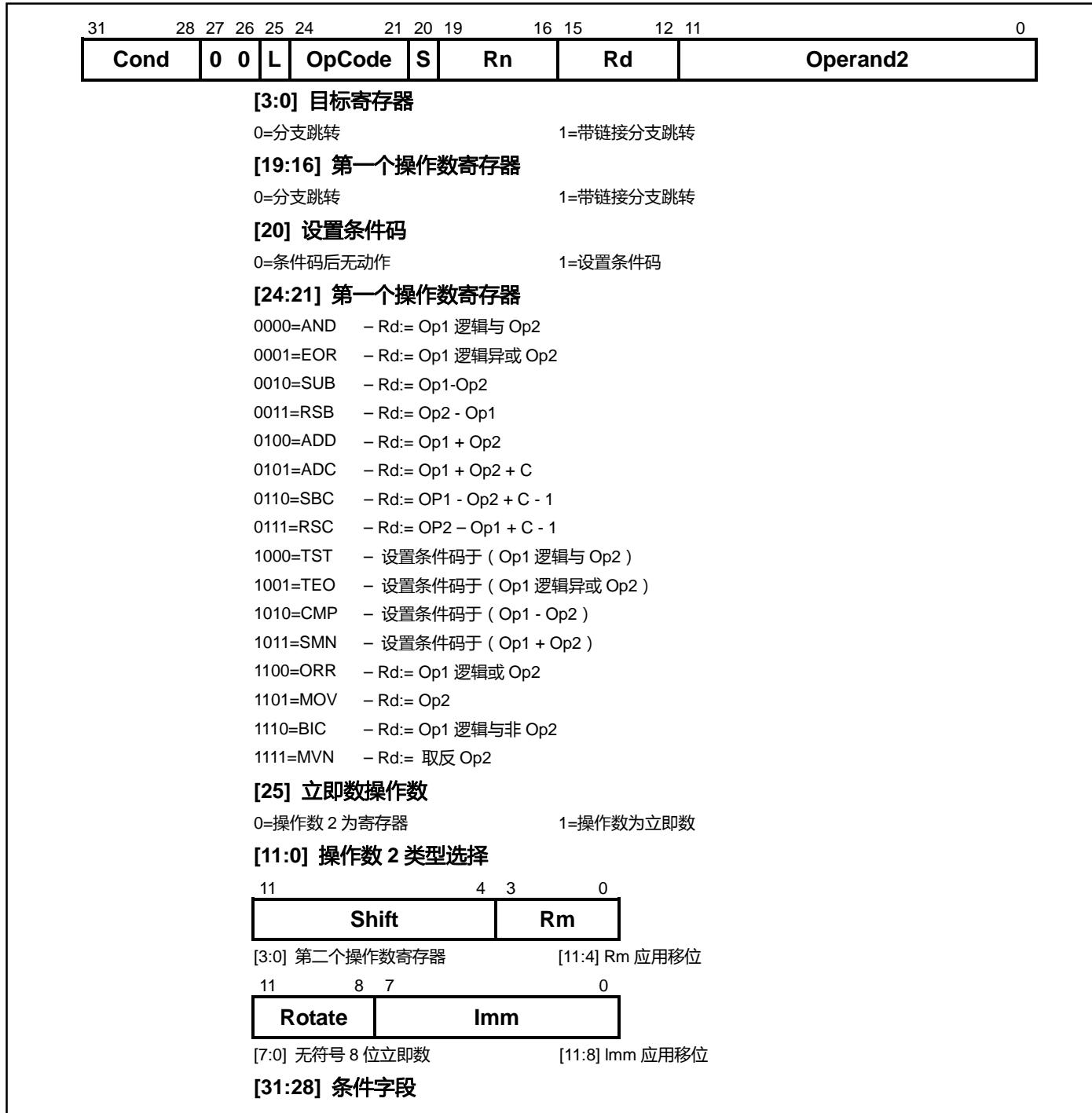


图 3-4. 数据处理指令

CPSR 标志位

数据处理操作可以归类为逻辑或算术。逻辑操作 (AND , EOR , TST , TEQ , ORR , MOV , BIC , MVN) 执行的是对操作数或操作数所产生的结果的全部对应位的逻辑动作。如果置位了 S 位 (并且 Rd 不是 R15 , 如下), 在 CPSR 中 : V 标志位不受影响 , 当筒形移位器完成时将置位 C 标志位 (或当移位操作为 LSL #0 时保持不变), 当且仅当结果全为零时置位 Z 标志位 , 结果的位[31]的逻辑值将置位 N 标志位。

表 3-3. ARM 数据处理指令

汇编助记符	操作码	操作
AND	0000	操作数 1 逻辑与操作数 2
EOR	0001	操作数 1 逻辑异或操作数 2
WUB	0010	操作数 1 - 操作数 2
RSB	0011	操作数 2 - 操作数 1
ADD	0100	操作数 1 + 操作数 2
ADC	0101	操作数 1 + 操作数 2 + 进位
SBC	0110	操作数 1 - 操作数 2 + 进位 - 1
RSC	0111	操作数 2 - 操作数 1 + 进位 - 1
TST	1000	同 AND , 但不记录结果
TEQ	1001	同 EOR , 但不记录结果
CMP	1010	同 SUB , 但不记录结果
CMN	1011	同 ADD , 但不记录结果
ORR	1100	操作数 1 逻辑或操作数 2
MOV	1101	操作数 2 (忽略操作数 1)
BIC	1110	操作数 1 逻辑与非操作数 2 (清除位)
MVN	1111	操作数 2 取反 (忽略操作数 1)

算术操作 (SUB , RSB , ADD , ADC , SBC , RSC , CMP , CMN) 都是以 32 位整型 (无符号数或补码形式有符号数 , 这两种是等价的) 来处理每个操作数。如果置位了 S 位 (并且 Rd 不是 R15 , 如下), 在 CPSR 中 : 当结果的位[31]发生溢出时置位 V 标志位 ; 当把操作数看成是无符号数时可能将其忽略 , 但如果操作数为补码形式有符号数时将会发出可能错误的警告。当 ALU 的位[31]完成时将置位 C 标志位 (或当移位操作为 LSL #0 时保持不变), 当且仅当结果全为零时置位 Z 标志位 , 结果的位[31]的值将置位 N 标志位 (如果将操作数看成是补码形式有符号数时表明一个负的结果)。

移位

当指定第二个操作数为一个移位寄存器，简形移位器的操作是由指令的移位字段来控制。该字段指示了要执行的移位操作的类型（逻辑左移或右移，算术右移或循环右移）。指令中的一个立即数字段或另一个寄存器的最低位字节（除了 R15）中包含了寄存器的移位数。不同移位类型的译码如图 3-5。

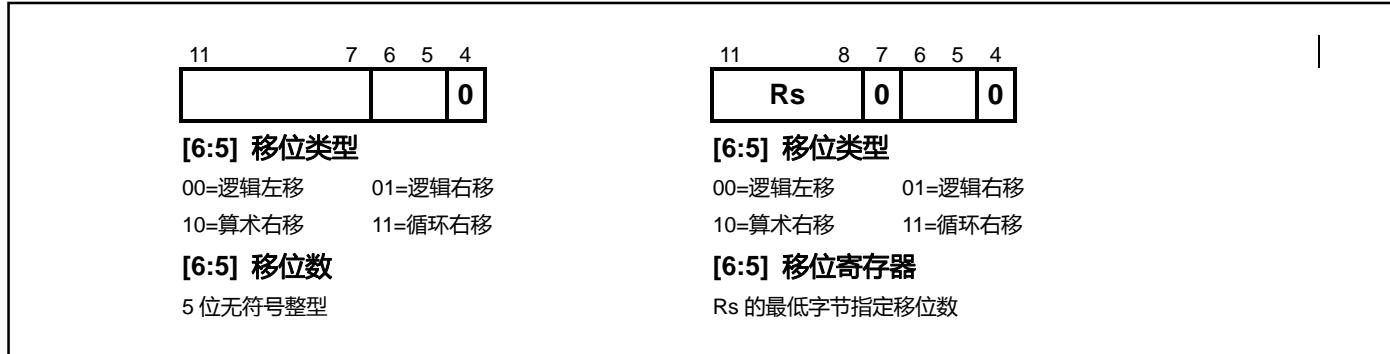


图 3-5. ARM 移位操作

指令指定移位数

当指令指定了移位数，它包含在一个从 0 到 31 之间任意数值的 5 位字段中。逻辑左移（LSL）获取 Rm 的内容并且由指定的移位数将其每一位移动到更高有效位（*more significant position*）。结果的最低有效位将用 0 填充，丢弃未映射到结果的 Rm 的高位，除非在最低有效位是被丢弃的位时，在 ALU 运行在逻辑类别（*class*）（见中止异常）时将会把移位进位的输出锁存到 CPSR 的 C 标志位中。例如，LSL #5 的结果如图 3-6 所示。

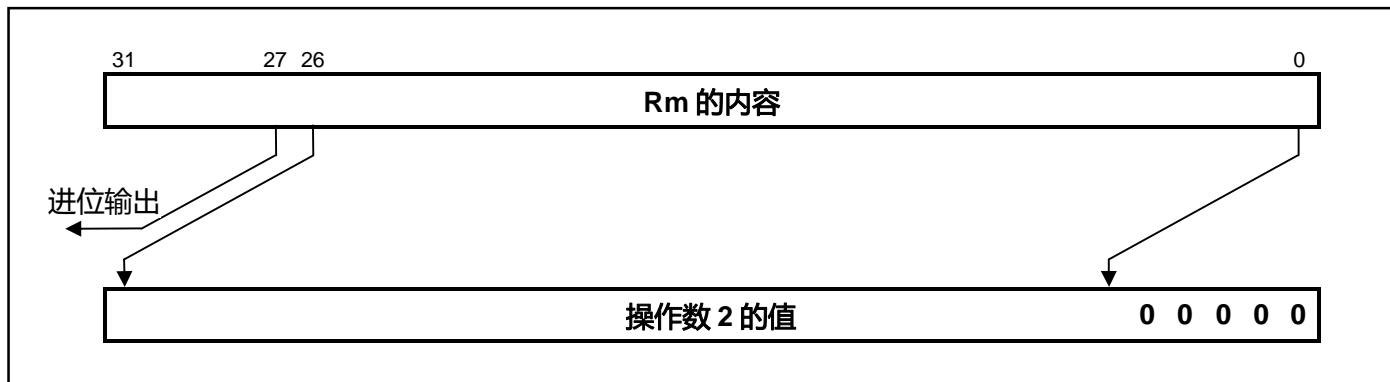


图 3-6. 逻辑左移

注释：LSL #0 是一种特殊情况，此时移位器进位输出为 CPSR 的 C 标志位的先前的值。Rm 的内容如同操作数 2 一样可以直接使用。逻辑右移(LSR)也类似，但是将 Rm 的内容移位到更低的有效位到结果。LSR #5 的结果如图 3-7 所示。

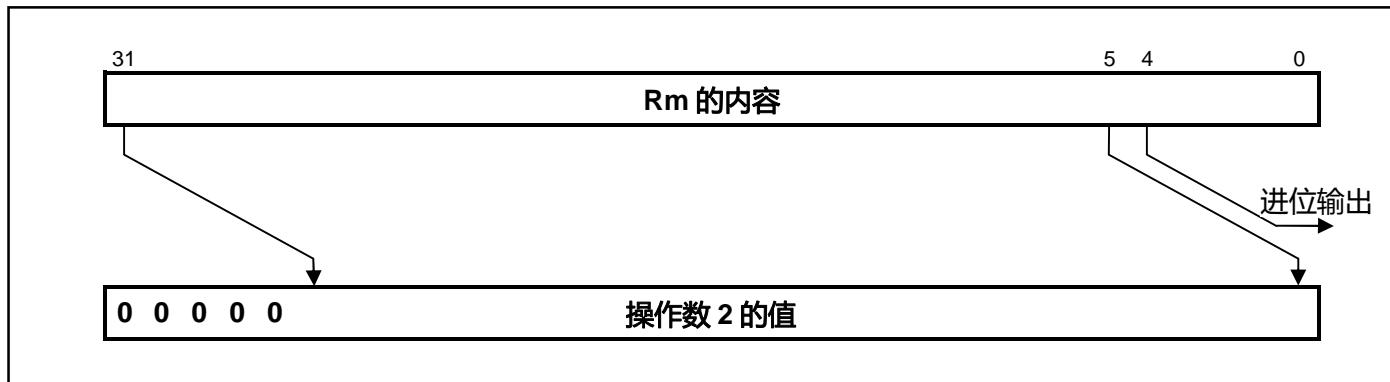


图 3-7. 逻辑右移

对于可能希望对应的移位字段的形式 LSR #0，习惯将其译码为 LSR #32，其结果为 0 并且把 Rm 的位[31]作为进位输出。逻辑右移零是多余的因为它与逻辑左移零相同，所以汇编将会转换 LSR #0（以及 ASR #0 和 ROR #0）为 LSL #0，并且允许指定 LSR #32。

算术右移（**ASR**）与逻辑右移类似，除了最高位填充的是 Rm 的位[31]，而不是 0 了。这将保持二进制补码形式有符号数的符号位。例如，ASR #5 的结果如图 3-8 所示。



图 3-8. 算术右移

对于可能希望给出的移位字段的形式 ASR #0，习惯译码为 ASR #32。Rm 的位[31]再次用于进位输出，并且操作数 2 的各个位也等于 Rm 的位[31]。因此结果为按 Rm 的位[31]的值全为 1 或全为 0。

循环右移（**ROR**）操作重新使用逻辑右移操作时“超出（overshoot）”的位，再引入这些“超出”位到结果的最高端，取代了逻辑右移操作用 0 填充最高端，例如，ROR #5 如图 3-9 所示。

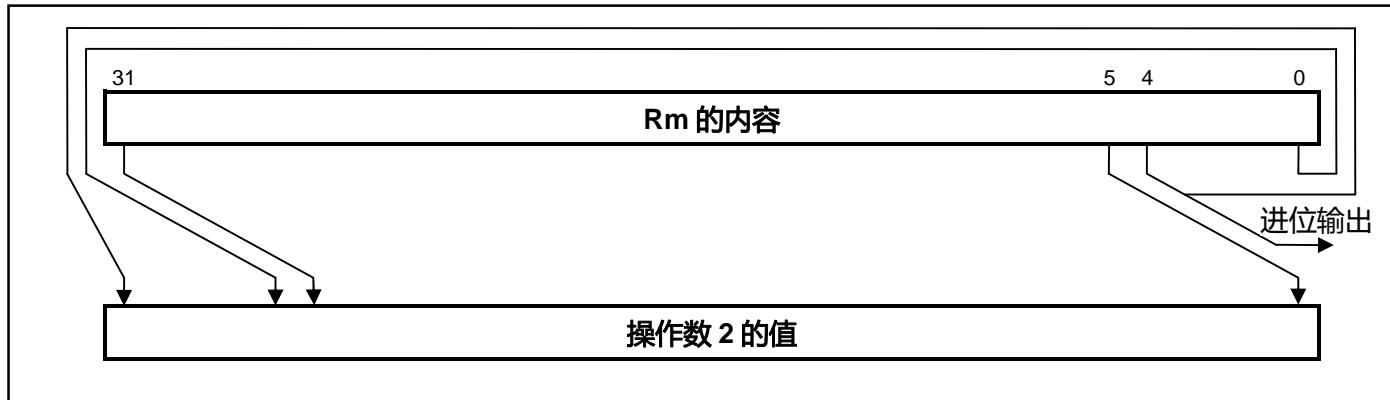


图 3-9. 循环右移

对于可能希望给出的移位字段的形式 ROR #0，习惯译码为简形移位器的特殊功能，扩展循环右移（**RRX**）。这是 33 位量的单个位循环右移，追加了 CPSR 的 C 标志位到 Rm 的内容的最高有效端而形成，如图 3-10 所示。



图 3-10. 扩展循环右移

寄存器指定移位数

只 Rs 的内容的有最低有效字节用于确定移位量。Rs 可以是除了 R15 外任何通用寄存器。

如果这个字节为 0，无符号的 Rm 的内容将当成第二操作数使用，就像移位器进位输出将 CPSR 的 C 标志位的先前的值将继续。

如果这个字节的值在 1 到 31 之间，移位结果将准确的与指令指定移位相匹配，相同的值和移位操作。

如果这个字节的值为 32 或更大，结果将为一个逻辑移位的扩展，上文描述：

1. LSL：为 32 时结果为 0，进位输出等于 Rm 的位[0]。
2. LSL：超过 32 时结果为 0，进位输出也为 0。
3. LSR：为 32 时结果为 0，进位输出等于 Rm 的位[31]。
4. LSR：超过 32 时结果为 0，进位输出也为 0。
5. ASR：为 32 或更大时结果填充和进位输出都等于 Rm 的位[31]。
6. ROR：为 32 时结果等于 Rm，进位输出等于 Rm 的位[31]。
7. ROR：为 n 时，n 远大于 32 时将给出与 ROR 在这个字节为 n-32 时相同的结果和进位输出；因此从 n 中重复地减去 32 直到该数在 1 到 32 的范围内，见上文。

注意：

必须让指令的位[7]中为 0，则寄存器被控制地移位；若为 1 将导致指令变为乘法或未定义指令。

立即数操作数循环

立即数操作数循环字段是一个指定对 8 位立即数移位操作的 4 位无符号整型。该值在 0 到 32 位范围内，并使其两次循环右移在循环字段中的值。这使得能够产生普通常量，例如 2 的所有乘方数。

R15 写入

当 Rd 为非 R15 的寄存器时，ALU 将会更新 CPSR 中的条件码标志位，如上文描述。

当 Rd 为 R15 并且指令中没有置位 S 标志位时，操作结果放置到 R15，CPSR 不受影响。

当 Rd 为 R15 并且置位了 S 标志位时，操作结果放置到 R15，当前模式相应的 SPSR 移动到 CPSR 中。此时允许状态的改变并自动恢复 PC 和 CPSR。这种指令形式不会使用在用户模式。

把 R15 当成操作数使用

如果在数据处理指令中将 R15 (PC) 当成操作数使用，可以直接使用。

PC 值将位指令的地址，加上 8 或 12 个字节发生预取指令。如果移位量是由指令所指定的，则 PC 将超前 8 个字节。如果位移量是由寄存器指定的，则 PC 将超前 12 个字节。

TEQ, TST, CMP 和 CMN 操作码

注意：

TEQ, TST, CMP 和 CMN 操作并不纪录结果，但会置位 CPSR 中的标志位。汇编应当总是置位指令的 S 标志位，即使没有在助记符中指出。

用 TEQ 指令构成的 TEQP 是用于早期的 ARM 处理器，不要使用它：使用 PSR 传送操作代替。

在 ARM920T 中 TEQP 的功能是处理器是在特权模式和未做事的用户模式下移动 SPSR_<mode> 到 CPSR 中。

指令周期时间

数据处理指令增加不同周期数耗时如下：

表 3-4. 指令周期时间

处理类型	周期
正常数据处理	1S
带寄存器指定移位的数据处理	1S + 1I
带 PC 写入的数据处理	1S + 1N
带寄存器指定移位和 PC 写入的数据处理	1S + 1I + 1N

注释：S，N 和 I 分别是定义连续周期（S-cycle），非连续周期（N-cycle）和内部周期（I-cycle）。

汇编语法

- MOV, MVN (单操作数指令)

 $\langle\text{opcode}\rangle\{\text{cond}\}\{\text{S}\} \text{Rd}, \langle\text{Op2}\rangle$
- CMP, CMN, TEQ, TST (指令不产生结果)

 $\langle\text{opcode}\rangle\{\text{cond}\} \text{Rn}, \langle\text{Op2}\rangle$
- AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, ORR, BIC

 $\langle\text{opcode}\rangle\{\text{cond}\}\{\text{S}\} \text{Rd}, \text{Rn}, \langle\text{Op2}\rangle$

其中：

$\langle\text{Op2}\rangle$	Rm{, <shift>} or, <#expression>
{cond}	两个字符条件助记符，如表 3-2 所示。
{S}	如果 S 出现设置条件码（CMP, CMN, TEQ, TST 隐含）
Rd, Rn, Rm	表达式计算到通用寄存器
<#expression>	如果使用的这个，汇编将会试图产生一个移位立即数的 8 位字段来匹配这个表达式，如果不能完成，这将给出一个错误
<shift>	<Shiftname> <register> 或 <shiftname> #expression，或者 RRX（带扩展的循环右移一位）
<shiftname>s	ASL, LSL, LSR, ASR, ROR（ASL 是 LSL 的同义词，他们汇编出相同的代码）

例子

ADDEQ R2, R4, R5	;如果置位了 Z 标志位使得 R2:=R4+R5
TEQS R4, #3	;测试 R4 是否等于 3（这里的 S 实际上是多余的，因为汇编会自动添加）
SUB R4, R5, R7, LSR R2	;逻辑右移 R7，移位数存在 R2 的低字节中，R5 减去其结果，并将答案放入 R4 中
MOV fred	;从子程序中返回
MOVS sub+ROM	;从异常中返回并将 SPSR_<mode>恢复到 CPSR

PSR 传送指令 (MRS , MSR)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。

MRS 和 MSR 指令是由数据处理操作的子集构成的，其执行的是不置位 S 标志位的 TEQ , TST , CMN 和 CMP 指令。译码如图 3-11 所示。

这些指令允许访问 CPSR 和 SPSR 寄存器。MRS 指令允许将 CPSR 或 SPSR_<mode>的内容移动到一个通用寄存器中去。MSR 指令允许将一个通用寄存器中的内容移动到 CPSR 或 SPSR_<mode>寄存器中去。

MSR 指令同样允许在不影响 CPSR 或 SPSR_<mode>的控制位的情况下将一个立即数或寄存器内容传送到条件字段 (N , Z , C 和 V)。在这种情况下，指定的寄存器内容或 32 位立即数的最高 4 位将写入到相关 PSR 的最高 4 位中去。

操作数限制

- 在用户模式下，改变 CPSR 时保护其控制位，所以只有 CPSR 的条件码标志位才能被改变。在其他（特权）模式整个 CPSR 都可以被改变。
- 请注意软件一定不要改变 CPSR 中 T 位的状态。如果发生了处理器将进入一个不可预测的状态。
- 在执行期间，所访问的 SPSR 寄存器是由处理器模式决定的。例如当处理器在 FIQ 模式时只能访问 SPSR_fiq。
- 一定不要指定 R15 为源寄存器或目标地址寄存器。
- 同样的，不要试图在用户模式下访问一个 SPSR，因为没有这样一个寄存器存在。

保留位

ARM920T 中只定义了 PSR 其中的 12 位 (N , Z , C , V , I , F , T 和 M[4:0])；保留剩余的位为将来的处理器版本。参考图 2-6 中 PSR 对各个位的描述。

为保证 ARM920T 的程序与将来处理器保持最大的兼容性，应当遵守以下规则：

- 当改变 PSR 的值时应当保护这些保留位。
- 当检查 PSR 的状态时程序应当不依赖于包含特殊值的保留位，因为将来的处理器可能会将这些位读作 1 或 0。当更改 PSR 寄存器的控制位时，应该使用读-修改-写的策略；这样需要使用 MRS 指令先将合适的 PSR 寄存器传送到一个通用寄存器中去，只改变相关的位，然后使用 MSR 指令将修改后的值返回给 PSR。

例子

以下顺序执行了一个模式的改变：

MRS	R0, CPSR	:复制 CPSR 到 R0
BIC	R0, R0, #0x1F	:清零模式位
ORR	R0, R0, #new_mode	:选择新的模式
MSR	CPSR, R0	:写回到修改了的 CPSR

当目标只是简单地改变 PSR 中条件码标志位时，可以在未干扰到控制位的情况下直接将值写入到标志位中去。以下指令置位了 N , Z , C 和 V 标志位：

MSR CPSR_flg, #0xF0000000 :不顾标志位的先前的状态将其都置位

不要试图将一个 8 位立即数写入到全部的 PSR 中的，因为这样的操作不能保证那些保留位。

MRS (PSR 内容传送到其他寄存器)

31	28 27	23 22 21	16 15	12 11	0
Cond	0 0 0 1 0	Ps	0 0 1 1 1 1	Rd	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[15:12] 目标寄存器**[22] 源 PSR**

0=CPSR

1=SPSR_<当前模式>

[31:28] 条件字段**MSR (其他寄存器内容传送到 PSR)**

31	28 27	23 22 21	12 11	4 3	0
Cond	0 0 0 1 0	Pd	1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0	Rm	

[15:12] 源寄存器**[22] 目标 PSR**

0=CPSR

1=SPSR_<当前模式>

[31:28] 条件字段**MSR (其他寄存器内容或立即数只传送到 PSR 的状态标志位)**

31	28 27 26 25 24 23 22 21	12 11	0
Cond	0 0 1 1 0 Pd 1 0 0 1 0 0 1 1 1 1		源操作数

[22] 目标 PSR

0=CPSR

1=SPSR_<当前模式>

[25] 立即数操作数

0=第二个操作数为寄存器

1=SPSR_<当前模式>

[31:28] 源操作数

11	4 3	0
0 0 0 0 0 0 0 0	Rm	

[3:0] 源寄存器**[11:4] 源操作数为立即数**

11	8 7	0
Rotate	Imm	

[7:0] 无符号 8 位立即数**[11:8] Imm 应用移位****[31:28] 条件字段**

图 3-11. PSR 传送

指令周期时间

PSR 传送耗时 1S 增加的周期，此处 S 是定义连续周期 (S-cycle)。

汇编语法

- MRS – 将 PSR 内容传送到其它寄存器
MRS{cond} Rd,<psr>
- MSR – 将其它寄存器内容传送到 PSR
MSR{cond} <psr>,Rm
- MSR – 将其它寄存器内容只传送到 PSR 的状态标志位
MSR{cond} <psrf>,Rm

分别将 N , C , Z 和 V 写入寄存器内容的最高有效 4 位。

- MSR – 将立即数只传送到 PSR 的状态标志位
MSR{cond} <psrf>,<#expression>
表达式应该作为一个将 N , Z , C 和 V 标志位分别写入到最高有效 4 位的 32 位值。

关键词：

{cond}	两个字符条件助记符，如表 3-2 所示。
Rd, Rm	表达式计算到一个非 R15 的通用寄存器中
<psr>	CPSR , CPSR_all , SPSR 或 SPSR_all (CPSR , CPSR_all 和 SPSR , SPSR_all 为同义词)
<psrf>	CPSR_flg 或 SPSR_flg
<#expression>	如果使用的这个，汇编将会试图产生一个移位立即数的 8 位字段来匹配这个表达式，如果不能完成，这将给出一个错误

例子

在用户模式指令工作如下：

MSR CPSR_all, Rm	; 将 Rm[31:28]写入到 CPSR[31:28]
MSR CPSR_flg, Rm	; 将 Rm[31:28]写入到 CPSR[31:28]
MSR CPSR_flg, #0xA0000000	; 将 0xA 写入到 CPSR[31:28] (置位 N , C ; 清零 Z , V)
MRS Rd, CPSR	; 将 CPSR[31:0]写入到 Rd[31:0]

在特权模式指令工作如下：

MSR CPSR_all, Rm	; 将 Rm[31:0]写入到 CPSR[31:0]
MSR CPSR_flg, Rm	; 将 Rm[31:28]写入到 CPSR[31:28]
MSR CPSR_flg, #0x50000000	; 将 0x5 写入到 CPSR[31:28] (置位 Z , V ; 清零 N , C)
MSR SPSR_all, Rm	; 将 Rm[31:0]写入到 SPSR_<模式>[31:0]
MSR SPSR_flg, Rm	; 将 Rm[31:28]写入到 SPSR_<模式>[31:28]
MSR SPSR_flg, #0xC0000000	; 将 Rm[31: 28]写入到 SPSR_<模式>[31:28] (置位 N , Z ; 清零 C , V)
MRS Rd SPSR	; 将 SPSR_<模式>[31:0]写入到 Rd[31:0]

乘法和乘加指令 (MUL , MLA)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-12 所示。

乘法和乘加指令采用的是一个 8 位布斯 (Booth) 算法来运行整数乘法。

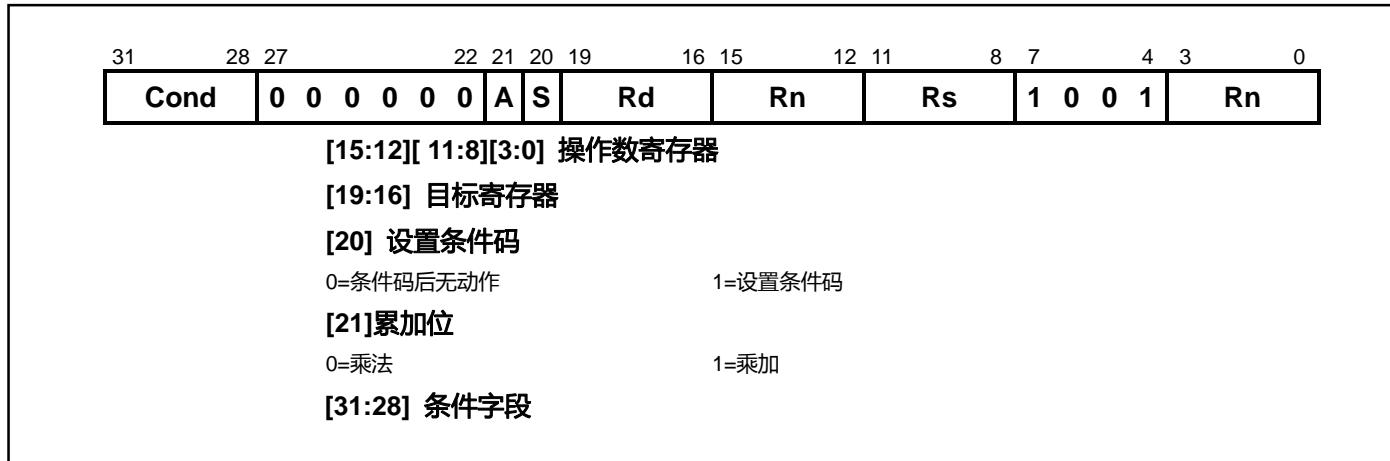


图 3-12. 乘法指令

乘法指令的形式为 $Rd := Rm \times Rs$ 。忽略 Rn ，并且设置为 0，这是为了兼容将来可能更新的指令集。乘加指令的形式为 $Rd := Rm \times Rs + Rn$ ，能在某些情况下免去一个明确的 ADD 指令。两种指令形式都是由可能被看作是有符号(补码形式) 整型或无符号整型的操作数来执行的。

32 位操作数的有符号乘法和无符号乘法的结果唯一不同的在高 32 位中，有符号乘法和无符号乘法结果的低 32 位是相同的。如果这些指令只产生一个乘法的低 32 位，有符号乘法和无符号乘法都能被使用。

例如考虑操作数的乘法运算：

操作数 A	操作数 B	结果
0xFFFFFFFF6	0x0000001	0xFFFFFFFF38

若操作数解释为有符号

操作数 A 的值为 -10，操作数 B 的值为 20，结果为 -200，正确的与 0xFFFFFFFF38 相等

若操作数解释为无符号

操作数 A 的值为 4294967286，操作数 B 的值为 20，结果为 85899345720，正确的与 0x13 FFFFFFF38 相等，所以最低有效 32 位为 0xFFFFFFFF38

操作数限制

目标寄存器 Rd 必须与不同于操作数寄存器 Rm。R15 必须不被用作操作数或目标寄存器。

所有其它寄存器的组合都将给出正确的结果，Rd，Rn 和 Rs 在需要时可能使用相同的寄存器。

CPSR 标志位

设置 CPSR 的状态标志位为可选，由指令的 S 位控制。N (负) 和 Z (零) 标志位由结果正确的置位 (N 等于结果的位[31]，当且仅当结果为零时置位 Z)；C (进位) 标志位由无意义的值置位；V (溢出) 标志位不受影响。

指令周期时间

执行 MUL 耗时 $1S+mI$ 周期，执行 MLA 耗时 $1S+(m+1)I$ 周期，S 和 I 分别是定义连续周期 (S-cycle) 和内部周期 (I-cycle)。

- m 需要完成乘法的 8 位乘数排列周期数，由 Rs 指定的乘数操作数的值控制。其可能值如下
 - 1 如果乘数操作数的位[32:8]全为 0 或全为 1
 - 2 如果乘数操作数的位[32:16]全为 0 或全为 1
 - 3 如果乘数操作数的位[32:24]全为 0 或全为 1
 - 4 所有其它情况

汇编语法

MUL{cond}{S} Rd,Rm,Rs
MLA{cond}{S} Rd,Rm,Rs,Rn

- {cond} 两个字符条件助记符，如表 3-2 所示。
- {S} 如果 S 出现设置条件码
- Rd, Rm, Rs, Rn 表达式计算到一个非 R15 的通用寄存器中

例子

```
MUL      R1, R2, R3          ; R1:=R2×R3
MLAEQS R1, R2, R3, R4    ;条件执行 R1:=R2×R3+R4，并设置条件码
```

长乘法和长乘加指令 (MULL , MLAL)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-13 所示。

长乘指令执行的是两个 32 位操作数的整数乘法，并产生 64 位结果。可选累加的有符号和无符号乘法使指令增加到 4 种变化。

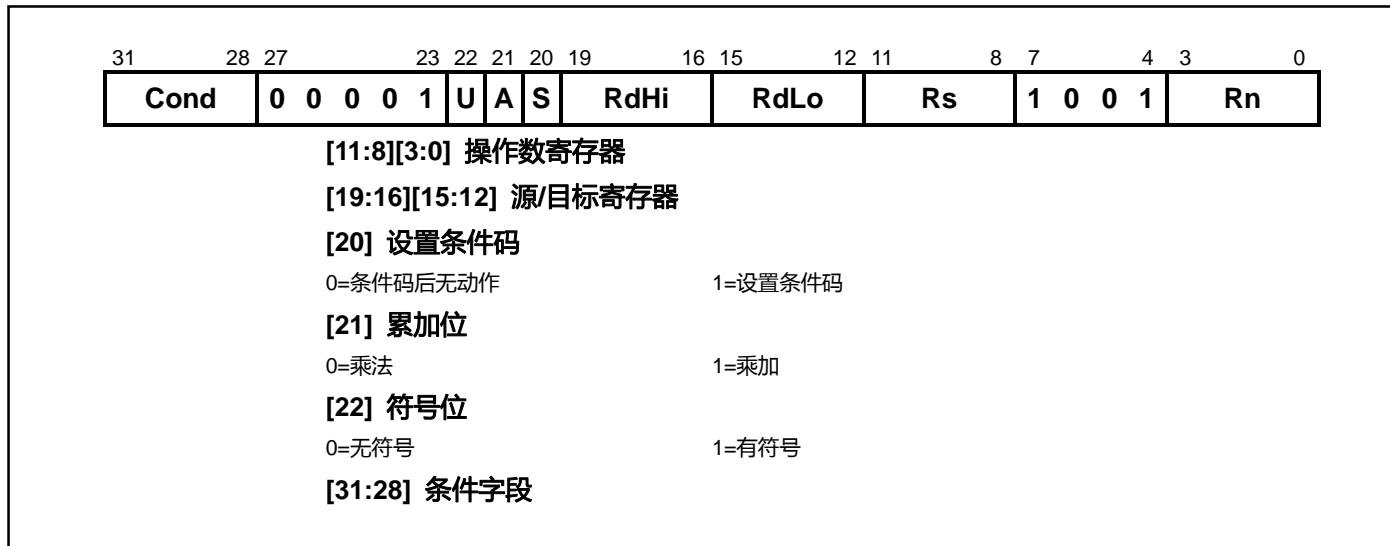


图 3-13. 长乘指令

相乘的形式 (UMULL 和 SMULL) 是用两个 32 位数相乘产生一个 64 位结果，其形式为 $RdHi, RdLo := Rm \times Rs$ 。64 位结果的低 32 位写入到 $RdLo$ ，而结果的高 32 位写入到 $RdHi$ 。

乘加的形式 (UMLAL 和 SMLAL) 是用两个 32 位数相乘并加上一个 64 位数后产生一个 64 位结果，其形式为 $RdHi, RdLo := Rm \times Rs + RdHi, RdLo$ 。64 位结果的低 32 位写入到 $RdLo$ ，而结果的高 32 位写入到 $RdHi$ 。

UMULL 和 UMLAL 指令把其所有操作数都看作无符号二进制数并写入一个无符号 64 位结果。SMLAL 指令其所有操作数都看作补码形式有符号数并写入一个补码形式有符号 64 位结果。

操作数限制

R15 必须不被用作操作数或目标寄存器。

RdHi , RdLo 和 Rm 都必须指定不同的寄存器

CPSR 标志位

设置 CPSR 的状态标志位为可选，由指令的 S 位控制。N (负) 和 Z (零) 标志位由结果正确的置位 (N 等于结果的位[63]，当且仅当结果的 64 位全为零时置位 Z)；C (进位) 和 V (溢出) 标志位都是由无意义的值置位。

指令周期时间

执行 MULL 耗时 $1S + (m+1)I$ 周期，执行 MLAL 耗时 $1S + (m+2)I$ 周期， m 是需要完成乘法的 8 位乘数排列周期数，由 Rs 指定的乘数操作数的值控制。其可能值如下：

有符号指令 SMULL , SMLAL :

- 1 如果乘数操作数的位[32:8]全为 0 或全为 1
- 2 如果乘数操作数的位[32:16]全为 0 或全为 1
- 3 如果乘数操作数的位[32:24]全为 0 或全为 1
- 4 所有其它情况

有符号指令 SMULL , SMLAL :

- 1 如果乘数操作数的位[32:8]全为 0
- 2 如果乘数操作数的位[32:16]全为 0
- 3 如果乘数操作数的位[32:24]全为 0
- 4 所有其它情况

S 和 I 分别是定义连续周期 (S-cycle) 和内部周期 (I-cycle)。

汇编语法

助记格式	描述	效果
UMULL{cond}{S} RdLo,RdHi,Rm,Rs	无符号长乘法	$32 \times 32 = 64$
UMLAL{cond}{S} RdLo,RdHi,Rm,Rs	无符号长乘加	$32 \times 32 + 64 = 64$
SMULL{cond}{S} RdLo,RdHi,Rm,Rs	有符号长乘法	$32 \times 32 = 64$
SMLAL{cond}{S} RdLo,RdHi,Rm,Rs	有符号长乘加	$32 \times 32 + 64 = 64$

其中：

- {cond} 两个字符条件助记符，如表 3-2 所示。
 {S} 如果 S 出现设置条件码
 RdLo, RdHi, Rm, Rs 表达式计算到一个非 R15 的通用寄存器中

例子

```
UMULL    R1,  R4,  R2,  R3      ;R4,R1:=R2×R3
UMLALS  R1,  R5,  R2,  R3      ;条件执行 R5,R1:=R2×R3+R5,R1，并设置条件码
```

单数据传送 (LDR , STR)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-14 所示。

单数据传送指令用于加载或存储个单字节或字的数据。传输中的存储器地址的计算是基址寄存器加上或减去一个偏移量得到的。

如果规定了自动变址 (*auto-indexing*), 此计算的结果将会写回到基址寄存器。

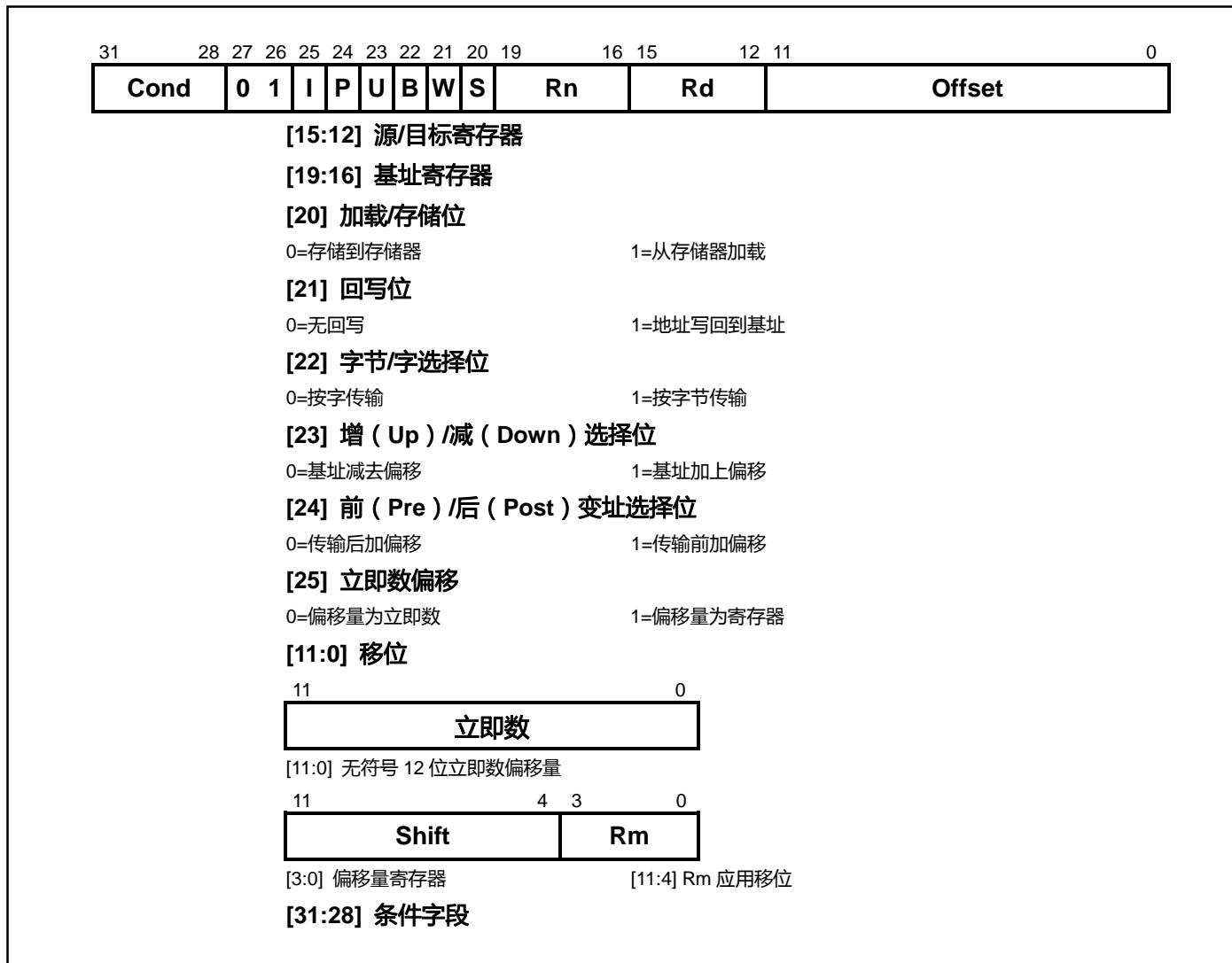


图 3-14. 单数据传送指令

偏移和自动变址

相对基址寄存器的偏移量可能为一个指令中的 12 位的无符号二进制立即数，或者是一个次寄存器（可能以某些方法移位了）。偏移量可能是从基址寄存器 R_n 上被加上 ($U=1$) 或被减去 ($U=0$)。执行偏移更改可能在使用基址为传输地址之前（前变址， $P=1$ ）或之后（后变址， $P=0$ ）。

W 位提供了可选择自增和自减寻址模式。被更改的基址值可能被写回到基址中去 ($W=1$)，或者保持原来的基址值不变 ($W=0$)。在后变址的情况下，回写位不是必须的并通常设置为 0，因而可以在偏移为 0 时保持原来的基址值不变。因此后变址数据传送通常回写被更改的基址。只在后变址数据传送使用 W 位是在特权模式代码，当设置 W 位强制在非特权模式下传输，允许操作系统在适当使用存储器管理硬件屏蔽时在系统中生成一个用户地址。

移位寄存器偏移

8 位的移位控制位在数据传送指令中章节中描述已经过。然而不能在此指令类型中使用寄存器指定移位数。见图 3-5。

字节与字

该指令类型可以按字节 ($B=1$) 或按字 ($B=0$) 在 ARM920T 寄存器和存储器之间进行数据传输。

ARM920T 核的 BIGEND 控制信号端将影响到 LDR (B) 和 STR (B) 指令的行为。两种可能的配置描述如下。

小端配置

单字节的加载 (LDRB) 预取数据，如果提供的地址以字为边界，则为装入数据总线 0 到 7 位，如果它为字地址加上一个字节，则为装入数据总线 8 到 15 位，等等。被选择的字节将放置在目标寄存器的低 8 位，寄存器的其余位填充 0。请参见图 2-2。

单字节的存储 (STRB) 重复 4 次将源寄存器的低 8 位输出访问数据总线 0 到 31 位。外部存储器系统应当激活相应的字节分系统来储存这些数据。

单字的加载 (LDR) 将正常按字对齐寻址使用。然而，一个按字边界的地址偏移将引起循环移位数据到寄存器中，寻址字节占据了位[7:0]。这意味着半字按字边界的偏移 0 和 2 的访问将被正确地加载到寄存器的位[15:0]。两次移位操作接着都必须清除或者标记扩展的高 16 位。

单字的存储 (STR) 应当生成一个字对齐地址。如果地址不是按字对齐，以字呈现的数据总线不受其影响。也就是说，存储寄存器的位[31]总是出现在数据总线输出的 31 位。

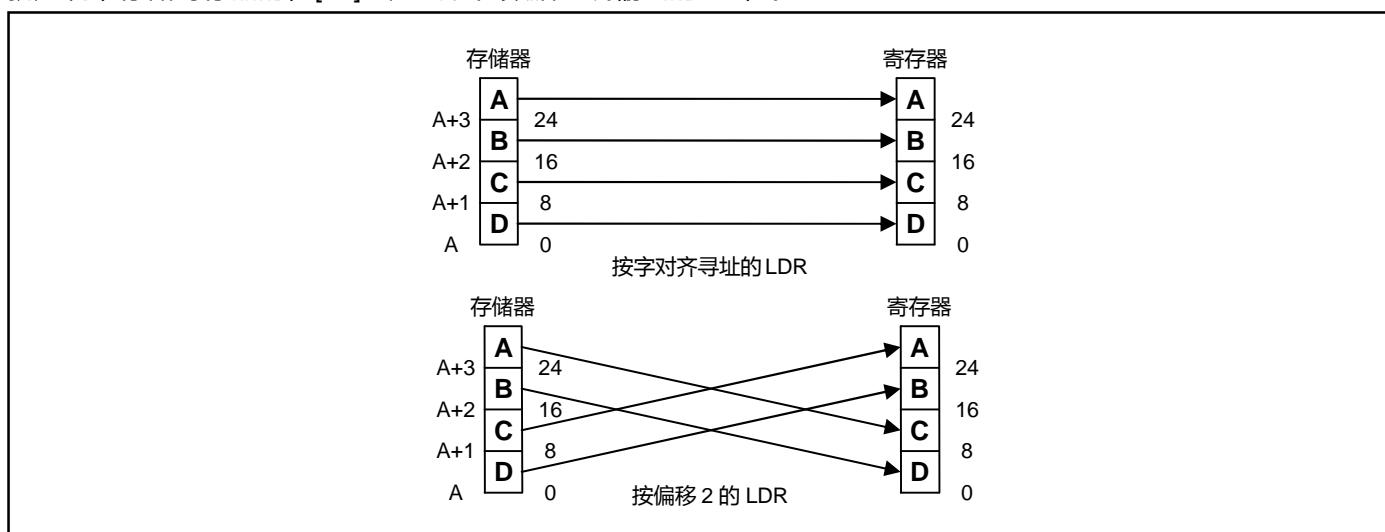


图 3-15. 小端模式偏移寻址

大端配置

单字节的加载 (LDRB) 预取数据 , 如果提供的地址以字为边界 , 则为装入数据总线 24 到 31 位 , 如果它为字地址加上一个字节 , 则为装入数据总线 16 到 23 位 , 等等。被选择的字节将放置在目标寄存器的低 8 位 , 寄存器的其余位填充 0 。请参见图 2-1。

单字节的存储 (STRB) 重复 4 次将源寄存器的低 8 位输出访问数据总线 0 到 31 位。外部存储器系统应当激活相应的字节分系统来储存这些数据。

单字的加载 (LDR) 将正常按字对齐寻址使用。然而 , 一个按字边界的地址偏移将引起循环移位数据到寄存器中 , 寻址字节占据了位 [7:0] 。这意味着半字按字边界的偏移 0 和 2 的访问将被正确地加载到寄存器的位 [15:0] 。两次移位操作接着都必须清除或者标记扩展的高 16 位。

单字的存储 (STR) 应当生成一个字对齐地址。如果地址不是按字对齐 , 以字呈现的数据总线不受其影响。就是说 , 存储寄存器的位 [31] 总是出现在数据总线输出的 31 位。

R15 的使用

如果 R15 被指定为基址寄存器 (Rn) , 必须不指定回写。当把 R15 作为基址寄存器使用时 , 你必须记住其包含了一个超前当前指令地址 8 个字节的地址。

R15 必须不被指定为寄存器偏移 (Rm) 。

当 R15 为一个寄存器存储 (STR) 指令的源寄存器 (Rd) 时 , 将指令地址加 12 字节存储值。

限制是根据基址寄存器的使用

当配制为数据中止 , 以下范例代码很难展开为基址寄存器 Rn , 并在中止处理前更新。某些时候它也许不可能预测其初始值。

中止结束后 , 以下范例代码很难展开为基址寄存器 Rn , 并在中止处理前更新。某些时候它也许不可能预测其初始值。

例子

LDR R0, [R1], R1

因此 , 一个后变址的 LDR 或 STR Rm 是与 Rn 相同的寄存器 , 可以不使用。

数据中止

一个到合法地址或从合法地址的传输可能会给存储器管理系统带来问题。举个例子 , 在一个使用虚拟存储器的系统中 , 需要的数据也许不存在与主存储器中。存储器管理能在处理器 ABORT 输入端为高时显示出问题 , 然后引起数据中止陷阱。由系统软件来解决问题的原因 , 然后重新启动指令并继续原程序。

指令周期时间

正常执行 LDR 耗时 $1S+1N+1I$ 周期，执行 LDR PC 耗时 $2S + 2N + 1I$ 增加周期，此处 S, N 和 I 分别是定义连续周期 (S-cycle)，非连续周期 (N-cycle) 和内部周期 (I-cycle)。执行 STR 指令耗时 $2N$ 增加周期

汇编语法

<LDR|STR>{cond}{B}{T} Rd,<Address>

其中

LDR	从存储器加载到寄存器
STR	从寄存器存储到存储器
{cond}	两个字符条件助记符，如表 3-2 所示。
{B}	如果出现 B 则按字节传输，否则按字传输
{T}	如果出现 T, W 位将被后变址指令置位，强制非特权模式来传输周期。当前变址寻址模式不允许指定或表明 T 位
Rd	表达式计算到一个有效通用寄存器中
Rn, Rm	表达式计算到一个通用寄存器中。如果 Rn 为 R15，汇编将会从偏移值中减 8，以允许 ARM920T 流水线操作。在此情况下不要指定基址回写。
<address>	可以为： 生成地址的表达式： 1 汇编将会试图生成一个用于 PC 为基址指令和由计算表达式得到的调整立即数偏移寻址。如果寻址超出范围，将产生错误。 前变址寻址说明： 2 [Rn] 零偏移 [Rn,<#expression>]{!} 偏移<expression>字节 [Rn,{+/-}Rm{,<shift>}]{!} 偏移+/-变址寄存器内容，由<shift>指定 后变址寻址说明： 3 [Rn],<#expression> 偏移<expression>字节 [Rn],{+/-}Rm{,<shift>} 偏移+/-变址寄存器内容，由<shift>指定 <shift> 通用移位操作（见数据处理指令），但你不能由寄存器指定移位量 {!} 如果出现！，写回基址寄存器（置位 W 位）

例子

STR	R1, [R2, R4]!	; 存储 R1 到 R2+R4 (都为寄存器) 并回写地址到 R2
STR	R1, [R2], R4	; 存储 R1 到 R2，回写 R2+R4 到 R2
LDR	R1, [R2, #16]	; 从 R2+16 的内容加载到 R1，但不回写
LDR	R1, [R2, R3, LSL#2]	; 从 R2+R3*4 的内容加载到 R1
LDREQB	R1, [R6, #5]	; 条件从 R6+5 加载到 R1 位[7:0]，位[31:8]填充 0
STR	R1, PLACE	; 生成 PC 相对于地址 PLACE 的偏移

PLACE

半字和有符号数据传送 (LDRH/STRH/LDRSB/STRSH)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-16 所示。

该指令用于加载或存储个半字数据和加载符号扩展字节或半字数据。传输中的存储器地址的计算是基址寄存器加上或减去一个偏移量得到的。如果需要自动变址此计算的结果会被写回到基址寄存器。

31	28	27	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0	
Cond	0	0	0	P	U	0	W	L		Rn		Rd	0	0	0	0	1	S	H	1	Rm

[3:0] 偏移寄存器
[6][5] S H
0 0 = SWP 指令 0 1 = 无符号半字
1 1 = 有符号字节 1 1 = 有符号半字
[15:12] 源/目标寄存器
[19:16] 基址寄存器
[20] 加载/存储位
0=存储到存储器 1=从存储器加载
[21] 回写位
0=无回写 1=地址写回到基址
[23] 增 (Up) /减 (Down) 选择位
0=基址减去偏移 1=基址加上偏移
[24] 前 (Pre) /后 (Post) 变址选择位
0=传输后加偏移 1=传输前加偏移
[31:28] 条件字段

图 3-16. 带寄存器偏移的半字和有符号数据传输

31	28	27	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
Cond	0	0	0	P	U	0	W	L		Rn		Rd	1	S	H	1	Offset	Offset		

[3:0] 立即数偏移 (低半字)
[6][5] S H
0 0 = SWP 指令 0 1 = 无符号半字
1 1 = 有符号字节 1 1 = 有符号半字
[11:8] 立即数偏移 (高半字)
[15:12] 源/目标寄存器
[19:16] 基址寄存器
[20] 加载/存储位
0=存储到存储器 1=从存储器加载
[21] 回写位
0=无回写 1=地址写回到基址
[23] 增 (Up) /减 (Down) 选择位
0=基址减去偏移 1=基址加上偏移
[24] 前 (Pre) /后 (Post) 变址选择位
0=传输后加偏移 1=传输前加偏移
[31:28] 条件字段

图 3-17. 带寄存器偏移和自动变址的半字和有符号数据传输

偏移和自动变址

相对基址寄存器的偏移量可能为一个指令中的 8 位的无符号二进制立即数，或者是一个次寄存器。8 位偏移量由指令字的位[11:8]和位[3:0] 连接组成的，位[11]为 MSB，位 0 为 LSB。偏移量可能是从基址寄存器 Rn 上被加上 (U=1) 或被减去 (U=0)。执行偏移更改可能在使用基址为传输地址之前 (前变址，P=1) 或之后 (后变址，P=0)。

W 位提供了可选择自增和自减寻址模式。被更改的基址值可能被写回到基址中去 (W=1)，或者保持原来的基址值不变 (W=0)。在后变址的情况下，回写位不是必须的并通常设置为 0，因而可以在偏移为 0 时保持原来的基址值不变。因此后变址数据传送通常回写被更改的基址。

当选者后变址寻址时不应当将回写位被置位 (W=1)。

半字加载和存储

设置 S=0 和 H=1 用于在 ARM920T 寄存器和存储器之间的无符号半字传输。

LDRH 和 STRH 指令的功能受 BIGEND 控制信号端的影响。两种可能的配置如以下段落描述。

单字节和半字加载

S 位控制有符号扩展数据的加载。当 S=1 时 H 位选择数据为字节 (H=0) 还是半字 (H=1)。当选择有符号操作时不应当将 L 位置为低。

LDRSB 指令加载被选择的字节装入目标寄存器的位[7:0]，目标寄存器的位[31:8]被设置为符号位位[7]的值。

LDRSH 指令加载被选择的半字装入目标寄存器的位[15:0]，目标寄存器的位[31:16]被设置为符号位位[15]的值。

LDRSB 和 LDRSH 指令的功能受 BIGEND 控制信号端的影响。两种可能的配置如以下段落描述。

字节存储顺序和字节/半字选择

小端配置

有符号字节的加载 (LDRSB) 预取数据，如果提供的地址以字为边界，则为装入数据总线 0 到 7 位，如果它为字地址加上一个字节，则为输入的数据总线 8 到 15 位，等等。被选择的字节将放置在目标寄存器的低 8 位，寄存器的其余位填充字节的符号位位[7]。请参见图 2-2。

半字加载 (LDRSH 或 LDRH) 预取数据，如果提供的地址以字为边界，则为装入数据总线 0 到 15 位，如果以半字为边界，则为装入数据总线 16 到 31 位 (A[1]=1)。提供的地址应当总是以半字为边界对齐。如果提供的地址的位[0]为高，ARM920T 将会加载一个不可预测的值。被选择的半字被放置在目标寄存器的低 16 位。无符号半字加载 (LDRH) 的寄存器的高 16 位填充 0，有符号加载 (LDRSH) 的寄存器的高 16 位填充半字的位[15]符号位。

半字存储 (STRH) 重复 2 次将源寄存器的低 16 位输出访问数据总线 0 到 31 位。外部存储器系统应当激活相应的字节分系统来储存这些数据。

注意地址必须以半字对齐，如果地址的位[0]为高，这将带来不可预测的行为。

大端配置

有符号字节加载 (LDRSB) 预取数据 , 如果提供的地址以字为边界 , 则为装入数据总线 24 到 31 位 , 如果它为字地址加上一个字节 , 则为装入数据总线 16 到 23 位 , 等等。被选择的字节将放置在目标寄存器的低 8 位 , 寄存器的其余位填充字节的位[7]符号位。请参见图 2-1。

半字加载 (LDRSH 或 LDRH) 预取数据 , 如果提供的地址以字为边界 , 则为装入数据总线 16 到 31 位 , 如果以半字为边界 , 则为装入数据总线 0 到 15 位 (A[1]=1) 。提供的地址应当总是以半字为边界对齐。如果提供的地址的位[0]为高 , ARM920T 将会加载一个不可预测的值。被选择的半字被放置在目标寄存器的低 16 位。无符号半字加载 (LDRH) 的寄存器的高 16 位填充 0 , 有符号加载 (LDRSH) 的寄存器的高 16 位填充半字的位[15]符号位。

半字存储 (STRH) 重复 2 次将源寄存器的低 16 位输出访问数据总线 0 到 31 位。外部存储器系统应当激活相应的字节分系统来储存这些数据。

注意地址必须以半字对齐 , 如果地址的位[0]为高 , 这将带来不可预测的行为。

R15 的使用

如果 R15 被指定为基址寄存器 (Rn) , 必须不指定回写。当把 R15 作为基址寄存器使用时 , 你必须记住其包含了一个超前当前指令地址 8 个字节的地址。

R15 必须不被指定为寄存器偏移 (Rm) 。

当 R15 为一个半字存储 (STRH) 指令的源寄存器 (Rd) 时 , 将指令地址加 12 字节存储值。

数据中止

一个到合法地址或从合法地址的传输可能会给存储器管理系统带来问题。举个例子 , 在一个使用虚拟存储器的系统中 , 需要的数据也许不存在与主存储器中。存储器管理能在处理器 ABORT 输入端为高时显示出问题 , 然后引起数据中止陷阱。由系统软件来解决问题的原因 , 然后重新启动指令并继续原程序。

指令周期时间

正常执行 LDR (H , SH , SB) 耗时 $1S+1N+1I$ 周期 , 执行 LDR (H , SH , SB) PC 耗时 $2S + 2N + 1I$ 增加周期。此处 S , N 和 I 分别是定义连续周期 (S-cycle) , 非连续周期 (N-cycle) 和内部周期 (I-cycle) 。执行 STRH 指令耗时 $2N$ 增加周期

汇编语法

<LDR|STR>{cond}<H|SH|SB> Rd,<Address>

其中

LDR	从存储器加载到寄存器
STR	从寄存器存储到储存器
{cond}	两个字符条件助记符 , 如表 3-2 所示。
H	传输半字量
SB	加载有符号扩展字节 (只为 LDR 的值)
SH	加载有符号扩展半字(只为 LDR 的值)
Rd	表达式计算到一个有效通用寄存器中
<address>	可以为 :
	生成地址的表达式 :
1	汇编将会试图生成一个用于 PC 为基址指令和由计算表达式得到的调整立即数偏移寻址。这将为一个相对 PC 的前变址寻址。如果寻址超出范围 , 将产生错误。
	前变址寻址说明 :
2	[Rn] 零偏移
	[Rn,<#expression>]{!} 偏移<expression>字节
	[Rn,{+/-}Rm{,<shift>}]{!} 偏移+/-变址寄存器内容 , 由<shift>指定
	后变址寻址说明 :
3	[Rn],<#expression> 偏移<expression>字节
	[Rn],{+/-}Rm{,<shift>} 偏移+/-变址寄存器内容 , 由<shift>指定
4	Rn 和 Rm 为表达式计算到一个放如寄存器中的数。如果 Rn 为 R15 , 汇编将会从偏移量中减去 8 , 以允许 ARM920T 流水线操作。在此情况下不要指定基址回写。
{!}	如果出现! , 写回基址寄存器 (置位 W 位)

例子

LDRH	R1, [R2, -R3]!	;从 R2-R3 (都为寄存器) 内包含半字地址的内容加载到 R1 , 并写回地址到 R2
STRH	R3, [R4, #14]	;从 R3 存储半字到 R4+14 中 , 无回写
LDRSB	R8, [R2], #-223	;从 R2+16 的内容加载到 R1 , 但不回写
LDRNESH	R11, [R0]	;从 R2 内包含字节地址的内容加载到带符号扩展的 R8 , 写回 R2-223 到 R2
HERE		;生成 PC 相对偏移量到地址 FRED
STRH	R5, [PC, #(FRED-HERE-8)]	;从 R5 存储半字到地址 FRED 中
FRED		

块数据传送 (LDM , STM)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-18 所示。

块数据传送指令用于加载 (LDM) 或存储 (STM) 当前可见寄存器的任意子集。其支持所有可能堆栈模式，维持递增或递减的存储器满堆栈或空堆栈，是一条保存或恢复上下文 (context)，或移动主存储器周围大块数据的非常高效的指令。

寄存器列表

指令能够引起在当前组的任意寄存器的传输（非用户模式程序同样能够从或到用户组的传输，见下）。寄存器列表是一个指令中的 16 位字段，每一位都相当于一个寄存器。寄存器字段的位[0]为 1 将会引起 R0 被传输，为 0 将不会引起其被传输；同样的位[1]控制 R1 的传输，等等。

寄存器的任意子集或所有寄存器都能被指定。唯一的限制是寄存器列表不应当为空。

任何时候 R15 被存储到存储器时，存储值为 STM 指令的地址加上 12。

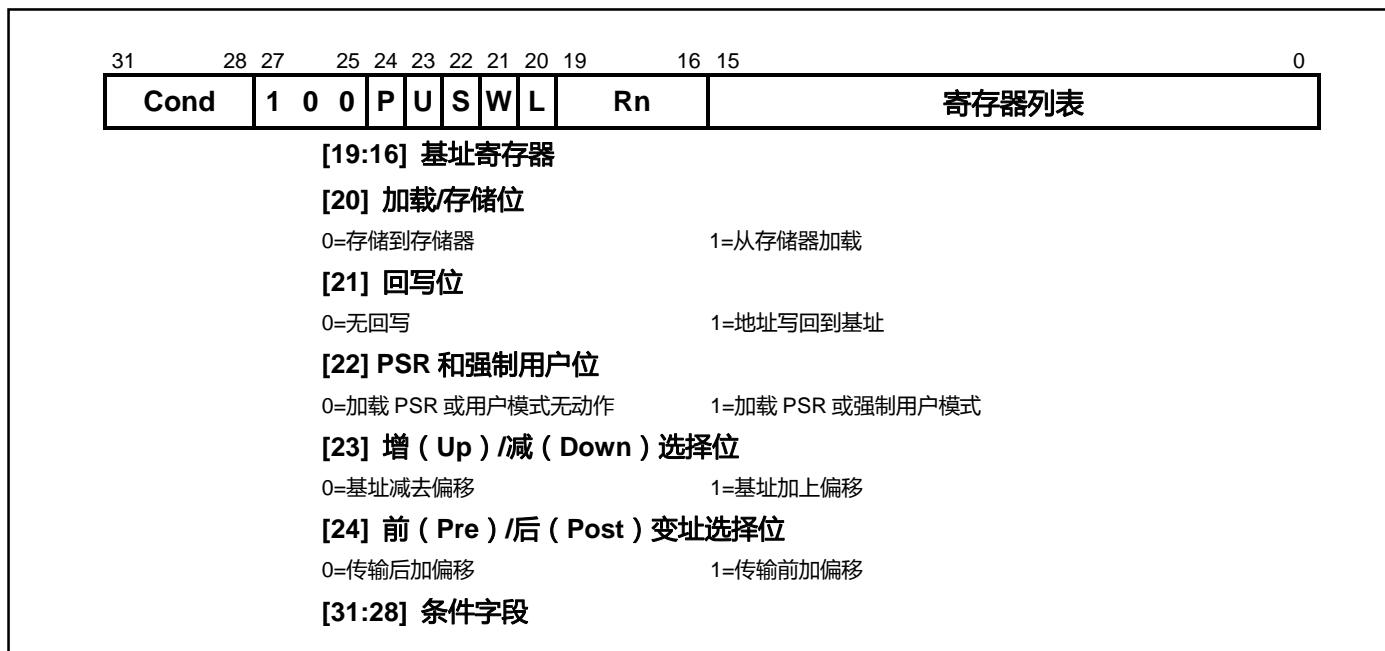


图 3-18. 块数据传输指令

寻址模式

传输地址由基址寄存器 (Rn) 的内容，前/后变址位 (P) 和增/减选择位 (U) 而决定。寄存器按低到高的顺序传输，故 R15 (如果列表含有) 将总是最后被传输。最低寄存器总是传输到/从最低的存储器地址。用图示的方法，考虑 R1 , R5 和 R7 的传输情况，当 Rn=0x1000 并且需要更改基址的回写 (W=1)。图 3.19-22 显示了寄存器的传输顺序，地址的使用和指令完成后 Rn 的值。

在所以情况下，更改基址的写回都不是必须的 (W=0)，Rn 将被固定其 0x1000 的初始值，除非其同样在一个加载多寄存器指令的传输列表中，它被加载值覆盖了。（请核对其意义）……

地址对齐

地址应当正常的以字对齐量，非字对齐地址应当不影响指令。然而，地址的低 2 位将显现在 A[1:0]，而且可能由存储器系统解释。

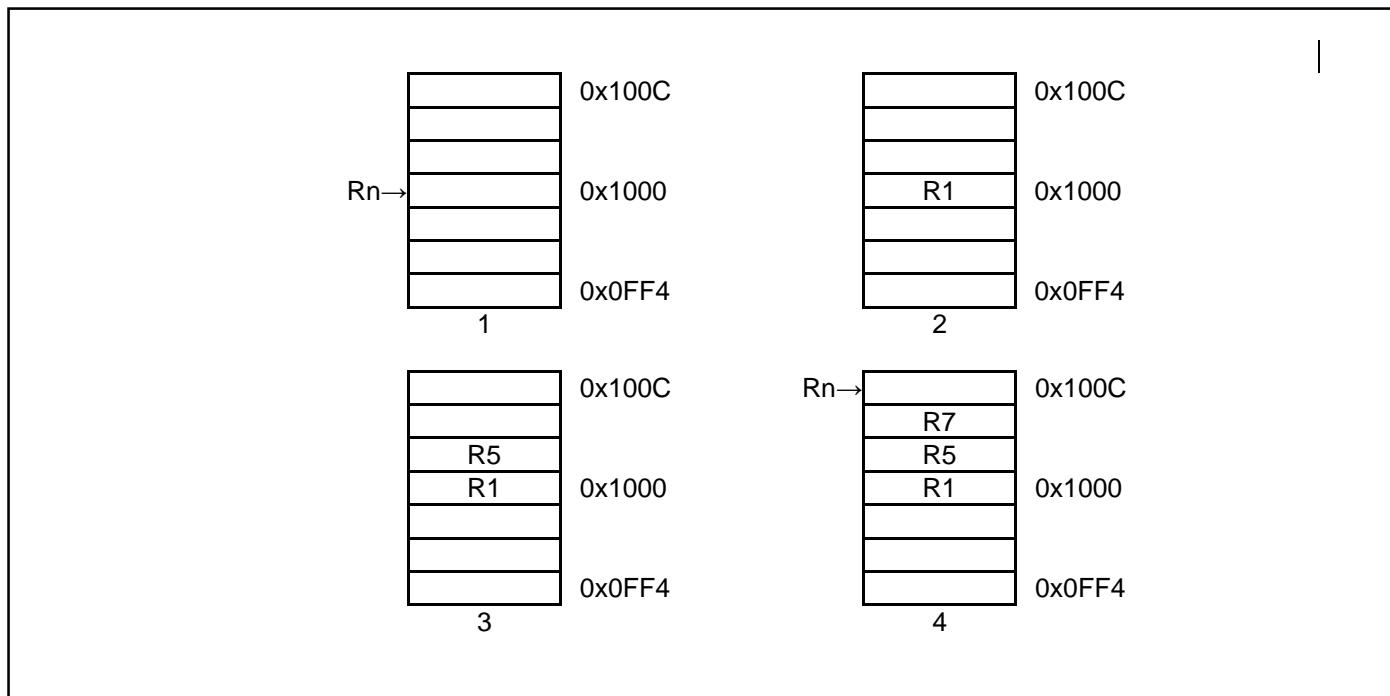


图 3-19. 后递增寻址

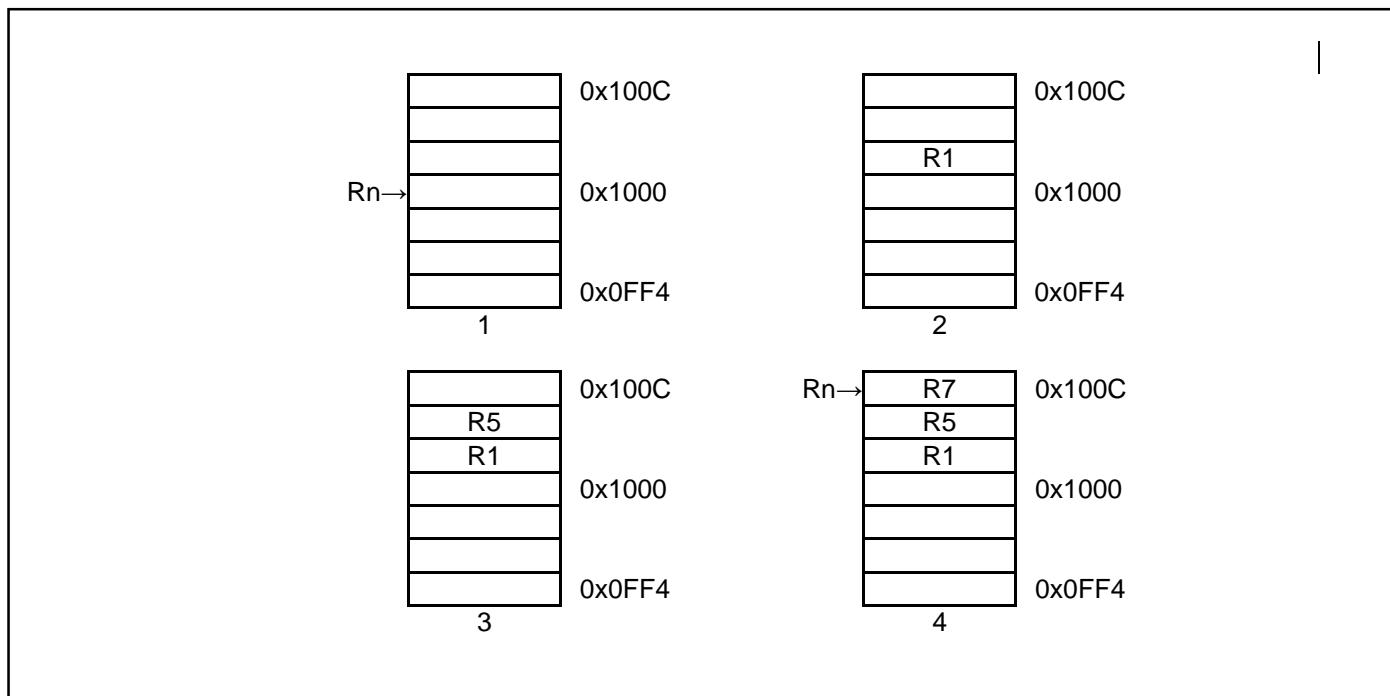


图 3-20. 前递增寻址

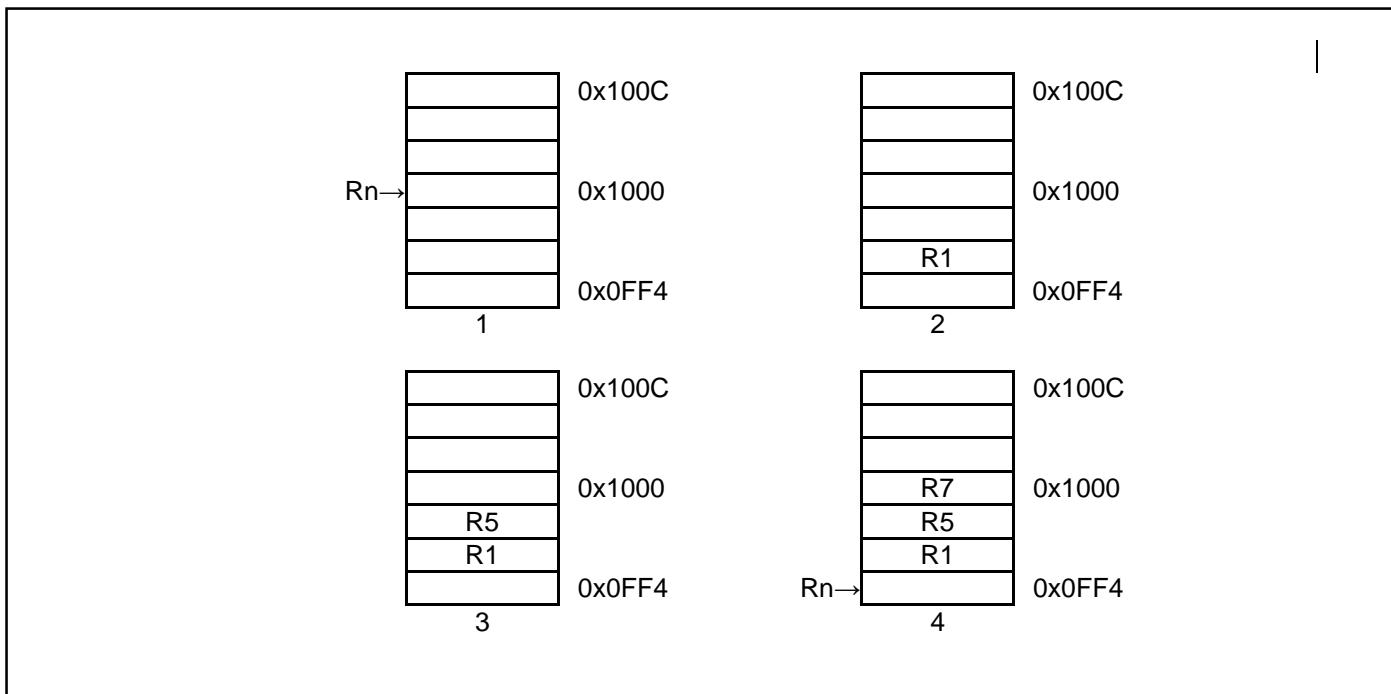


图 3-21. 后递减寻址

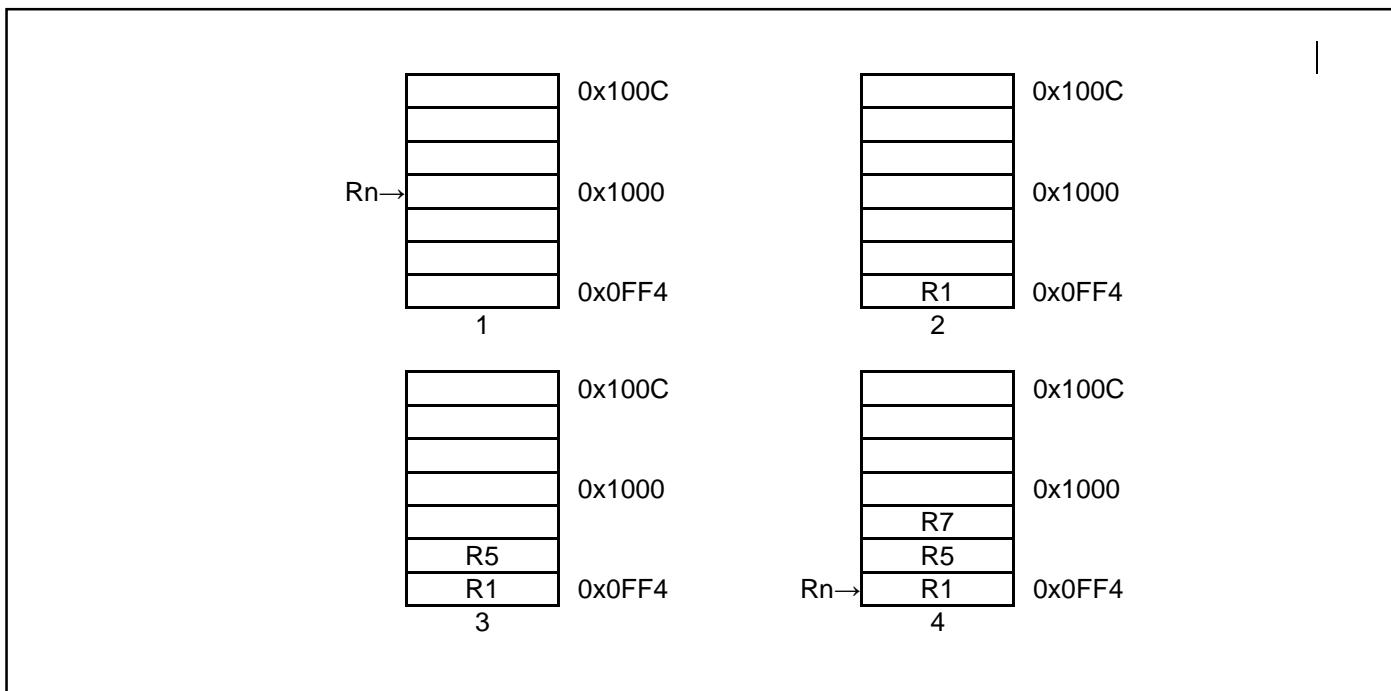


图 3-22. 前递减寻址

S 位的使用

当 LDM/STM 指令中置位了 S 位，其依靠传输列表中可用的 R15 和指令类型。S 位应当只在特权模式下执行的指令中被置位。

R15 在传输列表中并且 S 位置位的 LDM (模式改变)

如果指令为 LDM，同时 SPSR_<mode> 被传输到 CPSR，如 R15 被加载。

R15 在传输列表中并且 S 位置位的 STM (用户组传输)

传输的寄存器是用户组的，而不是相当前模式的。这有意于在处理器切换时保存用户状态。基址回写不应当在使用此机制时使用。

R15 不在传输列表中并且 S 位置位 (用户组传输)

包括 LDM 和 STM 指令，传输的是用户组寄存器，而不是相当前模式的寄存器组。这有意于在处理器切换时保存用户状态。基址回写不应当在使用此机制时使用。

当指令为 LDM，注意当下一周期时必须不从堆寄存器中读取（插入假指令，如在 LDM 后插入 MOV R0, R0 将确保安全）。

使用 R15 作为基址

R15 不应当在任何 LDM 或 STM 指令中用作基址寄存器。

寄存器列表中基址的内容

当指定了回写，基址将在指令的第二个周期末尾被写回。当为 STM 时，第一个寄存器在第二个周期开始时被写出。一个包含存储基址的 STM，将第一个寄存器作为基址来存储将因此存储无符号值，尽管基址在传输顺序的第二或更后，都将存储更改值。如果基址在列表中 LDM 将通常覆盖更新的基址。

数据中止

一些合法地址可能不被存储器管理系统接受，储器管理能够指示出引起 ABORT 信号端为高的地址的错误。这可以发生在当一个多寄存器加载或存储时的任何传输，并且如果 ARM920T 用于虚拟存储器系统中必须可重入。

STM 指令中的中止

如果中止发生在存储多寄存器指令时，ARM920T 耗费微动作直到指令完成，此后其进入数据中止陷阱。存储器管理有责任预防错误的写入存储器。如果指定了回写，处理器的内部状态将只会由于基址寄存器的更改而改变，并且必须在指令重试前由软件撤消（在确定了中止的情况下）。

LDM 指令中的中止

当 ARM920T 在加载多块指令检测到一个数据中止时，它将更改指令的运行来确保重入为可用。

- 当中止发生时停止寄存器的覆盖。异常中止时加载将不会发生，但先前的可能会覆盖寄存器。PC 总为被写入的最后的寄存器，所以通常是受到保护的。
- 如果请求了回写，恢复基址寄存器被更改的值。这确保了在基址寄存器也在传输列表并且可能在数据中止前被覆盖的情况下的可重入性。

当完成加载多块数时发生据中止陷阱，并且系统软件在重新启动指令前未对更改的基址有任何动作（确定中止的情况下）。

指令周期时间

正常执行 LDM 指令耗时 $nS+1N+1I$ 周期，执行 LDM PC 耗时 $(n+1)S + 2N + 1I$ 增加周期。此处 S, N 和 I 分别是定义连续周期 (S-cycle)，非连续周期 (N-cycle) 和内部周期 (I-cycle)。执行 STRM 指令耗时 $(n-1)2S+2N$ 增加周期，其中 n 为传输的字数。

汇编语法

<LDM|STM>{cond}<FD|ED|FA|EA|IA|IB|DA|DB> Rn{!},<Rlist>{^}

其中

{cond}	两个字符条件助记符，如表 3-2 所示。
Rn	表达式计算到一个有效寄存器中
<Rlist>	寄存器列表，用 {} 圈定寄存器范围 (如 {R0,R2-R7,R10})
{!}	如果需要回写则出现 (W=1)，否则 W=0
{^}	如果出现则置位 S 位，按顺序加载带 PC 的 CPSR，或当特权模式强制用户组的传输。

寻址模式名称

为是否指令使用堆栈支持或其他用途，每种寻址模式有不同的汇编助记符。名称与指令中的位的值对应关系如下表 3-6 所示。

表 3-6. 寻址模式名称

名 称	堆 栈	其 他	L 位	P 位	U 位
前递增加载	LDMED	LDMIB	1	1	1
后递增加载	LDMFD	LDMIA	1	0	1
前递减加载	LDMEA	LDMDB	1	1	0
后递减加载	LDMFA	LDMDA	1	0	0
前递增存储	STMFA	STMIB	0	1	1
后递增存储	STMEA	STMIA	0	0	1
前递减存储	STMFD	STMDB	0	1	0
后递减存储	STMED	STMDA	0	0	0

FD, ED, FA, EA 是定义由必须的堆栈结构决定的前/后变址和递增/减位。F 和 E 关联是“满”或“空”堆栈，即前寻址是否在在存储到堆栈前被执行(满)。A 和 D 关联是堆栈为递增还是递减。如果为递增，STM 将会递增 LDM 会递减，如果为递减，反之亦然。

IA, IB, DA, DB 允许控制 LDM/STM 不用于堆栈，简单的意为后递增，前递增，后递减，前递减。

例子

LDMFD	SP!, {R0, R1, R2}	:3 个寄存器出栈
STMIA	R0, {R0- R15}	:保存所有寄存器
LDMFD	SP!, {R15}	:R15←(SP), CPSR 不改变
LDMFD	SP!, {R15}^	:R15←(SP), CPSR←SPSR_<模式> (容许在特权模式)
STMFD	R13, {R0- R14}^	:保存用户模式堆栈上的寄存器 (容许在特权模式)

这些指令可能用于进入子程序时的状态保存，并在常规调用有效的返回时恢复：

STMED	SP!, {R0- R3, R14}	:保存用于工作区域的 R0 到 R3 和用于返回的 R14
BL	somewhere	:此嵌套调用将会覆盖 R14
LDMED	SP!, {R0- R3, R15}	:恢复工作区域并返回

单数据交换 (SWP)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-23 所示。

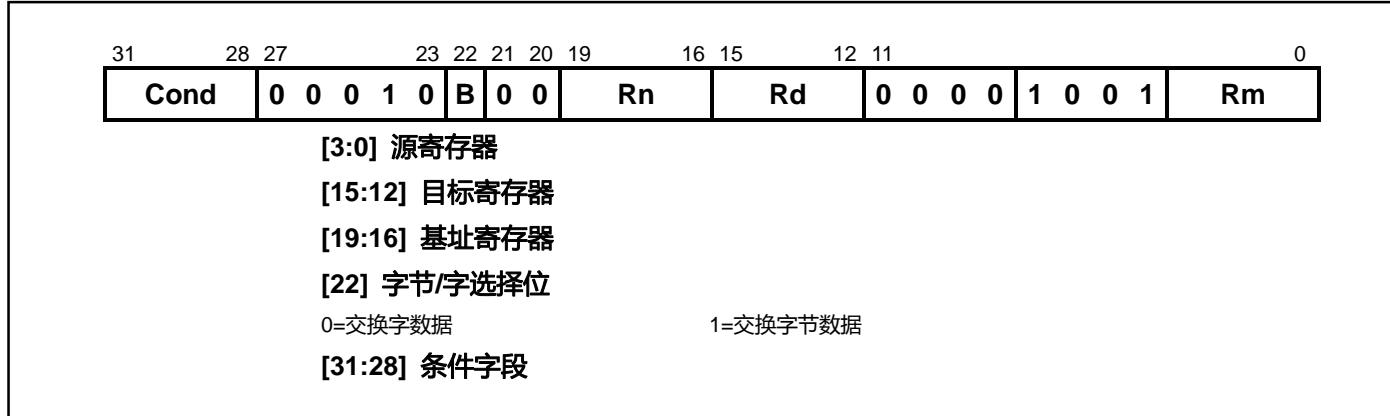


图 3-23. 交换指令

数据交换指令用于寄存器与外部存储器之间的单字节或单字数据交换。执行这条指令就如同存储器读取接着与存储器写入被“锁”在一起（处理器不能被打断直至操作完成，存储器管理器被警告对待其为不能分开的）。这种指令类型对于实现软件信号量特别有用。

由基址寄存器 (Rn) 的内容确定交换地址。处理器首先读取交换地址的内容，然后将源寄存器 (Rm) 的内容写入到交换地址中去，并将旧的存储器内容存储到目标寄存器 (Rd)。源寄存器和目标寄存器可能会指定相同的寄存器。

在读取与写入操作期间 **LOCK** 输出信号端变为高向外部存储器管理器表明它们为锁在一起的，并容许其无中断地完成。这对多处理器系统非常重要，只有交换指令是不可分的指令，可能会用于执行信号量；在处理器执行一个锁存操作时存储器的控制必须不能离开。

字节和字

这种指令类型可能用于交换 ARM920T 寄存器与存储器之间的一个字节 (B=1) 或一个字 (B=0)。实现 SWP 指令如同 LDR 接着 STR，并且描述此处功能在单数据传输章节中。特别是大小端配制的描述同样适用于 SWP 指令。

R15 的使用

在 SWP 指令中不要将 R15 用作操作数 (Rd, Rn 或 Rs)。

数据中止

如果用于交换的地址对于存储器管理器系统不可用，存储器管理器可以驱动 ABORT 为高来标志问题。在读取或写入周期期间都可以发生，并引起数据中止陷阱。由系统软件来决定问题的情况，然后指令能够从新启动并且继续原来的程序。

指令周期时间

执行 SWP 指令耗时 $1S+2N+1I$ 增加周期 , 此处 S , N 和 I 分别是定义连续周期(S-cycle), 非连续周期(N-cycle) 和内部周期 (I-cycle) 。

汇编语法

<SWP>{cond}{B} Rd,Rm,[Rn]

{cond} 两个字符条件助记符 , 如表 3-2 所示。

{B} 如果出现 B 则按字节传输 , 否则按字传输

Rd, Rm, Rn 表达式计算到一个有效寄存器中

例子

SWP R0, R1, [R2] ;按字寻址 R2 加载 R0 , 并存储 R1 到 R2 的地址中

SWPB R2, R3, [R4] ;按字节寻址 R4 加载 R2 , 并存储 R3 的位[7:0]到 R2 的地址中

SWPEQ R0, R0, [R1] ;按字寻址 R1 条件交换 R0

软件中断 (SWI)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-24 所示。

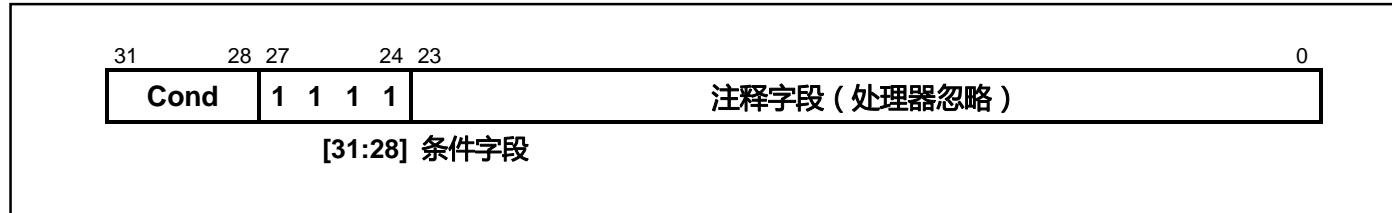


图 3-24. 软件中断指令

软件中断指令用于在受限制方式下进入管理模式。指令引起软件中断陷阱，并影响模式的改变。接着 PC 强制固定一个值 (0x08) 并且 CPSR 保存到 SPSR_svc。如果从用户更改的 SWI 向量地址相应被保护 (由外部存储器管理器硬件)，一个完全保护操作系统可能被构建

从管理模式返回

进入软件中断陷阱时保存 PC 到 R14_svc 中，SWI 指令后调整 PC 指向字。MOVS PC,R14_svc 将返回调用程序并恢复 CPSR。

注意链接机制是不可重入的，所以如果管理代码希望对自身使用软件中断，必须首先保存返回地址和 SPSR 的备份。

注释字段

指令的低 24 位被处理器忽略，可能用于传递信息给管理者代码。例如，管理者可能注意该字段并使用它作为引导进入执行各种管理者函数的程序进入点的阵列。

指令周期时间

执行软件中断指令耗时 $2S+1N$ 增加周期 , 此处 S 和 N 分别是定义连续周期(S-cycle)和非连续周期(N-cycle)。

汇编语法

SWI{cond} <expression>

{cond} 两个字符条件助记符 , 如表 3-2 所示。
<expression> 计算并放置在注释字段中(被 ARM920T 忽略)

例子

SWI	ReadC	;从读取流中获取下一个字符
SWI	WriteI+"k"	;在写入流中输出一个"K"
SWINE	0	;条件调用 , 命令字段中管理者为 0

管理者代码

之前的例子假定存在适当的管理者代码 , 例如 :

0x08	B Supervisor	;SWI 进入点
DCD	ZeroRtn	;管理者程序地址
DCD	ReadCRtn	;
DCD	WriteIRtn	;
.....		;
Zero	EQU 0	;
ReadC	EQU 256	;
WriteI	EQU 512	;
Supervisor		;SWI 必须放置程序在位[23:8] , 数据(如果有)在位[7:0]。假设 R13_svc 指向相应堆栈
STMFD	R13, {R0- R2, R14}	;保存工作寄存器和返回地址
LDR	R0, [R14, #4]	;获取 SWI 指令
BIC	R0, R0, #0xFF000000	;获取高 8 位
MOV	R1, R0, LSR#8	;获取程序偏移
ADR	R2, EntryTable	;获取进入列表的起始地址
LDR	R15, [R2, R1, LSL#2]	;分支跳转到相应程序
WriteIRtn		;带 R0 的位[7:0]的字符进入
.....		;
LDMFD	R13, {R0- R2, R15}^	;恢复工作区并返回 , 恢复处理器模式和标志位

协处理器数据操作 (CDP)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-25 所示。

这种指令类型是用于告知协处理器执行一些内部操作。没有通讯返回结果给 ARM920T，也不会等待操作完成。协处理器可以控制指令等待执行队列，并且其执行能够重叠其他作业，允许协处理器和 ARM920T 并行独立运行。

协处理器指令

S3C2440A 不像其它基于 ARM 核处理器，没有外部协处理器接口。同样也没有片上协处理器。

所以所有协处理器指令将会由 S3C2440A 引起未定义指令陷阱。这些协处理器指令可以由未定义陷阱处理程序仿真。虽然外部协处理器不能连接到 S3C2440A，协处理器指令仍然在此完整的描述。（记住任何在此章节中描述的外部协处理器都为软件仿真。）

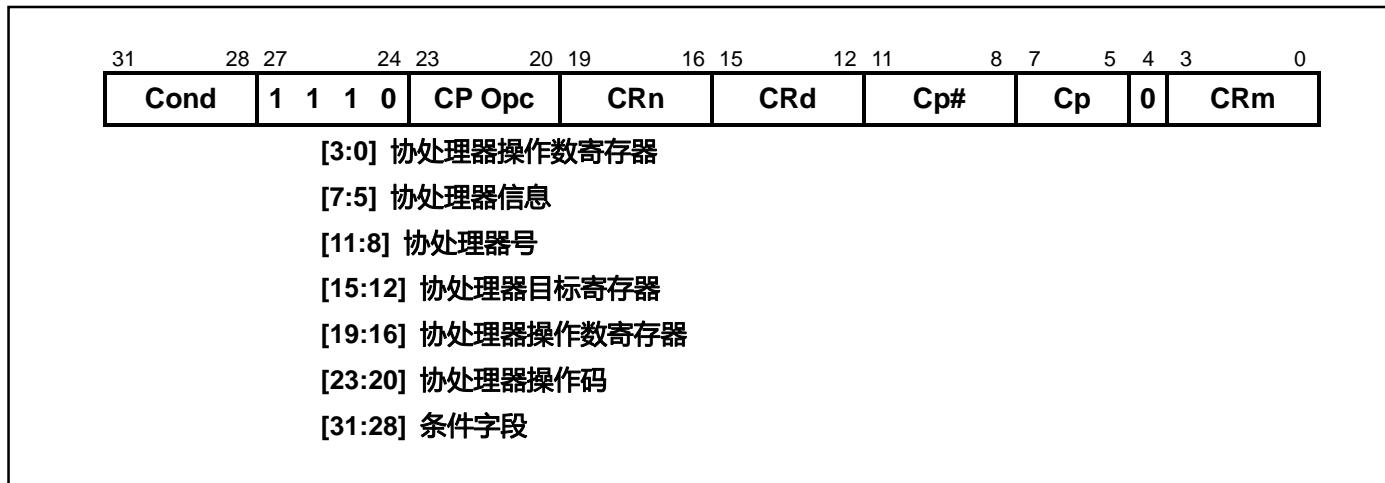


图 3-25. 协处理器数据操作指令

只有协处理器字段的位[4]和位[31:24]才对 ARM920T 有意义。其余位用于协处理器。中止字段名称用于协议，除了 CP# 专用，所有字段的使用都可能由特定协处理器重新定义。CP# 字段用于为每个协处理器赋值一个标识号（0 到 15 之间），协处理器将会忽略 CP# 字段中不包含其标识号的任何指令。

该指令的一般解释为协处理器应当执行一个在 CP Opc 字段（可能在 CP 字段中）中指定对 CRn 和 CRm 内容的操作，将结果放置在 CRd 中。

指令周期时间

执行协处理器数据操作指令耗时 $1S+bI$ 增加周期， b 为协处理器等待忙循环的耗费周期数。此处 S 和 I 分别是定义连续周期（S-cycle）和内部周期（N-cycle）。

汇编语法

CDP{cond} p#,<expression1>,cd,cn,cm{,<expression2>}

{cond} 两个字符条件助记符，如表 3-2 所示。

p# 协处理器必须的独有识别号

<expression1> 计算一个常量并放置在 CP Opc 字段中

cd, cn, cm 分别为有效协处理器寄存器 CRd, CRn 和 CRm 计算表达式

<expression2> 此字段出现，独有并放置在 CP 字段中的计算表达式

例子

CDP p1, 10, c1, c2, c3 ;请求协处理器 1 用 CR2 和 CR3 运行操作 10，结果放置在 CR1 中

CDPEQ p2, 5, c1, c2, c3, 2 ;如果置位了 Z 标志位请求协处理器 2 用 CR2 和 CR3 运行操作 5（类型 2），结果放置在 CR1 中

协处理器数据传送 (LDC , STC)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-26 所示。

这种指令类型是用于直接加载 (LDC) 或存储 (STC) 一个协处理器寄存器的子集到存储器中。ARM920T 有责任提供存储器地址和协处理器存放或访问的数据并控制传输的字数。

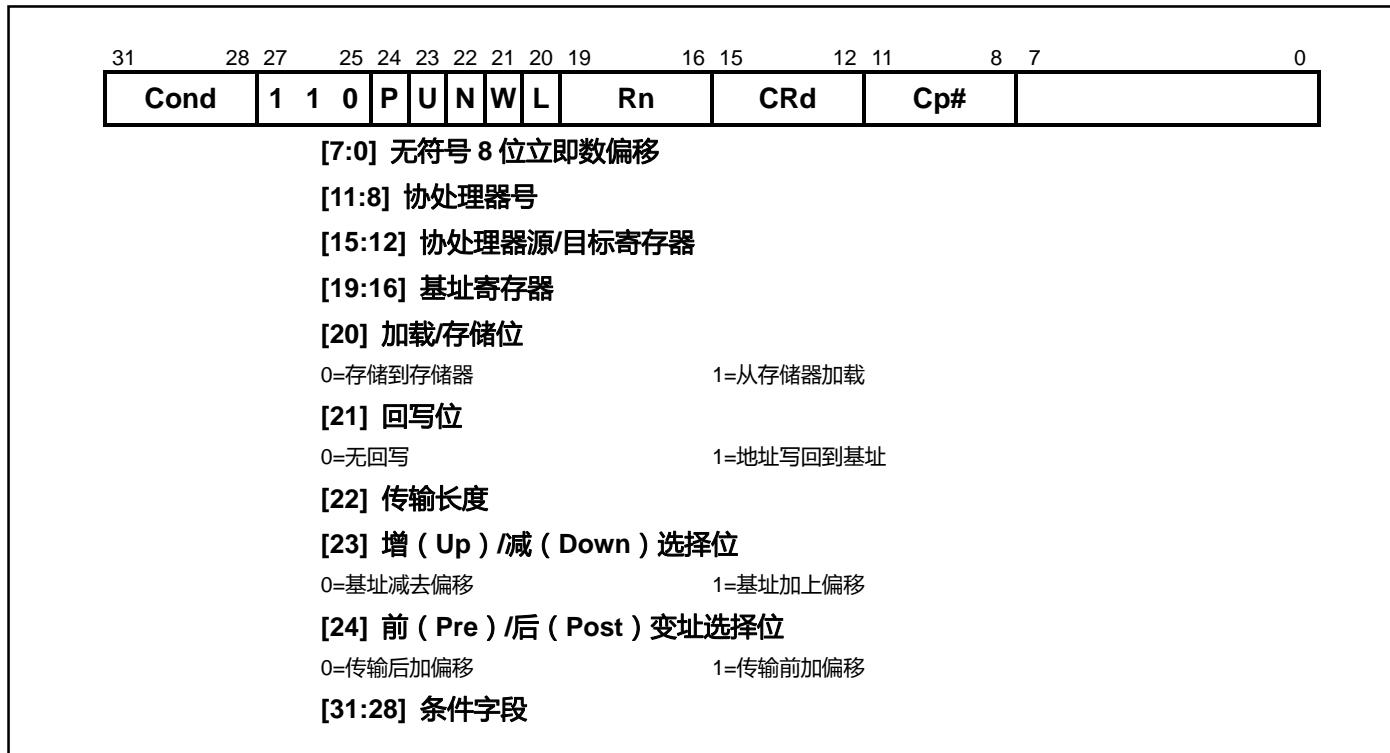


图 3-26. 协处理器数据传输指令

协处理器字段

CP#字段用于识别提供或接收数据所需的协处理器，只有此字段内容标识号对应的协处理器才应答。

CRd 字段和 N 位包含了可能由于协处理器的不同而解释为不同的协处理器的信息，但按照协议 CRd 作为被传输的寄存器（或当传输多于一个寄存器的传输时作为第一个寄存器），N 位用于选择两种传输长度选择项的其中一个。例如 N=0 选择单寄存器传输，N=1 为上下文切换选择所有寄存器传输。

寻址模式

ARM920T 有责任提供用于传输的存储器系统的地址，可见的寻址模式为用于单数据传输指令的子集。注意，虽然协处理器数据传输的立即数偏移为 8 位宽度，指定字偏移，但是单数据传输中却为 12 位宽度，指定字节偏移。

8 位无符号立即数偏移为左移 2 位并从基址寄存器 (Rn) 加上 (U=1) 或减去 (U=0)；此计算可能在基址作为传输地址前 (P=1) 或后 (P=0) 执行。被更改的基址值可能被写回到基址中 (W=1)，或保留旧基址值 (W=0)。注意后变址寻址模式需要明确置位 W 位，不像 LDR 和 STR 后变址寻址会自动回写。

由前变址指令而更改的基址寄存器的值在第一个字的传输作为地址。第二个字（若多于一个的传输）将送往或来自比第一次传输高一个字（4 字节）的地址，并且地址将在随后的传输中按字递增。

地址对齐

基址应当正常的按字边界量对齐。地址的低 2 位将出现在 A[1:0]中并且可能由存储器系统解释。

R15 的使用

如果 Rn 为 R15 , 使用该值将为指令加 8 字节的地址。必须不指定基址写回到 R15。

数据中止

如果地址合法但存储器管理器生成一个中止将会引起数据陷阱。改变基址的回写将会发生，但所有其他处理器状态将被保护。协处理器有部分责任在确定发生中止的原因后确保数据传输能够被重新启动，并确保任何后续动作，当指令重试时能重复其保证。

指令周期时间

执行协处理器数据传输指令耗时(n-1)S+2N+bI 增加周期 , 其中 n 为传输的字数据数量 , b 为协处理器等待忙循环的耗费周期数。此处 S , N 和 I 分别是定义连续周期 (S-cycle) , 非连续周期 (N-cycle) 和内部周期 (I-cycle)。

汇编语法

<LDC|STC>{cond}{L} p#,cd,<Address>

LDR 从存储器加载到协处理器

STR 从协处理器存储到存储器

{L} 出现时 , 处理为长传输 (N=1) , 否则处理为短传输 (N=1)

{cond} 两个字符条件助记符 , 如表 3-2 所示。

p# 协处理器必须的独有识别号

cd 放置在 CRd 字段的有效协处理器寄存器计算表达式

<address> 可以为 :

生成地址的表达式 :

1 汇编将会试图生成一个用于 PC 为基址指令和由计算表达式得到的调整立即数偏移寻址。这将为一个相对 PC 的前变址寻址。如果寻址超出范围 , 将产生错误。

前变址寻址说明 :

2 [Rn] 零偏移

[Rn,<#expression>]{!} 偏移<expression>字节

后变址寻址说明 :

3 [Rn],<#expression> 偏移<expression>字节

{!} 如果出现! , 写回到基址寄存器 (置位 W 位)

Rn 有效 ARM920T 寄存器计算表达式

注意 :

若 Rn 为 R15, 汇编将会从偏移值中减去 8 以允许 ARM920T 流水线操作。

例子

LDC	p1, c2, table	;从地址列表加载到协处理器 1 的 c2 , 使用相对 PC 的地址
STCEQL	p2, c3, [R5, #24]!	;条件将协处理器 2 的 c3 存储到 R5 中地址的高 24 字节中 , 写回此地址到 R5 , 使用长 ;传输操作 (可能存储多字)

注意 :

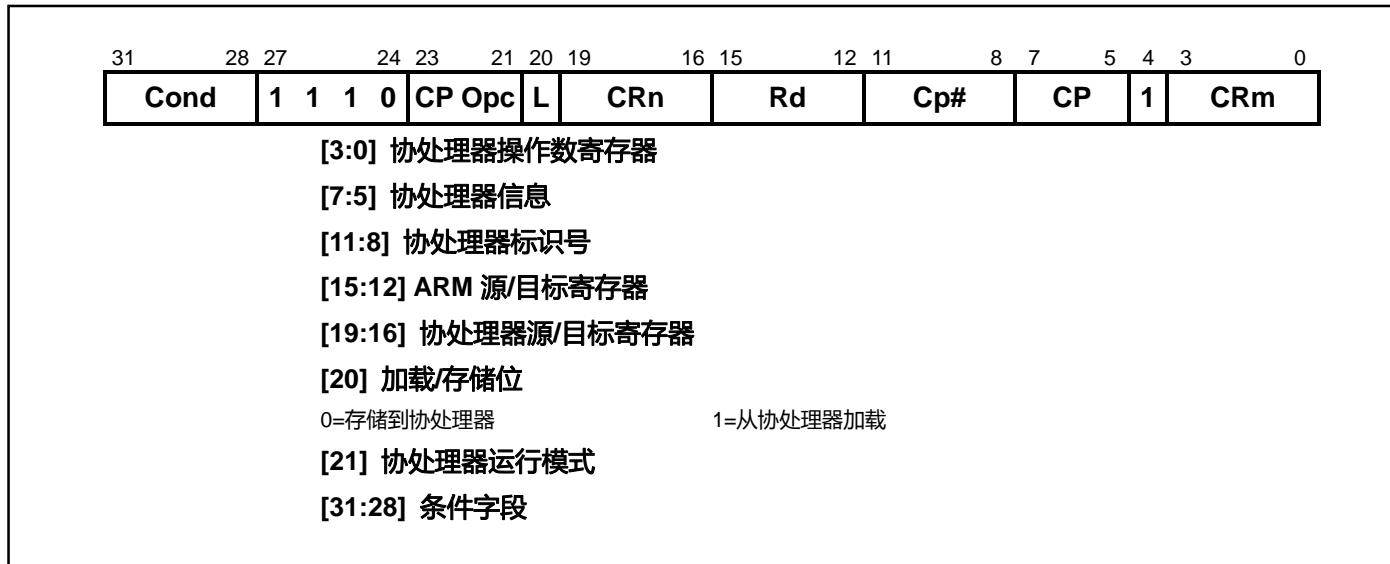
尽管地址偏移为字节的表达式 , 但指令偏移字段为字。汇编将相应调整偏移量。

协处理器寄存器传送 (MRC , MCR)

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-27 所示。

这种指令类型是用于 ARM920T 与协处理器直接信息通讯。例如协处理器到 ARM920T 的寄存器传输 (MRC) 指令在协处理器中的浮点数的 FIX，在协处理器中改变浮点数为一个 32 位整型，传输结果到 ARM920T 寄存器中。在协处理器中的 ARM920T 寄存器中 32 位数改变为浮点数的 FLOAT 表明 ARM920T 寄存器到协处理器的传输 (MCR) 使用。

该指令的重要使用是从协处理器传递控制信息到 ARM920T CPSR 标志位。例如在协处理器中比较两个浮点数的结果可以移动到 CPSR 来控制后续执行流。



协处理器字段

CP#字段用作所有协处理器指令，指定哪个协处理器被调用。

CP Opc , CRn , CP 和 CRm 字段只用于协处理器，出现在此的解释只是源自协议。当协处理器功能与此不兼容时容许其他解释。协议解释 CP Opc 和 CP 字段指定对于操作协处理器必须执行，CRn 为传输信息的协处理器源或目标寄存器，CRm 为第二个可能由于某种联系依赖指定特殊操作的协处理器寄存器。

传输到 R15

当一个协处理器寄存器传输到 ARM920T 的 R15 作为目标寄存器，传输字的位[31:28]将分别复制到 N , Z , C 和 V 标志位。传输字的其它位忽略，传输中 PC 和其他 CPSR 位不会受影响。

从 R15 传输

从 ARM920T 的 R15 作为源寄存器传输到协处理器寄存器，将存储到 PC+12。

指令周期时间

执行 MRC 指令耗时 $1S + (b+1)l + 1C$ 增加周期 此处 S , l 和 C 分别是定义连续周期(S-cycle) , 内部周期(l-cycle) 和协处理器寄存器传输周期 (C-cycle)。执行 MCR 指令耗时 $1S + bl + 1C$ 增加周期 b 为协处理器等待忙循环的耗费周期数。

汇编语法

<MCR|MRC>{cond} p#,<expression1>,Rd,cn,cm{,<expression2>}

MRC 从协处理器移动到 ARM920T 寄存器 (L=1)

MCR 从 ARM920T 移动到协处理器寄存器 (L=0)

{cond} 两个字符条件助记符 , 如表 3-2 所示。

p# 协处理器必须的独有识别号

<expression1> 放置在 CP Opc 字段中的常量表达式

Rd 有效 ARM920T 寄存器计算表达式

cn, cm 有效协处理器寄存器计算表达式 , 分别为 CRn 和 CRM

<expression2> 若出现 , 放置在 CP 字段中的常量表达式

例子

MRC	p2, 5, R3, C5, C6,	;请求协处理器 2 用 c5 和 c6 执行操作数 5 , 将结果 (有符号 32 位字) 传输回 R3
MRC	p6, 0, R4, c5, c6	;请求协处理器 6 用 R4 执行操作数 0 , 将结果放置在 c6
MRCEQ	p3, 9, R3, c5, c6 2	;条件请求协处理器 3 用 c5 和 c6 执行操作数 9 (类型 2), 将结果传输回 R3

未定义指令

该指令只在条件为真时执行。各种条件在表 3-2 中定义了。指令译码如图 3-28 所示。

31	28 27	25 24		5 4 3	0
Cond	0 1 1	x x	1	x x x x	

图 3-28. 未定义指令

然后条件为真，将引起未定义指令陷阱。

注意未定义机制涉及提供该指令给任何可能出现的协处理器，所有协处理器必须在 CPA 和 CPB 信号端为高时拒绝接受。

指令周期时间

执行该指令耗时 $1S+1I+1N$ 周期，此处 S, N 和 I 分别是定义连续周期 (S-cycle)，非连续周期 (N-cycle) 和内部周期 (I-cycle)。

汇编语法

汇编没有产生此指令的助记符。如果将来为某些特定应用而采用了，相应的助记符将会被加入汇编中。在此之前，不要使用这条指令。

指令集例子

以下例子显示了基本 ARM920T 指令组合提供的高效代码的方法。这些方法并没有节省大量执行时间的分配(尽管可能节省了一些)，大部分只精简了代码。

使用条件指令

逻辑或使用条件

CMP	<i>Rn, #p</i>	:如果 $Rn=p$ 逻辑或 $Rm=q$ ，则跳至 Label
BEQ	<i>Label</i>	
CMP	<i>Rm, #q</i>	
BEQ	<i>Label</i>	

可以替换为

CMP	<i>Rn, #p</i>	
CMPNE	<i>Rm, #q</i>	:如果不满足条件，尝试其他
BEQ	<i>Label</i>	

绝对值

TEQ	<i>Rn, #0</i>	:符号测试
RSBMI	<i>Rn, Rn, #0</i>	:如果需要，与二进制补码

乘 4, 5 或乘 6 (运行时间)

MOV	<i>Rc, Ra, LSL#2</i>	:乘 4
CMP	<i>CMP Rb, #5</i>	:测试值
ADDCS	<i>Rc, Rc, Ra</i>	:完成乘 5
ADDHI	<i>Rc, Rc, Ra</i>	:完成乘 6

集合分离和范围测试

TEQ	<i>Rc, #127</i>	:分离测试
CMPNE	<i>Rc, # "-1</i>	:范围测试
MOVLS	<i>Rc, # ""</i>	:如果 $Rc \leq 0$ 或 $Rc = \text{ASCII}(127)$ ，则 $Rc := "$

除法和余数

以 ARM 交叉开发工具包提供的 ANSI C 语言库部分形式的源代码提供的具体应用除法程序的数，可以由你自己提供。一个简短通用除法程序如下。

		:由 Ra 和 Rb 中的数进入	
	MOV	<i>Rcnt, #1</i>	:控制除法位
<i>Div1</i>	CMP	<i>Rb, #0x80000000</i>	:移动 Rb 直至大于 Ra
	CMPCC	<i>Rb, Ra</i>	
	MOVCC	<i>Rb, Rb, ASL#1</i>	
	MOVCC	<i>Rcnt, Rcnt, ASL#1</i>	
	BCC	<i>Div1</i>	
	MOV	<i>Rc, #0</i>	
<i>Div2</i>	CMP	<i>Ra, Rb</i>	:可减测试
	SUBCS	<i>Ra, Ra, Rb</i>	:若可以则减
	ADDCS	<i>Rc, Rc, Rcnt</i>	:放置相关位至结果
	MOVS	<i>Rcnt, Rcnt, LSR#1</i>	:移位控制位
	MOVNE	<i>Rb, Rb, LSR#1</i>	:除非结束，否则减半
	BNE	<i>Div2</i>	:整除结果放在 Rc，余数放在 Ra

ARM920T 溢出检测

1. 32 位结果无符号乘法的溢出

UMULL	<i>Rd, Rt, Rm, Rn</i>	:3 至 6 周期
TEQ	<i>Rt, #0</i>	:加一个周期和一个寄存器
BNE	<i>overflow</i>	

2. 32 位结果有符号乘法的溢出

SMULL	<i>Rd, Rt, Rm, Rn</i>	:3 至 6 周期
TEQ	<i>Rt, Rd ASR#31</i>	:加一个周期和一个寄存器
BNE	<i>overflow</i>	

3. 32 位结果无符号乘加的溢出

UMLAL	<i>Rd, Rt, Rm, Rn</i>	:4 至 7 周期
TEQ	<i>Rt, #0</i>	:加一个周期和一个寄存器
BNE	<i>overflow</i>	

4. 32 位结果有符号乘加的溢出

SMULL	<i>Rd, Rt, Rm, Rn</i>	:4 至 7 周期
TEQ	<i>Rt, Rd ASR#31</i>	:加一个周期和一个寄存器
BNE	<i>overflow</i>	

5. 64 位结果无符号乘加的溢出

UMULL	<i>Rl, Rh, Rm, Rn</i>	:3 至 6 周期
ADDS	<i>Rl, Rl, Ra1</i>	:低位加法
ADC	<i>Rh, Rh, Ra2</i>	:高位加法
BCS	<i>overflow</i>	:一个周期和两个寄存器

6. 64 位结果有符号乘加的溢出

SMULL	<i>Rl, Rh, Rm, Rn</i>	:3 至 6 周期
ADDS	<i>Rl, Rl, Ra1</i>	:低位加法
ADC	<i>Rh, Rh, Ra2</i>	:高位加法
BCS	<i>overflow</i>	:一个周期和两个寄存器

注意：

溢出检测不适合做无符号和有符号 64 位结果的乘法，因为在这种计算中不会发生溢出。

伪随机二进制序列发生器

经常需要产生（伪）随机数，最有效的算法是基于带专用逻辑或反馈的移位发生器，一定程度上类似循环冗余校验发生器。不幸的是 32 位发生器的次序最大长度需要多于一个反馈信号（*tap*）（即重复前 $2^{32}-1$ 个周期），所以此例子使用了带在位[33]和位[20]的反馈信号的 33 位寄存器。基本算法为新位:=位[33]逻辑异或位[20]，左移 33 位数，新位放置在最低位上；执行此操作是为全部新位的需要（即 32 位）。所有操作可以在 5S 周期内完成：

TST	<i>Rb, Rb, LSR#1</i>	:发送 Ra (32 位), Rb (1 位在 Rb 的 LSB)，使用 Rc 进入
MOVS	<i>Rc, Ra, RRX</i>	:最高位装入进位
ADC	<i>Rb, Rb, Rb,</i>	:33 位循环右移
EOR	<i>Rc, Rc, Ra, LSL#12</i>	:进位装入 Rb 的 LSB
EOR	<i>Ra, Rc, Rc, LSR#20</i>	; (涉和 !)
		; (同样涉和 !) 如同之前，新值送入 Ra, Rb

使用筒形移位器的常量乘法

2^n (1 , 2 , 4 , 8 , 16 , 32...) 的乘法

MOV *Ra, Rb, LSL #n*

2^{n+1} (3 , 5 , 9 , 17...) 的乘法

ADD *Ra, Ra, Ra, LSL #n*

2^{n-1} (3 , 7 , 15...) 的乘法

RSB *Ra, Ra, Ra, LSL #n*

乘 6

ADD *Ra, Ra, Ra, LSL #1* ;乘 3

MOV *Ra, Ra, LSL#1* ;再乘 2

乘 10 并加上附加值

ADD *Ra, Ra, Ra, LSL #2* ;乘 5

ADD *Ra, Rc, Ra, LSL #1* ;再乘 2 并加上附加值

*Rb := Ra*C* 的普通递归方式 , C 为常量 :

1. 如果 C 为偶数 , 表明 $C=2^n*D$, D 为奇数 :

D=1: **MOV** *Rb, Ra, LSL #n*

D<>1: {*Rb := Ra*D*}

MOV *Rb, Rb, LSL #n*

2. 如果 C 对 4 取余等于 1 , 表明 $C=2^n*D+1$, D 为奇数 , $n>1$:

D=1: **ADD** *Rb, Ra, Ra, LSL #n*

D<>1: {*Rb := Ra*D*}

ADD *Rb, Ra, Rb, LSL #n*

3. 如果 C 对 4 取余等于 3 , 表明 $C=2^n*D-1$, D 为奇数 , $n>1$:

D=1: **RSB** *Rb, Ra, Ra, LSL #n*

D<>1: {*Rb := Ra*D*}

RSB *Rb, Ra, Rb, LSL #n*

这并不是最佳的 , 但接近。一个非最佳的例子 , 乘 45 作为 :

RSB *Rb, Ra, Ra, LSL#2* ;乘 3

RSB *Rb, Ra, Rb, LSL#2* ;乘 $4^*3-1=11$

ADD *Rb, Ra, Rb, LSL#2* ;乘 $4^*11+1=45$

相当于

ADD *Rb, Ra, Ra, LSL#3* ;乘 9

ADD *Rb, Ra, Rb, LSL#2* ;乘 $5^*9=45$

未知对齐的加载字

;使用 Rb 和 Rc 进入 Ra (32 位) 内的地址 , 结果放入 Rd。注意 d 必须小于 c 如 0 , 1

BIC *Rb, Ra, #3* ;获取字对齐地址

LDMIA *Rb, {Rd, Rc}* ;获取 64 位包括应答

AND *Rb, Ra, #3* ;字节更改乘数

MOVS *Rb, Rb, LSL#3* ;...接位测试是否对齐

MOVNE *Rd, Rd, LSR Rb* ;产生结果字的低位 (若未对齐)

RSBNE *Rb, Rb, #32* ;获取其他移位量

ORRNE *Rd, Rd, Rc, LSL Rb* ;组合两半得到结果

4 THUMB 指令集

Thumb 指令集格式

Thumb 指令集为 16 位 ARM 指令集 (32 位格式) 的版本。ARM 指令集简化到 16 位版本。Thumb 指令损失了 ARM 指令集的多功能。从 ARM 指令精简 Thumb 指令是由 ARM920T 内核的 Thumb 精简装置完成的。

Thumb 指令是由 ARM 指令精简得到的，Thumb 指令为 16 位格式指令并受到一些限制。16 位格式的限制完全通告使用的 Thumb 指令。

格式概述

Thumb 指令集格式如下图所示。

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
1	0	0	0	Op		Offset5					Rs		Rd				移动移位寄存器			
2	0	0	0	1	1	I	Op	Rn/offset3			Rs		Rd				加法/减法			
3	0	0	1	Op		Rd				Offset8								移动/比较/加/减立即数		
4	0	1	0	0	0	0	Op				Rs		Rd				ALU 操作			
5	0	1	0	0	0	1	Op	H1	H2	Rs/Hs		Rd/Hd				高寄存器操作/分支跳转交换				
6	0	1	0	0	1	Rd			Word8								加载相对 PC			
7	0	1	0	1	L	B	0	Ro			Rb		Rd				带寄存器偏移的加载/存储			
8	0	1	0	1	H	S	1	Ro			Rb		Rd				加载/存储符号扩展字节/半字			
9	0	1	1	B	L	Offset5					Rb		Rd				带立即数偏移的加载/存储			
10	1	0	0	0	L	Offset5					Rb		Rd				加载/存储半字			
11	1	0	0	1	L	Rd			Word8								加载/存储相对 SP			
12	1	0	1	0	SP	Rd			Word8								地址加载			
13	1	0	1	1	0	0	0	0	S	SWord7								堆栈指针加偏移		
14	1	0	1	1	L	1	0	R	Rlist										出栈/入栈寄存器	
15	1	1	0	0	L	Rb			Rlist										加载/存储多块	
16	1	1	0	1	Cond				Softset8										条件分支跳转	
17	1	1	0	1	1	1	1	1	Value8										软件中断	
18	1	1	1	0	0	Offset11													无条件分支跳转	
19	1	1	1	0	H	Offset														带链接长分支跳转
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

图 4-1. Thumb 指令集格式

操作码概述

下表总结了 Thumb 指令集。关于特定指令的进一步信息请查阅相关章节

表 4-1. Thumb 指令集

助记符	指令	低寄存器操作数	高寄存器操作数	条件码设置
ADC	带进位加法	Y	-	Y
ADD	加法	Y	-	Y ⁽¹⁾
AND	逻辑与	Y	-	Y
ASR	算术右移	Y	-	Y
B	无条件分支跳转	Y	-	-
Bxx	条件分支跳转	Y	-	-
BIC	位清零	Y	-	-
BL	带链接分支跳转	-	-	-
BX	分支和状态切换跳转	Y	Y	-
CMN	负数比较	Y	-	Y
CMP	比较	Y	Y	Y
EOR	逻辑异或	Y	-	Y
LDMIA	加载多块	Y	-	-
LDR	加载字	Y	-	-
LDRB	加载字节	Y	-	-
LDRH	加载半字	Y	-	-
LSL	逻辑左移	Y	-	-
LDSB	加载符号扩展字节	Y	-	-
LDSH	加载符号扩展半字	Y	-	-
LSR	逻辑右移	Y	-	Y
MOV	移动寄存器	Y	Y	Y ⁽²⁾
MUL	乘法	Y	-	Y
MVN	移动寄存器取反后的数据	Y	-	Y
NEG	取反	Y	-	Y
ORR	逻辑或	Y	-	Y
POP	寄存器出栈	Y	-	-
PUSH	寄存器入栈	Y	-	-
ROR	循环右移	Y	-	Y
SBC	带进位减法	Y	-	Y
STMIA	存储多块	Y	-	-
STR	存储字	Y	-	-
STRB	存储字节	Y	-	-
STRH	存储半字	Y	-	-
SWI	软件中断	-	-	-
SUB	减法	Y	-	Y
TST	位测试	Y	-	Y

注释：1. 此指令格式 5, 12 和 13 版本不影响条件码

2. 此指令格式 5 版本不影响条件码

格式 1：移动移位寄存器

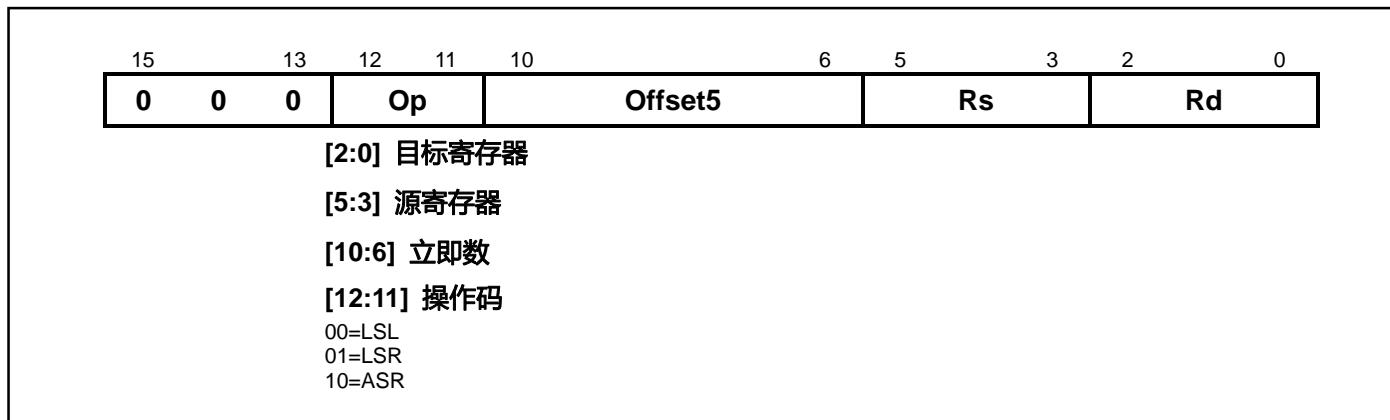


图 4-2. 格式 1

操作

这些指令移动一个低寄存器之间的移位数。Thumb 汇编语法如表 4-2 所示。

注释：

在此组的所有指令设置 CPSR 条件码。

表 4-2. 格式 1 指令概述

OP	Thumb 汇编	对应 ARM 汇编	描述
00	LSL Rd, Rs, #Offset5	MOVS Rd, Rs, LSL #Offset5	由 5 位立即数执行逻辑左移 Rs, 结果存储到 Rd
01	LSR Rd, Rs, #Offset5	MOVS Rd, Rs, LSR #Offset5	由 5 位立即数执行逻辑右移 Rs, 结果存储到 Rd
10	ASR Rd, Rs, #Offset5	MOVS Rd, Rs, ASR #Offset5	由 5 位立即数执行算术右移 Rs, 结果存储到 Rd

指令周期时间

此格式的所有指令都与表 4-2 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

LSR R2, R5, #27 ;R5 的内容逻辑右移 27 并将结果存储到 R2 中。结果设置条件码

格式 2：加法/减法

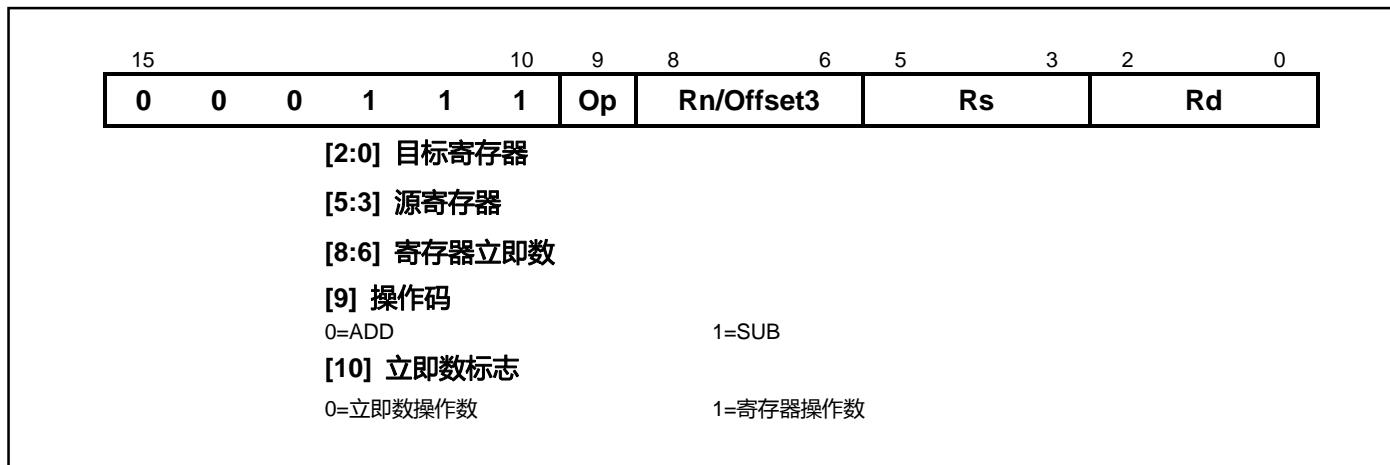


图 4-3. 格式 2

操作

这些指令允许低寄存器加上或减去一个低寄存器的内容后 3 位立即数。Thumb 汇编语法如表 4-3 所示。

注释：

在此组的所有指令设置 CPSR 条件码。

表 4-3. 格式 2 指令概述

OP	I	Thumb 汇编	对应 ARM 汇编	描述
0	0	ADD Rd, Rs, Rn	ADDS Rd, Rs, Rn	Rs 的内容加上 Rn 的内容。结果放到 Rd 中
0	1	ADD Rd, Rs, #Offset3	ADDS Rd, Rs, #Offset3	Rs 的内容加上 3 位立即数。结果放到 Rd 中
1	0	SUB Rd, Rs, Rn	SUBS Rd, Rs, Rn	Rs 的内容减去 Rn 的内容。结果放到 Rd 中
1	1	SUB Rd, Rs, #Offset3	SUBS Rd, Rs, #Offset3	Rs 的内容减去 3 位立即数。结果放到 Rd 中

指令周期时间

此格式的所有指令都与表 4-3 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

ADD	R0, R3, R4	;R0:=R3+R4 , 结果设置条件码
SUB	R6, R2, #6	;R6:=R2-6 , 结果设置条件码

格式 3：移动/比较/加/减立即数

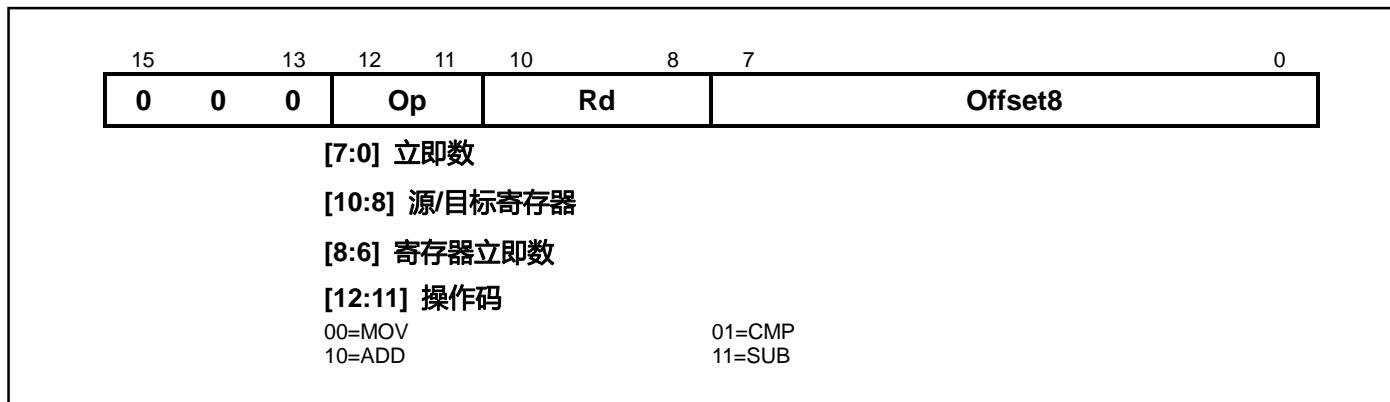


图 4-4. 格式 3

操作

此组指令执行低寄存器与 8 位立即数之间的操作。Thumb 汇编语法如表 4-4 所示。

注释：

在此组的所有指令设置 CPSR 条件码。

表 4-4. 格式 3 指令概述

OP	Thumb 汇编	对应 ARM 汇编	描述
00	MOV Rd, #Offset8	MOVS Rd, #Offset8	移动 8 位立即数到 Rd 中
01	CMP Rd, #Offset8	CMP Rd, #Offset8	比较 8 位立即数与 Rd 的内容
10	ADD Rd, #Offset8	ADDS Rd, Rd, #Offset8	Rd 的内容加上 8 位立即数，结果放到 Rd 中
11	SUB Rd, #Offset8	SUBS Rd, Rd, #Offset8	Rd 的内容减去 8 位立即数。结果放到 Rd 中

指令周期时间

此格式的所有指令都与表 4-4 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

MOV	R0, #128	;R0:=128，结果设置条件码
CMP	R2, #62	;R2-62 结果设置条件码
ADD	R1, #255	;R1:=R1+255，结果设置条件码
SUB	R6, #145	;R6:=R6-145，结果设置条件码

格式 4：ALU 操作

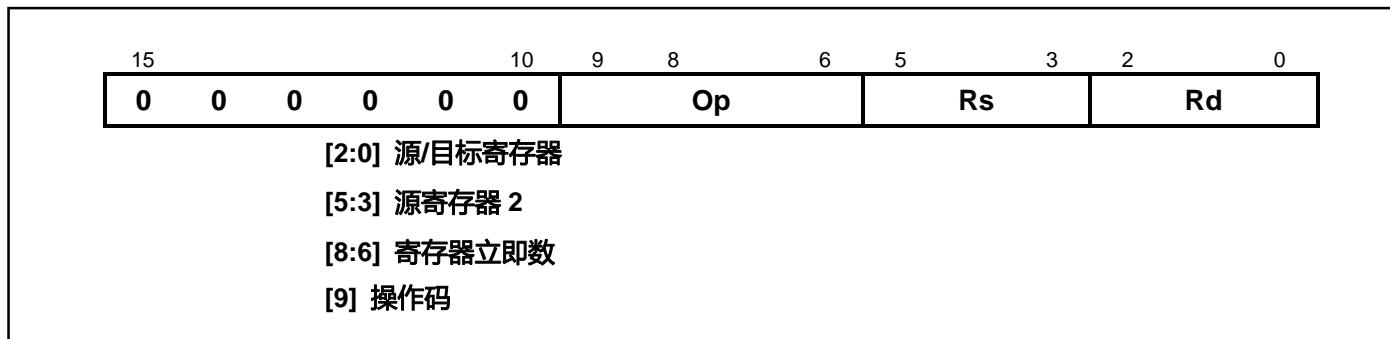


图 4-5. 格式 4

操作

以下指令使用低寄存器执行 ALU 操作。Thumb 汇编语法如表 4-5 所示。

注释：

在此组的所有指令设置 CPSR 条件码。

表 4-5. 格式 4 指令概述

OP	Thumb 汇编	对应 ARM 汇编	描述
0000	AND Rd, Rs	ANDS Rd, Rd, Rs	Rd:=Rd 逻辑与 Rs
0001	EOR Rd, Rs	EORS Rd, Rd, Rs	Rd:=Rd 逻辑异或 Rs
0010	LSL Rd, Rs	MOVS Rd, Rd, LSL Rs	Rd:=Rd 逻辑左移 Rs
0011	LSR Rd, Rs	MOVS Rd, Rd, LSR Rs	Rd:=Rd 逻辑右移 Rs
0100	ASR Rd, Rs	MOVS Rd, Rd, ASR Rs	Rd:=Rd 算术右移 Rs
0101	ADC Rd, Rs	ADCS Rd, Rd, Rs	Rd:=Rd+Rs+C 位
0110	SBC Rd, Rs	SBCS Rd, Rd, Rs	Rd:=Rd-Rs-逻辑非 C 位
0111	ROR Rd, Rs	MOVS Rd, Rd, ROR Rs	Rd:=Rd 循环右移 Rs
1000	TST Rd, Rs	TST Rd, Rs	设置 Rd 逻辑与 Rs 的条件码
1001	NEG Rd, Rs	RSBS Rd, Rs, #0	Rd=-Rs
1010	CMP Rd, Rs	CMP Rd, Rs	设置 Rd-Rs 的条件码
1011	CMN Rd, Rs	CMN Rd, Rs	设置 Rd+Rs 的条件码
1100	ORR Rd, Rs	ORRS Rd, Rd, Rs	Rd:=Rd 逻辑或 Rs
1101	MUL Rd, Rs	MULS Rd, Rs, Rd	Rd:=Rs×Rd
1110	BIC Rd, Rs	BICS Rd, Rd, Rs	Rd:=Rd 逻辑与非 Rs
1111	MVN Rd, Rs	MVNS Rd, Rs	Rd:=逻辑非 Rs

指令周期时间

此格式的所有指令都与表 4-5 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

EOR	R3, R4	;R3:=R3 逻辑 R4 , 结果设置条件码
ROR	R1, R0	;由 R0 的值循环右移 R1 , 结果存储到 R1 并设置条件码
NEG	R5, R3	;0 减去 R3 的内容 , 结果存储到 R5 , 设置 R5=- R3 条件码
CMP	R2, R6	;设置 R5=- R3 结果的条件码
MUL	R0, R7	;R0:=R7×R0 , 结果设置条件码

格式 5：加法/减法

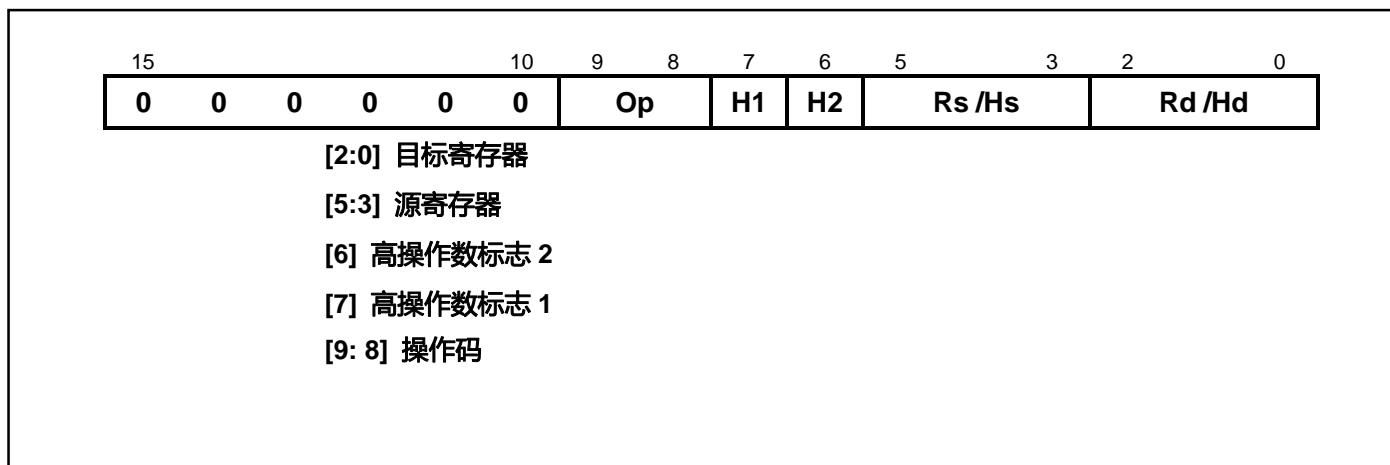


图 4-6. 格式 5

操作

该组中有 4 种指令的设置。前三个允许高低寄存器或两个高寄存器之间执行 ADD , CMP 和 MOV 操作。第四个 BX 允许执行可能用于切换处理器状态的分支跳转。Thumb 汇编语法如表 4-6 所示。

注释：

在此组只有 CMP (Op=01) 设置 CPSR 条件码。

表 4-6. 格式 5 指令概述

OP	H1	H2	Thumb 汇编	对应 ARM 汇编	描述
00	0	1	ADD Rd, Hs	ADD Rd, Rd, Hs	Rd[7:0]加上 Hs[15:8],
00	1	0	ADD Hd, Rs	ADD Hd, Hd, Rs	Hd[15:8]加上 Rs[7:0]
00	1	1	ADD Hd, Hs	ADD Hd, Hd, Hs	Hd[15:8]加上 Hs[15:8]
01	0	1	CMP Rd, Hs	CMP Rd, Hs	比较 Rd[7:0]与 Hs[15:8]。结果设置条件码
01	1	0	CMP Hd, Rs	CMP Hd, Rs	比较 Hd[15:8]与 Rs[7:0]。结果设置条件码
01	1	1	CMP Hd, Hs	CMP Hd, Hs	比较 Hd[15:8]与 Hs[15:8]。结果设置条件码
10	0	1	MOV Rd, Hs	MOV Rd, Hs	移动 Hs[15:8]的值到 Rd[7:0]
10	1	0	MOV Hd, Rs	MOV Hd, Rs	移动 Rs[7:0]的值到 Hd [15:8]
10	1	1	MOV Hd, Hs	MOV Hd, Hs	移动 Hs[15:8]的值到 Hd [15:8]
11	0	0	BX Rs	BX Rs	执行分支跳转 (可选状态改变) 到 Rs[7:0]中的地址
11	0	1	BX Hd	BX Hd	执行分支跳转 (可选状态改变) 到 Rs[15:8]中的地址

指令周期时间

此格式的所有指令都与表 4-6 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

BX 指令

BX 执行一个分支跳转到一个由高或低寄存器指定的起始地址的程序。

地址的位[0]确定进入持续的处理器状态

位[0]=0 引起处理器进入 ARM 状态

位[0]=1 引起处理器进入 Thumb 状态

注意：

如果这条指令未定义 H1=1 的功能，应当不使用。

例子

高寄存器操作

```
ADD    PC,   R5      ;PC:=PC+R5 , 不设置条件码  
SUB    R4,   R12     ;设置 R2-R12 条件码  
MOV    R15,  R14     ;移动 R14 ( LR ) 到 R15 ( PC ), 不设置条件码 , 如从子程序返回
```

分支和状态切换跳转

```
ADR    R1,   outofTHUMB  ;从 Thumb 状态切换到 ARM 状态  
MOV    R11,  R1        ;加载 outofTHUMB 的地址到 R1  
BX    R11             ;传输 R11 的内容到 PC。R11 的位[0]确定是进入 ARM 还是 Thumb 状态 , 假设此处  
                  ;为 ARM 状态  
ALIGN  
CODE32  
outofTHUMB          ;正在处理的 ARM 指令...
```

R15 用作操作数

如果 R15 作为操作数使用 , 该值将为指令 + 4 地址 , 清零位[0]。执行在非字对齐地址 Thumb 状态下的 BX PC , 其结果将无法预测。

格式 6：相对 PC 加载

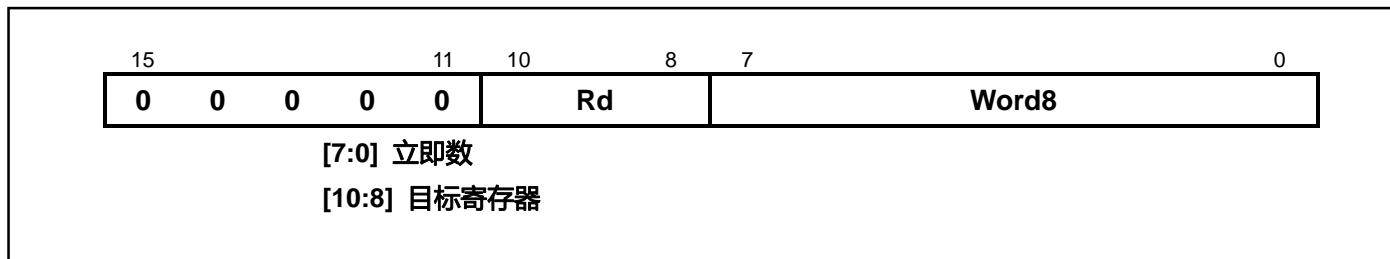


图 4-7. 格式 6

操作

该指令加载一个由 10 位立即数指定的相对 PC 偏移的地址。Thumb 汇编语法如表 4-7 所示。

表 4-7. 格式 6 指令概述

Thumb 汇编	对应 ARM 汇编	描述
LDR Rd, [PC, #Imm]	LDR Rd, [R15, #Imm]	当前 PC 值加上 Imm 中的无符号偏移量 (255 字节, 1020 字节) 加载结果地址字到 Rd 中

注释：

#Imm 指定的值是一个完整的 10 位地址，但必须按字对齐(位[1:0]位 0)，因为汇编会将#Imm>>2 放置到 Word8 字段中。PC 的值将比这条指令多 4 字节，但 PC 的位[0]强制为 0 以确保按字对齐。

指令周期时间

此格式的所有指令都与表 4-7 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

LDR R3, [PC, #844] ;加载 PC 加上 844 得到的地址到 R3 中。PC 的位[1]强制为 0。注意 Thumb 操作码
;将按 (contain) 211 作为 Word8 值

格式 7：带寄存器偏移的加载/存储

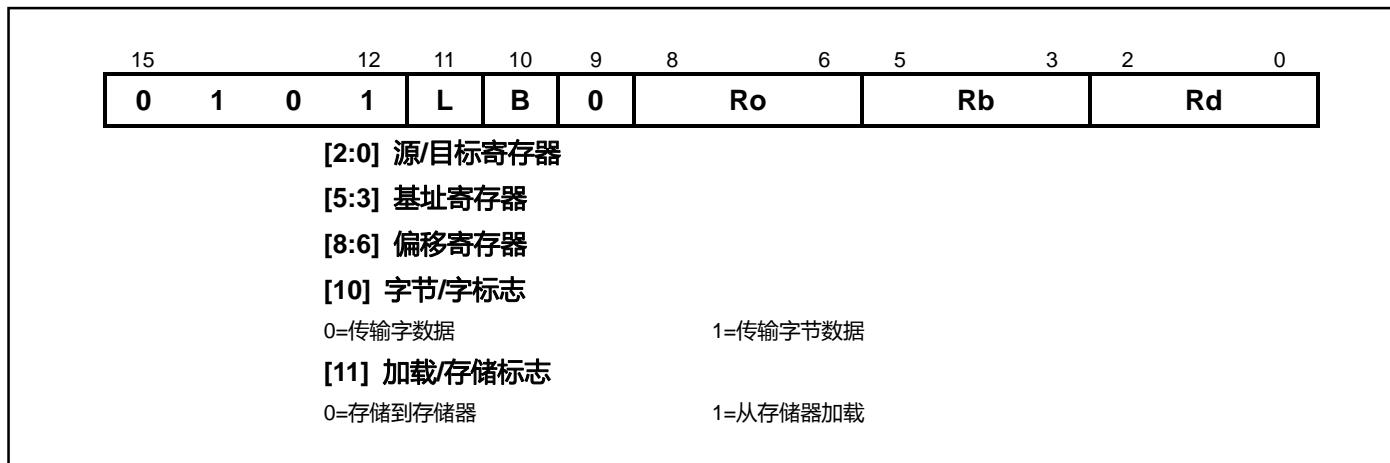


图 4-8. 格式 7

操作

这些指令在寄存器与存储器之间传输字或字节值。用一个寄存器[7:0]的前变址寻址存储器地址。Thumb 汇编语法如表 4-8 所示。

表 4-8. 格式 7 指令概述

L	B	Thumb 汇编	对应 ARM 汇编	描述
0	0	STR Rd, [Rb, Ro]	STR Rd, [Rb, Ro]	前变址字存储 :计算 Rb 与 Ro 的值之和为目标地址。 存储 Rd 的内容到结果地址
0	1	STRB Rd, [Rb, Ro]	STRB Rd, [Rb, Ro]	前变址字节存储 :计算 Rb 与 Ro 的值之和为目标地址。 存储 Rd 的内容到结果地址
1	0	LDR Rd, [Rb, Ro]	LDR Rd, [Rb, Ro]	前变址字加载 :计算 Rb 与 Ro 的值之和为源地址。 加载结果地址的内容到 Rd 中
1	1	LDRB Rd, [Rb, Ro]	LDRB Rd, [Rb, Ro]	前变址字节加载 :计算 Rb 与 Ro 的值之和为源地址。 加载结果地址的内容到 Rd 中

指令周期时间

此格式的所有指令都与表 4-8 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

```
LDR    R3,  [R2,  R6]          ;从 R6 加 R2 得到的地址加载字到 R3
LDRB   R2,  [R0,  R7]          ;R7 加 R0 得到的地址加载字节到 R2 中
```

格式 8：加载/存储符号扩展字节/半字



图 4-9. 格式 8

操作

这些指令加载任意符号扩展字节或半字，存储半字。Thumb 汇编语法如表 4-9 所示。

表 4-9. 格式 8 指令概述

H	S	Thumb 汇编	对应 ARM 汇编	描述
0	0	STRH Rd, [Rb, Ro]	STRH Rd, [Rb, Ro]	存储半字：基址加上 Ro 到 Rb 中。存储 Rd 的位[15:0]到结果地址
0	1	LDRH Rd, [Rb, Ro]	LDRH Rd, [Rb, Ro]	加载半字：基址加上 Ro 到 Rb 中。从结果地址加载到 Rd 的位[15:0]，并设置位[31:16]为 0
1	0	LDSB Rd, [Rb, Ro]	LDRSB Rd, [Rb, Ro]	加载符号扩展字节：基址加上 Ro 到 Rb 中。从结果地址加载到 Rd 的位[7:0]，并设置位[31:8]为位[7]
1	1	LDSH Rd, [Rb, Ro]	LDRSH Rd, [Rb, Ro]	加载符号扩展半字：基址加上 Ro 到 Rb 中。从结果地址加载到 Rd 的位[15:0]，并设置位[31:16]为位[15]

指令周期时间

此格式的所有指令都与表 4-9 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

STRH	R4, [R3, R0]	:存储 R4 的低 16 位到 R0 加 R3 得到的地址中
LDSB	R2, [R7, R1]	;R1 加 R7 得到的地址的符号扩展字节加载到 R2
LDSH	R3, [R4, R2]	;R2 加 R4 得到的地址的符号扩展半字加载到 R3

格式 9：带立即数偏移的加载/存储

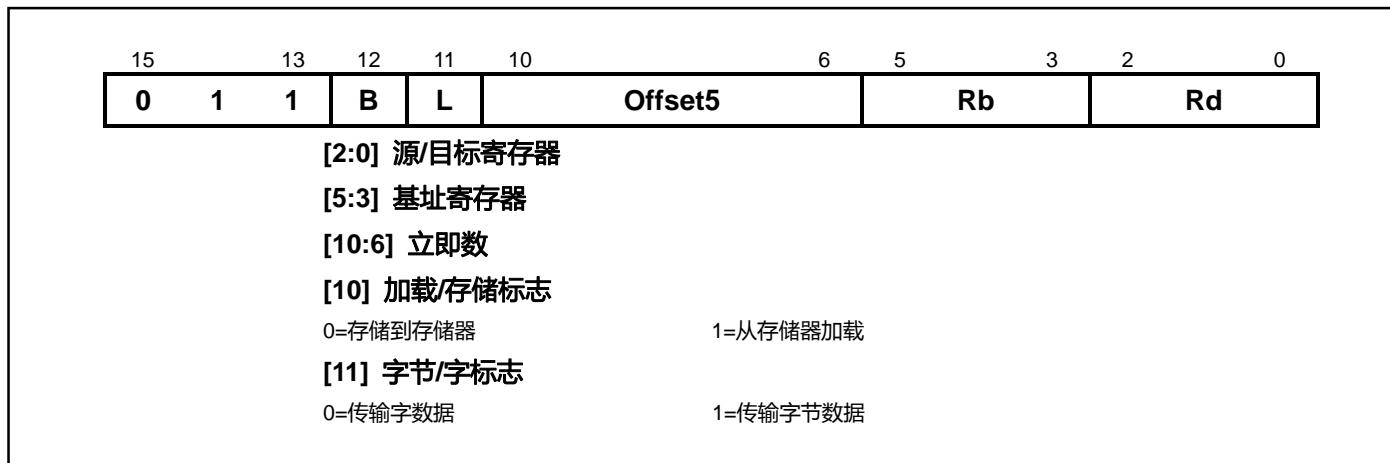


图 4-10. 格式 9

操作

这些指令在寄存器与存储器之间使用 5 位立即数或 7 位偏移量传输字节或字数据。Thumb 汇编语法如表 4-10 所示。

表 4-10. 格式 9 指令概述

B	L	Thumb 汇编	对应 ARM 汇编	描述
0	0	STR Rd, [Rb, #Imm]	STR Rd, [Rb, #Imm]	计算 Rb 与 Imm 的值之和为目标地址。存储 Rd 的内容到结果地址
0	1	LDR Rd, [Rb, #Imm]	LDR Rd, [Rb, #Imm]	计算 Rb 与 Imm 的值之和为源地址。加载结果地址的内容到 Rd 中
1	0	STRB Rd, [Rb, #Imm]	STRB Rd, [Rb, #Imm]	计算 Rb 与 Imm 的值之和为目标地址。存储 Rd 字节到结果地址
1	1	LDRB Rd, [Rb, #Imm]	LDRB Rd, [Rb, #Imm]	计算 Rb 与 Imm 的值之和为源地址。加载结果地址内字节到 Rd 中

注释：

按字访问 (B=0)，由#Imm 指定的值是一个完整的 7 位地址，但必须按字对齐（位[1:0]设置为 0），因为汇编会将#Imm>>2 放置在 Offset5 字段中。

指令周期时间

此格式的所有指令都与表 4-10 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

LDR <i>R2, [R5, #116]</i>	;从 R5 加上 116 得到的地址加载字到 R2。注意 Thumb 操作码将按 29 作为 ;Offset5 值
STRB <i>R1, [R0, #13]</i>	;存储 R1 的低 8 位到 R0 加上 13 得到的地址中。注意 Thumb 操作码将按 13 ;作为 Offset5 值

格式 10：加载/存储半字



图 4-11. 格式 10

操作

这些指令在低寄存器与存储器之间传输半字数据。寻址模式为使用 6 位立即数的前变址。Thumb 汇编语法如表 4-11 所示。

表 4-11. 格式 10 指令概述

L	Thumb 汇编	对应 ARM 汇编	描述
0	STRH Rd, [Rb, #Imm]	STRH Rd, [Rb, #Imm]	基址加上#Imm 到 Rb 中，存储 Rd 的位[15:0]到结果地址
1	LDRH Rd, [Rb, #Imm]	LDRH Rd, [Rb, #Imm]	基址加上#Imm 到 Rb 中，存储 Rd 的位[15:0]到结果地址，Rd 的位[31:16]设置为 0

注释：

#Imm 是一个完整的 6 位地址，但必须按半字对齐（位[0]设置为 0），因为汇编会将#Imm>>1 放置在 Offset5 字段中。

指令周期时间

此格式的所有指令都与表 4-11 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

STRH R6, [R1, #56]	;存储 R4 的低 16 位到 R1 加上 56 得到的地址中。注意 Thumb 操作码将按 28 作 ;为 Offset5 值
LDRH R4, [R7, #4]	;从 R7 加上 4 得到的地址加载字到 R4。注意 Thumb 操作码将按 2 作为 Offset5 ;值

格式 11：加载/存储相对 SP

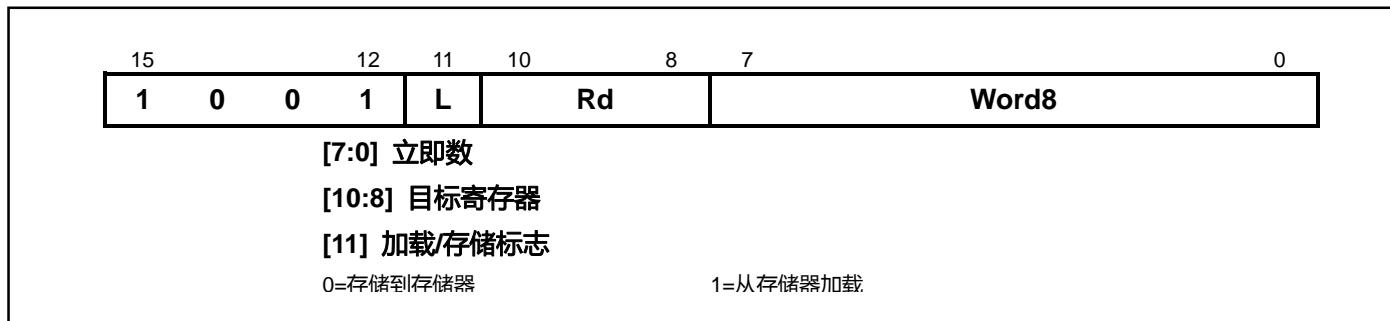


图 4-12. 格式 11

操作

该组中的指令执行加载或存储相对 SP。Thumb 汇编语法如表 4-12 所示。

表 4-12. 格式 11 指令概述

L	Thumb 汇编	对应 ARM 汇编	描述
0	STR Rd, [SP, #Imm]	STR Rd, [R13 #Imm]	基址加上#Imm 到 Rb 中，存储 Rd 的位[15:0]到结果地址
1	LDR Rd, [SP, #Imm]	LDR Rd, [R13 #Imm]	基址加上#Imm 到 Rb 中，存储 Rd 的位[15:0]到结果地址，Rd 的位[31:16]设置为 0

注释：

#Imm 提供的偏移是一个完整的 10 位地址，但必须按字对齐（位[1:0]设置为 0），因为汇编会将#Imm>>2 放置在 Word8 字段中

指令周期时间

此格式的所有指令都与表 4-12 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

STR R4, [SP, #492] ;存储当前 R4 的内容到 SP (R13) 加上 492 得到的地址注意 Thumb 操作码将按
;123 作为 Word8 值

格式 12：地址加载

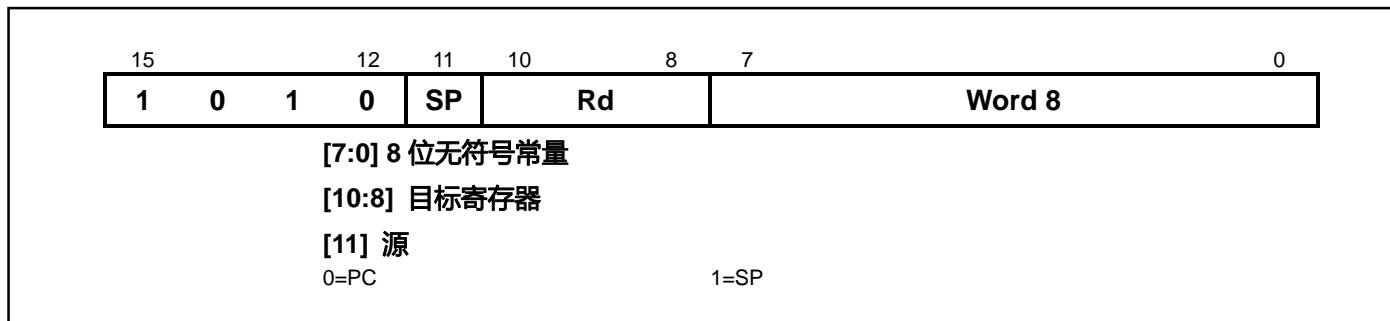


图 4-13. 格式 12

操作

这些指令计算 PC 或 SP 加上 10 位常量的结果产生的地址，并加载结果地址到一个寄存器中。Thumb 汇编语法如表 4-13 所示。

表 4-13. 格式 12 指令概述

L	Thumb 汇编	对应 ARM 汇编	描述
0	ADD Rd, PC, #Imm	ADD Rd, R15, #Imm	当前程序计数器(PC)的值加上#Imm 并加载其结果到 Rd 中
1	ADD Rd, SP, #Imm	ADD Rd, R13, #Imm	当前堆栈指针(SP)的值加上#Imm 并加载其结果到 Rd 中

注释：

#Imm 指定的值是一个完整的 10 位数，但必须按字对齐(位[0]设置为 0)，因为汇编会将#Imm>>2 放置在 Word8 字段中。

PC 是用作源寄存器(SP=0)，PC 的位[1]总是读做为 0。PC 的值将在位[1]强制为 0 前超过指令地址 4 个字节。CPSR 条件码不受此这些指令的影响。

指令周期时间

此格式的所有指令都与表 4-13 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

ADD R2, PC, #572 ADD R6, SP, #212	;R2:=PC+572，但不要设置条件码。PC 的位[1]强制为 0。注意 Thumb 操作码将相 ;当于 143 作为 Word8 值 ;R6:=SP+212，但不要设置条件码。注意 Thumb 操作码将按 53 作为 Word8 值
--	--

格式 13：堆栈指针加偏移

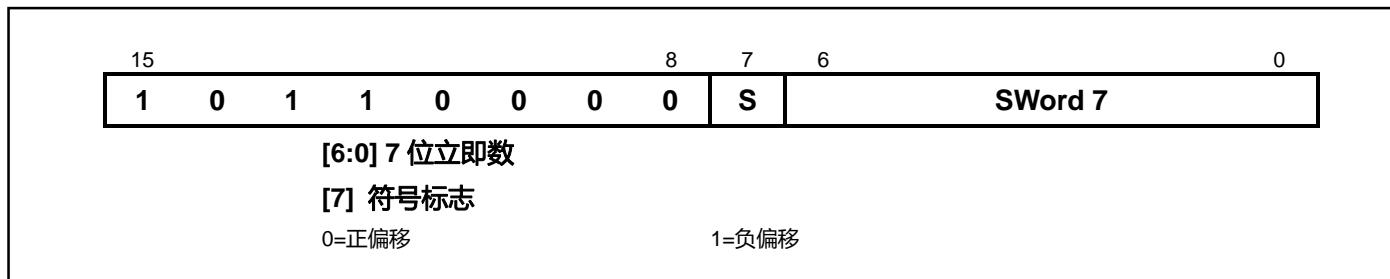


图 4-14. 格式 13

操作

此指令加上一个 9 位有符号常量到堆栈指针。

表 4-14. 格式 13 指令概述

L	Thumb 汇编	对应 ARM 汇编	描述
0	ADD SP, #Imm	ADD R13, R13, #Imm	堆栈指针 (SP) 加上#Imm
1	ADD SP, # -Imm	SUB R13, R13, #Imm	堆栈指针 (SP) 加上#-Imm

注释：

#Imm 指定的偏移可以最大 ± 508 , 但必须按字对齐(位[1:0]设置为 0), 因为汇编会在将#Imm 放置其在 SWord7 字段中之前转换为一个 8 位符号+长度的数。

指令周期时间

此格式的所有指令都与表 4-14 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

ADD	SP, #268	;SP (R13) := SP + 268 , 但不要设置条件码。 ;注意 THUMB 操作码将会等于 67 作为 Word7 值并且 S=0。
ADD	SP, #-104	;SP (R13) := SP - 104 , 但不要设置条件码。 ;注意 THUMB 操作码将会等于 26 作为 Word7 值并且 S=1。

格式 14：寄存器入栈/出栈



图 4-15. 格式 14

操作

该组指令允许寄存器 0 至 7 和可选 LR 压入堆栈，寄存器 0 至 7 和可选 PC 弹出堆栈。Thumb 汇编语法如表 4-15 所示。

注释：

该堆栈通常假定为满递减。

表 4-15. 格式 14 指令概述

L	R	Thumb 汇编	对应 ARM 汇编	描述
0	0	PUSH { Rlist }	STMDB R13!, { Rlist }	将 Rlist 指定寄存器压入堆栈。更新堆栈指针
0	1	PUSH { Rlist, LR }	STMDB R13!, { Rlist, R14 }	将链接寄存器和 Rlist 指定寄存器（如果有）压入堆栈。更新堆栈指针
1	0	POP { Rlist }	LDMIA R13!, { Rlist }	弹出堆栈的值到 Rlist 指定寄存器中。更新堆栈指针
1	1	POP { Rlist, PC }	LDMIA R13!, {Rlist, R15}	弹出堆栈的值到 Rlist 指定寄存器中。弹出堆栈 PC。更新堆栈指针

指令周期时间

此格式的所有指令都与表 4-15 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

PUSH {R0- R4, LR}	;存储 R0 , R1 , R2 , R3 , R4 和 R14 (LR) 到由 R13 (SP) 指向的堆栈中并更新 R13。用于子程序开始处保存工作区域和返回地址
POP {R2, R6, PC}	;从由 R13 (SP) 指向的堆栈加载 R2 , R6 和 R15 (PC) 并更新 R13。用于恢复工作区域和子程序的返回

格式 15：加载/存储多块

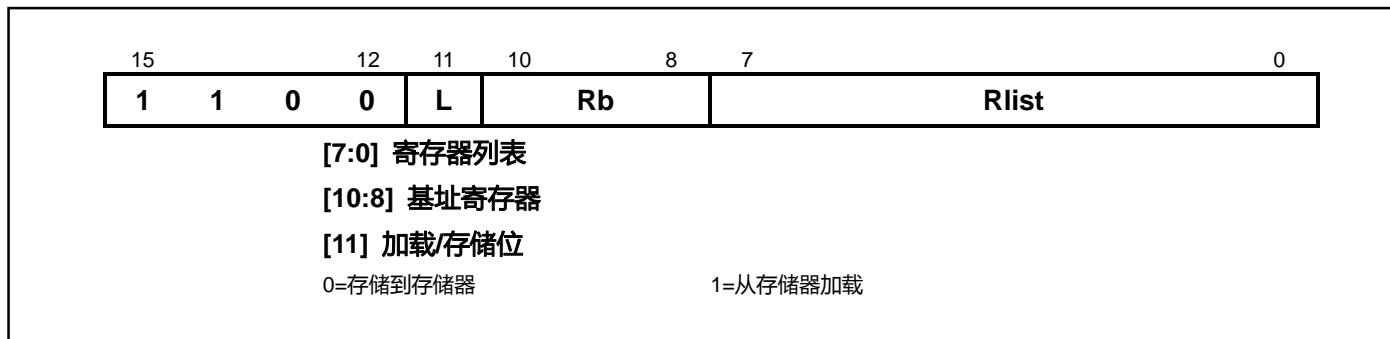


图 4-16. 格式 15

操作

这些指令允许加载或存储低寄存器多块。Thumb 汇编语法如表 4-16 所示。

表 4-16. 格式 15 指令概述

L	Thumb 汇编	对应 ARM 汇编	描述
0	STMIA Rb!, { Rlist }	STMIA Rb!, { Rlist }	存储 Rlist 指定的寄存器，开始于 Rb 中的基址。回写新基址
1	LDMIA Rb!, { Rlist }	LDMIA Rb!, { Rlist }	加载 Rlist 指定的寄存器，开始于 Rb 中的基址。回写新基址

指令周期时间

此格式的所有指令都与表 4-16 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

STMIA R0!, {R3- R7} ;存储 R3 至 R7 的内容，起始地址由 R0 指定，按字递增地址。回写值更新 R0

格式 16：条件分支跳转

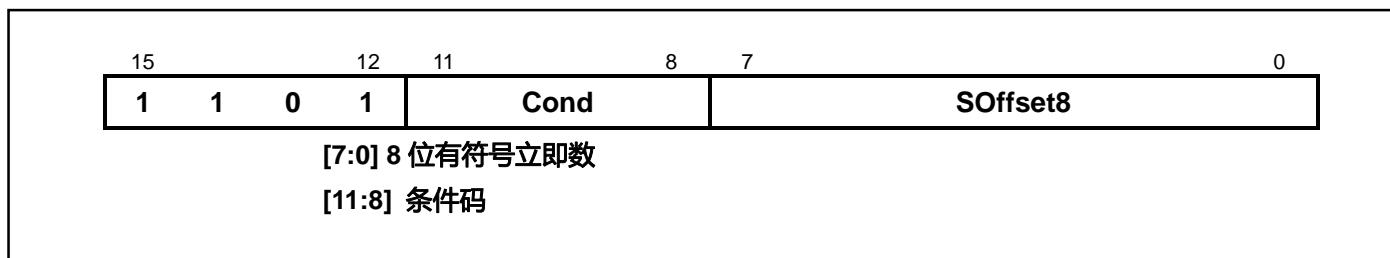


图 4-17. 格式 16

操作

该组指令全部根据 CPSR 条件码的状态执行条件分支跳转。由于 PC 将超前于当前指令 1 个字 (4 个字节)，故分支跳转偏移量还必须考虑预取操作。Thumb 汇编语法如表 4-17 所示。

表 4-17. 格式 16 指令概述

Cond	Thumb 汇编	对应 ARM 汇编	描述
0000	BEQ label	BEQ label	如果 Z 置位则分支跳转 (相等)
0001	BNE label	BNE label	如果 Z 清零则分支跳转 (不相等)
0010	BCS label	BCS label	如果 C 置位则分支跳转 (无符号大于等于)
0011	BCC label	BCC label	如果 C 清零则分支跳转 (无符号小于)
0100	BMI label	BMI label	如果 N 置位则分支跳转 (负数)
0101	BPL label	BPL label	如果 N 清零则分支跳转 (正数或零)
0110	BVS label	BVS label	如果 V 置位则分支跳转 (溢出)
0111	BVC label	BVC label	如果 V 清零则分支跳转 (未溢出)
1000	BHI label	BHI label	如果 C 置位且 Z 清零则分支跳转 (无符号大于)
1001	BLS label	BLS label	如果 C 清零且 Z 置位则分支跳转 (无符号小于等于)
1010	BGE label	BGE label	如果 N 置位且 V 清零或者 N 清零且 V 置位则分支跳转 (大于等于)
1011	BLT label	BLT label	如果 N 清零且 V 置位或者 N 置位且 V 清零则分支跳转 (小于)
1100	BGT label	BGT label	如果 Z 清零时 N 置位 V 置位或者 N 清零 V 清零则分支跳转 (小于等于)
1101	BLT label	BLE label	如果 Z 置位时 N 清零 V 清零或者 N 置位 V 置位则分支跳转 (大于等于)

注释：

1. 标号 (*label*) 指定了一个完整的 9 位二进制补码地址。此必须按半字对齐 (位[0]设置为 0)，因为汇编将会放置 *label>>1* 在 SOffset8 字段。
2. Cond=1110 没有定义，而且不要使用。Cond=1111 创建 SWI 指令：请清楚这点。

指令周期时间

此格式的所有指令都与表 4-17 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

```
CMP    R0, #45      ;如果 R0>45 则分支跳转到 over
BGT    over         ;注意 Thumb 操作码将按半字数据到偏移量

over              ;必须按半字对齐
```

格式 17：软件中断

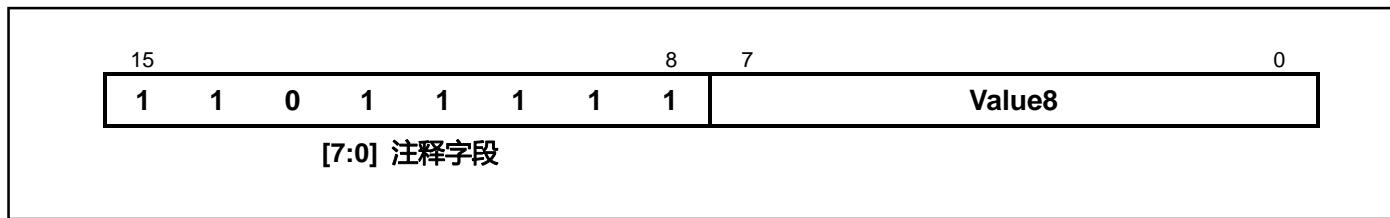


图 4-18. 格式 17

操作

SWI 指令执行了一个软件中断。SWI 将处理器切换到 ARM 状态并进入管理 (SVC) 模式。Thumb 汇编语法如表 4-18 所示。

表 4-18. 格式 17 指令概述

Thumb 汇编	对应 ARM 汇编	描述
SWI Value8	SWI Value8	执行软件中断： 移动下条指令地址到 LR 中，移动 CPSR 到 SPSR 中，加载 SWI 向量地址 (0x8) 到 PC。切换到 ARM 状态并进入管理模式。

注释：

Value8 只用于 SWI 处理程序；处理器忽略该值。

指令周期时间

此格式的所有指令都与表 4-18 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

SWI 18 ;软件中断例子。以 18 作为请求 SWI 号进入管理模式

格式 18：无条件分支跳转

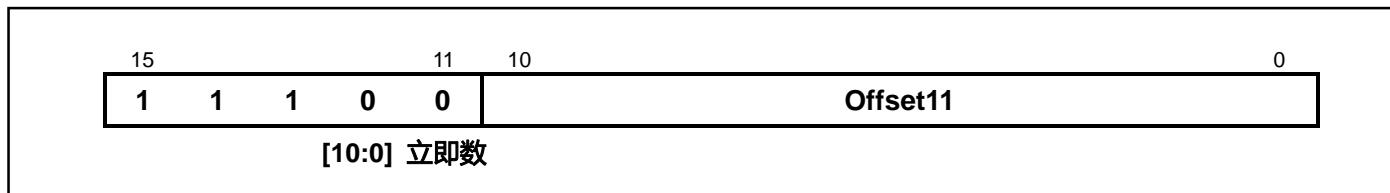


图 4-19. 格式 18

操作

该组指令执行相对 PC 分支跳转。Thumb 汇编语法如表 4-19 所示。分支跳转偏移必须考虑预取操作，因为 PC 超前当前指令 1 个字节（4 个字节）。

表 4-19. 格式 18 指令概述

Thumb 汇编	对应 ARM 汇编	描述
BEQ label	BEQ label (半字偏移)	分支跳转到相对 PC +/- Offset11<<1 此处标号是 PC +/- 2048 字节。

注释：

由标号指定的地址是一个 12 位二进制补码地址，但必须按半字对齐（位[0]设置为 0），因为汇编将会放置 label>>1 到 Offset11 字段。

指令周期时间

此格式的所有指令都与表 4-18 中的 ARM 指令对应。Thumb 指令的指令周期时间与对应 ARM 指令相同。

例子

here	B	here	;分支跳转到自身。汇编到 0xE7FE。（注意 PC 偏移的影响）
	B	jimmy	;分支跳转到 'jimmy'。注意 Thumb 操作码将按半字偏移
		
jimmy			;必须按半字对齐

格式 19：带链接长分支跳转

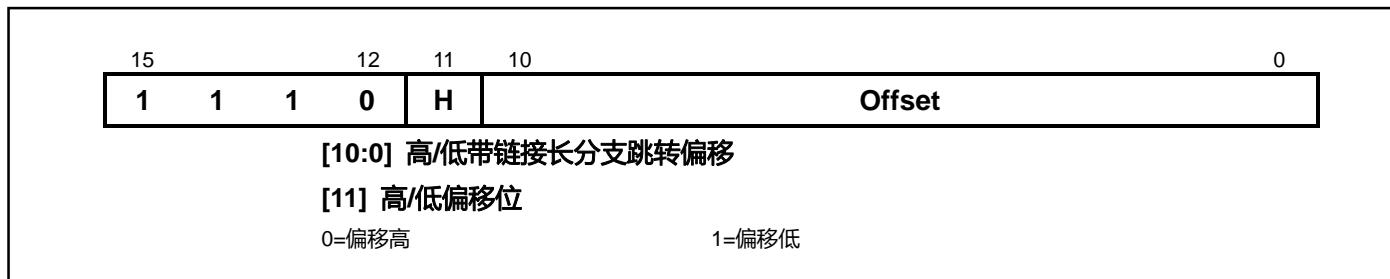


图 4-20. 格式 19

操作

此格式指定了一个带链接长分支跳转。

汇编将标号指定的 23 位二进制补码半字偏移分离为两个 11 位，忽略位[0]（必须设置为 0），并创建两个 Thumb 指令。

指令 1 (H=0)

第一条指令，偏移字段控制目标地址的高 11 位。这是左移 12 位并加上当前 PC 地址得到的。结果地址放置在 LR 中。

指令 2 (H=1)

第二条指令，偏移字段控制目标地址的低 11 位。这是左移 1 位并加上当前 PC 地址得到的。LR 此时按完整的 23 位地址来看待并被放置在 PC 中，BL 后指令的地址放置在 LR 中并置位了 LR 的位[0]。

分支跳转偏移必须考虑预取操作，因为 PC 超前当前指令 1 个字（4 个字节）。

表 4-20. 格式 19 指令概述

H	Thumb 汇编	对应 ARM 汇编	描述
0			LR:=PC+OffsetHigh<<12
1	BL label	无	temp:=下条指令地址 PC:=LR+OffsetLow<<1 LR:=temp 逻辑或 1

指令周期时间

此格式的指令无 ARM 指令对应。

例子

```
BL      faraway ;无条件分支跳转到 'faraway' 并放置随后指令地址(如 "next")到链接寄存器 R14 , ,
next
..... ;LR 的位[0]置位。注意 Thumb 操作码将按半字偏移。
faraway ;必须按半字对齐
```

指令集例子

以下例子显示了 Thumb 指令可能用于产生精简高效代码的方法。每个例子同样显示了对应 ARM 以对照。

使用偏移和加法的常量乘法

以下指令是各种使用 1 , 2 或 3 Thumb 指令的常量乘法。旁边为对应的 ARM 指令。对于其他常量，使用嵌入 MUL 指令的常量比使用 4 的倍数或更多要好。

Thumb	ARM
1. $2^n (1 , 2 , 4 , 8.....)$ 乘法 $LSL \quad Ra, \quad Rb, \quad LSL \#n$;MOV Ra, Rb, LSL #n
2. $2^{n+1} (3 , 5 , 9 , 17.....)$ 乘法 $LSL \quad Rt, \quad Rb, \quad \#n$ $ADD \quad Ra, \quad Rt, \quad Rb$;ADD Ra, Rb, Rb, LSL #n
3. $2^{n-1} (3 , 7 , 15.....)$ 乘法 $LSL \quad Rt, \quad Rb, \quad \#n$ $SUB \quad Ra, \quad Rt, \quad Rb$;RSB Ra, Rb, Rb, LSL #n
4. $-2^n (-2 , -4 , -8.....)$ 乘法 $LSL \quad Rt, \quad Rb, \quad \#n$ $MVN \quad Ra, \quad Ra$;MOV Ra, Rb, LSL #n ;RSB Ra, Ra, #0
5. $-2^{n-1} (-3 , -7 , -15.....)$ 乘法 $LSL \quad Rt, \quad Rb, \quad \#n$ $SUB \quad Ra, \quad Rb, \quad Rt$;SUB Ra, Rb, Rb, LSL #n ;
任何 $C=\{2^{n+1}, 2^{n-1}, -2^n \text{ or } -2^{n-1}\} * 2^n$ 的乘法	
实际上这是任何最后带移位的 2 到 5 的乘法。这允许以下增加的常量乘法。6 , 10 , 12 , 14 , 18 , 20 , 24 , 28 , 30 , 34 , 36 , 40 , 48 , 56 , 60 , 62.....	
(2...5) $LSL \quad Ra, \quad Ra, \quad \#n$	(2...5) ;MOV Ra, Ra, LSL #n

使用偏移和加法的常量乘法

这些例子显示了 Thumb 和 ARM 代码中的通用有符号除法和余数程序。

Thumb 代码

```

;signed_divide
;R0 的绝对值放入中 R3
    ASR    R2,  R0,  #31      ;R0 的符号决定 0 或-1 放入 R2 中
    EOR    R0,  R2            ;如果为负则异或-1 ( 0xFFFFFFFF )
    SUB   R3,  R0,  R2        ;并加上 1 ( 减 1 ) 来得到绝对值

;SUB 通常置位标志位，如果为 0 则需要前往并通报除法
    BEQ    divide_by_zero
;如果为负数逻辑异或 0xFFFFFFFF 并加 1 以获取 R1 的绝对值
    ASR    R0,  R1,  #31      ;R1 的符号决定 0 或-1 放入 R3 中
    EOR    R1,  R0            ;如果为负则异或-1 ( 0xFFFFFFFF )
    SUB   R1,  R0            ;并加上 1 ( 减 1 ) 来得到绝对值

;确定为以后使用的符号 (R0 和 R2 中的 0 或-1) 保存；商和余数的符号
    PUSH   {R0,  R2}

;辩解 (Justification)，依次移 1 位直至除数 (R0 的值)；只大于等于被除数 (R1 的值)。要做到这一点移位被除数。
;右移 1 并在移位值变为大于除数。
    LSR    R0,  R1,  #1
    MOV    R2,  R3
    B     %FT0
just_I  LSL    R2,  #1
0       CMP    R2,  R0
    BLS   just_I
    MOV    R0,  #0            ;设置累加器为 0
    B     %FT0              ;分支跳转到除法循环

div_I   LSR    R2,  #1
0       CMP    R1,  R2            ;测试减
    BCC   %FT0
    SUB   R1,  R2            ;若成功则减
0       ADC    R0,  R0            ;若减法成功则移位结果并加 1
    CMP    R2,  R3            ;当 R2 等于 R3 时结束
    BNE   div_I             ;( 我们测试过减 '1' 值 )

;此时确定商 (R0) 和余数 (R1) 的符号
    POP    {R2,  R3}
    EOR    R3,  R2
    EOR    R0,  R3            ;如结果符号=-1 则为取反
    SUB   R0,  R3
    EOR    R1,  R2            ;如除数符号=-1 则为余数
    SUB   R1,  R2
    MOV    pc,  lr

```

ARM 代码

```

:signed_divide                                ; 若有效零 a4 为最高位将随后移出
;R0 的绝对值放入中 R3
    ANDS  a4,  a1,  #&80000000
    RSBMI a1,  a1,  #0
    EORS  ip,  a4,  a2,  ASR #32
;ip 位[31]=结果的符号
;ip 位[30]= a2 的符号
    RSBCS a2,  a2,  #0
;中心部分为相同无符号除法代码 (除了 MOV a4, #0 以外，其不受影响有符号部分进入顺序)
    MOVS  a3,  a1
    BEQ   divide_by_zero
just_I
    CMP   a3,  a2,  LSR #1      ;辩解阶段一次移移 1 位
    MOVLS a3,  a3,  LSL #1
    BLO   s_loop
div_I
    CMP   a2,  a3
    ADC   a4,  a4,  a4
    SUBCS a2,  a2,  a3
    TEQ   a3,  a1
    MOVNE a3,  a3,  LSR #1
    BNE   s_loop2
    MOV   a1,  a4
    MOVS  ip,  ip,  ASL #1
    RSBCS a1,  a1,  #0
    RSBMI a2,  a2,  #0
    MOV   pc,  lr

```

常量乘法

常量除法通常可以由一个简短固定移位序列，加法和减法来执行。

此处有一个基于 ARM Cookbook 算法除以 10 的 Thumb 和 ARM 代码程序的例子。

Thumb 代码

```

udiv10                                ;参数为 a1 , 返回商到 a1 , 余数到 a2
    MOV   a2,  a1
    LSR   a3,  a1,  #2
    SUB   a1,  a3
    LSR   a3,  a1,  #4
    ADD   a1,  a3
    LSR   a3,  a1,  #8
    ADD   a1,  a3
    LSR   a3,  a1,  #16
    ADD   a1,  a3
    LSR   a1,  #3
    ASL   a3,  a1,  #2
    ADD   a3,  a1
    ASL   a3,  #1
    SUB   a2,  a3
    CMP   a2,  #10
    BLT   %FT0
    ADD   a1,  #1
    SUB   a2,  #10
0
    MOV   pc,  lr

```

ARM 代码

udiv10 ;参数为 a1 , 返回商到 a1 , 余数到 a2

```
SUB    a2,  a1,  #10
SUB    a1,  a1,  a1,  lsr #2
ADD    a1,  a1,  a1,  lsr #4
ADD    a1,  a1,  a1,  lsr #8
ADD    a1,  a1,  a1,  lsr #16
MOV    a1,  a1,  lsr #3
ADD    a3,  a1,  a1,  lsr #2
SUBS   a2,  a2,  A3,  lsr #1
ADDPL  a1,  a1,  #1
ADDMI  a2,  a2,  #10
MOV    pc,  lr
```

5 存储器控制器

概述

S3C2440A 存储器控制器为访问外部存储的需要器提供了存储器控制信号。

S3C2440A 包含以下特性：

- 大/小端 (通过软件选择)
- 地址空间：每个 Bank 有 128M 字节 (总共 1G/8 个 Bank)
- 大/小端 (通过软件选择)
- 除了 BANK0 (16/32 位) 之外，其它全部 BANK 都可编程访问宽度 (8/16/32 位)
- 总共 8 个存储器 Bank
 - 6 个存储器 Bank 为 ROM, SRAM 等
 - 其余 2 个存储器 Bank 为 ROM, SRAM, SDRAM 等
- 7 个固定的存储器 Bank 起始地址
- 1 个可变的存储器 Bank 起始地址并 Bank 大小可编程
- 所有存储器 Bank 的访问周期可编程
- 外部等待扩展总线周期
- 支持 SDRAM 自刷新和掉电模式

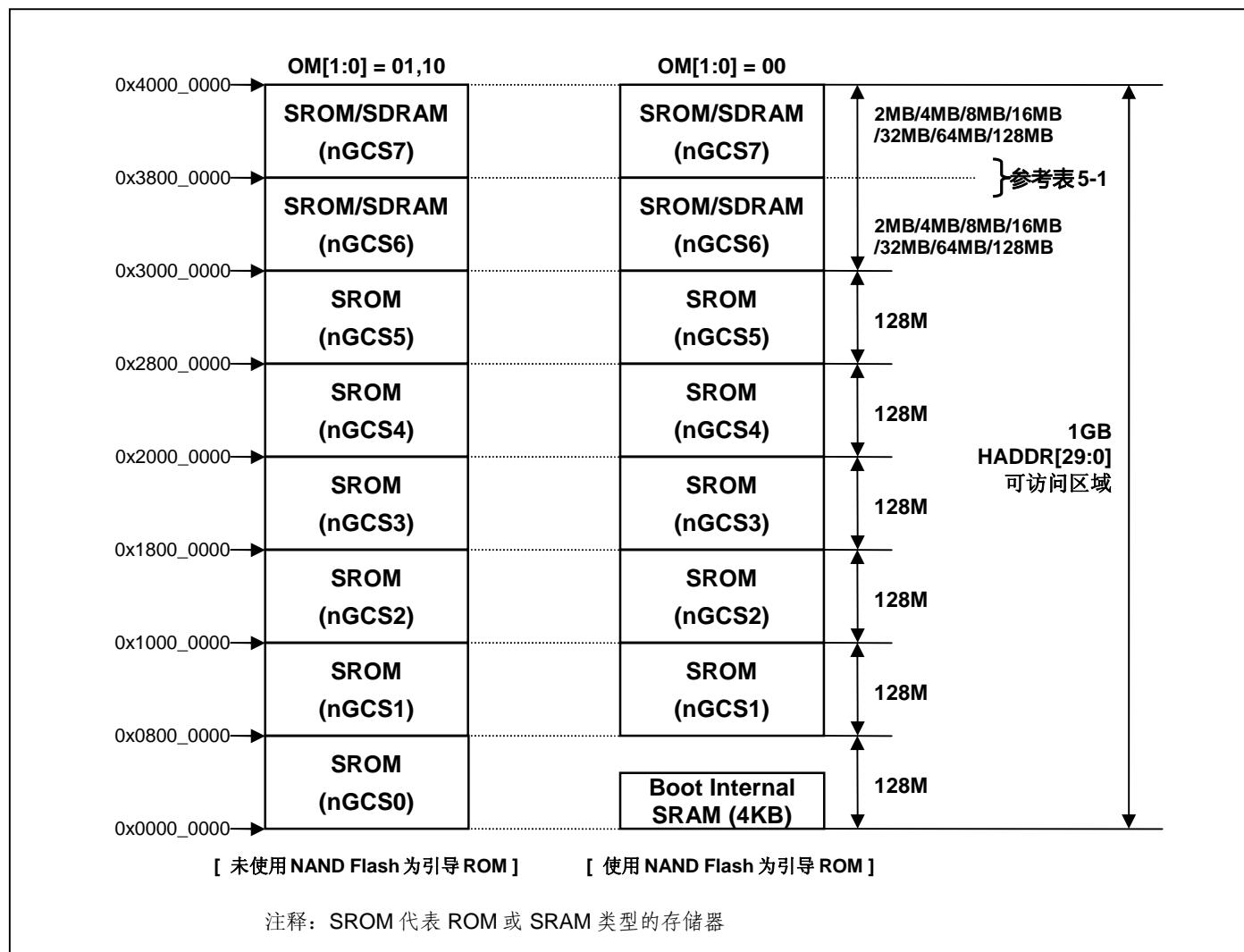


图 5-1. 复位后 S3C2440A 的存储器映射

表 5-1. BANK 6/7 地址

地址	2MB	4MB	8MB	16MB	32MB	64MB	128MB
Bank 6							
开始地址	0x3000_0000						
结束地址	0x301F_FFFF	0X303F_FFFF	0X307F_FFFF	0X30FF_FFFF	0X31FF_FFFF	0X33FF_FFFF	0X37FF_FFFF
Bank 7							
开始地址	0x3020_0000	0x3040_0000	0x3080_0000	0x3100_0000	0x3200_0000	0x3400_0000	0x3800_0000
结束地址	0X303F_FFFF	0X307F_FFFF	0X30FF_FFFF	0X31FF_FFFF	0X33FF_FFFF	0X37FF_FFFF	0X3FFF_FFFF

注释：

BANK 6 和 BANK 7 必须为相同的存储器大小。

功能描述

BANK0 总线宽度

BANK0 (nGCS0) 的数据总线应当配置为 16 位或 32 位的宽度。因为 BANK0 是作为引导 ROM 的 bank (映射到 0x0000_0000), 应当在第一个 ROM 访问前决定 BANK0 的总线宽度 , 其依赖于复位时 OM[1:0] 的逻辑电平。

OM1 (操作模式 1)	OM0 (操作模式 0)	引导 ROM 数据宽度
0	0	Nand Flash 模式
0	1	16 位
1	0	32 位
1	1	测试模式

存储器 (SROM/SDRAM) 地址引脚连接

存储器地址引脚	S3C2440A 地址 @ 8 位数据总线	S3C2440A 地址 @ 16 位数据总线	S3C2440A 地址 @ 32 位数据总线
A0	A0	A1	A2
A1	A1	A2	A3
...

SDRAM BANK 地址引脚连接例子

表 5-2. SDRAM Bank 地址引脚连接例子

Bank 大小	总线宽度	基本组成部分	存储器结构	Bank 地址
2M 字节	x8	16M 比特	(1M × 8 × 2Bank) × 1	A[20]
	x16		(512K × 16 × 2B) × 1	
4M 字节	x16	32M 比特	(1M × 8 × 2B) × 2	A[21]
	x16		(1M × 8 × 2B) × 2	
8M 字节	x16	16M 比特	(2M × 4 × 2B) × 4	A[22]
	x32		(1M × 8 × 2B) × 1	
	x8	64M 比特	(4M × 8 × 2B) × 1	A[22:21]
	x8		(2M × 8 × 4B) × 1	
	x16		(2M × 16 × 2B) × 1	A[22]
	x16		(1M × 16 × 4B) × 1	A[22:21]
	x32		(512K × 32 × 4B) × 1	
16M 字节	x32	16M 比特	(2M × 4 × 2B) × 8	A[23]
	x8	64M 比特	(8M × 4 × 2B) × 2	
	x8		(4M × 4 × 4B) × 2	A[23:22]
	x16		(4M × 8 × 2B) × 2	A[23]
	x16		(2M × 8 × 4B) × 2	A[23:22]
	x32		(2M × 16 × 2B) × 2	A[23]
	x32	128M 比特	(1M × 16 × 4B) × 2	A[23:22]
	x8		(4M × 8 × 4B) × 1	
	x16		(2M × 16 × 4B) × 1	
32M 字节	x16	64M 比特	(8M × 4 × 2B) × 4	A[24]
	x16		(4M × 4 × 4B) × 4	A[24:23]
	x32		(4M × 8 × 2B) × 4	A[24]
	x32		(2M × 8 × 4B) × 4	A[24:23]
	x16	128M 比特	(4M × 8 × 4B) × 2	
	x32		(2M × 16 × 4B) × 2	
	x8	256M 比特	(8M × 8 × 4B) × 1	
	x16		(4M × 16 × 4B) × 1	
64M 字节	x32	128M 比特	(4M × 8 × 4B) × 4	A[25:24]
	x16	256M 比特	(8M × 8 × 4B) × 2	
	x32		(4M × 16 × 4B) × 2	
	x8	512M 比特	(16M × 8 × 4B) × 1	
128M 字节	x32	256M 比特	(8M × 8 × 4B) × 4	A[26:25]
	x16	512M 比特	(32M × 4 × 4B) × 2	
	x8		(16M × 8 × 4B) × 2	
	x32		(8M × 16 × 4B) × 2	

nWAIT 引脚操作

如果使能了每个存储器 bank 对应的 WAIT 位 (BWSCON 中的 WSN 位), 存储器 bank 有效时 nOE 持续时间应当被外部 nWAIT 引脚延长。从 tacc-1 开始检测 nWAIT。nOE 将在采样 nWAIT 为高后的下个时钟被取消。nWE 信号端与 nOE 有相同的关系。

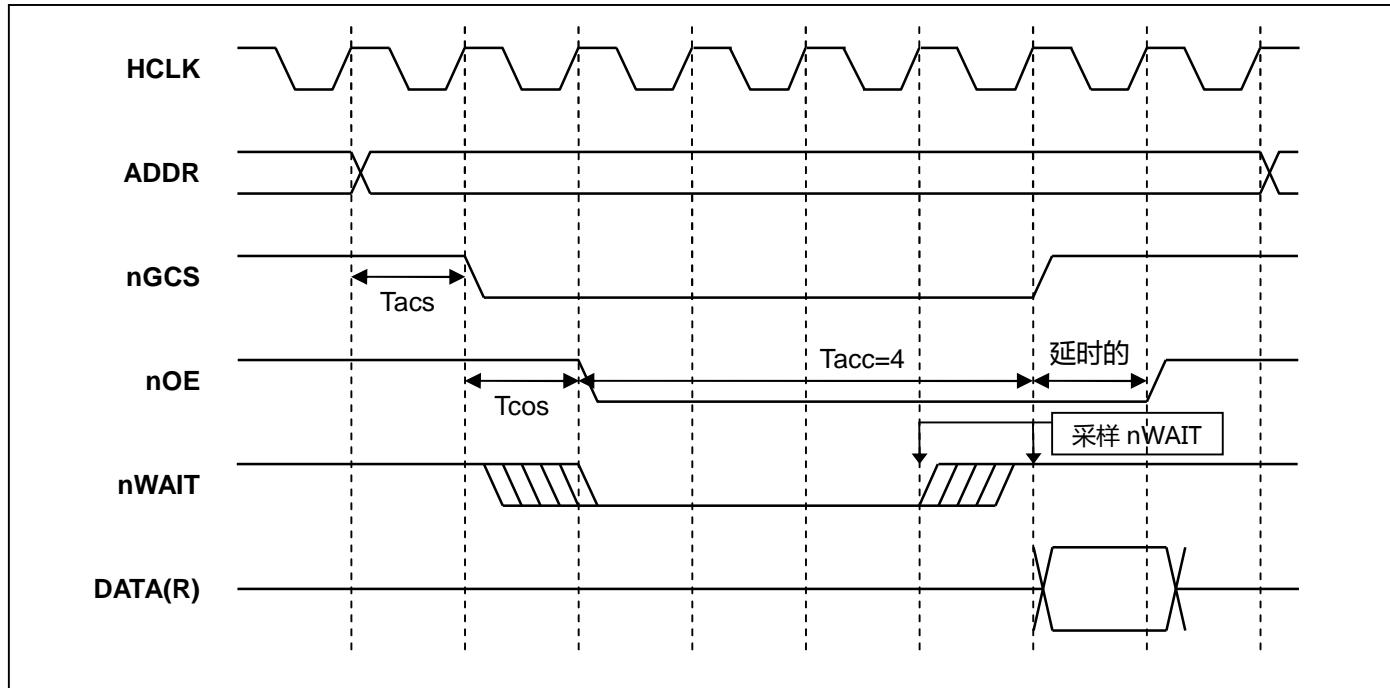


图 5-2. S3C2440A 外部 nWAIT 时序 示意图 ($T_{acc}=4$)

nXBREQ/nXBACK 引脚操作

如果发出了 nXBREQ , S3C2440A 将以拉低 nXBACK 应答。如果 nXBACK=L , 地址/数据总线和存储器控制信号将为高阻态 , 如表 1-1 所示。在 nXBREQ 取消后 , nXBACK 将同样取消。

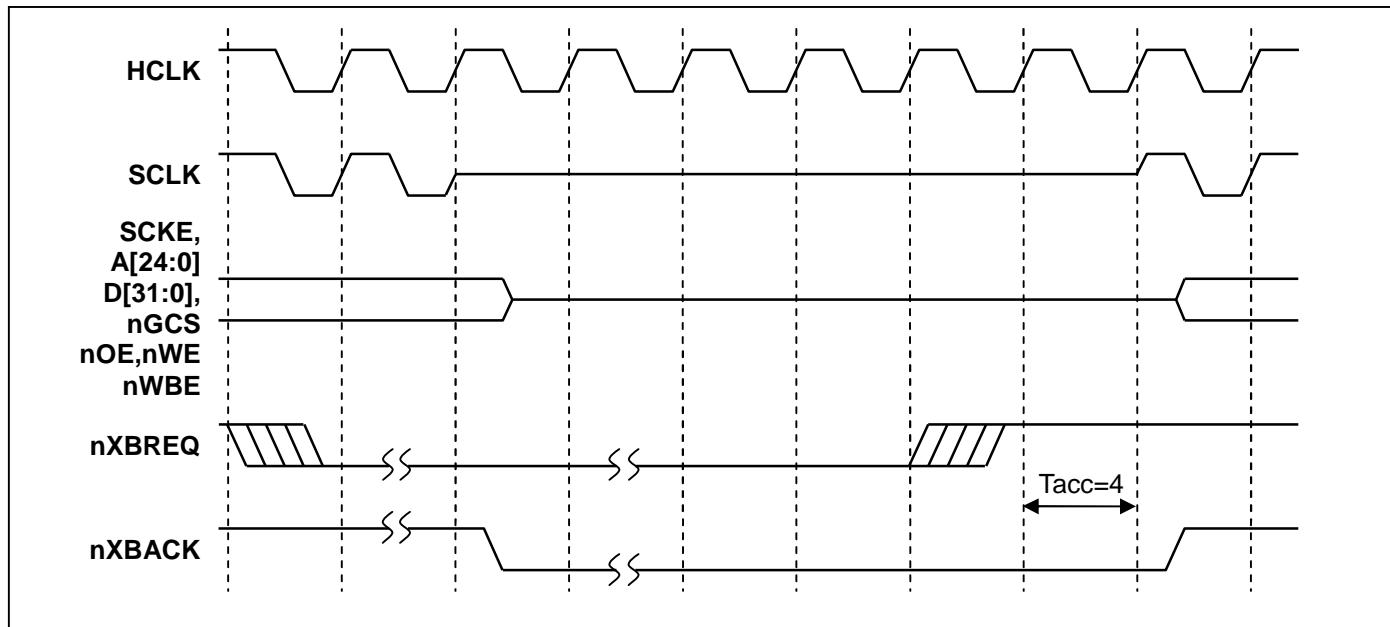


图 5-3. S3C2440A nXBREQ/nXBACK 时序示意图

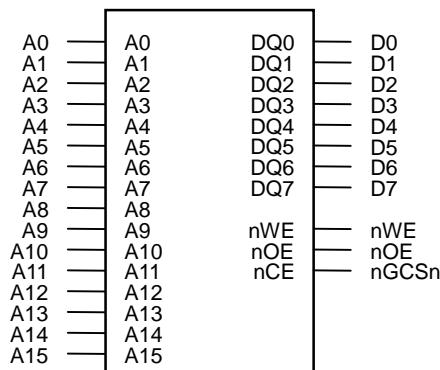
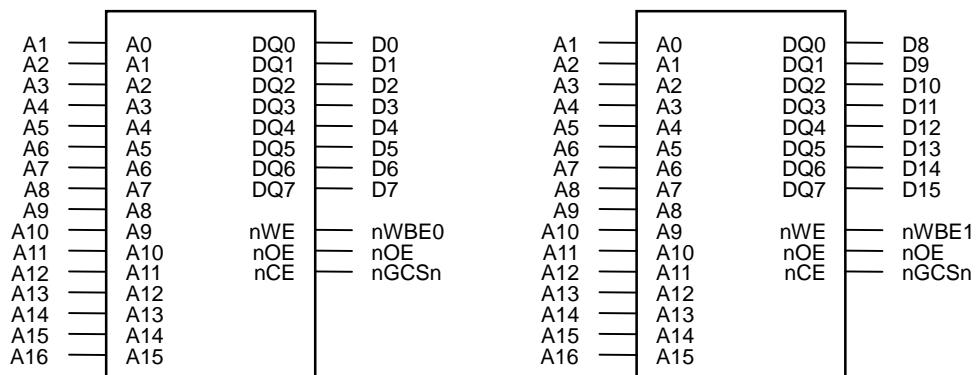
ROM 存储器接口例子

图 5-4. 8 位 ROM 存储器接口

图 5-5. 8 位 ROM \times 2 存储器接口

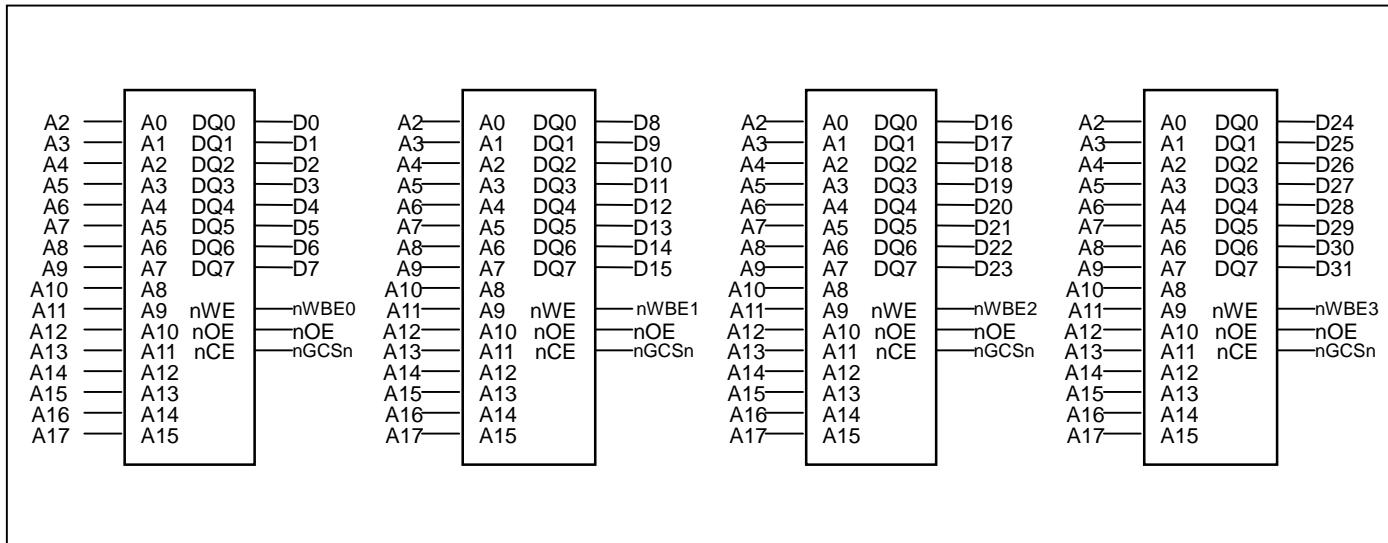
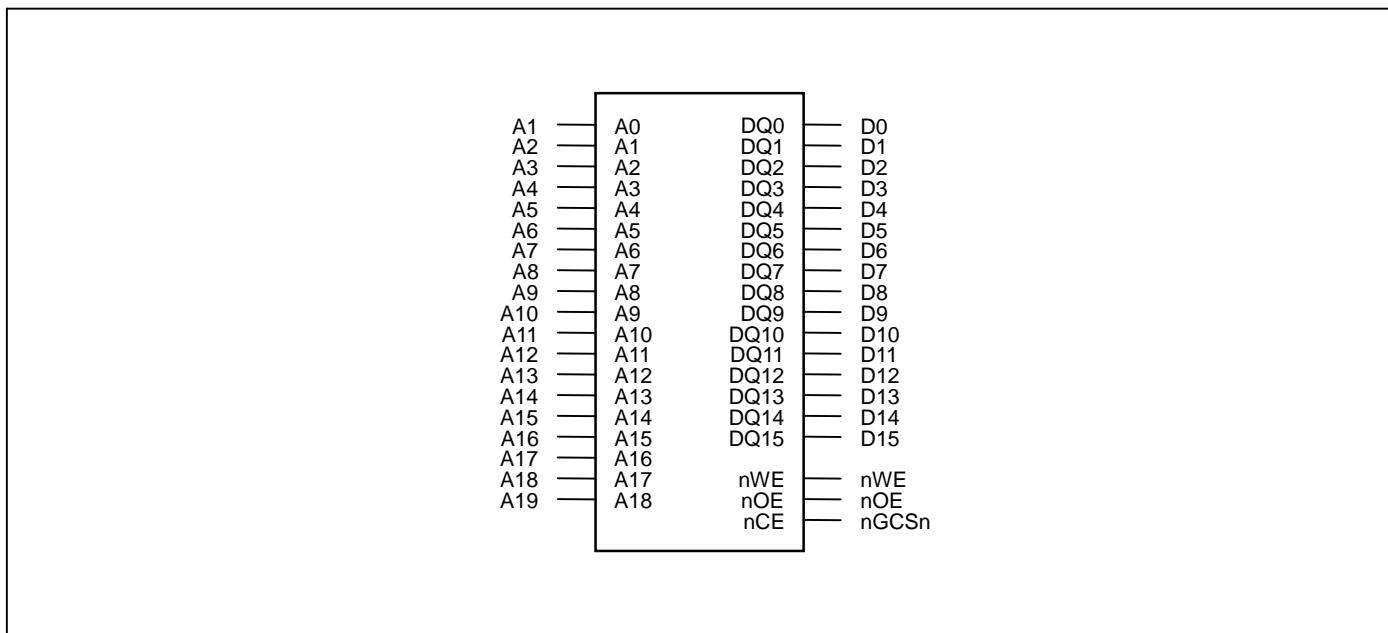
图 5-6. 8 位 ROM \times 4 存储器接口

图 5-7. 16 位 ROM 存储器接口

SRAM 存储器接口例子

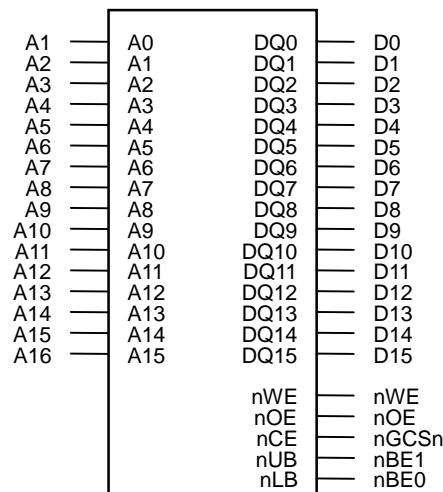


图 5-8. 16 位 SRAM 存储器接口

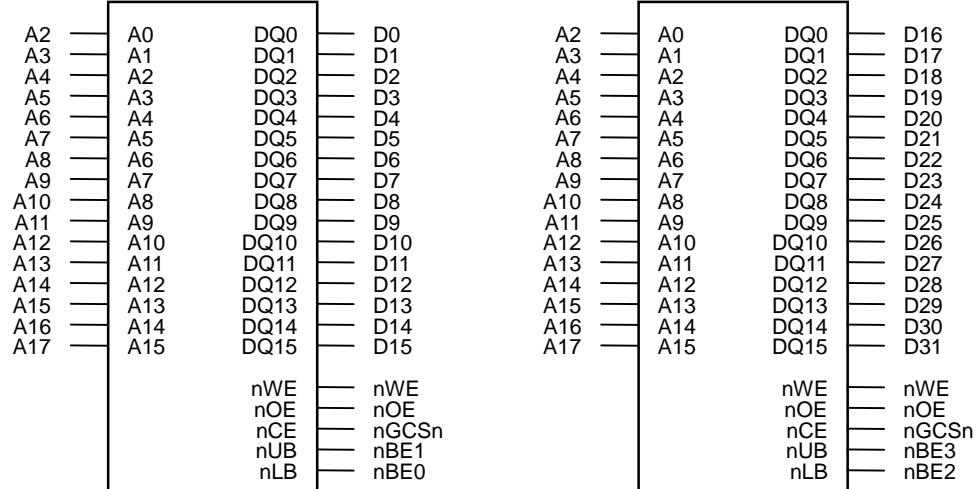


图 5-9. 16 位 SRAM \times 2 存储器接口

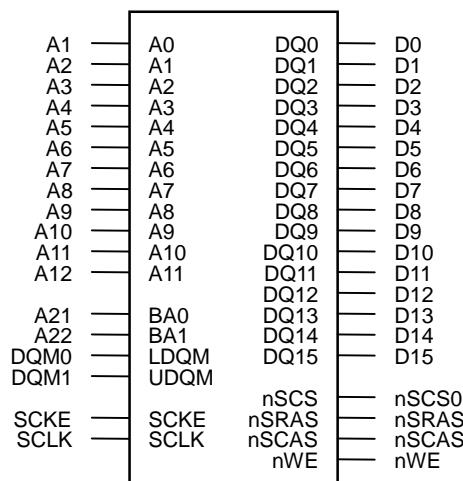


图 5-10. 16 位 SDRAM (4Mx16 , 4 个 Bank) 存储器接口

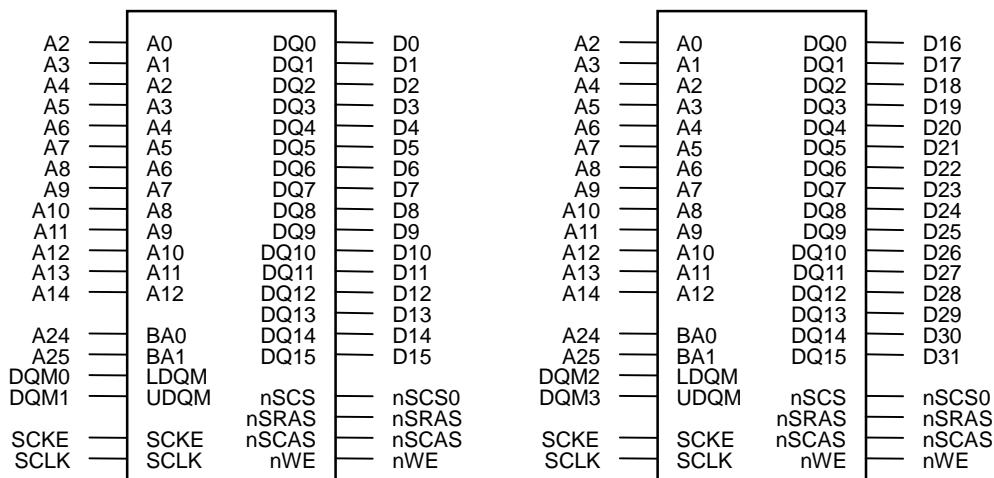


图 5-11. 16 位 SDRAMx2 (4Mx16x4 个 Bankx2ea) 存储器接口

注释：

参考表 5-2 的 SDRAM 的 Bank 地址的配置。

可编程访问周期

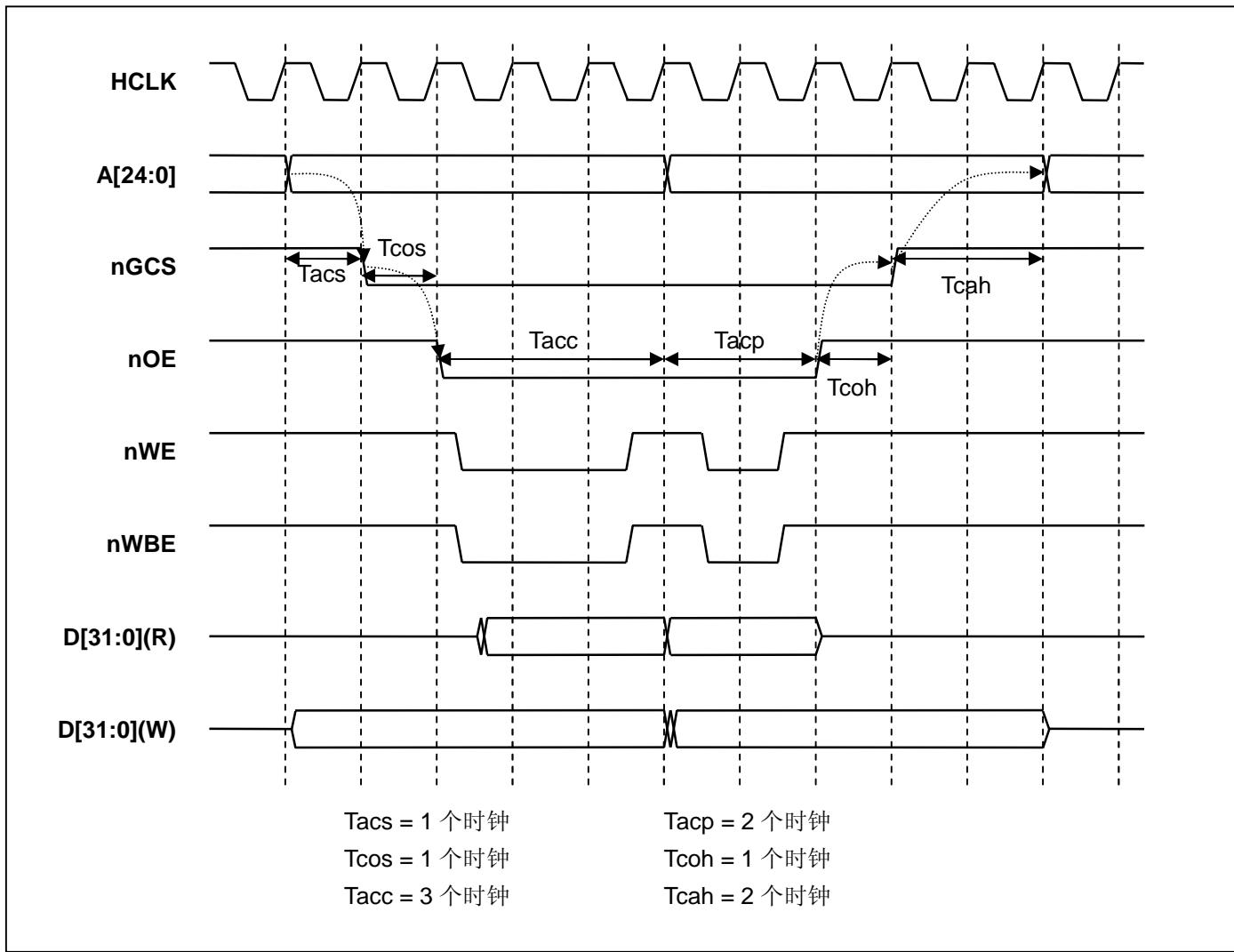


图 5-12. S3C2440A nGCS 时序示意图

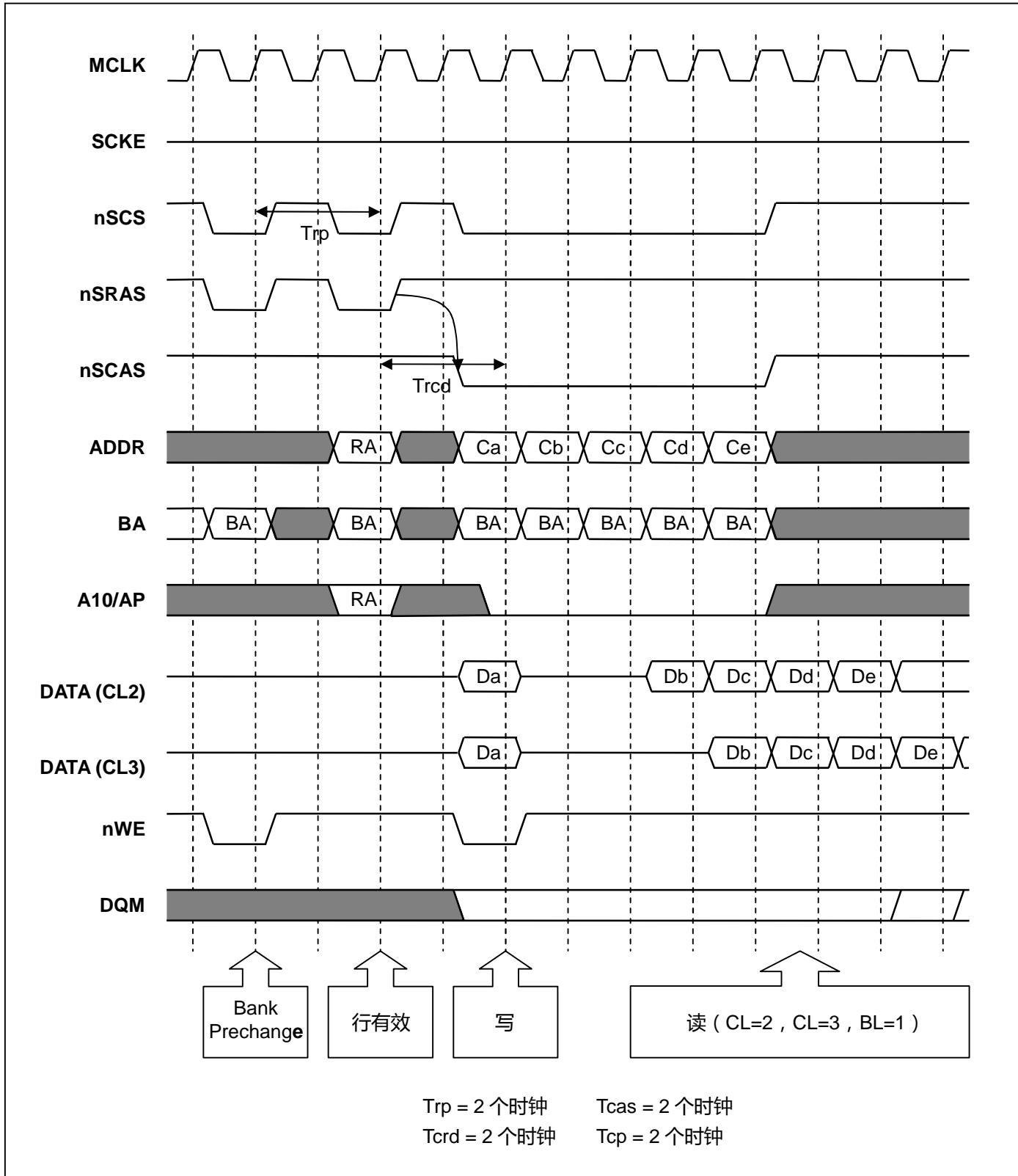


图 5-13. S3C2440A SDRAM 时序示意图

总线宽度和等待控制寄存器 (BWSCON)

寄存器	地址	R/W	描述	复位值
BWSCON	0x48000000	R/W	总线宽度和等待控制寄存器	0x0000000

BWSCON	位	描述				初始状态
ST7	[31]	决定 SRAM 是否对 Bank 7 使用 UB/LB 0 = 未使用 UB/LB (引脚对应 nWBE[3:0]) 1 = 使用 UB/LB (引脚对应 nBE[3:0])				0
WS7	[30]	决定 Bank 7 的 WAIT 状态 0 = WAIT 禁止 1 = WAIT 使能				0
DW7	[29:28]	决定 Bank 7 的数据总线宽度 00 = 8 位 01 = 16 位 10 = 32 位 11 = 保留				0
ST6	[27]	决定 SRAM 是否对 Bank 6 使用 UB/LB 0 = 未使用 UB/LB (引脚对应 nWBE[3:0]) 1 = 使用 UB/LB (引脚对应 nBE[3:0])				0
WS6	[26]	决定 Bank 6 的 WAIT 状态 0 = WAIT 禁止 1 = WAIT 使能				0
DW6	[25:24]	决定 Bank 6 的数据总线宽度 00 = 8 位 01 = 16 位 10 = 32 位 11 = 保留				0
ST5	[23]	决定 SRAM 是否对 Bank 5 使用 UB/LB 0 = 未使用 UB/LB (引脚对应 nWBE[3:0]) 1 = 使用 UB/LB (引脚对应 nBE[3:0])				0
WS5	[22]	决定 Bank 5 的 WAIT 状态 0 = WAIT 禁止 1 = WAIT 使能				0
DW5	[21:20]	决定 Bank 5 的数据总线宽度 00 = 8 位 01 = 16 位 10 = 32 位 11 = 保留				0
ST4	[19]	决定 SRAM 是否对 Bank 4 使用 UB/LB 0 = 未使用 UB/LB (引脚对应 nWBE[3:0]) 1 = 使用 UB/LB (引脚对应 nBE[3:0])				0
WS4	[18]	决定 Bank 4 的 WAIT 状态 0 = WAIT 禁止 1 = WAIT 使能				0
DW4	[17:16]	决定 Bank 4 的数据总线宽度 00 = 8 位 01 = 16 位 10 = 32 位 11 = 保留				0
ST3	[15]	决定 SRAM 是否对 Bank 3 使用 UB/LB 0 = 未使用 UB/LB (引脚对应 nWBE[3:0]) 1 = 使用 UB/LB (引脚对应 nBE[3:0])				0
WS3	[14]	决定 Bank 3 的 WAIT 状态 0 = WAIT 禁止 1 = WAIT 使能				0
DW3	[13:12]	决定 Bank 3 的数据总线宽度 00 = 8 位 01 = 16 位 10 = 32 位 11 = 保留				0
ST2	[11]	决定 SRAM 是否对 Bank 2 使用 UB/LB 0 = 未使用 UB/LB (引脚对应 nWBE[3:0]) 1 = 使用 UB/LB (引脚对应 nBE[3:0])				0
WS2	[10]	决定 Bank 2 的 WAIT 状态 0 = WAIT 禁止 1 = WAIT 使能				0
DW2	[9:8]	决定 Bank 2 的数据总线宽度 00 = 8 位 01 = 16 位 10 = 32 位 11 = 保留				0

总线宽度和等待控制寄存器 (BWSCON) (续)

ST1	[7]	决定 SRAM 是否对 Bank 1 使用 UB/LB 0 = 未使用 UB/LB (引脚对应 nWBE[3:0]) 1 = 使用 UB/LB (引脚对应 nBE[3:0])	0
WS1	[6]	决定 Bank 1 的 WAIT 状态 0 = WAIT 禁止 1 = WAIT 使能	0
DW1	[5:4]	决定 Bank 1 的数据总线宽度 00 = 8 位 01 = 16 位 10 = 32 位 11 = 保留	0
DW0	[2:1]	表明 Bank 1 的数据总线宽度 (只读) 01 = 16 位 10 = 32 位 该状态由 OM[1:0]引脚决定	-
保留	[0]	保留为 0	0

注释：

- 所有此存储器控制器主机时钟的类型与总线时钟是一致的。

例如，SRAM 中的 HCLK 等同于总线时钟，SDRAM 中的 SCLK 同样等同于总线时钟。此章节中（存储器控制器），一个时钟意味着一个总线时钟。

- nBE[3:0]关系“与 (AND)”信号 nWBE[3:0]和 nOE。

BANK 控制寄存器 (BANKCONn : nGCS0 至 nGCS5)

寄存器	地址	R/W	描述	复位值
BANKCON0	0x48000004	R/W	Bank0 控制寄存器	0x0700
BANKCON1	0x48000008	R/W	Bank1 控制寄存器	0x0700
BANKCON2	0x4800000C	R/W	Bank2 控制寄存器	0x0700
BANKCON3	0x48000010	R/W	Bank3 控制寄存器	0x0700
BANKCON4	0x48000014	R/W	Bank4 控制寄存器	0x0700
BANKCON5	0x48000018	R/W	Bank5 控制寄存器	0x0700

BANKCONn	位	描述	初始状态
Tacs	[14:13]	nGCSn 前的地址建立时间 00 = 0 个时钟 01 = 1 个时钟 10 = 2 个时钟 11 = 4 个时钟	00
Tcos	[12:11]	nOE 前的片选建立时间 00 = 0 个时钟 01 = 1 个时钟 10 = 2 个时钟 11 = 4 个时钟	00
Tacc	[10:8]	访问周期 000 = 1 个时钟 001 = 2 个时钟 010 = 3 个时钟 011 = 4 个时钟 100 = 6 个时钟 101 = 8 个时钟 110 = 10 个时钟 111 = 14 个时钟 注意：当使用 nWAIT 信号时，Tacc \geq 4 个时钟	111
Tcoh	[7:6]	nOE 后的片选保持时间 00 = 0 个时钟 01 = 1 个时钟 10 = 2 个时钟 11 = 4 个时钟	00
Tcah	[5:4]	nGCSn 后的地址保持时间 00 = 0 个时钟 01 = 1 个时钟 10 = 2 个时钟 11 = 4 个时钟	00
Tacp	[3:2]	Page 模式下的 Page 模式访问周期 00 = 0 个时钟 01 = 1 个时钟 10 = 2 个时钟 11 = 4 个时钟	00
PMC	[1:0]	Page 模式配置 00 = 正常 (1 个数据) 01 = 4 个数据 10 = 8 个数据 11 = 16 个数据	00

BANK 控制寄存器 (BANKCONn : nGCS6 至 nGCS7)

寄存器	地址	R/W	描述	复位值
BANKCON6	0x4800001C	R/W	Bank6 控制寄存器	0x18008
BANKCON7	0x48000020	R/W	Bank7 控制寄存器	0x18008

BANKCONn	位	描述	初始状态
MT	[16:15]	决定 Bank6 和 Bank7 的存储器类型 00 = ROM 或 SRAM 10 = 保留 (不要使用) 01 = 保留 (不要使用) 11 = 同步 DRAM	11
存储器类型 = ROM 或 SRAM [MT=00] (15 位)			
Tacs	[14:13]	nGCSn 前的地址建立时间 00 = 0 个时钟 10 = 2 个时钟 00 = 0 个时钟 10 = 2 个时钟	00
Tcos	[12:11]	nOE 前的片选建立时间 00 = 0 个时钟 10 = 2 个时钟 00 = 0 个时钟 10 = 2 个时钟	00
Tacc	[10:8]	访问周期 000 = 1 个时钟 010 = 3 个时钟 100 = 6 个时钟 110 = 10 个时钟 001 = 2 个时钟 011 = 4 个时钟 101 = 8 个时钟 111 = 14 个时钟	111
Tcoh	[7:6]	nOE 后的片选保持时间 00 = 0 个时钟 10 = 2 个时钟 00 = 0 个时钟 10 = 2 个时钟	00
Tcah	[5:4]	nGCSn 后的地址保持时间 00 = 0 个时钟 10 = 2 个时钟 00 = 0 个时钟 10 = 2 个时钟	00
Tacp	[3:2]	Page 模式下的 Page 模式访问周期 00 = 0 个时钟 10 = 2 个时钟 00 = 0 个时钟 10 = 2 个时钟	00
PMC	[1:0]	Page 模式配置 00 = 正常 (1 个数据) 10 = 8 个数据 00 = 正常 (1 个数据) 10 = 8 个数据	00
存储器类型 = SDRAM [MT=11] (4 位)			
Trcd	[3:2]	RAS 到 CAS 的延迟 00 = 2 个时钟 01 = 3 个时钟 10 = 4 个时钟	10
SCAN	[1:0]	列地址数 00 = 8 位 01 = 9 位 10 = 10 位	00

刷新控制寄存器

寄存器	地址	R/W	描述	复位值
REFRESH	0x48000024	R/W	SDRAM 刷新控制寄存器	0xAC0000

BANKCONn	位	描述	初始状态
REFEN	[23]	SDRAM 刷新使能 0 = 禁止 1 = 使能 (自或 CBR/自动刷新)	1
TREFMD	[22]	SDRAM 刷新模式 0 = CBR/自动刷新 1 = 自刷新 在自刷新期间，驱动 SDRAM 控制信号为适当电平	0
Trp	[21:20]	SDRAM RAS 预充电时间 00 = 2 个时钟 01 = 3 个时钟 10 = 4 个时钟 11 = 不支持	10
Tsrc		SDRAM 半行周期时间 00 = 4 个时钟 01 = 5 个时钟 10 = 6 个时钟 11 = 7 个时钟 SDRAM 行周期时间 : Trc=Trc+Trp 如果 Trp=3 个时钟并且 Tsrc=7 个时钟，则 Trc=3+7=10 个时钟	11
保留	[17:16]	未使用	00
保留	[15:11]	未使用	0000
刷新计数器	[10:0]	SDRAM 刷新计数值。参阅第 6 章 SDRAM 刷新控制器总线优先级部分。 刷新周期 = $(2^{11} - \text{刷新计数} + 1) / \text{HCLK}$ 例. 如果刷新周期为 7.8 μs 并且 HCLK 为 100 MHz, 刷新计数如下： $\text{刷新计数} = 2^{11} + 1 - 100 \times 7.8 = 1269$	0

Bank 大小寄存器

寄存器	地址	R/W	描述	复位值
BANKSIZE	0x48000028	R/W	可变 Bank 大小寄存器	0x0

BANKCONn	位	描述	初始状态
BURST_EN	[7]	ARM 核突发 (Burst) 操作使能 0 = 禁止突发操作 1 = 使能突发操作	0
保留	[6]	未使用	0
SCKE_EN	[5]	SDRAM 掉电模式使能 SCKE 控制 0 = 禁止 SDRAM 掉电模式 1 = 使能 SDRAM 掉电模式	0
SCLK_EN	[4]	只在 SDRAM 访问周期期间 SCLK 使能，以降低功耗。当未访问 SDRAM， SCLK 变为低电平 0 = SCLK 一直有效 1 = SCLK 只在访问期间有效 (推荐)	0
保留	[3]	未使用	0
BK76MAP	[2:0]	Bank6/7 存储器映射 010 = 128MB/128MB 001 = 64MB/64MB 000 = 32M/32M 111 = 16M/16M 110 = 8M/8M 101 = 4M/4M 100 = 2M/2M	010

SDRAM 模式 寄存器组寄存器 (MRSR)

寄存器	地址	R/W	描述	复位值
MRSRB6	0x4800002C	R/W	模式寄存器组寄存器 Bank6	XXX
MRSRB7	0x48000030	R/W	模式寄存器组寄存器 Bank7	XXX

BANKCONn	位	描述	初始状态
保留	[11:10]	未使用	-
WBL	[9]	写突发长度 0 = 突发 (固定) 1 = 保留	X
TM	[8:7]	测试模式 00 = 模式寄存器组 (固定) 01 = 保留 00 = 保留 00 = 保留	XX
CL	[6:4]	CAS 等待时间 (<i>latency</i>) 000 = 1 个时钟 010 = 2 个时钟 011 = 3 个时钟 其它 = 保留	XXX
BT	[3]	突发类型 0 = 连续 (固定) 1 = 保留	X
BL	[2:1]	突发长度 000 = 1 (固定) 其它 = 保留	XXX

注意：

当代码在 SDRAM 中运行时一定不要改变 MRSR 寄存器。

重要注意：

睡眠模式中，SDRAM 必须使能 SDRAM 自刷新模式。

6 NAND FLASH 控制器

概述

目前的 NOR Flash 存储器价格较高，相对而言 SDRAM 和 NAND Flash 存储器更经济，这样促使了一些用户在 NAND Flash 中执行引导代码，在 SDRAM 中执行主代码。

S3C2440A 引导代码可以在外部 NAND Flash 存储器上执行。为了支持 NAND Flash 的 BootLoader ,S3C2440A 配备了一个内置的 SRAM 缓冲器，叫做 “Steppingstone”。引导启动时，NAND Flash 存储器的开始 4K 字节将被加载到 Steppingstone 中并且执行加载到 Steppingstone 的引导代码。

通常引导代码会复制 NAND Flash 的内容到 SDRAM 中。通过使用硬件 ECC，有效地检查 NAND Flash 数据。在复制完成的基础上，将在 SDRAM 中执行主程序。

特性

- 1) 关于引导启动 : 引导代码在复位期间被传送到 4K 字节的 Steppingstone。传送后，引导代码将在 Steppingstone 中执行。
- 2) NAND Flash 存储器接口 : 支持 256 字，512 字节，1K 字和 2K 字节页。
- 3) 软件模式 : 用户可以直接访问 NAND Flash 存储器，例如此特性可用于 NAND Flash 存储器的读/擦除/编程。
- 4) 接口 : 8/16 位 NAND Flash 存储器接口总线
- 5) 硬件 ECC 生成，检测和指示 (软件纠错)。
- 6) SFR I/F : 支持小端模式是按字节/半字/字访问数据和 ECC 数据寄存器，和按字访问其它寄存器。
- 7) SteppingStone 接口 : 支持大/小端模式的按字节/半字/字访问。
- 8) SteppingStone 4KB 内部 SRAM 缓冲器可以在 NAND Flash 引导启动后用于其他用途。

方框图

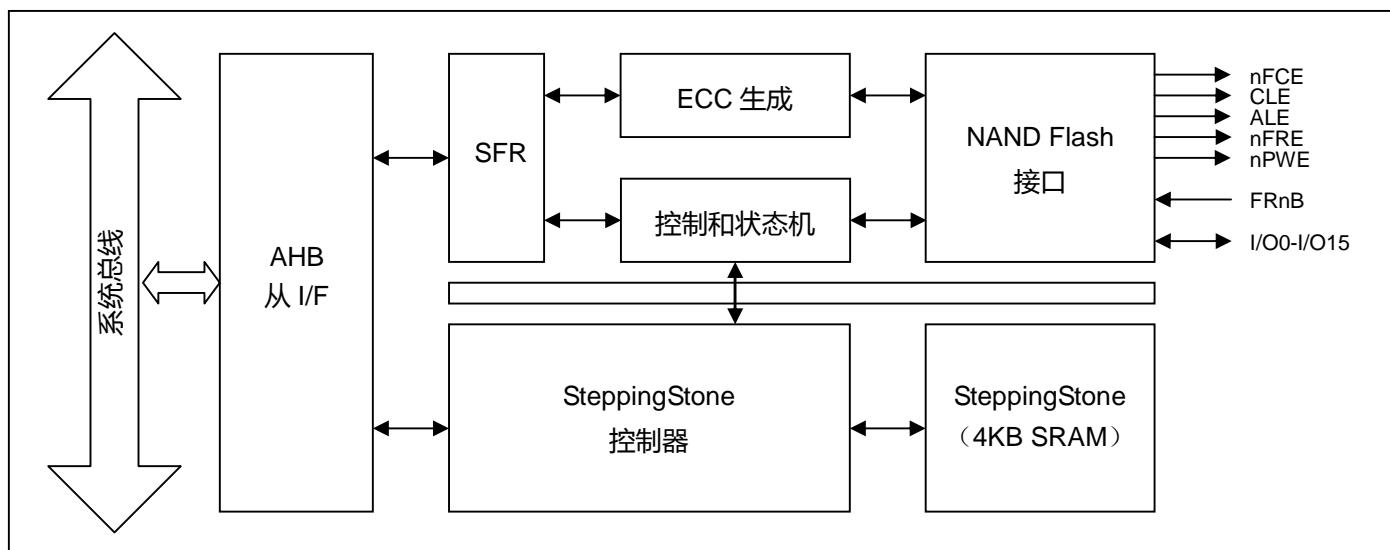


图 6-1 NAND Flash 控制器方框示意图

引导加载 (BootLoader) 功能

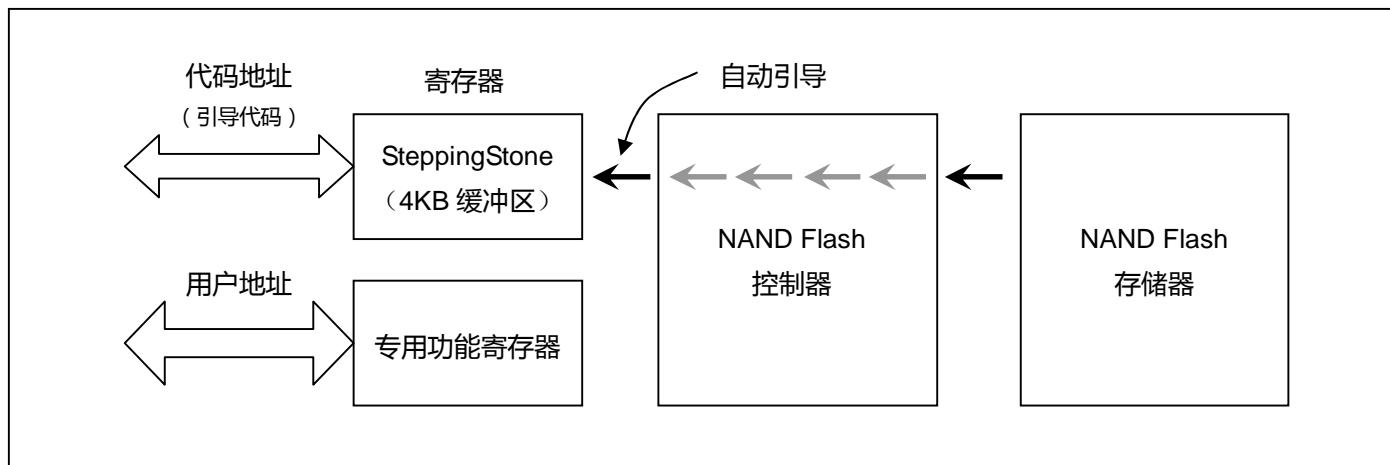


图 6-2 NAND Flash 控制器 Boot Loader 方框示意图

当复位时，NAND Flash 控制器将通过引脚状态 (NCON(先进闪存), GPG13(页大小), GPG14(地址周期) , GPG15(总线宽度)—请参考引脚配置) 来获取连接的 NAND Flash 的信息，在发生掉电或系统复位后，NAND Flash 控制器自动加载 4K 字节的 BootLoader 代码。在加载完 BootLoader 代码后，Steppingstone 中的 BootLoader 代码已经执行了。

注释：

当自动引导启动期间，ECC 不会去检测，所以，NAND Flash 的开始 4KB 不应当包含位相关的错误。

引脚配置

- OM[1:0] = 00 : 使能 NAND Flash 存储器引导启动
- NCON : NAND Flash 存储器选择 (普通/先进)
 - 0 : 普通 NAND Flash (256 字或 512 字节页大小 , 3 或 4 个地址周期)
 - 1 : 先进 NAND Flash (1K 字或 2K 字节页大小 , 4 或 5 个地址周期)
- GPG13 : NAND Flash 存储器页面容量选择
 - 0 : 页=256 字 (NCON=0) 或页=1K 字 (NCON=1)
 - 1 : 页=512 字节 (NCON=0) 或页=2K 字节 (NCON=1)
- GPG14 : NAND Flash 存储器地址周期选择
 - 0 : 3 个地址周期 (NCON=0) 或 4 个地址周期 (NCON=1)
 - 1 : 4 个地址周期 (NCON=0) 或 5 个地址周期 (NCON=1)
- GPG15 : NAND Flash 存储器总线宽度选择
 - 0 : 8 位宽度
 - 1 : 16 位宽度

注释 :

配置引脚 NCON, GPG[15:13]将在复位期间被取出 (*fetched*)。在正常状态下, 这些引脚必须设置为输入, 这是为了在由软件或意外情况而进入睡眠模式时, 这些引脚的状态不被改变。

NAND Flash 存储器配置列表

NCON0	GPG13	GPG14	GPG15
0 : 普通 NAND	0 : 256 字	0 : 3 个地址周期	0 : 8 位总线宽度
	1 : 512 字节	1 : 4 个地址周期	
0 : 先进 NAND	0 : 1K 字	0 : 4 个地址周期	1 : 16 位总线宽度
	1 : 2K 字节	1 : 5 个地址周期	

注释 :

关于超过 4 位, 可能总计组合为 16, 但不是所有的值都能使用

例 : NAND Flash 配置设置

器件	页面大小/总计大小	NCON0	GPG13	GPG14	GPG15
K9S1208V0M-xxxx	512 字节 / 512M 比特	0	1	1	0
K9K2G16U0M-xxxx	1K 字 / 2G 比特	1	0	1	1

NAND Flash 存储器时序

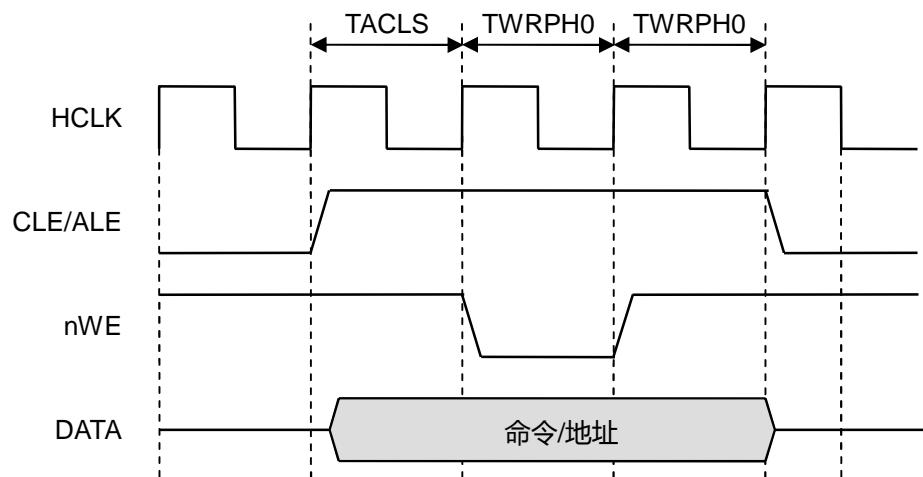


图 6-3. CLE 和 ALE 时序 ($TACLS=1$, $TWRPH0=0$, $TWRPH1=0$)

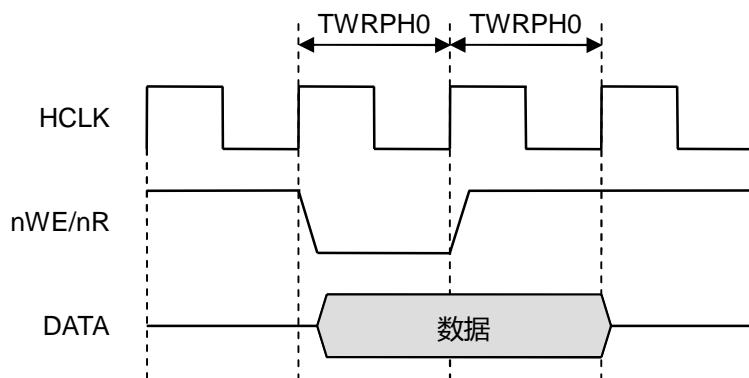


图 6-4 nWE 和 nRE 时序 ($TWRPH0=0$, $TWRPH1=0$)

软件模式

S3C2440A 只支持软件模式的访问。使用该模式，你可以完整的访问访问 NAND Flash 存储器。NAND Flash 控制器支持 NAND Flash 存储器的直接访问接口。

- 1) 写命令寄存器=NAND Flash 存储器命令周期
- 2) 写地址寄存器=NAND Flash 存储器地址周期
- 3) 写数据寄存器=写入数据到 NAND Flash 存储器 (写周期)
- 4) 读数据寄存器= 从 NAND Flash 存储器读取数据 (读周期)
- 5) 读主 ECC 寄存器和备份 ECC 寄存器=从 NAND Flash 存储器读取数据

注释：

软件模式下，你必须用定时查询或中断来检测 RnB 状态输入引脚。

数据寄存器配置

1) 16 位 NAND Flash 存储器接口

A 字访问

寄存器	大/小端	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFDATA	小端	2 nd I/O[15:8]	2 nd I/O[7:0]	1 st I/O[15:8]	1 st I/O[7:0]
NFDATA	大端	1 st I/O[15:8]	1 st I/O[7:0]	2 nd I/O[15:8]	2 nd I/O[7:0]

B 半字访问

寄存器	大/小端	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFDATA	大/小端	无效值	无效值	1 st I/O[15:8]	1 st I/O[7:0]

2) 8 位 NAND Flash 存储器接口

A 字访问

寄存器	大/小端	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFDATA	小端	4 th I/O[7:0]	3 rd I/O[7:0]	2 nd I/O[7:0]	1 st I/O[7:0]
NFDATA	大端	1 st I/O[7:0]	2 nd I/O[7:0]	3 rd I/O[7:0]	4 th I/O[7:0]

B 半字访问

寄存器	大/小端	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFDATA	小端	无效值	无效值	2 nd I/O[7:0]	1 st I/O[7:0]
NFDATA	大端	无效值	无效值	1 st I/O[7:0]	2 nd I/O[7:0]

C 字节访问

寄存器	大/小端	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFDATA	大/小端	无效值	无效值	无效值	1 st I/O[7:0]

SteppingStone (4K 字节 SRAM)

NAND Flash 控制器在引导启动时将 Steppingstone 作为缓冲器，你也可以使用此区域为其它用途。

EEC (错误纠正码)

NAND Flash 控制器由 4 个 ECC (错误纠正码) 模块组成。其中两个 ECC 模块 (一个用于 data[7:0] 另一个用于 data[15:8]) 可以被用于 (多达) 2048 字节的 ECC 奇偶校验码的生成 , 其它的 (一个用于 data[7:0] 另一个用于 data[15:8]) 可以被用于 (多达) 16 字节的 ECC 奇偶校验码的生成。

28 位 ECC 奇偶校验码 =22 位行奇偶校验 +6 位列

14 位 ECC 奇偶校验码 =8 位行奇偶校验 +6 位列

2048 字节 ECC 奇偶校验码分配表

	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
MECCn_0	P64	P64'	P32	P32'	P16	P16'	P8	P8'
MECCn_1	P1024	P1024'	P512	P512'	P256	P256'	P128	P128'
MECCn_2	P4	P4'	P2	P2'	P1	P1'	P2048	P2048'
MECCn_3	P8192	P8192'	P4096	P4096'	-	-	-	-

16 字节 ECC 奇偶校验码分配表

	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
SECCn_0	P16	P16'	P8	P8'	P4	P4'	P2	P2'
SECCn_1	P1	P1'	P64	P64'	P32	P32'	-	-

ECC 模块特性

ECC 生成由控制寄存器的 ECC 锁 (主 ECCL 锁 , 从 ECCL 锁) 位来控制。

ECC 寄存器配置 (大 / 小端)

1) 16 位 NAND Flash 存储器接口

寄存器	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFMECCD0	I/O[15:8]的 2 nd ECC	I/O[7:0]的 2 nd ECC	I/O[15:8]的 1 st ECC	I/O[7:0]的 1 st ECC
NFMECCD1	I/O[15:8]的 4 th ECC	I/O[7:0]的 4 th ECC	I/O[15:8]的 3 rd ECC	I/O[7:0]的 3 rd ECC

寄存器	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFSECCD	I/O[15:8]的 2 nd ECC	I/O[7:0]的 2 nd ECC	I/O[15:8]的 1 st ECC	I/O[7:0]的 1 st ECC

2) 8 位 NAND Flash 存储器接口

寄存器	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFMECCD0	-	I/O[7:0]的 2 nd ECC	-	I/O[7:0]的 1 st ECC
NFMECCD1	-	I/O[7:0]的 4 th ECC	-	I/O[7:0]的 3 rd ECC

寄存器	位[31:24]	位[2316]	位[15:8]	位[7:0]
NFSECCD	-	I/O[7:0]的 2 nd ECC	-	I/O[7:0]的 1 st ECC

ECC 编程指南

- 1) 软件模式下 , ECC 模块生成 ECC 奇偶校验码用于读/写数据。因此你可以写入 InitECC (NFCONT[4]) 位为'1' 并在读或写数据之前清除 MainECCLock (NFCONT[5]) 位为'0' (开锁) 来复位 ECC 的值。MainECCLock (NFCONT[5]) 和 SpareECCLock (NFCONT[6]) 是控制是否生成 ECC 奇偶校验码。
- 2) 每当读取或写入数据时 , ECC 模块将生成 ECC 奇偶校验码到寄存器 NFMECC0/1 中。
- 3) 当你完成了读或写一页以后 (不包括备份区域数据), 设置 MainECCLock 位为'1' (锁) 。ECC 奇偶校验码被锁住并且 ECC 状态寄存器的值也将不会被改变。
- 4) 要生成备份区域 ECC 奇偶校验码 , 清除 SpareECCLock (NFCONT[6]) 位为'0' (开锁) 即可。
- 5) 每当读取或写入数据时 , 备份区域 ECC 模块生成 ECC 奇偶校验码到寄存器 NFSECC 中。
- 6) 当你完成了读或写备份区域后 , 设置 SpareECCLock 位为'1' (锁) 。ECC 奇偶校验码被锁住并且 ECC 状态寄存器的值也将不会被改变。
- 7) 每完成一次你可以用这些值来标记备份区域或检查位错误。

注释 :

NFSECCD 是用于 ECC 在备份区域 (通常用户)

NAND Flash 存储器映射

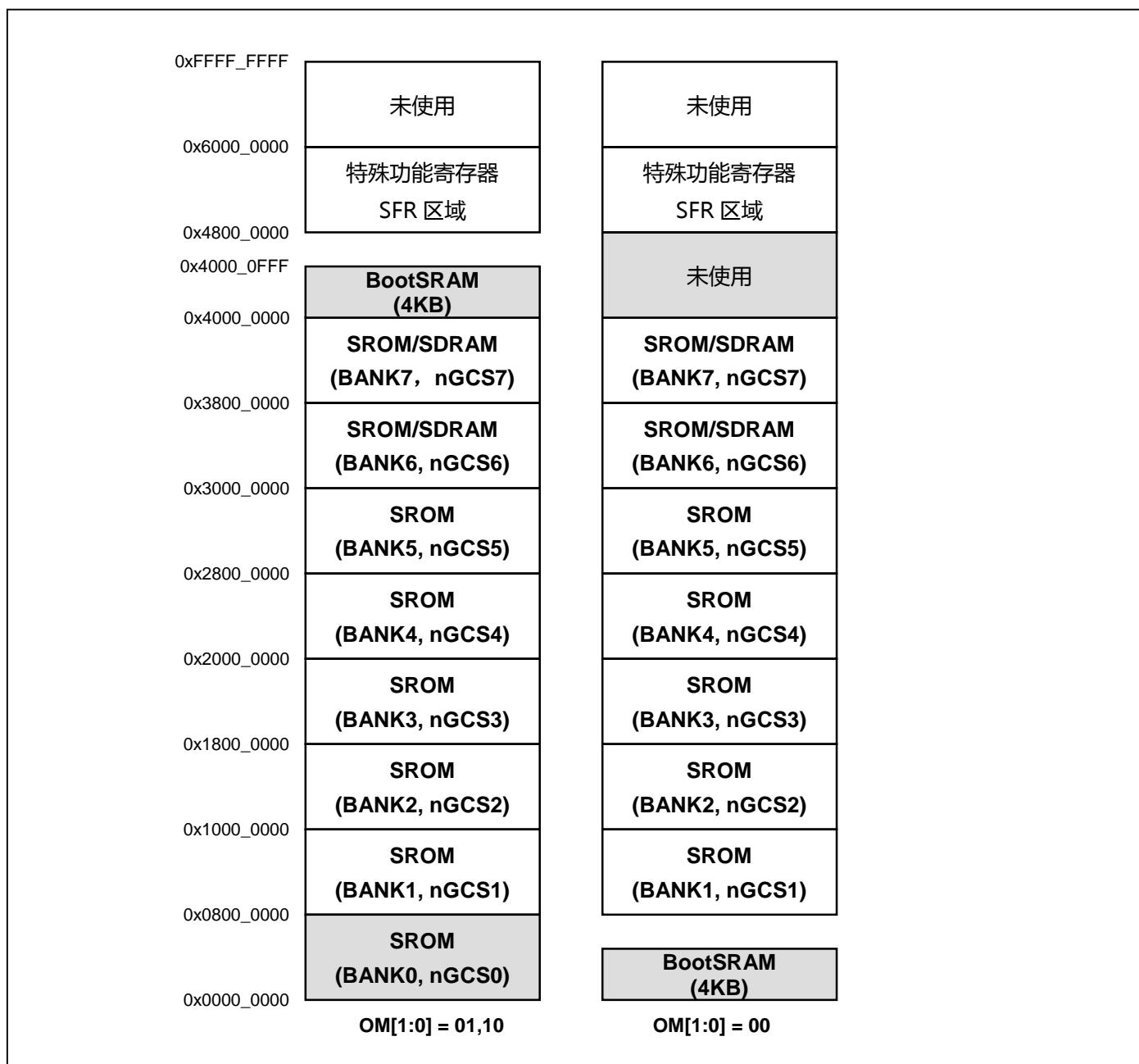


图 6-5. NAND Flash 存储器映射

注释：

SROM 是 ROM 或 SRAM 类型存储器的意思。

NAND Flash 存储器配置

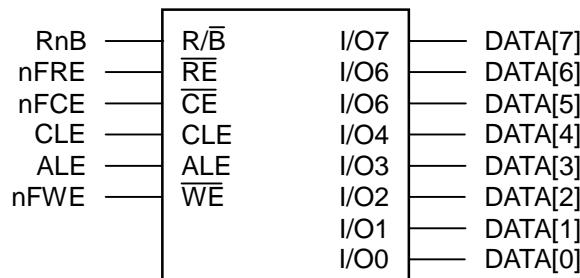


图 6-6 一个 8 位 NAND Flash 存储器接口

当你写地址时，从 data[7:0]和 data[15:8]发出相同的地址。

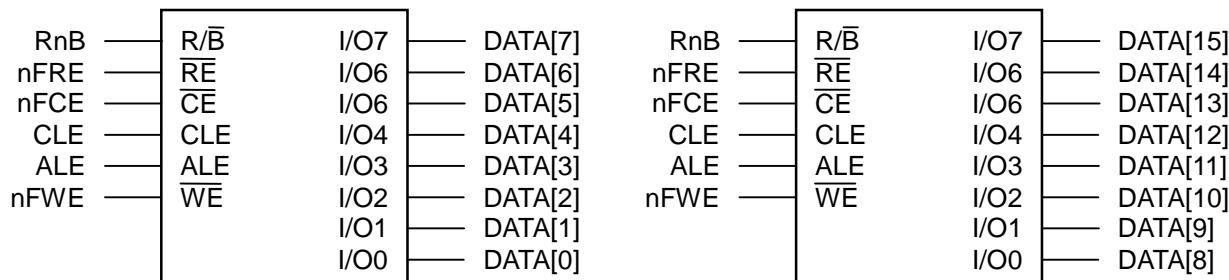


图 6-7 两个 8 位 NAND Flash 存储器接口

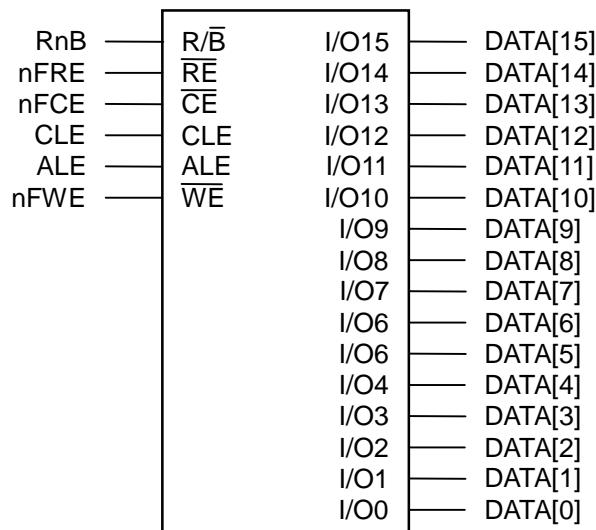


图 6-8 一个 16 位 NAND Flash 存储器接口

NAND Flash 配置寄存器

寄存器	地址	R/W	描述	复位值
NFCNF	0x4E000000	R/W	NAND Flash 配置寄存器	0x0000100X

NFCNF	位	描述	初始状态
保留	[15:14]	保留	-
TACLS	[13:12]	CLE 和 ALE 持续值设置 (0 至 3) Duration = HCLK × TACLS	01
保留	[11]	保留	0
TWRPH0	[10:8]	TWRPH0 持续值设置 (0~7) Duration = HCLK × (TWRPH0 + 1)	000
保留	[7]	保留	0
TWRPH1	[6:4]	TWRPH1 持续值设置 (0~7) Duration = HCLK × (TWRPH1 + 1)	000
AdvFlash (只读)	[3]	自动引导启动用的先进 NAND Flash 存储器。 0：支持 256 字或 512 字节/页的 NAND Flash 存储器； 1：支持 1K 字或 2K 字节/页的 NAND Flash 存储器。 此位由在复位和从睡眠模式中唤醒时的 NCON0 引脚状态所决定。	硬件设置 (NCON0)
PageSize (只读)	[2]	自动引导启动用的 NAND Flash 存储器的页面大小。 先进闪存页面大小 当 AdvFlash 为 0 时 0=256 字/页；1=512 字节/页 当 AdvFlash 为 1 时 0=1024 字/页；1=2048 字节/页 此位由在复位和从睡眠模式中唤醒时的 GPG13 引脚状态所决定。 复位后，GPG13 可以用于通用 I/O 口或外部中断。	硬件设置 (GPG13)
AddrCycle (只读)	[1]	自动引导启动用的 NAND Flash 存储器的地址周期。 先进闪存地址周期 当 AdvFlash 为 0 时 0=3 个地址周期；1=4 个地址周期 当 AdvFlash 为 1 时 0=4 个地址周期；1=5 个地址周期 此位由在复位和从睡眠模式中唤醒时的 GPG14 引脚状态所决定。 复位后，GPG14 可以用于通用 I/O 口或外部中断。	硬件设置 (GPG14)
BusWidth (R/W)	[0]	自动引导启动和普通访问用的 NAND Flash 存储器的输入输出总线宽度。 0=8 位总线；1=16 位总线 此位由在复位和从睡眠模式中唤醒时的 GPG15 引脚状态所决定。 复位后，GPG15 可以用于通用 I/O 口或外部中断。 此位可以被软件改变。	硬件设置 (GPG15)

控制寄存器

寄存器	地址	R/W	描述	复位值
NFCONT	0x4E000004	R/W	NAND Flash 控制寄存器	0x0384

NFCONT	位	描述	初始状态
保留	[15:14]	保留	0
Lock-tight	[13]	紧锁配置 (<i>Lock-tight</i>) 0 : 禁止紧锁 1 : 使能紧锁 只要此位被设置为 1 一次，你就不能清除了。只有复位或从睡眠模式中被唤醒才能使此位为禁止(即不能由软件清零)。当此位被设置为 1 的情况下，在 NFSBLK(0x4E000038) 到 NFEBLK(0x4E00003C)-1 的区域设置未被上锁，除了这些区域以外的区域，写入或擦除命令将会无效，只有只读命令有效。	0
Soft Lock	[12]	软件上锁设置 0 : 禁止上锁 1 : 使能上锁 软件锁定区域可以随时用软件修改。当此位被设置为 1 的情况下，在 NFSBLK(0x4E000038) 到 NFEBLK(0x4E00003C)-1 的区域设置未被上锁，除了这些区域以外的区域，写入或擦除命令将会无效，只有只读命令有效。当你试图写入或擦除这些锁定区域时，将发生非法访问 (NFSTAT[3] 位将会置位)。如果 NFSBLK 和 NFEBLK 相同时，整个区域都被锁定。	1
保留	[11]	保留	0
EnbIllegalAccINT	[10]	非法访问中断控制 0 : 禁止中断 1 : 使能中断 当 CPU 试图编程或擦除锁定区域 (由 NFSBLK(0x4E000038) 到 NFEBLK(0x4E00003C)-1 的区域设置) 而产生非法访问中断。	0
EnbRnBINT	[9]	RnB 状态输入信号传输中断控制 0 : 禁止 RnB 中断 1 : 使能 RnB 中断	0
RnB_TransMode	[8]	RnB 传输检测配置 0 : 检测上升沿 1 : 检测下降沿	0
保留	[7]	保留	0
SpareECCLock	[6]	锁定备份区域 ECC 产生 0 : 开锁备份 ECC 1 : 锁定备份 ECC 备份区域 ECC 寄存器为 NFSECC(0x4E000034)。	1
MainECCLock	[5]	锁定主数据区域 ECC 生成 0 : 开锁主数据区域 ECC 生成 1 : 锁定主数据区域 ECC 生成 主数据区域 ECC 状态寄存器为 NFMECC0/1(0x4E00002C/30)。	1
InitECC	[4]	初始化 ECC 编码器/译码器 (只写) 1 : 初始化 ECC 编码器/译码器	0
保留	[3:2]	保留	00
Reg_nCE	[1]	NAND Flash 存储器 nFCE 信号控制 0 : 强制 nFCE 为低 (使能片选) 1 : 强制 nFCE 为高 (禁止片选) 注意：在引导启动期间其自动被控制。只有 MODE 位为 1 该值才有效。	1
MODE	[0]	NAND Flash 控制器运行模式 0 : NAND Flash 控制器禁止 (不工作) 1 : NAND Flash 控制器使能	0

命令寄存器

寄存器	地址	R/W	描述	复位值
NFCMMD	0x4E000008	R/W	NAND Flash 命令集寄存器	0x00

NFCMMD	位	描述	初始状态
保留	[15:8]	保留	0x00
NFCMMD	[7:0]	NAND Flash 存储器命令值	0x00

地址寄存器

寄存器	地址	R/W	描述	复位值
NFADDR	0x4E00000C	R/W	NAND Flash 地址集寄存器	0x0000XX00

REG_ADDR	位	描述	初始状态
保留	[15:8]	保留	0x00
NFADDR	[7:0]	NAND Flash 存储器地址值	0x00

数据寄存器

寄存器	地址	R/W	描述	复位值
NFDATA	0x4E000010	R/W	NAND Flash 数据寄存器	0xXXXX

NFDATA	位	描述	初始状态
NFDATA	[31:0]	NAND Flash 读取/编程数据给 I/O 注释：请参考第 6-5 页的数据寄存器配置。	0xXXXX

主数据区域寄存器

寄存器	地址	R/W	描述	复位值
NFMECCD0	0x4E000014	R/W	用于主数据读取的 NAND Flash ECC 第一和第二寄存器。 注释：请参考第 6-8 页的 ECC 模块特性 。	0x00000000
NFMECCD1	0x4E000018	R/W	用于主数据读取的 NAND Flash ECC 第三和第四寄存器。 注释：请参考第 6-8 页的 ECC 模块特性 。	0x00000000

NFMECCD0	位	描述	初始状态
ECCData1_1	[31:24]	第二个 ECC 给 I/O[15:8]	0x00
ECCData1_0	[23:16]	第二个 ECC 给 I/O[7:0] 注释：软件模式下当你需要从 NAND Flash 存储器中读取第二个 ECC 值时读取此寄存器。	0x00
ECCData0_1	[15:8]	第一个 ECC 给 I/O[15:8]	0x00
ECCData0_0	[7:0]	第一个 ECC 给 I/O[7:0] 注释：软件模式下当你需要从 NAND Flash 存储器中读取第一个 ECC 值时读取此寄存器。此寄存器与 NFDATA 有相同的读取功能。	0x00

注意：按字访问才有效。

NFMECCD1	位	描述	初始状态
ECCData3_1	[31:24]	第四个 ECC 给 I/O[15:8]	0x00
ECCData3_0	[23:16]	第四个 ECC 给 I/O[7:0] 注释：软件模式下当你需要从 NAND Flash 存储器中读取第四个 ECC 值时读取此寄存器。	0x00
ECCData2_1	[15:8]	第三个 ECC 给 I/O[15:8]	0x00
ECCData2_0	[7:0]	第三个 ECC 给 I/O[7:0] 注释：软件模式下当你需要从 NAND Flash 存储器中读取第三个 ECC 值时读取此寄存器。此寄存器与 NFDATA 有相同的读取功能。	0x00

注意：按字访问才有效。

备份区域 ECC 寄存器

寄存器	地址	R/W	描述	复位值
NFSECCD	0x4E00001C	R/W	用于备份区域数据读取的 NAND Flash ECC 寄存器。	0x00000000

NFSECCD	位	描述	初始状态
ECCData1_1	[31:24]	第二个 ECC 给 I/O[15:8]	0x00
ECCData1_0	[23:16]	第二个 ECC 给 I/O[7:0] 注释：软件模式下当你需要从 NAND Flash 存储器中读取第二个 ECC 值时读取此寄存器。	0x00
ECCData0_1	[15:8]	第一个 ECC 给 I/O[15:8]	0x00
ECCData0_0	[7:0]	第一个 ECC 给 I/O[7:0] 注释：软件模式下当你需要从 NAND Flash 存储器中读取第一个 ECC 值时读取此寄存器。此寄存器与 NFDATA 有相同的读取功能。	0x00

注意：按字访问才有效。

NFCON 状态寄存器

寄存器	地址	R/W	描述	复位值
NFSTAT	0x4E000020	R/W	NAND Flash 运行状态寄存器	0xXX00

NFSTAT	位	描述	初始状态
保留	[7]	保留	X
保留	[6:4]	保留	0
IllegalAccess	[3]	软件锁定或紧锁一次使能。非法访问(编程, 擦除)存储器屏蔽此位设置 0 : 不检测非法访问 1 : 检测非法访问	0
RnB_TransDetect	[2]	当 RnB 由低变高时发生传输, 如果使能了此位则设置和发出中断。要清除此位时对其写入'1' 0 : 不检测 RnB 传输 1 : 检测 RnB 传输 传输配置设置在 RnB_TransMode(NFCONT[8])中	0
nCE (只读)	[1]	nCE 输出引脚的状态	1
RnB (只读)	[0]	RnB 输入引脚的状态 0 : NAND Flash 存储器忙 1 : NAND Flash 存储器运行就绪	1

ECC0/1 状态寄存器

寄存器	地址	R/W	描述	复位值
NFESTAT0	0x4E000024	R/W	NAND Flash ECC 状态寄存器给 I/O [7:0]	0x00000000
NFESTAT1	0x4E000028	R/W	NAND Flash ECC 状态寄存器给 I/O [15:8]	0x00000000

NFESTAT0	位	描述	初始状态
SErrorDataNo	[24:21]	指示备份区域中哪个数据出现错误	00
SErrorBitNo	[21:18]	指示备份区域中哪位出现错误	000
MErrorDataNo	[17:7]	指示主数据区域中哪个数据出现错误	0x00
MErrorBitNo	[6:4]	指示主数据区域中哪位出现错误	000
SpareError	[3:2]	指示是否发生备份区域位失败错误 00 : 无错误 10 : 多错误	00 01 : 1 位错误 (纠错) 11 : ECC 区域错误
MainError	[1:0]	指示是否发生主数据区域位失败错误 00 : 无错误 10 : 多错误	00 01 : 1 位错误 (纠错) 11 : ECC 区域错误

注意：

超出的值只在 ECC 寄存器和 ECC 状态寄存器包含有效的值时才有效。

NFESTAT1	位	描述	初始状态
SErrorDataNo	[24:21]	指示备份区域中哪个数据出现错误	00
SErrorBitNo	[21:18]	指示备份区域中哪位出现错误	000
MErrorDataNo	[17:7]	指示主数据区域中哪个数据出现错误	0x00
MErrorBitNo	[6:4]	指示主数据区域中哪位出现错误	000
SpareError	[3:2]	指示是否发生备份区域位失败错误 00 : 无错误 10 : 多错误	00 01 : 1 位错误 (纠错) 11 : ECC 区域错误
MainError	[1:0]	指示是否发生主数据区域位失败错误 00 : 无错误 10 : 多错误	00 01 : 1 位错误 (纠错) 11 : ECC 区域错误

注意：

超出的值只在 ECC 寄存器和 ECC 状态寄存器包含有效的值时才有效。

主数据区域 ECC0/1 状态寄存器

寄存器	地址	R/W	描述	复位值
NFMECC0	0x4E00002C	R	NAND Flash ECC 寄存器给 data [7:0]	0xXXXXXX
NFMECC1	0x4E000030	R	NAND Flash ECC 寄存器给 data [15:8]	0xXXXXXX

NFMECC0	位	描述	初始状态
MECC0_3	[31:24]	ECC3 给 data[7:0]	0xXX
MECC0_2	[23:16]	ECC2 给 data[7:0]	0xXX
MECC0_1	[15:8]	ECC1 给 data[7:0]	0xXX
MECC0_0	[7:0]	ECC0 给 data[7:0]	0xXX

NFMECC1	位	描述	初始状态
MECC0_3	[31:24]	ECC3 给 data[15:8]	0xXX
MECC0_2	[23:16]	ECC2 给 data[15:8]	0xXX
MECC0_1	[15:8]	ECC1 给 data[15:8]	0xXX
MECC0_0	[7:0]	ECC0 给 data[15:8]	0xXX

注释：

当 MainECClock(NFCONT[5])位为‘0’(开锁)时读或写主区域数据时 NAND Flash 控制器生成 NFMECC0/1。

备份区域 ECC 状态寄存器

寄存器	地址	R/W	描述	复位值
NFSECC	0x4E000034	R	NAND Flash ECC 寄存器给 I/O [15:0]	0xXXXXXX

NFSECC	位	描述	初始状态
SECC1_1	[31:24]	备份区域 ECC1 状态给 I/O [15:8]	0xXX
SECC1_0	[23:16]	备份区域 ECC0 状态给 I/O [15:8]	0xXX
SECC0_1	[15:8]	备份区域 ECC1 状态给 I/O [7:0]	0xXX
SECC0_0	[7:0]	备份区域 ECC0 状态给 I/O [7:0]	0xXX

注释：

当 SpareECClock (NFCONT[6])位为‘0’(开锁)时读或写主区域数据时 NAND Flash 控制器生成 NFMECC。

块地址寄存器

寄存器	地址	R/W	描述	复位值
NFSBLK	0x4E000038	R/W	NAND Flash 可编程开始块地址	0x000000
NFEBLK	0x4E00003C	R/W	NAND Flash 可编程结束块地址 可以编程 NAND Flash 的开始地址和结束地址。当使能了软件锁定或紧锁，开始地址和结束地址又与其相同时，NAND Flash 的全部区域都将被锁定。	0x000000

NFSBLK	位	描述	初始状态
SBLK_ADDR2	[23:16]	擦除块操作的 3 rd 地址	0x00
SBLK_ADDR1	[15:8]	擦除块操作的 2 nd 块地址	0x00
SBLK_ADDR0	[7:0]	擦除块操作的 1 st 块地址 (只有位[7:5]有效)	0x00

注释：

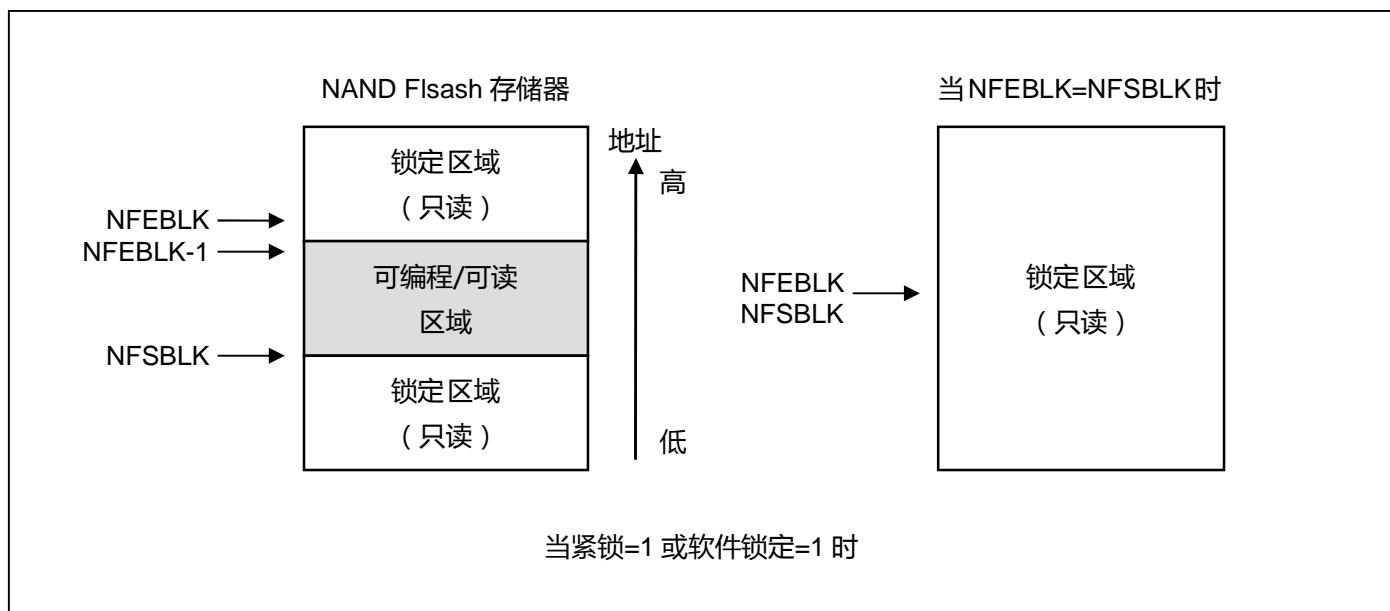
先进闪存的块地址开始于 3 个地址周期。因此块地址寄存器只需要 3 个字节。

NFEBLK	位	描述	初始状态
Eblk_ADDR2	[23:16]	擦除块操作的 3 rd 地址	0x00
Eblk_ADDR1	[15:8]	擦除块操作的 2 nd 块地址	0x00
Eblk_ADDR0	[7:0]	擦除块操作的 1 st 块地址 (只有位[7:5]有效)	0x00

注释：

先进闪存的块地址开始于 3 个地址周期。因此块地址寄存器只需要 3 个字节。

NFSLK 和 NFEBLK 可以在软件锁位(NFCONT[12])为使能时被改变。但如果紧锁位(NFCONT[13])置位时则不能被改变。



7

时钟和电源管理

概述

时钟和电源管理模块由三部分组成：时钟控制，USB 控制和电源控制。

S3C2440A 中的时钟控制逻辑可以产生必须的时钟信号，包括 CPU 的 FCLK，AHB 总线外设的 HCLK 以及 APB 总线外设的 PCLK。S3C2440A 包含两个锁相环（PLL）：一个提供给 FCLK、HCLK 和 PCLK，另一个专用于 USB 模块（48MHz）。时钟控制逻辑可以不使用 PLL 来减慢时钟，并且可以由软件连接或断开各外设模块的时钟，以降低功耗。

关于电源控制逻辑，S3C2440A 包含了各种电源管理方案来保证对给定任务的最佳功耗。S3C2440A 中的电源管理模块可以激活成四种模式：正常（NORMAL）模式、慢速（SLOW）模式、空闲（IDLE）模式和睡眠（SLEEP）模式。

普通（NORMAL）模式：这个模式提供时钟给 CPU，也提供给所有 S3C2440A 的外设。在此模式中，当所有外设都开启时功耗将将达到最大。它允许用户用软件控制外设的运行。例如如果一个定时器不是必须的，用户可以断开连接到定时器的时钟（CLKCON 寄存器），以降低功耗。

慢速（SLOW）模式：无 PLL 模式。不像普通模式，慢速模式使用一个外部时钟（XTIPLL 或 EXTCLK）直接作为 FCLK 给 S3C2440A，而没有使用 PLL。在此模式中，功耗只取决于外部时钟的频率。排除了因 PLL 而产生的功耗。

空闲（IDLE）模式：这个模块只断开了 CPU 内核的时钟（FCLK），但它提供时钟给所有其它外设。空闲模式产生了因 CPU 内核而产生的功耗减少的结果。任何中断请求给 CPU 都可以使其从空闲模式中唤醒。

睡眠（SLEEP）模式：这个模块与内部供电是分离的。因此在此模式中发生了没有因 CPU 和除唤醒逻辑以外的内部逻辑的功耗。要激活睡眠模式需要两个独立的供电电源。两个电源之一提供电源给唤醒逻辑。另一个提供电源给包括 CPU 在内的其它内部逻辑，而且应当能够控制供电的开和关。在睡眠模式中，第二个为 CPU 和内部逻辑供电电源将被关闭。可以由 EINT[15:0]或 RTC 闹铃中断产生从睡眠模式中唤醒。

功能描述

时钟结构

图 7-1 显示了时钟结构的方框图。主时钟源来自一个外部晶振 (XTIPLL) 或外部时钟 (EXTCLK)。时钟发生包含了一个连接到外部晶振的振荡器(震荡放大器),还含有 S3C2440A 所必须的两个用于产生高频率时钟的 PLL(锁相环)。

时钟源选择

表 7-1 显示了模式控制引脚 (OM3 和 OM2) 的组合关系的并为 S3C2440A 选择时钟源。nRESET 的上升沿时参考 OM3 和 OM2 引脚将 OM[3:2]的状态在内部锁定。

表 7-1. 引导启动 (Boot-Up) 时时钟源的选择

模式 OM[3:2]	MPLL 状态	UPLL 状态	主时钟源	USB 时钟源
00	开启	开启	晶振	晶振
01	开启	开启	晶振	外部时钟
10	开启	开启	外部时钟	晶振
11	开启	开启	外部时钟	外部时钟

注意：

1. 虽然 MPLL 在复位后就开始, MPLL 输出(Mpll)并没有作为系统时钟, 直到软件写入有效值来设置 MPLLCON 寄存器。在设置此值之前, 是将外部晶振或外部时钟源提供的时钟直接作为系统时钟。即使用户不想改变 MPLLCON 寄存器的默认值, 用户也应当写入与之相同的值到 MPLLCON 寄存器中。
2. OM[3:2]是用于当 OM[1:0]为 11 时决定一个测试模式。

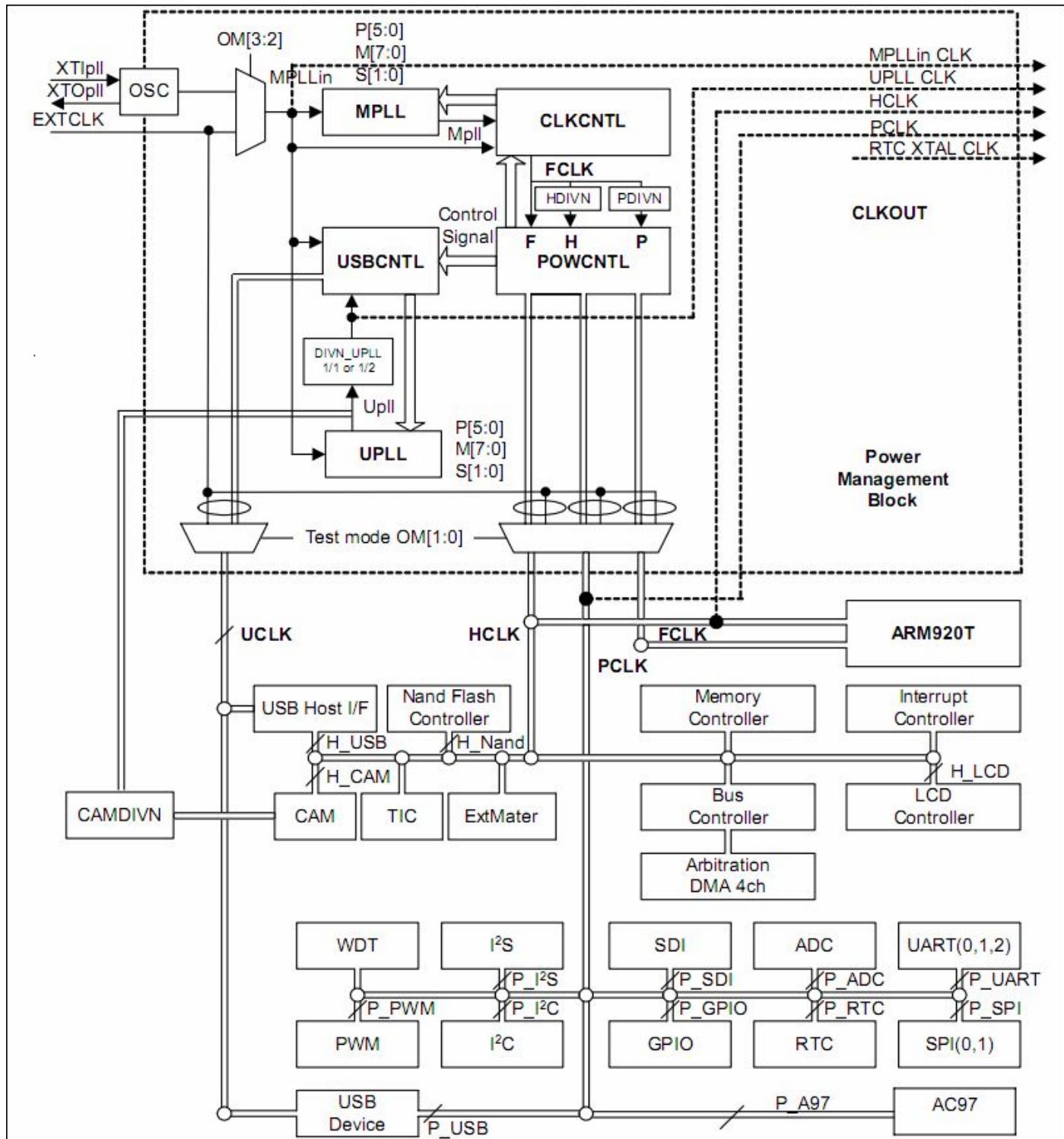


图 7-1. 时钟发生模块框图

锁相环 (PLL)

时钟发生器之中作为一个电路的 MPLL，参考输入信号的频率和相位同步出一个输出信号。在这种应用中，其包含了如图 7-2 所示的以下基本模块：用于生成与输入直流电压成比例的输出频率的压控振荡器 (VCO)、用于将输入频率 (Fin) 按 p 分频的分频器 P、用于将 VCO 输出频率按 m 分频并输入到相位频率检测器 (PFD) 中的分频器 M、用于将 VCO 输出频率按 s 分频成为 Mpll (输出频率来自 MPLL 模块) 的分频器 S、鉴相器、电荷泵以及环路滤波器。输出时钟频率 Mpll 相关参考输入时钟频率 Fin 有如下等式：

$$Mpll = (2 \times m \times Fin) / (p \times 2^s)$$

$$m = M (\text{分频器 } M \text{ 的值}) + 8, p = P (\text{分频器 } P \text{ 的值}) + 2$$

时钟发生器之中的 UPLL 在每方面都与 MPLL 类似。

以下部分描述了 PLL 的运行，包括鉴相器、电荷泵、压控振荡器 (VCO) 和环路滤波器。

鉴相器 (PFD)

PFD 检测 Fref 和 Fvco 之间的相位差，并在检测到相位差时产生一个控制信号（跟踪信号）。Fref 意思为参考频率，如图 7-2 所示。

电荷泵 (PUMP)

电荷泵将 PFD 控制信号转换为一个按比例变化的电压并通过外部滤波器来驱动 VCO。

环路滤波器

PFD 产生用于电荷泵的控制信号，在每次 Fvco 与 Fref 比较时可能产生很大的偏差（纹波）。为了避免 VCO 过载，使用低通滤波器采样并且滤除控制信号的高频分量。滤波器是一个典型由一个电阻和一个电容组成的单极性 RC 滤波器。

压控振荡器 (VCO)

从环路滤波器的输出电压驱动 VCO，引起其振荡频率线性增大或减小，如同均匀变化电压的功能。当 Fvco 与 Fref 频率和相位都在限期内相匹配时，PFD 停止发送控制信号给电荷泵，并转变为稳定输入电压给环路滤波器。VCO 频率保持恒定，PLL 则保持固定为系统时钟。

PLL 和时钟产生器的通常条件

PLL 和时钟发生器通常使用以下条件。

环路滤波器电容	C_{LF}	MPLLCAP: $1.3 \text{ nF} \pm 5\%$
		UPLLCAP: $700 \text{ pF} \pm 5\%$
外部 X-tal 频率	—	12 至 20 MHz (注释)
X-tal 的外部使用电容	C_{EXT}	15 至 22 pF

注释：

1. 该值可变。
2. $FCLK_{OUT}$ 必须大于 200MHz (这并不意味着 ARM 内核必须在高于 200MHz 下运行)。

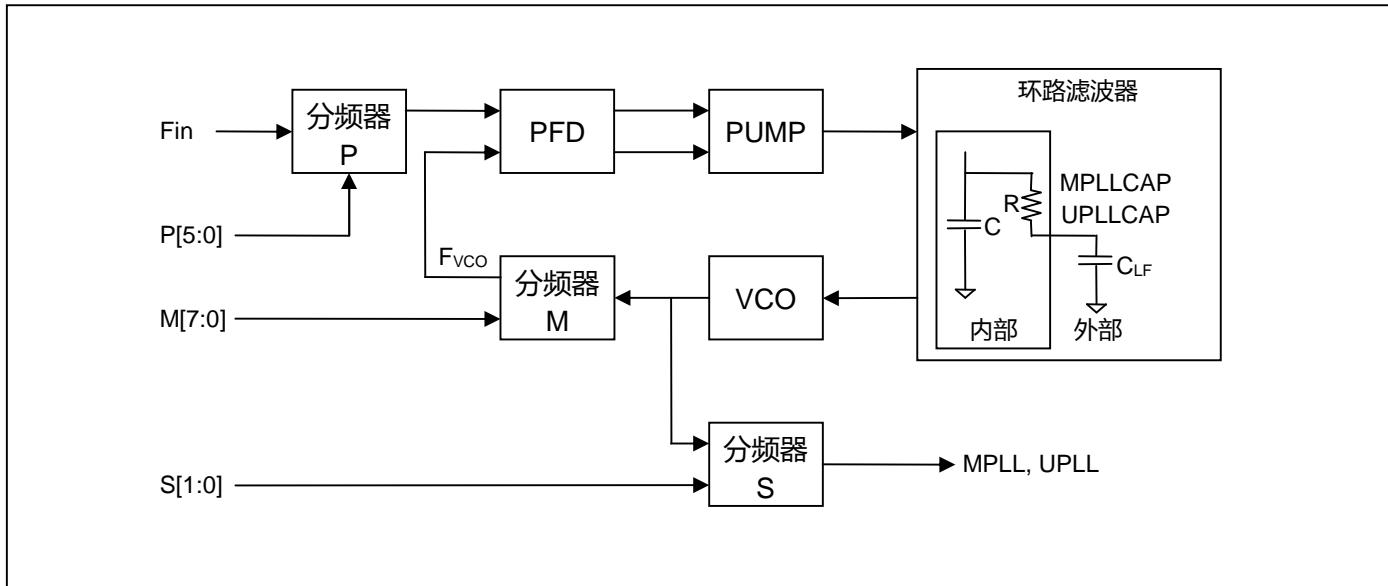


图 7-2. PLL (锁相环) 方框图

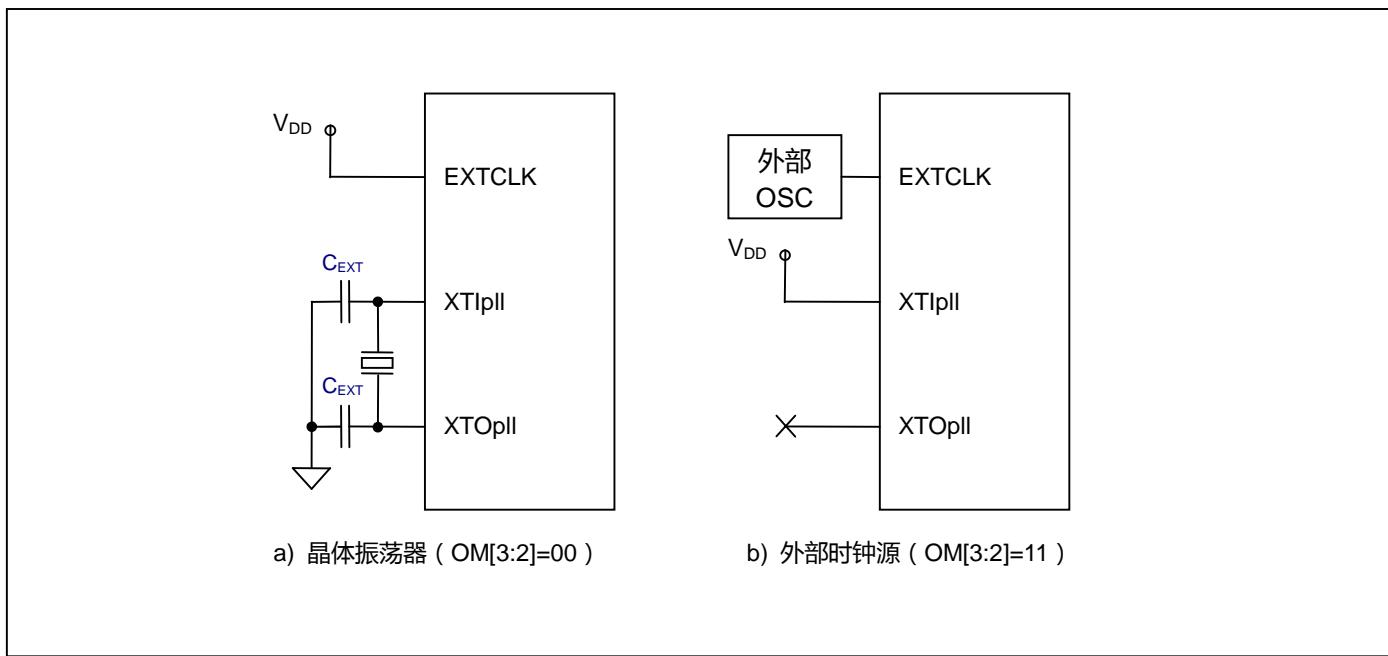


图 7-3. 主振荡电路例子

时钟控制逻辑

时钟控制逻辑决定使用的时钟源，即使用 PLL 时钟 (Mpll) 或直接使用外部时钟 (XTIpll 或 EXTCLK)。当配置了 PLL 为一个新频率值时，时钟控制逻辑先禁止 FCLK，直至使用 PLL 锁定时间使 PLL 稳定输出。时钟控制逻辑在上电复位时和从掉电模式中唤醒时同样是激活的。

上电复位 (XTIpll)

图 7-4 显示了上电复位期间时钟行为顺序。晶振在若干毫秒内开始振荡。当在 OSC (XTIpll) 时钟稳定后释放 nRESET，PLL 开始按默认 PLL 配置运行。但是通常认为上电复位后的 PLL 是不稳定的，因此在软件重新配置 PLLCON 寄存器之前 Fin 代替 Mpll (PLL 输出) 直接提供给 FCLK。即使用户不希望在复位后改变 PLLCON 寄存器的默认值，用户还是应该用软件写入相同的值到 PLLCON 寄存器中。

只有置 PLL 为一个新频率后，PLL 会开始锁定连续逼近新频率。可以在锁定时间后立即配置 FCLK 为 PLL 输出 (Mpll)。

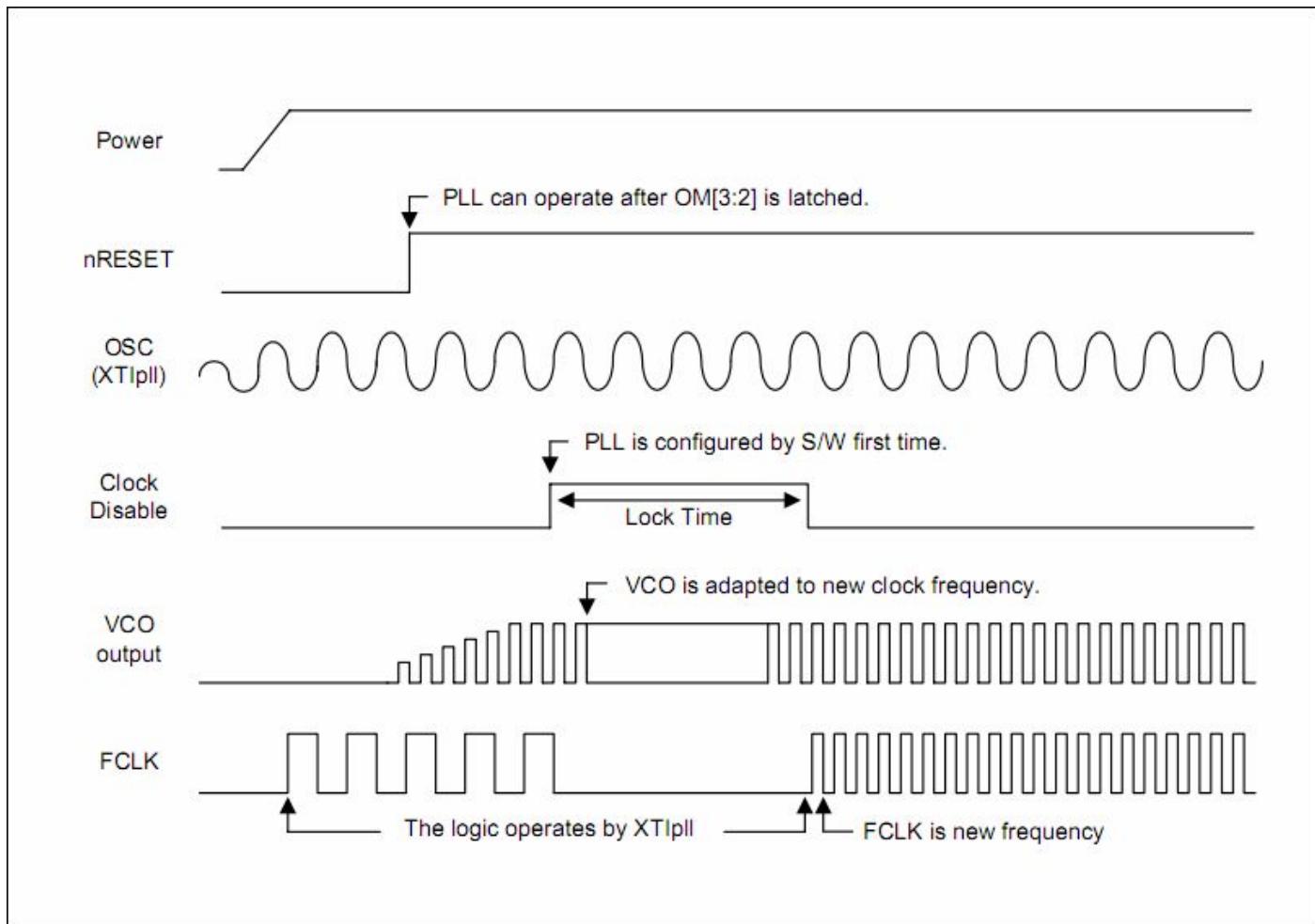


图 7-4. 上电复位顺序 (当外部时钟源为晶振时)

普通模式下改变 PLL 设置

在 S3C2440A 运行在普通模式期间，用户可以通过改写 PMS 的值来改变频率，并且将自动插入 PLL 锁定时间。在锁定时间期间，不提供时钟给 S3C2440A 的内部模块。图 7-5 显示了时序图。

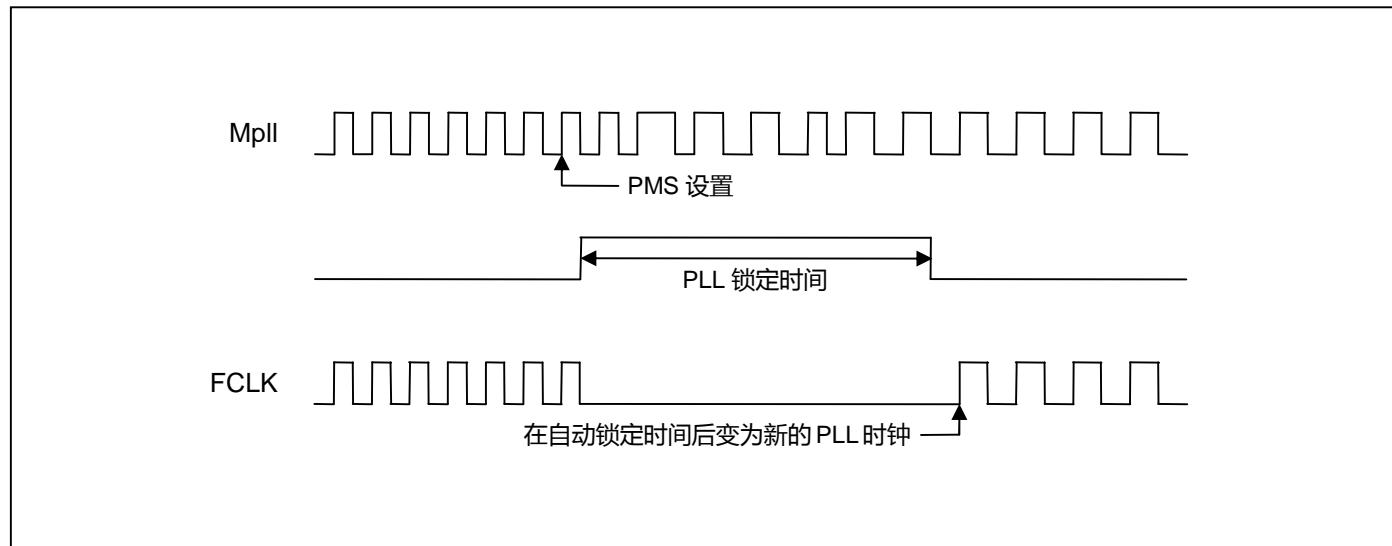


图 7-5. 通过设置 PMS 的值改变慢时钟

USB 时钟控制

USB 主机接口和 USB 设备接口都需要 48MHz 的时钟。S3C2440A 中，USB 专用 PLL (UPLL) 生成 48MHz 给 USB。在配置 PLL (UPLL) 之前不提供 UCLK。

条件	UCLK 状态	UPLL 状态
复位后	XTIpll 或 EXTCLK	开启
配置 UPLL 后	低电平：锁定时间期间 48MHz：PLL 锁定时间后	开启
CLKSLOW 寄存器关闭了 UPLL	XTIpll 或 EXTCLK	关闭
CLKSLOW 寄存器开启了 UPLL	48MHz	开启

FCLK , HCLK 和 PCLK

FCLK 是提供给 ARM920T 的时钟。

HCLK 是提供给用于 ARM920T , 存储器控制器 , 中断控制器 , LCD 控制器 , DMA 和 USB 主机模块的 AHB 总线的时钟。

PCLK 是提供给用于外设如 WDT , IIS , I2C , PWM 定时器 , MMC/SD 接口 , ADC , UART , GPIO , RTC 和 SPI 的 APB 总线的时钟。

S3C2440A 还支持对 FCLK、HCLK 和 PCLK 之间分频比例的选择。该比例由 CLKDIVN 控制寄存器中的 HDIVN 和 PDIVN 所决定。

HDIVN	PDIVN	HCLK3_HALF/ HCLK4_HALF	FCLK	HCLK	PCLK	分频比例
0	0	-	FCLK	FCLK	FCLK	1:1:1 (默认)
0	1	-	FCLK	FCLK	FCLK / 2	1:1:2
1	0	-	FCLK	FCLK / 2	FCLK / 2	1:2:2
1	1	-	FCLK	FCLK / 2	FCLK / 4	1:2:4
3	0	0/0	FCLK	FCLK / 3	FCLK / 3	1:3:3
3	1	0/0	FCLK	FCLK / 3	FCLK / 6	1:3:6
3	0	1/0	FCLK	FCLK / 6	FCLK / 6	1:6:6
3	1	1/0	FCLK	FCLK / 6	FCLK / 12	1:6:12
2	0	0/0	FCLK	FCLK / 4	FCLK / 4	1:4:4
2	1	0/0	FCLK	FCLK / 4	FCLK / 8	1:4:8
2	0	0/1	FCLK	FCLK / 8	FCLK / 8	1:8:8
2	1	0/1	FCLK	FCLK / 8	FCLK / 16	1:8:16

设置了 PMS 值后 , 必须接着设置 CLKDIVN 寄存器。设置 CLKDIVN 的值将在 PLL 锁定时间后起效。对于复位和改变电源管理模式该值同样起效。

设置值在 1.5 HCLK 后同样起效。只需要 1 HCLK 就能确认从默认 (1:1:1) 的分频比例到其它分频比例 (1:1:2 , 1:2:2 , 1:2:4) CLKDIVN 寄存器的值的改变。

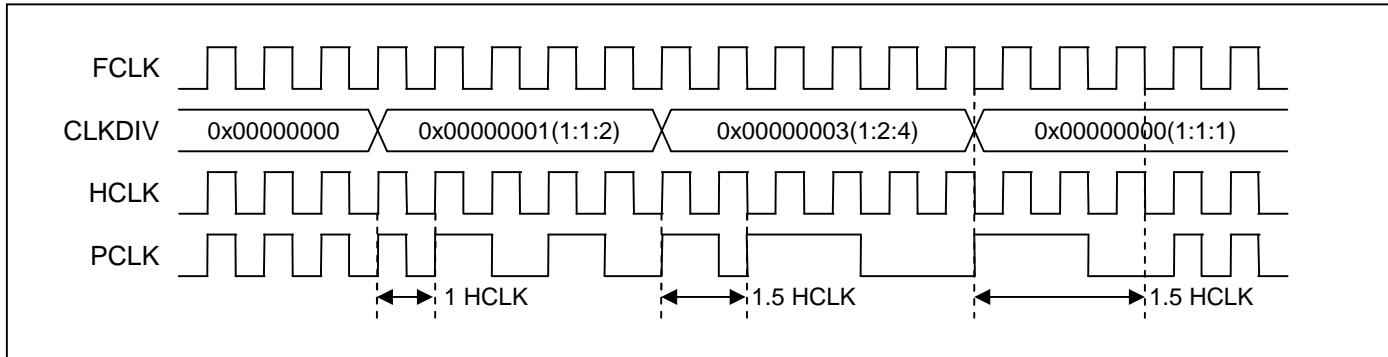


图 7-6. 内部时钟变化例子

注意 :

1. 应当谨慎设置 CLKDIVN , 不要使其超过 HCLK 和 PCLK 的最小值。
2. 如果 HDIVN 不为 0 , CPU 总线模式应该使用以下指令使其从快总线模式改变为异步总线模式 (S3C2440 不支持同步总线模式)。

MMU_SetAsyncBusMode

```
MRC    p15, 0, r0, c1, c0, 0
ORR    r0, r0, #R1_nF:OR:R1_iA
MCR    p15, 0, r0, c1, c0, 0
```

如果 HDIVN 不为 0 并且 CPU 总线模式为快总线模式 , CPU 运行在 HCLK。可以用此特性在不影响 HCLK 和 PCLK 的情况下改变 CPU 频率为一半或更多。

电源管理

电源管理模块使用软件来控制系统时钟，以降低 S3C2440A 中的功耗。这些方案与 PLL，时钟控制逻辑(FCLK，HCLK 和 PCLK) 和唤醒信号有关。图 7-7 显示了 S3C2440A 的时钟分配。

S3C2440A 有四种电源模式。以下部分描述了每个电源管理模式。不容许任意转变模式。模式间允许转变的情况请看图 7-8。

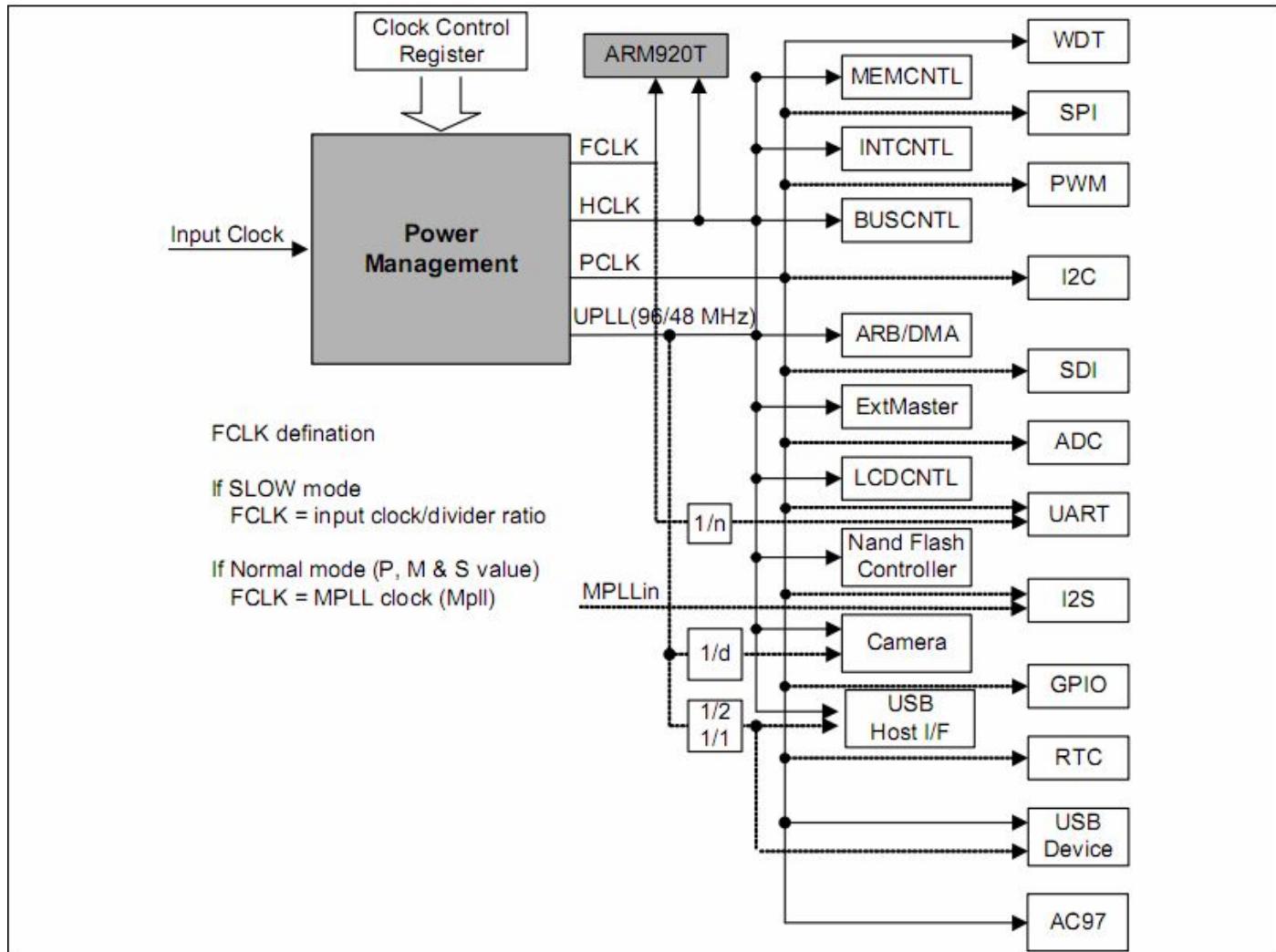


图 7-7. 时钟分配方框图

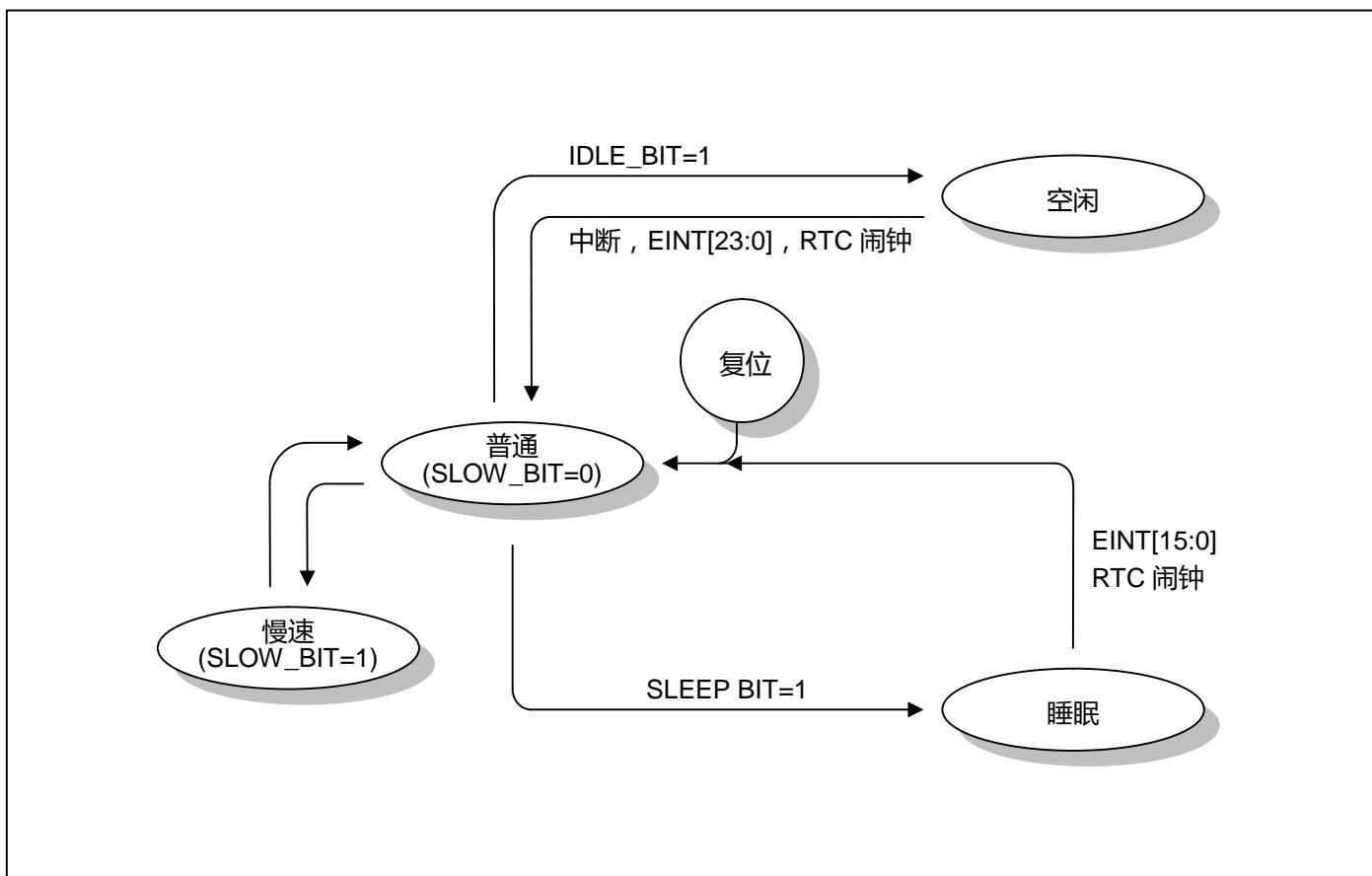


图 7-8. 电源管理状态图

表 7-2. 每种电源模式的时钟和电源状态

模式	ARM920T	AHB 模块 ⁽¹⁾ /WDT	电源管理	GPIO	32.768kHz RTC 时钟	APB 模块 ⁽²⁾ &USBH/LCD/NAND
普通	O	O	O	SEL	O	SEL
空闲	X	O	O	SEL	O	SEL
慢速	O	O	O	SEL	O	SEL
睡眠	OFF	OFF	等待唤醒事件	先前状态	O	OFF

注释：

1. 不包括 USB 主机, LCD 和 NAND
2. 不包括 WDT, 包括 CPU 访问 RTC 接口
3. SEL: 可选 (O, X), O: 使能; X: 禁止; OFF: 关闭电源

普通模式

普通模式中，包括电源管理模块、CPU 核心、总线控制器、存储器控制器、中断控制器、DMA 和外部主控在内的所有外设和基本模块完全可以运行。然而除基本模块外，提供给每个外设的时钟都可以由软件有选择的停止以降低功耗。

空闲模式

空闲模式中，停止了除总线控制器、存储器控制器、中断控制器、电源管理模块外的提供给 CPU 核心的时钟。要退出空闲模式，应当激活 EINT[23:0] 或 RTC 闹钟中断或其它中断（开启 GPIO 模块前 EINT 不可用）

进入空闲模式

如果置位 CLKCON[2] 为 1 来进入空闲模式，S3C2440A 将在一些延时后（直到电源控制逻辑收到 CPU 打包的 ACK 信号）进入空闲模式。

慢速模式

慢速模式中，可以应用慢时钟和排除来自 PLL 的功耗来降低功耗。CLKSLOW 控制寄存器中的 SLOW_VAL 和 CLKDIVN 控制寄存器决定了分频比例。

表 7-3. 慢时钟的 CLKSLOW 和 CLKDIVN 设置例子

SLOW_VAL	FCLK	HCLK		PCLK		UCLK
		1/1 选项 (HDIVN=0)	1/2 选项 (HDIVN=1)	1/1 选项 (HDIVN=0)	1/2 选项 (HDIVN=1)	
0 0 0	EXTCLK 或 XTlpll / 1	EXTCLK 或 XTlpll / 1	EXTCLK 或 XTlpll / 2	HCLK	HCLK / 2	48MHz
0 0 1	EXTCLK 或 XTlpll / 2	EXTCLK 或 XTlpll / 2	EXTCLK 或 XTlpll / 4	HCLK	HCLK / 2	48MHz
0 1 0	EXTCLK 或 XTlpll / 4	EXTCLK 或 XTlpll / 4	EXTCLK 或 XTlpll / 8	HCLK	HCLK / 2	48MHz
0 1 1	EXTCLK 或 XTlpll / 6	EXTCLK 或 XTlpll / 6	EXTCLK 或 XTlpll / 12	HCLK	HCLK / 2	48MHz
1 0 0	EXTCLK 或 XTlpll / 8	EXTCLK 或 XTlpll / 8	EXTCLK 或 XTlpll / 16	HCLK	HCLK / 2	48MHz
1 0 1	EXTCLK 或 XTlpll / 10	EXTCLK 或 XTlpll / 10	EXTCLK 或 XTlpll / 20	HCLK	HCLK / 2	48MHz
1 1 0	EXTCLK 或 XTlpll / 12	EXTCLK 或 XTlpll / 12	EXTCLK 或 XTlpll / 24	HCLK	HCLK / 2	48MHz
1 1 1	EXTCLK 或 XTlpll / 14	EXTCLK 或 XTlpll / 14	EXTCLK 或 XTlpll / 28	HCLK	HCLK / 2	48MHz

慢速模式中，将关闭 PLL 以降低 PLL 的带来的功耗。当在慢速模式中关闭 PLL 并且用户从慢速模式切换到普通模式中时，PLL 则需要时钟的稳定化时间（PLL 锁定时间）。这个 PLL 稳定化时间由带锁定时间计数寄存器的内部逻辑自动插入。PLL 开启后 PLL 稳定将耗时 300μs。PLL 锁定时间期间 FCLK 成为慢时钟

PLL 开启/关闭

PLL 只有在慢速模式中为降低功耗而被关闭。如果在任何其它模式关闭 PLL，MCU 的运行将不会得到保证。

当处理器处于慢速模式并试图改变其状态到 PLL 开启的其它状态，接着应该在 PLL 稳定后清除 SLOW_BIT 来移动到其它状态。

用户可以在 PLL 开启状态下使能 CLKSLOW 寄存器中的慢速模式位来改变频率。慢时钟是在慢速模式期间产生的。图 7-9 显示了时序图。

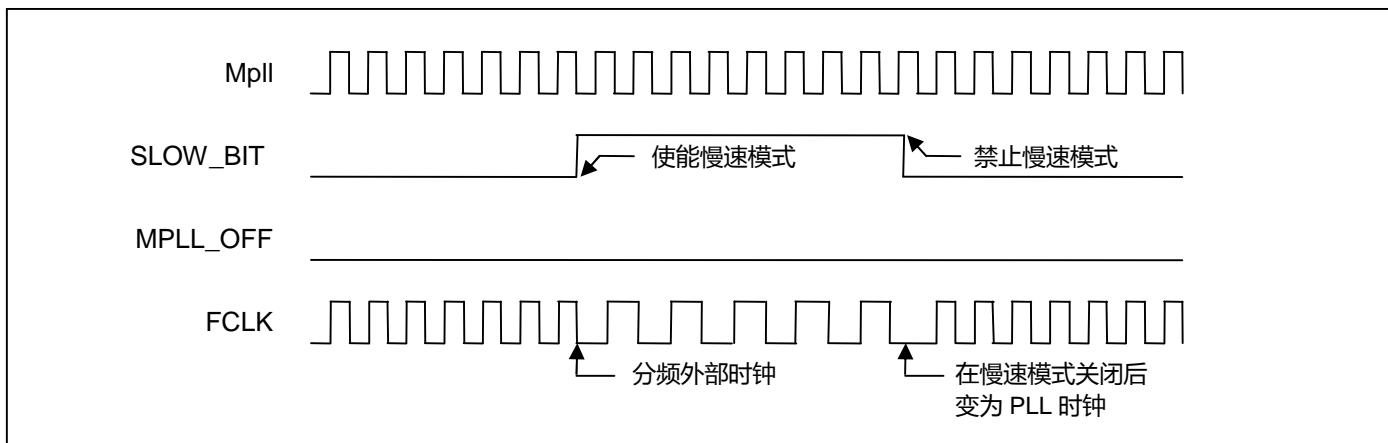


图 7-9. 在 PLL 开启状态下发出退出慢速模式命令

如果用户在 PLL 锁定时间后通过禁止 CLKSLOW 寄存器中的 SLOW_BIT 来实现从慢速模式切换到普通模式，只在禁止慢速模式后才会改变频率。图 7-10 显示了时序图。

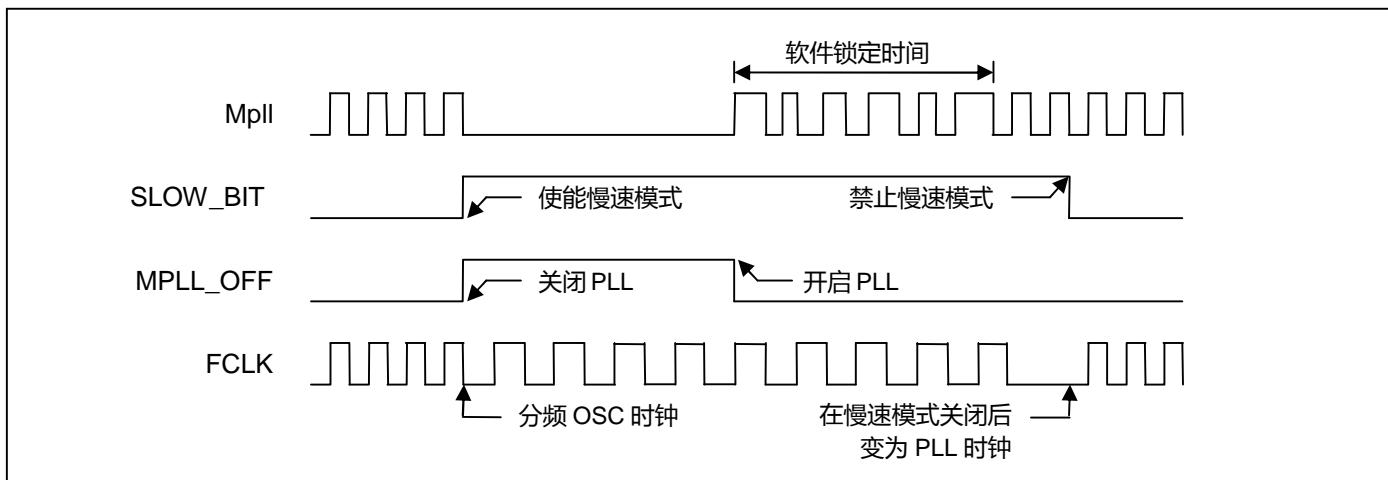


图 7-10. 在锁定时间后发出退出慢速模式命令

如果用户在 PLL 锁定时间后通过同时禁止 CLKSLOW 寄存器中的 SLOW_BIT 和 MPLL_OFF 位来实现从慢速模式切换到普通模式，只在 PLL 锁定时间后才会改变频率。图 7-11 显示了时序图。

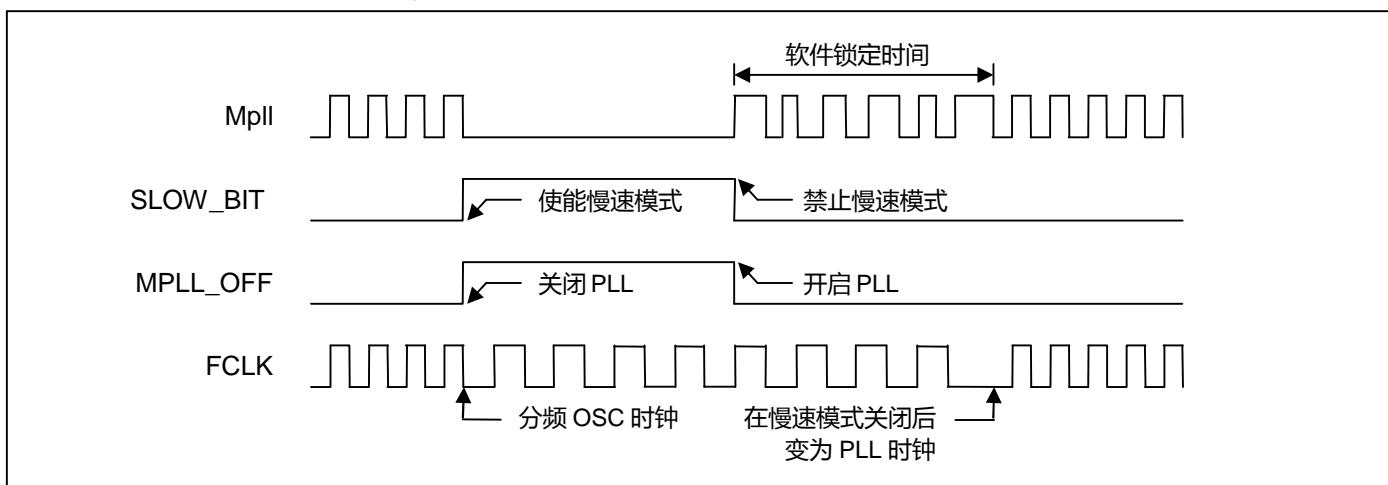


图 7-11. 发出退出慢速模式命令同时立即开启 PLL 命令

睡眠模式

此模块与内部电源是分离的。因此这个模式没有因 CPU 和除唤醒逻辑以外的内部逻辑而产生的功耗。激活睡眠模式需要两个独立的供电源。两个电源之一提供电源给唤醒逻辑。另一个提供电源给包括 CPU 在内的其它内部逻辑，而且应当能够控制供电的开和关。在睡眠模式中，第二个为 CPU 和内部逻辑供电电源将被关闭。可以由 EINT[15:0]或 RTC 闹铃中断产生从睡眠模式中唤醒。

以下为进入睡眠模式的步骤

1. 为睡眠模式合理设置 GPIO 配置。
2. 屏蔽 INTMSK 寄存器中所有中断。
3. 合理配置包括 RTC 闹钟在内的唤醒源。(不需要屏蔽唤醒源在 EINTMASK 中的对应位，目的是使得 SRCPND 或 EINTPEND 的对应位能置位。然而引发了唤醒源并且屏蔽了 EINTMASK 的对应位，唤醒也将发生但 SRCPND 或 EINTPEND 的对应位将不会被置位。)
4. 设置 USB 端口为挂起模式 (MISCCR[13:12]=11b)。
5. 保存一些有特殊含义的值到 GSTATUS[4:3]寄存器。这些寄存器在睡眠模式期间是被保护的。
6. 为数据总线 D[31:0]的上拉电阻配置 MISCCR[1:0]。如果有的外部总线保持器，例如 74LVCH162245，关闭上拉电阻。如果没有开启上拉电阻。另外存储器相关引脚设置为两种类型，另一个是非活动状态。
7. 清除 LCDCON1.ENVID 位来停止 LCD。
8. 读取 rREFRESH 和 rCLKCON 寄存器以填充 TLB。
9. 设置 REFRESH[22]=1b 使得 SDRAM 进入自刷新模式。
10. 等待直到 SDRAM 自刷新有效。
11. 设置 MISCCR[19:17]=111b 使得 SDRAM 信号 (SCLK0 , SCLK1 和 SCKE) 在睡眠模式期间受到保护。
12. 设置 CLKCON 寄存器中的睡眠模式位。

警告：

当系统运行在 NAND 引导启动模式时，硬件引脚配置 EINT[23:21]必须为从睡眠模式中唤醒后的启动设置为输入。

以下为从睡眠模式中唤醒的步骤

1. 如果引发了唤醒源之一将发出内部复位信号。它将与触发了外部 nRESET 引脚的情况相同。此复位持续时间由内部 16 位控制逻辑和由 $t_{RST} = (65535 / XTAL\text{-频率})$ 计算得到的复位触发时间而决定。
2. 检查 GSTATUS2[2]以了解是否是上电使得从睡眠模式中唤醒。
3. 设置 MISCCR[19:17]=000b 释放 SDRAM 信号保护。
4. 配置 SDRAM 存储器控制器。
5. 等待直到 SDRAM 自刷新被释放。通常 SDRAM 需要刷新所有 SDRAM 行的周期。
6. GSTATUS[3:4]中的信息可以用于用户自己的目的，因为在睡眠模式期间 GSTATUS[3:4]中的值是被保护的。
7. -对于 EINT[3:0]，检查 SRCPND 寄存器。
-对于 EINT[15:4]，检查 EINTPEND 而不是 SRCPND (SRCPND 将不会被置位尽管 EINTPEND 的某些位会被置位)。

表 7-4. 睡眠模式中引脚配制

引脚状态		引脚配置的标识
GPIO 引脚	配置为输入	上拉使能
	配置为输出	上拉禁止，输出为低
不含内部上拉控制的输入引脚	如果外部器件没有总是驱动引脚的电平	由外部上拉电阻上拉使能
连接到外部器件输出引脚	如果外部器件的电源为关闭	输出为低
	如果外部器件的电源为开启	高或低 (由外部器件的状态决定)
数据总线	如果存储器电源为关闭	输出为低
	并且存在外部缓冲器	如果缓冲器保持总线电平，上拉禁止
		并且无外部缓冲器

注意：

1. ADC 应该设置为备用模式。
2. USB 端口应该设置为挂起模式。

*此表只作为信息使用。用户应当考虑其自己的板级情况与应用。

VDDi 和 VDDIarm 的电源控制

睡眠模式中将由 PWREN 引脚控制 VDDi , VDDIarm , VDDMPLL 和 VDDUPLL 都关闭。

如果激活（高点平）了 PWREN 信号，VDDi 和 VDDIarm 将由一个外部电压调节器提供。如果 PWREN 引脚是非激活的（低电平），则 VDDi 和 VDDIarm 是关闭的。

注释：

即使可能关闭了 VDDi, VDDIarm, VDDMPLL 和 VDDUPLL, 也应当提供给其它电源引脚。

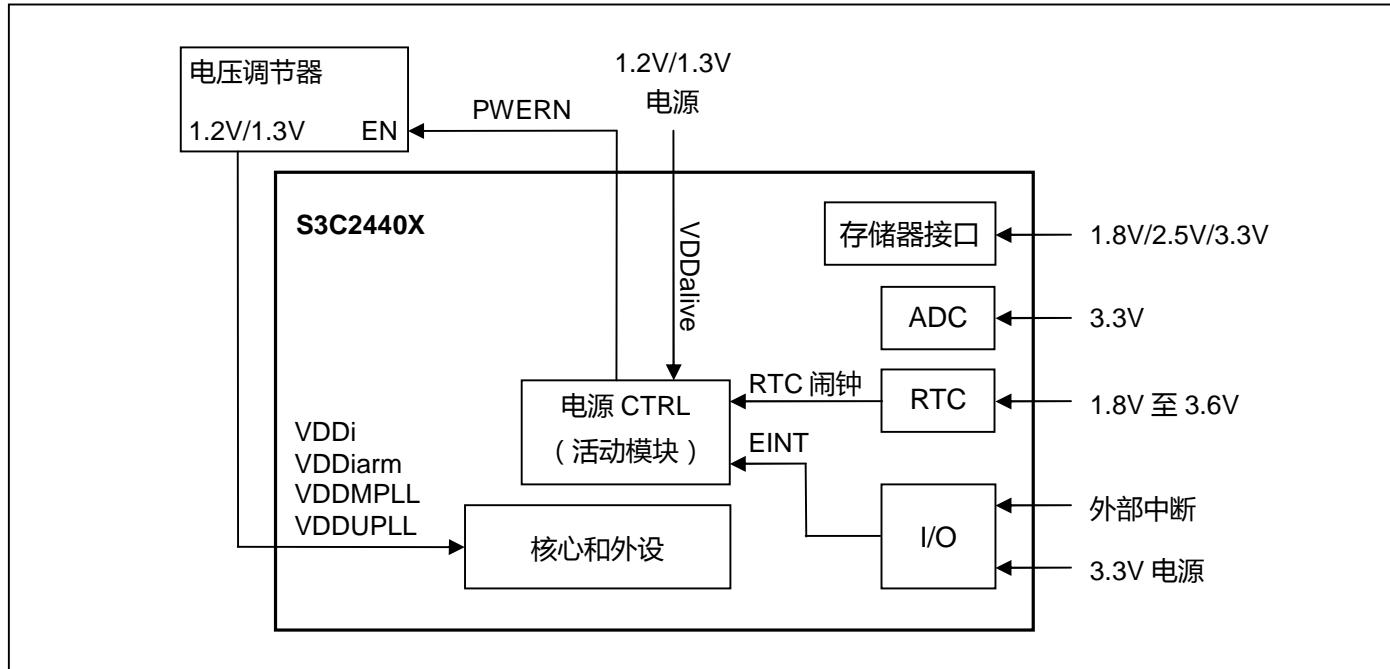


图 7-12. 睡眠模式

注意：

如果你在睡眠模式期间不使用触屏面板，则触摸端口（XP , XM , YP 和 YM）必须悬空。这是由于触摸端口（XP , XM , YP 和 YM）没有连接到 GND 电源端。又因为 XP , YP 在睡眠模式期间将保持为高电平。

为唤醒发出信号 EINT[15:0]

只有遇到以下条件才会唤醒睡眠模式中的 S3C2440A。

- 在 EINTn 输入引脚上发出电平信号（高电平或低电平）或边沿信号（上升沿或下降沿）。
- EINTn 引脚必须在 GPIO 控制寄存器中配置为 EINT。
- nBATT_FLT 引脚必须为高电平。这对于在 GPIO 控制寄存器中配置 EINTn 为考虑上文 a) 条件的外部中断引脚是非常重要。

只有在唤醒后，对应 EINTn 引脚将不会用于唤醒。这意味着该引脚可以再次作为外部输入中断请求引脚使用。

睡眠模式和数据总线的上拉电阻

睡眠模式中可以选定数据总线 (D[31:0]或 D[15:0]) 为高阻态或输出低电平的状态。

数据总线可以设置为开启上拉电阻的高阻态或为睡眠模式降低功耗而设置为关闭上拉电阻的输出低电平。

D[31:0]引脚的上拉电阻由 GPIO 控制寄存器 (MISCCR) 控制。然而如果还有一个外部总线保持器，如 74LVCH162245 在数据总线上，用户可以选择二者之一的状态：一个是关闭上拉电阻的输出低电平，另一个是关闭上拉电阻的高阻态，都为低功耗。

睡眠模式和输出端口状态

输出端口应当在关闭电源模式中设置为一个合理的逻辑电平使得电流消耗最小。如果输出端口引脚没有接负载则首选高电平。如果输出为低电平，内部寄生电阻将消耗电流；如果输出为高电平，将不会消耗电流。作为输出端口，如果输出状态为高电平将减少电流的消耗。

睡眠模式中推荐将输出端口设置为输出高电平。

电池故障信号 (nBATT_FLT)

nBATT_FLT 引脚有如下两个功能。

- 当 CPU 不处于睡眠模式时 nBATT_FLT 引脚将通过设置 BATT_FUNC(MISCCR[22:20]) 为 $10x'b$ 来引起中断请求。nBATT_FLT 的中断类型为为低电平时触发。
- 当 CPU 处于睡眠模式时 nBATT_FLT 的发出将通过设置 BATT_FUNC(MISCCR[22:20]) 为 $11x'b$ 来禁止从睡眠模式的唤醒。因此任何唤醒源都将被屏蔽。

如果发出了 nBATT_FLT，是为了保护电池电量过低的系统故障。

ADC 掉电

ADC 在 ADCCON 寄存器中包含了一个额外的掉电位。如果 S3C2440A 进入睡眠模式，则 ADC 应当进入其自己的掉电模式。

时钟发生器和电源管理特殊寄存器

锁定时间计数寄存器 (LOCKTIME)

寄存器	地址	R/W	描述	复位值
LOCKTIME	0x4C000000	R/W	PLL 锁定时间计数寄存器	0xFFFFFFFF

BWSCON	位	描述	初始状态
U_LT	[31:16]	UCLK 的 UPLL 锁定时间计数值 (U_LT 300μs)	0xFFFF
M_LT	[15:0]	FCLK , HCLK 和 PCLK 的 UPLL 锁定时间计数值 (M_LT 300μs)	0xFFFF

PLL 控制寄存器 (MPLLCON 和 UPLLCON)

寄存器	地址	R/W	描述	复位值
MPLLCON	0x4C000004	R/W	MPLL 配置寄存器	0x00096030
UPLLCON	0x4C000008	R/W	UPLL 配置寄存器	0x0004d030

PLLCON	位	描述	初始状态
MDIV	[19:12]	主分频器控制	0x96 / 0x4d
PDIV	[9:4]	预分频器控制	0x03 / 0x03
SDIV	[1:0]	后分频器控制	0x0 / 0x0

注意：

当你设置 MPLL 和 UPLL 的值时，你必须首先设置 UPLL 值再设置 MPLL 值。(大约需要 7 个 NOP 的间隔)

MPLL 控制寄存器

$$MPLL = (2 \times m \times Fin) / (p \times 2^s)$$

$$m = (MDIV + 8), p = (PDIV + 2), s = SDIV$$

UPLL 控制寄存器

$$UPLL = (m \times Fin) / (p \times 2^s)$$

$$m = (MDIV + 8), p = (PDIV + 2), s = SDIV$$

PLL 值选择向导 (MPLLCON)

1. $F_{out} = 2 \times m \times F_{in} / (p * 2^s)$, $F_{vco} = 2 \times m \times F_{in} / p$ 此处 : $m = MDIV + 8$, $p = PDIV + 2$, $s = SDIV$
2. $600MHz \leq F_{VCO} \leq 1.2GHz$
3. $200MHz \leq F_{CLKOUT} \leq 600MHz$
4. 不要设置 P 或 M 的值为 0, 这是因为设置 P=000000, M=00000000 将会引起 PLL 的故障。
5. P 和 M 的合理范围为 : $1 \leq P \leq 62$, $1 \leq M \leq 248$

注意 :

尽管有选择 PLL 的值的规则, 我们只推荐的 PLL 值推荐表中的值。如果你必须使用其它值, 请联系我们。

PLL 值选择表

输入频率	输出频率	MDIV	PDIV	SDIV
12.0000MHz	48.00 MHz (注释)	56(0x38)	2	2
12.0000MHz	96.00 MHz (注释)	56(0x38)	2	1
12.0000MHz	271.50 MHz	173(0xAD)	2	2
12.0000MHz	304.00 MHz	68(0x44)	1	1
12.0000MHz	405.00 MHz	127(0x7f)	2	1
12.0000MHz	532.00 MHz	125(0x7d)	1	1
16.9344MHz	47.98 MHz (注释)	60(0x3C)	4	2
16.9344MHz	95.96 MHz (注释)	60(0x3C)	4	1
16.9344MHz	266.72 MHz	118(0x76)	2	2
16.9344MHz	296.35 MHz	97(0x61)	1	2
16.9344MHz	399.65 MHz	110(0x6E)	3	1
16.9344MHz	530.61 MHz	86(0x56)	1	1
16.9344MHz	533.43 MHz	118(0x76)	1	1

注释 :

48.00MHz 和 96.00MHz 输出是用于 UPLLCON 寄存器。

时钟控制寄存器 (CLKCON)

寄存器	地址	R/W	描述	复位值
CLKCON	0x4C00000C	R/W	时钟生成控制寄存器	0xFFFFF0

CLKCON	位	描述	初始状态
AC97	[20]	控制进入 AC'97 模块的 PCLK 0 = 禁止 1 = 使能	1
Camera	[19]	控制进入摄像头模块的 HCLK 0 = 禁止 1 = 使能	1
SPI	[18]	控制进入 SPI 模块的 PCLK 0 = 禁止 1 = 使能	1
IIS	[17]	控制进入 IIS 模块的 PCLK 0 = 禁止 1 = 使能	1
IIC	[16]	控制进入 IIC 模块的 PCLK 0 = 禁止 1 = 使能	1
ADC (&Touch Screen)	[15]	控制进入 ADC 模块的 PCLK 0 = 禁止 1 = 使能	1
RTC	[14]	控制进入 RTC 模块的 PCLK。即使此位清除为 0 , RTC 定时器也活动 0 = 禁止 1 = 使能	1
GPIO	[13]	控制进入 GPIO 模块的 PCLK 0 = 禁止 1 = 使能	1
UART2	[12]	控制进入 UART2 模块的 PCLK 0 = 禁止 1 = 使能	1
UART1	[11]	控制进入 UART1 模块的 PCLK 0 = 禁止 1 = 使能	1
UART0	[10]	控制进入 UART0 模块的 PCLK 0 = 禁止 1 = 使能	1
SDI	[9]	控制进入 SDI 模块的 PCLK 0 = 禁止 1 = 使能	1
PWMTIMER	[8]	控制进入 PWM 定时器模块的 PCLK 0 = 禁止 1 = 使能	1
USB device	[7]	控制进入 USB 设备模块的 PCLK 0 = 禁止 1 = 使能	1
USB host	[6]	控制进入 USB 主机模块的 HCLK 0 = 禁止 1 = 使能	1
LCDC	[5]	控制进入 LCDC 模块的 HCLK 0 = 禁止 1 = 使能	1
NAND Flash Controller	[4]	控制进入 NAND Flash 控制器模块的 HCLK 0 = 禁止 1 = 使能	1
SLEEP	[3]	控制 S3C2440A 的睡眠模式 0 = 禁止 1 = 转换到空闲模式	0
IDLE BIT	[2]	进入空闲模式。此位不会自动清零 0 = 禁止 1 = 转换到睡眠模式	0
保留	[1:0]	保留	0

时钟慢速控制 (CLKSLOW) 寄存器

寄存器	地址	R/W	描述	复位值
CLKSLOW	0x4C000010	R/W	慢时钟控制寄存器	0x00000004

CLKSLOW	位	描述	初始状态
UCLK_ON	[7]	0 : 开启 UCLK (同时开启 UPLL 并自动插入 UPLL 锁定时间) 1 : 关闭 UCLK (同时关闭 UPLL)	0
保留	[6]	保留	-
MPLL_OFF	[5]	0 : 开启 PLL 在 PLL 稳定化时间 (至少 300μs) 后 , 可以清除 SLOW_BIT 为 0 1 : 关闭 PLL 只有当 SLOW_BIT 为 1 时才关闭 PLL	0
SLOW_BIT	[4]	0 : FCLK = MpII (MPLL 输出) 1 : 慢速模式 FCLK = 输入时钟 / (2 × SLOW_VAL) , 当 SLOW_VAL>0 FCLK = 输入时钟 , 当 SLOW_VAL=0 输入时钟 = XTlpll 或 EXTCLK	0
保留	[3]	保留	-
SLOW_VAL	[2:0]	当 SLOW_BIT 开启时慢时钟的分频器	0x4

时钟分频控制 (CLKDIVN) 寄存器

寄存器	地址	R/W	描述	复位值
CLKDIVN	0x4C000014	R/W	时钟分频控制寄存器	0x00000004

CLKDIVN	位	描述	初始状态
DIVN_UPLL	[3]	UCLK 选择寄存器 (UCLK 必须为 48MHz 给 USB) 0 : UCLK = UPLL 时钟 1 : UCLK = UPLL 时钟 / 2 当 UPLL 时钟被设置为 48MHz 时 , 设置为 0 当 UPLL 时钟被设置为 96MHz 时 , 设置为 1	0
HDIVN	[2:1]	00 : HCLK = FCLK/1 01 : HCLK = FCLK/2 10 : HCLK = FCLK/4 当 CAMDIVN[9] = 0 时 HCLK = FCLK/8 当 CAMDIVN[9] = 1 时 11 : HCLK = FCLK/3 当 CAMDIVN[8] = 0 时 HCLK = FCLK/6 当 CAMDIVN[8] = 1 时	00
PDIVN	[0]	0 : PCLK 是和 HCLK/1 相同的时钟 1 : PCLK 是和 HCLK/2 相同的时钟	0

摄像头时钟分频 (CAMDIVN) 寄存器

寄存器	地址	R/W	描述	复位值
CAMDIVN	0x4C000018	R/W	摄像头时钟分频寄存器	0x00000000

CAMDIVN	位	描述	初始状态
DVS_EN	[12]	0 : 关闭 DVS ARM 内核将正常运行在 FCLK (MPLL 输出) 1 : 开启 DVS ARM 内核将运行在与系统时钟的时钟 (HCLK)	0
保留	[11]	保留	0
保留	[10]	保留	0
HCLK4_HALF	[9]	当 CLKDIVN[2:1]=10b 时 HDIVN 分频率改变位 0 : HCLK = FCLK/4 1 : HCLK= FCLK/8 参考 CLKDIV 寄存器	0
HCLK3_HALF	[8]	当 CLKDIVN[2:1]=11b 时 HDIVN 分频率改变位 0 : HCLK = FCLK/3 1 : HCLK= FCLK/6 参考 CLKDIV 寄存器	0
CAMCLK_SEL	[4]	0 : 使用 UPLL 输出作为 CAMCLK (CAMCLK=UPLL 输出) 1 : CAMCLK_DIV 的值分频得到 CAMCLK	0
CAMCLK_DIV	[3:0]	CAMCLK 分频因子设置寄存器 (0 至 15) 摄像头时钟 = UPLL / [(CAMCLK_DIV +1)x2] 此位在 CAMCLK_SEL=1 时有效	0

8

直接存储器存取 DMA

概述

S3C2440A 支持 4 通道处于系统总线和外设总线间的 DMA 控制器。DMA 控制器的每个通道都可以无限制的执行系统总线与/或外设总线之间设备的数据移动。换句话说，每个通道都可以处理以下 4 种情况：

- 1) 源和目标都在系统总线上
- 2) 当目标在外设总线上时源在系统总线上
- 3) 当目标在系统总线上时源在外设总线上
- 4) 源和目标都在外设总线上

DMA 的主要优点是在无 CPU 干预的情况下能进行数据传输。DMA 的运行可以由软件开始，也可以来自内部外设或外部请求引脚的请求。

DMA 请求源

如果由 DCON 寄存器选择了硬件 DMA 请求模式，则 DMA 控制器的每个通道都可以在 4 个 DMA 源中选择 DMA 请求源的其中之一。（注意如果选择了软件请求模式，此 DMA 请求源没有一点意义）表 8-1 显示了每个通道的 4 个 DMA 源。

	源 0	源 1	源 2	源 3	源 4	源 5	源 6
通道 0	nXDREQ0	UART0	SDI	定时器	USB 设备 EP1	I2SSDO	PCMIN
通道 1	nXDREQ1	UART1	I2SSDI	SPI0	USB 设备 EP2	PCMOUT	SDI
通道 2	I2SSDO	I2SSDI	SDI	定时器	USB 设备 EP3	PCMIN	MICIN
通道 3	UART2	SDI	SPI1	定时器	USB 设备 EP4	MICIN	PCMOUT

此处的 nXDREQ0 和 nXDREQ1 代表两个外部源（外部设备），I2SSDO 和 I2SSDI 分别代表 IIS 的传送和接收。

DMA 操作

DMA 为其运行使用 3 状态 FSM (有限状态机) , 相关描述如下面三个阶段 :

状态1. 作为一个初始状态 , DMA 等待 DMA 请求。一旦请求到达则跳到状态 2。在此状态下 DMA ACK 和 INT REQ 为 0。

状态2. 在此状态 , DMA ACK 变为 1 而且计数器 (CURR_TC) 从 DCON[19:0] 寄存器中加载。注意 DMA ACK 保持为 1 直到之后将其清除。

状态3. 在此状态 , 处理 DMA 的原子操作的 sub-FSM 启动。sub-FSM 从源地址读取数据 , 接着写入目标地址。在此操作中考虑数据大小和传输大小(单次或突发)。此操作在全服务模式中一直重复直到计数器(CURR_TC) 变为 0 在单服务模式只执行一次。当 sub-FSM 完成了每个原子操作时主 FSM(此 FSM)倒计数 CURR_TC。此外当 CURR_TC 变为 0 并且 DCON[29] 寄存器的中断设置置位为 1 时主 FSM 发出 INT REQ 信号。另外如果遇到以下状况之一则清除了 DMA ACK。

- 1) 在全服务模式中 CURR_TC 变为 0 ;
- 2) 在单服务模式中完成原子操作。

注意在单服务模式中有三个主 FSM 的状态要执行并且接着要停止和等待其它 DMA REQ。如果 DMA REQ 出现了要重复所有的三个状态。所以发出 DMA ACK 并接着取消原子传输。与之对比 , 在全服务模式中 , 主 FSM 在状态 3 中等待直到 CURR_TC 变为 0。所以在所有传输期间发出 DMA ACK 并接着在当 TC 到达 0 时取消。

总之当且仅当在 CURR_TC 变为 0 时才发出 INT REQ , 与服务模式 (单顾服务模式或全顾服务模式) 无关。

外部 DMA 请求/应答 (DREQ/DACK) 协议

有 3 种外部 DMA 请求/应答协议类型 (单服务查询 , 单服务握手和全服务握手模式)。这些协议的每种类型都定义了描述信号如何像 DMA 一样请求和应答。

基本 DMA 时序

DMA 服务意味着在 DMA 运行期间执行成对的读取和写入周期 形成单次 DMA 操作。图 8-1 显示了 S3C2440A 的 DMA 操作的基本时序。

- XnXDREQ 和 XnXDACK 的建立时间和延迟时间在所有模式中都相同。
- 如果 XnXDREQ 的完成遇到其建立时间 , 它将同步两次并接着发出 XnXDACK。
- 发出 XnXDACK 后 , DMA 请求总线并且其如果得到总线将执行其的操作。

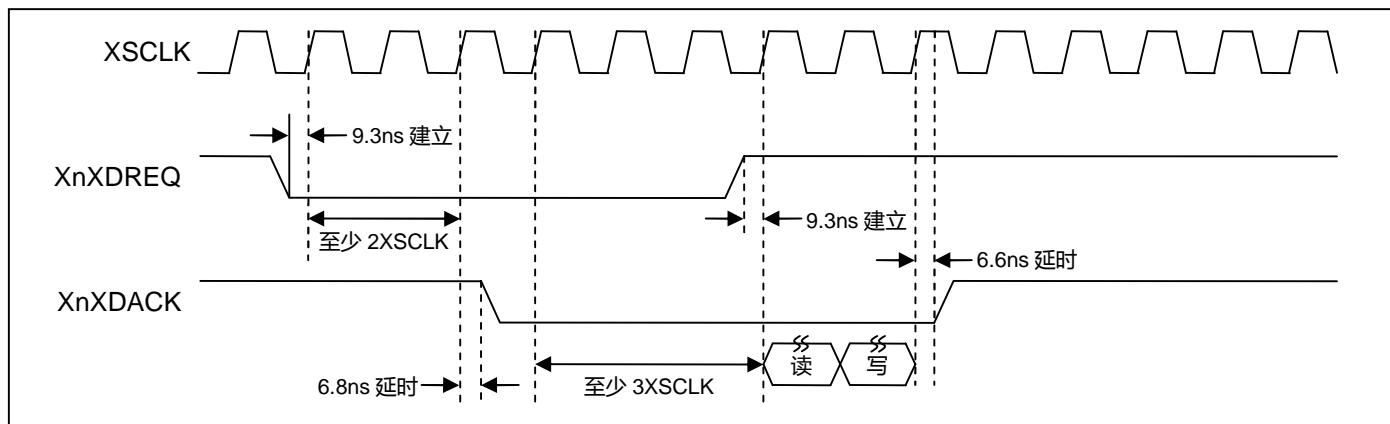


图 8-1. 基本 DMA 时序框图

查询/握手模式对比

查询和握手模式是描述 XnXDREQ 和 XnXDACK 之间的协议。图 8-2 显示了两个模式间的差异。

一次传输（单次/突发传输）的最后，DMA 将检查双同步 XnXDREQ 的状态。

查询模式

- 如果保持 XnXDREQ 的发出，立即开始下次的传输。否则其等待 XnXDREQ 的发出。

握手模式

- 如果发出 XnXDREQ，DMA 在 2 个周期内取消 XnXDACK。否则在取消 XnXDREQ 前一直等待

警告：在取消 XnXDACK（高电平）后就必须发出 XnXDREQ（低电平）。

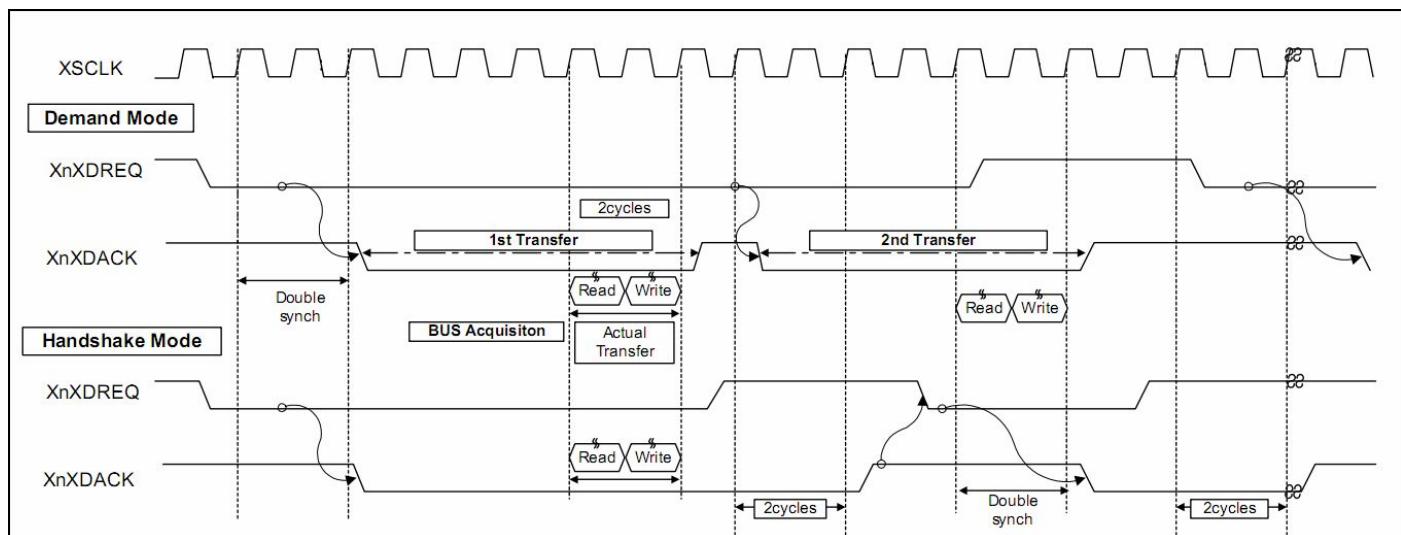


图 8-2. 查询/握手模式对比

传输大小

- 有两种不同的传输大小：单元 (unit) 和突发 4 (Burst 4)。
- DMA 在大块数据传输期间将牢牢的掌握总线。因此其它总线主机不能得到总线。

突发 4 传输大小

在突发 4 传输中分别可以执行 4 种连续读取和写入。

注释：

单元传输大小：执行一次读取和一次写入。

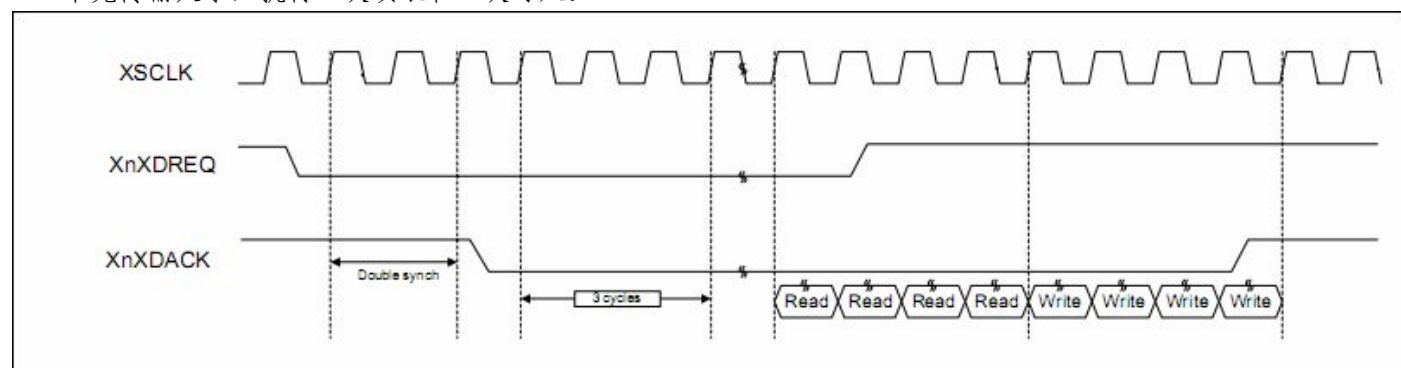


图 8-3. 突发 4 传输大小

例子

单元传输大小下查询模式的单服务

每种单元传输（单服务模式）都将需要 XnXDREQ 的发出。当发出 XnXDREQ（查询模式）继续运行并执行了一对读取和写入（单传输大小）。

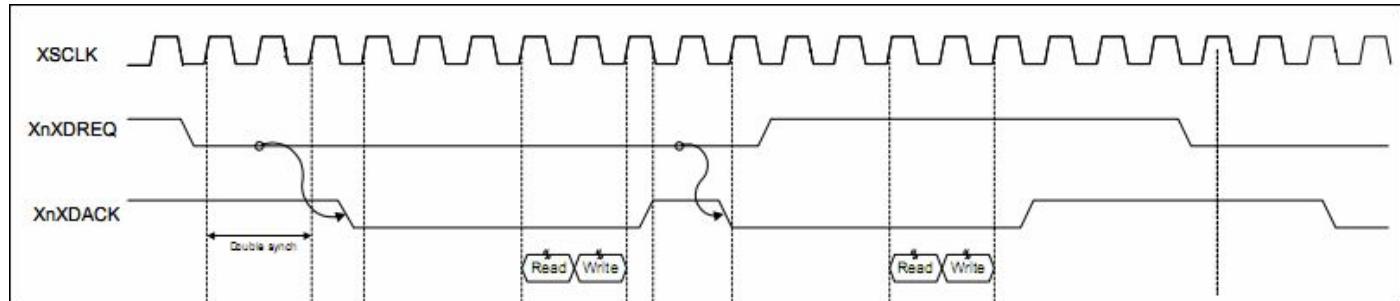


图 8-4. 单元传输大小下查询模式的单服务

单元传输大小下握手模式的单服务

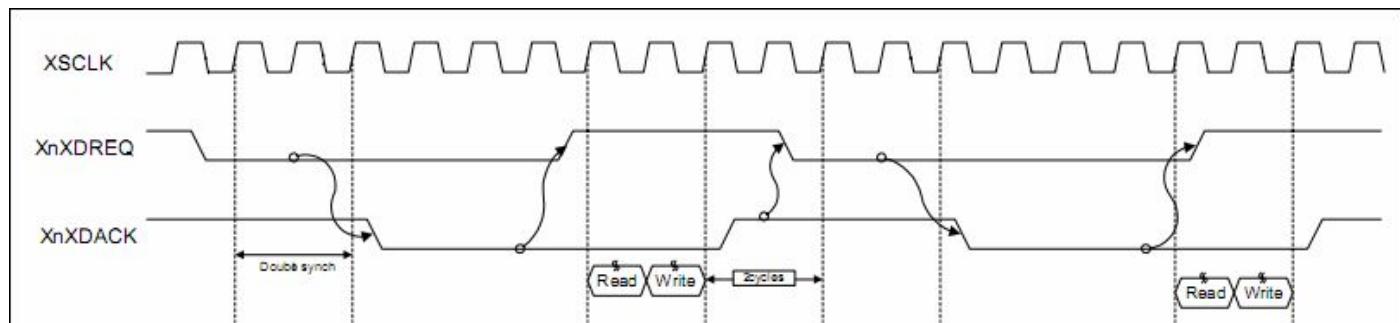


图 8-5. 单元传输大小下握手模式的单服务

单元传输大小下握手模式的全服务

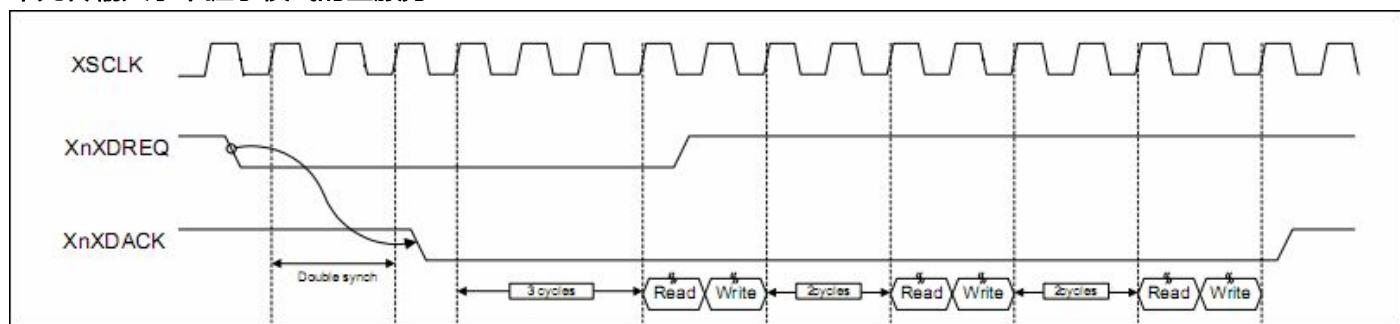


图 8-6. 单元传输大小下握手模式的全服务

DMA 特殊寄存器

每个 DMA 通道都有 9 个控制寄存器（总计 36 个，因为 DMA 控制器有 4 个通道）。其中 6 个控制寄存器控制 DMA 的传输，另外 3 个监视 DMA 控制器的状态。这些寄存器的详情如下。

DMA 初始源 (DISRC) 寄存器

寄存器	地址	R/W	描述	复位值
DISRC0	0x4B000000	R/W	DMA 0 初始源寄存器	0x00000000
DISRC1	0x4B000040	R/W	DMA 1 初始源寄存器	0x00000000
DISRC2	0x4B000080	R/W	DMA 2 初始源寄存器	0x00000000
DISRC3	0x4B0000C0	R/W	DMA 3 初始源寄存器	0x00000000

DISRCn	位	描述	初始状态
S_ADDR	[30:0]	要传输的源数据基本地址（开始地址）。当且仅当 CURR_SRC 为 0 并且 DMA ACK 为 1 时将此位的值锁存到 CURR_SRC 中。	0x00000000

DMA 初始源控制 (DISRCC) 寄存器

寄存器	地址	R/W	描述	复位值
DISRCC0	0x4B000004	R/W	DMA 0 初始源控制寄存器	0x00000000
DISRCC1	0x4B000044	R/W	DMA 1 初始源控制寄存器	0x00000000
DISRCC2	0x4B000084	R/W	DMA 2 初始源控制寄存器	0x00000000
DISRCC3	0x4B0000C4	R/W	DMA 3 初始源控制寄存器	0x00000000

DISRCCn	位	描述	初始状态
LOC	[1]	位[1]是用于源位置的选择 0 : 源在系统总线 (AHB) 上 1 : 源在外设总线 (APB) 上	0
INC	[0]	位[0]是用于地址增加的选择 0 : 增加 1 : 固定 如果为 0，地址将根据单次和突发模式中每次传输后其数据大小而增加。 如果为 1，在传输后地址不改变。（突发模式中，地址只在突发传输期间增加，但在传输后又回到其第一个值。）	0

DMA 初始目标 (DIDST) 寄存器

寄存器	地址	R/W	描述	复位值
DIDST0	0x4B000008	R/W	DMA 0 初始目标寄存器	0x00000000
DIDST1	0x4B000048	R/W	DMA 1 初始目标寄存器	0x00000000
DIDST2	0x4B000088	R/W	DMA 2 初始目标寄存器	0x00000000
DIDST3	0x4B0000C8	R/W	DMA 3 初始目标寄存器	0x00000000

DIDSTn	位	描述	初始状态
D_ADDR	[30:0]	要传输的目标基本地址 (开始地址)。当且仅当 CURR_DST 为 0 并且 DMA ACK 为 1 时将此位的值锁存到 CURR_SRC 中。	0x00000000

DMA 初始目标控制 (DIDSTC) 寄存器

寄存器	地址	R/W	描述	复位值
DIDSTC0	0x4B00000C	R/W	DMA 0 初始目标控制寄存器	0x00000000
DIDSTC1	0x4B00004C	R/W	DMA 1 初始目标控制寄存器	0x00000000
DIDSTC2	0x4B00008C	R/W	DMA 2 初始目标控制寄存器	0x00000000
DIDSTC3	0x4B0000CC	R/W	DMA 3 初始目标控制寄存器	0x00000000

DISRCCn	位	描述	初始状态
CHK_INT	[2]	当设置了自动再加载时发生中断的时间选择 0 : 在 TC 到达 0 时发生中断 1 : 在执行完自动再加载后发生中断	0
LOC	[1]	位[1]是用于目标位置的选择 0 : 目标在系统总线 (AHB) 上 1 : 目标在外设总线 (APB) 上	0
INC	[0]	位[0]是用于地址增加的选择 0 : 增加 1 : 固定 如果为 0 , 地址将根据单次和突发模式中每次传输后其数据大小而增加。 如果为 1 , 在传输后地址不改变。 (突发模式中 , 地址只在突发传输期间 增加 , 但在传输后又回到其第一个值。)	0

DMA 控制 (DCON) 寄存器

寄存器	地址	R/W	描述	复位值
DCON0	0x4B000010	R/W	DMA 0 控制寄存器	0x00000000
DCON1	0x4B000050	R/W	DMA 1 控制寄存器	0x00000000
DCON2	0x4B000090	R/W	DMA 2 控制寄存器	0x00000000
DCON3	0x4B0000D0	R/W	DMA 3 控制寄存器	0x00000000

DCONn	位	描述	初始状态
DMD_HS	[31]	查询模式和握手模式之间选择其中之一 0 : 选择查询模式 1 : 选择握手模式 两种模式下 DMA 控制器开始其传输并为发出的 DREQ 而发出 DACK。 两种模式之间的差异为是否需要等待取消 DACK。握手模式中，DMA 控制器在开始新的传输前等待取消 DREQ。如果其发现了取消 DREQ，其取消 DACK 并等待另一个 DREQ 的发出。与之对比，查询模式中 DMA 控制器不等待直到取消 DREQ。其只是取消 DACK 并且如果发出 DACK 接着开始另一个传输。我们建议外部 DMA 请求源使用握手模式以预防新传输的非预定开始。	0
SYNC	[30]	DREQ/DACK 的同步化选择 0 : DREQ 和 DACK 同步于 PCLK (APB 时钟) 1 : DREQ 和 DACK 同步于 HCLK (AHB 时钟) 因此如果有设备附加在 AHB 系统总线上时必须将此位设置为 1，当这些设备附加在 APB 系统上它将被设置为 0。设备附加在外部系统时用户应当按外部系统是同步于 AHB 系统还是 APB 系统来选择此位。	0
INT	[29]	CURR_TC (终点计数)的中断使能/禁止设置 0 : 禁止 CURR_TC 中断。用户必须观察状态寄存器的传输计数 (即定时查询) 1 : 当所有传输完成产生中断请求 (即 CURR_TC 变为 0)	0
TSZ	[28]	一个原子传输的传输大小选择 (即释放总线前每次 DMA 拥有总线执行传输) 0 : 执行一个单元传输 1 : 执行一个长度为 4 的突发传输	0
SERVMODE	[27]	单服务模式和全服务模式之间的服务模式选择 0 : 选择每次原子传输 (单次或突发 4) 后 DMA 停止和等待其它 DMA 请求的单服务模式。 1 : 选择传输计数达到 0 前重复请求得到原子传输的全服务模式。此模式不需要额外请求。 注意即使在全服务模式中，在每个原子传输后 DMA 释放总线并且为了预防其它总线主机的渴望得到总线而接着试图重新得到总线。	0

DMA 控制 (DCON) 寄存器 (续)

HWSRCSEL	[26:24]	每个 DMA 的 DMA 请求源的选择。	00	
		DCON0 : 000 : nXDREQ0 001 : UART0 010 : SDI 011 : Timer 100 : USB device EP1 101 : I2SSDO 110 : PCMIN		
		DCON1 : 000 : nXDREQ1 001 : UART1 010 : I2SSDI 011 : SPI 100 : USB device EP2 101 : PCMOUT 110 : SDI		
		DCON2 : 000 : I2SSDO 001 : I2SSDI 010 : SDI 011 : Timer 100 : USB device EP3 101 : PCMIN 110 : MICIN		
DCON3 : 000 : UART2 001 : SDI 010 : SPI 011 : Timer 100 : USB device EP4 101 : MICIN 110 : PCMOUT				
这些位控制 4 选 1 多路选择器 (4-1 MUX) 选择每个 DMA 的 DMA 请求源。当且仅当由 DCONn[23]选择了硬件请求模式时这些位才有意义。				
SWHW_SEL	[23]	DMA 源为软件 (软件请求模式) 或硬件 (硬件请求模式) 选择 0 : 选择软件请求模式并且 DMA 由 DMASKTRIG 控制寄存器的 SW_TRIG 位的置位触发 1 : 由位[26:24]选择 DMA 源触发 DMA 操作	0	
RELOAD	[22]	设置再加载开/关选项 0 : 当传输计数的当前值变为 0 时 (即执行了所有请求的传输) 执行自动再加载 1 : 当传输计数的当前值变为 0 时 DMA 通道 (DMA REQ) 关闭。设置通道开/关位 (DMASKTRIGN[1]) 为 0 (DREQ 关闭) 来预防非预定的新 DMA 操作的进一步开始	0	
DSZ	[21:20]	要传输的数据大小 00 = 字节 01 = 半字 10 = 字 11 = 保留	00	
TC	[19:0]	初始传输计数 (或传输节拍) 注意其实际传输的字节数由以下等式计算得到 : SZxTSZxTC。此处 DSZ , TSZ (1 或 4) 和 TC 分别代表数据大小 DCONn[21:20] , 传输大小 DCONn[28]和初始传输计数。当且仅当 CURR_TC 为 0 并且 DMA ACK 为 1 时该值将被加载到 CURR_TC。	00000	

DMA 状态 (DSTAT) 寄存器

寄存器	地址	R/W	描述	复位值
DSTAT0	0x4B000014	R	DMA 0 计数寄存器	000000h
DSTAT1	0x4B000054	R	DMA 1 计数寄存器	000000h
DSTAT2	0x4B000094	R	DMA 2 计数寄存器	000000h
DSTAT3	0x4B0000D4	R	DMA 3 计数寄存器	0000000h

DSTATn	位	描述	初始状态
STAT	[21:20]	此 DMA 控制器的状态 00 : 表明 DMA 控制器为就绪，等待其它 DMA 请求 01 : 表明 DMA 控制器正忙于传输	00b
CURR_TC	[19:0]	传输计数的当前值。注意传输计数是最初设置 DCONn[19:0]寄存器的值 在每个原子传输结束时减一的值。	00000h

DMA 当前源 (DCSRC) 寄存器

寄存器	地址	R/W	描述	复位值
DCSRC0	0x4B000018	R	DMA 0 当前源寄存器	0x00000000
DCSRC1	0x4B000058	R	DMA 1 当前源寄存器	0x00000000
DCSRC2	0x4B000098	R	DMA 2 当前源寄存器	0x00000000
DCSRC3	0x4B0000D8	R	DMA 3 当前源寄存器	0x00000000

DCSRCn	位	描述	初始状态
CURR_SRC	[30:0]	DMA _n 当前源地址	0x00000000

DMA 当前目标 (DCDST) 寄存器

寄存器	地址	R/W	描述	复位值
DCDST0	0x4B00001C	R	DMA 0 当前目标寄存器	0x00000000
DCDST1	0x4B00005C	R	DMA 1 当前目标寄存器	0x00000000
DCDST2	0x4B00009C	R	DMA 2 当前目标寄存器	0x00000000
DCDST3	0x4B0000DC	R	DMA 3 当前目标寄存器	0x00000000

DCSRCn	位	描述	初始状态
CURR_DST	[30:0]	DMA _n 当前目标地址	0x00000000

DMA 触发屏蔽 (DMASKTRIG) 寄存器

寄存器	地址	R/W	描述	复位值
DMASKTRIG0	0x4B000020	R/W	DMA 0 触发屏蔽寄存器	000
DMASKTRIG1	0x4B000060	R/W	DMA 1 触发屏蔽寄存器	000
DMASKTRIG2	0x4B0000A0	R/W	DMA 2 触发屏蔽寄存器	000
DMASKTRIG3	0x4B0000E0	R/W	DMA 3 触发屏蔽寄存器	000

DMASKTRIGn	位	描述	初始状态
STOP	[2]	<p>停止 DMA 的运行</p> <p>1 :一旦结束了当前的原子传输就停止 DMA。如果当前无运行的原子传输 ,立即停止 DMA。 CURR_TC , CURR_SRC 和 CURR_DST 将变为 0。</p> <p>注意：由于当前可能有原子传输，“停止”操作将会耗费若干周期。一旦通道开/关位 (DMASKTRIGn[1]) 设置为关闭就检测操作的结束（既实际停止时间）。此停止为“实际停止”。</p>	0
ON_OFF	[1]	<p>DMA 通道开/关位</p> <p>0 : 关闭 DMA 通道。(忽略此通道 DMA 请求)</p> <p>1 : 打开 DMA 通道并且处理此通道 DMA 请求。</p> <p>如果我们设置 DCONn[22]位为“非自动再加载”与/或 DMASKTRIGn 的停止位为“停止”时此位被自动设置为关闭，当 CURR_TC 到达 0 时此位变为 0。如果停止位为 1 时，一旦完成当前原子传输此位就变为 0。</p> <p>注意：此位在 DMA 运行期间不应当被手动修改（即只有在使用 DCON[22]或停止位时才可以改变此位）。</p>	0
SW_TRIG	[0]	<p>软件请求模式中触发 DMA 通道</p> <p>1 : 为此控制器请求 DMA 操作</p> <p>注意此触发在选择了软件请求模式 (DCONn[23]) 并且通道开/关被设置为 1 (通道打开) 时才有效。当 DMA 操作开始，此位将被自动清零。</p>	0

注意：

你可以改变 DISRC 寄存器、DIDST 寄存器和 DCON 寄存器中 TC 字段的值。这些改变只在结束当前传输后(即当 CURR_TC 变为 0 时)产生结果。另一方面，任何改动其它寄存器或字段将立即生效。因此要小心这些寄存器和字段的改变。

9

输入/输出端口

概述

S3C2440A 包含了 130 个多功能输入/输出口引脚并且它们为如下显示的八个端口：

- 端口 A (GPA) : 25 位输出端口
- 端口 B (GPB) : 11 位输入/输出端口
- 端口 C (GPC) : 16 位输入/输出端口
- 端口 D (GPD) : 16 位输入/输出端口
- 端口 E (GPE) : 16 位输入/输出端口
- 端口 F (GPF) : 8 位输入/输出端口
- 端口 G (GPG) : 16 位输入/输出端口
- 端口 H (GPH) : 9 位输入/输出端口
- 端口 J (GPJ) : 13 位输入/输出端口

每个端口都可以简单的由软件配置为各种系统配置和设计要求。你必须在开始主程序前定义使用的每个引脚的功能。如果没有使用某个引脚的复用功能，这个引脚可以配置为 I/O 口。

首先无缝的 (*seamlessly*) 配置引脚状态以避免出现问题。

表 9-1. S3C2440A 端口配置 (1/5)

端口 A	可选引脚功能			
GPA22	只输出	<u>nFCE</u>	-	-
GPA21	只输出	<u>nRSTOUT</u>	-	-
GPA20	只输出	<u>nFRE</u>	-	-
GPA19	只输出	<u>nFWE</u>	-	-
GPA18	只输出	<u>ALE</u>	-	-
GPA17	只输出	<u>CLE</u>	-	-
GPA16	只输出	<u>nGCS5</u>	-	-
GPA15	只输出	<u>nGCS4</u>	-	-
GPA14	只输出	<u>nGCS3</u>	-	-
GPA13	只输出	<u>nGCS2</u>	-	-
GPA12	只输出	<u>nGCS1</u>	-	-
GPA11	只输出	<u>ADDR26</u>	-	-
GPA10	只输出	<u>ADDR25</u>	-	-
GPA9	只输出	<u>ADDR24</u>	-	-
GPA8	只输出	<u>ADDR23</u>	-	-
GPA7	只输出	<u>ADDR22</u>	-	-
GPA6	只输出	<u>ADDR21</u>	-	-
GPA5	只输出	<u>ADDR20</u>	-	-
GPA4	只输出	<u>ADDR19</u>	-	-
GPA3	只输出	<u>ADDR18</u>	-	-
GPA2	只输出	<u>ADDR17</u>	-	-
GPA1	只输出	<u>ADDR16</u>	-	-
GPA0	只输出	<u>ADDR0</u>	-	-

表 9-1. S3C2440A 端口配置 (2/5)

端口 B	可选引脚功能			
GPB10	输入/输出	nXDREQ0	-	-
GPB9	输入/输出	nXDACK0	-	-
GPB8	输入/输出	nXDREQ1	-	-
GPB7	输入/输出	nXDACK1	-	-
GPB6	输入/输出	nXBREQ	-	-
GPB5	输入/输出	nXBACK	-	-
GPB4	输入/输出	TCLK0	-	-
GPB3	输入/输出	TOUT3	-	-
GPB2	输入/输出	TOUT2	-	-
GPB1	输入/输出	TOUT1	-	-
GPB0	输入/输出	TOUT0	-	-

端口 C	可选引脚功能			
GPC15	输入/输出	VD7	-	-
GPC14	输入/输出	VD6	-	-
GPC13	输入/输出	VD5	-	-
GPC12	输入/输出	VD4	-	-
GPC11	输入/输出	VD3	-	-
GPC10	输入/输出	VD2	-	-
GPC9	输入/输出	VD1	-	-
GPC8	输入/输出	VD0	-	-
GPC7	输入/输出	LCD_LPCREVB	-	-
GPC6	输入/输出	LCD_LPCREV	-	-
GPC5	输入/输出	LCD_LPCOE	-	-
GPC4	输入/输出	VM	-	-
GPC3	输入/输出	VFRAME	-	-
GPC2	输入/输出	VLINE	-	-
GPC1	输入/输出	VCLK	-	-
GPC0	输入/输出	LEND	-	-

表 9-1. S3C2440A 端口配置 (3/5)

端口 D	可选引脚功能			
GPD15	输入/输出	<u>VD23</u>	nSS0	-
GPD14	输入/输出	<u>VD22</u>	nSS1	-
GPD13	输入/输出	<u>VD21</u>	-	-
GPD12	输入/输出	<u>VD20</u>	-	-
GPD11	输入/输出	<u>VD19</u>	-	-
GPD10	输入/输出	<u>VD18</u>	SPICLK1	-
GPD9	输入/输出	<u>VD17</u>	SPIMOSI1	-
GPD8	输入/输出	<u>VD16</u>	SPIMISO1	-
GPD7	输入/输出	<u>VD15</u>	-	-
GPD6	输入/输出	<u>VD14</u>	-	-
GPD5	输入/输出	<u>VD13</u>	-	-
GPD4	输入/输出	<u>VD12</u>	-	-
GPD3	输入/输出	<u>VD11</u>	-	-
GPD2	输入/输出	<u>VD10</u>	-	-
GPD1	输入/输出	<u>VD9</u>	-	-
GPD0	输入/输出	<u>VD8</u>	-	-

端口 E	可选引脚功能			
GPE15	输入/输出	<u>IICSDA</u>	-	-
GPE14	输入/输出	<u>IICSCL</u>	-	-
GPE13	输入/输出	<u>SPICLK0</u>	-	-
GPE12	输入/输出	<u>SPIMOSIO</u>	-	-
GPE11	输入/输出	<u>SPIMISO0</u>	-	-
GPE10	输入/输出	<u>SDDAT3</u>	-	-
GPE9	输入/输出	<u>SDDAT2</u>	-	-
GPE8	输入/输出	<u>SDDAT1</u>	-	-
GPE7	输入/输出	<u>SDDAT0</u>	-	-
GPE6	输入/输出	<u>SDCMD</u>	-	-
GPE5	输入/输出	<u>SDCLK</u>	-	-
GPE4	输入/输出	<u>I2SSDO</u>	AC_SDATA_OUT	-
GPE3	输入/输出	<u>I2SSDI</u>	AC_SDATA_IN	-
GPE2	输入/输出	<u>CDCLK</u>	AC_nRESET	-
GPE1	输入/输出	<u>I2SSCLK</u>	AC_BIT_CLK	-
GPE0	输入/输出	<u>I2SLRCK</u>	AC_SYNC	-

表 9-1. S3C2440A 端口配置 (4/5)

可选引脚功能				
端口 F				
GPF7	输入/输出	EINT7	-	-
GPF6	输入/输出	EINT6	-	-
GPF5	输入/输出	EINT5	-	-
GPF4	输入/输出	EINT4	-	-
GPF3	输入/输出	EINT3	-	-
GPF2	输入/输出	EINT2	-	-
GPF1	输入/输出	EINT1	-	-
GPF0	输入/输出	EINT0	-	-

可选引脚功能				
端口 G				
PGP15	输入/输出	EINT23	-	-
PGP14	输入/输出	EINT22	-	-
PGP13	输入/输出	EINT21	-	-
PGP12	输入/输出	EINT20	-	-
PGP11	输入/输出	EINT19	TCLK1	-
PGP10	输入/输出	EINT18	nCTS1	-
PGP9	输入/输出	EINT17	nRTS1	-
PGP8	输入/输出	EINT16	-	-
PGP7	输入/输出	EINT15	SPICLK1	-
PGP6	输入/输出	EINT14	SPI MOSI1	-
PGP5	输入/输出	EINT13	SPI MISO1	-
PGP4	输入/输出	EINT12	LCD_PWREN	-
PGP3	输入/输出	EINT11	nSS1	-
PGP2	输入/输出	EINT10	nSS0	-
PGP1	输入/输出	EINT9	-	-
PGP0	输入/输出	EINT8	-	-

表 9-1. S3C2440A 端口配置 (1/5)

端口 H	可选引脚功能			
GPH10	输入/输出	<u>CLKOUT1</u>	-	-
GPH9	输入/输出	<u>CLKOUT0</u>	-	-
GPH8	输入/输出	<u>UEXTCLK</u>	-	-
GPH7	输入/输出	<u>RXD2</u>	nCTS1	-
GPH6	输入/输出	<u>TXD2</u>	nRTS1	-
GPH5	输入/输出	<u>RXD1</u>	-	-
GPH4	输入/输出	<u>TXD1</u>	-	-
GPH3	输入/输出	<u>RXD0</u>	-	-
GPH2	输入/输出	<u>TXD0</u>	-	-
GPH1	输入/输出	<u>nRTS0</u>	-	-
GPH0	输入/输出	<u>nCTS0</u>	-	-

端口 J	可选引脚功能			
GPJ12	输入/输出	<u>CAMRESET</u>	-	-
GPJ11	输入/输出	<u>CAMCLKOUT</u>	-	-
GPJ10	输入/输出	<u>CAMHREF</u>	-	-
GPJ9	输入/输出	<u>CAMVSYNC</u>	-	-
GPJ8	输入/输出	<u>CAMPCLK</u>	-	-
GPJ7	输入/输出	<u>CAMDATA7</u>	-	-
GPJ6	输入/输出	<u>CAMDATA6</u>	-	-
GPJ5	输入/输出	<u>CAMDATA5</u>	-	-
GPJ4	输入/输出	<u>CAMDATA4</u>	-	-
GPJ3	输入/输出	<u>CAMDATA3</u>	-	-
GPJ2	输入/输出	<u>CAMDATA2</u>	-	-
GPJ1	输入/输出	<u>CAMDATA1</u>	-	-
GPJ0	输入/输出	<u>CAMDATA0</u>	-	-

端口控制描述

端口配置寄存器 (GPACON 至 GPJCON)

S3C2440A 中，大多数端口为复用引脚。因此要决定每个引脚选择哪项功能。PnCON (引脚控制寄存器) 决定了每个引脚使用哪项功能。

如果在掉电模式中 PE0 至 PE7 用于唤醒信号，这些端口必须配置为输入模式。

端口数据寄存器 (GPADAT 至 GPJDAT)

如果端口配置为输出端口，可以写入数据到 PnDAT 的相应位。如果端口配置为输入端口，可以从 PnDAT 的相应位读取数据。

端口上拉寄存器 (GPBUP 至 GPJUP)

端口上拉寄存器控制每个端口组的使能/禁止上拉电阻。当相应位为 0 时使能引脚的上拉电阻。当为 1 时禁止上拉电阻。

如果使能了上拉电阻，那么上拉电阻与引脚的功能设置无关（输入、输出、DATA_n、EINT_n 等等）

杂项控制寄存器

此寄存器控制睡眠模式，USB 引脚和 CLKOUT 选择的数据端口上拉电阻。

外部中断控制寄存器

24 个外部中断由各种信号方式触发。EXTINT 寄存器为外部中断请求配置信号触发方式为低电平触发、高电平触发、下降沿触发、上升沿触发或双边沿触发。

由于每个外部中断引脚包含一个数字滤波器，中断控制可以确认请求信号是否长于 3 个时钟。

EINT[15:0]用于唤醒源

I/O 口控制寄存器

端口 A 控制寄存器 (GPACON , GPADAT)

寄存器	地址	R/W	描述	复位值
GPACON	0x56000000	R/W	配置端口 A 的引脚	0xFFFFFFFF
GPADAT	0x56000004	R/W	端口 A 的数据寄存器	-
保留	0x56000008	-	保留	-
保留	0x5600000C	-	保留	-

GPACON	位	描述	初始状态
GPA24	[24]	保留	1
GPA23	[23]	保留	1
GPA22	[22]	0 = 输出 1 = nFCE	1
GPA21	[21]	0 = 输出 1 = nRSTOUT	1
GPA20	[20]	0 = 输出 1 = nFRE	1
GPA19	[19]	0 = 输出 1 = nFWE	1
GPA18	[18]	0 = 输出 1 = ALE	1
GPA17	[17]	0 = 输出 1 = CLE	1
GPA16	[16]	0 = 输出 1 = nGCS[5]	1
GPA15	[15]	0 = 输出 1 = nGCS[4]	1
GPA14	[14]	0 = 输出 1 = nGCS[3]	1
GPA13	[13]	0 = 输出 1 = nGCS[2]	1
GPA12	[12]	0 = 输出 1 = nGCS[1]	1
GPA11	[11]	0 = 输出 1 = ADDR26	1
GPA10	[10]	0 = 输出 1 = ADDR25	1
GPA9	[9]	0 = 输出 1 = ADDR24	1
GPA8	[8]	0 = 输出 1 = ADDR23	1
GPA7	[7]	0 = 输出 1 = ADDR22	1
GPA6	[6]	0 = 输出 1 = ADDR21	1
GPA5	[5]	0 = 输出 1 = ADDR20	1
GPA4	[4]	0 = 输出 1 = ADDR19	1
GPA3	[3]	0 = 输出 1 = ADDR18	1
GPA2	[2]	0 = 输出 1 = ADDR17	1
GPA1	[1]	0 = 输出 1 = ADDR16	1
GPA0	[0]	0 = 输出 1 = ADDR0	1

GPADAT	位	描述	初始状态
GPA[24:0]	[24:0]	当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

注释 : nRSTOUT = nRESET & nWDTRST & SW_RESET

端口 B 控制寄存器 (GPBCON , GPBDAT , GPBUP)

寄存器	地址	R/W	描述	复位值
GPBCON	0x56000010	R/W	配置端口 B 的引脚	0x0
GPBDAT	0x56000014	R/W	端口 B 的数据寄存器	-
GPBUP	0x56000018	R/W	端口 B 的上拉使能寄存器	0x0
保留	0x5600001C	-	保留	-

GPBCON	位	描述				初始状态
GPB10	[21:20]	00 = 输入	01 = 输出	10 = nXDREQ0	11 = 保留	0
GPB9	[19:18]	00 = 输入	01 = 输出	10 = nXDACK0	11 = 保留	0
GPB8	[17:16]	00 = 输入	01 = 输出	10 = nXDREQ1	11 = 保留	0
GPB7	[15:14]	00 = 输入	01 = 输出	10 = nXDACK1	11 = 保留	0
GPB6	[13:12]	00 = 输入	01 = 输出	10 = nXBREQ	11 = 保留	0
GPB5	[11:10]	00 = 输入	01 = 输出	10 = nXBACK	11 = 保留	0
GPB4	[9:8]	00 = 输入	01 = 输出	10 = TCLK [0]	11 = 保留	0
GPB3	[7:6]	00 = 输入	01 = 输出	10 = TOUT3	11 = 保留	0
GPB2	[5:4]	00 = 输入	01 = 输出	10 = TOUT2	11 = 保留	0
GPB1	[3:2]	00 = 输入	01 = 输出	10 = TOUT1	11 = 保留	0
GPB0	[1:0]	00 = 输入	01 = 输出	10 = TOUT0	11 = 保留	0

GPBDAT	位	描述	初始状态
GPB[10:0]	[10:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

GPBUP	位	描述	初始状态
GPB[10:0]	[10:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚	0x0

端口 C 控制寄存器 (GPCCON , GPCDAT , GPCUP)

寄存器	地址	R/W	描述	复位值
GPCCON	0x56000020	R/W	配置端口 C 的引脚	0x0
GPCDAT	0x56000024	R/W	端口 C 的数据寄存器	-
GPCUP	0x56000028	R/W	端口 C 的上拉使能寄存器	0x0
保留	0x5600002C	-	保留	-

GPCCON	位	描述				初始状态
GPC15	[31:30]	00 = 输入	01 = 输出	10 = VD[7]	11 = 保留	0
GPC14	[29:28]	00 = 输入	01 = 输出	10 = VD[6]	11 = 保留	0
GPC13	[27:26]	00 = 输入	01 = 输出	10 = VD[5]	11 = 保留	0
GPC12	[25:24]	00 = 输入	01 = 输出	10 = VD[4]	11 = 保留	0
GPC11	[23:22]	00 = 输入	01 = 输出	10 = VD[3]	11 = 保留	0
GPC10	[21:20]	00 = 输入	01 = 输出	10 = VD[2]	11 = 保留	0
GPC9	[19:18]	00 = 输入	01 = 输出	10 = VD[1]	11 = 保留	0
GPC8	[17:16]	00 = 输入	01 = 输出	10 = VD[0]	11 = 保留	0
GPC7	[15:14]	00 = 输入	01 = 输出	10 = LCD_LPCREVB	11 = 保留	0
GPC6	[13:12]	00 = 输入	01 = 输出	10 = LCD_LPCREV	11 = 保留	0
GPC5	[11:10]	00 = 输入	01 = 输出	10 = LCD_LPCOE	11 = 保留	0
GPC4	[9:8]	00 = 输入	01 = 输出	10 = VM	11 = 保留	0
GPC3	[7:6]	00 = 输入	01 = 输出	10 = VFRAME	11 = 保留	0
GPC2	[5:4]	00 = 输入	01 = 输出	10 = VLINE	11 = 保留	0
GPC1	[3:2]	00 = 输入	01 = 输出	10 = VCLK	11 = 保留	0
GPC0	[1:0]	00 = 输入	01 = 输出	10 = LEND	11 = 保留	0

GPCDAT	位	描述	初始状态
GPC[15:0]	[15:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

GPCUP	位	描述	初始状态
GPC[15:0]	[15:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚	0x0

端口 D 控制寄存器 (GPDCON , GPDDAT , GPDUP)

寄存器	地址	R/W	描述	复位值
GPDCON	0x56000030	R/W	配置端口 D 的引脚	0x0
GPDDAT	0x56000034	R/W	端口 D 的数据寄存器	-
GPDUP	0x56000038	R/W	端口 D 的上拉使能寄存器	0xF000
保留	0x5600003C	-	保留	-

GPDCON	位	描述				初始状态
GPD15	[31:30]	00 = 输入	01 = 输出	10 = VD[23]	11 = nSS0	0
GPD14	[29:28]	00 = 输入	01 = 输出	10 = VD[22]	11 = nSS1	0
GPD13	[27:26]	00 = 输入	01 = 输出	10 = VD[21]	11 = 保留	0
GPD12	[25:24]	00 = 输入	01 = 输出	10 = VD[20]	11 = 保留	0
GPD11	[23:22]	00 = 输入	01 = 输出	10 = VD[19]	11 = 保留	0
GPD10	[21:20]	00 = 输入	01 = 输出	10 = VD[18]	11 = SPICLK1	0
GPD9	[19:18]	00 = 输入	01 = 输出	10 = VD[17]	11 = SPIMOSI1	0
GPD8	[17:16]	00 = 输入	01 = 输出	10 = VD[16]	11 = SPIMISO1	0
GPD7	[15:14]	00 = 输入	01 = 输出	10 = VD[15]	11 = 保留	0
GPD6	[13:12]	00 = 输入	01 = 输出	10 = VD[14]	11 = 保留	0
GPD5	[11:10]	00 = 输入	01 = 输出	10 = VD[13]	11 = 保留	0
GPD4	[9:8]	00 = 输入	01 = 输出	10 = VD[12]	11 = 保留	0
GPD3	[7:6]	00 = 输入	01 = 输出	10 = VD[11]	11 = 保留	0
GPD2	[5:4]	00 = 输入	01 = 输出	10 = VD[10]	11 = 保留	0
GPD1	[3:2]	00 = 输入	01 = 输出	10 = VD[9]	11 = 保留	0
GPD0	[1:0]	00 = 输入	01 = 输出	10 = VD[8]	11 = 保留	0

GPDDAT	位	描述	初始状态
GPD[15:0]	[15:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

GPDUP	位	描述	初始状态
GPD[15:0]	[15:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚	0xF000

端口 E 控制寄存器 (GPECON , GPEDAT , GPEUP)

寄存器	地址	R/W	描述	复位值
GPECON	0x56000040	R/W	配置端口 E 的引脚	0x0
GPEDAT	0x56000044	R/W	端口 E 的数据寄存器	-
GPEUP	0x56000048	R/W	端口 E 的上拉使能寄存器	0x0
保留	0x5600004C	-	保留	-

GPECON	位	描述				初始状态
GPE15	[31:30]	00 = 输入 01 = 输出 10 = IICSDA 11 = 保留 此引脚为开漏输出，没有上拉选项				0
GPE14	[29:28]	00 = 输入 01 = 输出 10 = IICSCL 11 = 保留 此引脚为开漏输出，没有上拉选项				0
GPE13	[27:26]	00 = 输入	01 = 输出	10 = SPICLK0	11 = 保留	0
GPE12	[25:24]	00 = 输入	01 = 输出	10 = SPIMOSI0	11 = 保留	0
GPE11	[23:22]	00 = 输入	01 = 输出	10 = SPIMISO0	11 = 保留	0
GPE10	[21:20]	00 = 输入	01 = 输出	10 = SDDAT3	11 = 保留	0
GPE9	[19:18]	00 = 输入	01 = 输出	10 = SDDAT2	11 = 保留	0
GPE8	[17:16]	00 = 输入	01 = 输出	10 = SDDAT1	11 = 保留	0
GPE7	[15:14]	00 = 输入	01 = 输出	10 = SDDAT0	11 = 保留	0
GPE6	[13:12]	00 = 输入	01 = 输出	10 = SDCMD	11 = 保留	0
GPE5	[11:10]	00 = 输入	01 = 输出	10 = SDCLK	11 = 保留	0
GPE4	[9:8]	00 = 输入	01 = 输出	10 = I2SDO	11 = AC_SDATA_OUT	0
GPE3	[7:6]	00 = 输入	01 = 输出	10 = I2SDI	11 = AC_SDATA_IN	0
GPE2	[5:4]	00 = 输入	01 = 输出	10 = CDCLK	11 = AC_nRESET	0
GPE1	[3:2]	00 = 输入	01 = 输出	10 = I2SSCLK	11 = AC_BIT_CLK	0
GPE0	[1:0]	00 = 输入	01 = 输出	10 = I2SLRCK	11 = AC_SYNC	0

GPEDAT	位	描述	初始状态
GPE[15:0]	[15:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

GPEUP	位	描述	初始状态
GPE[15:0]	[13:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚	0x0

端口 F 控制寄存器 (GPFCON , GPFDAT , GPFUP)

如果 GPF0 至 GPF7 在掉电模式中用于唤醒信号，端口将被设置为中断模式。

寄存器	地址	R/W	描述	复位值
GPFCON	0x56000050	R/W	配置端口 F 的引脚	0x0
GPFDAT	0x56000054	R/W	端口 F 的数据寄存器	-
GPFUP	0x56000058	R/W	端口 F 的上拉使能寄存器	0x00
保留	0x5600005C	-	保留	-

GPFCON	位	描述				初始状态
GPF7	[15:14]	00 = 输入	01 = 输出	10 = EINT[7]	11 = 保留	0
GPF6	[13:12]	00 = 输入	01 = 输出	10 = EINT[6]	11 = 保留	0
GPF5	[11:10]	00 = 输入	01 = 输出	10 = EINT[5]	11 = 保留	0
GPF4	[9:8]	00 = 输入	01 = 输出	10 = EINT[4]	11 = 保留	0
GPF3	[7:6]	00 = 输入	01 = 输出	10 = EINT[3]	11 = 保留	0
GPF2	[5:4]	00 = 输入	01 = 输出	10 = EINT[2]	11 = 保留	0
GPF1	[3:2]	00 = 输入	01 = 输出	10 = EINT[1]	11 = 保留	0
GPF0	[1:0]	00 = 输入	01 = 输出	10 = EINT[0]	11 = 保留	0

GPFDAT	位	描述			初始状态
GPF[7:0]	[7:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。			-

GPFUP	位	描述		初始状态
GPF[7:0]	[7:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚		0x00

端口 G 控制寄存器 (GPGCON , GPGDAT , GPGUP)

如果 GPG0 至 GPG7 在睡眠模式中用于唤醒信号，端口将被设置为中断模式。

寄存器	地址	R/W	描述	复位值
GPGCON	0x56000060	R/W	配置端口 G 的引脚	0x0
GPGDAT	0x56000064	R/W	端口 G 的数据寄存器	-
GPGUP	0x56000068	R/W	端口 G 的上拉使能寄存器	0xFC00
保留	0x5600006C	-	保留	-

GPEGCON	位	描述				初始状态
PGP15*	[31:30]	00 = 输入	01 = 输出	10 = EINT[23]	11 = 保留	0
PGP14*	[29:28]	00 = 输入	01 = 输出	10 = EINT[22]	11 = 保留	0
PGP13*	[27:26]	00 = 输入	01 = 输出	10 = EINT[21]	11 = 保留	0
PGP12	[25:24]	00 = 输入	01 = 输出	10 = EINT[20]	11 = 保留	0
PGP11	[23:22]	00 = 输入	01 = 输出	10 = EINT[19]	11 = TCLK[1]	0
PGP10	[21:20]	00 = 输入	01 = 输出	10 = EINT[18]	11 = nCTS1	0
PGP9	[19:18]	00 = 输入	01 = 输出	10 = EINT[17]	11 = nRTS1	0
PGP8	[17:16]	00 = 输入	01 = 输出	10 = EINT[16]	11 = 保留	0
PGP7	[15:14]	00 = 输入	01 = 输出	10 = EINT[15]	11 = SPICLK1	0
PGP6	[13:12]	00 = 输入	01 = 输出	10 = EINT[14]	11 = SPIMOSI1	0
PGP5	[11:10]	00 = 输入	01 = 输出	10 = EINT[13]	11 = SPIMISO1	0
PGP4	[9:8]	00 = 输入	01 = 输出	10 = EINT[12]	11 = LCD_PWRDN	0
PGP3	[7:6]	00 = 输入	01 = 输出	10 = EINT[11]	11 = nSS1	0
PGP2	[5:4]	00 = 输入	01 = 输出	10 = EINT[10]	11 = nSS0	0
PGP1	[3:2]	00 = 输入	01 = 输出	10 = EINT[9]	11 = 保留	0
PGP0	[1:0]	00 = 输入	01 = 输出	10 = EINT[8]	11 = 保留	0

注释：NAND Flash 引导启动模式中必须选择 GPG[15:13]为输入。

GPGDAT	位	描述	初始状态
PGP[15:0]	[15:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

GPGUP	位	描述	初始状态
PGP[15:0]	[15:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚	0xFC00

端口 H 控制寄存器 (GPHCON , GPHDAT , GPHUP)

寄存器	地址	R/W	描述	复位值
GPHCON	0x56000070	R/W	配置端口 H 的引脚	0x0
GPHDAT	0x56000074	R/W	端口 H 的数据寄存器	-
GPHUP	0x56000078	R/W	端口 H 的上拉使能寄存器	0x000
保留	0x5600007C	-	保留	-

GPHCON	位	描述				初始状态
GPH10	[21:20]	00 = 输入	01 = 输出	10 = CLKOUT1	11 = 保留	0
GPH9	[19:18]	00 = 输入	01 = 输出	10 = CLKOUT0	11 = 保留	0
GPH8	[17:16]	00 = 输入	01 = 输出	10 = UEXTCLK	11 = 保留	0
GPH7	[15:14]	00 = 输入	01 = 输出	10 = RXD[2]	11 = nCTS1	0
GPH6	[13:12]	00 = 输入	01 = 输出	10 = TXD[2]	11 = nRTS1	0
GPH5	[11:10]	00 = 输入	01 = 输出	10 = RXD[1]	11 = 保留	0
GPH4	[9:8]	00 = 输入	01 = 输出	10 = TXD[1]	11 = 保留	0
GPH3	[7:6]	00 = 输入	01 = 输出	10 = RXD[0]	11 = 保留	0
GPH2	[5:4]	00 = 输入	01 = 输出	10 = TXD[0]	11 = 保留	0
GPH1	[3:2]	00 = 输入	01 = 输出	10 = nRTS0	11 = 保留	0
GPH0	[1:0]	00 = 输入	01 = 输出	10 = nCTS0	11 = 保留	0

GPHDAT	位	描述	初始状态
GPH[10:0]	[10:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

GPHUP	位	描述	初始状态
GPH[10:0]	[10:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚	0x000

端口 J 控制寄存器 (GPJCON , GPJDAT , GPJUP)

寄存器	地址	R/W	描述	复位值
GPJCON	0x560000D0	R/W	配置端口 J 的引脚	0x0
GPJDAT	0x560000D4	R/W	端口 J 的数据寄存器	-
GPJUP	0x560000D8	R/W	端口 J 的上拉使能寄存器	0x0000
保留	0x560000DC	-	保留	-

GPEGCON	位	描述			初始状态
PGP12	[25:24]	00 = 输入	01 = 输出	10 = CAMRESET 11 = 保留	0
PGP11	[23:22]	00 = 输入	01 = 输出	10 = CAMCLKOUT 11 = TCLK[1]	0
PGP10	[21:20]	00 = 输入	01 = 输出	10 = CAMHREF 11 = nCTS1	0
PGP9	[19:18]	00 = 输入	01 = 输出	10 = CAMVSYNC 11 = nRTS1	0
PGP8	[17:16]	00 = 输入	01 = 输出	10 = CAMPCLK 11 = 保留	0
PGP7	[15:14]	00 = 输入	01 = 输出	10 = CAMDATA[7] 11 = SPICLK1	0
PGP6	[13:12]	00 = 输入	01 = 输出	10 = CAMDATA[6] 11 = SPIMOSI1	0
PGP5	[11:10]	00 = 输入	01 = 输出	10 = CAMDATA[5] 11 = SPIMISO1	0
PGP4	[9:8]	00 = 输入	01 = 输出	10 = CAMDATA[4] 11 = LCD_PWRDN	0
PGP3	[7:6]	00 = 输入	01 = 输出	10 = CAMDATA[3] 11 = nSS1	0
PGP2	[5:4]	00 = 输入	01 = 输出	10 = CAMDATA[2] 11 = nSS0	0
PGP1	[3:2]	00 = 输入	01 = 输出	10 = CAMDATA[1] 11 = 保留	0
PGP0	[1:0]	00 = 输入	01 = 输出	10 = CAMDATA[0] 11 = 保留	0

GPGDAT	位	描述	初始状态
PGP[12:0]	[12:0]	当端口配置为输入端口时，相应位为引脚状态。当端口配置为输出端口时，引脚状态将与相应位相同。当端口配置为功能引脚，将读取到未定义值。	-

GPGUP	位	描述	初始状态
PGP[12:0]	[12:0]	0 : 使能附加上拉功能到相应端口引脚 1 : 禁止附加上拉功能到相应端口引脚	0x000

杂项控制寄存器 (MISCCR)

睡眠模式中，数据总线 (D[31:0]或 D[15:0]) 可以被设置为高阻态和输出为'0'状态。但是由于 IO 口特性，数据总线上拉电阻必须被开启或关闭以降低功耗。D[31:0]引脚上拉电阻可以由 MISCCR 寄存器控制。

USB 主机或 USB 设备的与 USB 相关的引脚由此寄存器控制。

寄存器	地址	R/W	描述	复位值
MISCCR	0x56000080	R/W	杂项控制寄存器	0x10020

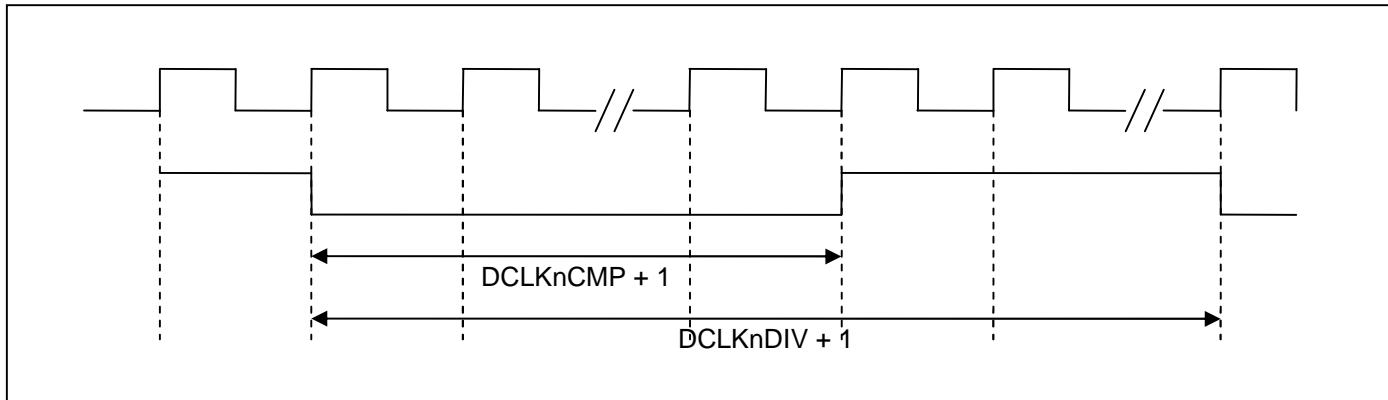
MISCCR	位	描述	初始状态
保留	[24]	保留为 0	0
保留	[23]	保留为 0	0
BATT_FUNC	[22:20]	电池故障选择。 0XX : nBATT_FLT=0 情况下，系统将处于复位状态。复位后，改变此位为其它值，此位只用于防止电池故障状态中的引导启动。 10X : 睡眠模式状态中，当 nBATT_FLT=0 时将唤醒系统。普通模式中，当 nBATT_FLT=0 时将发生电池故障中断。 110 : 睡眠模式状态中，在 nBATT_FLT=0 期间系统将忽略所有唤醒事件 (唤醒源唤醒)。普通模式中，nBATT_FLT 信号不会影响系统。 111 : nBATT_FLT 功能禁止。	000
OFFREFRESH	[19]	0 : 自刷新保持禁止 1 : 自刷新保持使能 当为 1 时，从睡眠模式中唤醒，将保持自刷新	0
nEN_SCLK1	[18]	SCLK1 输出使能 0 : SCLK1 = SCLK 1 : SCLK1 = 0	0
nEN_SCLK0	[17]	SCLK0 输出使能 0 : SCLK0 = SCLK 1 : SCLK0 = 0	0
nRSTCON	[16]	nRSTOUT 信号手动控制 0 : nRSTOUT 信号电平为低 ('0') 1 : nRSTOUT 信号电平为高 ('1')	1
保留	[15:14]	-	00
SEL_SUSPND1	[13]	USB 端口 1 挂起模式 0 = 普通模式 1 = 挂起模式	0
SEL_SUSPND0	[12]	USB 端口 0 挂起模式 0 = 普通模式 1 = 挂起模式	0
CLKSEL1 ⁽¹⁾	[10:8]	选择 CLKOUT1 引脚的源时钟 000 = MPLL 输出 001 = UPLL 输出 010 = RTC 时钟输出 011 = HCLK 100 = PCLK 101 = DCLK1 11x = 保留	000
保留	[7]	-	0
CLKSEL0 ⁽¹⁾	[6:4]	选择 CLKOUT0 引脚的源时钟 000 = MPLL 输入时钟 (XTAL) 001 = UPLL 输出 010 = FCLK 011 = HCLK 100 = PCLK 101 = DCLK0 11x = 保留	010
SEL_USBPAD	[3]	USB1 主机/设备选择寄存器 0 = 使用 USB1 为设备 1 = 使用 USB1 为主机	0
保留	[2]	-	0
SPUCR1	[1]	0 = DATA[31:16]端口上拉电阻使能	1 = DATA[31:16]端口上拉电阻禁止
SPUCR0	[0]	0 = DATA[15:0]端口上拉电阻使能	1 = DATA[15:0]端口上拉电阻禁止

注释 1) 我们建议不建议使用此输出脚作为其它设备的 PLL 时钟源。

DCLK 控制寄存器 (DCLKCON)

寄存器	地址	R/W	描述	复位值
DCLKCON	0x56000084	R/W	DCLK0/1 控制寄存器	0x00

DCLKCON	位	描述	初始状态
DCLK1CMP	[27:24]	DCLK1 比较值时钟触发值。(< DCLK1DIV) 如果 DCLK1CMP 为 n , 低电平持续为 (n+1) , 高电平持续为 ((DCLK1DIV + 1) - (n + 1))	0
DCLK1DIV	[23:20]	DCLK1 分频值 DCLK1 频率 = 源时钟 / (DCLK1DIV + 1)	
DCLK1SelCK	[17]	选择 DCLK1 源时钟 0 = PCLK 1 = UCLK (USB)	
DCLK1EN	[16]	DCLK1 使能 0 = DCLK1 禁止 1 = DCLK1 使能	
DCLK0CMP	[11:8]	DCLK0 比较值时钟触发值。(< DCLK0DIV) 如果 DCLK0CMP 为 n , 低电平持续为 (n+1) , 高电平持续为 ((DCLK0DIV + 1) - (n + 1))	
DCLK0DIV	[7:4]	DCLK0 分频值 DCLK0 频率 = 源时钟 / (DCLK0DIV + 1)	
DCLK0SelCK	[1]	选择 DCLK0 源时钟 0 = PCLK 1 = UCLK (USB)	
DCLK0EN	[0]	DCLK0 使能 0 = DCLK0 禁止 1 = DCLK0 使能	



EXTINTn (外部中断控制寄存器 n)

8 个外部中断可以由多种信号触发方式所请求。EXTINT 寄存器为外部中断配制信号触发方式为电平触发或边沿触发，同时还配制信号触发极性。

为了确认电平中断，由于噪声滤波必须保持 EXTINTn 引脚上有效逻辑电平至少 40ns。

寄存器	地址	R/W	描述	复位值
EXTINT0	0x56000088	R/W	外部中断控制寄存器 0	0x000000
EXTINT1	0x5600008C	R/W	外部中断控制寄存器 1	0x000000
EXTINT2	0x56000090	R/W	外部中断控制寄存器 2	0x000000

EXTINT0	位	描述	初始状态
EINT7	[30:28]	设置 EINT7 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
EINT6	[26:24]	设置 EINT6 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
EINT5	[22:20]	设置 EINT5 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
EINT4	[18:16]	设置 EINT4 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
EINT3	[14:12]	设置 EINT3 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
EINT2	[10:8]	设置 EINT2 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
EINT1	[6:4]	设置 EINT1 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
EINT0	[2:0]	设置 EINT0 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000

EXTINT1	位	描述	初始状态
FLTEN15	[31]	EINT15 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT15	[30:28]	设置 EINT15 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN14	[27]	EINT14 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT14	[26:24]	设置 EINT14 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN13	[23]	EINT13 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT13	[22:20]	设置 EINT13 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN12	[19]	EINT12 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT12	[18:16]	设置 EINT12 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN11	[15]	EINT11 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT11	[14:12]	设置 EINT11 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN10	[11]	EINT10 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT10	[10:8]	设置 EINT10 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN9	[7]	EINT9 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT9	[6:4]	设置 EINT9 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN8	[3]	EINT8 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT8	[2:0]	设置 EINT8 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000

EXTINT2	位	描述	初始状态
FLTEN23	[31]	EINT23 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT23	[30:28]	设置 EINT23 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN22	[27]	EINT22 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT22	[26:24]	设置 EINT22 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN21	[23]	EINT21 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT21	[22:20]	设置 EINT21 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN20	[19]	EINT20 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT20	[18:16]	设置 EINT20 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN19	[15]	EINT19 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT19	[14:12]	设置 EINT19 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN18	[11]	EINT18 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT18	[10:8]	设置 EINT18 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN17	[7]	EINT17 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT17	[6:4]	设置 EINT17 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000
FLTEN16	[3]	EINT16 的滤波器使能 0 = 滤波器禁止 1 = 滤波器使能	0
EINT16	[2:0]	设置 EINT16 的信号触发方式 000 = 低电平 001 = 高电平 01x = 下降沿触发 10x = 上升沿触发 11x = 双边沿触发	000

EINTFLT_n (外部中断滤波寄存器 n)

为了确认电平中断，由于噪声滤波必须保持 EXTINT_n 引脚上有效逻辑电平至少 40ns。

寄存器	地址	R/W	描述	复位值
EINTFLT0	0x56000094	R/W	保留	0x000000
EINTFLT1	0x56000098	R/W	保留	0x000000
EINTFLT2	0x5600009C	R/W	外部中断滤波寄存器 2	0x000000
EINTFLT3	0x560000A0	R/W	外部中断滤波寄存器 3	0x000000

EINTFLT2	位	描述	初始状态
EINTFLT19	[30:24]	EINT19 的滤波宽度	0x00
FLTCLK18	[23]	EINT18 的时钟 (由 OM 配制) 0 = PCLK 1 = EXTCLK/OSC_CLK	0
EINTFLT18	[22:16]	EINT18 的滤波宽度	0x00
FLTCLK17	[15]	EINT17 的时钟 (由 OM 配制) 0 = PCLK 1 = EXTCLK/OSC_CLK	0
EINTFLT17	[14:8]	EINT17 的滤波宽度	0x00
FLTCLK16	[7]	EINT16 的时钟 (由 OM 配制) 0 = PCLK 1 = EXTCLK/OSC_CLK	0
EINTFLT16	[6:0]	EINT16 的滤波宽度	0x00

EINTFLT3	位	描述	初始状态
FLTCLK23	[31]	EINT23 的时钟 (由 OM 配制) 0 = PCLK 1 = EXTCLK/OSC_CLK	0
EINTFLT23	[30:24]	EINT23 的滤波宽度	0x00
FLTCLK22	[23]	EINT22 的时钟 (由 OM 配制) 0 = PCLK 1 = EXTCLK/OSC_CLK	0
EINTFLT22	[22:16]	EINT22 的滤波宽度	0x00
FLTCLK21	[15]	EINT21 的时钟 (由 OM 配制) 0 = PCLK 1 = EXTCLK/OSC_CLK	0
EINTFLT21	[14:8]	EINT21 的滤波宽度	0x00
FLTCLK20	[7]	EINT20 的时钟 (由 OM 配制) 0 = PCLK 1 = EXTCLK/OSC_CLK	0
EINTFLT20	[6:0]	EINT20 的滤波宽度	0x00

EINTMASK (外部中断屏蔽寄存器)

寄存器	地址	R/W	描述	复位值
EINTMASK	0x560000A4	R/W	外部中断屏蔽寄存器	0x000FFFFF

EINTMASK	位	描述	初始状态
EINT23	[23]	0 = 使能中断 1 = 禁止中断	0
EINT22	[22]	0 = 使能中断 1 = 禁止中断	0
EINT21	[21]	0 = 使能中断 1 = 禁止中断	0
EINT20	[20]	0 = 使能中断 1 = 禁止中断	0
EINT19	[19]	0 = 使能中断 1 = 禁止中断	1
EINT18	[18]	0 = 使能中断 1 = 禁止中断	1
EINT17	[17]	0 = 使能中断 1 = 禁止中断	1
EINT16	[16]	0 = 使能中断 1 = 禁止中断	1
EINT15	[15]	0 = 使能中断 1 = 禁止中断	1
EINT14	[14]	0 = 使能中断 1 = 禁止中断	1
EINT13	[13]	0 = 使能中断 1 = 禁止中断	1
EINT12	[12]	0 = 使能中断 1 = 禁止中断	1
EINT11	[11]	0 = 使能中断 1 = 禁止中断	1
EINT10	[10]	0 = 使能中断 1 = 禁止中断	1
EINT9	[9]	0 = 使能中断 1 = 禁止中断	1
EINT8	[8]	0 = 使能中断 1 = 禁止中断	1
EINT7	[7]	0 = 使能中断 1 = 禁止中断	1
EINT6	[6]	0 = 使能中断 1 = 禁止中断	1
EINT5	[5]	0 = 使能中断 1 = 禁止中断	1
EINT4	[4]	0 = 使能中断 1 = 禁止中断	1
保留	[3:0]	保留	1111

EINTPEND (外部中断挂起寄存器)

寄存器	地址	R/W	描述	复位值
EINTPEND	0x560000A8	R/W	外部中断挂起寄存器	0x00

EINTPEND	位	描述	初始状态
EINT23	[23]	写“1”清零此位 0 = 不发生 1 = 发生中断	0
EINT22	[22]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT21	[21]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT20	[20]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT19	[19]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT18	[18]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT17	[17]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT16	[16]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT15	[15]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT14	[14]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT13	[13]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT12	[12]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT11	[11]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT10	[10]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT9	[9]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT8	[8]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT7	[7]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT6	[6]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT5	[5]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
EINT4	[4]	写“1”清零此位 0 = 使能中断 1 = 禁止中断	0
保留	[3:0]	保留	0000

GSTATUSn (通用状态寄存器)

寄存器	地址	R/W	描述	复位值
GSTATUS0	0x560000AC	R	外部引脚状态	-
GSTATUS1	0x560000B0	R/W	芯片序列号	0x32440001
GSTATUS2	0x560000B4	R/W	复位状态	0x1
GSTATUS3	0x560000B8	R/W	信息寄存器	0x0
GSTATUS4	0x560000BC	R/W	信息寄存器	0x0

GSTATUS0	位	描述	初始状态
nWAIT	[3]	nWAIT 引脚的状态	-
NCON	[2]	NCON 引脚的状态	-
RnB	[1]	RnB 引脚的状态	-
BATT_FLT	[0]	BATT_FLT 引脚的状态	-

GSTATUS1	位	描述	初始状态
CHIP ID	[31:0]	ID 寄存器 = 0x32440001	0x32440001

GSTATUS2	位	描述	初始状态
保留	[3]	保留	0
WDTRST	[2]	看门狗复位引起引导启动。写“1”清零。	0
SLEEP_RST	[1]	睡眠模式中唤醒复位引起引导启动。写“1”清零。	0
PWRST	[0]	上电复位引起引导启动。写“1”清零。	1

GSTATUS3	位	描述	初始状态
inform	[31:0]	信息寄存器。此寄存器由上电复位清零。不然则保护数据值。	0x0

GSTATUS4	位	描述	初始状态
inform	[31:0]	信息寄存器。此寄存器由上电复位清零。不然则保护数据值。	0x0

DSCn (驱动强度控制寄存器)

寄存器	地址	R/W	描述	复位值
DSC0	0x560000C4	R/W	强度控制寄存器 0	0x0
DSC1	0x560000C8	R/W	强度控制寄存器 1	0x0

DSC0	位	描述				初始状态
nEN_DSC	[31]	使能驱动强度控制 0 : 使能 1 : 禁止				0
保留	[30:10]	-				0
DSC_ADR	[9:8]	地址总线驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_DATA3	[7:6]	DATA[31:24] I/O 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_DATA2	[5:4]	DATA[23:16] I/O 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_DATA1	[3:2]	DATA[15:8] I/O 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_DATA0	[1:0]	DATA[7:0] I/O 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00

DSC1	位	描述				初始状态
DSC_SCK1	[29:28]	SCLK1 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_SCK0	[27:26]	SCLK0 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_SCKE	[25:24]	SCKE 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_SDR	[23:22]	nSRAS/nSCAS 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_NFC	[21:20]	Nand Flash 控制驱动强度 (nFCE , nFRE , nFWE , CLE , ALE) 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_BE	[19:18]	nBE[3:0]驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_WOE	[17:16]	nWE/nOE 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS7	[15:14]	nGCS7 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS6	[13:12]	nGCS6 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS5	[11:10]	nGCS5 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS4	[9:8]	nGCS4 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS3	[7:6]	nGCS3 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS2	[5:4]	nGCS2 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS1	[3:2]	nGCS1 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00
DSC_CS0	[1:0]	nGCS0 驱动强度 00: 12mA 10: 10mA 01: 8mA 11: 6mA				00

MSLCON (存储器睡眠控制寄存器)

选择睡眠模式中的存储器接口状态。

寄存器	地址	R/W	描述	复位值
MSLCON	0x560000CC	R/W	存储器睡眠控制寄存器	0x0

DSC0	位	描述	初始状态
PSC_DATA	[11]	睡眠模式中 DATA[31:0]引脚状态 0 : 高阻 1 : 输出 "0"	0
PSC_WAIT	[10]	睡眠模式中 nWAIT 引脚状态 0 : 输入 1 : 输出 "0"	0
PSC_RnB	[9]	睡眠模式中 RnB 引脚状态 0 : 输入 1 : 输出 "0"	0
PSC_NF	[8]	睡眠模式中 NAND Flash I/F 引脚状态 (nFCE , nFRE , nFWE , ALE , CLE) 0 : 非活动 (nFCE , nFRE , nFWE , ALE , CLE = 11100) 1 : 高阻	0
PSC_SDR	[7]	睡眠模式中 nSRAS , nSCAS 引脚状态 0 : 非活动 ("1") 1 : 高阻	0
PSC_DQM	[6]	睡眠模式中 DQM[3:0]/nWE[3:0]引脚状态 0 : 非活动 ("1") 1 : 高阻	0
PSC_OE	[5]	睡眠模式中 nOE 引脚状态 0 : 非活动 ("1") 1 : 高阻	0
PSC_WE	[4]	睡眠模式中 nWE 引脚状态 0 : 非活动 ("1") 1 : 高阻	0
PSC_GCS0	[3]	睡眠模式中 nGCS[0]引脚状态 0 : 非活动 ("1") 1 : 高阻	0
PSC_GCS1	[2]	睡眠模式中 nGCS[5:1]引脚状态 0 : 非活动 ("1") 1 : 输出 "0"	0
PSC_GCS6	[1]	睡眠模式中 nGCS[6]引脚状态 0 : 非活动 ("1") 1 : 输出 "0"	0
PSC_GCS7	[0]	睡眠模式中 nGCS[7]引脚状态 0 : 非活动 ("1") 1 : 输出 "0"	0

10 PWM 定时器

概述

S3C2440A 有 5 个 16 位定时器。其中定时器 0、1、2 和 3 具有脉宽调制 (**PWM**) 功能。定时器 4 是一个无输出引脚的内部定时器。定时器 0 还包含用于大电流驱动的死区发生器。

定时器 0 和 1 共用一个 8 位预分频器，定时器 2、3 和 4 共用另外的 8 位预分频器。每个定时器都有一个可以生成 5 种不同分频信号 ($1/2$, $1/4$, $1/8$, $1/16$ 和 $TCLK$) 的时钟分频器。每个定时器模块从相应 8 位预分频器得到时钟的时钟分频器中得到其自己的时钟信号。8 位预分频器是可编程的，并且按存储在 $TCFG0$ 和 $TCFG1$ 寄存器中的加载值来分频 $PCLK$ 。

定时计数缓冲寄存器 ($TCNTBn$) 包含了一个当使能了定时器时被加载到递减计数器中的初始值。定时比较缓冲寄存器 ($TCMPBn$) 包含了一个被加载到比较寄存器中的与递减计数器相比较的初始值。这种 $TCNTBn$ 和 $TCMPBn$ 的双缓冲特征保证了改变频率和占空比时定时器产生稳定的输出。

每个定时器有它自己的由定时器时钟驱动的 16 位递减计数器。当递减计数器到达零时，产生定时器中断请求通知 CPU 定时器操作已经完成。当定时器计数器到达零时，相应的 $TCNTBn$ 的值将自动被加载到递减计数器以继续下一次操作。然而，如果定时器停止了，例如，在定时器运行模式期间清除 $TCONn$ 的定时器使能位， $TCNTBn$ 的值将不会被重新加载到计数器中。

$TCMPBn$ 的值是用于脉宽调制 (**PWM**)。当递减计数器的值与定时器控制逻辑中的比较寄存器的值相匹配时定时器控制逻辑改变输出电平。因此，比较寄存器决定 PWM 输出的开启时间 (或关闭时间)。

特性

- 五个 16 位定时器
- 两个 8 位预分频器和两个 4 位分频器
- 可编程输出波形的占空比控制 (**PWM**)
- 自动重载模式或单稳脉冲模式
- 死区发生

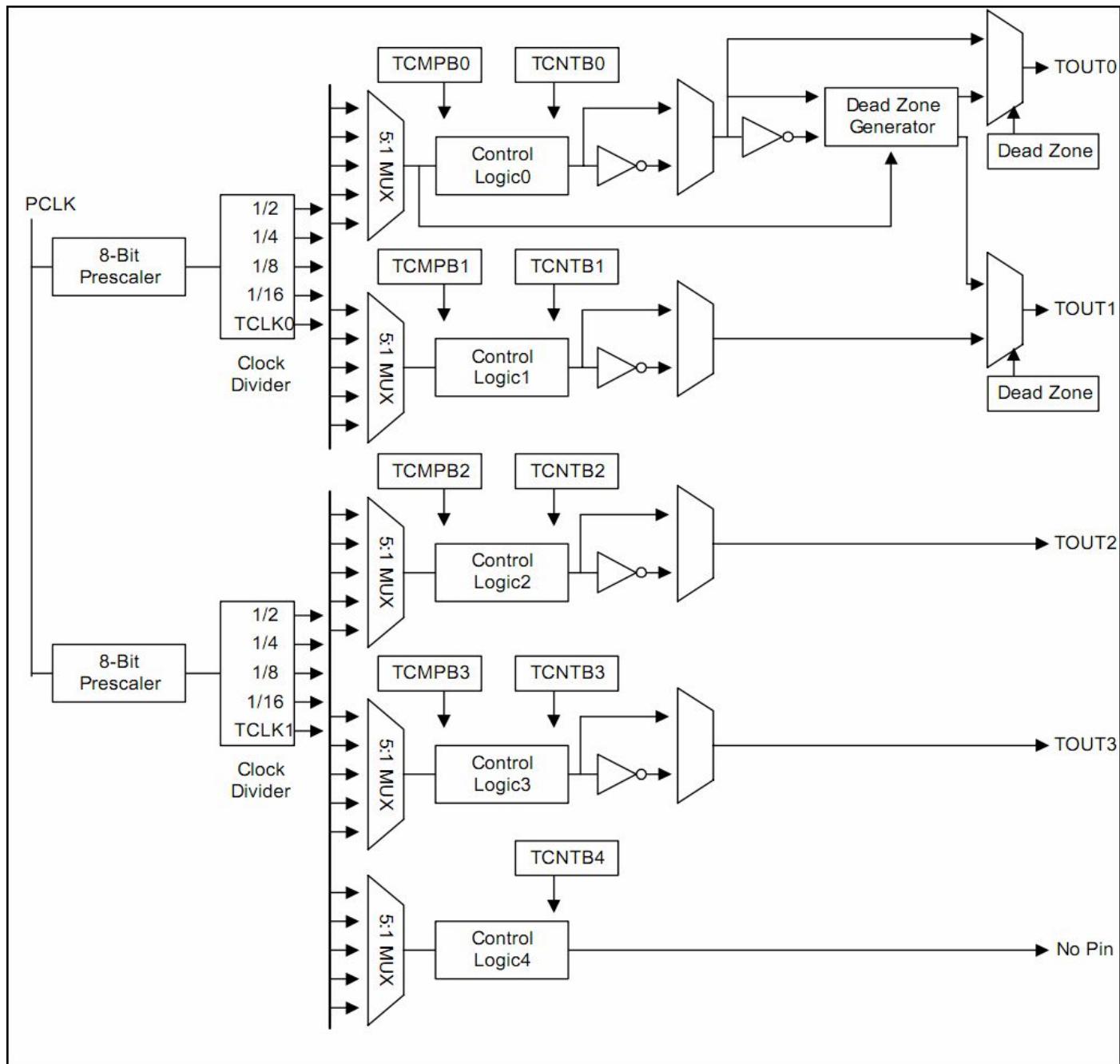


图 10-1. 16 位 PWM 定时器方框图

PWM 定时器操作

预分频器和分频器

一个 8 位预分频器和一个 4 位分频器产生出以下输出频率：

4 位分频器设置	最小分辨率 (预分频器=0)	最大分辨率 (预分频器=255)	最大间隔时间 (TCNTBn=65535)
1/2 (PCLK = 50 MHz)	0.0400 μ s (25.0000 MHz)	10.2400 μ s (97.6562 KHz)	0.6710 秒
1/4 (PCLK = 50 MHz)	0.0800 μ s (25.0000 MHz)	20.4800 μ s (97.6562 KHz)	1.3421 秒
1/8 (PCLK = 50 MHz)	0.1600 μ s (25.0000 MHz)	40.9601 μ s (97.6562 KHz)	2.6843 秒
1/16 (PCLK = 50 MHz)	0.3200 μ s (25.0000 MHz)	81.9188 μ s (97.6562 KHz)	5.3686 秒

基本时序操作

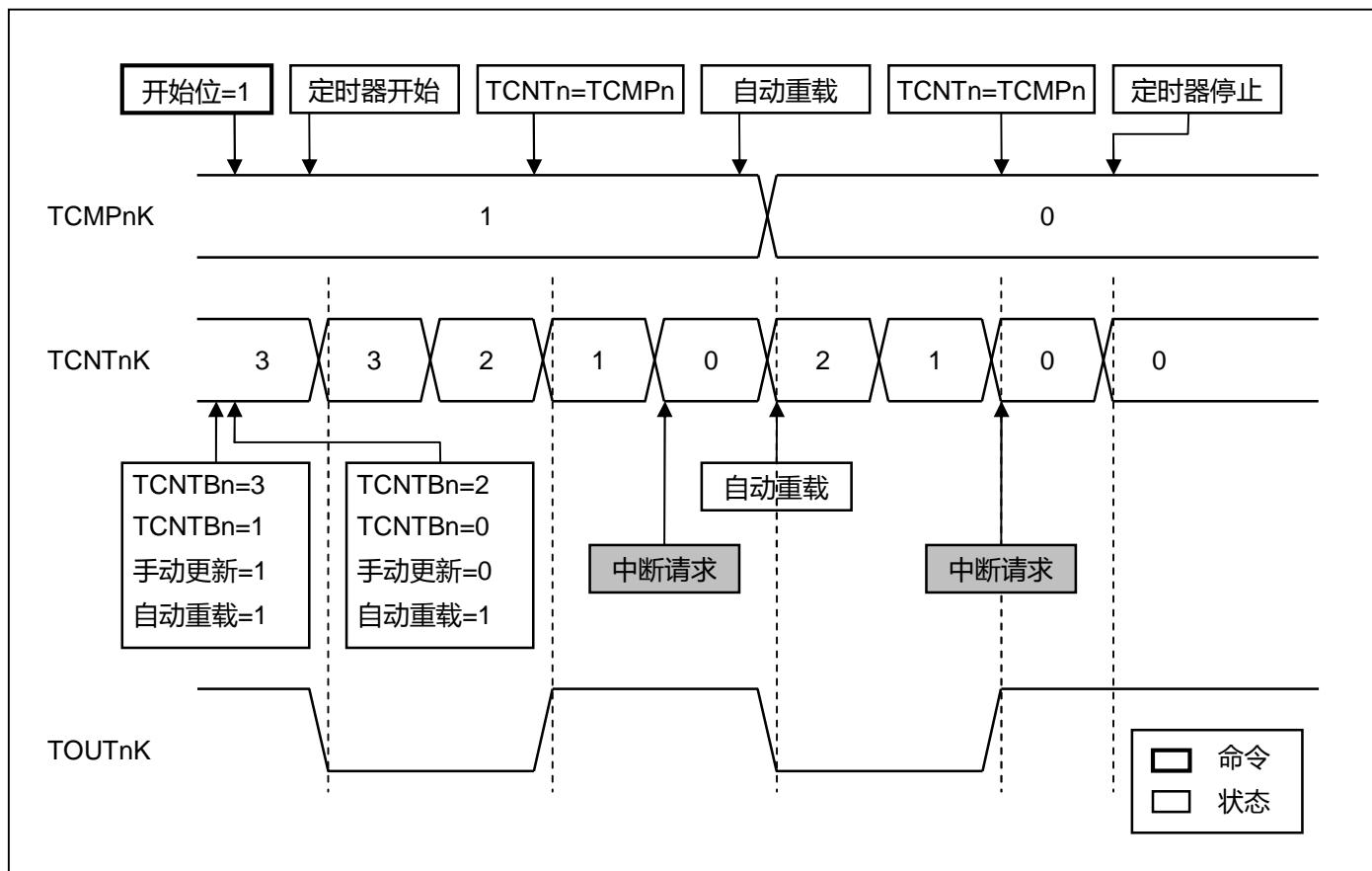


图 10-2. 定时器操作

一个定时器（除了定时器通道 5）包含 TCNTBn , TCNTn , TCMPBn 和 TCMPn。 (TCNTn 和 TCMPn 是内部寄存器的名称。从 TCNTOn 寄存器中可以读取 TCNTn)当定时器到达 0 时 TCNTBn 和 TCMPBn 被加载到 TCNTn 和 TCMPn 中。当 TCNTn 到达 0 时，如果中断为使能则将发生一个中断请求。

自动重载和双缓冲

S3C2440A PWM 定时器包含双缓冲功能，允许在不停止当前定时器操作的情况下为下次定时器操作改变重载值。所以即使设置了新的定时器值，当前定时器操作仍然顺利的被完成。

定时器值可以被写入到定时器计数缓冲寄存器 (TCNTBn) 中并且可以从定时器计数监视寄存器 (TCNTOn) 中读取当前定时器的计数值。如果读取 TCNTBn，读出的值不是指示当前计数器的状态而是下次定时器持续时间的重载值。

自动重载操作在 TCNTn 到达 0 时复制 TCNTBn 到 TCNTn。写入到 TCNTBn 的该值，只有在 TCNTn 到达 0 并且使能了自动重载时才被加载到 TCNTn。如果 TCNTn 变为 0 并且自动重载位为 0，TCNTn 不会进一步任何操作。

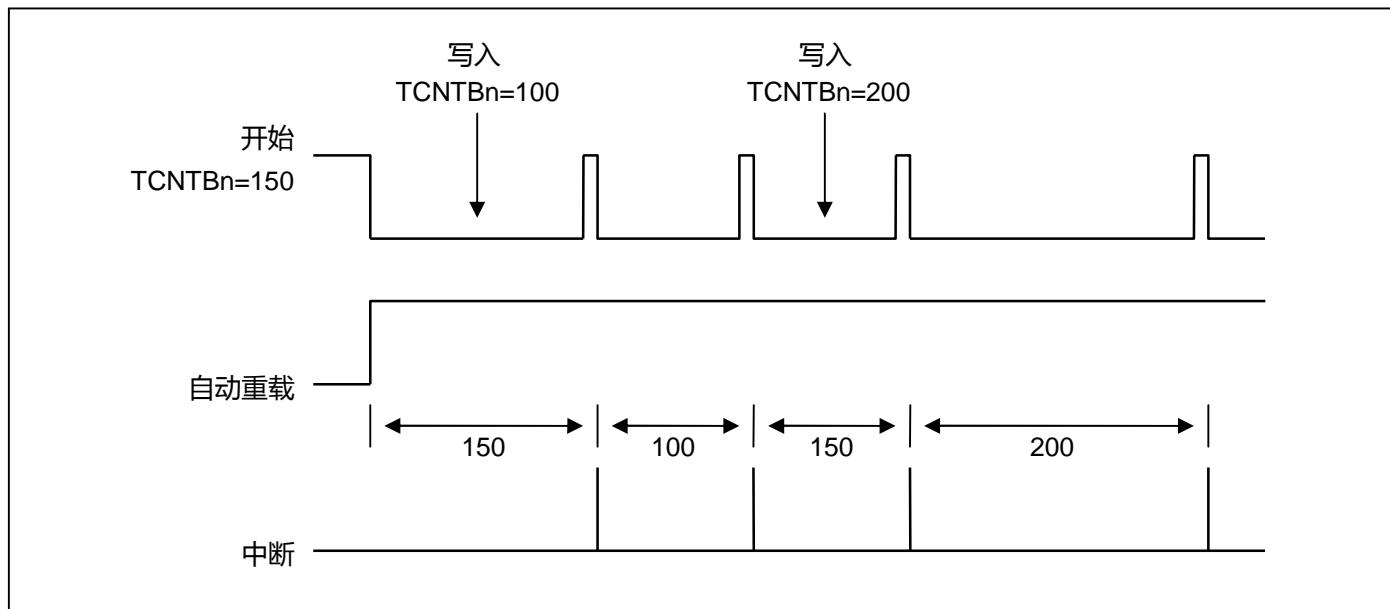


图 10-3. 双缓冲功能的例子

使用手动更新位和变相位初始化定时器

当递减计数器到达 0 时发生定时器的自动重载操作。所以必须预先由用户定义一个 TCNTn 的起始值。在这种情况下，必须通过手动更新位加载起始值。以下步骤描述了如何启动一个定时器：

- 1) 初始值写入到 TCNTBn 和 TCMPBn 中。
- 2) 设置相应定时器的手动更新位。推荐你配制变相开/关位。(无论是否使用变换极性)
- 3) 设置相应定时器的开始位来启动定时器(并且清除手动更新位)。

如果定时器被强制停止，TCNTn 保持计数器值并且不会从 TCNTBn 重载。如果需要设置一个新值，执行手动更新。

注意：

无论何时 TOUT 变相开/关位改变，是否定时器运行中都将改变 TOUT 逻辑值。因此，最好带手动更新位配制变相开/关位。

定时器操作

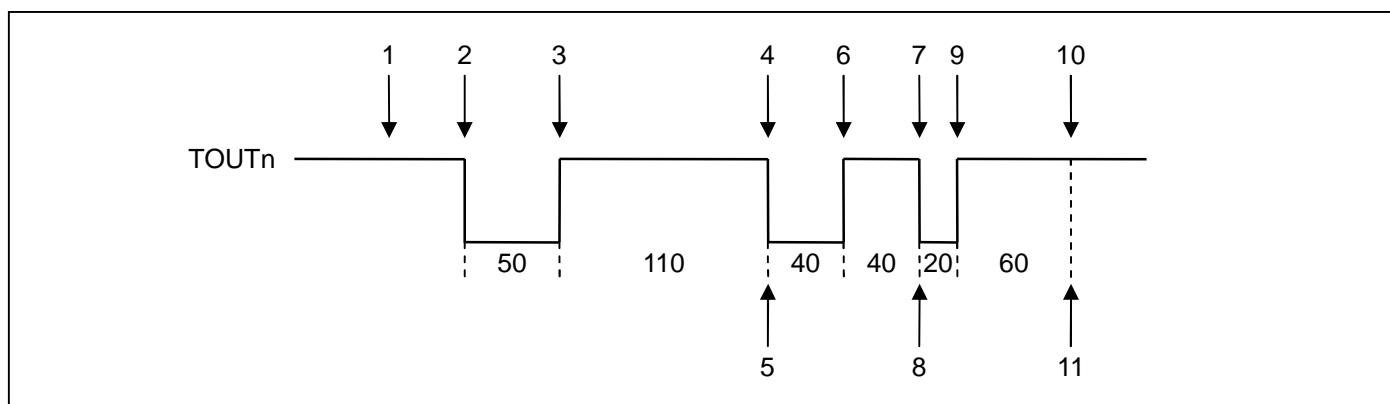


图 10-4. 定时器操作例子

上述图 10-4 显示了以下过程的结果：

1. 使能自动重载功能。设置 $TCNTBn$ 为 160 (50+110) 并且设置 $TCMPBn$ 为 110。置位手动更新位并且配制变相位(开/关)。手动更新位分别设置 $TCNTn$ 和 $TCMPn$ 到 $TCNTBn$ 和 $TCMPBn$ 的值中。然后分别设置 $TCNTBn$ 和 $TCMPBn$ 为 80 (40+40) 和 40，以决定下次重载值。
2. 设置启动位，预设手动更新位为 0，变相位为关，自动重载位为开。定时器在定时器分辨率内的等待时间后启动递减计数。
3. 当 $TCNTn$ 与 $TCMPn$ 的值相同时， $TOUTn$ 的逻辑电平从低电平变为高电平。
4. 当 $TCNTn$ 到达 0 时，发出中断请求并且 $TCNTBn$ 的值加载到暂存器中。在下一个定时器标记时刻，重载 $TCNTn$ 为暂存器 ($TCNTBn$) 的值。
5. 中断服务程序 (ISR) 中，为下一个持续时间分别设置 $TCNTBn$ 和 $TCMPBn$ 为 80 (20+60) 和 60。
6. 当 $TCNTn$ 与 $TCMPn$ 的值相同时， $TOUTn$ 的逻辑电平从低电平变为高电平。
7. 当 $TCNTn$ 到达 0 时，触发一个中断自动重载 $TCNTn$ 为 $TCNTBn$ 的值。
8. 中断服务程序 (ISR) 中，禁止自动重载和中断请求以停止定时器。
9. 当 $TCNTn$ 与 $TCMPn$ 的值相同时， $TOUTn$ 的逻辑电平从低电平变为高电平。
10. 尽管 $TCNTn$ 到达 0，但因为禁止了自动重载，所以 $TCNTn$ 并不会再次重载并且定时器已经停止了。
11. 不再产生中断请求。

脉宽调制 (PWM)

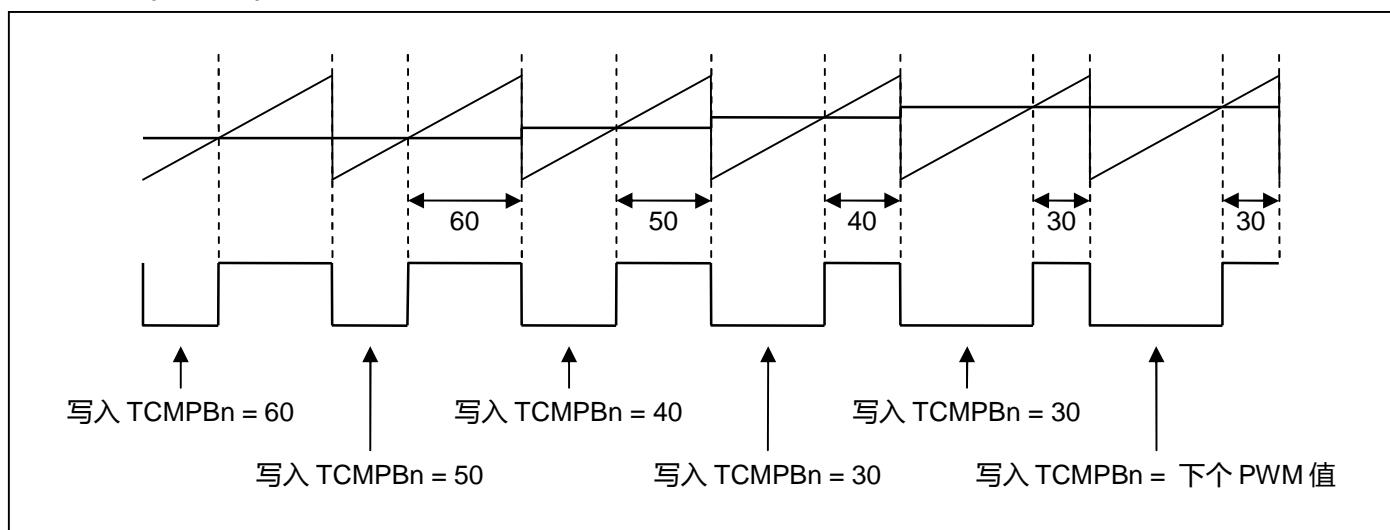


图 10-5. PWM 的例子

PWM 功能可以通过使用 TCMPBn 实现。PWM 频率由 TCNTBn 决定。图 10-5 显示了一个由 TCMPBn 决定的 PWM 值。

减小 TCMPBn 可以提高 PWM 值。增大 TCMPBn 可以降低 PWM 值。如果使能了输出变相器，则增/减颠倒。双缓冲功能允许为下一个 PWM 周期而由 ISR 或其它程序在当前 PWM 周期的任何点写入 TCMPBn 。

输出电平控制

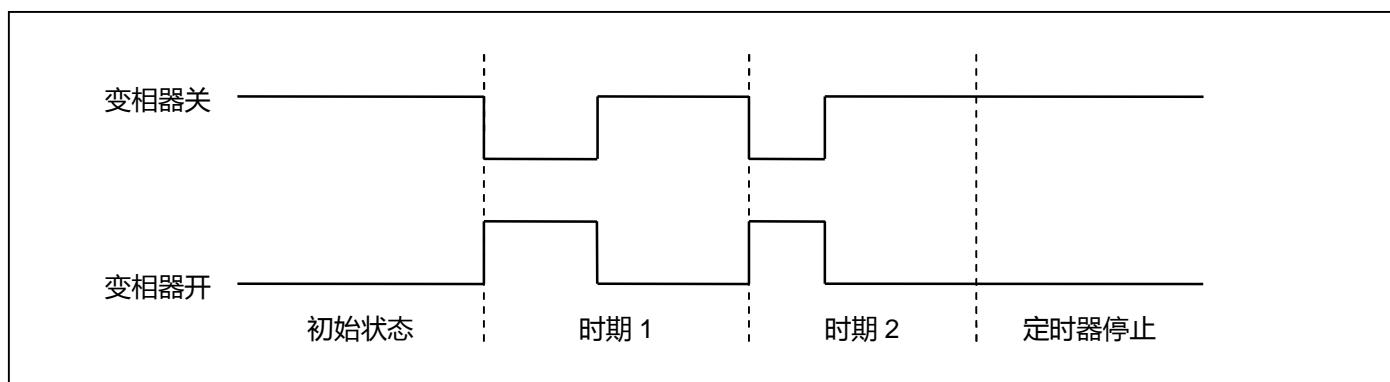


图 10-6. 变相器开/关

以下步骤描述了如何维持 TOUT 为高电平或低电平 (假设变相器为关)：

1. 关闭自动重载位。然后 TOUTn 变为高电平并且定时器在 TCNTn 到达 0 后停止 (推荐)。
2. 清除定时器启动/停止位为 0 来停止定时器。如果 $\text{TCNTn} \leq \text{TCMPn}$ 则输出电平为高电平。如果 $\text{TCNTn} > \text{TCMPn}$ ，则输出电平为低电平。
3. TOUTn 可以由 TCON 中的变相开/关位变换极性。变相器移除附加电路以适应输出电平。

死区发生器

死区 (Dead Zone) 是用于功率器件中的 PWM 控制。此功能允许在开关器件关闭与另一个开关器件的开启之间插入一个时间间隙。这个时间间隙禁止同时开启两个开关器件，即使是在非常短的时间。

TOUT0 是 PWM 的输出。nTOUT0 是 TOUT0 的倒置。如果使能了死区，TOUT0 和 nTOUT0 的输出波形将分别为 TOUT0_DZ 和 nTOUT0_DZ。nTOUT0_DZ 连接到 TOUT1 引脚。

在死区间隙中，永远不可能同时开启 TOUT0_DZ 和 nTOUT0_DZ。

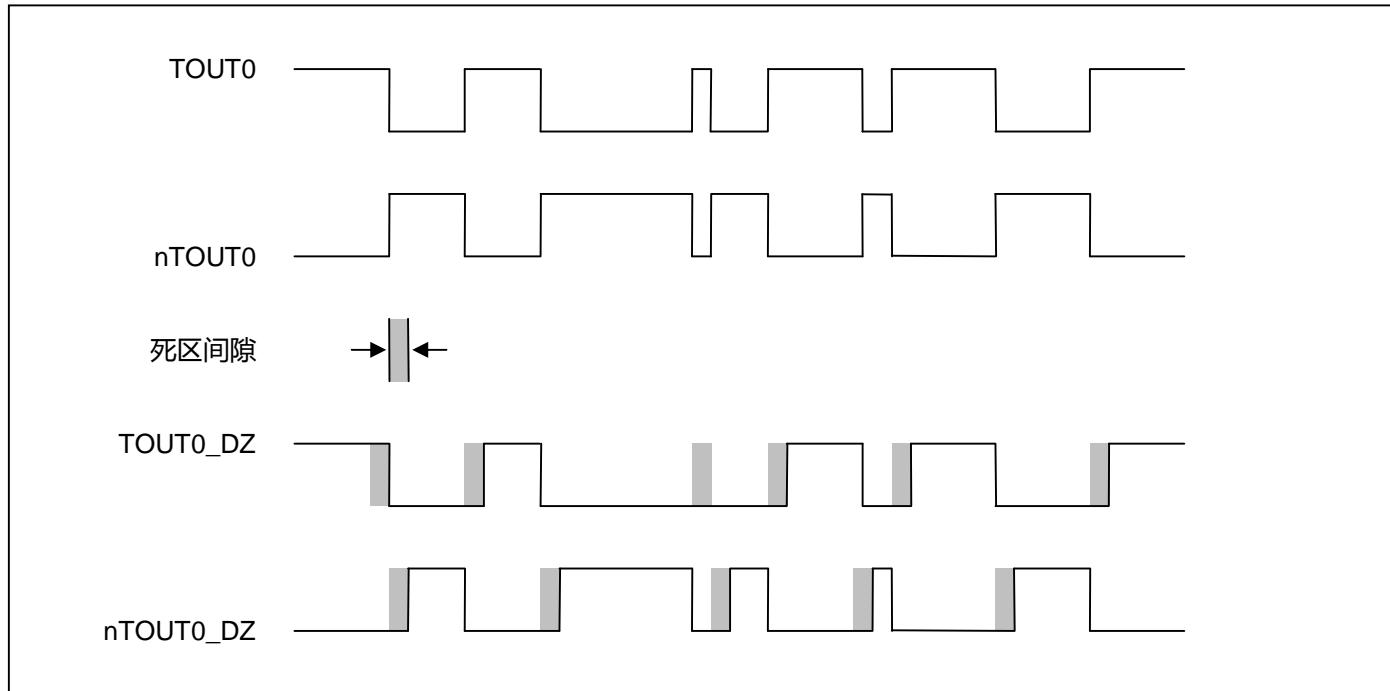


图 10-7. 使能死区功能时的波形

DMA 请求模式

PWM 定时器可以在每个特定时间产生 DMA 请求。定时器保持 DMA 请求信号 (nDMA_REQ) 为低电平直到定时器收到一个 ACK 信号。当定时器收到 ACK 信号则使请求信号暂停。通过设置 DMA 模式位 (在 TCFG1 寄存器中) 决定产生 DMA 请求的定时器。如果定时器中的一个被配制为 DMA 请求模式，则该定时器将不产生中断请求。其它的定时器将正常产生中断。

DMA 模式配制和 DMA/中断操作

DMA 模式	DMA 请求	定时器 0 中断	定时器 1 中断	定时器 2 中断	定时器 3 中断	定时器 4 中断
0000	未选择	开	开	开	开	开
0001	定时器 0	关	开	开	开	开
0010	定时器 1	开	关	开	开	开
0011	定时器 2	开	开	关	开	开
0100	定时器 3	开	开	开	关	开
0101	定时器 4	开	开	开	开	关
0110	未选择	开	开	开	开	开

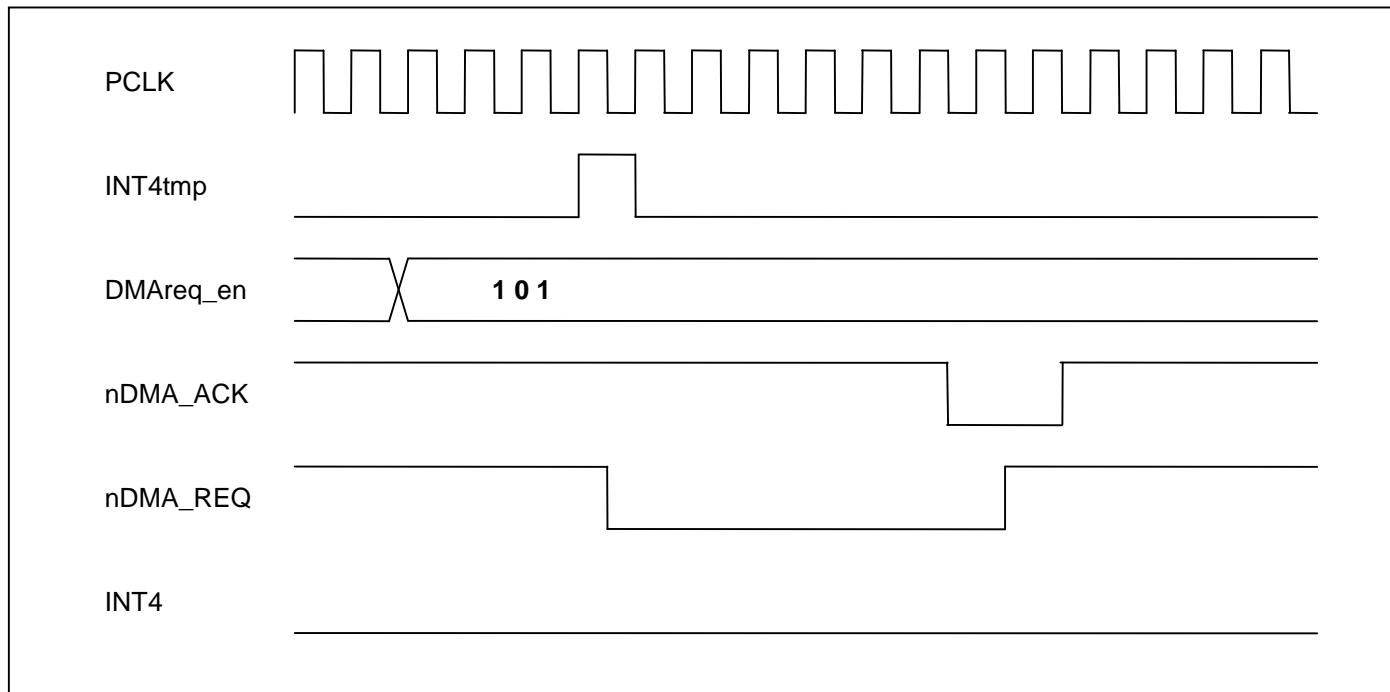


图 10-8. 定时器 4 DMA 模式操作

PWM 定时器控制寄存器

定时器配制寄存器 0 (TCFG0)

定时器输入时钟频率 = PCLK / {预分频值+1} / {分频值}

{预分频值} = 0~255

{分频值} = 2, 4, 8, 16

寄存器	地址	R/W	描述	复位值
TCFG0	0x51000000	R/W	配制两个 8 位预分频器	0x00000000

TCFG0	位	描述	初始状态
保留	[31:24]		0x00
死区长度	[23:16]	该 8 位决定了死区段。死区段持续为 1 的时间等于定时器 0 持续为 1 的时间。	0x00
Prescaler 1	[15:8]	该 8 位决定了定时器 2, 3 和 4 的预分频值	0x00
Prescaler 0	[7:0]	该 8 位决定了定时器 0 和 1 的预分频值	0x00

定时器配制寄存器 1 (TCFG1)

寄存器	地址	R/W	描述	复位值
TCFG1	0x51000004	R/W	5 路多路选择器和 DMA 模式选择寄存器	0x00000000

TCFG1	位	描述	初始状态
保留	[31:24]		00000000
DMA 模式	[23:20]	选择 DMA 请求通道 0000 = 未选择 (所有中断) 0001 = 定时器 0 0010 = 定时器 1 0011 = 定时器 2 0100 = 定时器 3 0101 = 定时器 4 0110 = 保留	0000
MUX 4	[19:16]	选择 PWM 定时器 4 的选通输入 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 外部 TCLK1	0000
MUX 3	[15:12]	选择 PWM 定时器 3 的选通输入 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 外部 TCLK1	0000
MUX 2	[11:8]	选择 PWM 定时器 2 的选通输入 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 外部 TCLK1	0000
MUX 1	[7:4]	选择 PWM 定时器 1 的选通输入 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 外部 TCLK1	0000
MUX 0	[3:0]	选择 PWM 定时器 0 的选通输入 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 外部 TCLK1	0000

定时器控制寄存器 1 (TCON)

寄存器	地址	R/W	描述	复位值
TCON	0x51000008	R/W	定时器控制寄存器	0x00000000

TCON	位	描述	初始状态
定时器 4 自动重载开/关	[22]	决定定时器 4 的自动重载开启或关闭 0 = 单稳态 1 = 间隙模式 (自动重载)	0
定时器 4 手动更新 (注释)	[21]	决定定时器 4 的手动更新 0 = 无操作 1 = 更新 TCNTB4	0
定时器 4 启动/停止	[20]	决定定时器 4 的启动或停止 0 = 停止定时器 4 1 = 启动定时器 4	0
定时器 3 自动重载开/关	[19]	决定定时器 3 的自动重载开启或关闭 0 = 单稳态 1 = 间隙模式 (自动重载)	0
定时器 3 输出变相开/关	[18]	决定定时器 3 的输出变相开启或关闭 0 = 关闭变相 1 = TOUT3 变换极性	0
定时器 3 手动更新 (注释)	[17]	决定定时器 3 的手动更新 0 = 无操作 1 = 更新 TCNTB3 和 TCMPB3	0
定时器 3 启动/停止	[16]	决定定时器 3 的启动或停止 0 = 停止定时器 3 1 = 启动定时器 3	0
定时器 2 自动重载开/关	[15]	决定定时器 2 的自动重载开启或关闭 0 = 单稳态 1 = 间隙模式 (自动重载)	0
定时器 2 输出变相开/关	[14]	决定定时器 2 的输出变相开启或关闭 0 = 关闭变相 1 = TOUT2 变换极性	0
定时器 2 手动更新 (注释)	[13]	决定定时器 2 的手动更新 0 = 无操作 1 = 更新 TCNTB2 和 TCMPB2	0
定时器 2 启动/停止	[12]	决定定时器 2 的启动或停止 0 = 停止定时器 2 1 = 启动定时器 2	0
定时器 1 自动重载开/关	[11]	决定定时器 1 的自动重载开启或关闭 0 = 单稳态 1 = 间隙模式 (自动重载)	0
定时器 1 输出变相开/关	[10]	决定定时器 1 的输出变相开启或关闭 0 = 关闭变相 1 = TOUT1 变换极性	0
定时器 1 手动更新 (注释)	[9]	决定定时器 1 的手动更新 0 = 无操作 1 = 更新 TCNTB1 和 TCMPB1	0
定时器 1 启动/停止	[8]	决定定时器 1 的启动或停止 0 = 停止定时器 1 1 = 启动定时器 1	0

定时器控制寄存器 1 (TCON) (续)

保留	[7:5]	保留	0
死区使能	[4]	决定死区操作 0 = 禁止 1 = 使能	0
定时器 0 自动重载开/关	[3]	决定定时器 0 的自动重载开启或关闭 0 = 单稳态 1 = 间隙模式 (自动重载)	0
定时器 0 输出变相开/关	[2]	决定定时器 0 的输出变相开启或关闭 0 = 关闭变相 1 = TOUT0 变换极性	0
定时器 0 手动更新 (注释)	[1]	决定定时器 0 的手动更新 0 = 无操作 1 = 更新 TCNTB0 和 TCMPB0	0
定时器 0 启动/停止	[0]	决定定时器 0 的启动或停止 0 = 停止定时器 0 1 = 启动定时器 0	0

注释：此位必须在下次写操作时清零。

定时器 0 计数缓冲寄存器和比较缓冲寄存器 (TCNTB0/TCMPB0)

寄存器	地址	R/W	描述	复位值
TCNTB0	0x5100000C	R/W	定时器 0 计数缓冲寄存器	0x00000000
TCMPB0	0x51000010	R/W	定时器 0 比较缓冲寄存器	0x00000000

TCNTB0	位	描述	初始状态
定时器 0 计数缓冲寄存器	[15:0]	设置定时器 0 的计数缓冲器的值	0x00000000

TCMPB0	位	描述	初始状态
定时器 0 比较缓冲寄存器	[15:0]	设置定时器 0 的比较缓冲器的值	0x00000000

定时器 0 计数监视寄存器 (TCNTO0)

寄存器	地址	R/W	描述	复位值
TCNTO0	0x51000014	R	定时器 0 计数监视寄存器	0x00000000

TCNTO0	位	描述	初始状态
定时器 0 计数监视寄存器	[15:0]	定时器 0 的计数监视值	0x00000000

定时器 1 计数缓冲寄存器和比较缓冲寄存器 (TCNTB1/TCMPB1)

寄存器	地址	R/W	描述	复位值
TCNTB1	0x51000018	R/W	定时器 1 计数缓冲寄存器	0x00000000
TCMPB1	0x5100001C	R/W	定时器 1 比较缓冲寄存器	0x00000000

TCNTB1	位	描述	初始状态
定时器 1 计数缓冲寄存器	[15:0]	设置定时器 1 的计数缓冲器的值	0x00000000

TCMPB1	位	描述	初始状态
定时器 1 比较缓冲寄存器	[15:0]	设置定时器 1 的比较缓冲器的值	0x00000000

定时器 1 计数监视寄存器 (TCNTO1)

寄存器	地址	R/W	描述	复位值
TCNTO1	0x51000020	R	定时器 1 计数监视寄存器	0x00000000

TCNTO1	位	描述	初始状态
定时器 1 计数监视寄存器	[15:0]	定时器 1 的计数监视值	0x00000000

定时器 2 计数缓冲寄存器和比较缓冲寄存器 (TCNTB2/TCMPB2)

寄存器	地址	R/W	描述	复位值
TCNTB2	0x51000024	R/W	定时器 2 计数缓冲寄存器	0x00000000
TCMPB2	0x51000028	R/W	定时器 2 比较缓冲寄存器	0x00000000

TCNTB2	位	描述	初始状态
定时器 2 计数缓冲寄存器	[15:0]	设置定时器 2 的计数缓冲器的值	0x00000000

TCMPB2	位	描述	初始状态
定时器 2 比较缓冲寄存器	[15:0]	设置定时器 2 的比较缓冲器的值	0x00000000

定时器 2 计数监视寄存器 (TCNTO2)

寄存器	地址	R/W	描述	复位值
TCNTO2	0x5100002C	R	定时器 2 计数监视寄存器	0x00000000

TCNTO2	位	描述	初始状态
定时器 2 计数监视寄存器	[15:0]	定时器 2 的计数监视值	0x00000000

定时器 3 计数缓冲寄存器和比较缓冲寄存器 (TCNTB3/TCMPB3)

寄存器	地址	R/W	描述	复位值
TCNTB3	0x51000030	R/W	定时器 3 计数缓冲寄存器	0x00000000
TCMPB3	0x51000034	R/W	定时器 3 比较缓冲寄存器	0x00000000

TCNTB3	位	描述	初始状态
定时器 3 计数缓冲寄存器	[15:0]	设置定时器 3 的计数缓冲器的值	0x00000000

TCMPB3	位	描述	初始状态
定时器 3 比较缓冲寄存器	[15:0]	设置定时器 3 的比较缓冲器的值	0x00000000

定时器 3 计数监视寄存器 (TCNTO3)

寄存器	地址	R/W	描述	复位值
TCNTO3	0x51000038	R	定时器 3 计数监视寄存器	0x00000000

TCNTO3	位	描述	初始状态
定时器 3 计数监视寄存器	[15:0]	定时器 3 的计数监视值	0x00000000

定时器 4 计数缓冲寄存器 (TCNTB4)

寄存器	地址	R/W	描述	复位值
TCNTB4	0x5100003C	R/W	定时器 4 计数缓冲寄存器	0x00000000

TCNTB4	位	描述	初始状态
定时器 4 计数缓冲寄存器	[15:0]	设置定时器 4 的计数缓冲器的值	0x00000000

定时器 4 计数监视寄存器 (TCNTO4)

寄存器	地址	R/W	描述	复位值
TCNTO4	0x51000040	R	定时器 4 计数监视寄存器	0x00000000

TCNTO4	位	描述	初始状态
定时器 4 计数监视寄存器	[15:0]	定时器 4 的计数监视值	0x00000000

11 UART

概述

S3C2440A 的通用异步收发器 (**UART**) 配有 3 个独立异步串行 I/O (SIO) 端口，每个都可以是基于中断或基于 DMA 模式的操作。换句话说，UART 可以通过产生中断或 DMA 请求来进行 CPU 和 UART 之间的数据传输。UART 通过使用系统时钟可以支持最高 115.2Kbps 的比特率。如果是外部器件提供 UEXTCLK 的 UART，则 UART 可以运行在更高的速度。每个 UART 通道包含两个的 64 字节的 FIFO 给发送和接收。

S3C2440A 的 UART 包括了可编程波特率，红外 (IR) 发送/接收，插入 1 个或 2 个停止位，5 位、6 位、7 位或 8 位的数据宽度以及奇偶校验。

每个 UART 包含一个波特率发生器、发送器、接收器和一个控制单元，如图 11-1 所示。波特率发生器可以由 PCLK、FCLK/n 或 UEXTCLK (外部输入时钟) 时钟驱动。发送器和接收器包含了 64 字节 FIFO 和数据移位器。将数据写入到 FIFO 接着在发送前复制到发送移位器中。随后将在发送数据引脚 (TxDn) 移出数据。与此同时从接收数据引脚 (RxDn) 移入收到的数据，接着从移位器复制到 FIFO。

特性

- 基于 DMA 或基于中断操作的 RxD0 , TxD0 , RxD1 , TxD1 , RxD2 和 TxD2
- UART 通道 0 , 1 和 2 带 IrDA 1.0 和 64 字节 FIFO
- UART 通道 0 和 1 带 nRTS0 , nCTS0 , nRTS1 和 nCTS1
- 支持握手发送/接收

方框图

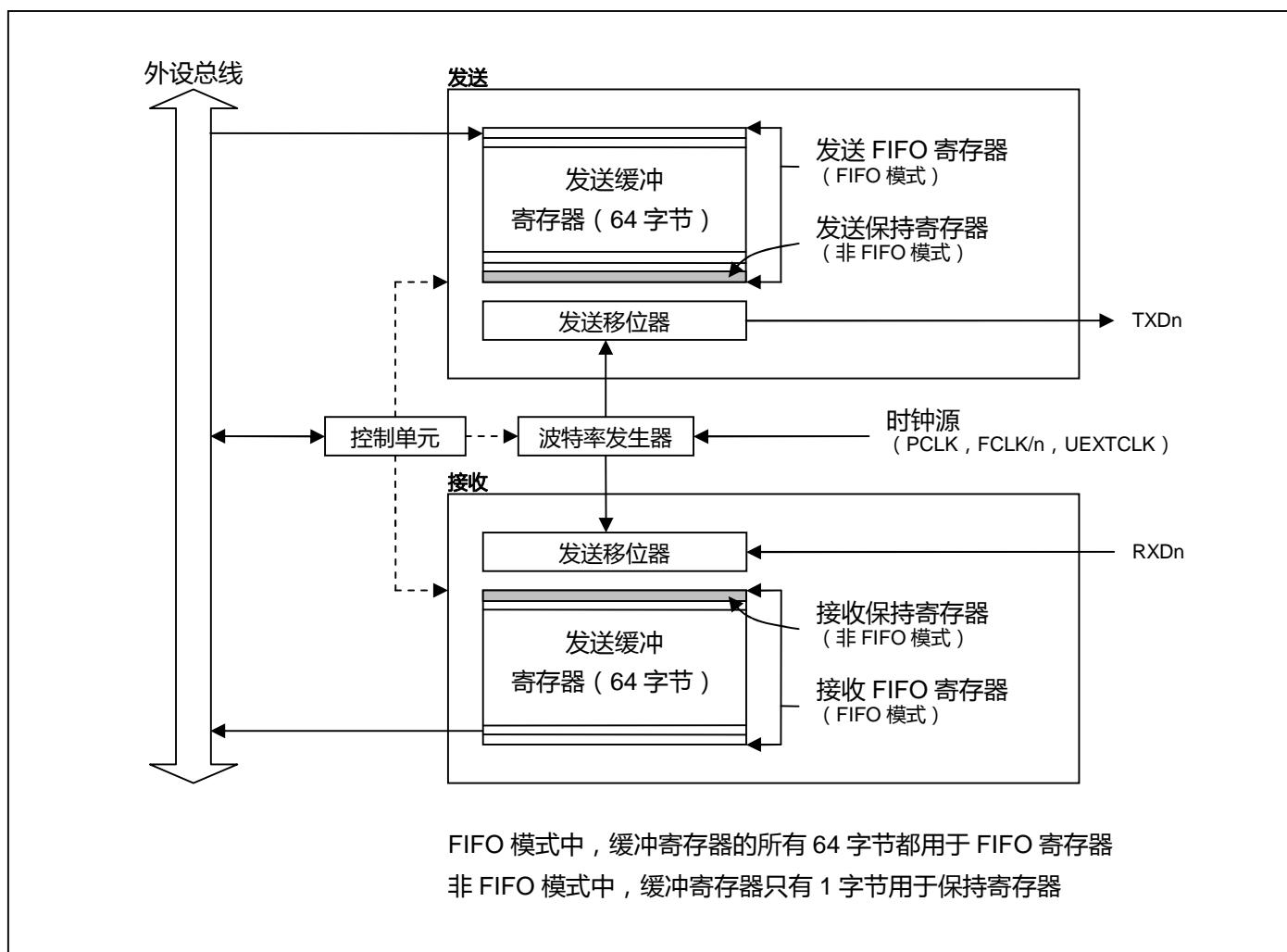


图 11-1 UART 方框图 (带 FIFO)

UART 操作

下述章节描述了 UART 的操作，包括了数据发送，数据接收，中断发生，波特率发生，环回（Loopback）模式，红外模式和自动流控制。

数据发送

可编程发送数据帧。由 1 个起始位、5 至 8 位数据位、1 个可选奇偶校验位以及 1 至 2 个停止位组成，是由行控制寄存器（ULCONn）指定。发送器也可以产生单帧发送期间强制串行输出为逻辑 0 状态的断点状态。此模块在完成发送当前发送字后发送断点信号。在发出断点信号后，其不断发送数据到 Tx FIFO（非 FIFO 模式情况下 Tx 保持寄存器）中。

数据接收

与发送类似，接收数据帧也是可编程的。由 1 个起始位、5 至 8 位数据位、1 个可选奇偶校验位以及 1 至 2 个停止位组成，是由行控制寄存器（ULCONn）指定。接收器能够检测出溢出（overrun）错误、奇偶校验错误、帧错误和断点状态，每个都可以设置一个错误标志。

- 溢出错误表明新数据在读出旧数据前覆盖了旧数据。
- 奇偶校验错误表明接收器检测出一个非预期奇偶校验字段。
- 帧错误表明接收到的数据没有有效的结束位。
- 断点状态表明 RxDn 的输入保持为逻辑 0 状态的时间长于单帧传输时间。

当其在 3 字时间期间（此间隔在字宽位的设置随后）并且在 FIFO 模式中 Rx FIFO 为非空时不接收任何数据时发生接收超时状态。

自动流控制 (AFC)

S3C2440A 的 UART 0 和 UART 1 支持 nRTS 和 nCTS 信号的自动流控制。假设它可以被连接到外部 UART。如果用户希望连接 UART 到一个调制解调器，UMCONn 寄存器中禁止自动流控制位并且由软件控制 nRTS 信号。

AFC 中，nRTS 依靠接收器的状态和 nCTS 信号控制传输的操作。UART 的发送器只在当激活了 nCTS 信号时发送数据到 FIFO 中 (AFC 中，nCTS 意味着其它 UART 的 FIFO 准备好了接收数据)。在 UART 接收数据前，当接收 FIFO 多于 32 字节的空间时必须激活 nRTS 并且当接收 FIFO 少于 32 字节的空间时必须取消激活的 nRTS (AFC 中，nRTS 意味着其自己的接收 FIFO 准备好了接收数据)。

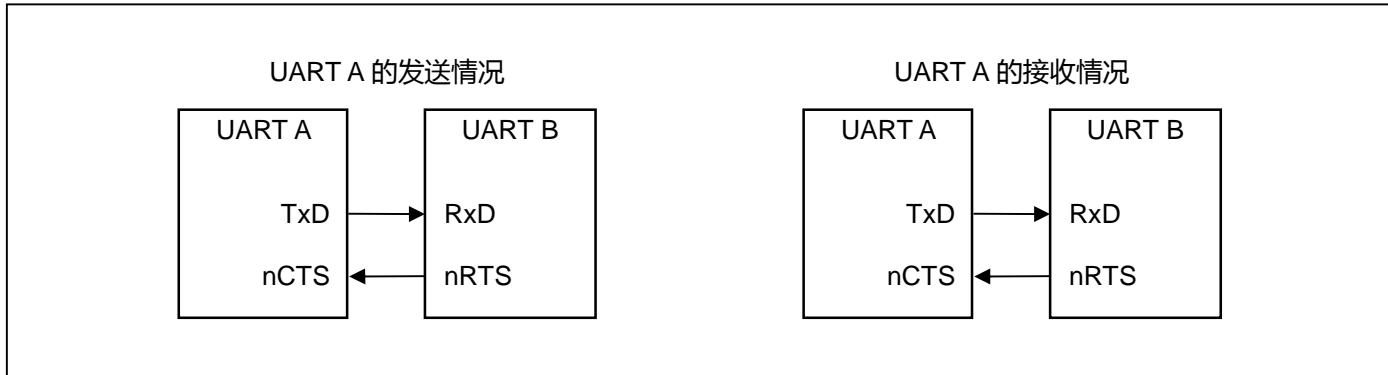


图 11-2. UART AFC 接口

注意：

UART 2 不支持 AFC 接口，因为 S3C2440A 没有 nRTS2 和 nCTS2。

非自动流控制的例子 (软件控制 nRTS 和 nCTS)

带 FIFO 的 Rx 操作

1. 选择接收模式 (中断或 DMA 模式)。
2. 检查 UFSTATn 寄存器中 Rx FIFO 的计数。如果该值小于 32，用户必须设置 UMCONn[0]的值为'1' (激活 nRTS)，并且如果该值大于等于 32 用户必须设置 UMCONn[0]的值为'0' (取消激活 nRTS)。
3. 重复步骤 2。

带 FIFO 的 Tx 操作

1. 选择发送模式 (中断或 DMA 模式)。
2. 检查 UMSTATn[0]的值。如果值为'1' (激活 nCTS)，用户写入数据到 Tx FIFO 寄存器中。

RS-232C 接口

如果用户希望连接 UART 到调制解调接口（代替零调制解调模式），则需要 nRTS，nCTS，nDSR，nDTR，DCD 和 nRI 信号。在此情况下，由于 AFC 不支持 RS-232C 接口，用户可以通过软件使用通用 I/O 口控制这些信号。

中断/DMA 请求产生

S3C2440A 的每个 UART 包括 7 种状态 (Tx/Rx/错误) 信号：溢出错误、奇偶校验错误、帧错误、断点、接收缓冲器数据就绪、发送缓冲器空以及发送移位器空，全部都由相应 UART 状态寄存器 (UTRSTATn/UERSTATn) 标示。

溢出错误、奇偶校验错误、帧错误和断点状态被认为是接收错误的状态。如果接收错误中断请求使能位在控制寄存器 UCONn 中设置为 1，则每个都可以引起接收错误中断请求。当检测到接收错误中断请求，读取 UERSTSTn 的值识别该信号引起请求。

当接收器在 FIFO 模式中转移接受移位器的数据到 Rx FIFO 寄存器中并且接收到的数据量达到 Rx FIFO 触发深度，并且如果在控制寄存器 (UCONn) 中的接收模式选择为 1 (中断请求或查询模式)，则发生接收中断。在非 FIFO 模式中，转移接受移位器的数据到接收保持寄存器将在中断请求和查询模式下引起 Rx 中断。

当发送器转移来自自身的发送 FIFO 寄存器的数据到其发送移位器并且在移出发送 FIFO 的数据量达到 Tx FIFO 触发深度，如果在控制寄存器中的发送模式选择了中断请求或查询模式，则发生 Tx 中断。在非 FIFO 模式中，转移来自发送保持寄存器中的数据到发送移位器在中断请求和查询模式下将引起 Tx 中断。

如果控制寄存器中的接收模式和发送模式被选择为 DMA_n 请求模式，则 DMA_n 请求发生所替代上述情况的 Tx 或 Rx 中断。

表 11-1. 中断与 FIFO 的关系

类型	FIFO 模式	非 FIFO 模式
Rx 中断	只要接收数据到达接收 FIFO 的触发深度则发生。 当 FIFO 中的数据量没有达到 Rx FIFO 触发深度并且超过 3 字时间 (接收超时) 都未接收任何数据时发生。此间隔在随后的字宽位设置。	只要接收缓冲器变为满则由接收保持寄存器发生。
Tx 中断	只要发送数据达到发送 FIFO (Tx FIFO 触发深度) 的触发深度则发生。	只要发送缓冲器变为空则由接收保持寄存器发生。
错误中断	当检测到帧错误、奇偶校验错误或断点信号时则发生。当其到达接收 FIFO 顶却没有读出其中数据时则发生。	所有错误都引起发生。然而如果同时发生另一个错误，只产生一个中断。

UART 错误状态 FIFO

UART 拥有 Rx FIFO 寄存器之外的错误状态 FIFO。错误状态 FIFO 指示出 FIFO 寄存器之中的数据错误接收。错误中断将只在数据包含错误并准备读出时发出。为了清除错误状态 FIFO，带错误的 URXHn 和 UERSTATn 必须被读取出来。

例如，假定 UART Rx FIFO 顺序收到 A、B、C、D 和 E 字符，在接收'B'时发生了帧错误，并且在接收'D'时发生了奇偶校验错误。

实际上 UART 接收错误并不会产生任何错误中断，因为不会读取接收带错误字符。一旦读取字符将引发错误中断。

图 11-3 显示另外 UART 接收包含 2 个错误的 5 个字符。

时间	顺序流程	错误中断	注释
#0	当没有字符被读出时	-	
#1	接收了 A , B , C , D 和 E	-	
#2	读出 A 后	帧错误 (B 中) 中断发生	'B'必须被读出
#3	读出 B 后	-	
#4	读出 C 后	奇偶校验错误 (D 中) 中断发生	'D'必须被读出
#5	读出 D 后	-	
#6	读出 E 后	-	

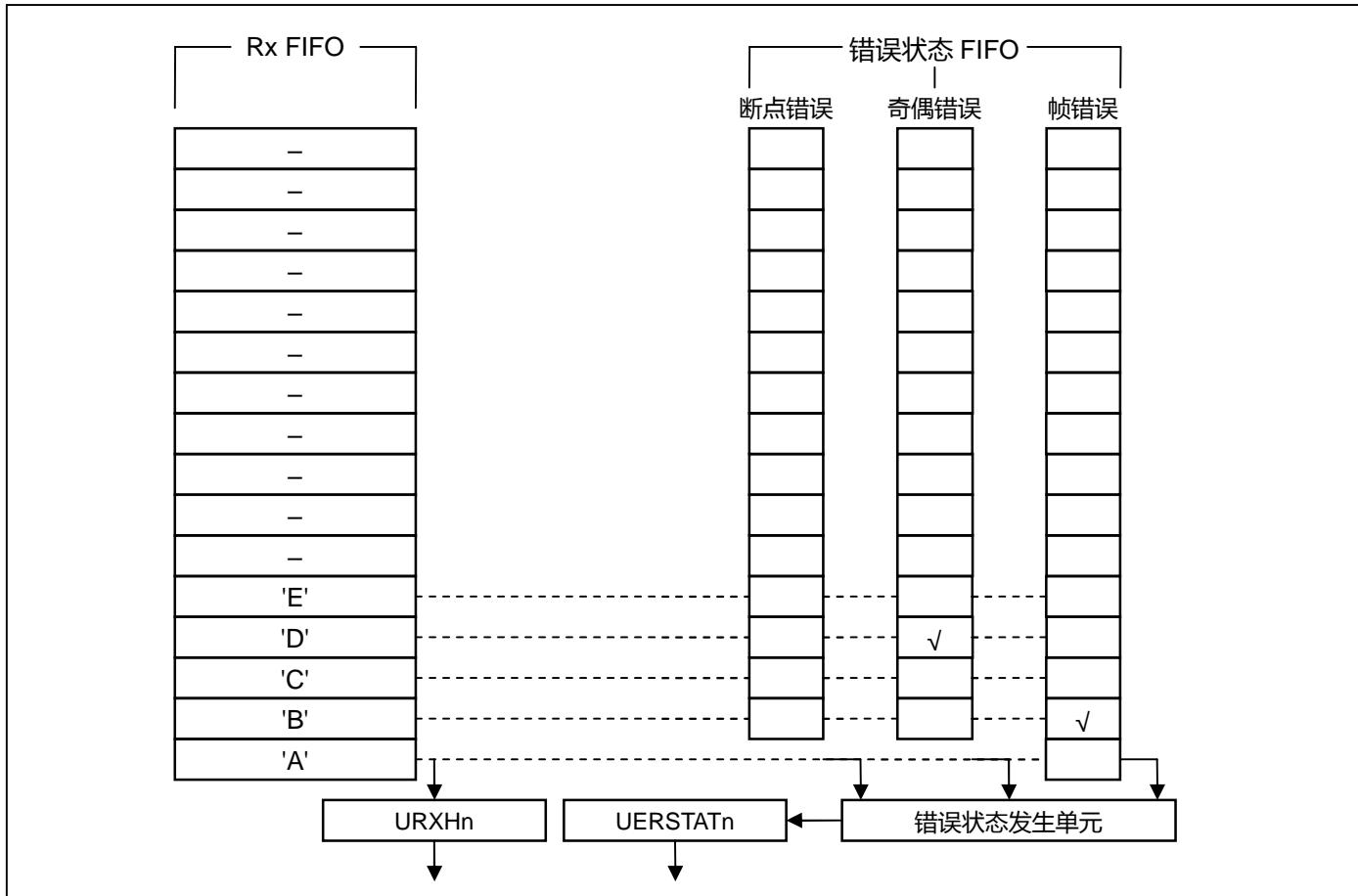


图 11-3. 显示 UART 接收带 2 个错误的 5 个字符的例子

波特率发生

每个 UART 的波特率发生器为发送器和接受器提供串行时钟。波特率发生器的源时钟可以选择 S3C2440A 的内部系统时钟或 UEXTCLK。换句话说，分频由设置 UCONn 的时钟选项选择。波特率时钟是通过 16 和由 UART 波特率分频寄存器 (UBRDIVn) 指定的 16 位分频系数来分频源时钟 (PCLK, FCLK/n 或 UEXTCLK) 产生的。UBRDIVn 由下列表达式决定：

$$\text{UBRDIVn} = (\text{int})(\text{UART 时钟} / (\text{波特率} \times 16)) - 1$$

(UART 时钟 : PCLK, FCLK/n 或 UEXTCLK)

当然，UBRDIVn 应该是从 1 至 ($2^{16}-1$)，只有在使用小于 PCLK 的 UEXTCLK 时设置为 0 (旁路模式)。

例如，如果波特率为 115200 bps 并且 UART 时钟为 40 MHz，则 UBRDIVn 为：

$$\begin{aligned}\text{UBRDIVn} &= (\text{int})(40000000 / (115200 \times 16)) - 1 \\ &= (\text{int})(21.7) - 1 \quad [\text{取最接近的整数}] \\ &= 22 - 1 = 21\end{aligned}$$

波特率误差容限

UART 帧误差应该小于 1.87% (3/160)。

$$t_{UPCLK} = (\text{UBRDIVn} + 1) \times 16 \times 1 \text{ 帧} / \text{PCLK}$$

t_{UPCLK} : 实际 UART 时钟

$$t_{UEXACT} = 1 \text{ 帧} / \text{波特率}$$

t_{UEXACT} : 理想 UART 时钟

$$\text{UART 误差} = (t_{UPCLK} - t_{UEXACT}) / t_{UEXACT} \times 100\%$$

注意：

1. 1 帧 = 起始位 + 数据位 + 奇偶校验位 + 停止位
2. 在指定字段中，可以最高支持 UART 波特率到 921.6K bps。例如，当 PCLK 为 60 MHz，可以在 1.69% 的 UART 误差下使用 921.6K bps。

环回模式

S3C2440A UART 提供了一个参考环回模式测试模式，有助于排除在通讯连接中的故障。结构上此模式允许 UART 的 RXD 和 TXD 的连接。因此在此模式中发送的数据通过 RXD 被接收器接收。这种特性允许处理器核查每条串行 IO 通道的内部发送和接收数据路径。通过设置 UART 控制寄存器 (UCONn) 中的环回位来选择此模式。

红外 (IR) 模式

S3C2440A 的 UART 模块支持红外 (IR) 发送和接收，可以通过 UART 行控制寄存器 (ULCONn) 中的红外位设置。图 11-4 说明了如何实现 IR 模式。

IR 发送模式中，发送出正常串行发送速率（当发送数据位为 0 时）的 3/16 速率的脉冲；IR 接收模式中，接收器必须检测 3/16 脉冲周期以识别出 0 值（见图 11-6 和 11-7 中的帧时序示意图）。

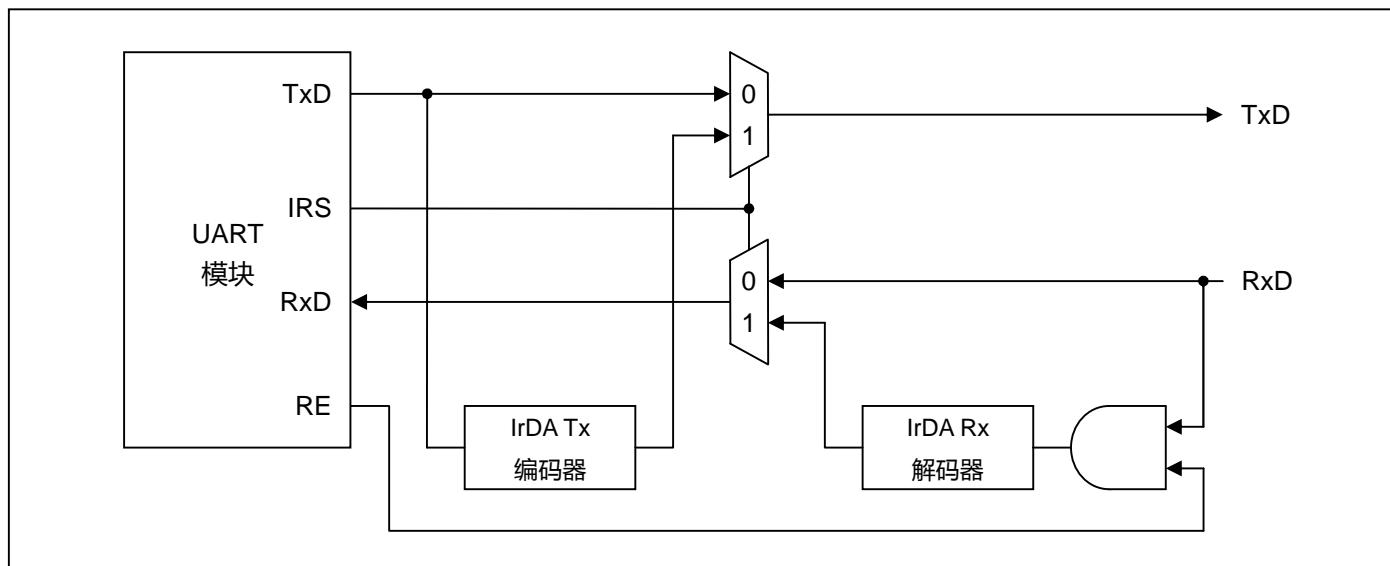


图 11-4. IrDA 功能方框图



图 11-5. 串行 I/O 帧时序示意图 (普通 UART)

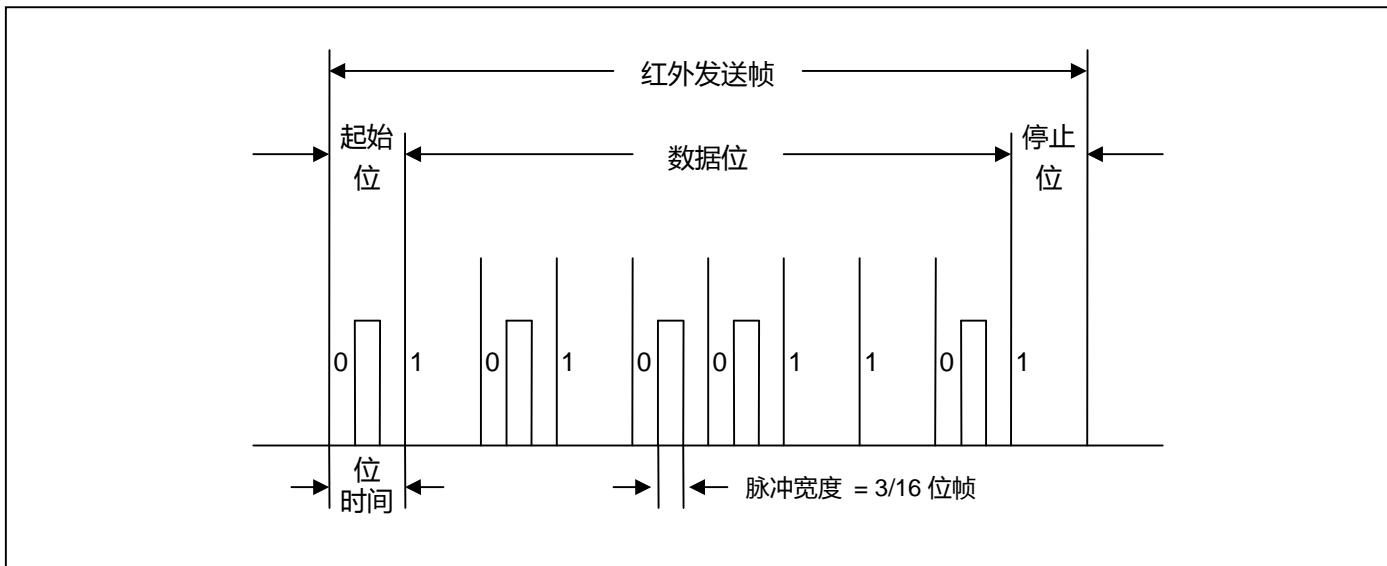


图 11-6. 红外发送模式帧时序示意图

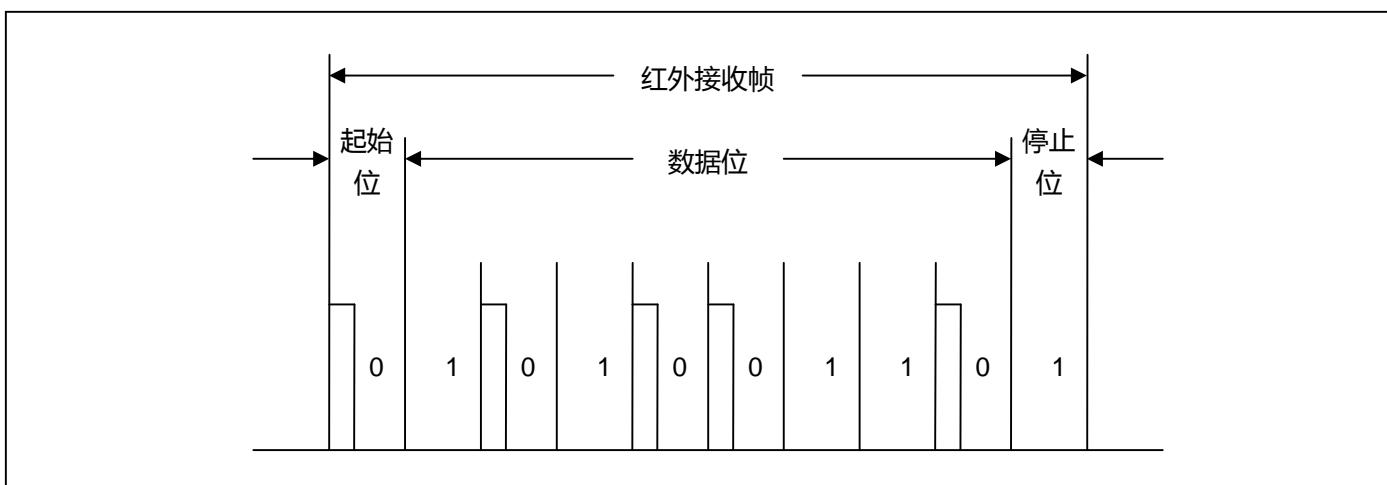


图 11-7. 红外接收模式帧时序示意图

UART 特殊寄存器

UART 线路控制寄存器

有 3 个 UART 线路控制寄存器，在 UART 模块中包含了 ULCON0、ULCON1 和 ULCON2。

寄存器	地址	R/W	描述	复位值
ULCON0	0x50000000	R/W	UART 通道 0 线路控制寄存器	0x00
ULCON1	0x50004000	R/W	UART 通道 1 线路控制寄存器	0x00
ULCON2	0x50008000	R/W	UART 通道 2 线路控制寄存器	0x00

ULCONn	位	描述	初始状态
保留	[7]	-	0
红外模式	[6]	决定是否使用红外模式 0 = 普通模式操作 1 = 红外 Tx/Rx 模式	0
奇偶校验模式	[5:3]	指定在 UART 发送和接收操作期间奇偶校验产生和检查的类型 0xx = 无奇偶校验 100 = 奇校验 101 = 偶校验 110 = 固定/检查奇偶校验为 1 111 = 固定/检查奇偶校验为 0	000
停止位数	[2]	指定用于结束帧信号的停止位的个数 0 = 每帧 1 个停止位 1 = 每帧 2 个停止位	0
字长度	[1:0]	指出每帧用于发送或接收的数据位的个数 00 = 5 位 01 = 6 位 10 = 7 位 11 = 8 位	00

UART 控制寄存器

有 3 个 UART 控制寄存器，在 UART 模块中包含了 UCON0、UCON1 和 UCON2。

寄存器	地址	R/W	描述	复位值
UCON0	0x50000004	R/W	UART 通道 0 控制寄存器	0x00
UCON1	0x50004004	R/W	UART 通道 1 控制寄存器	0x00
UCON2	0x50008004	R/W	UART 通道 2 控制寄存器	0x00

UCONn	位	描述	初始状态
FCLK 分频器	[15:12]	<p>当 UART 时钟源选择了 FCLK/n 时的分频器值。‘n’由 UCON0[15:12]、UCON1[15:12]、UCON2[14:12]所决定。</p> <p>UCON2[15]为 FCLK/n 时钟使能/禁止位。</p> <p>设置‘n’为 7 至 21 时，则使用 UCON0[15:12]；设置‘n’为 22 至 36 时，则使用 UCON0[15:12]；设置‘n’为 37 至 43 时，则使用 UCON0[14:12]。</p> <p>UCON2[15] : 0 = 禁止 FCLK/n 时钟；1 = 使能 FCLK/n 时钟。</p> <p>UCON0 的情况：UART 时钟 = FCLK / (分频器+6)；分频器>0。 UCON1，UCON2 必须为 0。</p> <p>例) 1 : UART 时钟 = FCLK/7 ; 2 : UART 时钟 = FCLK/8 ; 3 : UART 时钟 = FCLK/9 ;15 : UART 时钟 = FCLK/21。 UCON1 的情况：UART 时钟 = FCLK / (分频器+21)；分频器>0。 UCON0，UCON2 必须为 0。</p> <p>例) 1 : UART 时钟 = FCLK/22 ; 2 : UART 时钟 = FCLK/23 ; 3 : UART 时钟 = FCLK/24 ;15 : UART 时钟 = FCLK/36 UCON2 的情况：UART 时钟 = FCLK / (分频器+36)；分频器>0。 UCON0，UCON1 必须为 0。</p> <p>例) 1 : UART 时钟 = FCLK/37 ; 2 : UART 时钟 = FCLK/38 ; 3 : UART 时钟 = FCLK/39 ;15 : UART 时钟 = FCLK/43 如果 UCON0/1[15:12]和 UCON2[14:12]都全为‘0’，则分频器将为 44，即 UART 时钟 = FCLK/44。总分频范围在 7 到 44 之间。</p>	0000
时钟选择	[11:10]	<p>选择 PCLK，UEXTCLK 或 FCLK/n 给 UART 波特率。</p> <p>UBRDIVn = (int)(被选时钟 / (波特率 × 16)) – 1</p> <p>00 = PCLK 10 = PCLK 01 = UEXTCLK 11 = FCLK/n (如果希望选择 FCLK/n，应该在选择或取消选择 FCLK/n 后加上“NOTE”的代码。)</p>	00

UART 控制寄存器 (续)

Tx 中断类型	[9]	中断请求类型。 0 = 脉冲(非 FIFO 模式中 Tx 缓冲器一变为空或 FIFO 模式中达到 Tx FIFO 触发深度就请求中断) 1 = 电平(当非 FIFO 模式中 Tx 缓冲器为空或 FIFO 模式中达到 Tx FIFO 触发深度时请求中断)	0
Rx 中断类型	[8]	中断请求类型。 0 = 脉冲(非 FIFO 模式中 Rx 缓冲器接收到数据或 FIFO 模式中达到 Rx FIFO 触发深度则立刻请求中断) 1 = 电平(当非 FIFO 模式中 Rx 缓冲器正在接收数据或 FIFO 模式中达到 Rx FIFO 触发深度请求中断)	0
Rx 超时使能	[7]	当使能了 UART FIFO 使能/禁止 Rx 超时中断。该中断是一个接收中断 0 = 禁止 1 = 使能	0
Rx 错误状态 中断使能	[6]	异常时允许 UART 产生中断，如接收操作期间的断点、帧错误、奇偶错误或溢出错误。 0 = 不产生接收错误状态中断 1 = 产生接收错误状态中断	0
环回模式	[5]	设置环回模式为 1 使得 UART 进入环回模式。此模式只用于测试。 0 = 正常操作 1 = 环回模式	0
发出断点信号	[4]	设置此位使得 UART 在单帧期间发出一个断点信号。此位在发出断点信号后将自动清零。 0 = 正常传输 1 = 发出断点信号	0

注意：应该在选择或取消选择 FCLK/n 后加上以下代码。

```
rGPHCON = rGPHCON & ~(3<<16);           //GPH8 (UEXTCLK) 输入
Delay(1);                                //大约 100μs
rGPHCON = rGPHCON & ~(3<<16) | (1<<17); //GPH8 (UEXTCLK) UEXTCLK
```

UART FIFO 控制寄存器

有 3 个 UART FIFO 控制寄存器，在 UART 模块中包含了 UFCON0、UFCON1 和 UFCON2。

寄存器	地址	R/W	描述	复位值
UFCON0	0x50000008	R/W	UART 通道 0 FIFO 控制寄存器	0x0
UFCON1	0x50004008	R/W	UART 通道 1 FIFO 控制寄存器	0x0
UFCON2	0x50008008	R/W	UART 通道 2 FIFO 控制寄存器	0x0

UFCONn	位	描述	初始状态
Tx FIFO 触发深度	[7:6]	决定发送 FIFO 的触发深度 00 = 空 10 = 32 字节 01 = 16 字节 11 = 48 字节	00
Rx FIFO 触发深度	[5:4]	决定接收 FIFO 的触发深度 00 = 1 字节 10 = 16 字节 01 = 8 字节 11 = 32 字节	00
保留	[3]	-	0
Tx FIFO 复位	[2]	复位 FIFO 后自动清零 0 = 正常 1 = Tx FIFO 复位	0
Rx FIFO 复位	[1]	复位 FIFO 后自动清零 0 = 正常 1 = Rx FIFO 复位	0
FIFO 使能	[0]	0 = 禁止 1 = 使能	0

注意：

当 UART 未达到 FIFO 触发深度或在带 FIFO 的 DMA 接收模式中 3 字周期期间没有收到数据时，将发生 Rx 中断（接收超时），用户应该检查 FIFO 状态并读出剩余部分。

UART MODEM 控制寄存器

有 2 个 UART MODEM 控制寄存器，在 UART 模块中包含了 UMCON0 和 UMCON1。

寄存器	地址	R/W	描述	复位值
UMCON0	0x5000000C	R/W	UART 通道 0 Modem 控制寄存器	0x0
UMCON1	0x5000400C	R/W	UART 通道 1 Modem 控制寄存器	0x0
保留	0x5000800C	-	保留	未定义

UMCONn	位	描述	初始状态
保留	[7:5]	这些位必须为‘0’	000
自动流控制(AFC)	[4]	0 = 禁止 1 = 使能	0
保留	[3:1]	这些位必须为‘0’	000
请求传送	[0]	如果 AFC 位为使能，将忽略此值。这种情况下 S3C2440A 将自动控制 nRTS。如果 AFC 位为禁止，nRTS 必须由软件控制。 0 = 高电平（撤消 nRTS） 1 = 低电平（激活 nRTS）	0

注意：

UART 2 不支持 AFC 功能，因为 S3C2440A 没有 nRTS2 和 nCTS2。

UART TX/RX 状态寄存器

有 3 个 UART TX/RX 状态寄存器，在 UART 模块中包含了 UTRSTAT0，UTRSTAT1 和 UTRSTAT2。

寄存器	地址	R/W	描述	复位值
UTRSTAT0	0x50000010	R	UART 通道 0 Tx/Rx 状态寄存器	0x6
UTRSTAT1	0x50004010	R	UART 通道 1 Tx/Rx 状态寄存器	0x6
UTRSTAT2	0x50008010	R	UART 通道 2 Tx/Rx 状态寄存器	0x6

UTRSTATn	位	描述	初始状态
发送器空	[2]	当发送缓冲寄存器无有效数据要发送并且发送移位寄存器为空时将自动设置为 1。 0 = 非空 1 = 发送器（发送缓冲和移位寄存器）空	1
发送缓冲器空	[1]	当发送缓冲寄存器为空时自动设置为 1。 0 = 缓冲寄存器非空 1 = 空（非 FIFO 模式中，请求中断或 DMA。FIFO 模式中，当 Tx FIFO 触发深度设置为 00（空）时请求中断或 DMA） 如果 UART 使用 FIFO，用户应该用 UFSTAT 寄存器中的 Rx FIFO 计数位和 Rx FIFO 满位取对代此位的检查。	1
接收缓冲器 数据就绪	[0]	每当通过 RXDn 端口接收数据，接收缓冲寄存器包含了有效数据时自动设置为 1。 0 = 空 1 = 缓冲寄存器接收到有效数据（非 FIFO 模式中，请求中断或 DMA） 如果 UART 使用 FIFO，用户应该用 UFSTAT 寄存器中的 Rx FIFO 计数位和 Rx FIFO 满位取对代此位的检查。	0

UART 错误状态寄存器

有 3 个 UART Rx 错误状态寄存器，在 UART 模块中包含了 UERSTAT0，UERSTAT1 和 UERSTAT2。

寄存器	地址	R/W	描述	复位值
UERSTAT0	0x50000014	R	UART 通道 0 Rx 错误状态寄存器	0x0
UERSTAT1	0x50004014	R	UART 通道 1 Rx 错误状态寄存器	0x0
UERSTAT2	0x50008014	R	UART 通道 2 Rx 错误状态寄存器	0x0

UERSTATn	位	描述	初始状态
断点监测	[3]	表明接收到断点信号时将自动设置为 1。 0 = 未接收到断点 1 = 接收到断点（请求中断）	0
帧错误	[2]	每当接收操作期间发生帧错误时将自动设置为 1。 0 = 接收期间无帧错误 1 = 帧错误（请求中断）	0
奇偶校验错误	[1]	每当接收操作期间发生奇偶校验错误时将自动设置为 1。 0 = 非空 1 = 奇偶校验错误（请求中断）	0
溢出错误	[0]	每当接收操作期间发生溢出错误时将自动设置为 1。 0 = 非空 1 = 溢出错误（请求中断）	0

注意：

这些位 (UERSATn[3:0]) 都将在读取 UART 错误状态寄存器后时自动清零。

UART FIFO 状态寄存器

有 3 个 UART FIFO 状态寄存器，在 UART 模块中包含了 UFSTAT0, UFSTAT1 and UFSTAT2。

寄存器	地址	R/W	描述	复位值
UFSTAT0	0x50000018	R	UART 通道 0 FIFO 状态寄存器	0x00
UFSTAT1	0x50004018	R	UART 通道 1 FIFO 状态寄存器	0x00
UFSTAT2	0x50008018	R	UART 通道 2 FIFO 状态寄存器	0x00

UFSTATn	位	描述	初始状态
保留	[15]	-	0
Tx FIFO 满	[14]	每当发送操作期间发送 FIFO 为满时将自动设置为 1。 0 = 0 字节 ≤ Tx FIFO 数据 ≤ 63 字节 1 = 满	0
Tx FIFO 计数器	[13:8]	Tx FIFO 中的数据量	0
保留	[7]	-	0
Rx FIFO 满	[6]	每当接收操作期间接收 FIFO 为满时将自动设置为 1。 0 = 满 1 = 0 字节 ≤ Rx FIFO 数据 ≤ 63 字节	0
Rx FIFO 计数器	[5:0]	Rx FIFO 中的数据量	0

UART MODEM 状态寄存器

有 2 个 UART MODEM 状态寄存器，在 UART 模块中包含了 UMSTAT0，UMSTAT1。

寄存器	地址	R/W	描述	复位值
UMSTAT0	0x5000001C	R	UART 通道 0 Modem 状态寄存器	0x0
UMSTAT1	0x5000401C	R	UART 通道 1 Modem 状态寄存器	0x0
保留	0x5000801C	-	保留	未定义

UMSTATn	位	描述	初始状态
CTS 变化	[4]	指示 CPU 最后读取那次以后输入到 S3C2440A 的 nCTS 状态的变化。 (参考图 11-8) 0 = 未改变 1 = 变化了	0
保留	[3:1]	-	0
清除以发送	[0]	0 = CTS 信号未被激活 (nCTS 引脚为高电平) 1 = CTS 信号被激活 (nCTS 引脚为低电平)	0

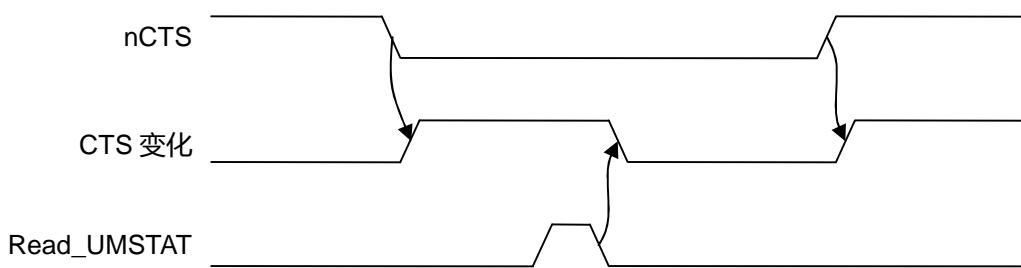


图 11-8. nCTS 和 Δ CTS 时序示意图

UART 发送缓冲寄存器 (保持寄存器和 FIFO 寄存器)

有 3 个 UART 发送缓冲状态寄存器，在 UART 模块中包含了 UTXH0，UTXH1 和 UTXH2。UTXHn 发送数据为 8 位数据。

寄存器	地址	R/W	描述	复位值
UTXH0	0x50000020(L) 0x50000023(B)	W (字节)	UART 通道 0 发送缓冲寄存器	-
UTXH1	0x50004020(L) 0x50004023(B)	W (字节)	UART 通道 1 发送缓冲寄存器	-
UTXH2	0x50008020(L) 0x50008023(B)	W (字节)	UART 通道 2 发送缓冲寄存器	-

UTXHn	位	描述	初始状态
TXDATA _n	[7:0]	UART _n 要发送的数据	-

注意：(L)：小端模式；(B)：大端模式。

UART 接收缓冲寄存器 (保持寄存器和 FIFO 寄存器)

有 3 个 UART 接收缓冲状态寄存器，在 UART 模块中包含了 URXH0，URXH1 和 URXH2。URXHn 发送数据为 8 位数据。

寄存器	地址	R/W	描述	复位值
URXH0	0x50000024(L) 0x50000027(B)	W (字节)	UART 通道 0 接收缓冲寄存器	-
URXH1	0x50004024(L) 0x50004027(B)	W (字节)	UART 通道 1 接收缓冲寄存器	-
URXH2	0x50008024(L) 0x50008027(B)	W (字节)	UART 通道 2 接收缓冲寄存器	-

URXHn	位	描述	初始状态
RXDATA _n	[7:0]	UART _n 接收到的数据	-

注意：

当发生溢出错误时，必须读出 URXHn。如果未读出，即使清除了 UERSTAT_n 的溢出位下次接收数据将同样发出溢出错误。

UART 波特率分频寄存器

有 3 个 UART 波特率分频寄存器，在 UART 模块中包含了 UBRDIV0，UBRDIV1 和 UBRDIV2。URXHn 发送数据为 8 位数据。

存储在波特率分频寄存器中的值 (UBRDIVn) 是用于决定如下的串行 Tx/Rx 时钟率 (波特率)：

$$\text{UBRDIVn} = (\text{int})(\text{UART 时钟} / (\text{波特率} \times 16)) - 1$$

(UART 时钟 : PCLK , FCLK/n 或 UEXTCLK)

此处，UBRDIVn 应该是从 1 到 (216-1)，但只在使用低于 PCLK 的 UEXTCLK 时可以设置为 0。

例如，如果波特率为 115200 bps 并且 UART 时钟为 40 MHz，UBRDIVn 为：

$$\text{UBRDIVn} = (\text{int})(40000000 / (115200 \times 16)) - 1$$

= (int)(21.7) - 1 [取最接近的整数]

$$= 22 - 1 = 21$$

寄存器	地址	R/W	描述	复位值
UBRDIV0	0x50000028	R/W	波特率分频寄存器 0	-
UBRDIV1	0x50004028	R/W	波特率分频寄存器 1	-
UBRDIV2	0x50008028	R/W	波特率分频寄存器 2	-

UBRDIVn	位	描述	初始状态
UBRDIV	[15:0]	波特率分频值 UBRDIVn > 0。 使用 UEXTCLK 作为输入时钟时，可以设置 UBRDIVn 为‘0’。	-

12 USB 主机控制器

概述

S3C2440A 支持 2 个端口的 USB 主机接口，如下：

- 兼容 OHCI Rev 1.0
- 兼容 USB Rev1.1
- 两路下行端口
- 支持低速及全速 USB 设备

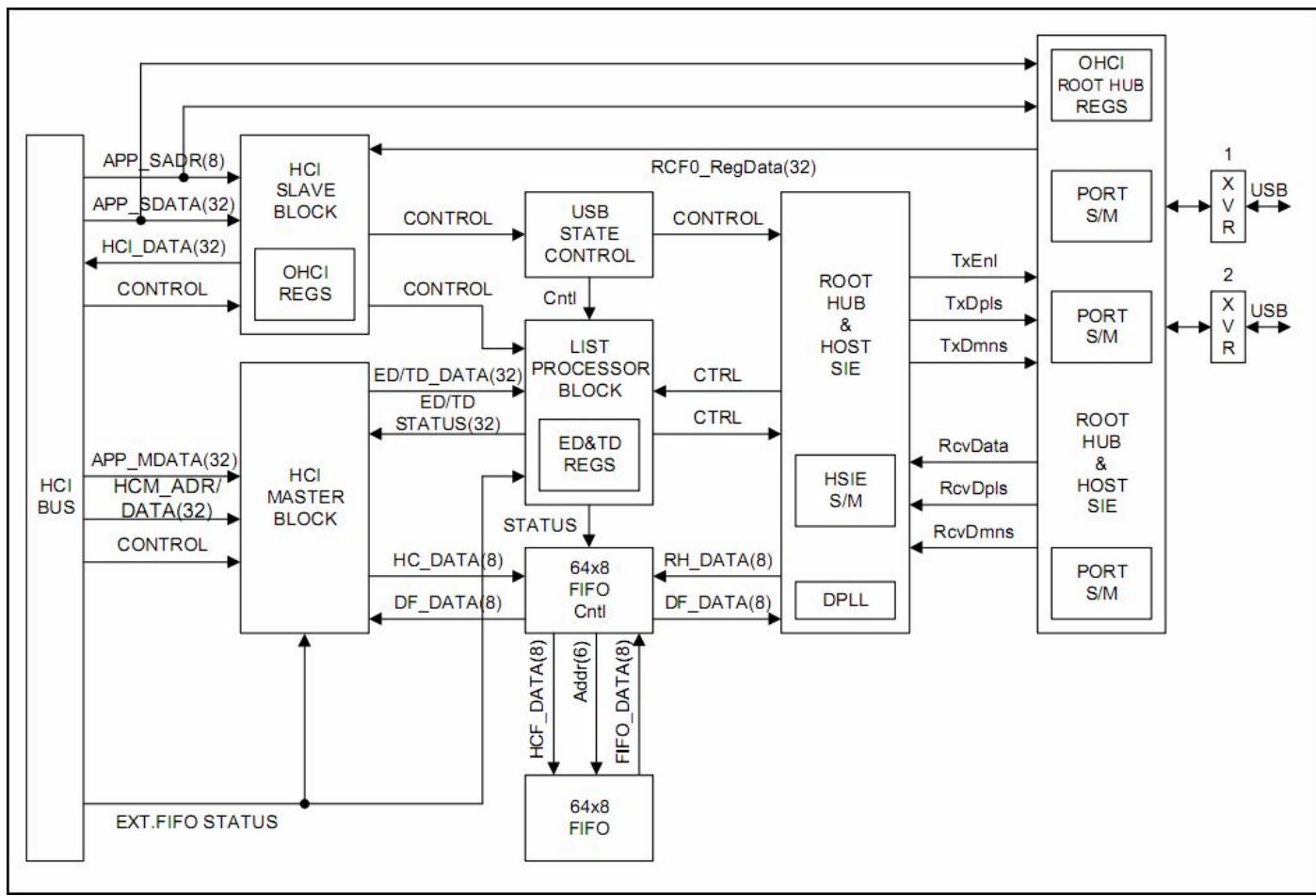


图 12-1. USB 主机控制器方框图

小端 (Little-Endian) 格式

S3C2440A USB 主机控制器符合 OHCI Rev 1.0。详细资料请参考 *Open Host Controller Interface Rev 1.0* 规范。

USB 主机控制器的 OHCI 寄存器

寄存器	地址	R/W	描述	复位值
HcRevision	0x49000000	—	控制及状态组	—
HcControl	0x49000004	—		—
HcCommonStatus	0x49000008	—		—
HcInterruptStatus	0x4900000C	—		—
HcInterruptEnable	0x49000010	—		—
HcInterruptDisable	0x49000014	—		—
HcHCCA	0x49000018	—	存储器指针组	—
HcPeriodCuttentED	0x4900001C	—		—
HcControlHeadED	0x49000020	—		—
HcControlCurrentED	0x49000024	—		—
HcBulkHeadED	0x49000028	—		—
HcBulkCurrentED	0x4900002C	—		—
HcDoneHead	0x49000030	—	帧控制组	—
HcRmInterval	0x49000034	—		—
HcFmRemaining	0x49000038	—		—
HcFmNumber	0x4900003C	—		—
HcPeriodicStart	0x49000040	—		—
HcLSThreshold	0x49000044	—		—
HcRhDescriptorA	0x49000048	—	逻辑根集线器 (Root Hub) 组	—
HcRhDescriptorB	0x4900004C	—		—
HcRhStatus	0x49000050	—		—
HcRhPortStatus1	0x49000054	—		—
HcRhPortStatus2	0x49000058	—		—

13 USB 设备控制器

概述

通用串行总线 (**USB**) 设备控制器被设计为提供了高性能全速功能的带 DMA 接口控制器解决方案。USB 设备控制器允许 DMA 的批量传输、中断传输和控制传输。

USB 设备控制器支持：

- 兼容 USB 规格 1.1 版本的全速 USB 设备控制器。
- 批量传输的 DMA 接口
- 带 FIFO 的 5 个端点
 - EP0 : 16 字节 (寄存器)
 - EP1 : 128 字节 IN/OUT FIFO (双口异步 RAM) : 中断或 DMA
 - EP2 : 128 字节 IN/OUT FIFO (双口异步 RAM) : 中断或 DMA
 - EP3 : 128 字节 IN/OUT FIFO (双口异步 RAM) : 中断或 DMA
 - EP4 : 128 字节 IN/OUT FIFO (双口异步 RAM) : 中断或 DMA
- 完整的 USB 收发器

特性

- 完全兼容 USB 规格 1.1 版本
- 全速 (12 Mbps) 设备
- 完整的 USB 收发器
- 支持控制、中断和批量传输
- 带 FIFO 的 5 个端点
 - 1 个带 16 字节 FIFO (EP0) 的双向控制端点
 - 4 个带 128 字节 FIFO (EP1、EP 2、EP 3 和 EP 4) 的双向批量端点
- 支持 DMA 接口给发送和接收 FIFO 批量端点 (EP1、EP 2、EP 3 和 EP 4)
- 独立的最大吞吐量为 128 字节的发送和接收 FIFO
- 支持挂起和远程唤醒功能

注意：

PCLK 应该高于 20 MHz 的稳定状态下用于 USB 控制器。

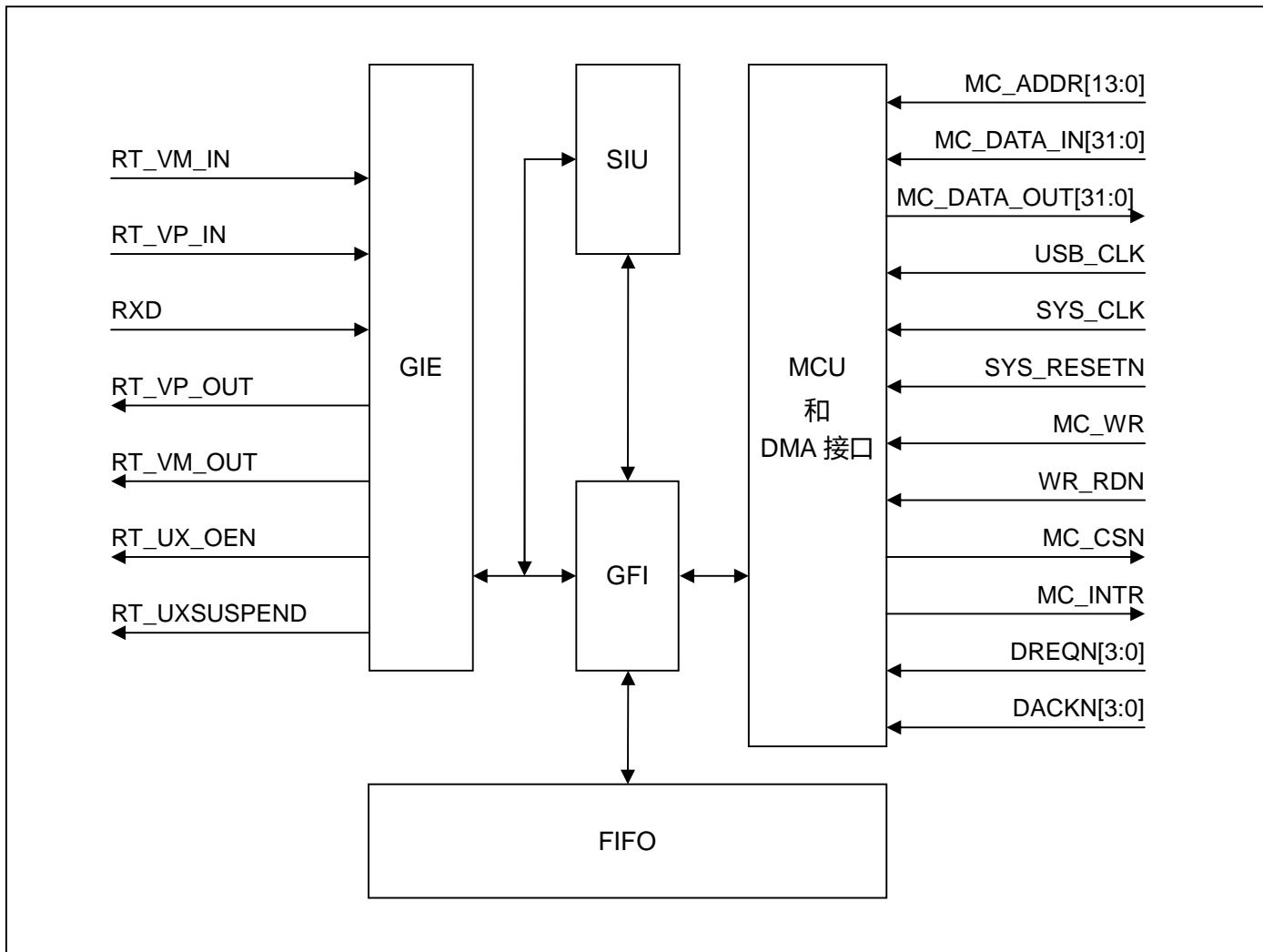


图 13-1. USB 设备控制器方框图

USB 设备控制器特殊寄存器

此节描述了 USB 设备控制器寄存器组的详细功能。所有特殊功能寄存器都可以按字节访问的或按字访问的。如果按字节模式访问则偏移地址在大端和小端模式中是不同的。所有保留位都为 0。

通常被标记的寄存器随 INDEX 寄存器 (INDEX_REG) (偏移地址 : 0X178) 值而定。例如如果希望改写 EP0 CSR 寄存器，必须在写 IN_CSR1 寄存器前写入 '0x00' 到 INDEX_REG 中。

注意：

必须在执行了主机复位信号后重置所有寄存器。

寄存器名称	描述	偏移地址
非标记寄存器		
FUNC_ADDR_REG	功能地址寄存器	0x140(L) / 0x143(B)
PWR_REG	电源管理寄存器	0x144(L) / 0x147(B)
EP_INT_REG (EP0-EP4)	端点中断寄存器	0x148(L) / 0x14B(B)
USB_INT_REG	USB 中断寄存器	0x158(L) / 0x15B(B)
EP_INT_EN_REG (EP0-EP4)	端点中断使能寄存器	0x15C(L) / 0x15F(B)
USB_INT_EN_REG	USB 中断使能寄存器	0x16C(L) / 0x16F(B)
FRAME_NUM1_REG	帧号 1 寄存器	0x170(L) / 0x173(B)
FRAME_NUM2_REG	帧号 2 寄存器	0x174(L) / 0x177(B)
INDEX_REG	标记寄存器	0x178(L) / 0x17B(B)
EP0_FIFO_REG	端点 0 FIFO 寄存器	0x1C0(L) / 0x1C3(B)
EP1_FIFO_REG	端点 1 FIFO 寄存器	0x1C4(L) / 0x1C7(B)
EP2_FIFO_REG	端点 2 FIFO 寄存器	0x1C8(L) / 0x1CB(B)
EP3_FIFO_REG	端点 3 FIFO 寄存器	0x1CC(L) / 0x1CF(B)
EP4_FIFO_REG	端点 4 FIFO 寄存器	0x1D0(L) / 0x1D3(B)
EP1_DMA_CON	端点 1 DMA 控制寄存器	0x200(L) / 0x203(B)
EP1_DMA_UNIT	端点 1 DMA 单元计数器寄存器	0x204(L) / 0x207(B)
EP1_DMA_FIFO	端点 1 DMA FIFO 计数器寄存器	0x208(L) / 0x20B(B)
EP1_DMA_TTC_L	端点 1 传输计数器低字节寄存器	0x20C(L) / 0x20F(B)
EP1_DMA_TTC_M	端点 1 寄存器	0x210(L) / 0x213(B)
EP1_DMA_TTC_H	端点 1 寄存器	0x214(L) / 0x217(B)
EP2_DMA_CON	端点 2 DMA 控制寄存器	0x218(L) / 0x21B(B)
EP2_DMA_UNIT	端点 2 DMA 单元计数器寄存器	0x21C(L) / 0x21F(B)
EP2_DMA_FIFO	端点 2 DMA FIFO 计数器寄存器	0x220(L) / 0x223(B)
EP2_DMA_TTC_L	端点 2 传输计数器低字节寄存器	0x224(L) / 0x227(B)
EP2_DMA_TTC_M	端点 2 传输计数器中字节寄存器	0x228(L) / 0x22B(B)
EP2_DMA_TTC_H	端点 2 传输计数器高字节寄存器	0x22C(L) / 0x22F(B)

USB 设备控制器特殊寄存器)(续)

寄存器名称	描述	偏移地址
EP3_DMA_CON	端点 3 DMA 控制寄存器	0x240(L) / 0x243(B)
EP3_DMA_UNIT	端点 3 DMA 单元计数器寄存器	0x244(L) / 0x247(B)
EP3_DMA_FIFO	端点 3 DMA FIFO 计数器寄存器	0x248(L) / 0x24B(B)
EP3_DMA_TTC_L	端点 3 传输计数器低字节寄存器	0x24C(L) / 0x24F(B)
EP3_DMA_TTC_M	端点 3 传输计数器中字节寄存器	0x250(L) / 0x253(B)
EP3_DMA_TTC_H	端点 3 传输计数器高字节寄存器	0x254(L) / 0x247(B)
EP4_DMA_CON	端点 4 DMA 控制寄存器	0x258(L) / 0x25B(B)
EP4_DMA_UNIT	端点 4 DMA 单元计数器寄存器	0x25C(L) / 0x25F(B)
EP4_DMA_FIFO	端点 4 DMA FIFO 计数器寄存器	0x260(L) / 0x263(B)
EP4_DMA_TTC_L	端点 4 传输计数器低字节寄存器	0x264(L) / 0x267(B)
EP4_DMA_TTC_M	端点 4 传输计数器中字节寄存器	0x268(L) / 0x26B(B)
EP4_DMA_TTC_H	端点 4 传输计数器高字节寄存器	0x26C(L) / 0x26F(B)
普通标记寄存器		
MAXP_REG	端点最大包寄存器	0x180(L) / 0x183(B)
输入标记寄存器		
IN_CSR1_REG/EP0_CSR	EP 输入控制状态寄存器 1/ EP0 控制状态寄存器	0x184(L) / 0x187(B)
IN_CSR2_REG	EP 输入控制状态寄存器 2	0x188(L) / 0x18B(B)
输出标记寄存器		
OUT_CSR1_REG	EP 输出控制状态寄存器 1	0x190(L) / 0x193(B)
OUT_CSR2_REG	EP 输出控制状态寄存器 2	0x194(L) / 0x197(B)
OUT_FIFO_CNT1_REG	EP 输出写计数寄存器 1	0x198(L) / 0x19B(B)
OUT_FIFO_CNT2_REG	EP 输出写计数寄存器 2	0x19C(L) / 0x19F(B)

功能地址寄存器 (FUNC_ADDR_REG)

此寄存器提供由主机指定的 USB 设备控制器地址。微控制器 (MCU) 通过一个到此寄存器的 SET_ADDRESS 描述符来写入接收值。此地址是用于下个令牌。

寄存器	地址	R/W	描述	复位值
FUNC_ADDR_REG	0x52000140(L) 0x52000143(B)	R/W (字节)	功能地址寄存器	0x00

FUNC_ADDR_REG	位	MCU	USB	描述	初始状态
ADDR_UPDATE	[7]	R /置位	R /清零	每当其更新此寄存器中的 FUNCTION_ADDR 字段时由 MCU 置位。此位将在 DATA_END 位在 EP0_CSR 中时由 USB 清零。	0
FUNCTION_ADDR	[6:0]	R/W	R	MCU 写入由主机指定的唯一地址到这个字段。	00

电源管理寄存器 (PWR_REG)

此寄存器在 USB 模块中起电源控制寄存器的作用。

寄存器	地址	R/W	描述	复位值
PWR_REG	0x52000144(L) 0x52000147(B)	R/W (字节)	电源管理寄存器	0x00

PWR_REG	位	MCU	USB	描述	初始状态
保留	[7:4]	-	-	-	-
USB_RESET	[3]	R	置位	如果从主机收到复位信号则由 USB 置位。只要复位信号保持着总线则此位仍然为置位。	00
MCU_RESUME	[2]	R/W	R /清零	由 MCU 置位给 MCU 重置。如果此位在挂起模式中被置位，USB 产生持续 10ms 的重置信号。	
SUSPEND_MODE	[1]	R	置位 /清零	当设备进入挂起模式时自动由 USB 置位。在以下条件下被清零： 1)为了结束远程重置信号 MCU 通过写'0'清除 MCU_RESUME 位。 2)收到主机的重置信号	
SUSPEND_EN	[0]	R/W	R	挂起模式使能控制位 0 = 禁止 (默认) 设备将不会进入挂起模式 1 = 使能挂起模式	

中断寄存器 (EP_INT_REG/USB_INT_REG)

USB 核包含两个中断寄存器。这些寄存器起到当中断时 MCU 的状态寄存器的作用。这些位通过写‘1’(不是‘0’)到要置位的每一位来清除。

一旦 MCU 发生中断，MCU 应该读取中断相关寄存器的内容并且如果需要写回清除其内容。

寄存器	地址	R/W	描述	复位值
EP_INT_REG	0x52000148(L) 0x5200014B(B)	R/W (字节)	EP 中断挂起/清除寄存器	0x00

EP_INT_REG	位	MCU	USB	描述	初始状态
EP1 至 EP4 中断	[4:1]	R /清零	置位	批量/中断输入端点： 以下情况下由 USB 置位： 1. 清除了 IN_PKT_RDY 位 2. 刷新了 FIFO 3. 置位了 SENT_STALL 批量/中断输出端点： 以下情况下由 USB 置位： 1. 置位了 OUT_PKT_RDY 位 2. 置位了 SENT_STALL 位	0
EP0 中断	[0]	R /清零	置位	端点 0 对应中断。 以下情况下由 USB 置位： 1. 置位了 OUT_PKT_RDY 位 2. 清除了 IN_PKT_RDY 位 3. 置位了 SENT_STALL 位 4. 置位了 SETUP_END 位 5. 清除了 DATA_END 位 (其表明了控制传输的结束)	0

寄存器	地址	R/W	描述	复位值
USB_INT_REG	0x52000158(L) 0x5200015B(B)	R/W (字节)	EP 中断挂起/清除寄存器	0x00

USB_INT_REG	位	MCU	USB	描述	初始状态
RESET 中断	[2]	R /清零	置位	当收到复位信号时由 USB 置位	0
RESUME 中断	[1]	R /清零	置位	挂起模式中当收到重置信号时由 USB 置位。如果由于 USB 复位发生了重置，接着首先中断带 RESUME 中断的 MCU。一旦时钟重置并且 SE0 状态持续 3ms，USB RESET 中断将生效。	0
SUSPEND 中断	[0]	R /清零	置位	当收到挂起信号时由 USB 置位。, 每当总线无活动多于 3ms 将置位此位。因此，如果 MCU 不在第一个挂起中断后停止时钟，它将会持续每 3ms 中断一次，直到 USB 总线无活动。默认是禁止此中断。	0

中断使能寄存器 (EP_INT_EN_REG/USB_INT_EN_REG)

对应每个中断寄存器，USB 设备控制器还包含了两个字段使能寄存器 (除了重置中断使能)。默认情况下 USB 复位中断为使能的。

如果位 = 0，则禁止中断。如果位 = 1，则允许中断。

寄存器	地址	R/W	描述	复位值
EP_INT_EN_REG	0x5200015C(L) 0x5200015F(B)	R/W (字节)	决定使能哪个中断	0xFF

EP_INT_EN_REG	位	MCU	USB	描述	初始状态
EP4_INT_EN	[4]	R/W	R	EP4 中断使能位 0 = 禁止中断 1 = 使能	1
EP3_INT_EN	[3]	R/W	R	EP3 中断使能位 0 = 禁止中断 1 = 使能	1
EP2_INT_EN	[2]	R/W	R	EP2 中断使能位 0 = 禁止中断 1 = 使能	1
EP1_INT_EN	[1]	R/W	R	EP1 中断使能位 0 = 禁止中断 1 = 使能	1
EP0_INT_EN	[0]	R/W	R	EP0 中断使能位 0 = 禁止中断 1 = 使能	1

寄存器	地址	R/W	描述	复位值
USB_INT_EN_REG	0x5200016C(L) 0x5200016F(B)	R/W (字节)	决定使能哪个中断	0x04

USB_INT_EN_REG	位	MCU	USB	描述	初始状态
RESET_INT_EN	[2]	R/W	R	复位中断使能位 0 = 禁止中断 1 = 使能	1
保留	[1]	-	-	-	0
SUSPEND_INT_EN	[0]	R/W	R	挂起中断使能位 0 = 禁止中断 1 = 使能	0

帧号寄存器 (FPAME_NUM1_REG/FRAME_NUM2_REG)

当主机发送 USB 数据包时，每个帧的开始 (SOF) 包包含一个帧号。USB 设备控制器捕获到这个帧号并自动加载到此寄存器中。

寄存器	地址	R/W	描述	复位值
FRAME_NUM1_REG	0x52000170(L) 0x52000173(B)	R/W (字节)	帧号低字节寄存器	0x00

FRAME_NUM_REG	位	MCU	USB	描述	初始状态
FRAME_NUM1	[7:0]	R	W	帧号低字节值	00

寄存器	地址	R/W	描述	复位值
FRAME_NUM2_REG	0x52000174(L) 0x52000177(B)	R/W (字节)	帧号高字节寄存器	0x00

FRAME_NUM_REG	位	MCU	USB	描述	初始状态
FRAME_NUM2	[7:0]	R	W	帧号高字节的值	00

标记寄存器 (INDEX_REG)

标记寄存器是用于有效地指示某个端点寄存器。MCU 可以在内核使用 INDEX 寄存器内部访问端点的端点寄存器(MAXP_REG, IN_CSR1_REG, IN_CSR2_REG, OUT_CSR1_REG, OUT_CSR2_REG, OUT_FIFO_CNT1_REG 和 OUT_FIFO_CNT2_REG)。

寄存器	地址	R/W	描述	复位值
INDEX_REG	0x52000178(L) 0x5200017B(B)	R/W (字节)	寄存器的标记寄存器	0x00

INDEX_REG	位	MCU	USB	描述	初始状态
INDEX	[7:0]	R/W	R	指示当前端点	00

最大数据包寄存器 (MAXP_REG)

寄存器	地址	R/W	描述	复位值
MAXP_REG	0x52000180(L) 0x52000183(B)	R/W (字节)	端点最大包寄存器	0x01

MAXP_REG	位	MCU	USB	描述	初始状态
MAXP	[3:0]	R/W	R	0000 : 保留 0001 : MAXP = 8 字节 0100 : MAXP = 32 字节 0100 : MAXP = 32 字节 0010 : MAXP = 16 字节 1000 : MAXP = 64 字节 EP0 时，推荐 MAXP=8。 EP1 至 4，推荐 MAXP=64。并且如果 MAXP=64，将会自动使能双数据包模式。	0001

端点 0 控制状态寄存器 (EP0_CSR)

此寄存器包含端点 0 的控制和状态位。当一个控制事件与 IN 和 OUT 令牌相相关联时，只有一个 CSR 寄存器映射到 IN CSR1 寄存器。(共用 IN1_CSR 并且可以通过写标记寄存器为“0”访问和读/写 IN1_CSR)

寄存器	地址	R/W	描述	复位值
EP0_CSR	0x52000184(L) 0x52000187(B)	R/W (字节)	端点 0 状态寄存器	0x00

EP0_CSR	位	MCU	USB	描述	初始状态
SERVICED_SET_UP_END	[7]	W	清零	MCU 应当写“1”到此位来清除 SETUP_END	0
SERVICED_OUT_PKT_RDY	[6]	W	清零	MCU 应当写“1”到此位来清除 OUT_PKT_RDY	0
SEND_STALL	[5]	R/W	清零	如果解码到一个无效的令牌时，MCU 应当在清除 OUT_PKT_RDY 的同时写“1”到此位。 0 = 完成 STALL 状态 1 = USB 发出一个 STALL 并且与当前控制传输握手	0
SETUP_END	[4]	R	置位	当 DATA_END 置位前控制传输结束时由 USB 置位。 当 USB 置位此位时，产生一个中断给 MCU。当发生这种情况时，USB 刷新 FIFO 并作废 MCU 访问 FIFO。	0
DATA_END	[3]	置位	清零	以下条件下由 MCU 置位： 1. 加载最后数据包到 FIFO 中后，同时 IN_PKT_RDY 被置位 2. 当其在卸载最后数据包后清除了 OUT_PKT_RDY 3. 零长度数据相	0
DATA_END	[2]	清零	置位	如果控制事件由于违反协议而停止由 USB 置位。当此位置位时产生一个中断。MCU 应当写“0”来清除此位	0
IN_PKT_RDY	[1]	置位	清零	在写入数据包到 EP0 FIFO 中后由 MCU 置位。一旦包被成功送到主机则 USB 清除此位。当 USB 清除此位则产生一个中断，为 MCU 加载下个包。零长度数据相，MCU 同时置位 DATA_END。	0
OUT_PKT_RDY	[0]	R	置位	一旦一个有效令牌写入到 FIFO 中由 USB 置位。当 USB 置位此位时产生一个中断。MCU 通过写“1”到 SERVICED_OUT_PKT_RDY 位来清除此位。	0

端点输入控制状态寄存器 (IN_CSR1_REG/IN_CSR2_REG)

此寄存器包含端点 0 的控制和状态位。当一个控制事件与 IN 和 OUT 令牌相相关联时，只有一个 CSR 寄存器映射到 IN CSR1 寄存器。(共用 IN1_CSR 并且可以通过写标记寄存器为“0”访问和读/写 IN1_CSR)

寄存器	地址	R/W	描述	复位值
IN_CSR1_REG	0x52000184(L) 0x52000187(B)	R/W (字节)	端点输入状态寄存器 1	0x00

IN_CSR1_REG	位	MCU	USB	描述	初始状态
保留	[7]	-	-	-	-
CLR_DATA_TOGGLE	[6]	R/W	R /清零	建立步骤中使用 0 : DATA0 和 DATA1 的交替 1 : 清除数据切换位并且包中的 PID 将持续 DATA0	0
SENT_STALL	[5]	R /清零	置位	当一个 IN 令牌发出 STALL 握手时由 USB 置位，在 MCU 置位 SEND_STALL 位后启动 STALL 握手。当 USB 发出 STALL 握手，IN_PKT_RDY 被清除。	0
SEND_STALL	[4]	W/R	R	0 : MCU 清除此位来完成 STALL 状态 1 : MCU 发送 STALL 握手给 USB	0
FIFO_FLUSH	[3]	R/W	清零	如果打算刷新输入相关 FIFO 中的包由 MCU 置位。当此位由 USB 置位。当发生这种情况时中断 MCU。如果一个令牌正在处理，USB 一直等待直到在 FIFO 刷新之前完成了传输。如果加载两个包到 FIFO 中，只有第一个包（打算发送该包给主机）被刷新，并且对应的 IN_PKT_RDY 位也被清零。	0
保留	[2:1]	-	-	-	-
IN_PKT_RDY	[0]	R /置位	清零	在写一个数据包到 FIFO 中后由 MCU 置位。一旦包被成功的发送到主机 USB 清除此位。当 USB 清除此位则产生中断，故 MCU 可以加载下一个包。如果 MCU 置位 SEND_STALL 位，则不能置位此位。	0

端点输入控制状态寄存器 (IN_CSR1_REG/IN_CSR2_REG) (续)

寄存器	地址	R/W	描述	复位值
IN_CSR2_REG	0x52000188(L) 0x5200018B(B)	R/W (字节)	端点输入状态寄存器 2	0x20

IN_CSR2_REG	位	MCU	USB	描述	初始状态
AUTO_SET	[7]	R/W	R	如果置位，每当 MCU 写 MAXP 数据，IN_PKT_RDY 将无须任何 MCU 的干预自动由内核置位。如果 MCU 的写入少于 MAXP 数据，IN_PKT_RDY 位必须由 MCU 置位。	0
ISO	[6]	R/W	R	只用于可编程传输类型的端点 0：配制端点为批量模式 1：保留	0
MODE_IN	[5]	R/W	R	只用于可编程传输类型的端点 0：配制端点方向为 OUT 1：配制端点方向为 IN	1
IN_DMA_INT_EN	[4]	R/W	R	决定当 IN_PKT_RDY 状态发生时是否发出中断。此位只用于 DMA 模式。 0：中断使能 1：中断禁止	0
保留	[3:0]	-	-	-	-

端点输出控制状态寄存器 (OUT_CSR1_REG/OUT_CSR2_REG)

寄存器	地址	R/W	描述	复位值
OUT_CSR1_REG	0x52000190(L) 0x52000193(B)	R/W (字节)	端点输出状态寄存器 1	0x00

OUT_CSR1_REG	位	MCU	USB	描述	初始状态
CLR_DATA_TOGGLE	[7]	R/W	清零	当 MCU 写 1 到此位时，复位数据切换序列位到 DATA0	0
SENT_STALL	[6]	R /清零	置位	当 OUT 令牌带 STALL 握手的结束则由 USB 置位。如果其发送多于 OUT 令牌的 MAXP 数据则 USB 发出一个 STALL 握手给主机。	0
SEND_STALL	[5]	R/W	R	0 : MCU 清除此位来结束 STALL 状态握手，IN PKT RDY 被清除 1 : MCU 发送 STALL 握手给 USB。MCU 清除此位来结束 STALL 状态握手，IN PKT RDY 被清除	0
FIFO_FLUSH	[4]	R/W	清零	MCU 写入 1 来刷新 FIFO。此位只在 OUT_PKT_RDY (D0) 置位时才能被置位。由 MCU 卸载的包将被刷新。	0
保留	[3:1]	-	-	-	0
OUT_PKT_RDY	[0]	R /清零	置位	在其加载一个数据包到 FIFO 中后由 USB 置位。一旦 MCU 从 FIFO 中读取包，此位应该被 MCU 清除 (写 "0")	0

寄存器	地址	R/W	描述	复位值
OUT_CSR2_REG	0x52000194(L) 0x52000197(B)	R/W (字节)	端点输出状态寄存器 2	0x00

OUT_CSR2_REG	位	MCU	USB	描述	初始状态
AUTO_CLR	[7]	R/W	R	如果 MCU 置位，每当 MCU 从 FIFO 读取数据，OUT_PKT_RDY 将无须任何 MCU 的干预自动由逻辑清除。	0
ISO	[6]	R/W	R	决定端点传输类型 0 : 配置端点为批量传输 1 : 保留	0
OUT_DMA_INT_MASK	[5]	R/W	R	决定是否发出中断 0 : 中断使能 1 : 中断禁止	0

端点输出写计数寄存器 (OUT_FIFO_CNT1_REG/OUT_FIFO_CNT2_REG)

此寄存器保存着包的字节数，该数由 MCU 卸载。

寄存器	地址	R/W	描述	复位值
OUT_FIFO_CNT1_REG	0x52000198(L) 0x5200019B(B)	R/W (字节)	端点输出计数寄存器 1	0x00

OUT_FIFO_CNT1_REG	位	MCU	USB	描述	初始状态
OUT_CNT_LOW	[7:0]	R	W	写计数的低字节	0x00

寄存器	地址	R/W	描述	复位值
OUT_FIFO_CNT2_REG	0x5200019C(L) 0x5200019F(B)	R/W (字节)	端点输出计数寄存器 2	0x00

OUT_FIFO_CNT2_REG	位	MCU	USB	描述	初始状态
OUT_CNT_HIGH	[7:0]	R	W	写计数的高字节。通常 OUT_CNT_HIGH 可能总为 0	0x00

端点 FIFO 寄存器 (EPn_FIFO_REG)

EPn_FIFO_REG 使能 MCU 访问 EPn FIFO。

寄存器	地址	R/W	描述	复位值
EP0_FIFO	0x520001C0(L) 0x520001C3(B)	R/W (字节)	端点 0 FIFO 寄存器	0XX
EP1_FIFO	0x520001C4(L) 0x520001C7(B)	R/W (字节)	端点 1 FIFO 寄存器	0XX
EP2_FIFO	0x520001C8(L) 0x520001CB(B)	R/W (字节)	端点 2 FIFO 寄存器	0XX
EP3_FIFO	0x520001CC(L) 0x520001CF(B)	R/W (字节)	端点 3 FIFO 寄存器	0XX
EP4_FIFO	0x520001D0(L) 0x520001D3(B)	R/W (字节)	端点 4 FIFO 寄存器	0XX

EPn_FIFO	位	MCU	USB	描述	初始状态
FIFO_DATA	[7:0]	R/W	R/W	FIFO 数值	0x00

DMA 接口控制寄存器 (EPn_DMA_CON)

EPn_FIFO_REG 使能 MCU 访问 EPn FIFO。

寄存器	地址	R/W (字节)	描述	复位值
EP1_DMA_CON	0x52000200(L) 0x52000203(B)	R/W (字节)	EP1 DMA 接口控制寄存器	0x00
EP2_DMA_CON	0x52000218(L) 0x5200021B(B)	R/W (字节)	EP2 DMA 接口控制寄存器	0x00
EP3_DMA_CON	0x52000240(L) 0x52000243(B)	R/W (字节)	EP3 DMA 接口控制寄存器	0x00
EP4_DMA_CON	0x52000258(L) 0x5200025B(B)	R/W (字节)	EP4 DMA 接口控制寄存器	0x00

EPn_DMA_CON	位	MCU	USB	描述	初始状态
RUN_OB	[7]	R/W	W	读) DMA 运行观察 0 : DMA 为停止 1 : DMA 正在运行 写) 忽略 EPn_DMA_TTC_n 寄存器 0 : 如果 EPn_DMA_TTC_n 到达 0 将停止 DMA 请求 1 : 即使 EPn_DMA_TTC_n 到达 0 也继续 DMA 请求	0
STATE	[6:4]	R	W	DMA 状态监视	0
DEMAND_MODE	[3]	R/W	R	DMA 需求模式使能位 0 : 需求模式禁止 1 : 需求模式使能	0
OUT_RUN_OB/ OUT_DMA_RUN	[2]	R/W	R/W	功能上分离为写和读操作 写操作 : '0' = 停止 '1' = 运行 读操作 : OUT DMA 运行操作	0
IN_DMA_RUN	[1]	R/W	R	开始 DMA 操作 0 = 停止 1 = 运行	0
DMA_MODE_EN	[0]	R/W	R /清零	设置 DMA。如果 RUN_OB 已经被写为 0 并且 EPn_DMA_TTC_n 到达 0 , DMA_MODE_EN 位将由 USB 清除。 0 = 中断模式 1 = DMA 模式	0

DMA 单元计数器寄存器 (EPN_DMA_UNIT)

此寄存器在需求模式中有效。其它模式中，此寄存器的值必须设置为‘0x01’。

寄存器	地址	R/W	描述	复位值
EP1_DMA_UNIT	0x52000204(L) 0x52000207(B)	R/W (字节)	EP1 DMA 传输单元计数器基本寄存器	0x00
EP2_DMA_UNIT	0x5200021C(L) 0x5200021F(B)	R/W (字节)	EP2 DMA 传输单元计数器基本寄存器	0x00
EP3_DMA_UNIT	0x52000244(L) 0x52000247(B)	R/W (字节)	EP3 DMA 传输单元计数器基本寄存器	0x00
EP4_DMA_UNIT	0x5200025C(L) 0x5200025F(B)	R/W (字节)	EP4 DMA 传输单元计数器基本寄存器	0x00

DMA_UNIT	位	MCU	USB	描述	初始状态
EPn_UNIT_CNT	[7:0]	R/W	R	EP DMA 传输单元计数器值	0x00

DMA FIFO 计数器寄存器 (EPN_DMA_FIFO)

此寄存器含有要通过 DMA 传输的 FIFO 中的字节大小的值。在使能了 OUT_DMA_RUN 的情况下，OUT FIFO 写计数寄存器中的值将被自动加载到此寄存器中。DMA 模式情况下，MCU 应该通过软件合理设置此值。

寄存器	地址	R/W	描述	复位值
EP1_DMA_FIFO	0x52000208(L) 0x5200020B(B)	R/W (字节)	EP1 DMA 传输 FIFO 计数器基本寄存器	0x00
EP2_DMA_FIFO	0x52000220(L) 0x52000223(B)	R/W (字节)	EP2 DMA 传输 FIFO 计数器基本寄存器	0x00
EP3_DMA_FIFO	0x52000248(L) 0x5200024B(B)	R/W (字节)	EP3 DMA 传输 FIFO 计数器基本寄存器	0x00
EP4_DMA_FIFO	0x52000260(L) 0x52000263(B)	R/W (字节)	EP4 DMA 传输 FIFO 计数器基本寄存器	0x00

DMA_FIFO	位	MCU	USB	描述	初始状态
EPn_FIFO_CNT	[7:0]	R/W	R	EP DMA 传输 FIFO 计数器值	0x00

DMA 总计传输计数器寄存器 (EPn_DMA_TTC_L , M , H)

此寄存器包含使用 DMA 传输的字节数 (总计 20 位计数器)

寄存器	地址	R/W	描述	复位值
EP1_DMA_TTC_L	0x5200020C(L) 0x5200020F(B)	R/W (字节)	EP1 DMA 总计传输计数器 (低字节)	0x00
EP1_DMA_TTC_M	0x52000210(L) 0x52000213(B)	R/W (字节)	EP1 DMA 总计传输计数器 (中字节)	0x00
EP1_DMA_TTC_H	0x52000214(L) 0x52000217(B)	R/W (字节)	EP1 DMA 总计传输计数器 (高字节)	0x00
EP2_DMA_TTC_L	0x52000224(L) 0x52000227(B)	R/W (字节)	EP2 DMA 总计传输计数器 (低字节)	0x00
EP2_DMA_TTC_M	0x52000228(L) 0x5200022B(B)	R/W (字节)	EP2 DMA 总计传输计数器 (中字节)	0x00
EP2_DMA_TTC_H	0x5200022C(L) 0x5200022F(B)	R/W (字节)	EP2 DMA 总计传输计数器 (高字节)	0x00
EP3_DMA_TTC_L	0x5200024C(L) 0x5200024F(B)	R/W (字节)	EP3 DMA 总计传输计数器 (低字节)	0x00
EP3_DMA_TTC_M	0x52000250(L) 0x52000253(B)	R/W (字节)	EP3 DMA 总计传输计数器 (中字节)	0x00
EP3_DMA_TTC_H	0x52000254(L) 0x52000257(B)	R/W (字节)	EP3 DMA 总计传输计数器 (高字节)	0x00
EP4_DMA_TTC_L	0x52000264(L) 0x52000267(B)	R/W (字节)	EP4 DMA 总计传输计数器 (低字节)	0x00
EP4_DMA_TTC_M	0x52000268(L) 0x5200026B(B)	R/W (字节)	EP4 DMA 总计传输计数器 (中字节)	0x00
EP4_DMA_TTC_H	0x5200026C(L) 0x5200026F(B)	R/W (字节)	EP4 DMA 总计传输计数器 (高字节)	0x00

DMA_TX	位	MCU	USB	描述	初始状态
EPn_TTC_L	[7:0]	R/W	R	DMA 总计传输计数值 (低字节)	0x00
EPn_TTC_M	[7:0]	R/W	R	DMA 总计传输计数值 (中字节)	0x00
EPn_TTC_H	[3:0]	R/W	R	DMA 总计传输计数值 (高字节)	0x00

14 中断控制器

概述

S3C2440A 中的中断控制器接受来自 60 个中断源的请求。提供这些中断源的是内部外设，如 DMA 控制器、UART、IIC 等等。在这些中断源中，UARTn、AC97 和 EINTn 中断对于中断控制器而言是“或”关系。

当从内部外设和外部中断请求引脚收到多个中断请求时，中断控制器在仲裁步骤后请求 ARM920T 内核的 FIQ 或 IRQ。

仲裁步骤由硬件优先级逻辑决定并且写入结果到帮助用户通告是各种中断源中的哪个中断发生了的中断挂起寄存器中。

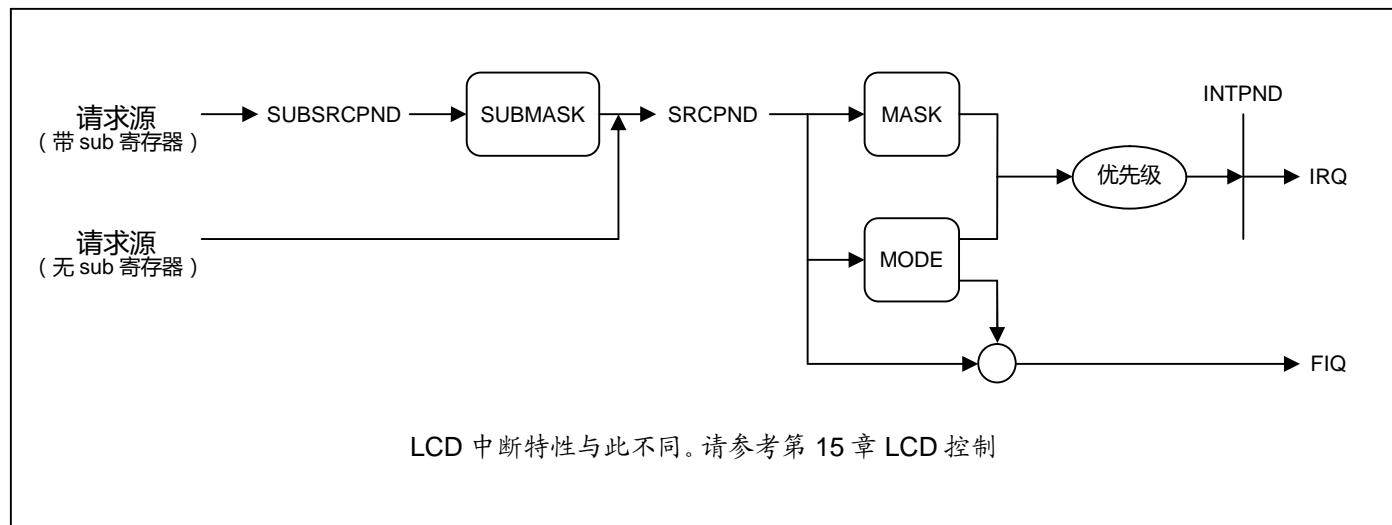


图 14-1. 中断处理框图

中断控制器操作

程序状态寄存器 (PSR) 的 F 位和 I 位

如果 ARM920T CPU 中的 PSR 的 F 位被置位为 1 , CPU 不会接受来自中断控制器的快中断请求 (FIQ)。同样的如果 PSR 的 I 位被置位为 1 , CPU 不会接受来自中断控制器的中断请求 (IRQ)。因此 , 中断控制器可以通过清除 PSR 的 F 位和 I 位为 0 并且设置 INTMSK 的相应位为 0 来接收中断。

中断模式

ARM920T 有两种中断模式的类型 : FIQ 或 IRQ。所有中断源在中断请求时决定使用哪种类型。

中断挂起寄存器

S3C2440A 有两个中断挂起寄存器 : 源挂起寄存器 (SRCPND) 和中断挂起寄存器 (INTPND)。这些挂起寄存器表明一个中断请求是否为挂起。当中断源请求中断服务 , SRCPND 寄存器的相应位被置位为 1 , 并且同时在仲裁步骤后 INTPND 寄存器仅有 1 位自动置位为 1。如果屏蔽了中断 , 则 SRCPND 寄存器的相应位被置位为 1。这并不会引起 INTPND 寄存器的位的改变。当 INTPND 寄存器的挂起位为置位 , 每当 I 标志或 F 标志被清除为 0 中断服务程序将开始。SRCPND 和 INTPND 寄存器可以被读取和写入 , 因此服务程序必须首先通过写 1 到 SRCPND 寄存器的相应位来清除挂起状态并且通过相同方法来清除 INTPND 寄存器中挂起状态。

中断屏蔽寄存器

此寄存器表明如果中断相应的屏蔽位被置位为 1 则禁止该中断。如果某个 INTMSK 的中断屏蔽位为 0 , 将正常服务中断。如果 INTMSK 的中断屏蔽位为 1 并且产生了中断 , 将置位源挂起位。

中断源

中断控制器支持 60 个中断源，如下表所示。

源	描述	仲裁组
INT_ADC	ADC EOC 和触屏中断 (INT_ADC_S/INT_TC)	ARB5
INT_RTC	RTC 闹钟中断	ARB5
INT_SPI1	SPI1 中断	ARB5
INT_UART0	UART0 中断 (ERR、RXD 和 TXD)	ARB5
INT_IIC	IIC 中断	ARB4
INT_USBH	USB 主机中断	ARB4
INT_USBD	USB 设备中断	ARB4
INT_NFCON	Nand Flash 控制中断	ARB4
INT_UART1	UART1 中断 (ERR、RXD 和 TXD)	ARB4
INT_SPI0	SPI0 中断	ARB4
INT_SDI	SDI 中断	ARB3
INT_DMA3	DMA 通道 3 中断	ARB3
INT_DMA2	DMA 通道 2 中断	ARB3
INT_DMA1	DMA 通道 1 中断	ARB3
INT_DMA0	DMA 通道 0 中断	ARB3
INT_LCD	LCD 中断 (INT_FrSyn 和 INT_FiCnt)	ARB3
INT_UART2	UART2 中断 (ERR、RXD 和 TXD)	ARB2
INT_TIMER4	定时器 4 中断	ARB2
INT_TIMER3	定时器 3 中断	ARB2
INT_TIMER2	定时器 2 中断	ARB2
INT_TIMER1	定时器 1 中断	ARB2
INT_TIMER0	定时器 0 中断	ARB2
INT_WDT_AC97	看门狗定时器中断 (INT_WDT、INT_AC97)	ARB1
INT_TICK	RTC 时钟滴答中断	ARB1
nBATT_FLT	电池故障中断	ARB1
INT_CAM	摄像头接口 (INT_CAM_C、INT_CAM_P)	ARB1
EINT8_23	外部中断 8 至 23	ARB1
EINT4_7	外部中断 4 至 7	ARB1
EINT3	外部中断 3	ARB0
EINT2	外部中断 2	ARB0
EINT1	外部中断 1	ARB0
EINT0	外部中断 0	ARB0

中断次级源

次级源	描述	源
INT_AC97	AC97 中断	INT_WDT_AC97
INT_WDT	看门狗中断	INT_WDT_AC97
INT_CAM_P	摄像头接口中 P 端口捕获中断	INT_CAM
INT_CAM_C	摄像头接口中 C 端口捕获中断	INT_CAM
INT_ADC_S	ADC 中断	INT_ADC
INT_TC	触摸屏中断 (笔起/落)	INT_ADC
INT_ERR2	UART2 错误中断	INT_UART2
INT_TXD2	UART2 发送中断	INT_UART2
INT_RXD2	UART2 接收中断	INT_UART2
INT_ERR1	UART1 错误中断	INT_UART1
INT_TXD1	UART1 发送中断	INT_UART1
INT_RXD1	UART1 接收中断	INT_UART1
INT_ERR0	UART0 错误中断	INT_UART0
INT_TXD0	UART0 发送中断	INT_UART0
INT_RXD0	UART0 接收中断	INT_UART0

中断优先级

每个仲裁器可以处理基于 1 位仲裁器模式控制 (ARB_MODE) 和选择控制信号 (ARB_SEL) 的 2 位的 6 个中断请求 , 如下 :

- 如果 ARB_SEL 位为 00b , 优先级顺序为 REQ0、REQ1、REQ2、REQ3、REQ4 和 REQ5。
- 如果 ARB_SEL 位为 01b , 优先级顺序为 REQ0、REQ2、REQ3、REQ4、REQ1 和 REQ5。
- 如果 ARB_SEL 位为 10b , 优先级顺序为 REQ0、REQ3、REQ4、REQ1、REQ2 和 REQ5。
- 如果 ARB_SEL 位为 11b , 优先级顺序为 REQ0、REQ4、REQ1、REQ2、REQ3 和 REQ5。

请注意仲裁器的 REQ0 的优先级总是最高并且 REQ5 的优先级总是最低。此外 , 通过改变 ARB_SEL 位 , 可以轮换 REQ1 到 REQ4 的顺序。

此处 , 如果 ARB_MODE 位被设置为 0 , ARB_SEL 位不能自动改变 , 这使得仲裁器操作在固定优先级模式中 (注意即使在此模式中 , 也不能通过手动改变 ARB_SEL 位来重新配制优先级) 。另一方面 , 如果 ARB_MODE 为 1 , ARB_SEL 位会被轮换方式而改变 , 例如如果 REQ1 被服务 , ARB_SEL 位被自动改为 01b 以便 REQ1 进入到最低的优先级。ARB_SEL 改变的详细结果如下 :

- 如果 REQ0 或 REQ5 被服务 , ARB_SEL 位不会改变
- 如果 REQ1 被服务 , ARB_SEL 位被改为 01b。
- 如果 REQ2 被服务 , ARB_SEL 位被改为 10b。
- 如果 REQ3 被服务 , ARB_SEL 位被改为 11b。
- 如果 REQ4 被服务 , ARB_SEL 位被改为 00b。

中断优先级发生模块

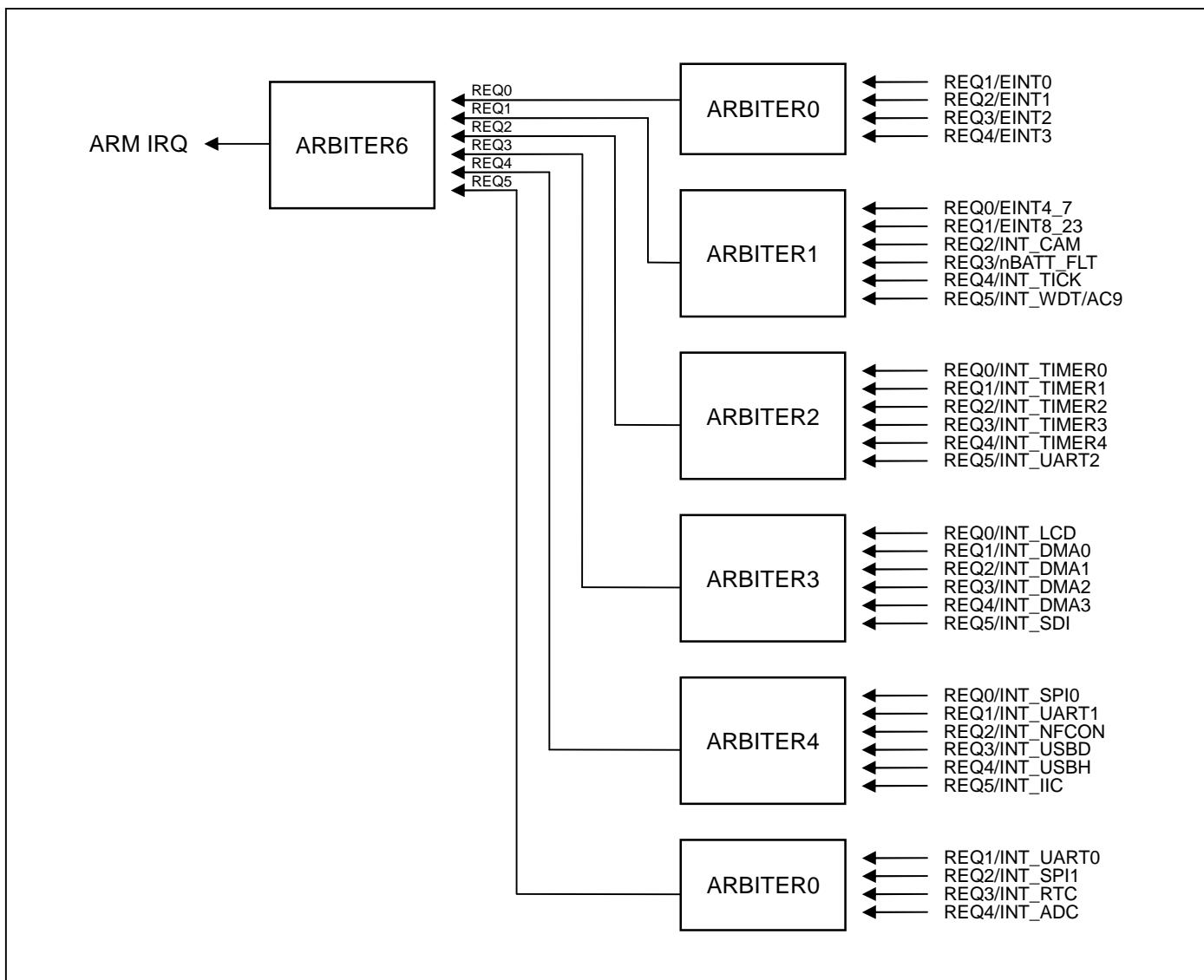


图 14-2. 优先级发生模块

中断控制器特殊寄存器

此处中断控制器中有 5 个控制寄存器：源挂起寄存器、中断模式寄存器、屏蔽寄存器、优先级寄存器和中断挂起寄存器。

所有来自中断源的中断请求首先被记录到源挂起寄存器中。基于中断模式寄存器，它们被分配到 2 个组中，包括快中断请求 (FIQ) 和中断请求 (IRQ)。IRQ 的多仲裁过程是基于优先级寄存器。

源挂起 (SRCPND) 寄存器

SRCPND 寄存器由 32 位组成，其每一位都涉及一个中断源。如果中断源产生了中断则相应的位被设置为 1 并且等待中断服务。因此此寄存器指示出是哪个中断源正在等待请求服务。注意 SRCPND 寄存器的每一位都是由中断源自动置位，其不顾 INTMASK 寄存器中的屏蔽位。另外 SRCPND 寄存器不受中断控制器的优先级逻辑的影响。

在指定中断源的中断服务程序中，必须通过清除 SRCPND 寄存器的相应位来正确的获得来自相同源的中断请求。如果从 ISR 中返回并且未清除相应位，则中断控制器的操作就好像其它中断请求已经从同一个源进入了。换句话说，如果 SRCPND 寄存器的指定位被设置为 1，其通常被认作一个有效中断请求正在等待服务。

清除相应位的时间依赖于用户的需要。如果希望收到来自相同来源的其它有效请求，则应该首先清除相应位，并且接着使能中断。

可以通过写入一个数据到此寄存器来清除 SRCPND 寄存器的指定位。其只清除那些数据中被设置为 1 的相应位置的 SRCPND 位。那些数据中被设置为 0 的相应位置的位保持不变。

寄存器	地址	R/W	描述	复位值
SRCPND	0X4A000000	R/W	指示中断请求状态 0 = 中断未被请求 1 = 中断源声明了中断请求	0x00000000

源挂起 (SRCPND) 寄存器 (续)

SRCPND	位	描述	初始状态
INT_ADC	[31]	0 = 未请求 1 = 请求	0
INT_RTC	[30]	0 = 未请求 1 = 请求	0
INT_SPI1	[29]	0 = 未请求 1 = 请求	0
INT_UART0	[28]	0 = 未请求 1 = 请求	0
INT_IIC	[27]	0 = 未请求 1 = 请求	0
INT_USBH	[26]	0 = 未请求 1 = 请求	0
INT_USBD	[25]	0 = 未请求 1 = 请求	0
INT_NFCON	[24]	0 = 未请求 1 = 请求	0
INT_UART1	[23]	0 = 未请求 1 = 请求	0
INT_SPI0	[22]	0 = 未请求 1 = 请求	0
INT_SDI	[21]	0 = 未请求 1 = 请求	0
INT_DMA3	[20]	0 = 未请求 1 = 请求	0
INT_DMA2	[19]	0 = 未请求 1 = 请求	0
INT_DMA1	[18]	0 = 未请求 1 = 请求	0
INT_DMA0	[17]	0 = 未请求 1 = 请求	0
INT_LCD	[16]	0 = 未请求 1 = 请求	0
INT_UART2	[15]	0 = 未请求 1 = 请求	0
INT_TIMER4	[14]	0 = 未请求 1 = 请求	0
INT_TIMER3	[13]	0 = 未请求 1 = 请求	0
INT_TIMER2	[12]	0 = 未请求 1 = 请求	0
INT_TIMER1	[11]	0 = 未请求 1 = 请求	0
INT_TIMER0	[10]	0 = 未请求 1 = 请求	0
INT_WDT_AC97	[9]	0 = 未请求 1 = 请求	0
INT_TICK	[8]	0 = 未请求 1 = 请求	0
nBATT_FLT	[7]	0 = 未请求 1 = 请求	0
INT_CAM	[6]	0 = 未请求 1 = 请求	0
EINT8_23	[5]	0 = 未请求 1 = 请求	0
EINT4_7	[4]	0 = 未请求 1 = 请求	0
EINT3	[3]	0 = 未请求 1 = 请求	0
EINT2	[2]	0 = 未请求 1 = 请求	0
EINT1	[1]	0 = 未请求 1 = 请求	0
EINT0	[0]	0 = 未请求 1 = 请求	0

中断模式 (INTMOD) 寄存器

此寄存器由 32 位组成，其每一位都涉及一个中断源。如果某个指定为被设置为 1，则在 FIQ (快中断) 模式中处理相应中断。否则则在 IRQ 模式中处理。

寄存器	地址	R/W	描述	复位值
INTMOD	0X4A000004	R/W	中断模式寄存器 0 = IRQ 模式 1 = FIQ 模式	0x00000000

注意：如果中断模式在 INTMOD 寄存器中设置为 FIQ 模式，则 FIQ 中断将不会影响 INTPND 和 INTOFFSET 寄存器。这种情况下，这 2 个寄存器只对 IRQ 中断源有效。

INTMOD	位	描述	初始状态
INT_ADC	[31]	0 = IRQ 1 = FIQ	0
INT_RTC	[30]	0 = IRQ 1 = FIQ	0
INT_SPI1	[29]	0 = IRQ 1 = FIQ	0
INT_UART0	[28]	0 = IRQ 1 = FIQ	0
INT_IIC	[27]	0 = IRQ 1 = FIQ	0
INT_USBH	[26]	0 = IRQ 1 = FIQ	0
INT_USBD	[25]	0 = IRQ 1 = FIQ	0
INT_NFCON	[24]	0 = IRQ 1 = FIQ	0
INT_UART1	[23]	0 = IRQ 1 = FIQ	0
INT_SPI0	[22]	0 = IRQ 1 = FIQ	0
INT_SDI	[21]	0 = IRQ 1 = FIQ	0
INT_DMA3	[20]	0 = IRQ 1 = FIQ	0
INT_DMA2	[19]	0 = IRQ 1 = FIQ	0
INT_DMA1	[18]	0 = IRQ 1 = FIQ	0
INT_DMA0	[17]	0 = IRQ 1 = FIQ	0
INT_LCD	[16]	0 = IRQ 1 = FIQ	0
INT_UART2	[15]	0 = IRQ 1 = FIQ	0
INT_TIMER4	[14]	0 = IRQ 1 = FIQ	0
INT_TIMER3	[13]	0 = IRQ 1 = FIQ	0
INT_TIMER2	[12]	0 = IRQ 1 = FIQ	0
INT_TIMER1	[11]	0 = IRQ 1 = FIQ	0
INT_TIMER0	[10]	0 = IRQ 1 = FIQ	0
INT_WDT_AC97	[9]	0 = IRQ 1 = FIQ	0
INT_TICK	[8]	0 = IRQ 1 = FIQ	0
nBATT_FLT	[7]	0 = IRQ 1 = FIQ	0
INT_CAM	[6]	0 = IRQ 1 = FIQ	0
EINT8_23	[5]	0 = IRQ 1 = FIQ	0
EINT4_7	[4]	0 = IRQ 1 = FIQ	0
EINT3	[3]	0 = IRQ 1 = FIQ	0
EINT2	[2]	0 = IRQ 1 = FIQ	0
EINT1	[1]	0 = IRQ 1 = FIQ	0
EINT0	[0]	0 = IRQ 1 = FIQ	0

中断屏蔽 (INTMSK) 寄存器

此寄存器由 32 位组成，其每一位都涉及一个中断源。如果某个指定为被设置为 1，则 CPU 不会去服务来自相应中断源（请注意即使在这种情况下，SRCPND 寄存器的相应位也设置为 1）的中断请求。如果屏蔽位为 0，则可以服务中断请求。

寄存器	地址	R/W	描述	复位值
INTMSK	0X4A000008	R/W	决定屏蔽哪个中断源。被屏蔽的中断源将不会服务 0 = 中断服务可用 1 = 屏蔽中断服务	0xFFFFFFFF

INTMSK	位	描述	初始状态
INT_ADC	[31]	0 = 可服务 1 = 屏蔽	1
INT_RTC	[30]	0 = 可服务 1 = 屏蔽	1
INT_SPI1	[29]	0 = 可服务 1 = 屏蔽	1
INT_UART0	[28]	0 = 可服务 1 = 屏蔽	1
INT_IIC	[27]	0 = 可服务 1 = 屏蔽	1
INT_USBH	[26]	0 = 可服务 1 = 屏蔽	1
INT_USBD	[25]	0 = 可服务 1 = 屏蔽	1
INT_NFCON	[24]	0 = 可服务 1 = 屏蔽	1
INT_UART1	[23]	0 = 可服务 1 = 屏蔽	1
INT_SPI0	[22]	0 = 可服务 1 = 屏蔽	1
INT_SDI	[21]	0 = 可服务 1 = 屏蔽	1
INT_DMA3	[20]	0 = 可服务 1 = 屏蔽	1
INT_DMA2	[19]	0 = 可服务 1 = 屏蔽	1
INT_DMA1	[18]	0 = 可服务 1 = 屏蔽	1
INT_DMA0	[17]	0 = 可服务 1 = 屏蔽	1
INT_LCD	[16]	0 = 可服务 1 = 屏蔽	1
INT_UART2	[15]	0 = 可服务 1 = 屏蔽	1
INT_TIMER4	[14]	0 = 可服务 1 = 屏蔽	1
INT_TIMER3	[13]	0 = 可服务 1 = 屏蔽	1
INT_TIMER2	[12]	0 = 可服务 1 = 屏蔽	1
INT_TIMER1	[11]	0 = 可服务 1 = 屏蔽	1
INT_TIMER0	[10]	0 = 可服务 1 = 屏蔽	1
INT_WDT_AC97	[9]	0 = 可服务 1 = 屏蔽	1
INT_TICK	[8]	0 = 可服务 1 = 屏蔽	1
nBATT_FLT	[7]	0 = 可服务 1 = 屏蔽	1
INT_CAM	[6]	0 = 可服务 1 = 屏蔽	1
EINT8_23	[5]	0 = 可服务 1 = 屏蔽	1
EINT4_7	[4]	0 = 可服务 1 = 屏蔽	1
EINT3	[3]	0 = 可服务 1 = 屏蔽	1
EINT2	[2]	0 = 可服务 1 = 屏蔽	1
EINT1	[1]	0 = 可服务 1 = 屏蔽	1
EINT0	[0]	0 = 可服务 1 = 屏蔽	1

优先级寄存器 (PRIORITY)

寄存器	地址	R/W	描述	复位值
PRIORITY	0X4A00000C	R/W	IRQ 优先级控制寄存器	0x7F

PRIORITY	位	描述	初始状态
ARB_SEL6	[20:19]	仲裁器组 6 优先级顺序设置 00 = REQ 0-1-2-3-4-5 10 = REQ 0-3-4-1-2-5	00 01 = REQ 0-2-3-4-1-5 11 = REQ 0-4-1-2-3-5
ARB_SEL5	[18:17]	仲裁器组 5 优先级顺序设置 00 = REQ 0-1-2-3-4-5 10 = REQ 0-3-4-1-2-5	00 01 = REQ 0-2-3-4-1-5 11 = REQ 0-4-1-2-3-5
ARB_SEL4	[16:15]	仲裁器组 4 优先级顺序设置 00 = REQ 0-1-2-3-4-5 10 = REQ 0-3-4-1-2-5	00 01 = REQ 0-2-3-4-1-5 11 = REQ 0-4-1-2-3-5
ARB_SEL3	[14:13]	仲裁器组 3 优先级顺序设置 00 = REQ 0-1-2-3-4-5 10 = REQ 0-3-4-1-2-5	00 01 = REQ 0-2-3-4-1-5 11 = REQ 0-4-1-2-3-5
ARB_SEL2	[12:11]	仲裁器组 2 优先级顺序设置 00 = REQ 0-1-2-3-4-5 10 = REQ 0-3-4-1-2-5	00 01 = REQ 0-2-3-4-1-5 11 = REQ 0-4-1-2-3-5
ARB_SEL1	[10:9]	仲裁器组 1 优先级顺序设置 00 = REQ 0-1-2-3-4-5 10 = REQ 0-3-4-1-2-5	00 01 = REQ 0-2-3-4-1-5 11 = REQ 0-4-1-2-3-5
ARB_SEL0	[8:7]	仲裁器组 0 优先级顺序设置 00 = REQ 0-1-2-3-4-5 10 = REQ 0-3-4-1-2-5	00 01 = REQ 0-2-3-4-1-5 11 = REQ 0-4-1-2-3-5
ARB_MODE6	[6]	仲裁器组 6 优先级轮换使能 0 = 优先级不轮换	1 1 = 优先级轮换使能
ARB_MODE5	[5]	仲裁器组 5 优先级轮换使能 0 = 优先级不轮换	1 1 = 优先级轮换使能
ARB_MODE4	[4]	仲裁器组 4 优先级轮换使能 0 = 优先级不轮换	1 1 = 优先级轮换使能
ARB_MODE3	[3]	仲裁器组 3 优先级轮换使能 0 = 优先级不轮换	1 1 = 优先级轮换使能
ARB_MODE2	[2]	仲裁器组 2 优先级轮换使能 0 = 优先级不轮换	1 1 = 优先级轮换使能
ARB_MODE1	[1]	仲裁器组 1 优先级轮换使能 0 = 优先级不轮换	1 1 = 优先级轮换使能
ARB_MODE0	[0]	仲裁器组 0 优先级轮换使能 0 = 优先级不轮换	1 1 = 优先级轮换使能

中断挂起 (INTPND) 寄存器

中断挂起寄存器中 32 位的每一位都表明了是否相应未屏蔽并且正在等待中断服务的中断请求具有最高的优先级。当 INTPND 寄存器在优先级逻辑后被定位了，只有 1 位可以设置为 1 并且产生中断请求 IRQ 给 CPU。IRQ 的中断服务程序中可以读取此寄存器来决定服务 32 个中断源的那个源。

就如 SRCPND 寄存器，必须在中断服务程序中清除了 SRCPND 寄存器后清除此寄存器。可以通过写入数据到此寄存器中来清除 INTPND 寄存器的指定位。只会清除数据中设置为 1 的相应 INTPND 寄存器位的位置。数据中设置为 0 的相应位的位置则保持不变。

寄存器	地址	R/W	描述	复位值
INTPND	0X4A000010	R/W	指示中断请求状态 0 = 未请求中断 1 = 中断源已声明中断请求	0x7F

注意：如果 FIQ 模式中断发生，则 INTPND 的相应位将不会打开因为 INTPND 寄存器只对 IRQ 模式中断可见。

INTPND	位	描述	初始状态
INT_ADC	[31]	0 = 未请求 1 = 请求	0
INT_RTC	[30]	0 = 未请求 1 = 请求	0
INT_SPI1	[29]	0 = 未请求 1 = 请求	0
INT_UART0	[28]	0 = 未请求 1 = 请求	0
INT_IIC	[27]	0 = 未请求 1 = 请求	0
INT_USBH	[26]	0 = 未请求 1 = 请求	0
INT_USBD	[25]	0 = 未请求 1 = 请求	0
INT_NFCON	[24]	0 = 未请求 1 = 请求	0
INT_UART1	[23]	0 = 未请求 1 = 请求	0
INT_SPI0	[22]	0 = 未请求 1 = 请求	0
INT_SDI	[21]	0 = 未请求 1 = 请求	0
INT_DMA3	[20]	0 = 未请求 1 = 请求	0
INT_DMA2	[19]	0 = 未请求 1 = 请求	0
INT_DMA1	[18]	0 = 未请求 1 = 请求	0
INT_DMA0	[17]	0 = 未请求 1 = 请求	0
INT_LCD	[16]	0 = 未请求 1 = 请求	0
INT_UART2	[15]	0 = 未请求 1 = 请求	0
INT_TIMER4	[14]	0 = 未请求 1 = 请求	0
INT_TIMER3	[13]	0 = 未请求 1 = 请求	0
INT_TIMER2	[12]	0 = 未请求 1 = 请求	0
INT_TIMER1	[11]	0 = 未请求 1 = 请求	0
INT_TIMER0	[10]	0 = 未请求 1 = 请求	0
INT_WDT_AC97	[9]	0 = 未请求 1 = 请求	0
INT_TICK	[8]	0 = 未请求 1 = 请求	0
nBATT_FLT	[7]	0 = 未请求 1 = 请求	0
INT_CAM	[6]	0 = 未请求 1 = 请求	0
EINT8_23	[5]	0 = 未请求 1 = 请求	0
EINT4_7	[4]	0 = 未请求 1 = 请求	0
EINT3	[3]	0 = 未请求 1 = 请求	0
EINT2	[2]	0 = 未请求 1 = 请求	0
EINT1	[1]	0 = 未请求 1 = 请求	0
EINT0	[0]	0 = 未请求 1 = 请求	0

中断偏移 (INTOFFSET) 寄存器

中断偏移寄存器中的值表明了是哪个 IRQ 模式的中断请求在 INTPND 寄存器中。此位可以通过清楚 SRCPND 和 INTPND 自动清除。

寄存器	地址	R/W	描述	复位值
INTOFFSET	0x4A000014	R	指示 IRQ 中断请求源	0x00000000

中断源	偏移量	中断源	偏移量
INT_ADC	31	INT_UART2	15
INT_RTC	30	INT_TIMER4	14
INT_SPI1	29	INT_TIMER3	13
INT_UART0	28	INT_TIMER2	12
INT_IIC	27	INT_TIMER1	11
INT_USBH	26	INT_TIMER0	10
INT_USBD	25	INT_WDT_AC97	9
INT_NFCON	24	INT_TICK	8
INT_UART1	23	nBATT_FLT	7
INT_SPI0	22	INT_CAM	6
INT_SDI	21	EINT8_23	5
INT_DMA3	20	EINT4_7	4
INT_DMA2	19	EINT3	3
INT_DMA1	18	EINT2	2
INT_DMA0	17	EINT1	1
INT_LCD	16	EINT0	0

注意：FIQ 模式中断不会影响 INTOFFSET 寄存器因为该寄存器只对 IRQ 模式中断有效。

次级源挂起 (SUBSRCPND) 寄存器

可以通过写入数据到此寄存器来清除 SUBSRCPND 寄存器的指定位。只有数据中那些被设置为 1 的相应 SUBSRCPND 寄存器的位的位置才能被清除。数据中那些被设置为 0 的相应位的位置则保持不变。

寄存器	地址	R/W	描述	复位值
SUBSRCPND	0X4A000018	R/W	指示中断请求状态 0 = 未请求中断 1 = 中断源已声明中断请求	0x00000000

SUBSRCPND	位	描述	初始状态
保留	[31:15]	未使用	0
INT_AC97	[14]	0 = 未请求 1 = 请求	0
INT_WDT	[13]	0 = 未请求 1 = 请求	0
INT_CAM_P	[12]	0 = 未请求 1 = 请求	0
INT_CAM_C	[11]	0 = 未请求 1 = 请求	0
INT_ADC_S	[10]	0 = 未请求 1 = 请求	0
INT_TC	[9]	0 = 未请求 1 = 请求	0
INT_ERR2	[8]	0 = 未请求 1 = 请求	0
INT_TXD2	[7]	0 = 未请求 1 = 请求	0
INT_RXD2	[6]	0 = 未请求 1 = 请求	0
INT_ERR1	[5]	0 = 未请求 1 = 请求	0
INT_TXD1	[4]	0 = 未请求 1 = 请求	0
INT_RXD1	[3]	0 = 未请求 1 = 请求	0
INT_ERR0	[2]	0 = 未请求 1 = 请求	0
INT_TXD0	[1]	0 = 未请求 1 = 请求	0
INT_RXD0	[0]	0 = 未请求 1 = 请求	0

映射到 SRCPND

SRCPND	SUBSRCPND	备注
INT_UART0	INT_RXD0 , INT_TXD0 , INT_ERR0	
INT_UART1	INT_RXD1 , INT_TXD1 , INT_ERR1	
INT_UART2	INT_RXD2 , INT_TXD2 , INT_ERR2	
INT_ADC	INT_ADC_S , INT_TC	
INT_CAM	INT_CAM_C , INT_CAM_P	
INT_WDT_AC97	INT_WDT , INT_AC97	

中断次级屏蔽 (INTSUBMSK) 寄存器

此寄存器有 11 位，其每一位都与一个中断源相联系。如果某个指定位被设置为 1，则相应中断源的中断请求不会被 CPU 所服务（请注意即使在这种情况下，SRCPND 寄存器的相应位也设置为 1）。如果屏蔽位为 0，则可以服务中断请求。

寄存器	地址	R/W	描述	复位值
INTSUBMSK	0X4A00001C	R/W	决定屏蔽哪个中断源。被屏蔽的中断源将不会服务 0 = 中断服务可用 1 = 屏蔽中断服务	0xFFFF

INTSUBMSK	位	描述	初始状态
保留	[31:15]	未使用	0
INT_AC97	[14]	0 = 可服务 1 = 屏蔽	1
INT_WDT	[13]	0 = 可服务 1 = 屏蔽	1
INT_CAM_P	[12]	0 = 可服务 1 = 屏蔽	1
INT_CAM_C	[11]	0 = 可服务 1 = 屏蔽	1
INT_ADC_S	[10]	0 = 可服务 1 = 屏蔽	1
INT_TC	[9]	0 = 可服务 1 = 屏蔽	1
INT_ERR2	[8]	0 = 可服务 1 = 屏蔽	1
INT_TXD2	[7]	0 = 可服务 1 = 屏蔽	1
INT_RXD2	[6]	0 = 可服务 1 = 屏蔽	1
INT_ERR1	[5]	0 = 可服务 1 = 屏蔽	1
INT_TXD1	[4]	0 = 可服务 1 = 屏蔽	1
INT_RXD1	[3]	0 = 可服务 1 = 屏蔽	1
INT_ERR0	[2]	0 = 可服务 1 = 屏蔽	1
INT_TXD0	[1]	0 = 可服务 1 = 屏蔽	1
INT_RXD0	[0]	0 = 可服务 1 = 屏蔽	1

15 LCD 控制器

概述

S3C2440A 中的 LCD 控制器由从位于系统存储器的视频缓冲区到外部 LCD 驱动器的转移 LCD 图像数据逻辑组成。LCD 控制器支持单色 LCD 的单色、2 位每像素（4 阶灰度）或 4 位每像素（16 阶灰度）模式，通过使用基于时间的抖动算法和帧频控制（FRC）方法，其可以连接到 8 位每像素（256 色）的彩色 LCD 面板和连接到 12 位每像素（4096 色）的 STN LCD。

其支持 1 位每像素、2 位每像素、4 位每像素和 8 位每像素的调色 TFT 彩色 LCD 面板连接，以及 16 位每像素和 24 位每像素的无调色真彩显示。

可以编程 LCD 控制器来支持不同涉及屏幕水平和垂直像素数、数据接口的数据线宽度、接口时序和刷新率的需要。

特性

STN LCD 显示：

- 支持 3 种类型的 LCD 面板：4 位双扫描、4 位单扫描和 8 位单扫描显示类型
- 支持单色、4 阶灰度和 16 阶灰度
- 支持 256 色和 4096 色的彩色 STN LCD 面板
- 支持多种屏幕尺寸

典型实际屏幕尺寸：640×480、320×240、160×160 等

最大虚拟屏幕尺寸为 4M 字节

256 色模式最大虚拟屏幕尺寸：4096×1024、2048×2048、1024×4096 等

TFT LCD 显示：

- 支持 TFT 的 1、2、4、8 bpp（位每像素）调色显示
- 支色彩 TFT 的 16、24 bpp 无调色显示
- 支持 24 位每像素模式下最大 16M 色 TFT
- 支持多种屏幕尺寸

典型实际屏幕尺寸：640×480、320×240、160×160 等

最大虚拟屏幕尺寸为 4M 字节

64K 色模式最大虚拟屏幕尺寸：2048×1024 等

共同特性

LCD 控制器有一个支持从位于系统存储器的视频缓冲器接收图像数据的专用 DMA。其功能同样包括：

- 专用中断功能 (INT_FrSyn 和 INT_FiCnt)
- 使用系统存储器作为显存
- 支持多种虚拟屏 (支持硬件水平/垂直滚动)
- 可编程不同显示面板的时序控制
- 支持大/小端字节顺序，和 WinCE 数据格式一样
- 支持 2 种类型 SEC TFT LCD 面板

(三星 3.5 吋竖屏/256K 色/反光型和半透型 a-Si TFT LCD)

LTS350Q1-PD1 : 带触摸屏和前光源单元的 TFT LCD 面板 (反光型)

LTS350Q1-PD2 : 只是 TFT LCD 面板

LTS350Q1-PE1 : 带触摸屏和前光源单元的 TFT LCD 面板 (半透型)

LTS350Q1-PE2 : 只是 TFT LCD 面板

注意：WinCE 不支持 12 位封装数据格式。请检查是否 WinCE 可以支持 12 位色模式。

外接信号

STN	TFT	SEC TFT (LTS350Q1-PD1/2)	SEC TFT (LTS350Q1-PE1/2)
VFRAME (帧同步信号)	VSYNC (垂直同步信号)	STV	STV
VLINE (行同步脉冲信号)	Hsync (水平同步信号)	CPV	CPV
VCLK (像素时钟信号)	VCLK (像素时钟信号)	LCD_HCLK	LCD_HCLK
VD[23:0] (LCD 像素数据输出端口)	VD[23:0] (LCD 像素数据输出端口)	VD[23:0]	VD[23:0]
VM (LCD 驱动器的 AC 偏压信号)	VDEN (数据使能信号)	TP	TP
-	LEND (行结束信号)	STH	STH
LCD_PWREN	LCD_PWREN	LCD_PWREN	LCD_PWREN
-	-	LPC_OE	LCC_INV
-	-	LPC_REV	LCC_REV
-	-	LPC_REV_B	LCC_REV_B

方框图

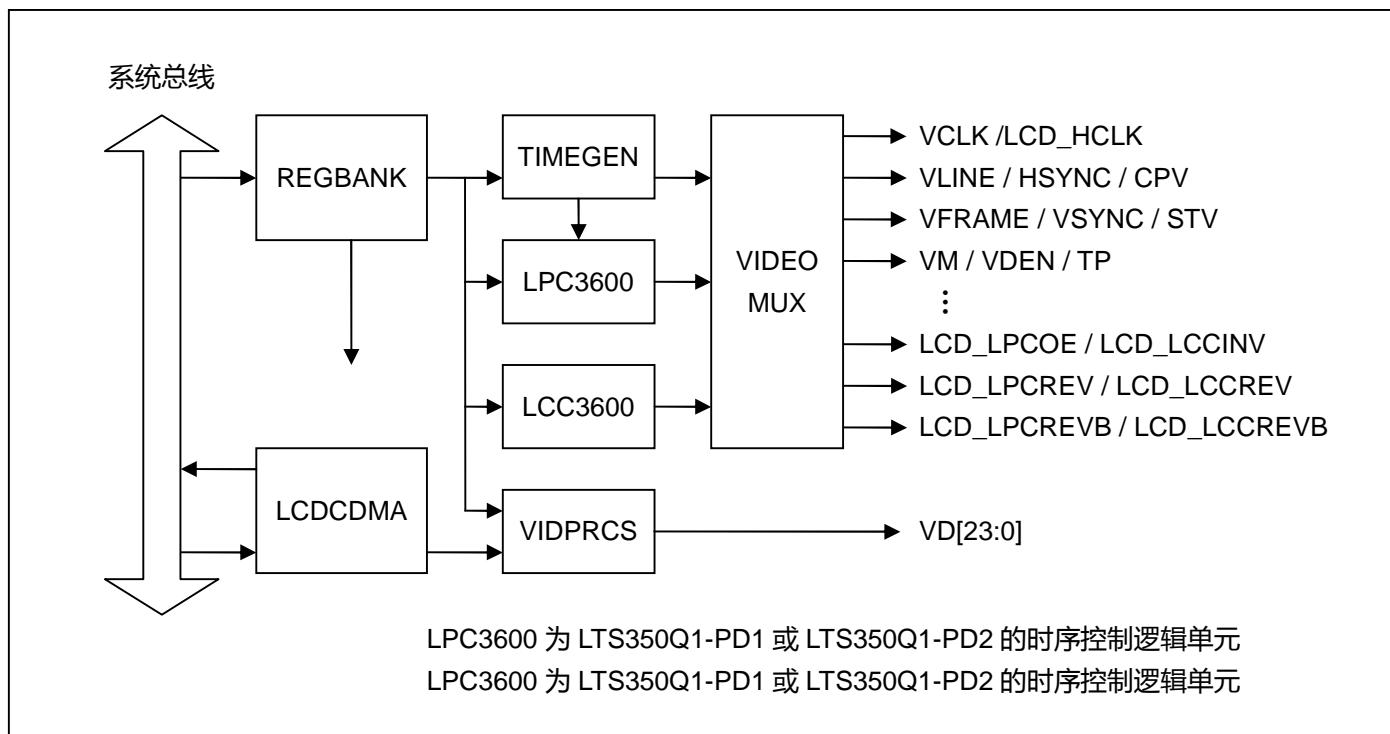


图 15-1. LCD 控制器方框图

S3C2440A LCD 控制器是用于传输视频数据和产生必要的控制信号，如 VFRAME、VLINE、VCLK、VM 等等。除控制信号外，S3C2440A 还有视频数据的数据端口，如图 15-1 所示的 VD[23:0]。LCD 控制器包括 REGBANK、LCDCDMA、VIDPRCS、TIMEGEN 和 LPC3600（见图 15-1 LCD 控制器方框图）。REGBANK 有 17 个可编程寄存器集和用于配制 LCD 控制器的 256×16 个调色存储器。LCDCDMA 专用于 DMA，它可以自动从帧存储器到 LCD 驱动器传输视频数据。通过使用专用 DMA，可以在屏幕上显示视频数据而不需要 CPU 的介入。VIDPRCS 接收来自 LCDCDMA 的视频数据并且在将其变换为适当格式为后通过 VD[23:0]数据端口发送视频数据到 LCD 驱动器，例如 4/8 位信号信号扫描或 4 位双扫描显示模式。TIMEGEN 由可编程逻辑组成来支持发现不同 LCD 驱动器的一般接口时序和速率的变化需要。TIMEGEN 模块产生 VFRAME、VLINE、VCLK、VM 等等。

数据流描述如下：

FIFO 存储器 LCDCDMA。当 FIFO 为空或部分为空时，LCDCDMA 请求基于突发存储器传输模式（连续 4 字（16 字节）每单次突发请求的存储器刷新，无需总线传输期间随着总线主控到其它总线主机）从帧存储器刷新数据。当存储器控制器中的总线仲裁器同意了传输请求，将会从系统存储器到内部 FIFO 传输 4 个连续字数据。FIFO 分别由 12 字 FIFOL 和 16 字 FIFOH 组成，总计 28 字大小。S3C2440A 有 2 个 FIFO 来支持双扫描显示模式。在单扫描模式情况中，只能使用 FIFO 的一个（FIFOH）。

STN LCD 控制器操作

时序发生器 (TIMEGEN)

TIMEGEN 产生 LCD 驱动器的控制信号，如 VFRAME、VLINE、VCLK 和 VM。这些控制信号与 REGBANK 中的 LCDCON1/2/3/4/5 寄存器配制有着紧密关系。基于这些可编程的 REGBANK 中 LCD 控制寄存器的配制，TIMEGEN 可以产生合适的可编程控制信号来支持多种不同类型的 LCD 驱动器。

在一次每帧的频率下为全部第一行持续时间而发出 VFRAME 脉冲。发出 VFRAME 信号引起 LCD 的行指向显示的最上方来重新开始显示。

VM 信号帮助 LCD 驱动器交替用于开启或关闭像素的行和列电压的极性。VM 信号的触发率取决于 LCDCON1 寄存器的 MMODE 位和 LCDCON4 寄存器的 MVAL 字段。如果 MMODE 为 0，则 VM 信号被配制为每帧触发。如果 MMODE 为 1，则 VM 信号被配制为每次由 MVAL[7:0] 值的 VLINE 指定数量的消逝事件触发。图 15-4 显示了一个 MMODE=0 和 MVAL[7:0]=0x2 的 MMODE=1 例子。当 MMODE=1 时，VM 频率与 MVAL[7:0] 有关，显示如下：

$$\text{VM 频率} = \text{VLINE 频率} / (2 \times \text{MVAL})$$

VFRAME 和 VLINE 脉冲发生取决于 LCDCON2/3 寄存器中 HOZVAL 字段和 LINEVAL 字段的配制。每个字段又与 LCD 大小和显示模式有关。换句话说，HOZVAL 和 LINEVAL 可以由 LCD 面板大小和显示模式根据以下等式决定：

$$\text{HOZVAL} = (\text{水平显示大小} / \text{有效 VD 数据行数}) - 1$$

$$\text{彩色模式中 : 水平显示大小} = 3 \times \text{水平像素数}$$

4 位单扫描显示模式中，有效 VD 数据行数应该为 4。在 4 位双扫描显示情况中，有效 VD 数据行数应该同样为 4 而 8 位单扫描显示模式中，有效 VD 数据行数应该为 8。

$$\text{LINEVAL} = (\text{虚拟显示大小}) - 1 : \text{单扫描显示类型情况中}$$

$$\text{LINEVAL} = (\text{虚拟显示大小} / 2) - 1 : \text{双扫描显示类型情况中}$$

VCLK 信号速率取决于 LCDCON1 寄存器中 CLKVAL 字段的配制。表 15-1 定义了 VCLK 与 CLKVAL 的关系。CLKVAL 的最小值为 2。

$$\text{VCLK (Hz)} = \text{HCLK} / (\text{CLKVAL} \times 2)$$

帧频为 VFRAM 信号频率。帧频与 LCDCON1/2/3/4 寄存器中 WLH[1:0] (VLINE 脉宽) WDLY[1:0] (VLINE 脉冲后 VCLK 的延迟宽度)、HOZVAL、LINEBLANK 和 LINEVAL 字段以及 VCLK 和 HCLK 密切相关。多数 LCD 驱动器需要它们自己适当的帧频。帧频计算如下：

$$\text{帧频 (Hz)} = 1 / [\{ (1 / \text{VCLK}) \times (\text{HOZVAL} + 1) + (1 / \text{HCLK}) \times (\mathbf{A} + \mathbf{B} + (\text{LINEBLANK} \times 8)) \} \times (\text{LINEVAL} + 1)]$$

$$\mathbf{A} = 2^{(4+WLH)}, \mathbf{B} = 2^{(4+WDLY)}$$

表 15-1. VCLK 与 CLKVAL 之间的关系 (STN、HCLK = 60MHz)

CLKVAL	60MHz/X	VCLK
2	60 MHz / 4	15.0 MHz
3	60 MHz / 6	10.0 MHz
...
1023	60 MHz / 2046	29.3 kHz

视频操作

S3C2440A LCD 控制器支持 8 位色模式 (256 色模式) , 12 位色模式 (4096 色模式) , 4 阶灰度模式 , 16 阶灰度模式和单色模式。要求灰阶或彩色模式通过基于时间的抖动算法和帧频控制 (FRC) 方法来完成灰阶或彩色的色深。该选项可以通过跟随一个可编程查找表 , 其将在后面解释。单的模式不顾这些模块 (FRC 和查找表) 和基本上通过移位视频数据到 LCD 驱动器来串行化 FIFOH (和 FIFOL , 如果使用双扫描显示类型) 的数据到 4 位 (或 8 位 , 如果使用 4 位双扫描或 8 位单扫描显示类型) 流。

以下章节描述根据查找表和 FRC 的灰阶和彩色模式的操作。

查找表

S3C2440A 能支持查找表给彩色或灰阶映射的各种选择 , 以确保用户的灵活操作。查找表就是调色板 , 其允许色深或灰度的选择 (4 阶灰度模式情况中的 16 灰阶中的 4 灰阶的选择 , 256 色模式情况中的 16 红色深中的 8 红色深、 16 绿色深中的 8 绿色深及 16 蓝色深中的 4 蓝色深的选择) 。换句话说 , 用户可以在 4 阶灰度模式中通过使用查找表来选择 16 灰阶中的 4 灰阶。不能在 16 阶灰度模式中选择灰度 ; 所有 16 阶灰度必须在可能的 16 灰阶中选择。 256 色情况中 , 分配 3 位给红色 , 3 位给绿色及 2 位给蓝色。 256 色意味着颜色由 8 红、 8 绿和 4 蓝 ($8 \times 8 \times 4 = 256$) 色深组合而成的。彩色模式中 , 可以使用查找表给合适的选择。可以选择 16 可能的红色深中的 8 红色深、 16 绿色深中的 8 绿色深和 16 蓝色深中的 4 蓝色深。 4096 色模式情况中不能像 256 色模式中一样有选择。

灰度模式操作

S3C2440A LCD 控制器支持 2 种灰度模式 : 2 位每像素灰度 (4 阶灰度) 和 4 位每像素灰度 (16 阶灰度) 。 2 位每像素灰度模式是使用查找表 (BLUELUT) , 其允许 4 灰阶还是 16 可能的灰阶的选择。 2 位每像素灰度模式查找表是使用等同于彩色模式中蓝查找表的蓝查找表 (BLUELUT) 寄存器中的 BLUEVAL[15:0] 。将由 BLUEVAL[3:0] 值标记灰度 0 。如果 BLUEVAL[3:0] 为 9 , 则等级 0 将被表示为 16 灰阶中的灰阶 9 。如果 BLUEVAL[3:0] 为 15 , 则等级 0 将被表示为 16 灰阶中的灰阶 15 , 等等。接着以上相同方法 , 等级 1 也将由 BLUEVAL[7:4] 标记 , 等级 2 也将由 BLUEVAL[11:8] 标记和等级 3 也将由 BLUEVAL[15:12] 标记。 BLUEVAL[15:0] 中的这 4 组将表示等级 0 、等级 1 、等级 2 和等级 3 。 16 阶灰度中 , 没有像 16 阶灰度中一样的选择。

256 级色模式操作

S3C2440A LCD 控制器能支持 8 位每像素 256 色显示模式。彩色显示模式可以通过使用抖动算法和 FRC 来产生 256 级的色深。编码 8 位每像素到 3 位给红色、 3 位给绿色和 2 位给蓝色。彩色显示模式使用不同查找表给红色、绿色和蓝色。每个查找表是使用 REDLUT 寄存器的 REDVAL[31:0] 、 GREENLUT 寄存器的 GREENVAL[31:0] 和 BLUELUT 寄存器的 BLUEVAL[15:0] 作为可编程查找表入口。

类似于灰度显示 ,REDLUR 寄存器中 4 位的 8 组或字段 即 REDVAL[31:28] 、 REDLUT[27:24] 、 REDLUT[23:20] 、 REDLUT[19:16] 、 REDLUT[15:12] 、 REDLUT[11:8] 、 REDLUT[7:4] 和 REDLUT[3:0] , 被分配给每个红色等级。可能的 4 位 (每个字段) 组合都为 16 , 并且每个红色等级应该被分配到可能的 16 种情况中的 1 级。换句话说 , 用户可以通过使用此查找表的类型来选择合适的红色等级。对于绿色 , GREENLUT 寄存器的 GREENVAL[31:0] 被分配作为查找表 , 与红色情况相同操作。类似的 , BLUELUT 寄存器的 BLUEVAL[15:0] 同样被分配作为查找表。对于蓝色 , 2 位被分配给 4 蓝色等级 , 与 8 红色或绿色等级不同。

4096 级色模式操作

S3C2440A LCD 控制器能支持 12 位每像素 4096 色显示模式。彩色显示模式可以通过使用抖动算法和 FRC 来产生 256 级的色深。 12 位每像素被编码到 4 位给红色 , 4 位给绿色和 4 位给蓝色。 4096 色显示模式不使用查找表。

抖动和帧频控制

STN LCD 显示情况中（除了单色），必须通过抖动算法来处理视频数据。DITHFRC 模块有两种功能，降低闪烁的基于时间的抖动算法和 STN 面板显示上灰度和色深的帧频控制（FRC）。这里将描述 STN 面板上灰度和色深显示的基于 FRC 的主要原理。例如，要显示总计 16 等级中的第三灰度（3/16），应该 3 次像素打开和 13 次像素关闭。换句话说，应该选择 16 帧中的 3 帧，其中 3 帧应该在特定像素上有像素打开，剩余的 13 帧在特定像素上有像素关闭。应该周期性显示这 16 帧。这就是如何在屏幕上显示灰度的基本原理，即所谓的由 FRC 的灰度显示。实际例子如表 15-2 所示。为了表示表中第 14 灰度，需要一个 6/7 占空比，这意味着有 6 次像素打开和 1 次像素关闭。灰度的另一种情况同样如表 15-2 所示。

STN LCD 显示中应该提醒一项，即由于相邻帧的同时像素打开和关闭的闪烁噪声。例如，如果第一帧的全部像素打开并且下一帧的所有像素都关闭，闪烁噪声将为最大化。为了降低屏幕上的闪烁噪声，帧之间的像素打开和关闭的平均几率应该相同。为了实现这一点应该使用多样化每帧相邻像素的方式的基于时间抖动算法。以下为详细解释。对于 16 阶灰度，FRC 应该有以下灰度和 FRC 之间的关系。第 15 灰度应该总为像素打开，第 14 灰度应该有 6 次像素打开和 1 次像素关闭，第 13 灰度应该有 4 次像素打开和 2 次像素关闭，……，第 0 灰度应该为像素关闭，如表 15-2 所示。

表 15-2. 抖动占空比例子

预抖动数据 (灰度号)	占空比	预抖动数据 (灰度号)	占空比
15	1	7	1/2
14	6/7	6	3/7
13	4/5	5	2/5
12	3/4	4	1/3
11	5/7	3	1/4
10	2/3	2	1/5
9	3/5	1	1/7
8	4/7	0	0

显示类型

LCD 控制器支持 3 种 LCD 驱动器类型：4 位双扫描、4 位单扫描、8 位单扫描显示模式。图 15-2 显示了单色显示的这 3 种不同显示类型，图 15-3 显示了彩色显示的这 3 种不同显示类型。

4 位双扫描显示类型

一个 4 位双扫描显示使用 8 条并行数据线同时移位数据到显示的上半部分和下半部分。8 条并行数据线中 4 位数据被移位到高半部分并且另 4 位数据被移位到低半部分，如图 15-2 所示。当显示的每一半已经被移位和转移时到达帧末尾。从 LCD 控制器的 LCD 输出的 8 个引脚（VD[7:0]）可以直接连接到 LCD 驱动器。

4 位单扫描显示类型

4 位单扫描显示使用 4 条并行数据线逐一移位数据到显示的连续单水平行，直到全部帧都已经被移位和转移。从 LCD 控制器的 LCD 输出的 4 个引脚（VD[3:0]）可以直接连接到 LCD 驱动器，并且未使用 LCD 输出的另 4 个引脚（VD[7:4]）。

8 位单扫描显示类型

8 位单扫描显示使用 8 条并行数据线逐一移位数据到显示的连续单水平行，直到全部帧都已经被移位和转移。从 LCD 控制器的 LCD 输出的 8 个引脚（VD[7:0]）可以直接连接到 LCD 驱动器。

256 色显示

彩色显示每像素需要 3 位（红、绿和蓝）图像数据，因此每水平行的水平移位寄存器数等于单一水平行的像素数的 3 倍。这将导致水平移位寄存器对于每水平行的像素数长 3 倍。此 RGB 被移位到 LCD 驱动器作为经过并行数据线的连续位。图 15-3 显示了 3 种彩色显示类型的并行数据线中 RGB 和像素次序。

4096 色显示

彩色显示每像素需要 3 位（红、绿和蓝）图像数据，因此每水平行的水平移位寄存器数等于单一水平行的像素数的 3 倍。此 RGB 被移位到 LCD 驱动器作为经过并行数据线的连续位。此 RGB 次序由视频缓冲器中的视频数据顺序决定。

存储器数据格式 (STN , BSWP = 0)**单 4 位双扫描显示：**

视频缓冲存储器：

地址	数据
0000H	A[31:0]
0004H	B[31:0]
.	.
.	.
1000H	L[31:0]
1004H	M[31:0]
.	.
.	.

LCD 面板

A[31] A[30] A[0] B[31] B[30] B[0]
L[31] L[30] L[0] M[31] M[30] M[0]

单 4 位单扫描显示和 8 位单扫描显示：

视频缓冲存储器：

地址	数据
0000H	A[31:0]
0004H	B[31:0]
0008H	C[31:0]
.	.
.	.

LCD 面板

A[31] A[30] A[29] A[0] B[31] B[30] B[0] C[31] C[0]

4 阶灰度模式中，视频数据的 2 位相当于 1 个像素。

16 阶灰度模式中，视频数据的 4 位相当于 1 个像素。

256 色模式中，视频数据的 8 位（红 3 位、绿 3 位、蓝 2 位）相当于 1 个像素。字节中彩色数据格式如下：

位[7:5]	位[4:2]	位[1:0]
红色	绿色	蓝色

2096 色模式中：

封装的 12 BPP 彩色模式

视频数据的 12 位（红 4 位、绿 4 位、蓝 4 位）相当于 1 个像素。下表显示了字中彩色数据格式：(视频数据必须固定以 3 字为边界 (8 像素) , 如下)

RGB 次序

数据	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
字#1	红(1)	绿(1)	蓝(1)	红(2)	绿(2)	蓝(2)	红(3)	绿(3)
字#2	蓝(3)	红(4)	绿(4)	蓝(4)	红(5)	绿(5)	蓝(5)	红(6)
字#3	绿(6)	蓝(6)	红(7)	绿(7)	蓝(7)	红(8)	绿(8)	蓝(8)

未封装的 12 BPP 彩色模式

视频数据的 12 位（红 4 位、绿 4 位、蓝 4 位）相当于 1 个像素。下表显示了字中彩色数据格式：

RGB 次序

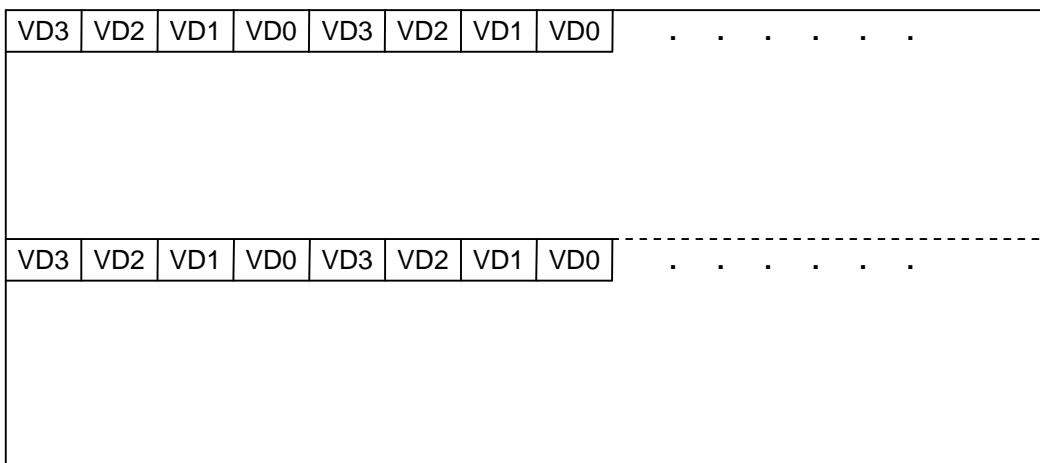
数据	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
字#1	-	红(1)	绿(1)	蓝(1)	-	红(2)	绿(2)	蓝(2)
字#2	-	红(3)	绿(3)	蓝(3)	-	红(4)	绿(4)	蓝(4)
字#3	-	红(5)	绿(5)	蓝(5)	-	红(6)	绿(6)	蓝(6)

16 BPP 彩色模式

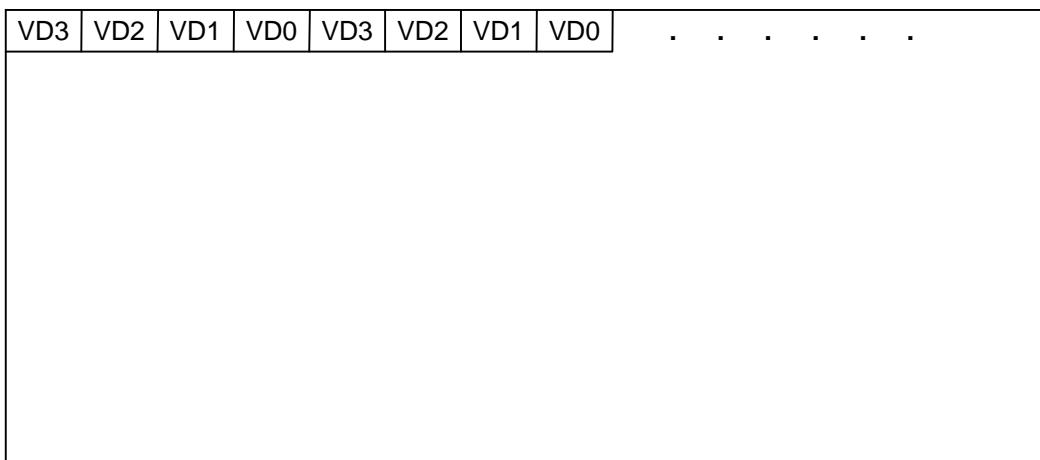
视频数据的 16 位（红 5 位、绿 6 位、蓝 5 位）相当于 1 个像素。但是 STN 控制器将只会使用 12 位彩色数据。这意味着只有每个彩色数据的高 4 位将被用作像素数据 (R[15:12]、G[10:7]、B[4:1])。下表显示了字中彩色数据格式：

RGB 次序

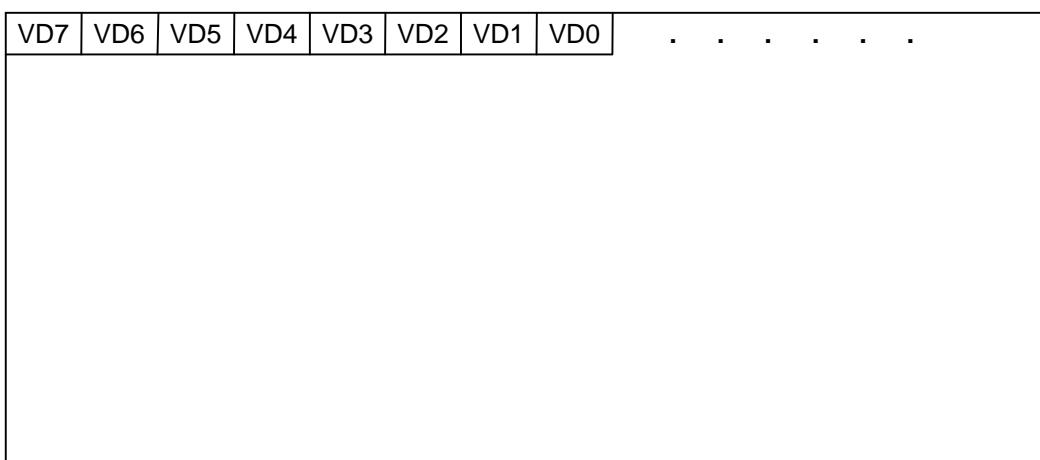
数据	[31:28]	[27:21]	[20:16]	[15:11]	[10:5]	[4:0]
字#1	红(1)	绿(1)	蓝(1)	红(2)	绿(2)	蓝(2)
字#2	红(3)	绿(3)	蓝(3)	红(4)	绿(4)	蓝(4)
字#3	红(5)	绿(5)	蓝(5)	红(6)	绿(6)	蓝(6)



4 位双扫描显示



4 位单扫描显示



8 位单扫描显示

图 15-2. 单色显示类型 (STN)

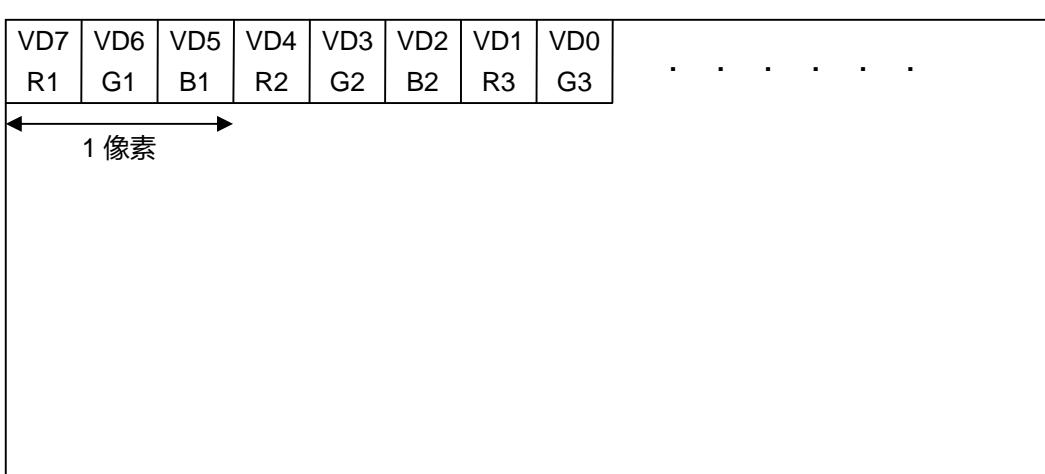
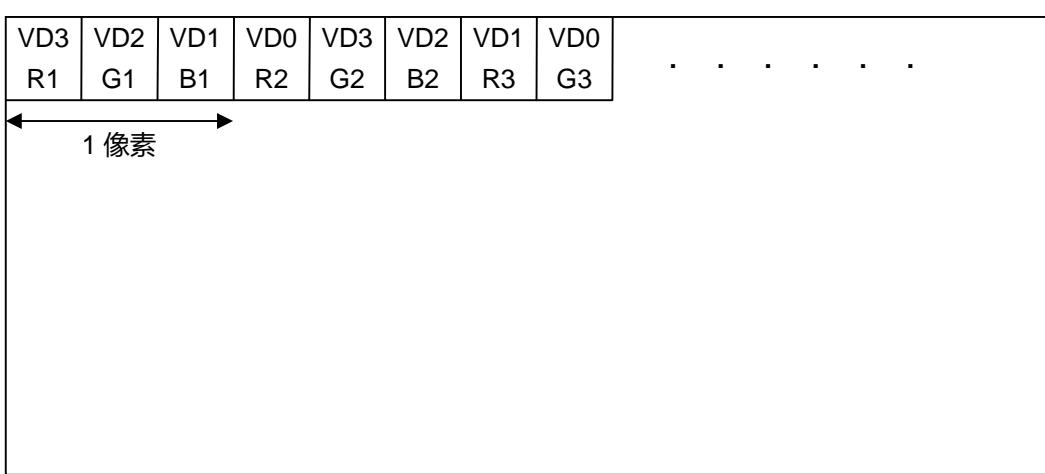


图 15-3. 彩色显示类型 (STN)

时序要求

应该使用 VD[7:0]信号移到从存储器转 LCD 驱动器的图像数据。VCLK 信号是用于数据到 LCD 驱动器移位寄存器的时钟。数据的每水平行移位到 LCD 驱动器移位寄存器后，发出 VLINE 信号以显示面板的行。

VM 信号提供一个 AC 信号给显示。LCD 用此信号来交替用于打开或关闭像素的行和列电压的极性，这时由于每当承受 DC 电压时 LCD 等离子趋向退化。可以配制其为每帧触发或每 VLINE 信号的可编程数量触发。

图 15-4 显示了 LCD 驱动器接口的时序要求

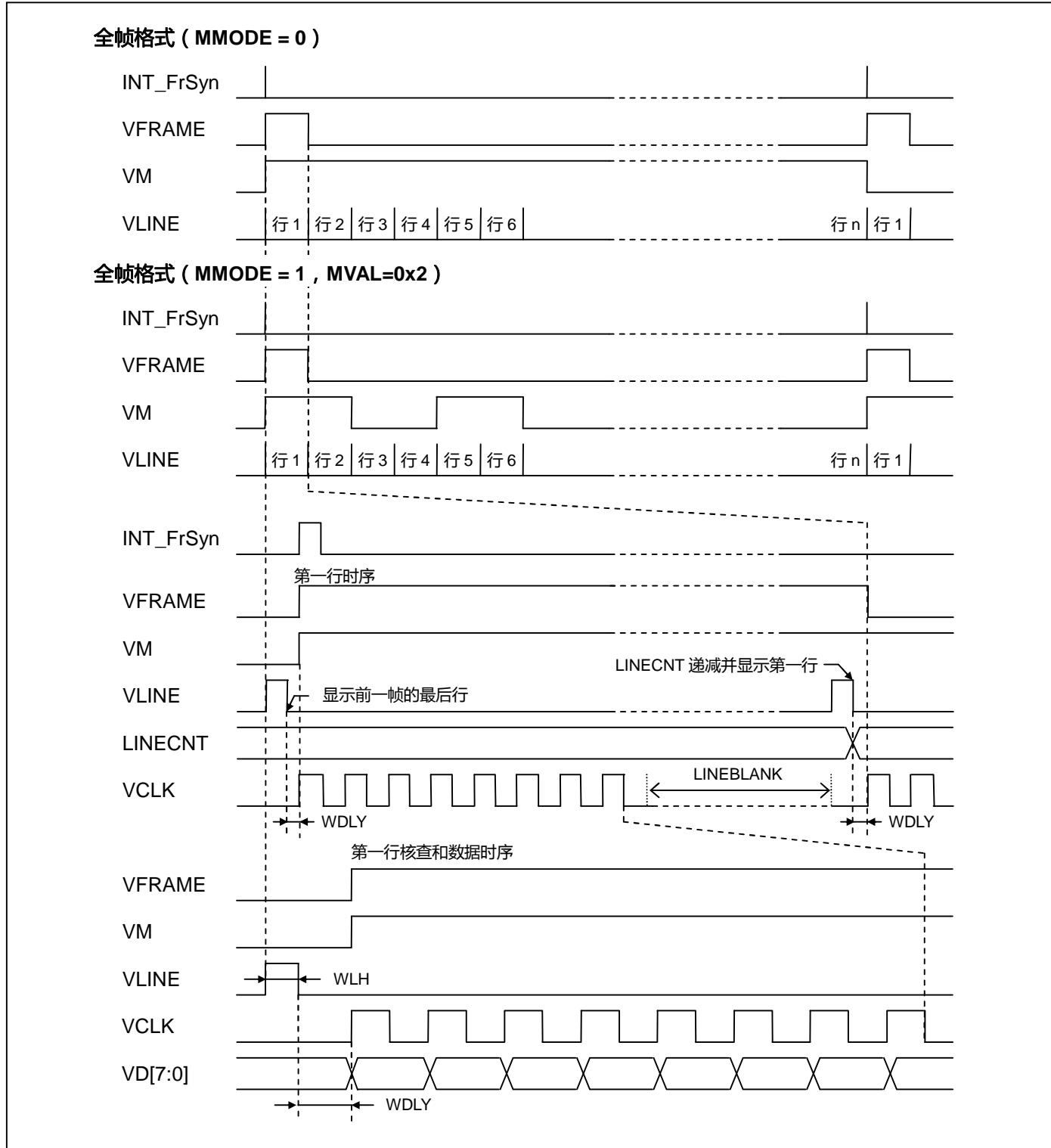


图 15-4. 8 位单扫描显示类型 STN LCD 时序

TFT LCD 控制器操作

TIMEGEN 产生控制信号给 LCD 驱动器，例如 VSYNC、Hsync、VCLK、VDEN 和 LEND 信号。这些控制信号与 REGBANK 中 LCDCON1/2/3/4/5 寄存器的配制有极大的关系。基于 REGBANK 中的 LCD 控制寄存器的这些可编程配制，TIMEGEN 可以产生可编程控制信号适合支持多种不同 LCD 驱动器的类型。

发出 VSYNC 信号来引起 LCD 的行指针重新从显示的顶处开始。

VSYNC 和 Hsync 脉冲的产生取决于 LCDCON2/3 寄存器中 HOZVAL 字段和 LINEVAL 字段的配制。HOZVAL 和 LINEVAL 可以按照下列等式由 LCD 面板大小决定：

- HOZVAL = (水平显示大小) - 1
- LINEVAL = (垂直显示大小) - 1

VCLK 信号的频率取决于 LCDCON1 寄存器中的 CLKVAL 字段。表 15-3 定义了 VCLK 和 CLKVAL 之间的关系。CLKVAL 的最小值为 0。

$$VCLK (\text{Hz}) = HCLK / [(CLKVAL + 1) \times 2]$$

帧频即为 VSYNC 信号频率。帧频与 LCDCON1 和 LCDCON2/3/4 寄存器中的 VSYNC、VBPD、VFPD、LINEVAL、Hsync、HBPD、HFPD、HOZVAL、和 CLKVAL 字段有关系。多数 LCD 驱动器需要它们自己适当的帧频。

$$\text{帧频} (\text{Hz}) = 1 / [\{ (1 / VCLK) \times (HOZVAL + 1) + (1 / HCLK) \times (A + B + (LINEBLANK \times 8)) \} \times (LINEVAL + 1)]$$

$$A = 2^{(4+WLH)}, B = 2^{(4+WDLY)}$$

表 15-3. VCLK 和 CLKVAL 之间的关系 (TFT, HCLK = 60MHz)

CLKVAL	60MHz/X	VCLK
1	60 MHz / 4	15.0 MHz
2	60 MHz / 6	10.0 MHz
...
1023	60 MHz / 2048	30.0 kHz

视频操作

S3C2440A 中的 TFT LCD 控制器支持 1、2、4 或 8bpp (位每像素) 调色显示和 16 或 24 bpp 无调色真彩显示。

256 色调色板

S3C2440A 可以支持 256 色调色板给各种色彩映射的选择，以提供灵活操作给用户。

存储器数据格式 (TFT)

此章节包括一些每种显示模式的例子。

24BPP 显示

(BSWP = 0 , HWSWP = 0 , BPP24BL = 0)

	D[31:24]	D[23:0]
000H	空位	P1
004H	空位	P2
008H	空位	P3
...		

(BSWP = 0 , HWSWP = 0 , BPP24BL = 1)

	D[31:24]	D[23:0]
000H	P1	空位
004H	P2	空位
008H	P3	空位
...		



24BPP 下 VD 引脚描述

VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
红	7	6	5	4	3	2	1	0																
绿									7	6	5	4	3	2	1	0								
蓝																	7	6	5	4	3	2	1	0

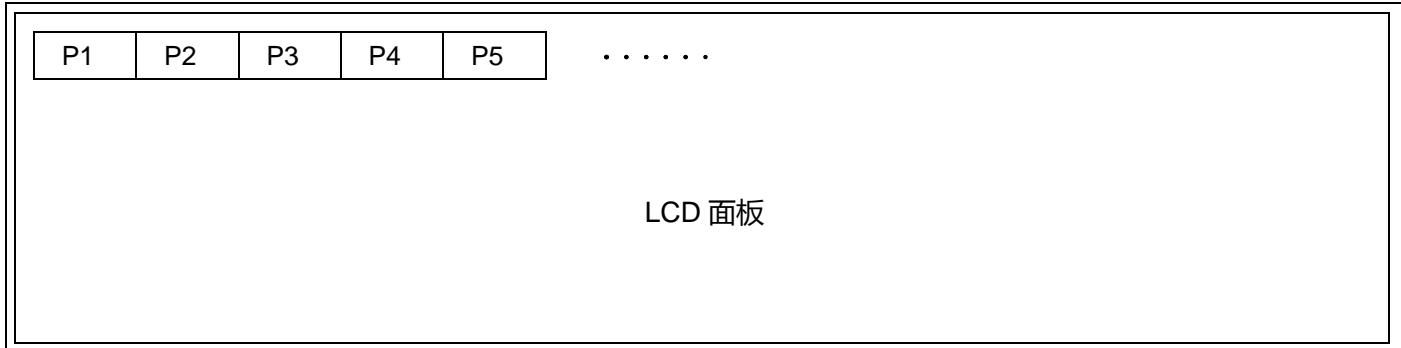
16BPP 显示

(BSWP = 0 , HWSWP = 0)

	D[31:16]	D[15:0]
000H	P1	P2
004H	P3	P4
008H	P5	P6
...		

(BSWP = 0 , HWSWP = 1)

	D[31:16]	D[15:0]
000H	P2	P1
004H	P4	P3
008H	P6	P5
...		

**16BPP 下 VD 引脚描述**

(5:6:5)

VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
红	4	3	2	1	0	NC									NC								NC	
绿								5	4	3	2	1	0											
蓝																	4	3	2	1	0			

(5:5:5:I)

VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
红	4	3	2	1	0	I	NC								NC								NC	
绿									4	3	2	1	0	I										
蓝																	4	3	2	1	0	I		

注释：未使用 VD 引脚可以用作 GPIO

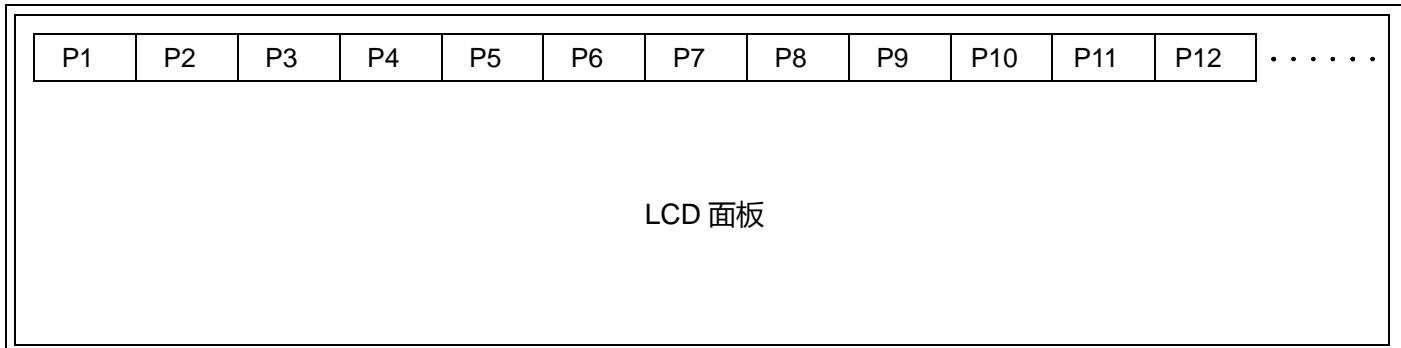
8BPP 显示

(BSWP = 0 , HWSWP = 0)

	D[31:24]	D[23:16]	D[15:8]	D[7:0]
000H	P1	P2	P3	P4
004H	P5	P6	P7	P8
008H	P9	P10	P11	P12
...				

(BSWP = 1 , HWSWP = 0)

	D[31:24]	D[23:16]	D[15:8]	D[7:0]
000H	P4	P3	P2	P1
004H	P8	P7	P6	P5
008H	P12	P11	P10	P9
...				

**4BPP 显示**

(BSWP = 0 , HWSWP = 0)

	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P1	P2	P3	P4	P5	P6	P7	P8
004H	P9	P10	P11	P12	P13	P14	P15	P16
008H	P17	P18	P19	P20	P21	P22	P23	P24
...								

(BSWP = 1 , HWSWP = 0)

	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P7	P8	P5	P6	P3	P4	P1	P2
004H	P15	P16	P13	P14	P11	P12	P9	P10
008H	P23	P24	P21	P22	P19	P20	P17	P18
...								

2BPP 显示

(BSWP = 0 , HWSWP = 0)

D	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P1	P2	P3	P4	P5	P6	P7	P8
004H	P17	P18	P19	P20	P21	P22	P23	P24
008H	P33	P34	P35	P36	P37	P38	P39	P40
...								

D	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P9	P10	P11	P12	P13	P14	P15	P16
004H	P25	P26	P27	P28	P29	P30	P31	P32
008H	P41	P42	P43	P44	P45	P46	P47	P48
...								

256 调色板使用 (TFT)

调色板配制和格式控制

S3C2440A 提供了 256 色调色板给 TFT LCD 控制。用户可以在从 64K 色中选择这 2 种格式中 256 色。256 色调色板由 256 (深度) × 16 位 SPSRAM 组成。调色板支持 5:6:5 (R:G:B) 格式和 5:5:5:1 (R:G:B:I) 格式。当用户使用 5:5:5:1 格式时，强度数据 (I) 是用作每个 RGB 数据的共用 LSB 位。因此 5:5:5:1 格式与 R(5+I):G(5+I):B(5+I) 格式相同。在 5:5:5:1 格式中，例如，用户可以写调色板为如表 15-5 并接着接通 VD 引脚到 TFT LCD 面板 (R(5+I) = VD[23:19] + VD[18]、VD[10] 或 VD[2]，G(5+I) = VD[15:11] + VD[18]、VD[10] 或 VD[2]，B(5+I) = VD[7:3] + VD[18]、VD[10] 或 VD[2]。)，并且设置 LCDCON5 寄存器的 FRM565 为 0。

表 15-4. 5:6:5 格式

索引\位的位置	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	地址
00H	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	¹⁾ 0X4D000400
01H	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	0X4D000404
.....																
FFH	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	0X4D0007FC
VD 数	23	22	21	20	19	15	14	13	12	11	10	7	6	5	4	3	

表 15-5. 5:6:5:1 格式

索引\位的位置	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	地址
00H	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	¹⁾ 0X4D000400
01H	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	0X4D000404
.....																
FFH	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	0X4D0007FC
VD 数	23	22	21	20	19	15	14	13	12	11	7	6	5	4	3	²⁾	

注释：

1. 0x4D000400 为调色板开始地址
2. VD18、VD10 和 VD2 为相同输出值，I。
3. DATA[31:16] 为无效。

调色板读/写

当用户执行调色板的读/写操作时，必须检查 LCDCON5 寄存器 HSTATUS 和 VSTATUS，在 HSTATUS 和 VSTATUS 的 ACTIVE 状态期间禁止读/写操作。

临时调色板配制

S3C2440A 允许用户无需进行复杂的更改来单色填充帧，只需要填充单色到帧缓冲器或调色板。通过写入一个在 LCD 面板上显示的颜色的值来显示单色帧到 TPAL 寄存器的 TPALVAL 并且使能 TPALEN。

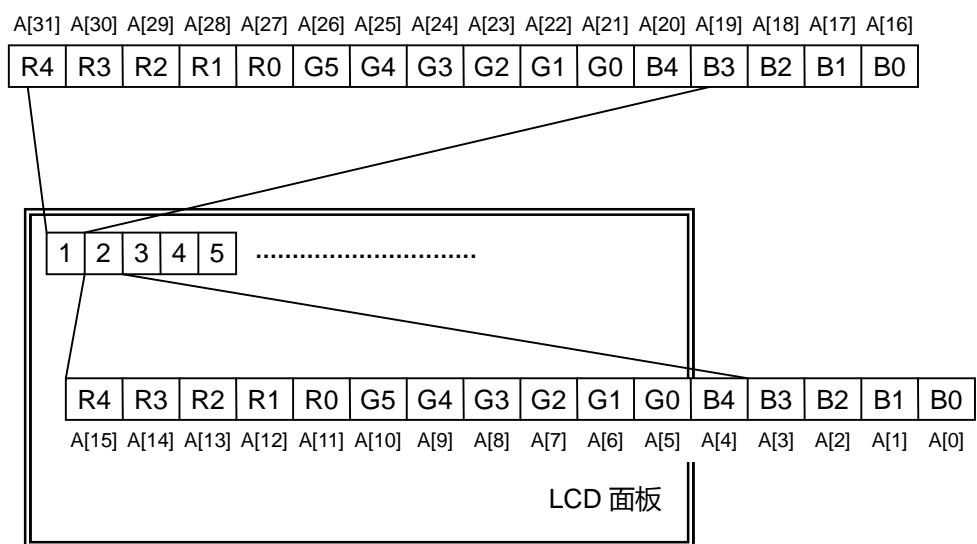
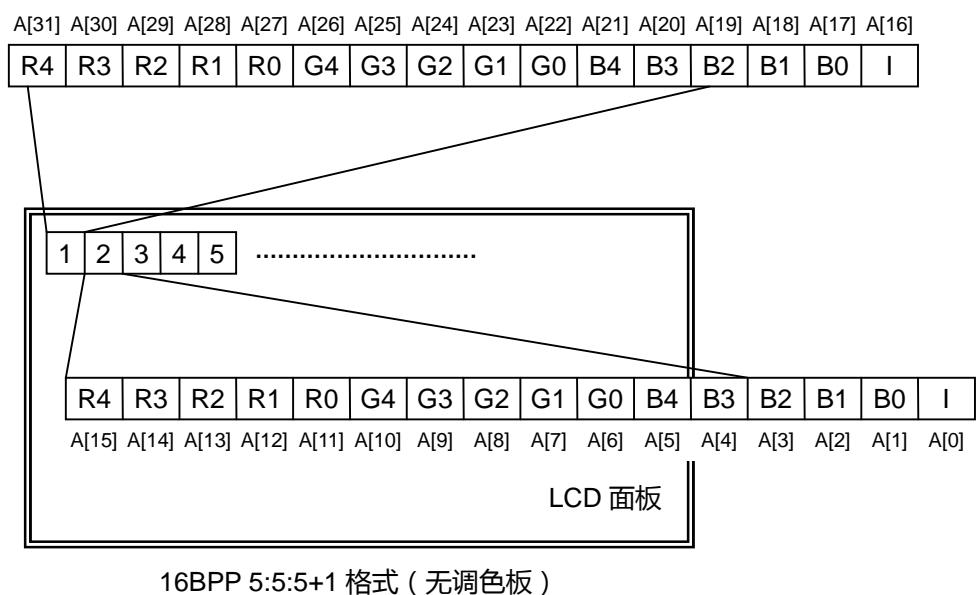


图 15-5. 16BPP 显示类型 (TFT)

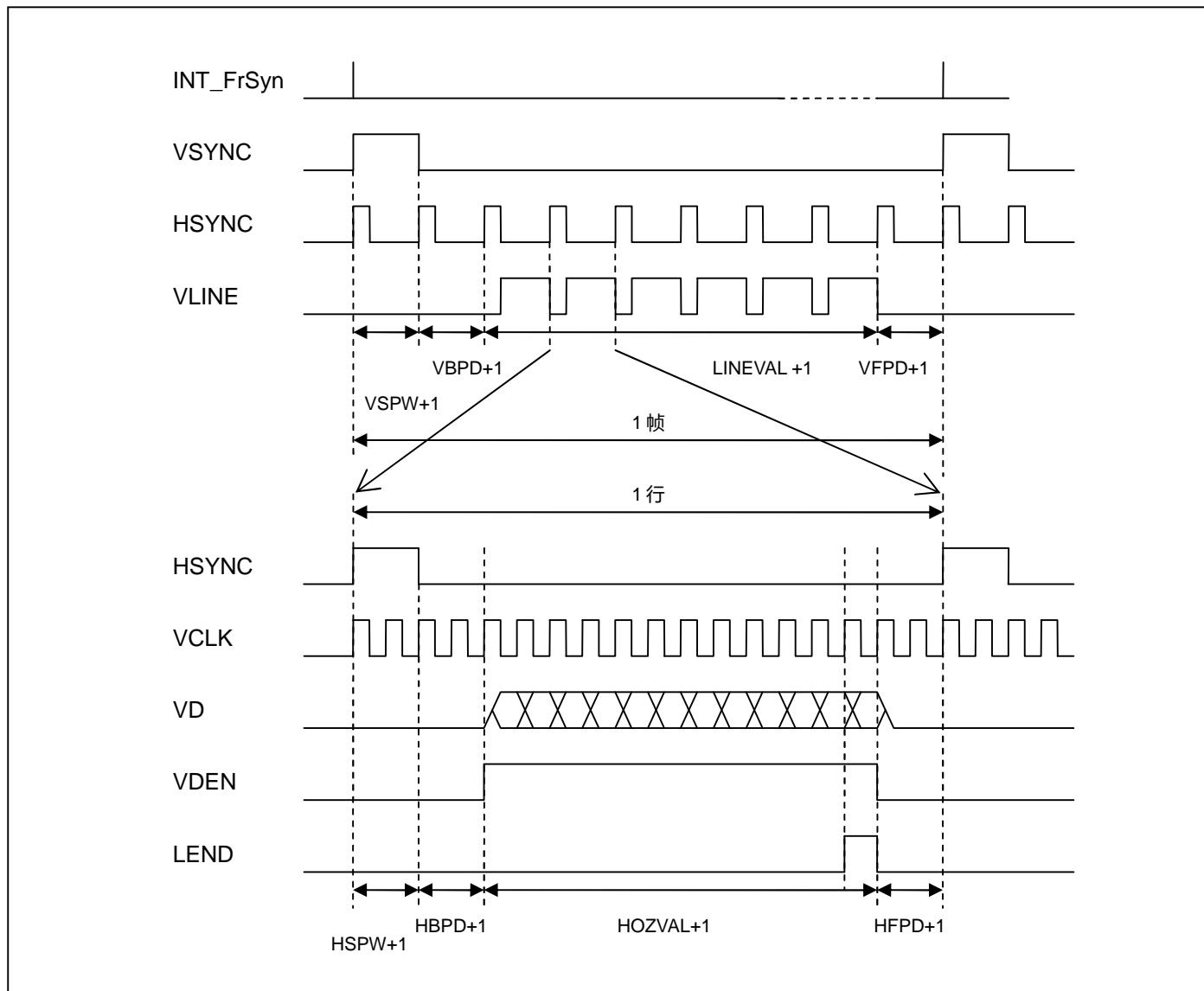


图 15-6. TFT LCD 时序例子

三星 TFT LCD 面板 (3.5 吋竖屏/256K 色/反光型 a-Si/半透型 a-Si TFT LCD)

S3C2440A 支持以下 SEC TFT LCD 面板。

1. SAMSUNG 3.5 吋竖屏/256K 色/反光型 a-Si TFT LCD。

LTS350Q1-PD1 : 带触摸屏和前光源单元的 TFT LCD 面板

LTS350Q1-PD2 : 只是 TFT LCD 面板

2. SAMSUNG 3.5 吋竖屏/256K 色/半透型 a-Si TFT LCD。

LTS350Q1-PE1 : 带触摸屏和前光源单元的 TFT LCD 面板

LTS350Q1-PE2 : 只是 TFT LCD 面板

S3C2440A 提供了使用 LTS350Q1-PD1 / PD2 和 LTS350Q1-PE1 / PE2 的如下时序信号

LTS350Q1-PD1 / PD2	LTS350Q1-PE1 / PE2
STH : 水平开始脉冲 TP : 源驱动器数据加载脉冲 INV : 数字数据反转 LCD_HCLK : 水平采样时钟 CPV : 垂直移位时钟 STV : 垂直开始脉冲 OE : 门打开使能 REV : 反转信号 REVB : 反转信号	STH : 水平开始脉冲 TP : 源驱动器数据加载脉冲 INV : 数字数据反转 LCD_HCLK : 水平采样时钟 CPV : 垂直移位时钟 STV : 垂直开始脉冲 LCCINV : 源驱动器 IC 采样反转信号 REV : VCOM 调制信号 REVB : 反转信号

因此 LTS350Q1-PD1/2 和 PE1/2 可以无需另加时序控制逻辑的连接到 S3C2440A。但是用户应该还是应用 Vcom 发生器电路 , 个别电压 ,INV 信号和灰度电压发生电路 , 这些是由 LTS350Q1-PD1/2 和 PE1/2 的产品信息(规格) 所推荐的。详细时序示意图也在 LTS350Q1-PD1/2 和 PE1/2 的产品信息 (规格) 中描述了。

相关文档 (LTS350Q1-PD1/2 和 PE1/2 的产品信息), 由三星电子有限公司的 AMLCD 客户技术中心制订。

注意 :

- S3C2440A 含有 HCLK, 其作为 AHB 总线的时钟。
- SEC TFT LCD 面板 (LTS350Q1-PD1/2 和 PE1/2) 含有水平采样时钟 (HCLK)。
- 这两个 HCLK 可能会引起混淆。因此, 请注意 S3C2440A 的 HCLK 与其它为 LCD_HCLK 的 LTS350 的 HCLK。请检查 SEC TFT LCD 面板 (LTS350Q1-PD1/2 和 PE1/2) 的 HCLK 被改为 LCD_HCLK。

虚拟显示 (TFT/STN)

S3C2440A 支持硬件水平或垂直滚屏。要实现滚屏，需要改变 LCDSADDR1/2 寄存器中 LCDBASEU 和 LCDBASEL 的字段（见图 15-8），删除 PAGEWIDTH 和 OFFSIZE 的值。

储存在视频缓冲器中的图像应该在尺寸上大于 LCD 面板屏幕。

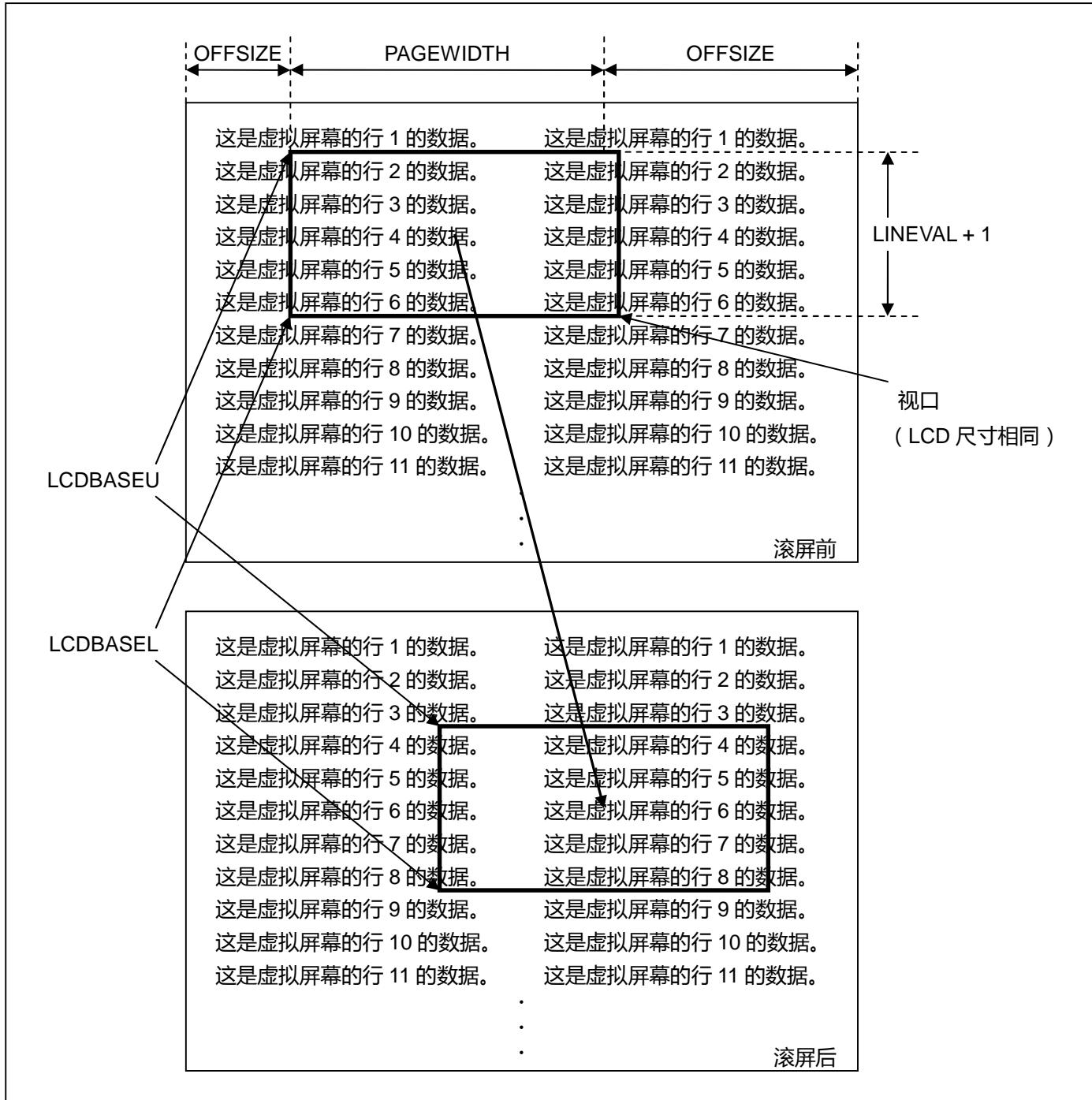


图 15-7. 虚拟显示中滚屏的例子 (单扫描)

LCD 电源使能 (STN/TFT)

S3C2440A 提供了电源使能 (PWREN) 功能。当 PWREN 设置为使得 PWREN 信号使能时 , LCD_PWREN 引脚的输出值被 ENVID 控制。换句话说 , 如果 LCD_PWREN 引脚连接了 LCD 面板的电源开/关控制引脚 , LCD 面板的电源将自动的由 ENVID 的设置来控制。

S3C2440A 同样支持 INVPWREN 位来反转 PWREN 信号的极性。

此功能只在当 LCD 面板拥有其自己的电源开/关控制端口并且当端口连接到了 LCD_PWREN 引脚时才可用。

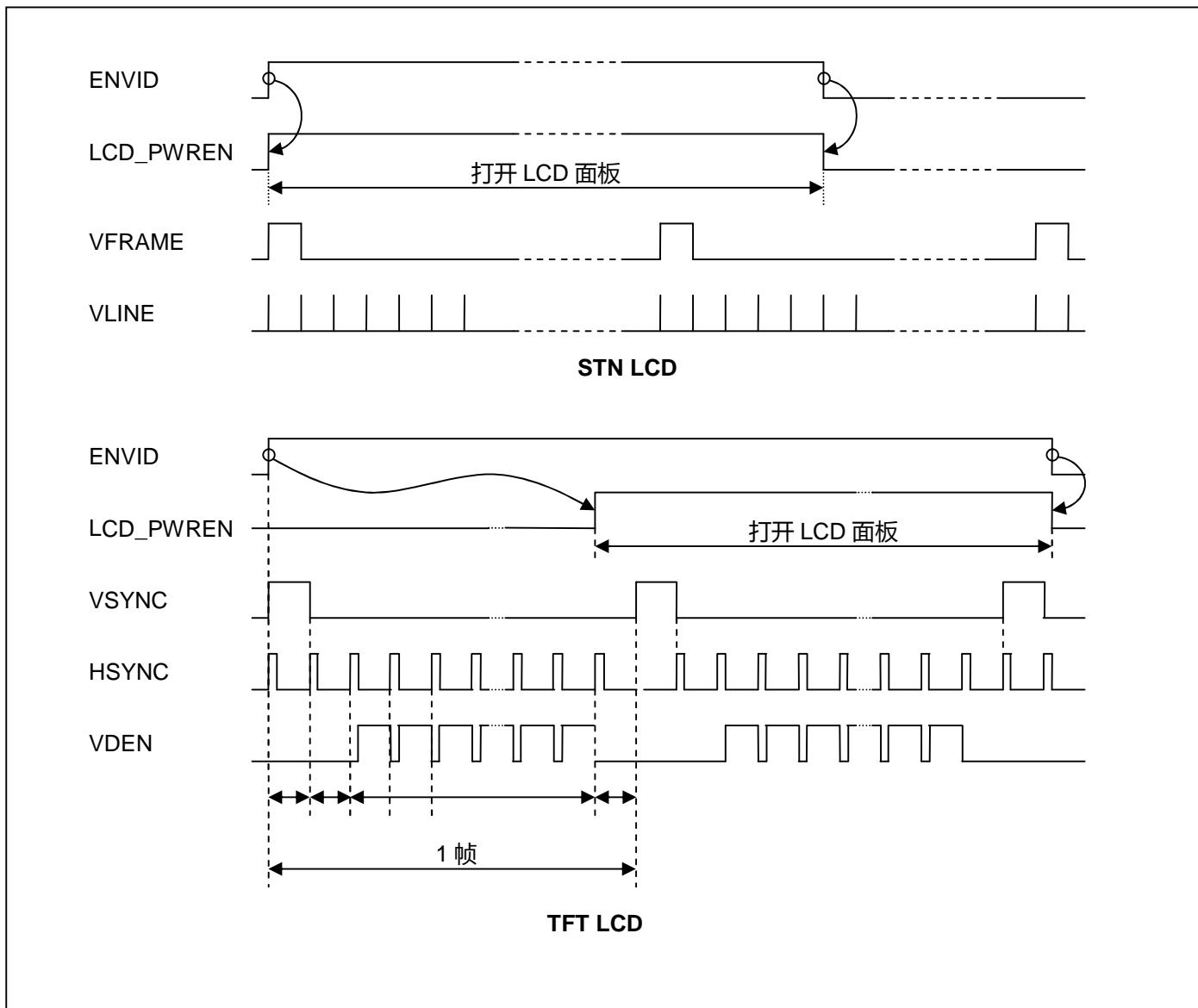


图 15-8. 功能的例子 (PWREN=1 , INVPWREN=0)

LCD 控制器特殊寄存器

LCD 控制 1 寄存器

寄存器	地址	R/W	描述	复位值
LCDCON1	0X4D000000	R/W	LCD 控制 1 寄存器	0x00000000
LCDCON1		位	描述	初始状态
LINECNT(只读)	[27:18]	提供行计数器的状态。从 LINEVAL 递减计数到 0。		0000000000
CLKVAL	[17:8]	决定 VCLK 的频率和 CLKVAL[9:0]。 STN : VCLK = HCLK / (CLKVAL × 2) (CLKVAL ≥ 2) TFT : VCLK = HCLK / [(CLKVAL + 1) × 2] (CLKVAL ≥ 0)		0000000000
MMODE	[7]	决定 VM 的触发频率 0 = 每帧 1 = 由 MVAL 定义此频率		0
PNRMODE	[6:5]	选择显示模式 00 = 4 位双扫描显示模式 (STN) 01 = 4 位单扫描显示模式 (STN) 10 = 8 位单扫描显示模式 (STN) 11 = TFT LCD 面板		00
BPPMODE	[4:1]	选择 BPP (位每像素) 模式 0000 = STN 的 1 bpp , 单色模式 0010 = STN 的 4 bpp , 16 阶灰度模式 0100 = STN 的封装 12 bpp , 彩色模式 (4096 色) 0101 = STN 的未封装 12 bpp , 彩色模式 (4096 色) 0110 = STN 的封装 16 bpp , 彩色模式 (4096 色) 1000 = TFT 的 1 bpp 1010 = TFT 的 4 bpp 1100 = TFT 的 16 bpp 0001 = STN 的 2 bpp , 4 阶灰度模式 0011 = STN 的 8 bpp , 彩色模式 (256 色) 1001 = TFT 的 2 bpp 1011 = TFT 的 8 bpp 1101 = TFT 的 24 bpp		0000
ENVID	[0]	LCD 视频输出和逻辑使能/禁止。 0 = 禁止视频输出和 LCD 控制信号 1 = 允许视频输出和 LCD 控制信号		0

LCD 控制 2 寄存器

寄存器	地址	R/W	描述	复位值
LCDCON2	0X4D000004	R/W	LCD 控制 2 寄存器	0x00000000

LCDCON2	位	描述	初始状态
VBPD	[31:24]	TFT : 垂直后沿为帧开始时 , 垂直同步周期后的无效行数。 STN : STN LCD 时应该设置此位为 0。	0x00
LINEVAL	[23:14]	TFT/STN : 此位决定了 LCD 面板的垂直尺寸。	0000000000
VFPD	[13:6]	TFT : 垂直前沿为帧结束时 , 垂直同步周期前的无效行数。 STN : STN LCD 时应该设置此位为 0。	00000000
VSPW	[5:0]	TFT : 通过计算无效行数垂直同步脉冲宽度决定 VSYNC 脉冲的高电平宽度。 STN : STN LCD 时应该设置此位为 0。	000000

LCD 控制 3 寄存器

寄存器	地址	R/W	描述	复位值
LCDCON3	0X4D000008	R/W	LCD 控制 3 寄存器	0x00000000

LCDCON3	位	描述	初始状态
HBPD (TFT)	[25:19]	TFT 水平后沿为 HSYNC 的下降沿与有效数据的开始之间的 VCLK 周期数。	0000000
WDLY (STN)		STN : WDLY[1:0]位通过计数 HCLK 数来决定 VLINE 与 VCLK 之间的延迟。保留 WDLY[7:2]。 00 = 16 HCLK 01 = 32 HCLK 10 = 48 HCLK 11 = 64 HCLK	
HOZVAL	[18:8]	TFT/STN : 此位决定了 LCD 面板的水平尺寸。必须决定 HOZVAL 来满足 1 行的总字节为 4n 字节。如果单色模式中 LCD 的 x 尺寸为 120 个点，但不能支持 x=120，因为 1 行是由 16 字节 (2n) 所组成。LCD 面板驱动器将舍弃额外的 8 个点。	000000000000
HFPD (TFT)	[7:0]	TFT 水平后沿为有效数据的结束与 HSYNC 的上升沿之间的 VCLK 周期数。	0X00
LINEBLANK (STN)		STN : 此位表明一次水平行持续时间中的空时间。此位微调 VLINE 的频率。LINEBLANK 的单位为 HCLK×8。 例子) 如果 LINEBLANK 的值为 10，则在 80 个 HCLK 期间插入空时间到 VCLK。	

编程注释：STN LCD 情况中，(LINEBLANK + WLH + WDLY)值应该大于(14+12Tmax)。

$$(LINEBLANK + WLH + WDLY) \geq (14 + 8 \times T_{max1} + 4 \times T_{max2}) = 14 + 12 \times T_{max}$$

说明：

- (1) 14: SDRAM 自动刷新总线获取周期
- (2) $8 \times T_{max1}$: 缓存填充周期 x 最慢存储器存取时间 (例如, ROM)
- (3) $4 \times T_{max2}$: 0xC 至 0xE 地址帧存储器存取时间

LCD 控制 4 寄存器

寄存器	地址	R/W	描述	复位值
LCDCON4	0X4D00000C	R/W	LCD 控制 4 寄存器	0x00000000

LCDCON4	位	描述	初始状态
MVAL	[15:8]	STN : 此位定义如果 MMODE 位被置位为逻辑'1'的 VM 信号将要触发的频率。	0x00
HSPW (TFT)	[7:0]	TFT 通过计算 VCLK 的数水平同步脉冲宽度决定 HSYNC 脉冲的高电平宽度	0X00
WLH (STN)		STN 通过计算 HCLK 的数 WLH[1:0]位决定 VLINE 脉冲的高电平宽度。 保留 WLH[7:2]。 00 = 16 HCLK 01 = 32 HCLK 10 = 48 HCLK 11 = 64 HCLK	

LCD 控制 5 寄存器

寄存器	地址	R/W	描述	复位值
LCDCON5	0X4D000010	R/W	LCD 控制 5 寄存器	0x00000000

LCDCON5	位	描述	初始状态
保留	[31:17]	保留此位并且应该为'0'	0
VSTATUS	[16:15]	TFT : 垂直状态 (只读) 00 = VSYNC 01 = 后沿 10 = ACTIVE 11 = 前沿	00
HSTATUS	[14:13]	TFT : 水平状态 (只读) 00 = VSYNC 01 = 后沿 10 = ACTIVE 11 = 前沿	00
BPP24BL	[12]	TFT : 此位决定 24 bpp 视频存储器的顺序 0 = LSB 有效 1 = MSB 有效	0
FRM565	[11]	TFT : 此位选择 16 bpp 输出视频数据的格式 0 = 5:5:5:1 格式 0 = 5:6:5 格式	0
INVVCLK	[10]	STN/TFT : 此位控制 VCLK 有效沿的极性 0 = VCLK 下降沿取视频数据 1 = VCLK 上升沿取视频数据	0
INVVLINe	[9]	STN/TFT : 此位表明 VLINE/HSYNC 脉冲极性 0 = 正常 1 = 反转	0
INVVFRAME	[8]	STN/TFT : 此位表明 VFRAME/VSYNC 脉冲极性 0 = 正常 1 = 反转	0
INVVD	[7]	STN/TFT : 此位表明 VD (视频数据) 脉冲极性 0 = 正常 1 = 反转 VD	0
INVVDEN	[6]	TFT : 此位表明 VDEN 信号极性 0 = 正常 1 = 反转	0
INVPWREN	[5]	STN/TFT : 此位表明 PWREN 信号极性 0 = 正常 1 = 反转	0
INVLEND	[4]	TFT : 此位表明 LEND 信号极性 0 = 正常 1 = 反转	0
PWREN	[3]	STN/TFT : LCD_PWREN 输出信号使能/禁止 0 = 禁止 PWREN 信号 1 = 允许 PWREN 信号	0
ENLEND	[2]	TFT : LEND 输出信号使能/禁止 0 = 禁止 LEND 信号 1 = 允许 LEND 信号	0
BSWP	[1]	STN/TFT : 字节交换控制位 0 = 交换禁止 1 = 交换使能	0
HWSWP	[0]	STN/TFT : 半字节交换控制位 0 = 交换禁止 1 = 交换使能	0

帧缓冲器开始地址 1 寄存器

寄存器	地址	R/W	描述	复位值
LCDSADDR1	0X4D000014	R/W	STN/TFT : 帧缓冲器开始地址 1 寄存器	0x00000000

LCDSADDR1	位	描述	初始状态
LCD BANK	[29:21]	这些位表明系统存储器中视频缓冲器的 bank 位置的 A[30:22]。即使当移动视口时也不能改变 LCD BANK 的值。LCD 帧缓冲器应该在 4MB 连续区域内，以保证当移动视口时不会改变 LCD BANK 的值。因此应该谨慎使用 malloc() 函数。	0x00
LCD BASEU	[20:0]	对于双扫描 LCD：这些位表明递增地址计数器的开始地址的 A[21:1]，它是用于双扫描 LCD 的递增帧存储器或单扫描 LCD 的帧存储器。 对于单扫描 LCD：这些位表明 LCD 帧缓冲器的开始地址的 A[21:1]。	0x000000

帧缓冲器开始地址 2 寄存器

寄存器	地址	R/W	描述	复位值
LCDSADDR2	0X4D000018	R/W	STN/TFT : 帧缓冲器开始地址 2 寄存器	0x00000000

LCDSADDR2	位	描述	初始状态
LCD BASEL	[20:0]	对于双扫描 LCD：这些位表明递减地址计数器的开始地址的 A[21:1]，它是用于双扫描 LCD 的递减帧存储器。 对于单扫描 LCD：这些位表明 LCD 帧缓冲器的结束地址的 A[21:1]。 $\text{LCD BASEL} = ((\text{帧结束地址}) \gg 1) + 1$ $= \text{LCD BASEU} + (\text{PAGEWIDTH+OFFSIZE}) \times (\text{LINEVAL}+1)$	0x0000

注意：

当 LCD 控制器为打开时用户可以改变 LCD BASEU 和 LCD BASEL 的值来实现滚屏。但是用户一定不要在帧中改变前为了 LCD FIFO 取得下一帧数据，而在帧的结束时通过有关 LCD CON1 寄存器中 LINECNT 字段来改变 LCD BASEU 和 LCD BASEL 寄存器的值。

因此如果改变了帧，预取 FIFO 数据将为过时的并且 LCD 控制器将显示一个错误屏幕。为了检查 LINECNT 应该屏蔽中断。如果正好在读取 LINECNT 之后执行了任何中断，因为中断服务程序 (ISR) 的执行时间则读取到的 LINECNT 值可能为过时的。

帧缓冲器开始地址 3 寄存器

寄存器	地址	R/W	描述	复位值
LCDSADDR3	0X4D00001C	R/W	STN/TFT : 虚拟屏地址设置	0x00000000

LCDSADDR3	位	描述	初始状态
OFFSIZE	[21:11]	虚拟屏偏移尺寸 (半字数)。 此值定义了显示在之前 LCD 行的最后半字的地址与要在新 LCD 行中显示的第一半字的地址之间的差。	000000000000
PAGEWIDTH	[10:0]	虚拟屏页宽度 (半字数)。 此值定义了帧中视口的宽度。	0000000000

注意：

当 ENVID 位为 0 时必须改变 PAGEWIDTH 和 OFFSIZE 的值。

例 1. LCD 面板 = 320 × 240 , 16 阶灰度 , 单扫描

帧开始地址 = 0x0C500000

偏移点数 = 2048 个点 (512 半字)

LINEVAL = 240 - 1 = 0xEF

PAGEWIDTH = 320 × 4 / 16 = 0x50

OFFSIZE = 512 = 0x200

LCDBANK = 0x0C500000 >> 22 = 0x31

LCDBASEU = 0x100000 >> 1 = 0x80000

LCDBASEL = 0x80000 + (0x50 + 0x200) × (0xEF + 1) = 0xA2B00

例 2. LCD 面板 = 320 × 240 , 16 阶灰度 , 双扫描

帧开始地址 = 0x0C500000

偏移点数 = 2048 个点 (512 半字)

LINEVAL = 120-1 = 0x77

PAGEWIDTH = 320 × 4 / 16 = 0x50

OFFSIZE = 512 = 0x200

LCDBANK = 0x0C500000 >> 22 = 0x31

LCDBASEU = 0x100000 >> 1 = 0x80000

LCDBASEL = 0x80000 + (0x50 + 0x200) × (0x77 + 1) = 0x91580

例 2. LCD 面板 = 320 × 240 , 彩色 , 单扫描

帧开始地址 = 0x0C500000

偏移点数 = 1024 个点 (512 半字)

LINEVAL = 240 - 1 = 0xEF

PAGEWIDTH = 320 × 8 / 16 = 0xA0

OFFSIZE = 512 = 0x200

LCDBANK = 0x0C500000 >> 22 = 0x31

LCDBASEU = 0x100000 >> 1 = 0x80000

LCDBASEL = 0x80000 + (0xA0 + 0x200) × (0xEF + 1) = 0xA7600

红色查找表寄存器

寄存器	地址	R/W	描述	复位值
REDLUT	0X4D000020	R/W	STN : 红色查找表寄存器	0x00000000

REDLUT	位	描述	初始状态
REDVAL	[31:0]	这些位定义了将在由 8 种可能的红色组合中选择哪 16 色深 000 = REDVAL[3:0] 001 = REDVAL[7:4] 010 = REDVAL[11:8] 011 = REDVAL[15:12] 100 = REDVAL[19:16] 101 = REDVAL[23:20] 110 = REDVAL[27:24] 111 = REDVAL[31:28]	0x00000000

绿色查找表寄存器

寄存器	地址	R/W	描述	复位值
GREENLUT	0X4D000024	R/W	STN : 绿色查找表寄存器	0x00000000

GREENLUT	位	描述	初始状态
GREENVAL	[31:0]	这些位定义了将在由 8 种可能的绿色组合中选择哪 16 色深 000 = GREENVAL[3:0] 001 = GREENVAL[7:4] 010 = GREENVAL[11:8] 011 = GREENVAL[15:12] 100 = GREENVAL[19:16] 101 = GREENVAL[23:20] 110 = GREENVAL[27:24] 111 = GREENVAL[31:28]	0x00000000

蓝色查找表寄存器

寄存器	地址	R/W	描述	复位值
BLUELUT	0X4D000028	R/W	STN : 蓝色查找表寄存器	0x0000

BLUELUT	位	描述	初始状态
BLUEVAL	[15:0]	这些位定义了将在由 4 种可能的蓝色组合中选择哪 16 色深 00 = BLUEVAL[3:0] 01 = BLUEVAL[7:4] 10 = BLUEVAL[11:8] 11 = BLUEVAL[15:12]	0x0000

注意：

不要使用从 0x14A0002C 到 0x14A00048 的地址。这些区域保留给测试模式。

抖动模式寄存器

寄存器	地址	R/W	描述	复位值
DITHMODE	0X4D00004C	R/W	STN : 抖动模式寄存器。 此寄存器复位值为 0x00000 , 但是用户可以改变此值为 0x12210。(此寄存器的最后值参考样本源程序)	0x00000

DITHMODE	位	描述	初始状态
DITHMODE	[18:0]	使用以下值给 LCD : 0x00000 或 0x12210	0x00000

临时调色板寄存器

寄存器	地址	R/W	描述	复位值
TPAL	0X4D000050	R/W	TFT : 临时调色板寄存器。 此寄存器的值将为下帧的视频数据	0x00000000

TPAL	位	描述	初始状态
TPALEN	[24]	临时调色板寄存器使能位。 0 = 禁止 0 = 开始	0
TPALVAL	[23:0]	临时调色板值寄存器。 TPALVAL[23:16] : 红色 TPALVAL[15:8] : 绿色 TPALVAL[7:0] : 蓝色	0x000000

LCD 中断挂起寄存器

寄存器	地址	R/W	描述	复位值
LCDINTPND	0X4D000054	R/W	表明 LCD 中断挂起寄存器	0x0

LCDINTPND	位	描述	初始状态
INT_FrSyn	[1]	LCD 帧同步中断挂起位。 0 = 未请求中断 1 = 帧发出中断请求	0
INT_FiCnt	[0]	LCD FIFO 中断挂起位。 0 = 未请求中断 1 = 当 LCD FIFO 到达触发深度十请求 LCD FIFO 中断	0

LCD 源挂起寄存器

寄存器	地址	R/W	描述	复位值
LCDSRCPND	0X4D000058	R/W	表明 LCD 中断源挂起寄存器	0x0

LCDSRCPND	位	描述	初始状态
INT_FrSyn	[1]	LCD 帧同步中断源挂起位。 0 = 未请求中断 1 = 帧发出中断请求	0
INT_FiCnt	[0]	LCD FIFO 中断源挂起位。 0 = 未请求中断 1 = 当 LCD FIFO 到达触发深度十请求 LCD FIFO 中断	0

LCD 中断屏蔽寄存器

寄存器	地址	R/W	描述	复位值
LCDINTMSK	0X4D00005C	R/W	决定屏蔽哪个中断源。 被屏蔽的中断源将不会被服务。	0x3

LCDINTMSK	位	描述	初始状态
FIWSEL	[2]	决定 LCD FIFO 的触发深度。 0 = 4 字 1 = 8 字	
INT_FrSyn	[1]	屏蔽 LCD 帧同步中断。 0 = 中断服务可用 1 = 屏蔽中断服务	1
INT_FiCnt	[0]	屏蔽 LCD FIFO 中断 0 = 中断服务可用 1 = 屏蔽中断服务	1

TCON 控制寄存器

寄存器	地址	R/W	描述	复位值
TCONSEL	0X4D000060	R/W	此寄存器控制 LPC3600/LCC3600 模式	0xF84

TCONSEL	位	描述	初始状态
LCC_TEST2	[11]	LCC3600 测试模式 2 (只读)	1
LCC_TEST1	[10]	LCC3600 测试模式 1 (只读)	1
LCC_SEL5	[9]	选择 STV 极性	1
LCC_SEL4	[8]	选择 CPV 信号引脚 0	1
LCC_SEL3	[7]	选择 CPV 信号引脚 1	1
LCC_SEL2	[6]	选择行/点反转	0
LCC_SEL1	[5]	选择 DG/普通模式	0
LCC_EN	[4]	决定 LCC3600 使能/禁止 0 = LCC3600 禁止 1 = LCC3600 使能	0
CPV_SEL	[3]	选择 CPV 脉冲低电平宽度	0
MODE_SEL	[2]	选择 DE/同步模式 0 = 同步模式 1 = DE 模式	1
RES_SEL	[1]	选择输出分辨率类型 0 = 320×240 1 = 240×320	0
LPC_EN	[0]	决定 LPC3600 使能/禁止 0 = LPC3600 禁止 1 = LPC3600 使能	0

注意：

不允许 LPC_EN 和 LCC_EN 都使能。同一时间只有一个 TCON 可以被使能。

寄存器设置向导 (STN)

LCD 控制器可以通过特殊寄存器设置来支持多种屏幕尺寸。

CLKVAL 的值决定了 VCLK 的频率。必须决定此值使得 VCLK 值为大于数据传输率。LCD 控制器的 VD 端口的数据传输率被用于决定 CLKVAL 寄存器的值。

数据传输率由以下等式给出：

$$\text{数据传输率} = HS \times VS \times FR \times MV$$

- HS : 水平 LCD 尺寸
- VS : 垂直 LCD 尺寸
- FR : 帧频
- MV : 模式依赖值

表 15-6. 每种显示模式的 MV 值

模式	MV 值
单色 , 4 位单扫描显示	1/4
单色 , 8 位单扫描显示或 4 位双扫描显示	1/8
4 阶灰度 , 4 位单扫描显示	1/4
4 阶灰度 , 8 位单扫描显示或 4 位双扫描显示	1/8
16 阶灰度 , 4 位单扫描显示	1/4
16 阶灰度 , 8 位单扫描显示或 4 位双扫描显示	1/8
彩色 , 4 位单扫描显示	3/4
彩色 , 8 位单扫描显示或 4 位双扫描显示	3/8

LCDBASEU 寄存器值为帧缓冲器的第一个地址值。burst4 字存取时必须忽略低 4 位。LCDBASEL 寄存器值随 LCD 尺寸和 LCDBASEU 而定。LCDBASEL 的值由以下等式给出：

$$- \quad \text{LCDBASEL} = \text{LCDBASEU} + \text{LCDBASEL} \text{ 偏移}$$

例 1 :

160 × 160、4 阶灰度、80 帧/秒、4 位单扫描显示、HCLK 频率为 60MHz、WLH = 1、WDLY = 1

数据传输率 = $160 \times 160 \times 80 \times 1/4 = 512\text{kHz}$

CLKVAL = 58 , VCLK = 517KHz

HOZVAL = 39 , LINEVAL = 159

LINEBLANK = 10

LCDBASEL = LCDBASEU + 3200

注意 :

系统负载越高 , CPU 性能就越差

例 2 (虚拟屏寄存器):

4 阶灰度、虚拟屏尺寸 = 1024×1024 、LCD 尺寸 = 320×240 、LCDBASEU = 0x64、4 位双扫描

1 个半字 = 8 像素 (4 阶灰度)

虚拟屏 1 行 = 128 个半字 = 1024 个像素

LCD 1 行 = 320 个像素 = 40 个半字

OFFSIZE = $128 - 40 = 88 = 0x58$

PAGEWIDTH = 40 = 0x28

LCDBASEL = LCDBASEU + (PAGEWIDTH + OFFSIZE) × (LINEVAL + 1) = $100 + (40 + 88) \times 120 = 0x3C64$

灰度选择向导

S3C2440A LCD 控制器可以使用帧频控制(FRC)来产生 16 阶灰度。FRC 特性可能会引起未料到的灰度类型。这些多余的错误类型可能在快相应 LCD 或低帧频时表现出来。

由于 LCD 灰度的品质 (*quality*) 是由 LCD 自己的特性而定 , 用户必须在观察了用户自己的 LCD 的所有灰度后选择一个合适灰度。

通过以下步骤选择灰度品质 :

1. 从 SAMSUNG 获取最新抖动方式寄存器值。
2. 在 LCD 中显示 16 条灰度栅。
3. 改变帧频到最佳值。
4. 改变 VM 交替周期以获取最好品质。
5. 观察 16 条灰度栅 , 选择一条在 LCD 上显示好的灰度。
6. 只使用好的灰度给品质

LCD 刷新总线带宽计算向导

S3C2440A LCD 控制器可以支持多种 LCD 显示尺寸。为了选择合适的尺寸 (为无闪烁 LCD 系统应用), 用户必须考虑由 LCD 显示尺寸决定的 LCD 刷新总线带宽、位每像素 (bpp) 帧频、存储器总线宽度、存储器类型等。

$$\text{LCD 数据率 (字节/秒)} = \text{bpp} \times (\text{水平显示尺寸}) \times (\text{垂直显示尺寸}) \times (\text{帧频}) / 8$$

$$\text{LCD DMA 突发计数 (次数/秒)} = \text{LCD 数据率 (字节/秒)}; \text{LCD DMA 使用 4 字 (16 字节) 突发}$$

P_{dma} 意思为 LCD DMA 存取周期。换句话说 , P_{dma} 的值表示视频数据获取的 4 拍突发 (4 字突发) 的周期。因此 P_{dma} 随存储器类型和存储器设置而定。

最后 , LCD 系统是负载由 LCD DMA 突发计数和 P_{dma} 决定。

$$- \quad \text{LCD 系统负载} = \text{LCD DMA 突发计数} \times P_{\text{dma}}$$

例 3 :

640 × 480 、 8bpp 、 60 帧/秒、 16 位数据宽度、 SDRAM ($\text{Trp} = 2\text{HCLK} / \text{Trcd} = 2\text{HCLK} / \text{CL}=2\text{HCLK}$) 并且 HCLK 频率为 60 MHz

$$\text{LCD 数据率} = 8 \times 640 \times 480 \times 60 / 8 = 18.432 \text{Mbyte/s}$$

$$\text{LCD DMA 突发计数} = 18.432 / 16 = 1.152 \text{M/s}$$

$$P_{\text{dma}} = (\text{Trp} + \text{Trcd} + \text{CL} + (2 \times 4) + 1) \times (1/60\text{MHz}) = 0.250\text{ms}$$

$$\text{LCD 系统负载} = 1.152 \times 250 = 0.288$$

$$\text{系统总线占用率} = (0.288/1) \times 100 = 28.8\%$$

寄存器设置向导 (TFT LCD)

CLKVAL 寄存器的值决定了 VCLK 的频率和帧频。

$$\text{帧频} = 1 / [\{ (VSPW+1) + (VBPD+1) + (LINEVAL + 1) + (VFPD+1) \} \times \{ (HSPW+1) + (HBPD +1) \\ + (HFDPD+1) + (HOZVAL + 1) \} \times \{ 2 \times (CLKVAL+1) / (HCLK) \}]$$

实际应用中，必须考虑系统时序，以避免由于存储器带宽竞争引起的 LCD 控制器的 FIFO 欠载状态。

例 4：

TFT 分辨率：240 × 240

VSPW = 2 , VBPD = 14 , LINEVAL = 239 , VFPD = 4

HSPW = 25 , HBPD = 15 , HOZVAL = 239 , HFDPD = 1

CLKVAL = 5

HCLK = 60 MHz

必须由 LCD 尺寸和驱动器规格参考以下参数：

VSPW、VBPD、LINEVAL、VFPD、HSPW、HBPD、HOZVAL 和 HFDPD

如果目标帧频为 60 到 70Hz，则 CLKVAL 应该为 5。

因此帧频 = 67Hz

16 模/数转换器及触摸屏接口

概述

10 位 CMOS ADC (模/数转换器) 是一个 8 通道模拟输入的再循环类型设备。其转换模拟输入信号为 10 位二进制数字编码 , 最大转换率为 2.5MHz A/D 转换器时钟下的 500 KSPS。A/D 转换器支持片上采样-保持功能和掉电模式的操作。

触摸屏接口可以控制/选择触摸屏 X 、 Y 方向的引脚 (XP , XM , YP , YM) 的变换。触摸屏接口包括触摸屏引脚控制逻辑和带中断发生逻辑的 ADC 接口逻辑。

特性

- 分辨率 : 10 位
- 差分线性误差 : $\pm 1.0 \text{ LSB}$
- 积分线性误差 : $\pm 2.0 \text{ LSB}$
- 最大转换率 : 500 KSPS
- 功耗低
- 供电电压 : 3.3V
- 模拟输入范围 : 0 至 3.3V
- 片上采样-保持功能
- 普通转换模式
- 分离的 X/Y 方向转换模式
- 自动 (顺序) X/Y 方向转换模式
- 等待中断模式

ADC 和触摸屏接口操作

方框图

图 16-1 显示了 A/D 转换器和触摸屏接口的功能方框图。注意 A/D 转换器设备是再循环类型的。

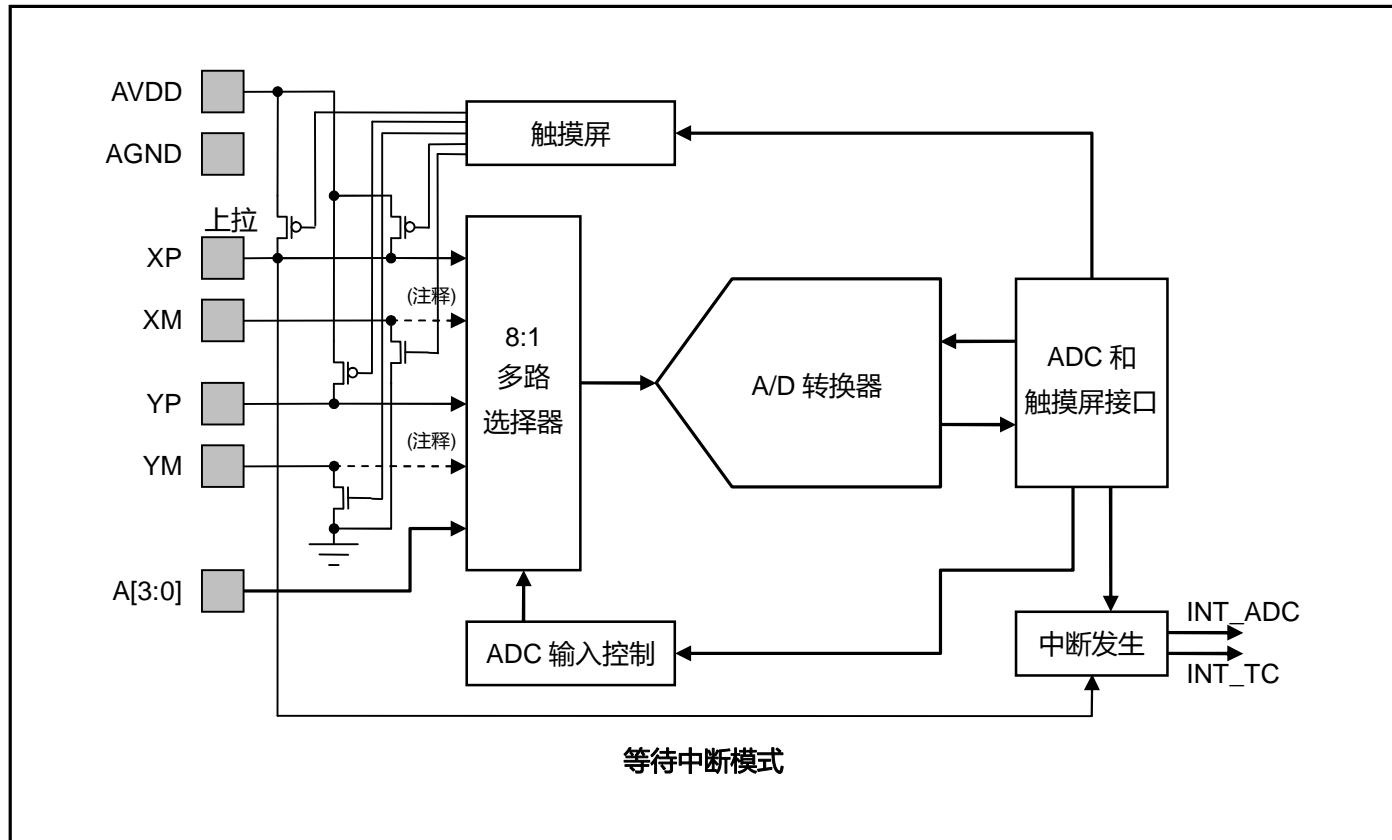


图 16-1. ADC 和触摸屏接口功能方框图

注释 : (记号)

当使用触摸屏设备时，触摸屏接口的 XM 或 YM 只连接到地。

当未使用触摸屏设备时，XM 或 YM 被连接到模拟输入信号给普通 ADC 转换。

功能描述

A/D 转换时间

当 PCLK 频率在 50MHz 并且预分频器的值为 49 时，共 10 位的转换时间为：

$$A/D \text{ 转换器频率} = 50\text{MHz} / (49+1) = 1\text{MHz}$$

$$\text{转换时间} = 1/(1\text{MHz} / 5 \text{ 周期}) = 1/200\text{KHz} = 5\mu\text{s}$$

注释：

此 A/D 转换器被设计为最高工作在 2.5MHz 时钟下，因此转换率可以达到 500 KSPS。

触摸屏接口模式

1. 普通转换模式

单转换模式是最合适的通用 ADC 转换。此模式可以通过设置 ADCCON (ADC 控制寄存器) 初始化并且通过读写 ADCDAT0 (ADC 数据寄存器 0) 就能够完成。

2. 分离的 X/Y 方向转换模式

触摸屏控制器可以工作在两个转换模式之一。方向转换模式如下方法操作。X 方向模式写 X 方向转换数据到 ADCDAT0，故触摸屏接口产生中断源给中断控制器。Y 方向模式写 Y 方向转换数据到 ADCDAT1，故触摸屏接口产生中断源给中断控制器。

3. 自动 (顺序) X/Y 方向转换模式

自动 (顺序) X/Y 方向转换模式操作如下。触摸屏控制器顺序变换触摸 X 方向和 Y 方向。在自动方向转变模式中触摸控制器在写入 X 测量数值到 ADCDAT0 和写入 Y 测量数值到 ADCDAT1 后，触摸屏接口产生中断源给中断控制器。

4. 等待中断模式

当笔尖落下时触摸屏控制器产生中断(INT_TC)信号。等待中断模式设置值为 rADCTSC=0xd3 ; // XP_PU ,XP_Dis , XM_Dis , YP_Dis , YM_En

触摸屏控制器产生中断信号 (INT_TC) 后，必须清除等待中断模式。(XY_PST 设置到无操作模式)

待机模式

当 ADCCON [2]被设置为'1'时激活待机模式。此模式中，停止 A/D 转换操作并且 ADCDAT0、ADCDAT1 寄存器包含的是先前转换的数据。

编程笔记

1. A/D 转换的数据可以通过中断或查询方式访问。中断方式的总体转换时间为从 A/D 转换器开始到转换数据的读取，可能由于中断服务程序的返回时间和数据访问时间而延迟。查询方式是通过检查转换结束标志位的 ADCCON[15]，可以确定读取 ADCDAT 寄存器的时间。

2. 还提供了其它启动 A/D 转换的方法。在转换的读启动模式 ADCCON[1]设置为 1 后，A/D 转换启动同时读取数据。

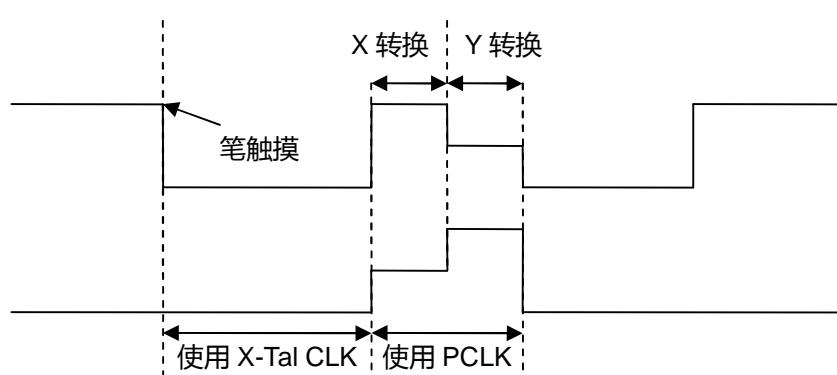


图 16-2. ADC 触摸屏操作信号

ADC 和触摸屏接口特殊寄存器

ADC 控制寄存器 (ADCCON)

寄存器	地址	R/W	描述	复位值
ADCCON	0x5800000	R/W	ADC 控制寄存器	0x3FC4

ADCCON	位	描述	初始状态
ECFLG	[15]	转换结束标志位 (只读) 0 = A/D 正在转换 1 = A/D 转换已结束	0
PRSCEN	[14]	A/D 转换器预分频器使能 0 = 禁止 1 = 使能	0
PRSCVL	[13:6]	A/D 转换器预分频值 数值范围 : 0 至 255 注意 : ADC 频率应该设置为低于 PCLK 的 1/5。(例如 PCLK=10MHz, 则 ADC 频率 <2MHz)	0xFF
SEL_MUX	[5:3]	模拟输入通道选择 000 = AIN 0 001 = AIN 1 100 = YM 101 = YP 010 = AIN 2 110 = XM 011 = AIN 3 111 = XP	0
STDBM	[2]	待机模式选择 0 = 正常工作模式 1 = 待机模式	1
READ_START	[1]	读启动 A/D 转换 0 = 禁止读启动操作 1 = 使能读启动操作	0
ENABLE_START	[0]	使能 A/D 转换启动。如果 READ_START 为使能，则此值无效。 0 = 无操作 1 = A/D 转换启动且此位在启动后被清零。	0

注意 :

- 当触摸屏引脚 (YM、YP、XM 和 XP) 为禁止时，这些端口可以被用于 ADC 的模拟输入端口 (AIN4、AIN5、AIN6 和 AIN7)。
- 当从待机模式中变换到正常工作模式时，ADC 的预分频器必须在最后的 3 个 ADC 时钟前使能。

ADC 触摸屏控制寄存器 (ADCTSC)

寄存器	地址	R/W	描述	复位值
ADCTSC	0x5800004	R/W	ADC 触摸屏控制寄存器	0x58

ADCTSC	位	描述	初始状态
UD_SEN	[8]	检测笔尖起落状态 0 = 检测笔尖落下中断信号 1 = 检测笔尖抬起中断信号	0
YM_SEN	[7]	YM 开关使能 0 = YM 输出驱动器禁止 1 = YM 输出驱动器使能	0
YP_SEN	[6]	YP 开关使能 0 = YP 输出驱动器使能 1 = YP 输出驱动器禁止	1
XM_SEN	[5]	XM 开关使能 0 = XM 输出驱动器禁止 1 = XM 输出驱动器使能	0
XP_SEN	[4]	XP 开关使能 0 = XP 输出驱动器使能 1 = XP 输出驱动器禁止	1
PULL_UP	[3]	上拉开关使能 0 = XP 上拉使能 1 = XP 上拉禁止	1
AUTO_PST	[2]	自动顺序 X 方向和 Y 方向转换 0 = 正常 ADC 转换 1 = 自动顺序 X 方向和 Y 方向测量	0
XY_PST	[1:0]	手动测量 X 方向或 Y 方向 00 = 无操作模式 01 = X 方向测量 10 = Y 方向测量 11 = 等待中断模式	0

注意：

- 当等待触摸屏中断时，XP_SEN 位应该被设置为‘1’（XP 输出禁止）并且 PULL_UP 位应该被设置为 ‘0’（XP 上拉使能）。
- 只有在自动顺序 X/Y 方向转换时 AUTO_PST 位应该被设置为‘1’。
- 在睡眠模式期间应该分离 XP、YP 与 GND 源以避免漏电电流。因为 XP、YP 将在睡眠模式中保持为‘H’状态。X/Y 方向转换的触摸屏引脚状态。

	XP	XM	YP	YM	ADC 通道选择
X 方向	V _{ref}	GND	高阻	高阻	YP
Y 方向	高阻	高阻	V _{ref}	GND	XP

ADC 启动延时寄存器 (ADCDLY)

寄存器	地址	R/W	描述	复位值
ADCDLY	0x5800008	R/W	ADC 启动或初始化延时寄存器	0x00ff

ADCDLY	位	描述	初始状态
DELAY	[15:0]	正常转换模式、XY 方向模式、自动方向模式 →ADC 转换启动延时值。 注意：不要使用 0 这个值 (0x0000)	00ff

ADC 转换数据寄存器 (ADCDAT0)

寄存器	地址	R/W	描述	复位值
ADCDAT0	0x580000C	R	ADC 转换数据寄存器	-

ADCDAT0	位	描述	初始状态
UPDOWN	[15]	等待中断模式中笔尖的起落状态 0 = 笔尖落下态 1 = 笔尖抬起态	-
AUTO_PST	[14]	自动顺序 X 方向和 Y 方向转换 0 = 正常 ADC 转换 1 = 顺序 X 方向、Y 方向测量	-
XY_PST	[13:12]	手动 X 方向或 Y 方向测量 00 = 无操作模式 01 = X 方向测量 10 = Y 方向测量 11 = 等待中断模式	-
保留	[11:10]	保留	-
XPDATA (正常 ADC)	[9:0]	X 方向转换数值 (包括正常 ADC 转换数值) 数值范围 : 0 至 3FF	-

ADC 转换数据寄存器 (ADCCDAT1)

寄存器	地址	R/W	描述	复位值
ADCCDAT1	0x5800010	R	ADC 转换数据寄存器	-

ADCCDAT1	位	描述	初始状态
UPDOWN	[15]	等待中断模式中笔尖的起落状态 0 = 笔尖落下态 1 = 笔尖抬起态	-
AUTO_PST	[14]	自动顺序 X 方向和 Y 方向转换 0 = 正常 ADC 转换 1 = 顺序 X 方向、Y 方向测量	-
XY_PST	[13:12]	手动 X 方向或 Y 方向测量 00 = 无操作模式 01 = X 方向测量 10 = Y 方向测量 11 = 等待中断模式	-
保留	[11:10]	保留	-
YPDATA	[9:0]	Y 方向转换数值 数值范围 : 0 至 3FF	-

ADC 触摸屏起落中断检测寄存器 (ADCUPDN)

寄存器	地址	R/W	描述	复位值
ADCUPDN	0x5800014	R/W	笔尖抬起或落下中断状态寄存器	0x0

ADCUPDN	位	描述	初始状态
TSC_UP	[1]	笔尖抬起中断 0 = 无笔尖抬起状态 1 = 笔尖抬起中断发生	0
TSC_DN	[0]	笔尖落下中断 0 = 无笔落下起状态 1 = 笔尖落下中断发生	0

17 实时时钟

概述

实时时钟 (RTC) 单元可以在当系统电源关闭后通过备用电池工作。RTC 可以通过使用 STRB/LDRB ARM 操作发送 8 位二-十进制交换码 (BCD) 值数据给 CPU。这些数据包括年、月、日、星期、时、分和秒的时间信息。RTC 单元工作在外部 32.768kHz 晶振并且可以执行闹钟功能。

特性

- BCD 数：年、月、日、星期、时、分和秒
- 闰年发生器
- 闹钟功能：闹钟中断或从掉电模式唤醒
- 已解决的 2000 年问题
- 独立电源引脚 (RTCVDD)
- 支持 RTOS 内核时钟节拍 (tick) 的毫秒节拍时间中断

实时时钟操作

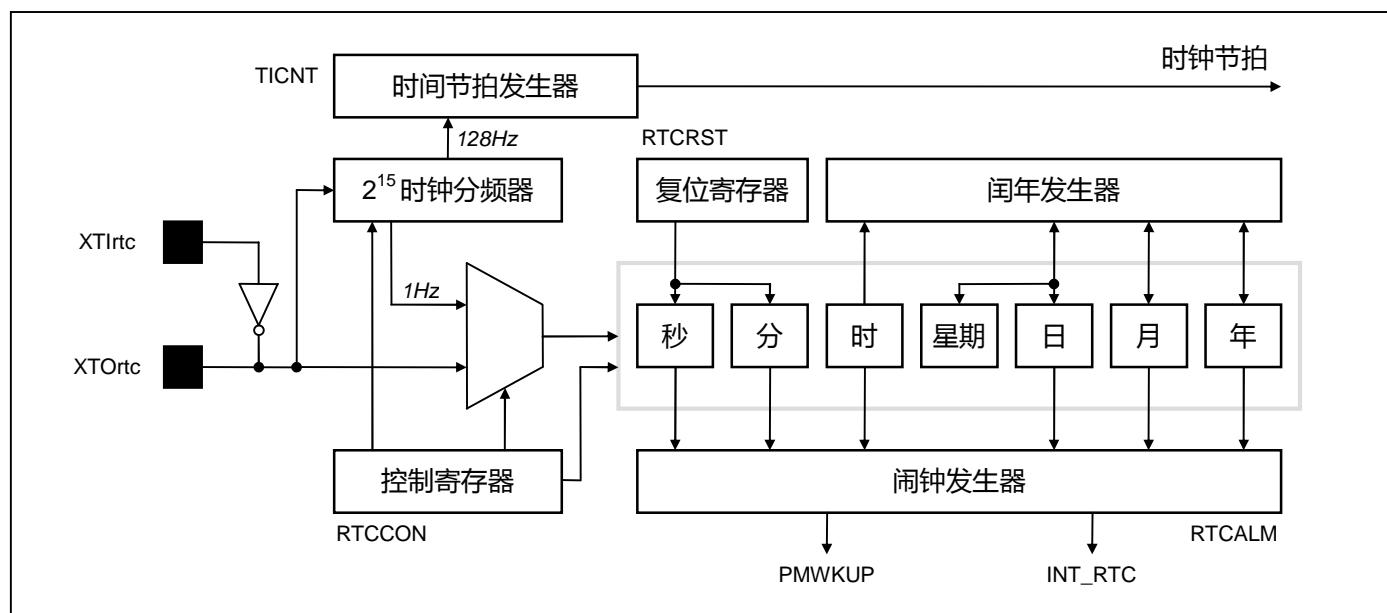


图 17-1. 实时时钟方框图

闰年发生器

闰年发生器能够基于 BCDDATE、BCDMON 和 BCDYEAR 的数据，从 28、29、30 或 31 中决定哪个是每月的最后日。此模块决定最后日时会考虑闰年因素。8 位计数器只能表示为 2 个 BCD 数字，因此其不能判决"00"年(最后两位数为 0 的年份)是否为闰年。例如，其不能判别 1900 和 2000 年。请注意 1900 年不是闰年，而 2000 年是闰年。因此，S3C2440A 中 00 的两位数是表示 2000 年，而表示 1900 年。

读/写寄存器

为了写 RTC 模块中的 BCD 寄存器，RTCCON 寄存器的位[0]必须设置为高。为了显示，年、月、日、时、分和秒，CPU 应该分别读取 RTC 模块中的 BCDSEC、BCDMIN、BCDHOUR、BCDDAY、BCDDATE、BCDMON 和 BCDYEAR 寄存器中的数据。然而可能存在 1 秒的偏差，因为读取了多个寄存器。例如，当用户从 BCDYEAR 到 BCDMIN 读取寄存器，其结果假定为 2059(年)、12(月)、31(日)、23(时)和 59(分)。当用户读取 BCDSEC 寄存器并且值的范围是从 1 到 59(秒)，这没有问题，但是如果该值为 0 秒，则年、月、日、时和分可能要变为 2060(年)、1(月)、1(日)、0(时)和 0(分)，因为存在着 1 秒的偏差。在这种情况下，如果 BCDSEC 为 0 则用户应该重新读取 BCDYEAR 到 BCDSEC。

备用电池操作

RTC 逻辑可以由备用电池驱动，即使如果系统电源关闭了则由 RTCVDD 引脚供电给 RTC 模块。当关闭了电源则应该阻塞掉 CPU 和 RTC 逻辑的接口，并且备用电池只驱动振荡电路和 BCD 计数器来最小化功耗。

闹钟功能

RTC 在掉电模式中或正常工作模式中的指定时间产生一个闹钟信号。在正常工作模式中，只激活闹钟中断(INT_RTC)信号。在掉电模式中，除了 INT_RTC 之外还激活电源管理唤醒(PMWKUP)信号。RTC 闹钟寄存器(RTCALM)决定了闹钟使能/禁止状态和闹钟时间设置的条件。

节拍时间中断

RTC 节拍时间是用于中断请求。TICNT 寄存器有一个中断使能位和中断的计数值。当节拍时间中断发生时计数值达到'0'。然后中断周期如下：

- 周期 = (n+1) / 128 秒
- n：节拍时间计数值(1 至 127)

此 RTC 时间节拍可能被用于实时操作系统(RTOS)内核时间节拍。如果时间节拍是由 RTC 时间节拍所产生的，RTOS 与时间的功能将通常同步到实际时间。

32.768KHz 晶振连接实例

图 17-2 显示了 RTC 单元振荡在 32.768KHz 的电路。

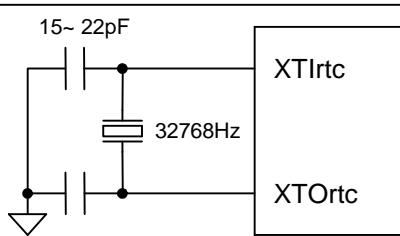


图 17-2. 主振荡电路例子

实时时钟特殊功能寄存器

实时时钟控制 (RTCCON) 寄存器

RTCCON 寄存器由 4 位组成，如控制 BCD 寄存器读/写使能的 RTCEN、CLKSEL、CNTSEL 和测试用的 CLKRST。

RTCEN 位可以控制所有 CPU 与 RTC 之间的接口，因此在系统复位后在 RTC 控制程序中必须设置为 1 来使能数据的读/写。同样的在掉电前，RTCEN 位应该清除为 0 来预防误写入 RTC 寄存器中。

寄存器	地址	R/W	描述	复位值
RTCCON	0x57000040(L) 0x57000043(B)	R/W (字节)	RTC 控制寄存器	0x0

RTCCON	位	描述	初始状态
CLKRST	[3]	RTC 时钟计数复位 0 = 不复位 1 = 复位	0
CNTSEL	[2]	BCD 计数选择 0 = 融入 BCD 计数器 1 = 保留 (分离 BCD 计数器)	0
CLKSEL	[1]	BCD 时钟选择 0 = XTAL 1/215 分频时钟 1 = 保留 (XTAL 时钟只用于测试)	0
RTCEN	[0]	RTC 控制使能 0 = 禁止 1 = 使能 注意：只能执行 BCD 时间计数和读操。	0

注意：

1. 所有 RTC 寄存器必须按字节为单位使用 STRB 或 LDRB 指令或 char 型指针访问。
2. (L): 小端
(B): 大端

节拍时间计数 (TICNT) 寄存器

寄存器	地址	R/W	描述	复位值
TICNT	0x57000044(L) 0x57000047(B)	R/W (字节)	节拍时间寄存器	0x0

TICNT	位	描述	初始状态
TICK INT 使能	[7]	节拍时间中断使能。 0 = 禁止 1 = 使能	0
TICK 时间计数	[6:0]	节拍时间计数值 (1 至 127)。 此计数器值内部递减并且用户不能在工作中读取此计数器的值。	000000

RTC 闹钟控制 (RTCALM) 寄存器

RTCALM 寄存器决定了闹钟使能和闹钟时间。请注意 RTCALM 寄存器在掉电模式中同时通过 INT_RTC 和 PMWKUP 产生闹钟信号，但是在正常工作模式中只产生 INT_RTC。

寄存器	地址	R/W	描述	复位值
RTCALM	0x57000050(L) 0x57000053(B)	R/W (字节)	RTC 闹钟控制寄存器	0x0

RTCALM	位	描述	初始状态
保留	[7]	-	0
ALMEN	[6]	全局闹钟使能 0 = 禁止 1 = 使能	0
YEAREN	[5]	年闹钟使能 0 = 禁止 1 = 使能	0
MONREN	[4]	月闹钟使能 0 = 禁止 1 = 使能	0
DATEEN	[3]	日闹钟使能 0 = 禁止 1 = 使能	0
HOUREN	[2]	时闹钟使能 0 = 禁止 1 = 使能	0
MINEN	[1]	分闹钟使能 0 = 禁止 1 = 使能	0
SECEN	[0]	秒闹钟使能 0 = 禁止 1 = 使能	0

闹钟秒数据 (ALMSEC) 寄存器

寄存器	地址	R/W	描述	复位值
ALMSEC	0x57000054(L) 0x57000057(B)	R/W (字节)	闹钟秒数据寄存器	0x0

ALMSEC	位	描述	初始状态
保留	[7]	-	0
SECDATA	[6:4]	闹钟秒 BCD 值 0 至 5	000
	[3:0]	0 至 9	0000

闹钟分数据 (ALMMIN) 寄存器

寄存器	地址	R/W	描述	复位值
ALMMIN	0x57000058(L) 0x5700005B(B)	R/W (字节)	闹钟分数据寄存器	0x0

ALMMIN	位	描述	初始状态
保留	[7]	-	0
MINDATA	[6:4]	闹钟分 BCD 值 0 至 5	000
	[3:0]	0 至 9	0000

闹钟时数据 (ALMHOUR) 寄存器

寄存器	地址	R/W	描述	复位值
ALMHOUR	0x5700005C(L) 0x5700005F(B)	R/W (字节)	闹钟时数据寄存器	0x0

ALMHOUR	位	描述	初始状态
保留	[7:6]	-	0
HOURDATA	[5:4]	闹钟时 BCD 值 0 至 2	00
	[3:0]	0 至 9	0000

闹钟日数据 (ALMDATE) 寄存器

寄存器	地址	R/W	描述	复位值
ALMDATE	0x57000060(L) 0x57000063(B)	R/W (字节)	闹钟日数据寄存器	0x01

ALMDATE	位	描述	初始状态
保留	[7:6]	-	00
DATEDATA	[5:4]	闹钟日 BCD 值 0 至 3	00
	[3:0]	0 至 9	0001

闹钟月数据 (ALMMON) 寄存器

寄存器	地址	R/W	描述	复位值
ALMMON	0x57000064(L) 0x57000067(B)	R/W (字节)	闹钟月数据寄存器	0x01

ALMMON	位	描述	初始状态
保留	[7:5]	-	00
MONDATA	[4]	闹钟月 BCD 值 0 至 1	0
	[3:0]	0 至 9	0001

闹钟年数据 (ALMYEAR) 寄存器

寄存器	地址	R/W	描述	复位值
ALMYEAR	0x57000068(L) 0x5700006B(B)	R/W (字节)	闹钟年数据寄存器	0x0

ALMYEAR	位	描述	初始状态
YEARDATA	[7:0]	闹钟年 BCD 值 00 至 99	0x0

BCD 秒 (BCDSEC) 寄存器

寄存器	地址	R/W	描述	复位值
BCDSEC	0x57000070(L) 0x57000073(B)	R/W (字节)	BCD 秒寄存器	未定义

BCDSEC	位	描述	初始状态
保留	[7]	-	-
SECDATA	[6:4]	秒 BCD 值 0 至 5	-
	[3:0]	0 至 9	-

BCD 分 (BCDMIN) 寄存器

寄存器	地址	R/W	描述	复位值
BCDMIN	0x57000074(L) 0x57000077(B)	R/W (字节)	BCD 分寄存器	未定义

BCDMIN	位	描述	初始状态
保留	[7]	-	-
MINDATA	[6:4]	分 BCD 值 0 至 5	-
	[3:0]	0 至 9	-

BCD 时 (BCDHOUR) 寄存器

寄存器	地址	R/W	描述	复位值
BCDHOUR	0x57000078(L) 0x5700007B(B)	R/W (字节)	BCD 时寄存器	未定义

BCDSEC	位	描述	初始状态
保留	[7:6]	-	-
HOURDATA	[5:4]	时 BCD 值 0 至 2	-
	[3:0]	0 至 9	-

BCD 日 (BCDDATE) 寄存器

寄存器	地址	R/W	描述	复位值
BCDDATE	0x5700007C(L) 0x5700007F(B)	R/W (字节)	BCD 日寄存器	未定义

BCDDATE	位	描述	初始状态
保留	[7:6]	-	-
DATEDATA	[5:4]	日 BCD 值 0 至 3	-
	[3:0]	0 至 9	-

BCD 星期 (BCDDAY) 寄存器

寄存器	地址	R/W	描述	复位值
BCDMON	0x57000080(L) 0x57000083(B)	R/W (字节)	BCD 星期寄存器	未定义

BCDDAY	位	描述	初始状态
保留	[7:3]	-	-
DAYDATA	[2:0]	星期 BCD 值 1 至 7	-

BCD 月 (BCDMON) 寄存器

寄存器	地址	R/W	描述	复位值
BCDMON	0x57000084(L) 0x57000087(B)	R/W (字节)	BCD 月寄存器	未定义

BCDMON	位	描述	初始状态
保留	[7:5]	-	-
MONDATA	[4]	月 BCD 值 0 至 1	-
	[3:0]	0 至 9	-

BCD 年 (BCDYEAR) 寄存器

寄存器	地址	R/W	描述	复位值
BCDYEAR	0x57000088(L) 0x5700008B(B)	R/W (字节)	BCD 年寄存器	未定义

BCDYEAR	位	描述	初始状态
YEARDATA	[7:0]	年 BCD 值 00 至 99	-

18 看门狗定时器

概述

S3C2440A 的看门狗电石气是用于当其由于噪声和系统错误引起的故障干扰时恢复控制器的工作。它可以被用作普通 16 位内部定时器来请求中断服务。看门狗定时器产生 128 个 PCLK 周期的复位信号。

特性

- 带中断请求的普通内部定时器模式
- 当定时器计数值达到 0 时 (超时) 激活 128 个 PCLK 周期的内部复位信号。

看门狗定时器操作

图 18-1 显示了看门狗定时器的功能方框图。看门狗定时器只使用 PCLK 作为其时钟源。预分频 PCLK 频率来产生相应看门狗定时器时钟，再将其结果频率分频。

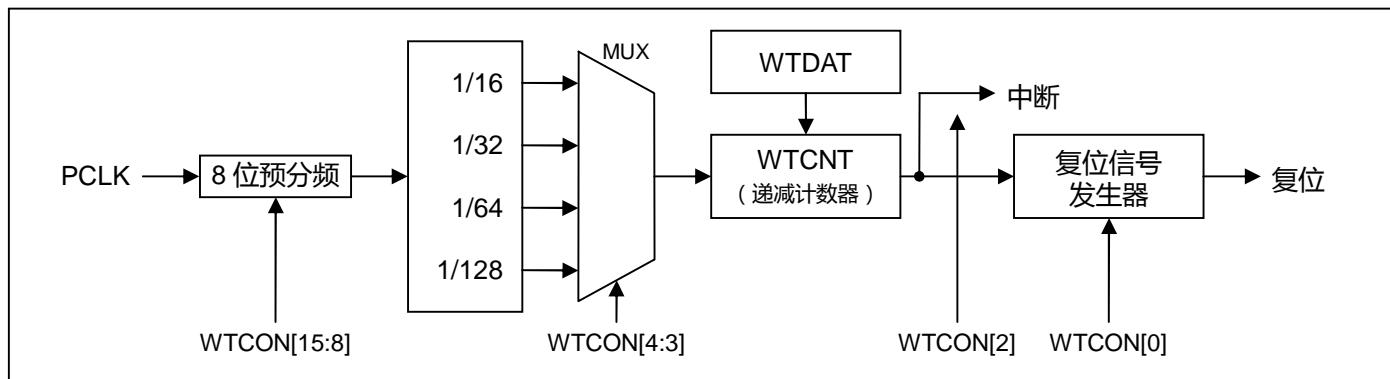


图 18-1. 看门狗定时器方框图

预分频值和分频系数是由看门狗定时器控制寄存器 (WTCON) 所指定的。预分频值的有效范围从 0 到 $2^8 - 1$ 。分频系数可以选择为 16、32、64 或 128。

使用以下等式来计算看门狗定时器的频率和每个定时器时钟周期的持续时间：

$$t_{\text{看门狗}} = 1 / [PCLK / (\text{预分频值} + 1) / \text{分频系数}]$$

WTDAT 和 WTCNT

一旦使能了看门狗定时器，看门狗定时器数据寄存器 (WTDAT) 的值不能被自动重载到定时计数器 (WTCNT) 中。由于这个理由，必须在看门狗定时器启动前写入一个初始值到看门狗定时器计数寄存器 (WTCNT) 中。

调试环境的考虑

当 S3C2440A 处于使用嵌入式 ICE 的调试模式，看门狗定时器必须无操作。看门狗定时器可以决定是否其处于当前来自 CPU 内核信号 (DBGACK 信号) 的调试模式中。一旦声明了 DBGACK 信号，看门狗定时器的复位输出将无效，就像看门狗定时器终止了。

看门狗定时器特殊寄存器

看门狗定时器控制 (WTCON) 寄存器

WTCON 寄存器允许用户使能或禁止看门狗定时器、从 4 个不同源选择时钟信号、使能或禁止中断和使能或禁止看门狗定时器输出。看门狗定时器是用于恢复 S3C2440A 上电后若有故障重时新启动；如果不希望控制器重新启动，则应该禁止看门狗定时器。

如果用户希望使用看门狗定时器作为普通定时器，则应使能中断并且禁止看门狗定时器。

寄存器	地址	R/W	描述	复位值
WTCON	0x53000000	R/W	看门狗定时器控制寄存器	0x8021

WTCON	位	描述	初始状态
预分频值	[15:8]	预分频值。该值范围从 0 到 255 ($2^8 - 1$)	0x80
保留	[7:6]	保留。正常工作中这两位必须为 00	00
看门狗定时器	[5]	看门狗定时器的使能或禁止位 0 = 禁止 1 = 使能	1
时钟选择	[4:3]	全局闹钟使能 00 : 16 01 : 32 10 : 64 11 : 128	00
中断产生	[2]	中断的使能或禁止位 0 = 禁止 1 = 使能	0
保留	[1]	保留。正常工作中此位必须为 0	0
复位使能/禁止	[0]	看门狗定时器复位输出的使能或禁止位 1 : 看门狗超时时发出 S3C2440A 复位信号 0 : 禁止看门狗定时器的复位功能	1

看门狗定时器数据 (WTDAT) 寄存器

WTDAT 寄存器是用于指定超时宽度。WTDAT 的内容不能在初始化看门狗定时器操作时被自动加载到定时计数器中。然而，使用 0x8000 (初始值) 将促使首次超时。这种情况中 WTDAT 的值将被自动重载到 WTDAT 中。

寄存器	地址	R/W	描述	复位值
WTDAT	0x53000004	R/W	看门狗定时器数据寄存器	0x8000

WTDAT	位	描述	初始状态
计数重载值	[15:0]	看门狗定时器重载的计数值。	0x8000

看门狗定时器计数 (WTCNT) 寄存器

WTCNT 寄存器包含正常工作期间看门狗定时器的当前值。请注意当看门狗定时器开始使能时 WTDAT 寄存器的内容不能自动加载到定时计数寄存器中，因此在使能前 WTCNT 寄存器必须设置初始值。

寄存器	地址	R/W	描述	复位值
WTCNT	0x53000008	R/W	看门狗定时器计数寄存器	0x8000

WTCNT	位	描述	初始状态
计数值	[15:0]	看门狗定时器的当前计数值。	0x8000

19 MMC/SD/SDIO 控制器

特性

- 兼容 SD 存储器卡规格 (1.0 版本) / MMC 规格 (2.11)
- 兼容 SDIO 卡规格 (1.0 版本)
- 16 字 (64 字节) 数据发送/接收 FIFO
- 40 位命令寄存器
- 136 位响应寄存器
- 8 位预分频逻辑 (频率 = 系统时钟 / (P + 1))
- 正常和 DMA 数据传输模式 (字节、半字或字传输)
- DMA burst4 存取支持 (只支持字传输)
- 1 位/4 位 (宽总线) 模式和块/流模式切换支持

方框图

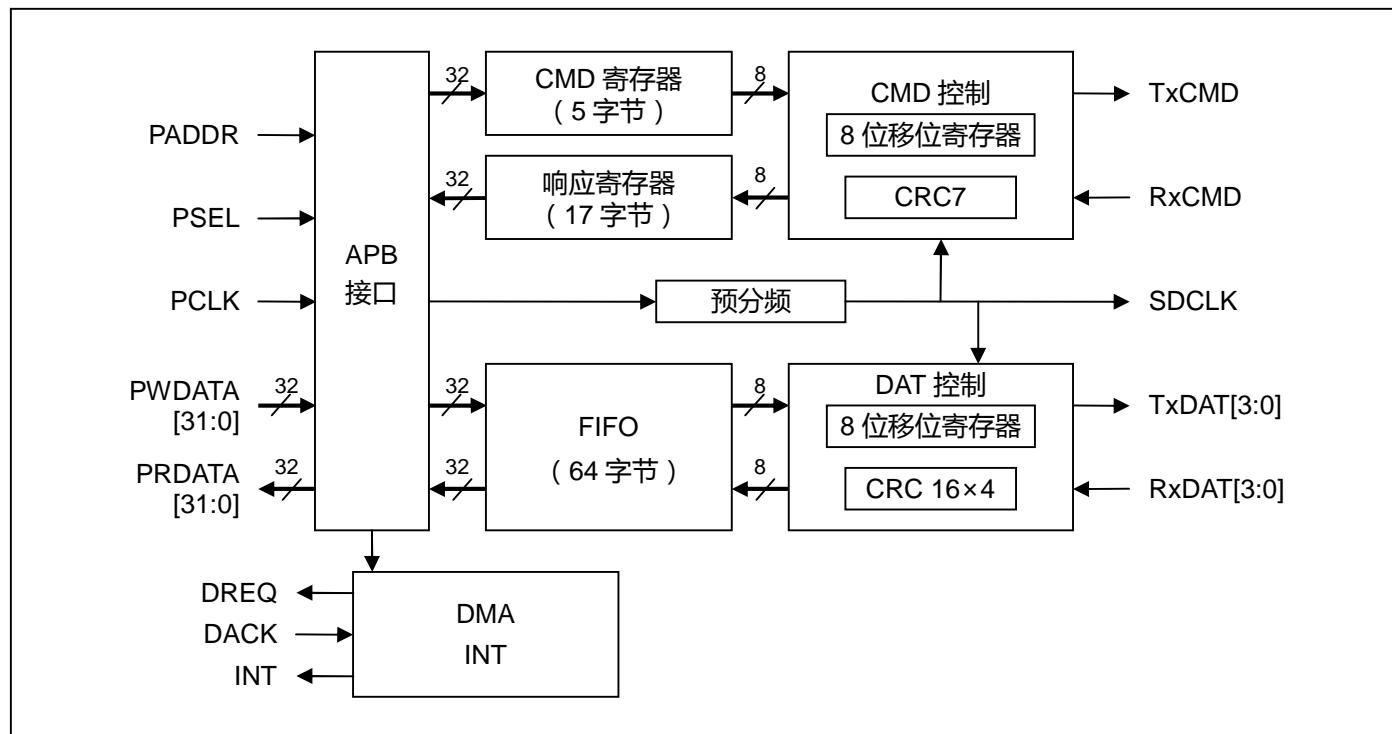


图 19-1. SD 接口方框图

SD 操作

串行时钟线同步 5 根数据线上信息的采样和移位。传输频率是通过设置 SDIPRE 寄存器相应位来控制。可以更改其频率来调整波特率数据寄存器值。

编程过程 (通用)

编程 SDI 模块需要以下几个基本步骤：

1. 设置 SDICON 配制适当的时钟和中断使能。
2. 设置 SDIPRE 配制为适当值。
3. 等待为初始化卡的 74 个 SDCLK 时钟周期。

CMD 通路编程

1. 写 32 位命令参数给 SDICmdArg。
2. 决定命令类型并设置 SDICmdCon 来启动命令发送。
3. 当 SDICmdSta 的特定标志置位时确认 SDICMD 通路操作的结束。
4. 如果命令类型为无响应时标志为 CmdSent。
5. 如果命令类型带响应时标志为 RspFin。
6. 通过写'1'到相应位来清除 SDICmdSta 的标志。

DAT 通路编程

1. 写数据超时时间到 SDITimer。
2. 写块大小 (块长度) 到 SDIBSize (通常为 0x80 个字)
3. 确定块方式，宽总线或 DMA 等并通过设置 SDIDatCon 启动数据传输。
4. Tx 数据→写数据到数据寄存器 (SDIDAT)，其中 Tx FIFO 为可用 (TFDET 为置位)，或一半 (TFFull 为置位)，或空 (TFEEmpty 为置位)。
5. Rx 数据→从数据寄存器 (SDIDAT) 读数据，其中 Rx FIFO 为可用 (RFDET 为置位)，或满 (RFFull 为置位)，或一半 (RFHalf 为置位)，再或最后数据就绪 (RFLast 为置位)。
6. 当 SDIDatSta 的 DatFin 标志置位时确认 SDI DAT 通路操作的结束。
7. 通过写'1'到相应位来清除 SDIDatSta 的标志。

SDIO 操作

有两个 SDIO 操作的功能：SDIO 中断接收和读等待请求发生。当 SDICON 寄存器的 RcvIOInt 位和 RwaitEn 位分别有效时这两个功能可操作。并且两个功能都有如下步骤和条件。

SDIO 中断

SD 1 位模式中，接受所有范围来自 RxDAT[1]引脚的中断。

SD 4 位模式中，共享数据接收与中断接收的 RxDAT[1]引脚。当中断检测范围（中断时间）为：

1. 单块：A 与 B 之间的时间
 - A：数据包的完成后 2 个时钟
 - B：下个 withdata 命令的发出结束位的完成
2. 多块、PrdType = 0：A 与 B 之间的时间，C 处重新启动
 - A：数据包的完成后 2 个时钟
 - B：A 后 2 个时钟
 - C：中止命令响应的结束位后 2 个时钟
3. 多块、PrdType = 1：A 与 B 之间的时间，A 处重新启动
 - A：数据包的完成后 2 个时钟
 - B：A 后 2 个时钟
 - 最后块的情况下，中断时间在 A 时开始，但不结束于 B（由 CMD53 引起）

读等待请求

无论是 1 位模式还是 4 位模式，在以下状况发送读等待请求信号到 TxDAT[2]引脚。

- 读多块操作中，请求信号在数据块结束后 2 个时钟开始发送。
- 当用户设置 SDIDatSta 寄存器的 RwaitReq 为 1 时发送结束。

SDI 特殊寄存器

SDI 控制寄存器 (SDICON)

寄存器	地址	R/W	描述	复位值
SDICON	0x5A000000	R/W	SDI 控制寄存器	0x0

SDICON	位	描述	初始状态
保留	[31:9]	-	-
SDreset	[8]	SDMMC 复位。复位整个 SD/MMC 模块。此位自动清零 0 = 正常模式 1 = SDMMC 复位	0
保留	[7:6]	-	0
CTYP	[5]	时钟类型。决定使用哪种时钟类型作为 SDCLK. 0 = SD 类型 1 = MMC 类型	0
ByteOrder	[4]	字节顺序类型。决定当读(写)按字对齐数据来自(到)SD 主机 FIFO 字节顺序类型 0 = 类型 A 0 = 类型 B	0
RcvIOInt	[3]	接收来自卡的 SDIO 中断。决定是否 SD 主机接收来自卡的 SDIO 中断 (SDIO) 0 = 忽略 1 = 接收 SDIO 中断	0
RWaitEn	[2]	读等待使能。决定当 SD 主机在多块读取模式中等待下个块时读等待信号产生。此位需要延迟来自卡的要传输的下个块 (SDIO) 0 = 禁止 (未产生) 1 = 读等待使能 (使用 SDIO)	0
保留	[1]	-	0
ENCLK	[0]	时钟输出使能。决定是否 SDCLK 输出使能 0 = 禁止 (预分频关) 1 = 时钟使能	0

注释：字节顺序类型

类型 A: (按字访问) D[7:0] → D[15:8] → D[23:16] → D[31:24]
(按半字访问) D[7:0] → D[15:8]

类型 B: (按字访问) D[31:24] → D[23:16] → D[15:8] → D[7:0]
(按半字访问) D[15:8] → D[7:0]

SDI 波特率预分频寄存器 (SDIPRE)

寄存器	地址	R/W	描述	复位值
SDIPRE	0x5A000004	R/W	SDI 波特率预分频寄存器	0x01

SDIPRE	位	描述	初始状态
预分频值	[7:0]	决定 SDI 时钟 (SDCLK) 率如上述等式 波特率 = PCLK / (预分频值 + 1)	0x01

注意：预分频值应该大于 0。

SDI 命令参数寄存器 (SDICmdArg)

寄存器	地址	R/W	描述	复位值
SDICmdArg	0x5A000008	R/W	SDI 命令参数寄存器	0x0

SDICmdArg	位	描述	初始状态
CmdArg	[31:0]	命令参数	0x00000000

SDI 命令控制寄存器 (SDICmdCon)

寄存器	地址	R/W	描述	复位值
SDICmdCon	0x5A00000C	R/W	SDI 命令控制寄存器	0x0

SDICommand	位	描述	初始状态
保留	[31:13]	-	-
AbortCmd	[12]	中止命令。决定命令类型是否为给中止 (SDIO) 0 = 正常命令 1 = 中止命令 (CMD12 , CMD52)	0
WithData	[11]	数据的命令。决定命令类型是否带数据 (SDIO) 0 = 无数据 1 = 带数据	0
LongRsp	[10]	决定是否主机接收 136 位长响应 0 = 短响应 1 = 长响应	0
WaitRsp	[9]	决定是否主机等待响应 0 = 无响应 1 = 等待响应	0
CMST	[8]	命令开始。决定命令操作是否开始。此位自动清零 0 = 命令就绪 1 = 命令开始	0
CmdIndex	[7:0]	带开始 2 位的命令索引 (8 位)	0x0

SDI 命令状态寄存器 (SDICmdSta)

寄存器	地址	R/W	描述	复位值
SDICmdSta	0x5A000010	R/(C)	SDI 命令状态寄存器	0x0

SDICmdSta	位	描述	初始状态
保留	[31:13]	-	-
RspCrc	[12] R/C	响应 CRC 失败。当命令收到响应时 CRC 校验失败。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = CRC 失败	0
CmdSent	[11] R/C	命令发送。送出命令 (不关心响应)。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 命令结束	0
CmdTout	[10] R/C	命令超时。命令响应超时 (64 个 CLK)。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 超时	0
RspFin	[9] R/C	响应接收结束。命令收到响应。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 响应结束	0
CmdOn	[8]	CMD 线进行中。命令传输处理中 0 = 未发现 1 = 处理中	0
RspIndex	[7:0]	带开始 2 位的响应索引 6 位 (8 位)	0x00

SDI 响应寄存器 0 (SDIRSP0)

寄存器	地址	R/W	描述	复位值
SDIRSP0	0x5A000014	R	SDI 响应寄存器 0	0x0

SDIRSP0	位	描述	初始状态
Response0	[31:0]	卡状态[31:0] (短), 卡状态[127:96] (长)	0x00000000

SDI 响应寄存器 1 (SDIRSP1)

寄存器	地址	R/W	描述	复位值
SDIRSP1	0x5A000018	R	SDI 响应寄存器 1	0x0

SDIRSP1	位	描述	初始状态
RCRC7	[31:24]	CRC7 (带结束位 , 短), 卡状态[95:88] (长)	0x00
Response1	[23:0]	未使用 (短), 卡状态[87:64] (长)	0x000000

SDI 响应寄存器 2 (SDIRSP2)

寄存器	地址	R/W	描述	复位值
SDIRSP2	0x5A00001C	R	SDI 响应寄存器 2	0x0

SDIRSP2	位	描述	初始状态
Response2	[31:0]	未使用 (短), 卡状态[63:32] (长)	0x00000000

SDI 响应寄存器 3 (SDIRSP3)

寄存器	地址	R/W	描述	复位值
SDIRSP3	0x5A000020	R	SDI 响应寄存器 3	0x0

SDIRSP3	位	描述	初始状态
Response3	[31:0]	未使用 (短), 卡状态[31:0] (长)	0x00000000

SDI 数据/忙定时器寄存器 (SDIDTimer)

寄存器	地址	R/W	描述	复位值
SDIDTimer	0x5A000024	R/W	SDI 数据忙定时器寄存器	0x0

SDIDTimer	位	描述	初始状态
保留	[31:23]	-	-
DataTimer	[22:0]	数据/忙超时时间	0x10000

SDI 块大小寄存器 (SDIBSize)

寄存器	地址	R/W	描述	复位值
SDIBSize	0x5A000028	R/W	SDI 块大小寄存器	0x0

SDIBSize	位	描述	初始状态
保留	[31:12]	-	-
BlkSize	[11:0]	块大小值 (0 至 4095 字节), 流模式中无需关心	0x000

注意：多块情况下 BlkSize 必须按字对齐（4 字节）大小。（ BlkSize[1:0] = 00 ）

SDI 数据控制寄存器 (SDIDatCon)

寄存器	地址	R/W	描述	复位值
SDIDatCon	0x5A00002C	R/W	SDI 数据控制寄存器	0x0

SDIDatCon	位	描述	初始状态
保留	[31:25]	-	-
Burst4	[24]	Burst4 使能。使能 DMA 模式中 Burst4 模式。应该只有当数据大小为字时置位此位 0 = 禁止 1 = Burst4 使能	0
DataSize	[23:22]	数据大小。指出带 FIFO 的传输大小，其类型为字、半字或字节 00 = 字节传输 01 = 半字传输 10 = 字传输 11 = 保留	0
PrdType	[21]	SDIO 中断周期类型。决定当传输了最后的数据块时 SDIO 中断时间是否为 2 个周期或扩展更多周期 (SDIO) 0 = 正好 2 个周期 1 = 更多周期 (如单块)	0
TARSP	[20]	响应后发送。决定当数据传输是否在收到响应后开始 0 = DatMode 设置后立刻开始 1 = 收到响应后 (假定 DatMode 设置为 2'b11)	0
RACMD	[19]	命令后响应。决定当数据接收是否在命令发送后开始 0 = DatMode 设置后立刻开始 1 = 命令发送后 (假定 DatMode 设置为 2'b10)	0
BACMD	[18]	命令后忙。决定当忙接收是否在命令发送后开始 0 = DatMode 设置后立刻开始 1 = 命令发送后 (假定 DatMode 设置为 2'b01)	0
BlkMode	[17]	块模式。数据传输模式 0 = 流数据传输 1 = 块数据传输	0
WideBus	[16]	宽总线使能。决定使能宽总线模式 0 = 标准总线模式 (只使用 SDIDAT[0]) 1 = 宽总线模式 (使用 SDIDAT[3:0])	0
EnDMA	[15]	DMA 使能。使能 DMA 0 = 禁止 (查询) 1 = DMA 使能 当 DMA 完成操作，则应该禁止此位。	0
DTST	[14]	数据传输开始。决定是否启动数据传输。此位自动清零。 0 = 数据就绪 1 = 数据启动传输	0
DatMode	[13:12]	数据传输模式。决定数据传输的操控 00 = 无操作 01 = 只检查忙模式 10 = 数据接收模式 11 = 数据发送模式	00
BlkNum	[11:0]	块数 (0 至 4095)，流模式中无需关心	0x000

注意：如果希望 TARSP、RACMD 或 BACMD 位 (SDIDatCon[20:18]) 之一为“1”，需要在 SDICmdCon 寄存器写 SDIDatCon 寄存器。(通常为 SDIO 需要)

SDI 数据持续 (Remain) 计数器寄存器 (ADIDatCnt)

寄存器	地址	R/W	描述	复位值
SDIDatCnt	0x5A000030	R	SDI 数据持续计数器寄存器	0x0

SDIDatCnt	位	描述	初始状态
保留	[31:24]	-	-
BlkNumCnt	[23:12]	持续块数	0x000
BlkCnt	[11:0]	1 块的持续数据字节	0x000

SDI 数据状态寄存器 (ADIDatSta)

寄存器	地址	R/W	描述	复位值
ADIDatSta	0x5A000034	R/(C)	SDI 数据状态寄存器	0x0

ADIDatSta	位	描述	初始状态
保留	[31:12]	-	-
NoBusy	[11] R/C	不忙。只检查忙模式中发出 CMD 包后 16 个周期期间不激活忙。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 无忙信号	0
RWaitReq	[10] R/C	读等待发生。发送读等待请求信号到 SD 卡。通过设置此位为 1 来清除此标志并且停止请求信号。 0 = 未发现 1 = 读等待请求发生	0
IOIntDet	[9] R/C	SDIO 中断发现。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = SDIO 中断发现	0
保留	[8]	-	0
CrcSta	[7] R/C	CRC 状态失败。当数据块发送时 CRC 状态错误 (CRC 校验失败)。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = CRC 状态失败	0
DatCrc	[6] R/C	数据接收 CRC 失败。数据块接收错误 (CRC 校验失败)。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 接收 CRC 失败	0
DatTout	[5] R/C	数据传输超时。数据/忙接收超时。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 超时	0
DatFin	[4] R/C	数据传输结束。数据传输完成 (数据计数器为 0)。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 命令结束	0
BusyFin	[3] R/C	忙结束。只检查忙结束。通过设置此位为 1 来清除此标志。 0 = 未发现 1 = 忙结束发现	0
保留	[2]	-	0
TxDatOn	[1]	Tx 数据处理中。数据发送处理中 0 = 未发现 1 = 数据 Tx 处理中	0
RxDatOn	[0]	Rx 数据处理中。数据接收处理中 0 = 未发现 1 = 数据 Rx 处理中	0

SDI FIFO 状态寄存器 (SDIFSTA)

寄存器	地址	R/W	描述	复位值
SDIFSTA	0x5A000038	R/(C)	SDI FIFO 状态寄存器	0x0

SDIFSTA	位	描述	初始状态
保留	[31:16]	-	-
FRST	[16] C	FIFO 复位。复位 FIFO 值。此位自动清零 00 = 未发现 01 = FIFO 满	0
FFfail	[15:14] R/C	FIFO 失败错误。当 FIFO 发生溢出/欠载 (<i>underrun</i>) 数据保存时 FIFO 失败错误。通过设置此位为 1 来清除此标志。 00 = 未发现 01 = FIFO 失败 10 = 最后传输中 FIFO 失败 (只需复位 FIFO) 11 = 保留	0
TFDET	[13]	Tx 发现可用 FIFO。此位指出当 DatMode 为数据发送模式时 FIFO 数据可以用于发送。如果 DMA 模式使能，则 SD 主机请求 DMA 操作。 0 = 未发现 (FIFO 满) 1 = 发现 ($0 \leq \text{FIFO} \leq 63$)	0
RFDET	[12]	Rx 发现可用 FIFO。此位指出当 DatMode 为数据接收模式时 FIFO 数据可以用于接收。如果 DMA 模式使能，则 SD 主机请求 DMA 操作。 0 = 未发现 (FIFO 空) 1 = 发现 ($1 \leq \text{FIFO} \leq 64$)	0
TFHalf	[11]	Tx FIFO 半满。每当 Tx FIFO 少于 33 字节时此位设置为 1。 0 = $33 \leq \text{Tx FIFO} \leq 64$ 1 = $0 \leq \text{Tx FIFO} \leq 32$	0
TFEmpty	[10]	Tx FIFO 空。每当 Tx FIFO 为空时此位设置为 1。 0 = $1 \leq \text{Tx FIFO} \leq 64$ 1 = 空 (0 字节)	0
RFLast	[9] R/C	Rx FIFO 最后数据就绪。每当 Rx FIFO 发生表现最后数据时此位设置为 1。通过设置此位为 1 来清除此标志。 0 = 还未接收到 1 = Rx FIFO 收到最后数据	0
RFFull	[8]	Rx FIFO 满。每当 Rx FIFO 满了此位设置为 1。 0 = $0 \leq \text{Rx FIFO} \leq 63$ 1 = 满 (64 字节)	0
RFHalf	[7]	Rx FIFO 半满。每当 Rx FIFO 多于 31 字节时此位设置为 1。 0 = $0 \leq \text{Rx FIFO} \leq 31$ 1 = $32 \leq \text{Rx FIFO} \leq 64$	0
FFCNT	[6:0]	FIFO 计数。FIFO 中的数据量 (字节)	0000000

注意 :尽管最后 Rx 数据大小为大于 FIFO 数据的持续计数，但可以读取这些数据。如果此事件发生，应该清除 FFfail 字段和 FIFO 复位字段。

SDI 中断屏蔽寄存器 (SDIIntMsk)

寄存器	地址	R/W	描述	复位值
SDIIntMsk	0x5A00003C	R/W	SDI 中断屏蔽寄存器	0x0

SDIIntMsk	位	描述	初始状态
保留	[31:19]	-	-
NoBusyInt	[18]	NoBusy 中断使能。决定如果忙信号无效则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
RspCrcInt	[17]	RspCrc 中断使能。决定如果响应 CRC 则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
CmdSentInt	[16]	CmdSent 中断使能。决定如果命令已发送 (无必须响应) 则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
CmdToutInt	[15]	CmdTout 中断使能。决定如果命令响应超时则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
RspEndInt	[14]	RspEnd 中断使能。决定如果收到响应则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
RWReqInt	[13]	RWReq 中断使能。决定如果读等待请求发生则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
IntDetInt	[12]	IntDet 中断使能。决定如果 SD 主机从卡收到 SDIO 中断 (SDIO) 则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
FFfailInt	[11]	FFfail 中断使能。决定如果 FIFO 失败错误发生则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
CrcStalInt	[10]	CrcStal 中断使能。决定如果 CRC 状态错误发生则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
DatCrcInt	[9]	DatCrc 中断使能。决定如果数据接收 CRC 失败则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
DatToutInt	[8]	DatTout 中断使能。决定如果数据接收超时发生则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
DatFinInt	[7]	DatFin 中断使能。决定如果数据计数器为 0 则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
BusyFinInt	[6]	BusyFin 中断使能。决定如果只在忙检查完成则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
保留	[5]	-	0
TFHalfInt	[4]	TFHalf 中断使能。决定如果 Tx FIFO 半添满则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
TFEmptInt	[3]	TFEmpt 中断使能。决定如果 Tx FIFO 为空则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
RFLastInt	[2]	RFLast 中断使能。决定如果 Rx FIFO 得到最后数据则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
RFFullInt	[1]	RFFull 中断使能。决定如果 Rx FIFO 添满则 SDI 产生中断 0 = 禁止 1 = 中断使能	0
RFHalfInt	[0]	RFHalf 中断使能。决定如果 Rx FIFO 半添满则 SDI 产生中断 0 = 禁止 1 = 中断使能	0

SDI 数据寄存器 (SDIDAT)

寄存器	地址	R/W	描述	复位值
SDIDAT	0x5A000040,44,48,4C (Li/W, Li/HW, Li/B, Bi/W) 0x5A000041(Bi/HW), 0x5A000043(Bi/B)	R/W	SDI 数据寄存器	0x0

SDIDAT	位	描述	初始状态
数据寄存器	[31:0]	此字段等于 SDI 通道上要发送或接收的数据	0x00000000

注释：

- (Li/W, Li/HW, Li/B): 当端模式为小端时以字/半字/字节为单位存取
- (Bi/W): 当端模式为大端时以字为单位存取
- (Bi/HW): 当端模式为大端时以半字为单位存取
- (Bi/B): 当端模式为大端时以字节为单位存取

20 IIC 总线接口

概述

S3C2440A RISC 微处理器可以支持一个多主控 IIC 总线串行接口。一条专用串行数据线 (SDA) 和一条专用串行时钟线 (SCL) 传递连接到 IIC 总线的总线主控和外设之间的信息。SDA 和 SCL 线都为双向的。

多主控 IIC 总线模式中，多个 S3C2440A RISC 微处理器可以发送或接收串行数据来自或到从设备。主机 S3C2440A 可以通过 IIC 总线启动和结束数据传输。S3C2440A 中的 IIC 总线是使用标准总线仲裁步骤。

为了控制多主控 IIC 总线操作，必须写入值到以下寄存器中：

- 多主控 IIC 总线控制寄存器，IICCON
- 多主控 IIC 总线控制/状态寄存器，IICSTAT
- 多主控 IIC 总线 Tx/Rx 数据移位寄存器，IICDS
- 多主控 IIC 总线地址寄存器，IICADD

当释放了 IIC 总线时，SDA 和 SCL 线应该都保持为高电平。一个高到低 SDA 的变化可以启动一个起始条件。SCL 稳定保持在高电平时的一个低到高 SDA 的变化可以启动一个停止条件。

起始和停止条件通常由主设备产生。第一个数据字节为 7 位地址值，其在启动起始条件后放到总线上，可以确定出主设备要选择的从设备。第 8 位是决定传输方向（读或写）。

每个放到 SDA 线上的字节都应该总共为 8 位。字节可以在总线传输操作期间无限制的发送或接收。数据通常从最高有效位 (MSB) 开始发送，并且每个字节应该立即通过应答 (ACK) 位跟上。

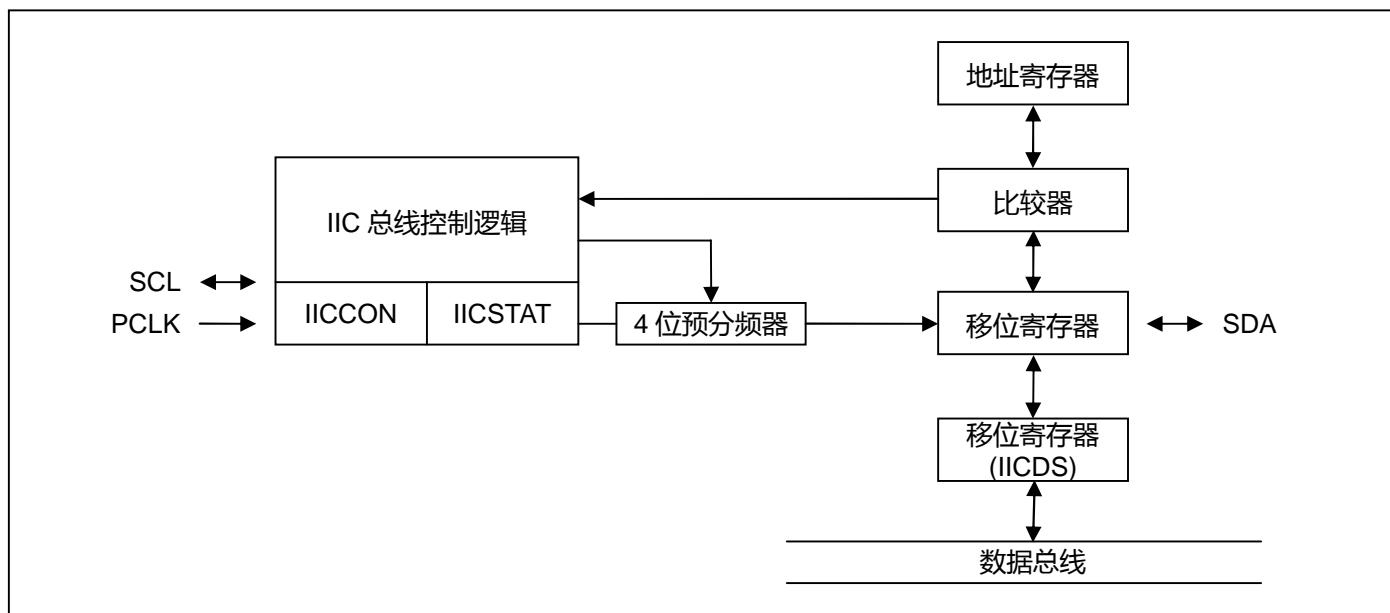


图 20-1. IIC 总线方框图

IIC 总线接口

S3C2440A 的 IIC 总线接口有 4 种工作模式：

- 主机发送模式
- 主机接收模式
- 从机发送模式
- 从机接收模式

这些工作模式彼此的功能关系描述如下。

起始和停止条件

当 IIC 总线接口不活动时，其通常在从机模式。换句话说，该接口在从 SDA 线上检测到起始条件（当 SCL 时钟信号为高时的一个高到低 SDA 的变化可以启动一个起始条件）之前应该处于从机模式。当接口状态被改为主机模式时，可以起始发送数据到 SDA 上并且产生 SCL 信号。

起始条件可以传输 1 字节串行数据到 SDA 线上，而停止条件可以结束数据的传输。停止条件是在当 SCL 为高时的 SDA 线低到高的变化。起始和停止条件总由主机产生。当产生了一个起始条件时 IIC 总线变为忙。停止条件将使得 IIC 总线空闲。

当主机发起一个起始条件时，其应该送出一个从机地址来通知从设备。地址字段的 1 字节由 7 位地址和 1 位传输方向标志（表现为读或写）组成。如果位[8]为 0，其表示一个写操作（发送操作）；如果位[8]为 1，其表示一个数据读取的请求（接收操作）。

主机将通过发送一个停止条件来完成传输操作。如果主机希望持续发生数据到总线上，其应该在同一个从地址产生再一个起始条件。这样就可以执行各种格式的读写操作。

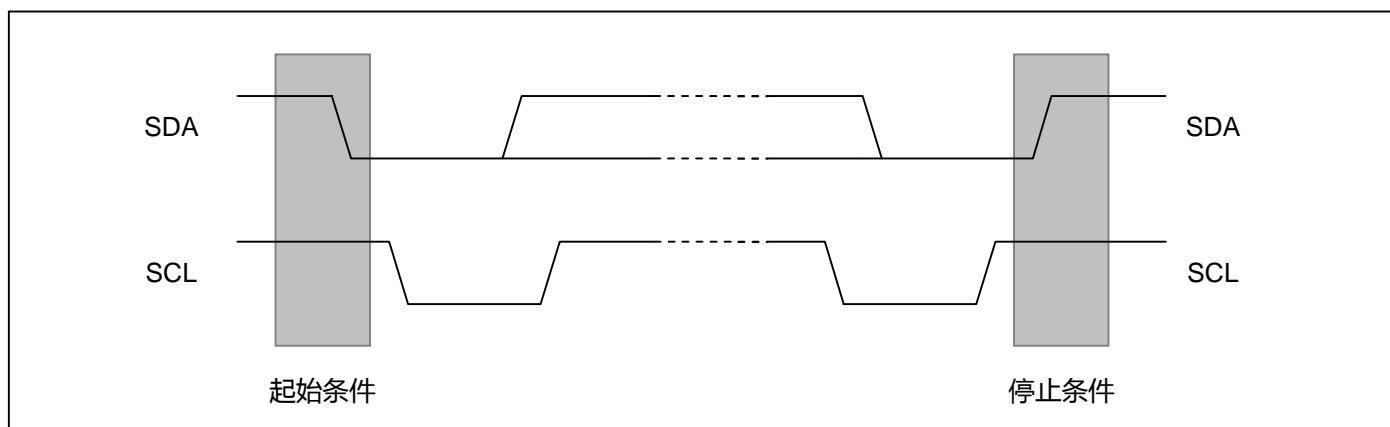


图 20-2. 起始和停止条件

数据传输格式

放置到 SDA 线上的每个字节应该以 8 位为长度。每次传输字节可以无限制的发送。起始条件随后的第一个字节应该包含地址字段。当 IIC 总线工作在主机模式时可以由主机发送该地址字段。每个字节都应该跟随一个应答 (ACK) 位。总是最先发送串行数据和地址的 MSB。

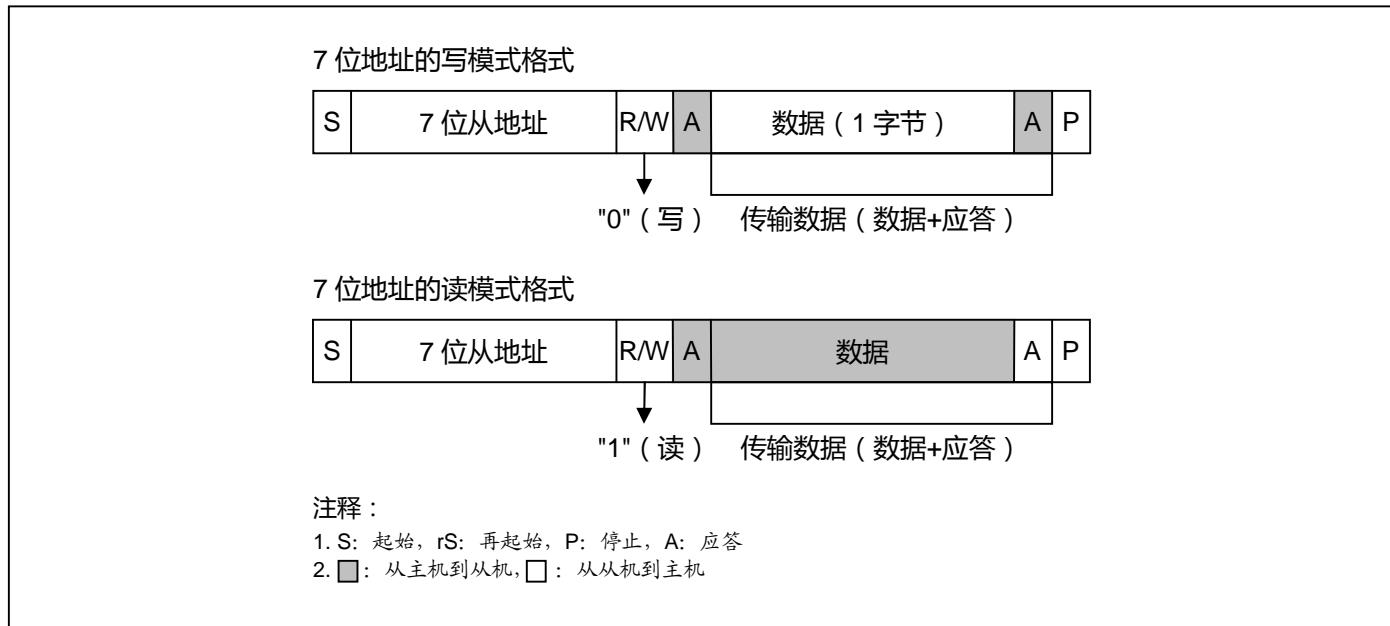


图 20-3. IIC 总线接口数据格式

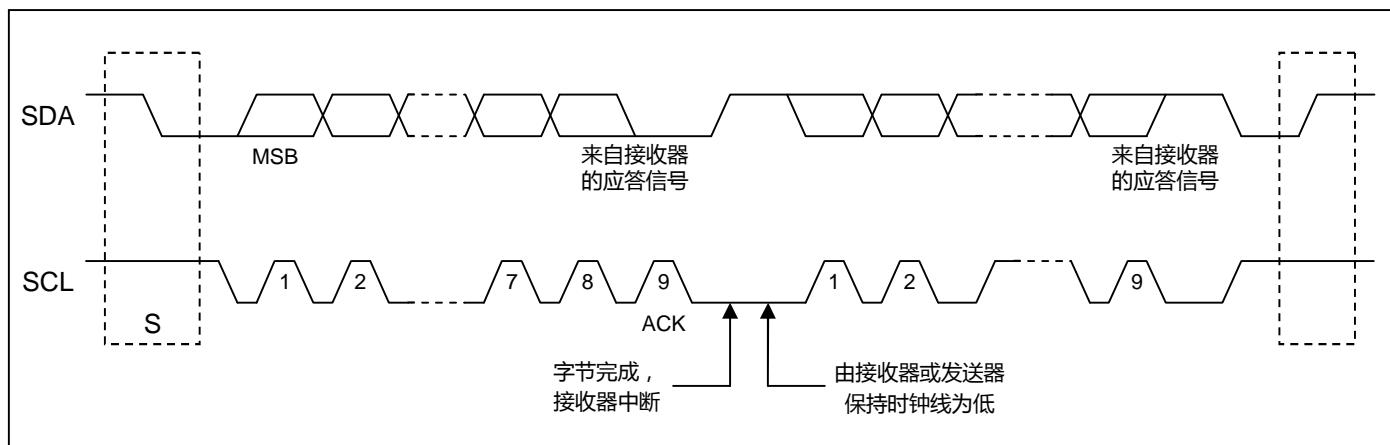


图 20-4. IIC 总线上的数据传输

发送 ACK 信号

为了完成一次单字节传输操作，接收器应该发送一个 ACK 位给发送器。ACK 脉冲应该发生在 SCL 线的第 9 个时钟。前 8 个时钟是提供给单字节传输的。主机需要产生时钟脉冲来发送 ACK 位。

当发送器收到 ACK 时钟脉冲时应该通过拉高 SDA 线来释放 SDA 线。当接收器在 ACK 时钟脉冲期间也应该驱动 SDA 线为低来在第 9 个脉冲的高电平时期内保持 SDA 为低。

ACK 位发送功能可以由软件 (IICSTAT) 使能或禁止。然而，需要 SCL 的第 9 个时钟上的 ACK 脉冲来完成单字节的传输操作。

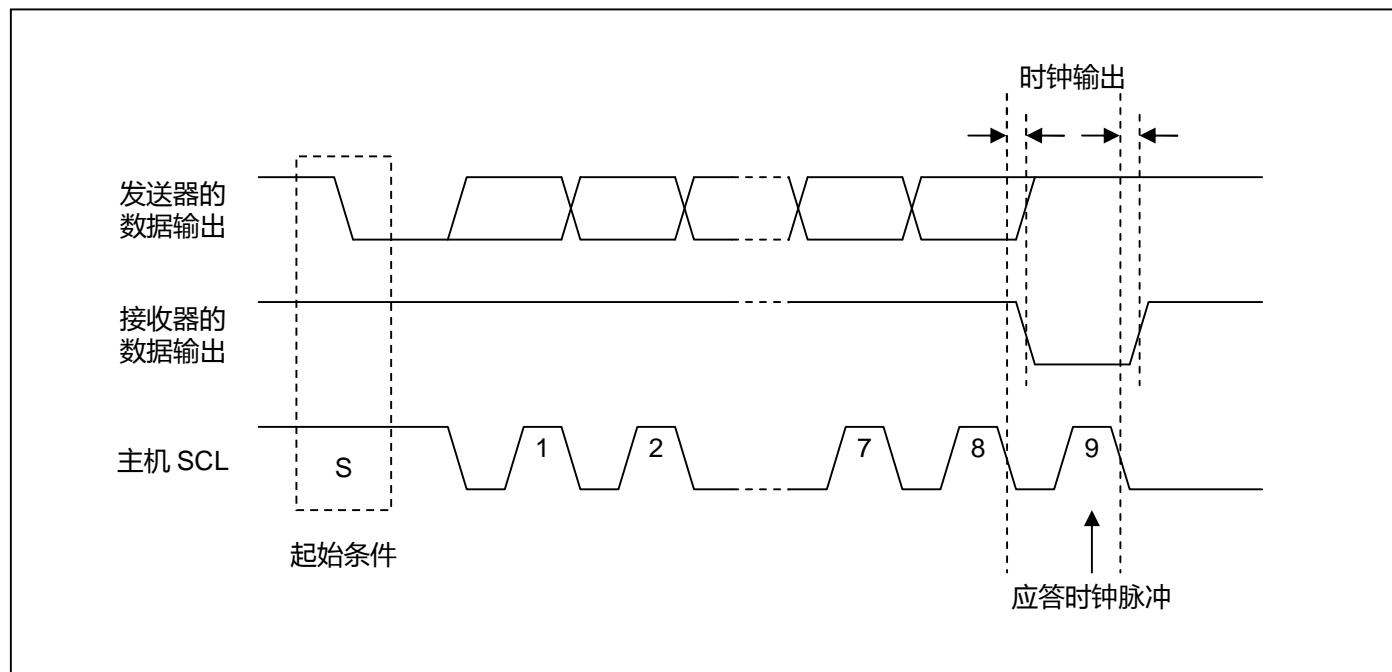


图 20-5. IIC 总线上的应答

读写操作

发送模式中当发送了数据时，在 IIC 总线数据移位(IICDS)寄存器收到新数据之前 IIC 总线接口将会一直等待。在新数据写入到寄存器之前，SCL 线将会保持为低，然后在其写入后释放。S3C2440A 应该等待中断来确定当前数据发送的完成。在 CPU 收到中断请求后，需要再次写一个新数据到 IICDS 寄存器中。

接收模式中当收到了数据时，在读取 IICDS 寄存器前 IIC 接口将会一直等待。在新数据读出前，SCL 线将会保持为低，然后在其读取后释放。S3C2440A 应该等待中断来确定当前数据接收的完成。在 CPU 收到中断请求后，需要从 IICDS 寄存器中读取数据。

总线仲裁步骤

发生在 SDA 线上的仲裁是预防总线上两个主机的竞争。如果 SDA 为高电平的主机检测到另一个主机的 SDA 激活了低电平，其将不会启动数据传输，这是因为总线上的当前电平与其（前者）拥有的电平不符合。将扩展仲裁步骤直到 SDA 线变为高。

然而，当主机同时拉低 SDA 线时，每个主机都应该判断是否分配了主控给自己。为了判断则每个主机应该检测地址位。当每个主机都产生的从地址时，它们也应该检测 SDA 线上的地址位，这是因为 SDA 线个更倾向于获得低电平而不是保持为高电平。假定一个主机产生了一个低电平作为第一个地址位，同时其它主机保持为高。在这种情况下，主机都将检测到总线上的低电平，因为低电平状态在电平上优先于高电平状态。当发生这种情况时，产生低电平（作为地址的第一位）的主机将得到主控，同时产生高电平（作为地址的第一位）的主机应该退出主控。如果主机都产生低电平作为地址的第一位，它们应该继续通过第二个地址位仲裁。这种仲裁将持续到最后地址位的结束。

中止条件

如果从接收器不能应答从地址的确认，其应该保持 SDA 线的电平为高。这种情况下，主机应该产生一个停止条件并且中止传输。

如果主机接收器受到了传输中止的影响，其应该通过取消来自从机收到的最后数据字节后 ACK 的产生来指示从发送操作的结束。从发送器应该随后释放 SDA 来允许主机产生停止条件。

配制 IIC 总线

可以编程 IICCON 寄存器中的 4 位预分频器值来控制串行时钟 (SCL) 的频率。IIC 总线接口地址被储存在 IIC 总线地址 (IICADD) 寄存器中。（默认 IIC 总线接口地址包含一个未知值。）

每种模式中操作流程图

必须在任何 IIC Tx/Rx 操作之前执行以下步骤。

1. 如果需要，写自己从地址到 IICADD 寄存器。
2. 设置 IICCON 寄存器
 - a) 使能中断
 - b) 定义 SCL 周期
3. 设置 IICSTAT 以使能串行输出

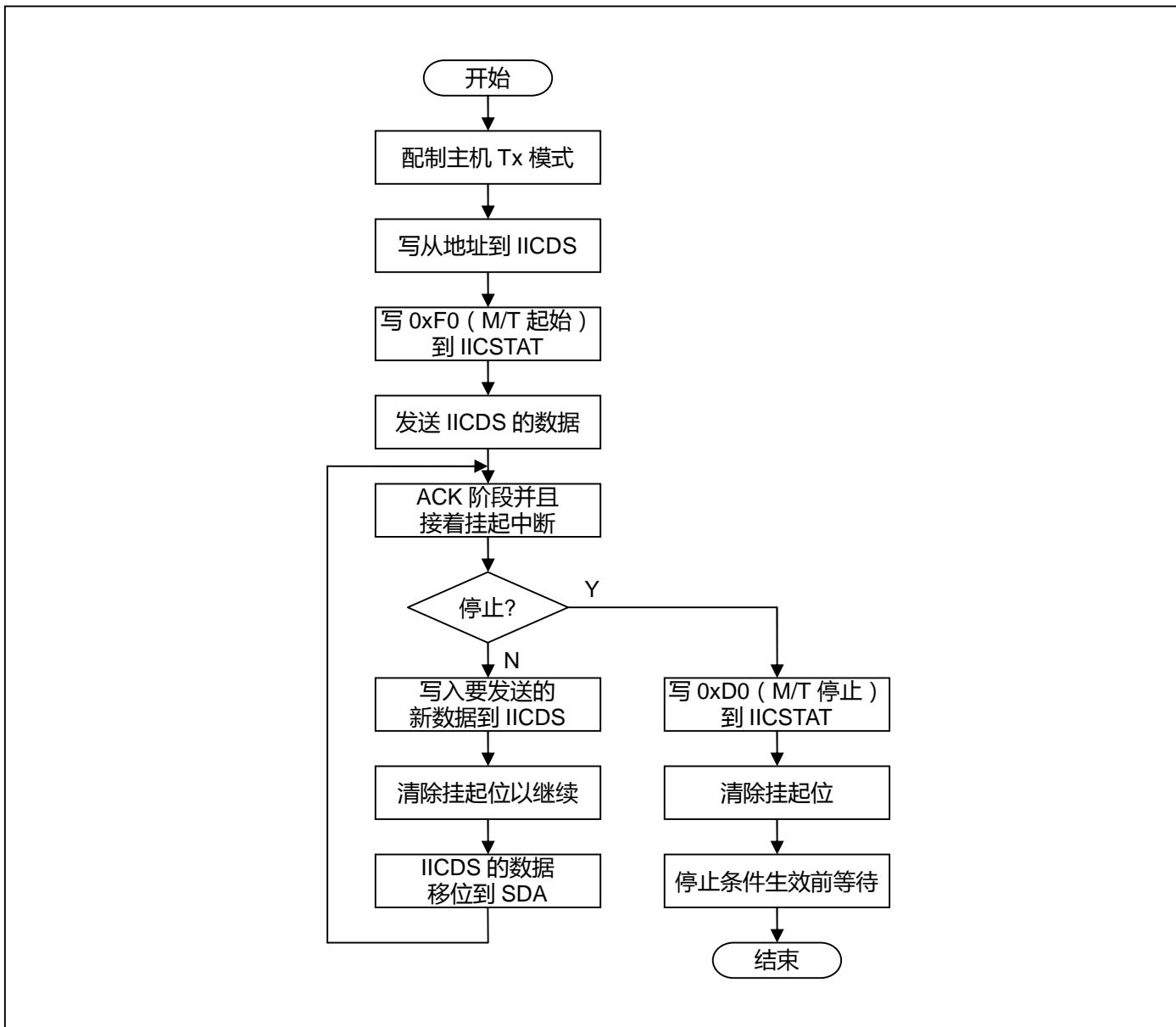


图 20-6. 主机发送器模式操作

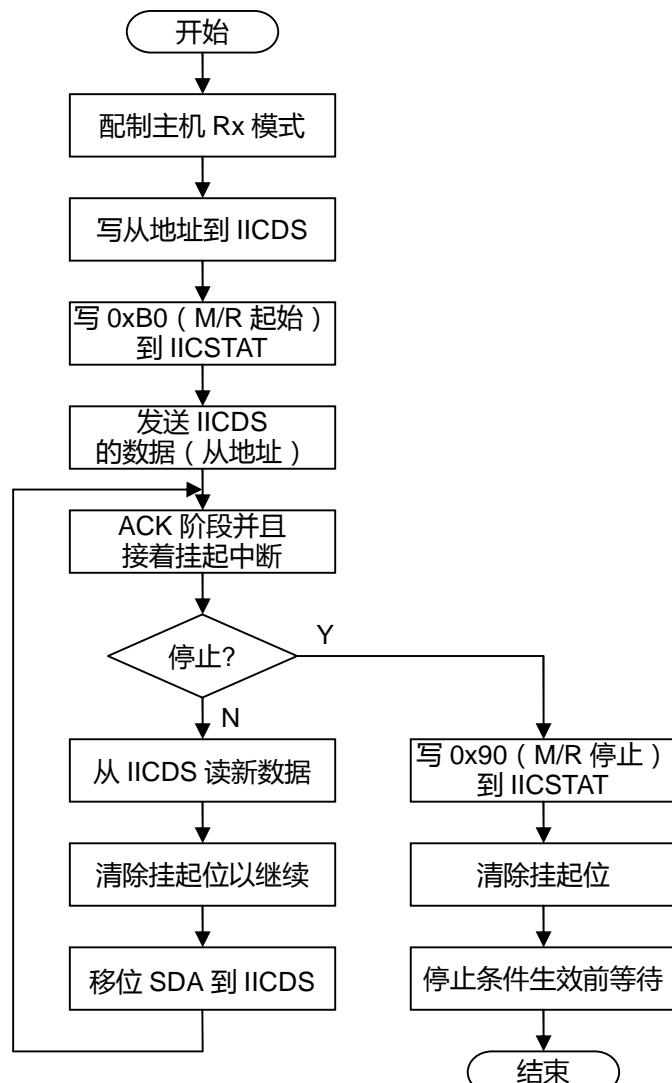


图 20-7. 主机接收器模式操作

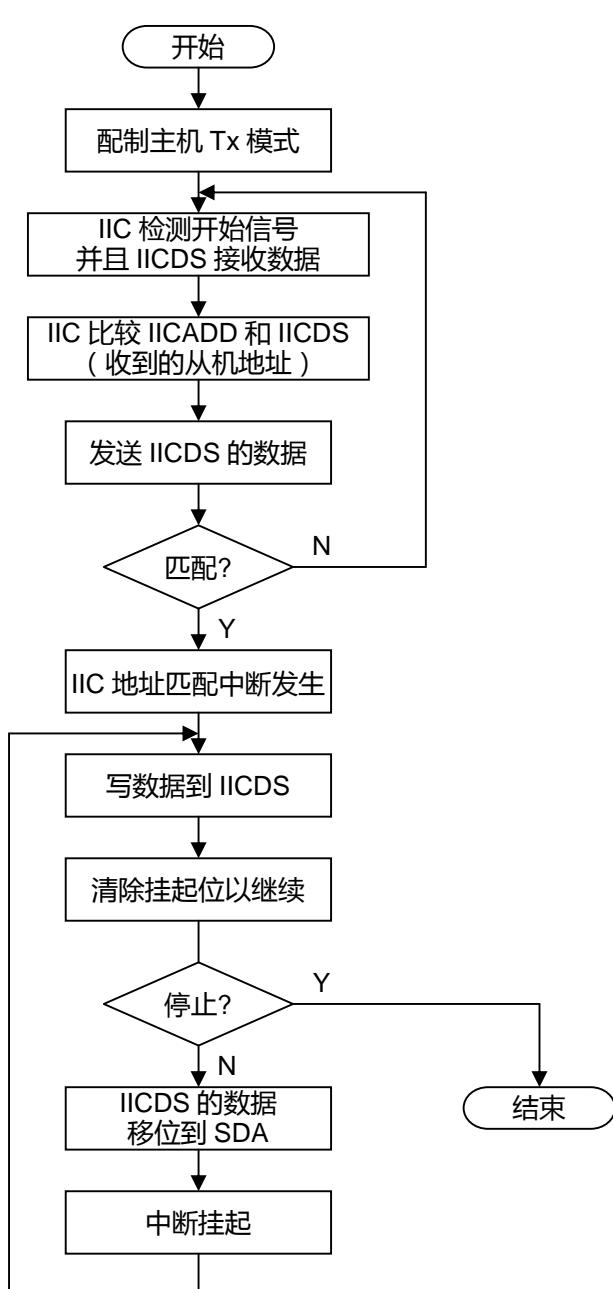


图 20-8. 从机发送器模式操作

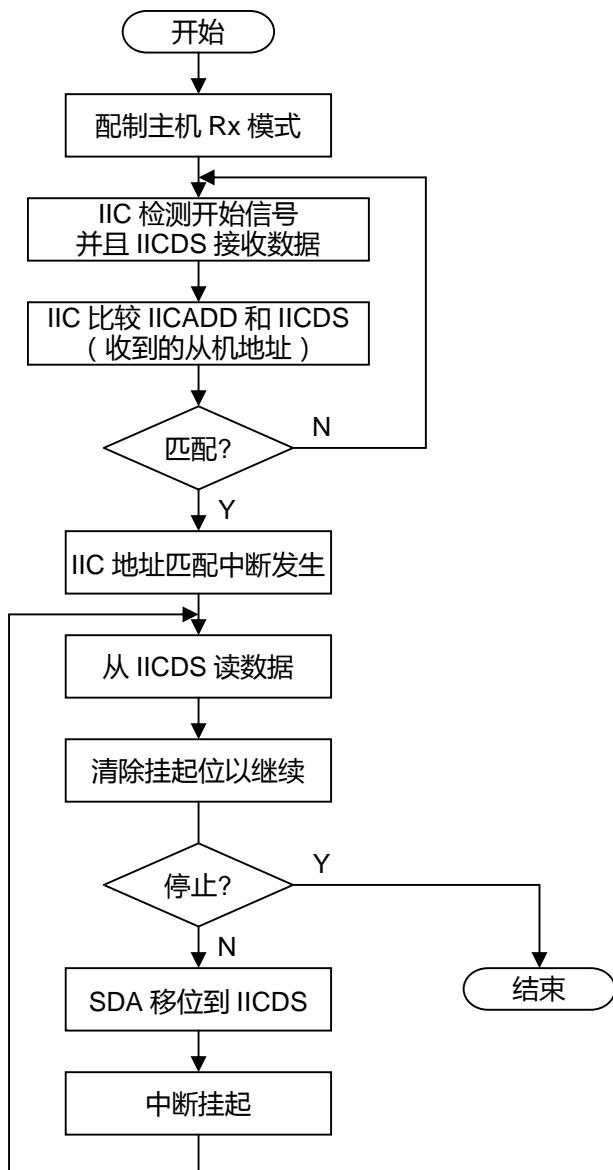


图 20-9. 从机接收器模式操作

IIC 总线接口特殊寄存器

多主机 IIC 总线控制 (IICCON) 寄存器

寄存器	地址	R/W	描述	复位值
IICCON	0x54000000	R/W	IIC 总线控制寄存器	0x0X

IICCON	位	描述	初始状态
应答发生 ⁽¹⁾	[7]	IIC 总线应答使能位 0 = 禁止 1 = 允许 Tx 模式中， IICSDA 在应答时间为高。 Rx 模式中， IICSDA 在应答时间为低。	0
Tx 时钟源选择	[6]	IIC 总线发送时钟预分频器的时钟源选择位 0 : IICCLK = fPCLK /16 1 : IICCLK = fPCLK /512	0
Tx/Rx 中断 ⁽⁵⁾	[5]	IIC 总线 Tx/Rx 中断使能/禁止位 0 = 禁止 1 = 允许	0
中断挂起标志 ⁽²⁾⁽³⁾	[4]	IIC 总线 Tx/Rx 中断挂起标志。不能写 1 到此位。当此位读取到 1 时， IICSDA 限制为低并且停止 IIC。清除此位为 0 以继续操作。 0 : 1) 无中断挂起 (读时) 2) 清除挂起条件并且继续操作 (写时) 1 : 1) 中断挂起 (读时) 2) N/A (写时)	0
发送时钟值 ⁽⁴⁾	[3:0]	IIC 总线发送时钟预分频器。 IIC 总线发送时钟预分频器是由此 4 位预分频值按以下公式决定的： $Tx \text{ 时钟} = IICCLK / (IICCON[3:0] + 1)$	未定义

注释：

1. EEPROM 接口， Rx 模式中为了产生停止条件在读取最后数据之前会禁止产生应答。
2. IIC 总线中断发生在： 1) 当完成了 1 字节发送或接收操作； 2) 当广播呼叫或从地址匹配发生时； 3) 如果总线仲裁失败。
3. 为了在 SCL 上升沿之前调整 SDA 的建立时间，必须在清除 IIC 中断挂起位前写 IICDS。
4. IICCLK 由 IICCON[6] 所决定的。
Tx 时钟可以由 SCL 变化时间改变。
当 IICCON[6] = 0, IICCON[3:0] = 0x0 或 0x1 为不可用。
5. 如果 IICCON[5]=0, IICCON[4] 不正确的工作。
因此，推荐即使不使用 IIC 中断也设置 IICCON[5]=1。

多主机 IIC 总线控制/状态 (IICSTAT) 寄存器

寄存器	地址	R/W	描述	复位值
IICSTAT	0x54000004	R/W	IIC 总线控制/状态寄存器	0x0

IICSTAT	位	描述	初始状态
模式选择	[7:6]	IIC 总线主机/从机 Tx/Rx 模式选择位 00 : 从接收模式 01 : 从发送模式 10 : 主接收模式 11 : 主发送模式	00
忙信号状态/ 起始停止条件	[5]	IIC 总线忙信号状态位 0 : (读) 不忙 (读时) (写) 停止信号产生 1 : (读) 忙 (读时) (写) 起始信号产生 只在起始信号后将自动传输 IICDS 中的数据	0
串行输出	[4]	IIC 总线数据输出使能/禁止位 0 : 禁止 Rx/Tx 1 : 使能 Rx/Tx	0
仲裁状态标志	[3]	IIC 总线仲裁过程状态标志位 0 : 总线仲裁成功 1 : 串行 I/O 间总线仲裁失败	0
从地址状态标志	[2]	IIC 总线从地址状态标志位 0 : 发现起始/停止条件清除 1 : 收到从地址与 IICADD 中地址值匹配	0
地址零状态标志	[1]	IIC 总线地址零状态标志位 0 : 发现起始/停止条件清除 1 : 收到从地址为 00000000b	0
最后收到位 状态标志	[0]	IIC 总线最后收到位状态标志位 0 : 最后收到位为 0 (已收到 ACK) 1 : 最后收到位为 1 (未收到 ACK)	0

多主机 IIC 总线地址 (IICADD) 寄存器

寄存器	地址	R/W	描述	复位值
IICADD	0x54000008	R/W	IIC 总线地址寄存器	0x0XX

IICADD	位	描述	初始状态
从地址	[7:0]	从 IIC 总线锁存的 7 位从地址。 当 IICSTAT 中串行输出使能=0 时，IICADD 为写使能。可以在任意时间读取 IICADD 的值，不用去考虑当前输出使能位 (IICSTAT) 的设置。 从地址 : [7:1] 未映射 : [0]	XXXXXXXX

多主机 IIC 总线发送/接收数据移位 (IICDS) 寄存器

寄存器	地址	R/W	描述	复位值
IICDS	0x5400000C	R/W	IIC 总线发送/接收数据移位寄存器	0x0XX

IICDS	位	描述	初始状态
数据移位	[7:0]	IIC 总线 Tx/Rx 操作的 8 位数据移位寄存器。 当 IICSTAT 中串行输出使能=1，IICDS 为写使能。可以在任意时间读取 IICDS 的值，不用去考虑当前输出使能位 (IICSTAT) 的设置。	XXXXXXXX

多主机 IIC 总线线控制 (IICLC) 寄存器

寄存器	地址	R/W	描述	复位值
IICLC	0x54000010	R/W	IIC 总线多主机线控制寄存器	0x00

IICLC	位	描述	初始状态
滤波器使能	[2]	IIC 总线滤波使能位。 当 SDA 端口工作在输入时，应该设置此位为高。这个滤波器可以防止两个 PCLK 时间期间由于干扰而发生错误。 0 : 禁止滤波器 1 : 使能滤波器	0
SDA 输出延时	[1:0]	IIC 总线 SDA 线延时长度选择位。 SDA 线按以下时钟时间 (PCLK) 延时 00 : 0 个时钟 01 : 5 个时钟 10 : 10 个时钟 11 : 15 个时钟	00

21 IIS 总线接口

概述

目前，在市场上以 CD、数字音频录音带、数字音响处理器和数字 TV 音响等的形式的许多数字音频系统都吸引着消费者。S3C2440A 的内置 IC 音频 **IIS** 总线接口可以用于实现 CODEC 接口到外部 8/16 位立体声音频 CODEC IC 给迷你光碟和便携式应用。IIS 总线接口同时支持 IIS 总线数据格式和 MSB 对齐数据格式。该接口还提供 FIFO 存取的 DMA 传输模式来代替中断。其能够同时发送和接收数据以及在同一时间交替的发送或接收数据。

方框图

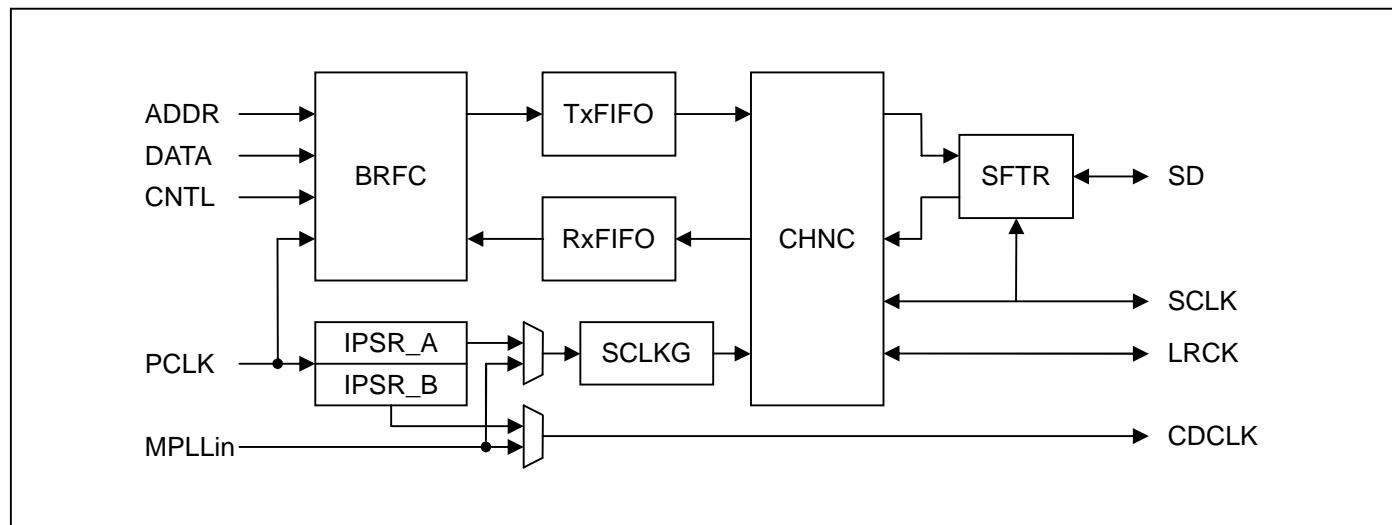


图 21-1. IIS 总线方框图

功能描述

总线接口、寄存器组和状态机 (BRFC)：总线接口逻辑和 FIFO 存取都是由状态机控制的。

5 位双预分频器 (IPSR)：一个预分频器是用于 IIS 总线接口的主时钟发生器，另一个是用于作为外部 CODEC 的时钟发生器。

64 字节 FIFO (TxFIFO 和 RxFIFO)：发送数据传输中，数据被写入到 TxFIFO；接收数据传输中，从 RxFIFO 读取数据。

主机 IISCLK 发生器 (SCLKG)：在主机模式中，从主机时钟产生串行位时钟。

通道发生器和状态机 (CHNC)：通道状态机控制和产生 IISCLK 和 IISLRCK。

16 位移位寄存器 (SFTR)：发送模式中并行数据被移位到串行数据输出，并且接收模式中串行数据输入被移位到并行数据。

只发送或只接收模式

正常传输

IIS 控制寄存器包含 FIFO 就绪标志位给发送和接收 FIFO。当 FIFO 准备好了发送数据，如果发送 FIFO 非空则 FIFO 就绪标志设置为'1'。如果发送 FIFO 为空，则 FIFO 就绪标志设置为'0'。接收中的 FIFO 未满，接收 FIFO 的 FIFO 就绪标志设置为'1'；它表明 FIFO 是否准备好了接收数据。如果接收 FIFO 为满，则 FIFO 就绪标志设置为'0'。这些标志可以在 CPU 准备写或读 FIFO 时决定。用此方法可以在 CPU 正在存取发送和接收 FIFO 时发送或接收串行数据。

DMA 传输

此模式中，通过 DMA 控制器发送或接收 FIFO 为可存取的。发送或接收模式中的 DMA 服务请求由 FIFO 就绪标志自动生成。

发送并接收模式

此模式中，IIS 总线接口可以同时发送和接收。

音频串行接口格式

IIS 总线格式

IIS 总线有 4 根线，包括串行数据输入 (IISDI)、串行数据输出 (IISDO)、左/右通道选择 (IISLRCK) 和串行位时钟 (IISCLK)；主机是产生 IISLRCK 和 IISCLK 的器件。

串行数据按二进制补码首先发送 MSB。首先发送 MSB 是因为发送器和接收器可能存在不同的字长。发送器不需要知道接收器可以处理多少位，接收器也不需要知道发送了多少位。

当系统字长大于发送字长时，缩短（最低有效数据位设置为'0'）字以便发送。如果接收器收到的超过其字长的位，忽略 LSB 后面的位。另一方面，如果接收器收到少于其字长的位，缺少的位固定设置为'0'。因此，尽管 LSB 的位置是依赖于字长，但 MSB 有固定的位置。发送器在每当 IISLRCK 改变的一个时钟周期发送下个字的 MSB。

由发送器发送的串行数据会与时钟信号的上升（低到高）或下降（高到低）沿同步。然而串行数据必须在串行时钟信号的上升沿锁存到接收器中，因此当发送数据同步于上升沿时则有一些限制。

LR 通道选择线表明要发送的通道。IISLRCK 能在时钟信号的上升或下降沿被改变，但其不需要同步。从机中，此信号时钟信号的上升沿锁存。IISLRCK 线在发送 MSB 之前的时钟周期改变。这将允许从发送器给器件的串行数据的同步时序将建立发送。此外其允许接收器储存前一个字并且为下一个字清除输入。

MSB (LEFT)对齐

MSB/左对齐总线格式与 IIS 总线格式结构上相同。只是格式上与 IIS 总线格式不同，MSB 对齐格式实现了发送器每当 IISLRCK 改变时总是发送下一个字的 MSB。

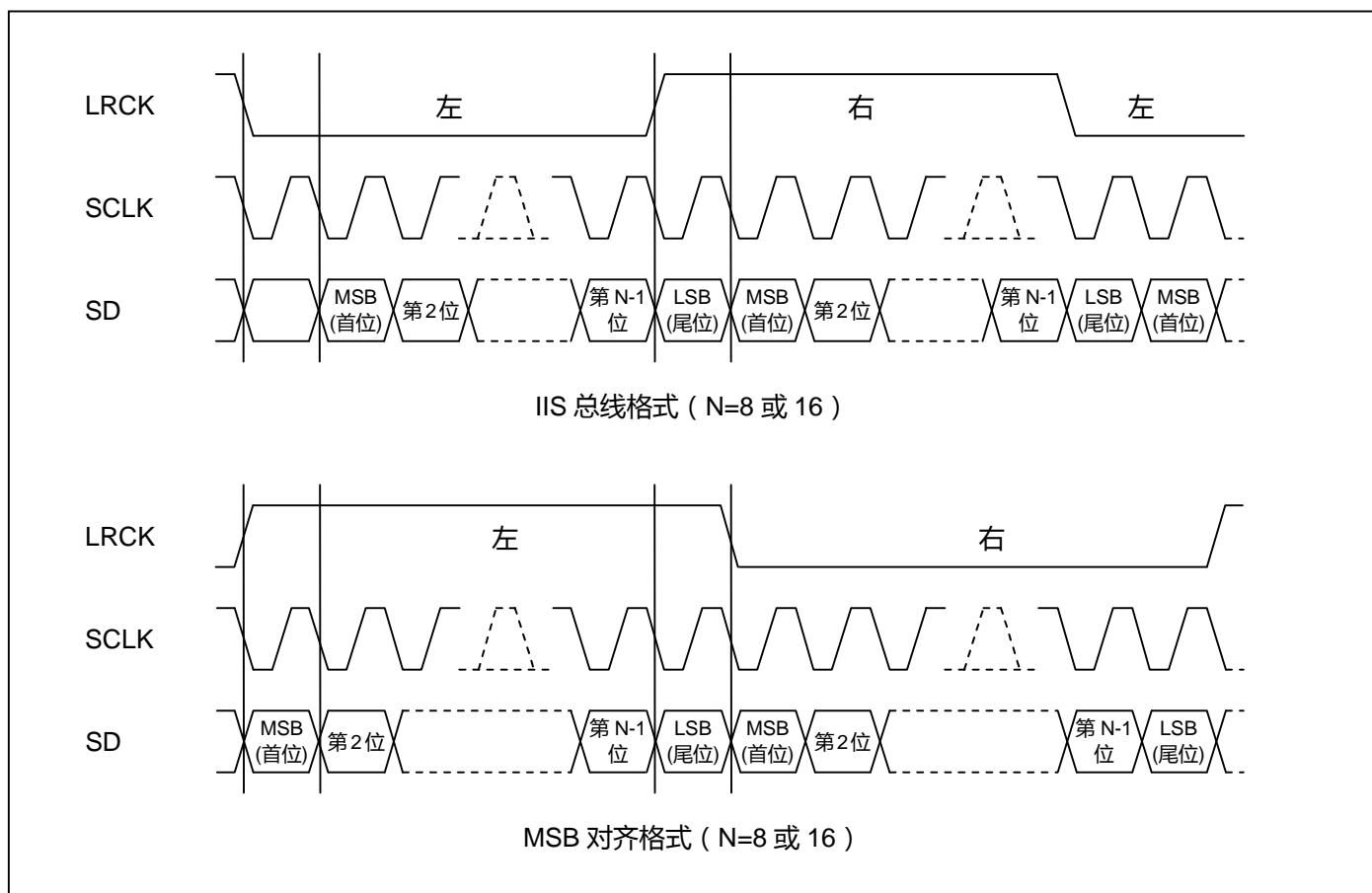


图 21-2. IIS 总线和 MSB (左) 对齐数据接口格式

采样频率和主时钟

主时钟频率 (PCLK 或 MPLLIn) 可以由采样频率选择 , 如表 21-1 所示。由于主时钟是由 IIS 预分频器配制的 , 应该适当的决定预分频值和主时钟类型 (256 或 384fs) 。串行位时钟频率类型 (16/32/48fs) 可以由串行位每通道和主时钟来选择 , 如表 21-2 所示。

表 21-1. CODEC 时钟 (CODECLK = 256 或 384fs)

IISLRCK (fs)	8.000 kHz	11.025 kHz	16.000 kHz	22.050 kHz	32.000 kHz	44.100 kHz	48.000 kHz	64.000 kHz	88.200 kHz	96.000 kHz
256fs										
CODECLK (MHz)	2.0480	2.8224	4.0960	5.6448	8.1920	11.2896	12.2880	16.3840	22.5792	24.5760
384fs										
	3.0720	4.2336	6.1440	8.4672	12.2880	16.9344	18.4320	24.5760	33.8688	36.8640

表 21-2. 可用串行位时钟频率 (IISCLK = 16 或 32 or 48fs)

串行位每通道	8 位	16 位
串行时钟频率 (IISCLK)		
@CODECLK = 256fs	16fs , 32fs	32fs
@CODECLK = 384fs	16fs , 32fs , 48fs	32fs , 48fs

IIS 总线接口特殊寄存器

IIS 控制 (IISCON) 寄存器

寄存器	地址	R/W	描述	复位值
IISCON	0x55000000(Li/HW,Li/W,Bi/W) 0x55000002(Bi/HW)	R/W	IIS 控制寄存器	0x100

IISCON	位	描述	初始状态
左/右通道指示 (只读)	[8]	0 = 左 1 = 右	1
发送 FIFO 就绪标志 (只读)	[7]	0 = 空 1 = 非空	0
接收 FIFO 就绪标志 (只读)	[6]	0 = 满 1 = 未满	0
发送 DMA 服务请求	[5]	0 = 禁止 1 = 使能	0
接收 DMA 服务请求	[4]	0 = 禁止 1 = 使能	0
发送通道空闲命令	[3]	空闲状态中 IISLRCK 为无效 (暂停 Tx) 0 = 非空闲 1 = 空闲	0
接收通道空闲命令	[2]	空闲状态中 IISLRCK 为无效 (暂停 Rx) 0 = 非空闲 1 = 空闲	0
IIS 预分频器	[1]	0 = 禁止 1 = 使能	0
IIS 接口	[0]	0 = 禁止 (停止) 1 = 使能 (启动)	0

注释：

1. 可以使用在大/小端，模式中 STRB/STRH/STR 和 LDRB/LDRH/LDR 指令或 char/short int/int 类型指针访问 IISCON 寄存器的每个字节、半字和字单元。
2. (Li/HW/W): 小端/半字/字； (Bi/HW/W): 大端/半字/字。

IIS 模式 (IISMOD) 寄存器

寄存器	地址	R/W	描述	复位值
IISMOD	0x55000004(Li/W,Li/HW,Bi/W) 0x55000006(Bi/HW)	R/W	IIS 模式寄存器	0x0

IISMOD	位	描述	初始状态
主时钟选择	[10]	主时钟选择 0 = PCLK 1 = MPPLin	0
主/从机模式选择	[8]	0 = 主机模式 (IISLRCK 和 IISCLK 为输出模式) 1 = 从机模式 (IISLRCK 和 IISCLK 为输入模式)	0
发送/接收模式选择	[7:6]	00 = 无传输 01 = 接收模式 10 = 发送模式 11 = 发送并接收模式	00
左/右通道的有效电平	[5]	0 = 主机模式 (IISLRCK 和 IISCLK 为输出模式) 1 = 从机模式 (IISLRCK 和 IISCLK 为输入模式)	0
串行接口格式	[4]	0 = IIS 兼容格式 1 = MSB (左) 对齐格式	0
串行数据每通道	[3]	0 = 8 位 1 = 16 位	0
主时钟频率选择	[2]	0 = 256fs (fs : 采样频率) 1 = 384fs	0
串行位时钟频率选择	[1:0]	00 = 16fs 01 = 32fs 10 = 48fs 11 = N/A	00

注释：

1. 可以使用在大/小端，模式中 STRB/STRH/STR 和 LDRB/LDRH/LDR 指令或 char/short int/int 类型指针访问 IISMOD 寄存器的每个字节、半字和字单元。
2. (Li/HW/W): 小端/半字/字； (Bi/HW/W): 大端/半字/字。

IIS 预分频 (IISPSR) 寄存器

寄存器	地址	R/W	描述	复位值
IISPSR	0x55000008(Li/HW,Li/W,Bi/W) 0x5500000A(Bi/HW)	R/W	IIS 预分频寄存器	0x0

IISPSR	位	描述	初始状态
预分频器控制 A	[9:5]	数值：0 至 31 注释：预分频器 A 生成了用于内部模块的主时钟，并且分频系数为 N+1。	00000
预分频器控制 B	[4:0]	数值：0 至 31 注释：预分频器 B 生成了用于外部的模块主时钟，并且分频系数为 N+1。	00000

注释：

1. 可以使用在大/小端，模式中 STRB/STRH/STR 和 LDRB/LDRH/LDR 指令或 char/short int/int 类型指针访问 IISPSR 寄存器的每个字节、半字和字单元。
2. (Li/HW/W): 小端/半字/字； (Bi/HW/W): 大端/半字/字。

IIS FIFO 控制 (IISFCON) 寄存器

寄存器	地址	R/W	描述	复位值
IISFCON	0x5500000C(Li/HW,Li/W,Bi/W) 0x5500000E(Bi/HW)	R/W	IIS FIFO 接口寄存器	0x0

IISFCON	位	描述	初始状态
发送 FIFO 访问模式选择	[15]	0 = 普通 1 = DMA	0
接收 FIFO 访问模式选择	[14]	0 = 普通 1 = DMA	0
发送 FIFO	[13]	0 = 禁止 1 = 使能	
接收 FIFO	[12]	0 = 禁止 1 = 使能	
发送 FIFO 数据计数 (只读)	[11:6]	数据计数值 = 0 至 32	000000
接收 FIFO 数据计数 (只读)	[5:0]	数据计数值 = 0 至 32	000000

注释：

1. 可以使用在大/小端，模式中 STRB/STRH/STR 和 LDRB/LDRH/LDR 指令或 char/short int/int 类型指针访问 IISFCON 寄存器的每个字节、半字和字单元。
2. (Li/HW/W): 小端/半字/字； (Bi/HW/W): 大端/半字/字。

IIS FIFO (IISFIFO) 寄存器

寄存器	地址	R/W	描述	复位值
IISFIFO	0x55000010(Li/HW) 0x55000012(Bi/HW)	R/W	IIS FIFO 寄存器	0x0

IISFIFO	位	描述	初始状态
FENTRY	[15:0]	IIS 的发送/接收数据	0x0

注释：

1. 可以使用在大/小端模式中 STRH 和 LDRH 指令或 short int/int 类型指针访问 IISFIFO 寄存器的每个半字和字单元。
2. (Li/HW): 小端/半字; (Bi/HW): 大端/半字。

22 SPI

概述

S3C2440A 的串行外设接口 (**SPI**) 可以与串行数据传输连接。S3C2440A 包含了 2 个 SPI , 每个都有 2 个分别用于发送和接收的 8 位移位寄存器。一次 SPI 传输期间 , 同时发送 (串行移出) 和接收 (串行移入) 数据。由相应控制寄存器设置指定 8 位串行数据的频率。如果只希望发送 , 则接收数据可以保持伪位 (*dummy*) 。此外如果只希望接收 , 则需要发送伪位'1'数据

特性

- 支持 2 个通道 SPI
- 兼容 SPI 协议 (2.11 版本)
- 8 位发送移位寄存器
- 8 位接收移位寄存器
- 8 位预分频逻辑
- 查询、中断和 DMA 传输模式

方框图

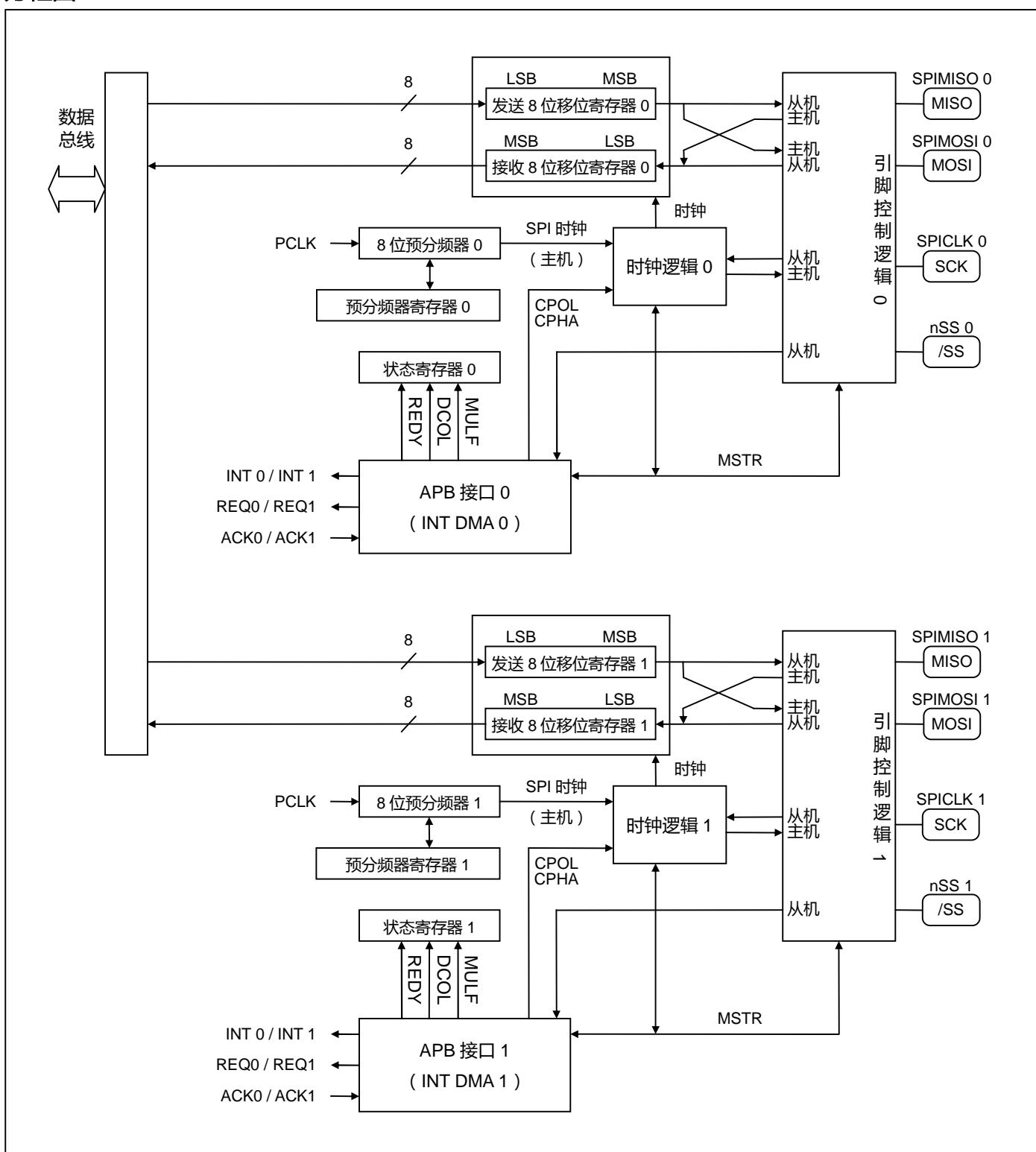


图 22-1. SPI 方框图

SPI 操作

通过使用 SPI 接口，S3C2440A 可以与外部器件同时发送/接收 8 位数据。与 2 根数据线同步串行时钟线来移位和采样信息。当 SPI 接口为主机时，可以通过设置 SPPREn 寄存器中的相应位来控制发送频率。可以修改其频率来调整波特率数据寄存器值。当 SPI 为从机时，由其它主机提供时钟。当程序员写字节数据到 SPTDATn 寄存器，将同时开始 SPI 的发送/接收操作。在一些情况下，应该在写字节数据到 SPTDATn 之前激活 nSS。

编程过程

当一个字节数据写入到 SPTDATn 寄存器中时，如果置位了 SPCONn 寄存器的 ENSCK 和 MSTR 则 SPI 开始发送。可以使用典型的编程步骤来操作一个 SPI 卡。

按照这些基本步骤来编程 SPI 模型：

1. 设置波特率预分频寄存器 (SPPREn)
2. 设置 SPCONn 来正确配制 SPI 模型
3. 写数据 0xFF 到 SPTDATn 10 次来初始化 MMC 或 SD 卡
4. 设置起 nSS 作用的 GPIO 引脚为低来激活 MMC 或 SD 卡
5. 发送数据；检查发送就绪标志的状态 (READY = 1)，并接着写数据到 SPTDATn。
6. 接收数据 (1)：SPCONn 的 TAGD 位禁止 = 普通模式
7. ; 写 0xFF 到 SPTDATn，然后确认 READY 的置位，并接着从读缓冲器读取数据
8. 接收数据 (2)：SPCONn 的 TAGD 位使能 = 自动发送杂数据模式
9. ; 确认 READY 的置位，并接着从读缓冲器读取数据 (然后自动开始传输)
10. 设置起 nSS 作用的 GPIO 引脚为高来释放 MMC 或 SD 卡

SPI 传输格式

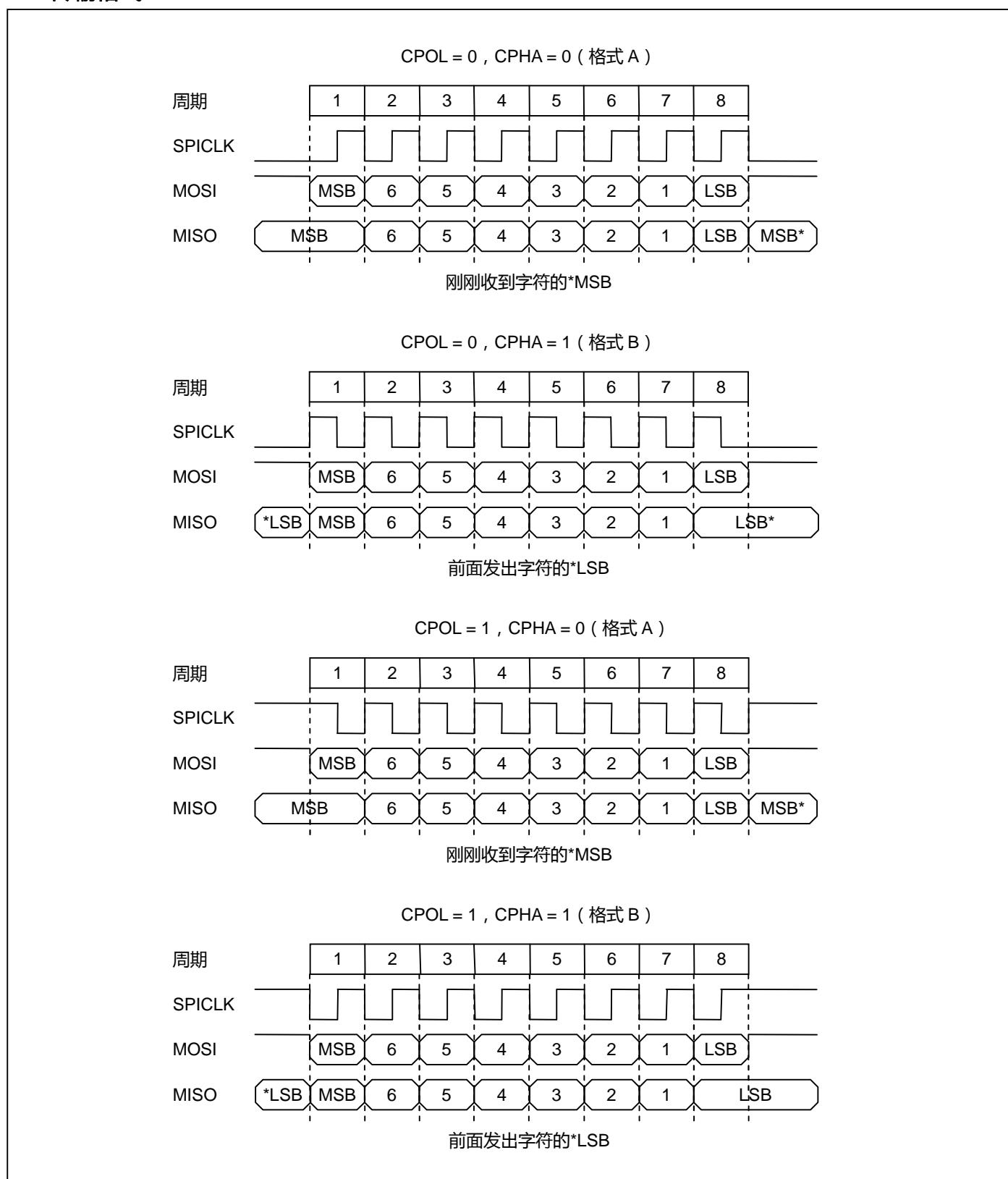


图 22-2. SPI 传输格式

DMA 发送过程

1. 配制 SPI 为 DMA 模式
2. 正确的配制 DMA
3. SPI 请求 DMA 服务
4. DMA 发送 1 字节数据给 SPI
5. SPI 发送数据给卡
6. 返回步骤 3，直到 DMA 计数变为 0
7. SMOD 位配制 SPI 为中断或查询模式

DMA 接收过程

1. SMOD 位配制 SPI 为 DMA 模式并且置位 TAGD 位
2. 正确的配制 DMA
3. SPI 从卡接收 1 字节数据
4. SPI 请求 DMA 服务
5. DMA 接收来自 SPI 的数据
6. 自动写数据 0xFF 到 SPTDATn
7. 返回步骤 4，直到 DMA 计数变为 0
8. SMOD 位配制 SPI 为查询模式并且清零 TAGD 位
9. 如果 SPSTAn 的 READY 标志置位，然后读取最后字节数据

注释：

1. 总共收到的数据 = DMA TC 值 + 查询模式中的最后数据（步骤 9）。
2. 主机或从机模式中，如果希望使用 DMA 则应该设置 TAGD 位为‘1’。

SPI 特殊寄存器

SPI 控制寄存器

寄存器	地址	R/W	描述	复位值
SPCON0	0x59000000	R/W	SPI 通道 0 控制寄存器	0x00
SPCON1	0x59000020	R/W	SPI 通道 1 控制寄存器	0x00

SPCONn	位	描述	初始状态
SMOD	[6:5]	SPI 模式选择。决定如何读/写 SPTDAT 00 = 查询模式 01 = 中断模式 10 = DMA 模式 11 = 保留	00
ENSCK	[4]	SCK 使能。决定是否希望 SCK 使能 (主机) 0 = 禁止 1 = 使能	0
MSTR	[3]	主/从机选择。决定希望的模式 (主机或从机) 0 = 禁止 1 = 使能 注意： 从机模式中，这需要主机初始化 Tx/Rx 的建立时间。	0
CPOL	[2]	时钟极性选择。决定时钟是高电平有效还是低电平有效 0 = 低电平有效 1 = 高电平有效	0
CPHA	[1]	时钟相位选择。选择 2 种基本不同传输格式之一 0 = 格式 A 1 = 格式 B	0
TAGD	[0]	自动发送杂数据模式使能。决定是否必须接收数据 0 = 普通模式 1 = 自动发送杂数据模式 注意： 普通模式中，如果只希望接收数据则应该发送空 0xFF 数据。	0

SPI 状态寄存器

寄存器	地址	R/W	描述	复位值
SPSTA0	0x59000004	R	SPI 通道 0 状态寄存器	0x01
SPSTA1	0x59000024	R	SPI 通道 1 状态寄存器	0x01

SPSTAn	位	描述	初始状态
保留	[7:3]	-	-
DCOL	[2]	数据冲突错误标志。如果当传输正在进行中时写了 SPTDATn 或读了 SPRDATn 此标志置位，并且可以通过读取 SPSTAn 清除。 0 = 未发现 1 = 发生冲突错误	0
MULF	[1]	多主机错误标志。当 SPI 配置为主机时如果 nSS 信号变为有效低电平并且 SPPINn 的 ENMUL 位为多主机错误检测模式则置位此标志。通过读取 SPSTAn 清除 MULF。 0 = 未发现 1 = 发现多主机错误	0
READY	[0]	传输就绪标志。此位指示出 SPTDATn 或 SPRDATn 准备好了发送或接收。通过写数据到 SPTDATn 自动清零此标志。 0 = 未就绪 1 = 数据 Tx/Rx 就绪	1

SPI 引脚控制寄存器

当 SPI 系统使能时，除 nSS 引脚外的其它引脚的方向是由 SPCONn 寄存器的 MSTR 位控制的。nSS 引脚的方向通常为输入。

当 SPI 为主机时，如果 SPPIN 的 ENMUL 位有效，则 nSS 引脚被用于检查多主机错误，并且应该使用另一个 GPIO 来选择一个从机。

如果 SPI 配制为从机，nSS 引脚用于主机选择作为从机的 SPI。

寄存器	地址	R/W	描述	复位值
SPPIN0	0x59000008	R/W	SPI 通道 0 引脚控制寄存器	0x00
SPPIN1	0x59000028	R/W	SPI 通道 1 引脚控制寄存器	0x00

SPPINn	位	描述	初始状态
保留	[7:3]	-	-
ENMUL	[2]	多主机错误检测使能。当 SPI 系统为主机时 nSS 引脚用作输入来检测多主机错误。 0 = 禁止 (通用) 1 = 多主机错误检测使能	
保留	[1]	-	
KEEP	[0]	决定当 1 字节发送完成时 MOSI 的驱动或释放 (主机)。 0 = 释放 1 = 驱动为之前电平	

SPIMISO(MISO)和 SPIMOSI(MOSI)数据引脚是用于发送和接收串行数据。当 SPI 配制为主机时，SPIMISO (MISO)为主机数据输入线，SPIMOSI (MOSI)为主机数据输出线，SPICLK (SCK)为时钟输出线。当 SPI 变为从机时，这些引脚执行相反作用。多主机系统中，分别配制 SPICLK (SCK)引脚、SPIMOSI (MOSI)引脚和 SPIMISO (MISO)引脚作为一组。当其它 SPI 器件工作在主机选择 S3C2440A SPI 作为从机时主机 SPI 可以发觉多主机错误。当察觉到这种错误时，立即执行以下动作。但是如果希望检测此错误则必须事先置位 SPPINn 的 ENMUL 位。

1. SPCONn 的 MSTR 位强制为 0 来工作在从机模式中。
2. 置位 SPSTAn 的 MULF 标志并且发生 SPI 中断。

SPI 波特率预分频寄存器

寄存器	地址	R/W	描述	复位值
SPPRE0	0x5900000C	R/W	SPI 通道 0 波特率预分频寄存器	0x00
SPPRE1	0x5900002C	R/W	SPI 通道 1 波特率预分频寄存器	0x00

SPPREn	位	描述	初始状态
预分频值	[7:0]	决定 SPI 时钟率。 波特率 = PCLK / 2 / (预分频值 + 1)	0x00

注意：波特率应该低于 25 MHz。

SPI 发送数据寄存器

寄存器	地址	R/W	描述	复位值
SPTDAT0	0x59000010	R/W	SPI 通道 0 Tx 数据寄存器	0x00
SPTDAT1	0x59000030	R/W	SPI 通道 1 Tx 数据寄存器	0x00

SPTDATn	位	描述	初始状态
Tx 数据寄存器	[7:0]	此字段包含通过 SPI 通道要发送的数据	0x00

SPI 接收数据寄存器

寄存器	地址	R/W	描述	复位值
SPRDAT0	0x59000014	R	SPI 通道 0 Rx 数据寄存器	0xFF
SPRDAT1	0x59000034	R	SPI 通道 1 Rx 数据寄存器	0xFF

SPRDAT n	位	描述	初始状态
Rx 数据寄存器	[7:0]	此字段包含通过 SPI 通道接收到的数据	0xFF

23 摄像头接口

概述

此章将讲解规格并阐述摄像头接口。S3C2440A 中的 **CAMIF(摄像头接口)**是由 7 部分组成的一——**图案 MUX**、**捕获单元**、**预览缩放器**、**编码缩放器**、**预览 DMA**、**编码 DMA** 和 **SFR (特殊功能寄存器)**。CAMIF 支持 ITU-R BT.601/656 YCbCr 8 位标准。最大输入尺寸为 4096x4096 像素(2048x2048 像素用于缩放)并且存在 2 个缩放器。预览缩放器是专用于产生小尺寸图像如 **PIP (画中画)** 和编码缩放器是专用于产生编码有用的图像如平面类型 YCbCr 4:2:0 或 4:2:2。2 个主 DMA 可以为移动平台对捕获到的图像镜像和旋转。这些特性对于翻盖手机非常有用并且产生的测试图案可用于校准如 CAMHREF、CAMVSYNC 的输入同步信号中。此外，视频同步信号和像素时钟的极性可以通过设置寄存器在 CAMIF 方面中被反转。

特性

- ITU-R BT.601/656 8 位模式外部接口支持
- DZI (数字放大) 功能
- 可编程视频同步信号的极性
- 最大无缩放 4096x4096 像素输入支持 (可缩放 2048x2048 像素支持输入)
- 最大 4096x4096 像素输出支持给编码通路
- 最大 640x480 像素输出支持给预览通路
- 图像镜像和旋转 (X 轴镜像、Y 轴镜像和 180° 旋转)
- PIP 和编码输入图像产生 (RGB 16/24 位格式和 YCbCr 4:2:0/4:2:2 格式)

信号描述

表 23-1. 摄像头接口信号描述

名称	I/O	有效	描述
CAMPCLK	I	-	像素时钟，由摄像头处理器驱动
CAMVSYNC	I	H/L	帧同步，由摄像头处理器驱动
CAMHREF	I	H/L	水平同步，由摄像头处理器驱动
CAMDATA[7:0]	I	-	像素数据，由摄像头处理器驱动
CAMCLKOUT	O	-	给摄像头处理器的主时钟
CAMRESET	O	H/L	摄像头处理器软件复位或掉电

注释：

I/O 方向是在 AP 方面。I: 输入，O: 输出。

方框图

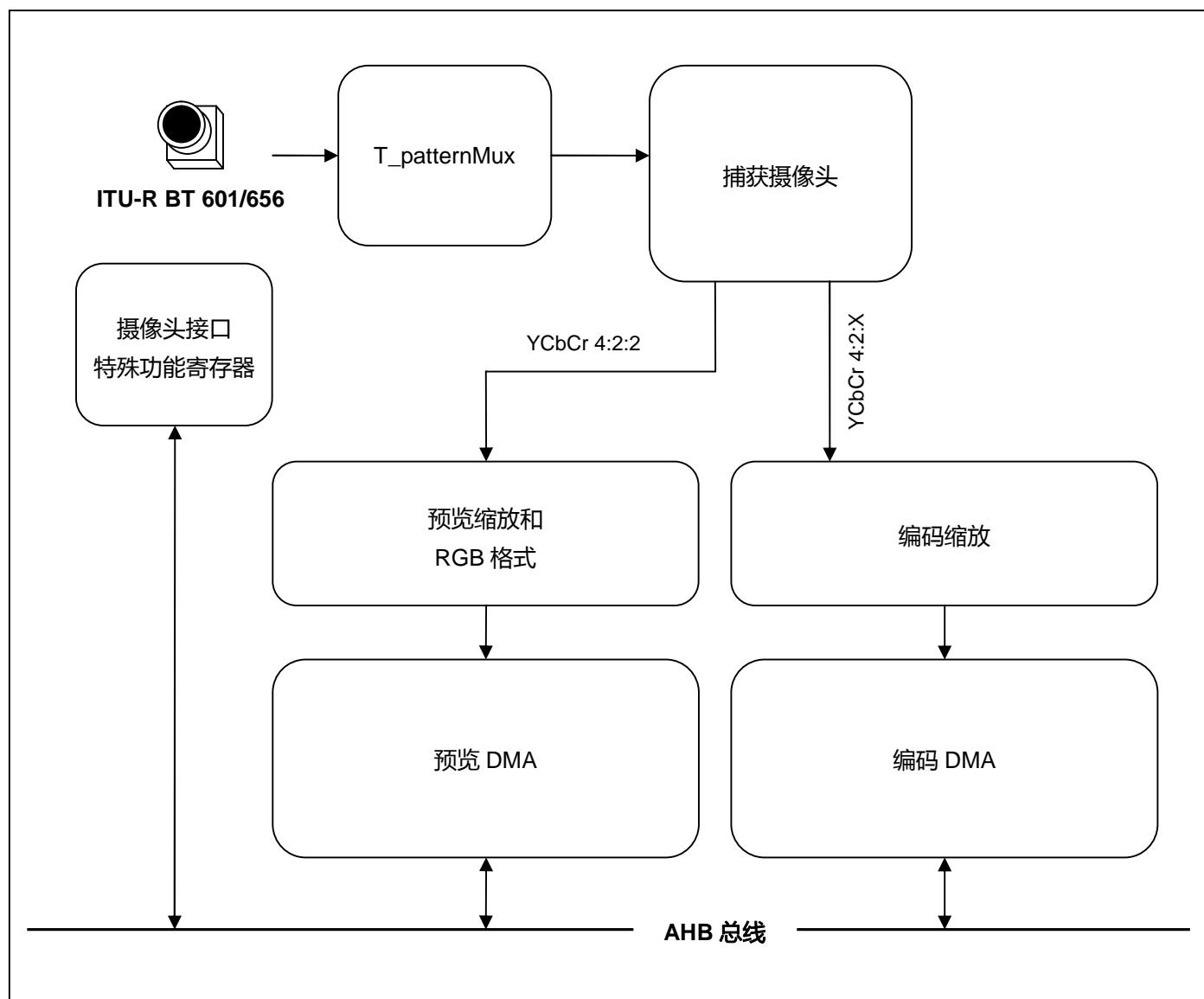


图 23-1. CAMIF 概况

时序图

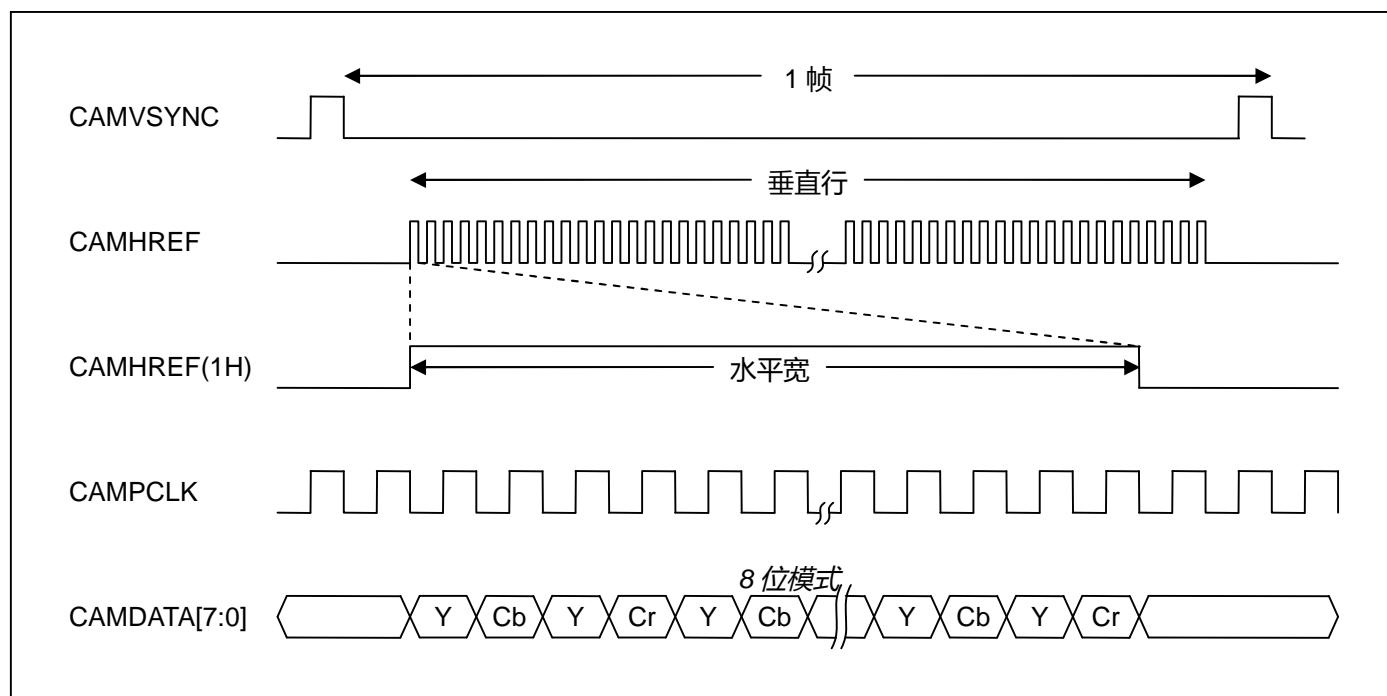


图 23-2. ITU-R BT 601 输入时序图

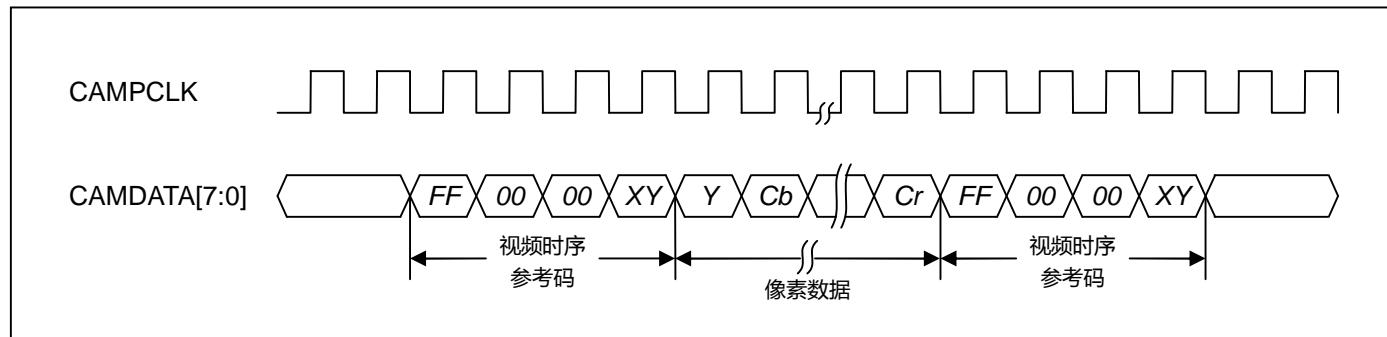


图 23-3. ITU-R BT 656 输入时序图

ITU-R BT 656 格式中有两个时序参考信号，一个是在每次视频数据块开始（有效视频的开始，SAV），另一个是在每次视频数据块结束（有效视频的结束，EAV），如图 23-3 和表 23-2 所示。

注意：

1. 应该在捕获开始后的 VSYNC 脉冲后激活 HREF。
2. 不能按照场消隐来周期性 SAV (H=0)。

表 23-2. ITU-656 格式的视频时序参考码

数据位号	第一个字	第二个字	第三个字	第四个字
9(MSB)	1	0	0	1
8	1	0	0	F
7	1	0	0	V
6	1	0	0	H
5	1	0	0	P3
4	1	0	0	P2
3	1	0	0	P1
2	1	0	0	P0
1(注释)	1	0	0	0
0	1	0	0	0

为兼容现有 8 位接口，位 D1 和 D0 的值没有定义。

F = 0 (场 1 期间), 1 (场 2 期间)

V = 0 (别处), 1 (场消隐期间)

H = 0 (SAV 中：有效视频的开始), 1 (EAV 中：有效视频的结束)

P0、P1、P2、P3 = 保护位

摄像头接口逻辑可以在收到如“FF-00-00”的数据后捕获像到 H (SAV、EAV) 和 V (帧同步) 的同步位。

注释：

所有外部摄像头接口 IO 都推荐为施密特触发器类型 IO 以降低噪声。

摄像头接口操作

两条 DMA 通路

CAMIF 有两条 DMA 通路 (Path)。P 通路 (预览通路) 和 C 通路 (编码通路) 在 AHB 总线上是各自分离的。从系统总线上看，两条通路是独立的。P 通路为 PIP 储存 RGB 图像数据到存储器中。C 通路为如 MPEG-4、H.263 等编码储存 YCbCr 4:2:0 或 4:2:2 格式数据到存储器中。着两条主通路可以支持多变的应用，像 DSC (数码相机)，MPEG-4 视频会议、视频录像等。例如，P 通路图像可以作为预览图像，并且 C 通路图像可以作为 DSC 应用中的 JPEG 图像。设置寄存器可以单独禁止 P 通路或 C 通路。

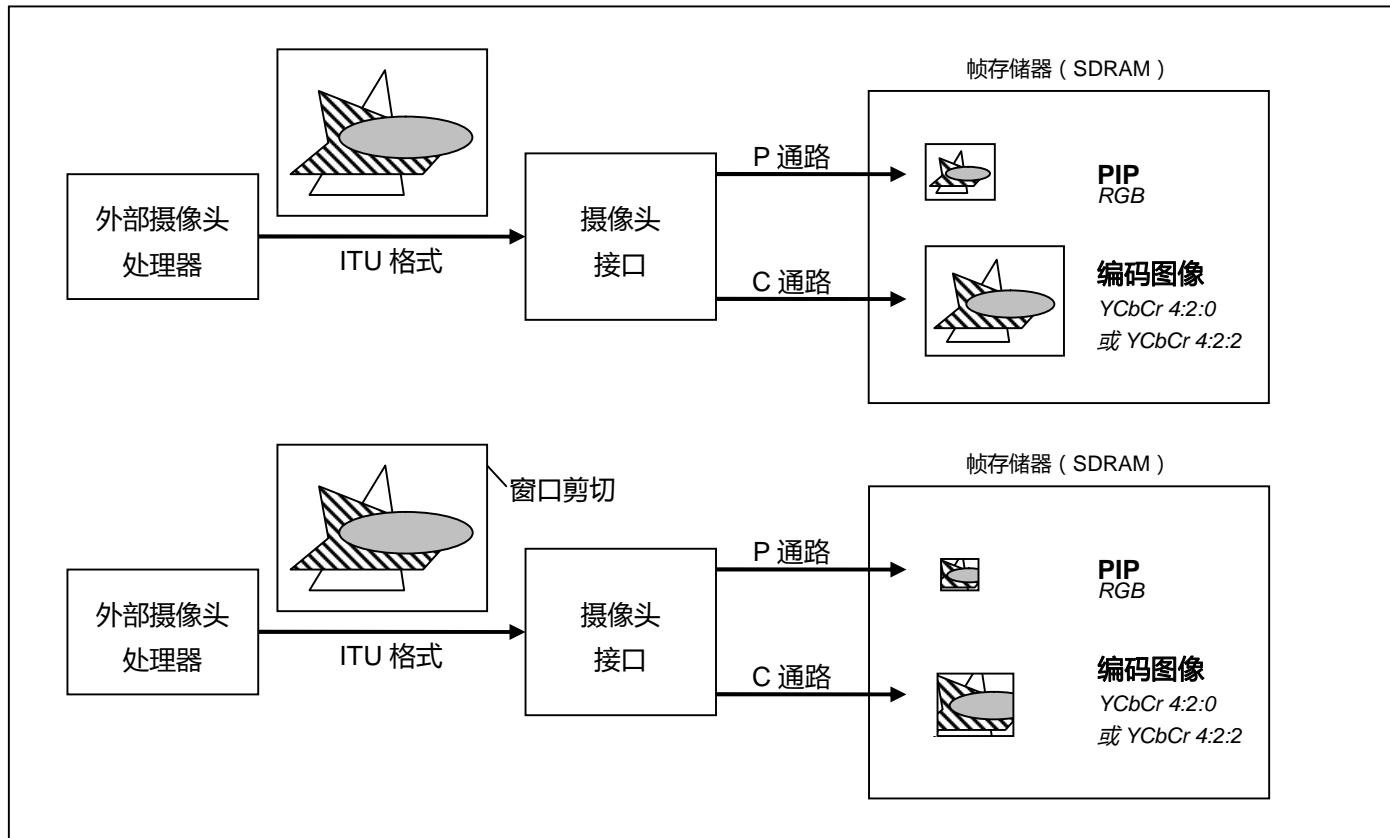


图 23-4. 两条 DMA 通路

时钟域

CAMIF 有两个时钟域。一个是系统总线时钟 **HCLK**。另一个是像素时钟 **CAMPCLK**。**系统总线时钟必须比像素时钟快。**图 23-5 显示了 CAMCLKOUT 必须从像素频率如 USB PLL 时钟分频得到。如果使用了外部时钟振荡器，则应该悬空 CAMCLKOUT。内部缩放时钟为系统时钟。这两个时钟域不需要相互同步。其它信号如 CAMPCLK 应该同样被连接到施密特触发器电平转换器。

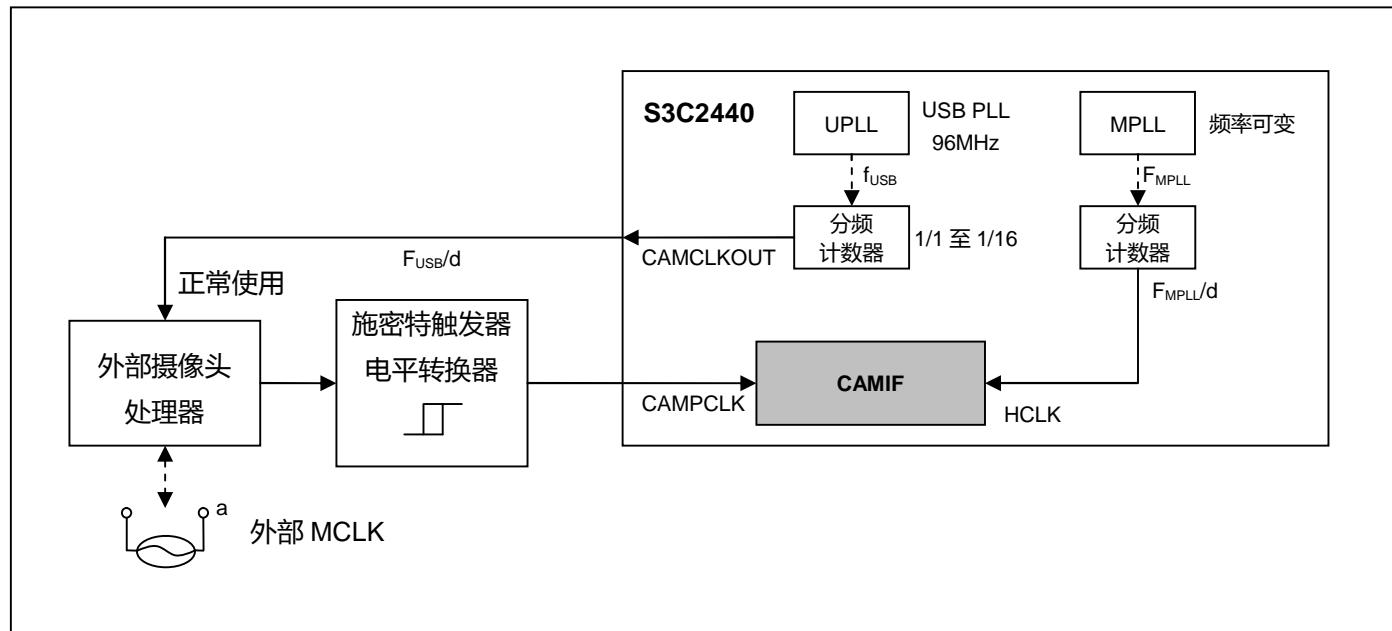


图 23-5. CAMIF 时钟产生

帧存储器体系

每个 P 通路和 C 通路的帧存储器由 4 个乒乓存储器组成，如图 23.6 所示。C 通路乒乓存储器有 3 个元素存储器——亮度 Y、色度 Cb 和色度 Cr。如果 AHB 总线传输不能在一次水平行周期中完成 DMA 操作，则它可能导致误运作。

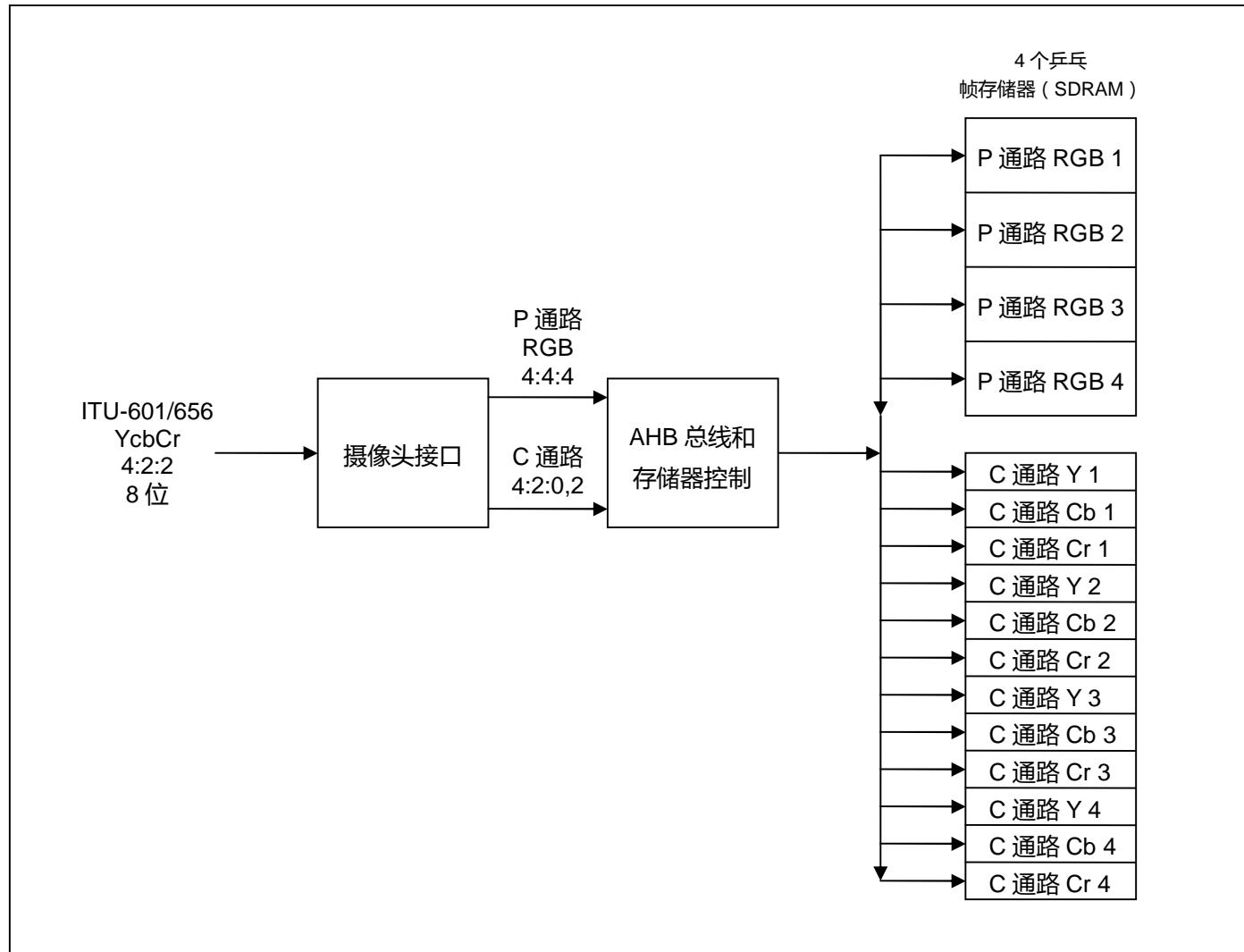


图 23-6. 乒乓存储器体系

存储器储存方式

编码通路中使用小端方式储存到帧存储器中。储存像素是从 LSB 到 MSB。AHB 总线传输的是 32 位字数据。因此，CAMIF 使得每个 Y-Cb-Cr 字按都为小端方式。对于预览通路，存在着两种不同格式。RGB 24 位格式的一个像素（1 个彩色像素）为一个字。另一方面，RGB 16 位格式的两个像素为一个字。请参考下图。

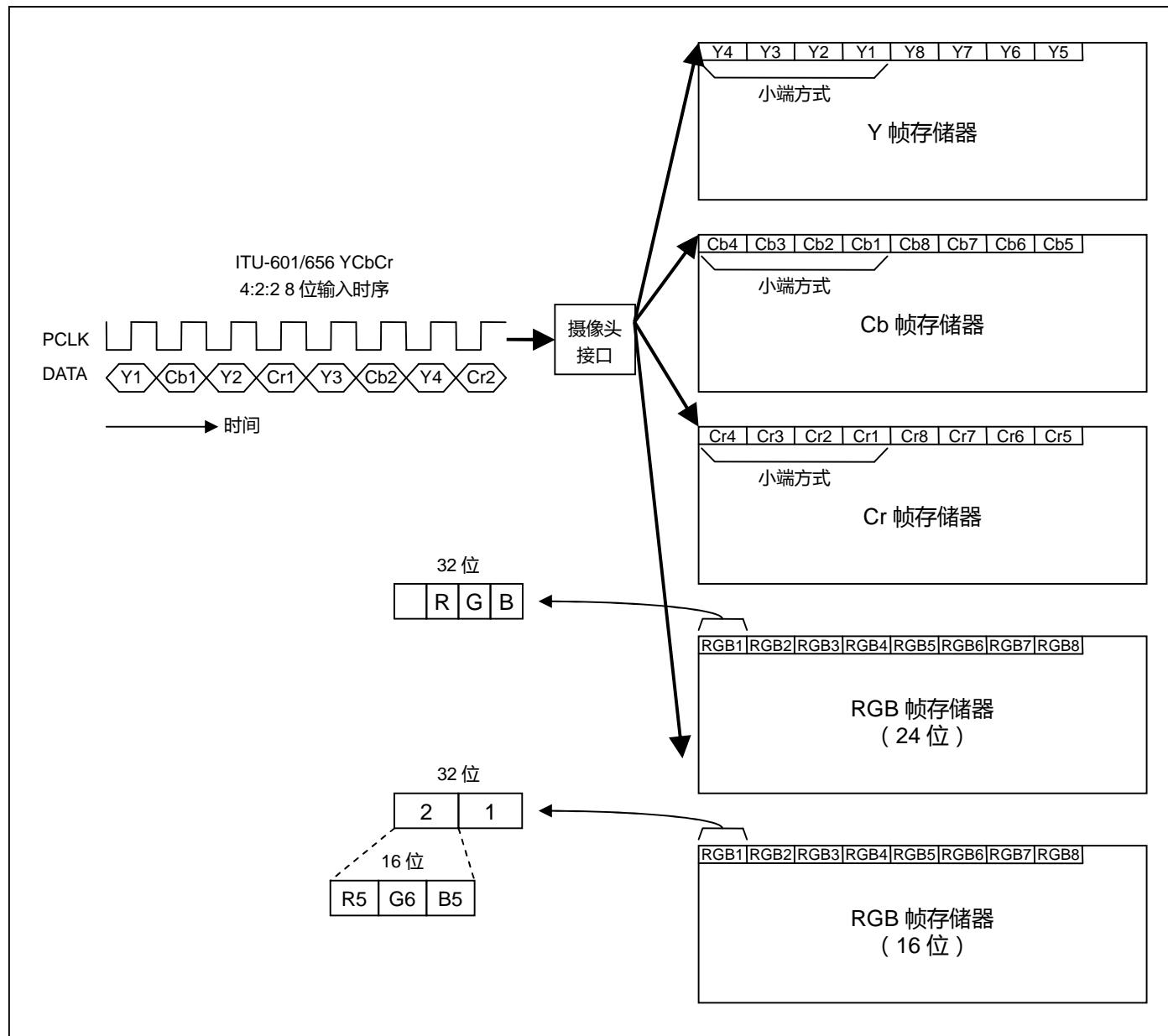


图 23-7. 存储器储存方式

寄存器设置的时序图

首先对帧捕获命令的寄存器设置可以在帧周期中的任何地方发生。但是推荐在第一次 CAMVSYNC 为“L”状态并且可以从状态 SFR(请见下页)读取 CAMVSYNC 信息时设置它。包括 ImgCptEn 在内的所有命令在 CAMVSYNC 的下降沿起效。但是请注意除第一个 SFR 设置之外，所有的命令都应该在 ISR (中断服务程序) 中进行编程。特别是当改变了触发大小的相关信息时应该禁止捕获操作。

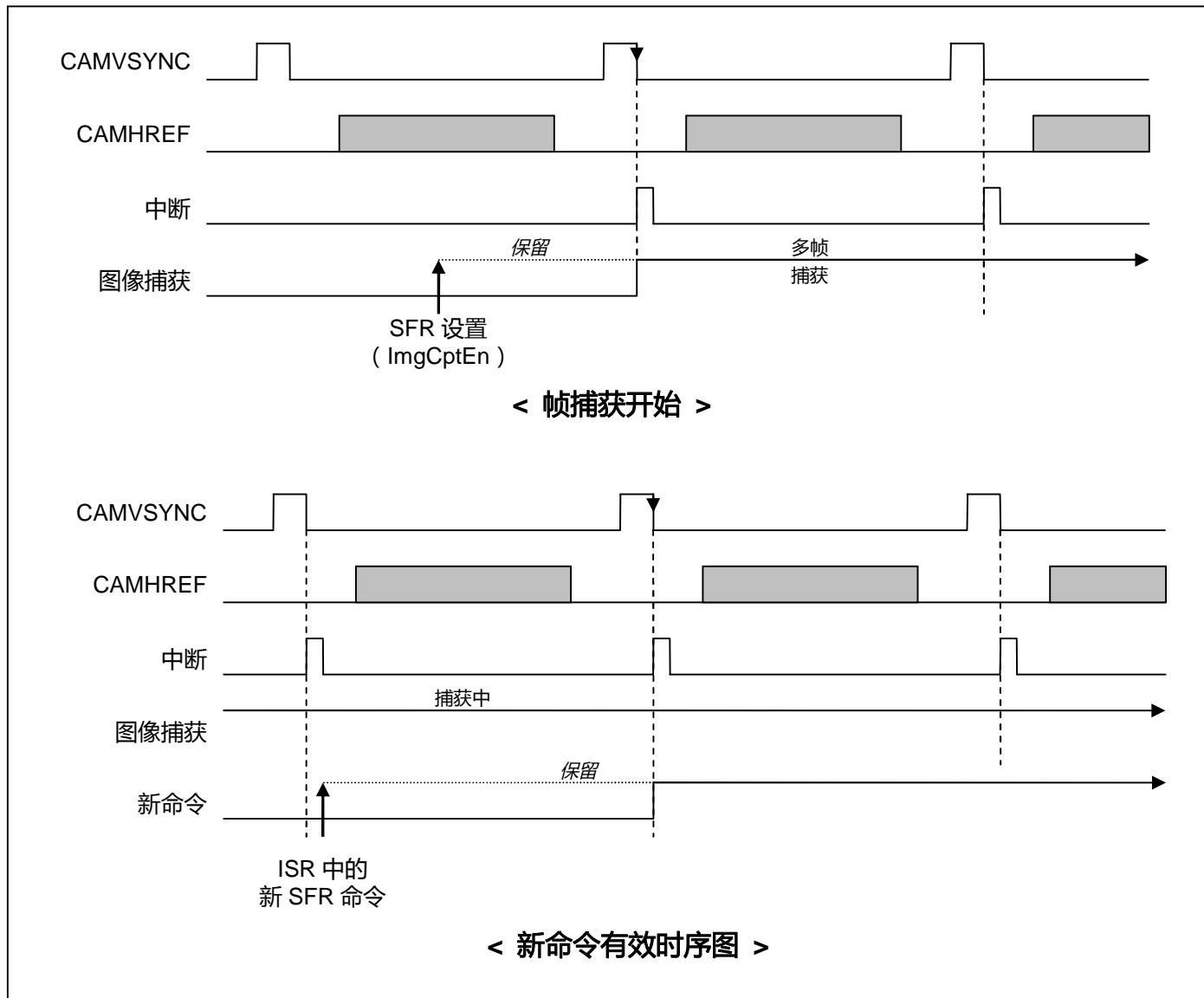


图 23-8. 寄存器设置的时序图

注意：

如果当预览通路工作时而编码通路未工作将发生编码通路的 FIFO 溢出。如果希望在这种情况下使用编码通路，应该停止预览通路并且通过使用 CIGCTRL 寄存器的 SwRst 位来复位 CSMIF。接着清除编码通路的溢出并且设置所希望的特殊功能寄存器。

最后 IRQ 的时序图

除最后 IRQ 以外的 IRQ 是在图像捕获之前产生的。最后 IRQ 的意思是可以通过以下时序图设置捕获结束。LastIRQEn 为自动清零并且，如上所述，ISR 中的 SFR 设置是给下一帧命令。因此对于恰当的 IRQ，LastIRQEn 与 ImgCptEn/ImgCptEn_CoSc /ImgCptEnPrSC 之间需要按以下顺序。推荐 ImgCptEn /ImgCptEn_CoSc /ImgCptEnPrSC 在 ISR 中的 SFR 设置的最后的同一时间设置。ISR 中读取的 FrameCnt 意思为下帧计数。在下图中，最后捕获帧计数为“1”。这表明帧 1 是帧 0 至帧 3 中的最后捕获帧。FrameCnt 在 IRQ 上升沿加 1。

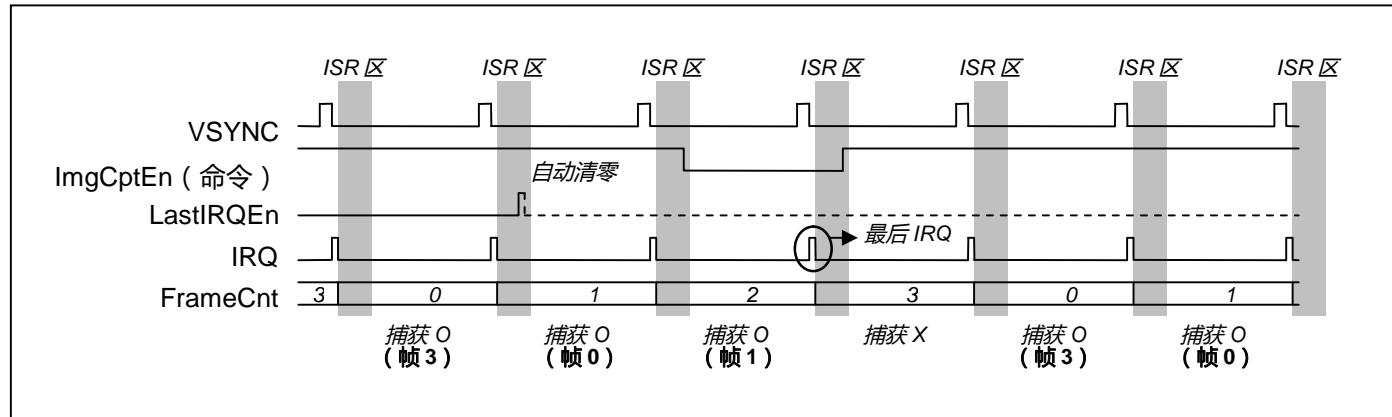


图 23-9. 最后 IRQ 的时序图

摄像头接口特殊寄存器

源帧寄存器

寄存器	地址	R/W	描述	复位值
CISRCFMT	0x4F000000	R/W	输入源帧寄存器	0

CISRCFMT	位	描述	初始状态
ITU601_656n	[31]	0 = ITU-R BT.656 YCbCr 8 位模式使能 1 = ITU-R BT.601 YCbCr 8 位模式使能	0
UVOffset	[30]	Cb、Cr 值偏移控制 0 = +0 (通常使用) 1 = +128	- 对于 YCbCr - 对于 YUV
保留	[29]	保留此位并且必须为 0。	0
SourceHsize	[28:16]	源水平像素数 (必须为 8 的倍数)	0
Order422	[15:14]	输入 8 位模式的输入 YcbCr 顺序信息 00 = YCbYCr 10 = CbYCrY	01 = YCrYCb 11 = CrYCbY
SourceVsize	[12:0]	源垂直像素数	

注释：

推荐以下顺序来预防编码通路中捕获操作的第一帧的 FIFO 溢出。

<ITU 601 格式>

1. CISRCFMT[31] <- '1'
2. 软件复位
3. 初始化摄像头接口
4. 开始捕获

<ITU 601 格式>

1. CISRCFMT[31] <- '1'
2. 软件复位
3. 初始化摄像头接口
4. ISRCFMT[31] <- '0' // 对于 ITU 656 格式
5. 开始捕获
6. 清除第一次 ISR 的编码溢出

窗口选择寄存器

寄存器	地址	R/W	描述	复位值
CIWDOFST	0x4F000004	R/W	窗口偏移寄存器	0

CIWDOFST	位	描述	初始状态
WinOfsEn	[31]	0 = 无偏移 1 = 窗口偏移使能	0
ClrOvCoFiY	[30]	0 = 正常 1 = 清除输入编码 FIFO Y 溢出指示标志	
WinHorOfst	[26:16]	窗口水平偏移 (此数为除 WinHorOfst×2 以外的水平像素数，必须为 8 的倍数) $* \text{WinHorOfst} \geq (\text{SourceHsize} - 640 \times \text{PreHorRatio_Pr}) / 2$	
ClrOvCoFiCb	[15]	0 = 正常 1 = 清除输入编码 FIFO Cb 溢出指示标志	
ClrOvCoFiCr	[14]	0 = 正常 1 = 清除输入编码 FIFO Cr 溢出指示标志	
ClrOvPrFiCb	[13]	0 = 正常 1 = 清除输入预览 FIFO Cb 溢出指示标志	
ClrOvPrFiCr	[12]	0 = 正常 1 = 清除输入预览 FIFO Cr 溢出指示标志	
WinVerOfst	[10:0]	窗口垂直偏移	

注释：

推荐在开始捕获操作前清除所有溢出位。

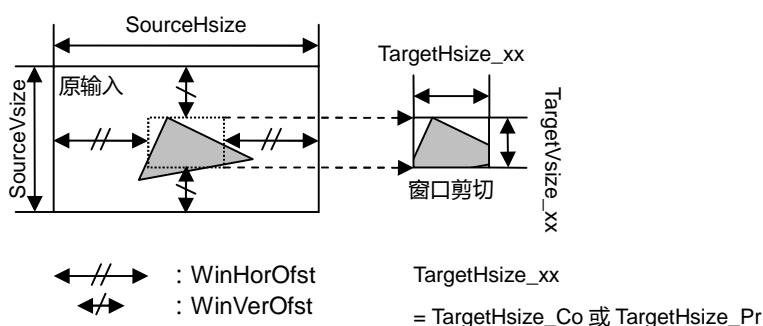


图 23-10. 窗口偏移示意图

全局控制寄存器

寄存器	地址	R/W	描述	复位值
CIGCTRL	0x4F000008	R/W	全局控制寄存器	0

CIGCTRL	位	描述	初始状态
SwRst	[31]	摄像头接口软件复位	0
CamRst	[30]	外部摄像头处理器复位或掉电	0
保留	[29]	保留此位并且该值必须为 1。	1
TestPattern	[28:27]	只有在 ITU-T 601 8 位模式时应该设置此寄存器。ITU-T 656 模式中将不会被允许。(最大 1280 × 1024) 00 = 外部摄像头处理器输入 (正常) 10 = 水平递增测试图案 01 = 彩色栅测试图案 11 = 垂直递增测试图案	0
InvPolCAMPCLK	[26]	0 = 正常 1 = 反转 CAMPCLK 的极性	0
InvPolCAMVSYNC	[25]	0 = 正常 1 = 反转 CAMVSYNC 的极性	0
InvPolCAMHREF	[24]	0 = 正常 1 = 反转 CAMHREF 的极性	0

Y1 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOYSA1	0x4F000018	R/W	编码 DMA 的 Y 第一帧开始地址	0

CICOYSA1	位	描述	初始状态
CICOYSA1	[31:0]	编码 DMA 的 Y 第一帧开始地址	0

注意：

缓冲器的地址必须为 1024 的倍数。

Y2 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOYSA2	0x4F00001C	R/W	编码 DMA 的 Y 第二帧开始地址	0

CICOYSA2	位	描述	初始状态
CICOYSA2	[31:0]	编码 DMA 的 Y 第二帧开始地址	0

Y3 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOYSA3	0x4F000020	R/W	编码 DMA 的 Y 第三帧开始地址	0

CICOYSA3	位	描述	初始状态
CICOYSA3	[31:0]	编码 DMA 的 Y 第三帧开始地址	0

Y4 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOYSA4	0x4F000024	R/W	编码 DMA 的 Y 第四帧开始地址	0

CICOYSA4	位	描述	初始状态
CICOYSA4	[31:0]	编码 DMA 的 Y 第四帧开始地址	0

CB1 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCBSA1	0x4F000028	R/W	编码 DMA 的 Cb 第一帧开始地址	0

CICOCBSA1	位	描述	初始状态
CICOCBSA1	[31:0]	编码 DMA 的 Cb 第一帧开始地址	0

CB2 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCBSA2	0x4F00002C	R/W	编码 DMA 的 Cb 第二帧开始地址	0

CICOCBSA2	位	描述	初始状态
CICOCBSA2	[31:0]	编码 DMA 的 Cb 第二帧开始地址	0

CB3 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCBSA3	0x4F000030	R/W	编码 DMA 的 Cb 第三帧开始地址	0

CICOCBSA3	位	描述	初始状态
CICOCBSA3	[31:0]	编码 DMA 的 Cb 第三帧开始地址	0

CB4 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCBSA4	0x4F000034	R/W	编码 DMA 的 Cb 第四帧开始地址	0

CICOCBSA4	位	描述	初始状态
CICOCBSA4	[31:0]	编码 DMA 的 Cb 第四帧开始地址	0

CR1 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCRSA1	0x4F000038	R/W	编码 DMA 的 Cr 第一帧开始地址	0

CICOCRSA1	位	描述	初始状态
CICOCRSA1	[31:0]	编码 DMA 的 Cr 第一帧开始地址	0

CR2 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCRSA2	0x4F00003C	R/W	编码 DMA 的 Cr 第二帧开始地址	0

CICOCRSA2	位	描述	初始状态
CICOCRSA2	[31:0]	编码 DMA 的 Cb 第二帧开始地址	0

CR3 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCRSA3	0x4F000040	R/W	编码 DMA 的 Cr 第三帧开始地址	0

CICOCRSA3	位	描述	初始状态
CICOCRSA3	[31:0]	编码 DMA 的 Cr 第三帧开始地址	0

CR4 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CICOCRSA4	0x4F000044	R/W	编码 DMA 的 Cr 第四帧开始地址	0

CICOCRSA4	位	描述	初始状态
CICOCRSA4	[31:0]	编码 DMA 的 Cr 第四帧开始地址	0

编码目标格式寄存器

寄存器	地址	R/W	描述	复位值
CICOTRGFMT	0x4F000048	R/W	编码 DMA 的目标图像格式	0

寄存器	位	描述	初始状态
In422_Co	[31]	0 = YCbCr 4:2:0 编码缩放器输入图像格式。这种情况下，在编码缩放器前已经执行了水平平行的减少（通常使用） 1 = YCbCr 4:2:2 编码缩放器输入图像格式。	0
Out422_Co	[30]	0 = YCbCr 4:2:0 编码缩放器输出图像格式。此模式常用于 MPEG-4 和硬件 JPEG DCT (通常使用) 1 = YCbCr 4:2:2 编码缩放器输出图像格式。此模式常用于软件 JPEG。	0
TargetHsize_Co	[28:16]	编码 DMA 的目标图像水平像素数（16 的倍数）	0
FlipMd_Co	[15:14]	编码 DMA 的图像镜像和旋转 00 = 正常 01 = X 轴镜像 10 = Y 轴镜像 11 = 180° 旋转	0
TargetVsize_Co	[12:0]	编码 DMA 的目标图像垂直像素数	0

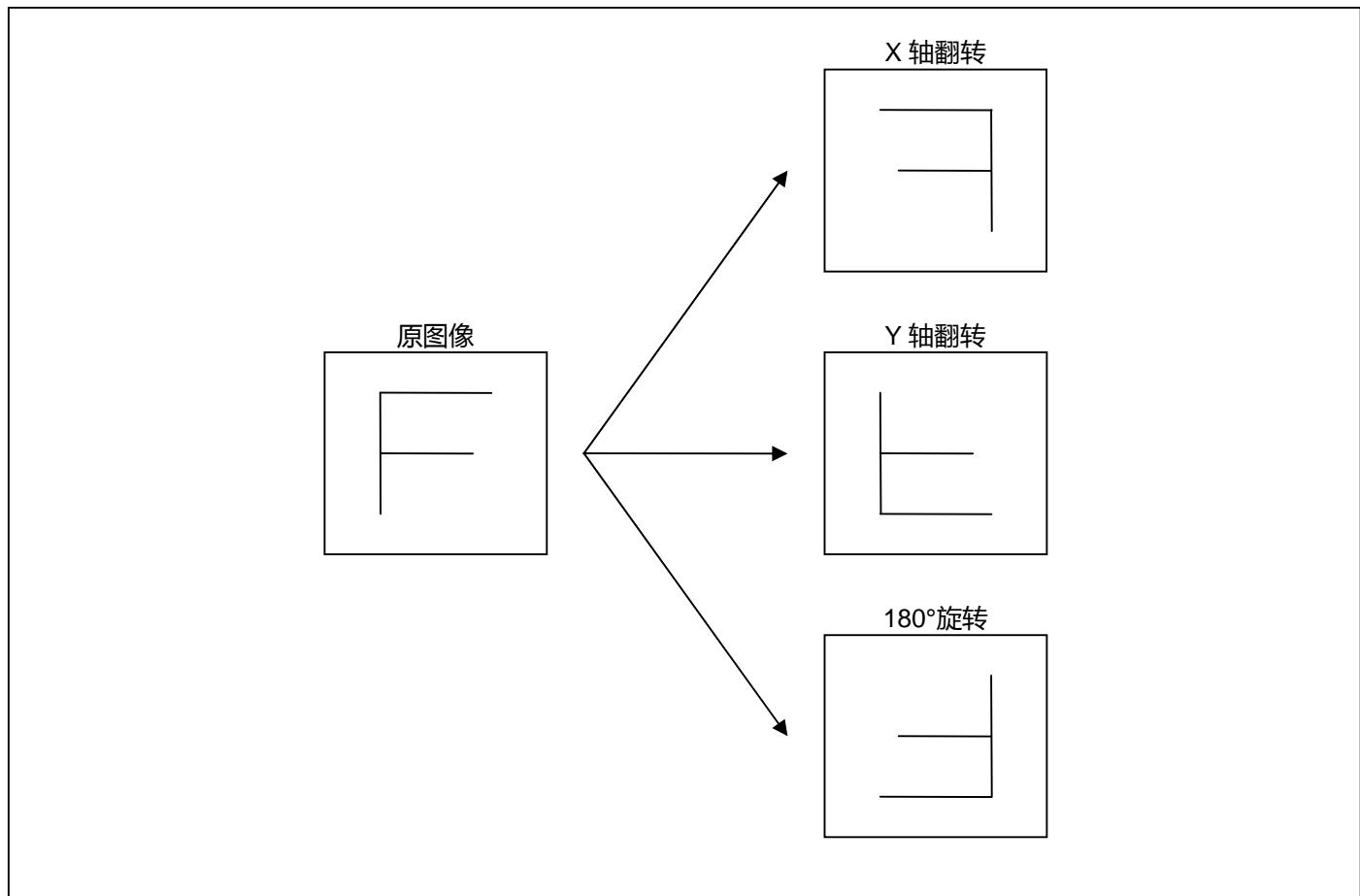


图 23-11. 图像镜像和旋转

编码 DMA 控制寄存器

寄存器	地址	R/W	描述	复位值
CICOCTRL	0x4F00004C	R/W	编码 DMA 控制相关	0

CICOCTRL	位	描述	初始状态
Yburst1_Co	[23:19]	编码 Y 帧的主突发长度	0
Yburst2_Co	[18:14]	编码 Y 帧的剩余突发长度	0
Cburst1_Co	[13:9]	编码 Cb/Cr 帧的主突发长度	0
Cburst2_Co	[8:4]	编码 Cb/Cr 帧的剩余突发长度	0
LastIRQEn_Co	[2]	0 = 正常 1 = 在帧捕获的结束时使能最后 IRQ (此位自动清零)	0

注释：

所有突发长度必须为 2、4、8 或 16 中的一个。

例 1：目标图像尺寸：QCIF (水平 Y 宽度 = 176 像素。1 像素 = 1 字节。1 字 = 4 像素)

$$176 / 4 = 44 \text{ 字}$$

$$44 \% 8 = 4 \rightarrow \text{主突发} = 8, \text{剩余突发} = 4$$

例 2：目标图像尺寸：VGA (水平 Y 宽度 = 640 像素。1 像素 = 1 字节。1 字 = 4 像素)

$$640 / 4 = 160 \text{ 字}$$

$$60 \% 16 = 0 \rightarrow \text{主突发} = 16, \text{剩余突发} = 16$$

例 3：目标图像尺寸：QCIF (水平 C 宽度 = 88 像素。1 像素 = 1 字节。1 字 = 4 像素)

$$88 / 4 = 22 \text{ 字}$$

$$22 \% 4 = 2 \rightarrow \text{主突发} = 4, \text{剩余突发} = 2 (\text{HTRANS==INCR})$$

编码缩放器和预览缩放器的寄存器设置指南

SRC_Width 和 **DST_Width** 满足字边界约束，这样水平像素数可以被表示为 kn ，其中 $n = 1, 2, 3, \dots, k = 1/2/8$ 分别给 24 bppRGB / 16bppRGB / YCbCr420 图像。TargetHsize 不应该大于 SourceHsize。同样 TargetVsize 不应该大于 SourceVsize

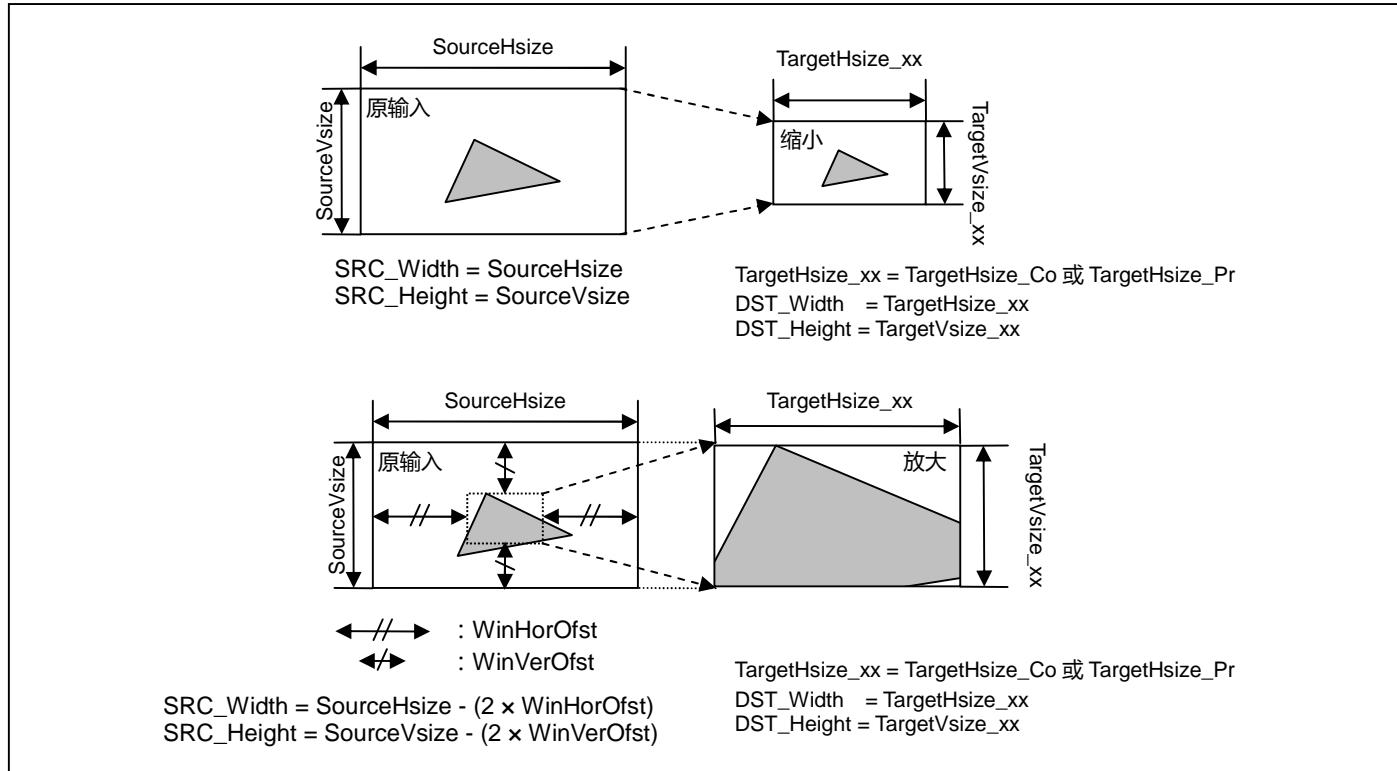


图 23-12. 缩放示意图

预缩放的其它控制寄存器如图像尺寸、预缩放率、预缩放变化率和主缩放率，都是按照以下等式定义的。

```

If ( SRC_Width >= 64 × DST_Width ) { Exit(-1); /* 超出水平缩放范围 */ }
else if (SRC_Width >= 32 × DST_Width) { PreHorRatio_xx = 32; H_Shift = 5; }
else if (SRC_Width >= 16 × DST_Width) { PreHorRatio_xx = 16; H_Shift = 4; }
else if (SRC_Width >= 8 × DST_Width) { PreHorRatio_xx = 8; H_Shift = 3; }
else if (SRC_Width >= 4 × DST_Width) { PreHorRatio_xx = 4; H_Shift = 2; }
else if (SRC_Width >= 2 × DST_Width) { PreHorRatio_xx = 2; H_Shift = 1; }
else { PreHorRatio_xx = 1; H_Shift = 0; }
PreDstWidth_xx = SRC_Width / PreHorRatio_xx;
MainHorRatio_xx = ( SRC_Width << 8 ) / ( DST_Width << H_Shift);

If ( SRC_Height >= 64 × DST_Height ) { Exit(-1); /* 超出垂直缩放范围 */ }
else if (SRC_Height >= 32 × DST_Height) { PreVerRatio_xx = 32; V_Shift = 5; }
else if (SRC_Height >= 16 × DST_Height) { PreVerRatio_xx = 16; V_Shift = 4; }
else if (SRC_Height >= 8 × DST_Height) { PreVerRatio_xx = 8; V_Shift = 3; }
else if (SRC_Height >= 4 × DST_Height) { PreVerRatio_xx = 4; V_Shift = 2; }
else if (SRC_Height >= 2 × DST_Height) { PreVerRatio_xx = 2; V_Shift = 1; }
else { PreVerRatio_xx = 1; V_Shift = 0; }
PreDstHeight_xx = SRC_Height / PreVerRatio_xx;
MainVerRatio_xx = ( SRC_Height << 8 ) / ( DST_Height << V_Shift );
SHfactor_xx = 10 - ( H_Shift + V_Shift );

```

注释：

预览通路包含 640 个像素行缓冲器。（编码通路包含 2048 个像素行缓冲器）因此最大为 1280 像素，输入图像必须超过 1/2 的预缩放来捕获有效图像。 $((SourceHsize - 2 \times WinHorOfst) / PreHorRatio_Pr) <= 640$

编码预缩放控制寄存器 1

寄存器	地址	R/W	描述	复位值
CICOSCPRARATIO	0x4F000050	R/W	编码预缩放率控制	0

CICOSCPRARATIO	位	描述	初始状态
SHfactor_Co	[31:28]	编码预缩放的变化系数	0
PreHorRatio_Co	[22:16]	编码预缩放的水平比	0
PreVerRatio_Co	[6:0]	编码预缩放的垂直比	0

编码预缩放控制寄存器 2

寄存器	地址	R/W	描述	复位值
CICOSCPREDST	0x4F000054	R/W	编码预缩放目标格式	0

CICOSCPREDST	位	描述	初始状态
PreDstWidth_Co	[27:16]	编码预缩放的目标宽度	0
PreDstHeight_Co	[11:0]	编码预缩放的目标高度	0

编码主缩放控制寄存器

寄存器	地址	R/W	描述	复位值
CICOSCCTRL	0x4F000058	R/W	编码主缩放控制	0

CICOSCCTRL	位	描述	初始状态
ScalerBypass_Co	[31]	编码缩放器旁路给最高 2048×2048 尺寸（这种情况下，ImgCptEn_CoSC 和 ImgCptEn_PrSC 应该为 0，但 ImgCptEn 应该为 1。不允许其捕获预览图像。此模式打算捕获 JPEG 输入图像给 DSC 应用）。这种情况下，输入像素缓冲只取决于输入 FIFO，因此在此模式中系统总线不应该为忙。	0
ScaleUpDown_Co	[30:29]	编码缩放器的缩小/放大标志（1:1 缩放比中，此位应该为“1”） 00 = 缩小 11 = 放大	00
MainHorRatio_Co	[24:16]	编码主缩放的水平缩放比	0
CoScalerStart	[15]	编码缩放器开始	0
MainVerRatio_Co	[8:0]	编码主缩放的垂直缩放比	0

编码 DMA 目标区域寄存器

寄存器	地址	R/W	描述	复位值
CICOTAREA	0x4F00005C	R/W	编码缩放器目标区域	0

CICOTAREA	位	描述	初始状态
CICOTAREA	[25:0]	编码 DMA 的目标区域 = 目标 H 尺寸×目标 V 尺寸	0

编码状态寄存器

寄存器	地址	R/W	描述	复位值
CICOSTATUS	0x4F000064	R	编码通路状态	0

CICOSTATUS	位	描述	初始状态
OvFiY_Co	[31]	编码源 FIFO Y 的溢出状态	0
OvFiCb_Co	[30]	编码源 FIFO Cb 的溢出状态	0
OvFiCr_Co	[29]	编码源 FIFO Cr 的溢出状态	0
VSYNC	[28]	摄像头 VSYNC(此位可以参考 CPU 第一次 SFR 设置。并且其在 ITU-R BT 656 模式中也可见)	0
FrameCnt_Co	[27:26]	编码 DMA 的帧计数 (此计数器值包括了下帧数)	0
WinOfstEn_Co	[25]	窗口偏移使能状态	0
FlipMd_Co	[24:23]	编码 DMA 的翻转模式	0
ImgCptEn_CamIf	[22]	摄像头接口的图像捕获使能	0
ImgCptEn_CoSC	[21]	编码通路的图像捕获使能	0

RGB1 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CIPRCLRSA1	0x4F00006C	R/W	编码 DMA 的 RGB 第一帧开始地址	0

CIPRCLRSA1	位	描述	初始状态
CIPRCLRSA1	[31:0]	编码 DMA 的 RGB 第一帧开始地址	0

RGB2 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CIPRCLRSA2	0x4F000070	R/W	编码 DMA 的 RGB 第二帧开始地址	0

CIPRCLRSA2	位	描述	初始状态
CIPRCLRSA2	[31:0]	编码 DMA 的 RGB 第二帧开始地址	0

RGB3 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CIPRCLRSA3	0x4F000074	R/W	编码 DMA 的 RGB 第三帧开始地址	0

CIPRCLRSA3	位	描述	初始状态
CIPRCLRSA3	[31:0]	编码 DMA 的 RGB 第三帧开始地址	0

RGB4 开始地址寄存器

寄存器	地址	R/W	描述	复位值
CIPRCLRSA4	0x4F000078	R/W	编码 DMA 的 RGB 第四帧开始地址	0

CIPRCLRSA4	位	描述	初始状态
CIPRCLRSA4	[31:0]	编码 DMA 的 RGB 第四帧开始地址	0

预览目标格式寄存器

寄存器	地址	R/W	描述	复位值
CIPRTRGFMT	0x4F00007C	R/W	预览 DMA 的目标图像格式	0

CIPRTRGFMT	位	描述	初始状态
TargetHsize_Pr	[28:16]	预览 DMA 的目标图像的水平像素数 (偶数)	0
FlipMd_Pr	[15:14]	预览 DMA 的图像镜像和旋转 00 = 正常 10 = Y 轴镜像 11 = 180° 旋转	0
TargetVsize_Pr	[12:0]	预览 DMA 的目标图像的垂直像素数	0

预览 DMA 控制寄存器

寄存器	地址	R/W	描述	复位值
CIPRCTRL	0x4F000080	R/W	预览 DMA 控制相关	0

CIPRCTRL	位	描述	初始状态
RGBburst1_Pr	[23:19]	预览 RGB 帧的主突发长度	0
RGBburst2_Pr	[18:14]	预览 RGB 帧的剩余突发长度	0
LastIRQEn_Pr	[2]	0 = 正常 1 = 帧捕获结束时使能最后 IRQ (此位自动清零)	0

注释：

所有突发长度必须为 2、4、8 或 16 中的一个。

例 1：目标图像尺寸：RGB 32 位格式的 QCIF (水平宽度 = 176 像素。1 像素 = 1 字节)

$$176 \text{ 像素} = 176 \text{ 字}$$

$$176 \% 16 = 0 \rightarrow \text{主突发} = 16, \text{剩余突发} = 16$$

例 2：目标图像尺寸：RGB 16 位格式的 VGA (水平宽度 = 640 像素。2 像素 = 1 字节)

$$640 / 2 = 320 \text{ 字}$$

$$160 \% 16 = 0 \rightarrow \text{主突发} = 16, \text{剩余突发} = 16$$

预览预缩放控制寄存器 1

寄存器	地址	R/W	描述	复位值
CIPRSCPRERATIO	0x4F000084	R/W	预览预缩放率控制	0

CIPRSCPRERATIO	位	描述	初始状态
SHfactor_Pr	[31:28]	预览预缩放的变化系数	0
PreHorRatio_Pr	[22:16]	预览预缩放的水平比	0
PreVerRatio_Pr	[6:0]	预览预缩放的垂直比	0

预览预缩放控制寄存器 2

寄存器	地址	R/W	描述	复位值
CIPRSCPREDST	0x4F000088	R/W	预览预缩放目标格式	0

CIPRSCPREDST	位	描述	初始状态
PreDstWidth_Pr	[27:16]	预览预缩放的目标宽度	0
PreDstHeight_Pr	[11:0]	预览预缩放的目标高度	0

预览主缩放控制寄存器

寄存器	地址	R/W	描述	复位值
CIPRSCCTRL	0x4F00008C	R/W	预览主缩放控制	0

CIPRSCCTRL	位	描述	初始状态
Sample_Pr	[31]	格式转换的采样方式 (此位推荐固定为 1)	0
RGBformat_Pr	[30]	0 = 16 位 RGB 0 = 24 位 RGB	0
ScaleUpDown_Pr	[29:28]	预览缩放器的放大/缩小标志 (1:1 缩放比中 , 此位应该为 "1") 00 = 缩小 11 = 放大	00
MainHorRatio_Pr	[24:16]	预览主缩放的水平缩放比	0
PrScalerStart	[15]	预览缩放器开始	0
MainVerRatio_Pr	[8:0]	预览主缩放的垂直缩放比	0

预览 DMA 目标区域寄存器

寄存器	地址	R/W	描述	复位值
CIPRTAREA	0x4F000090	R/W	预览缩放器目标区域	0

CIPRTAREA	位	描述	初始状态
CIPRTAREA	[25:0]	预览 DMA 的目标区域 = 目标 H 尺寸×目标 V 尺寸	0

预览状态寄存器

寄存器	地址	R/W	描述	复位值
CIPRSTATUS	0x4F000098	R	预览通路状态	0

CIPRSTATUS	位	描述	初始状态
OvFiCb_Pr	[31]	预览源 FIFO Cb 的溢出状态	0
OvFiCr_Pr	[30]	预览源 FIFO Cr 的溢出状态	0
FrameCnt_Pr	[27:26]	预览 DMA 的帧计数	0
FlipMd_Pr	[24:23]	预览 DMA 的翻转模式	0
ImgCptEn_PrSC	[21]	预览通路的图像捕获使能	0

图像捕获使能寄存器

此寄存器必须在最后设置

寄存器	地址	R/W	描述	复位值
CIIMGCPT	0x4F0000A0	R/W	图像捕获使能命令	0

CIGCTRL	位	描述	初始状态
ImgCptEn	[31]	摄像头接口全局捕获使能	0
ImgCptEn_CoSc	[30]	编码缩放器的捕获使能。缩放旁路模式中此位必须为'0'	0
ImgCptEn_PrSc	[29]	预览缩放器的捕获使能。缩放旁路模式中此位必须为'0'	0

24 AC'97 控制器

概述

S3C2440A 的 AC'97 控制器单元支持 AC'97 2.0 修订版特性。AC'97 控制器使用音频控制器链接 (AC-Link) 与 AC'97 编解码器相连接。控制器发送立体声 PCM 数据到编解码器。编解码器中的外部数/模转换器 (DAC) 接着转换音频采样到模拟音频波形。同样的，控制器从编解码器接收立体声 PCM 数据和单声道 MIC 数据并且接着储存它们到存储器中。本章描述 AC'97 控制器单元的编程模型。本章节的信息需要对 AC'97 2.0 修订版规范有所了解。

注意：

一定不要同时使用 AC'97 控制器和 I²S。

特性

- 立体声 PCM 输入、立体声 PCM 输出和单声道 MIC 输入的独立通道。
- 基于 DMA 操作和基于中断操作。
- 所有通道只支持 16 位采样。
- 可变采样率 AC'97 编解码器接口 (48 KHz 及以下)。
- 16 位、16 个人口 FIFO 每通道。
- 只支持主编解码器

AC'97 控制器操作

方框图

图 24-1 显示了 S3C2440A 的 AC'97 控制器的功能方框图。来自 AC-link 的 AC'97 信号是一个点对点的同步串行互联，它支持全双工数据传输。所有数字音频流和命令/状态信息通过 AC-link 通信。

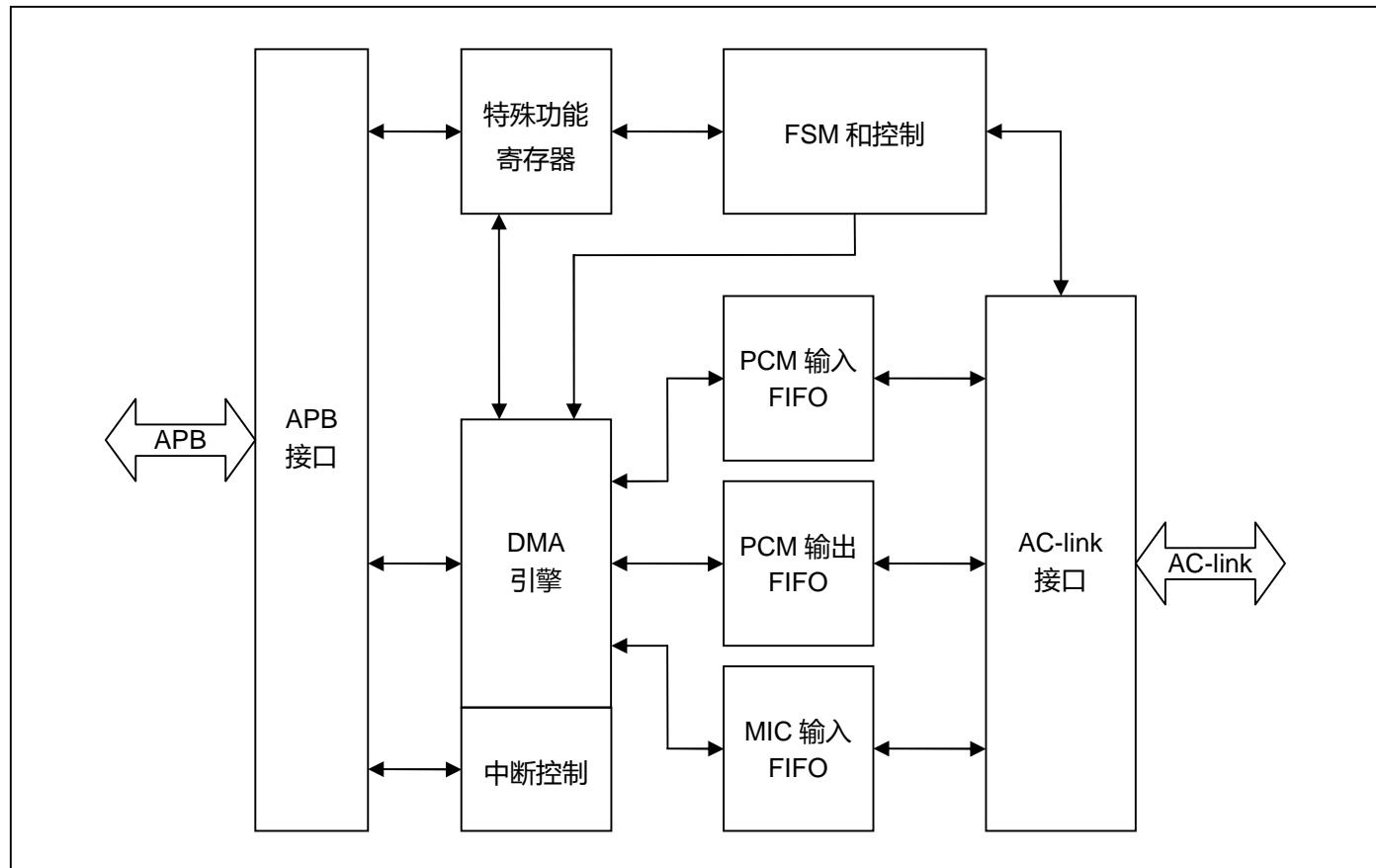


图 24-1. AC'97 方框图

内部数据路径

图 24-2 显示了 S3C2440A 的 AC'97 控制器的内部数据路径。它包含由 16 位 16 入口缓冲器组成的立体声脉冲编码调制 (PCM) 输入、立体声 PCM 输出和单声道 MIC 输入缓冲器。同样的它还有一个经过 AC-link 的 20 位 I/O 移位寄存器。

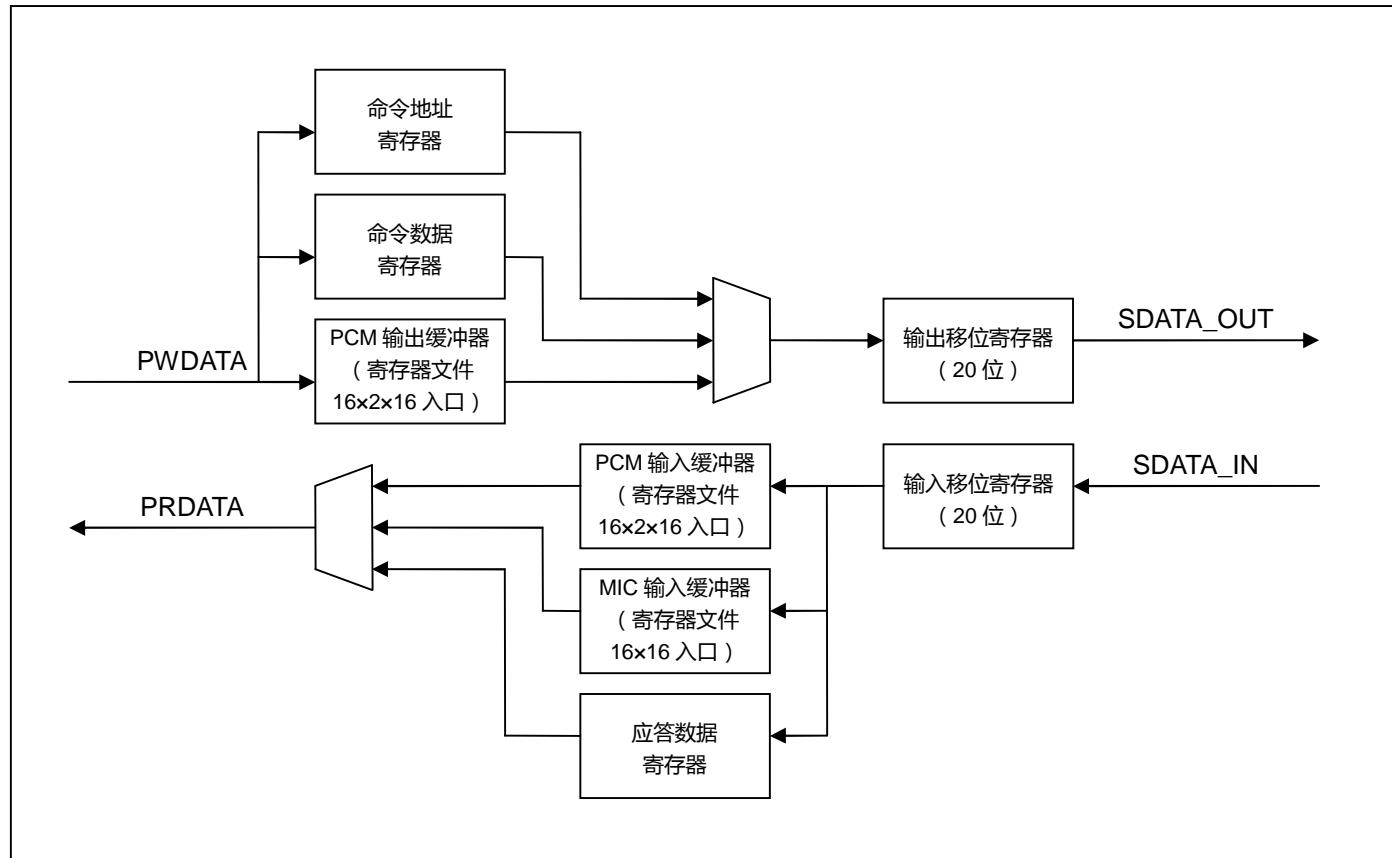


图 24-2. 内部数据路径

AC'97 操作流程图

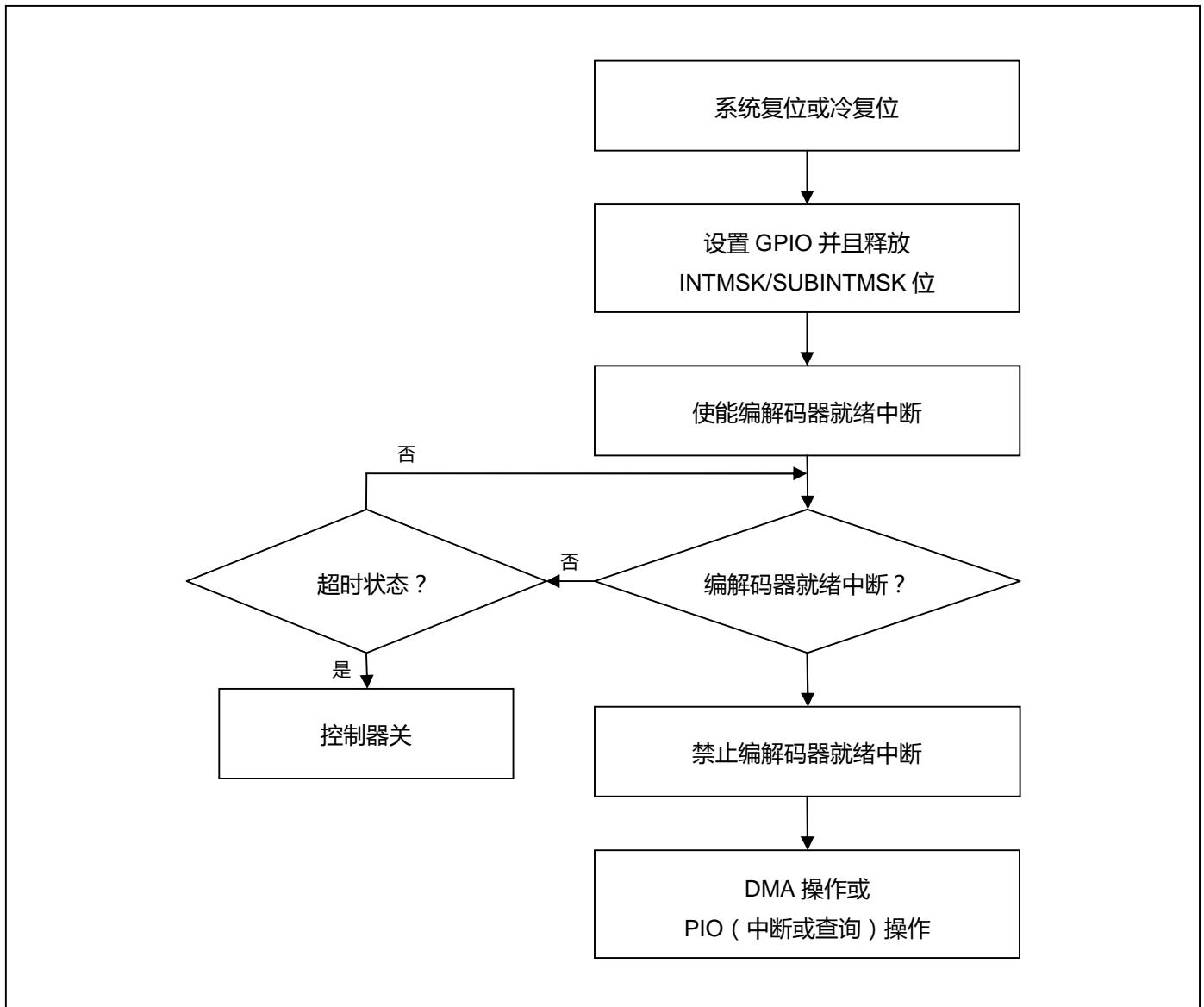


图 24-3. AC'97 操作流程图

AC-LINK 数字接口协议

每个 AC'97 编解码器都包含一个 5 个引脚的数字串行接口，链接它到 S3C2440A 的 AC'97 控制器。AC-link 为全双工、固定时钟、PCM 数字流。其利用一个时分复用 (TDM) 方案来处理控制寄存器访问和多输入输出音频流。AC-link 结构划分每个音频帧到 12 个出去的和 12 个进来的音频流。每个流有 20 位采样分辨率并且需要最低 16 位分辨率的一个 DAC 和一个 ADC。

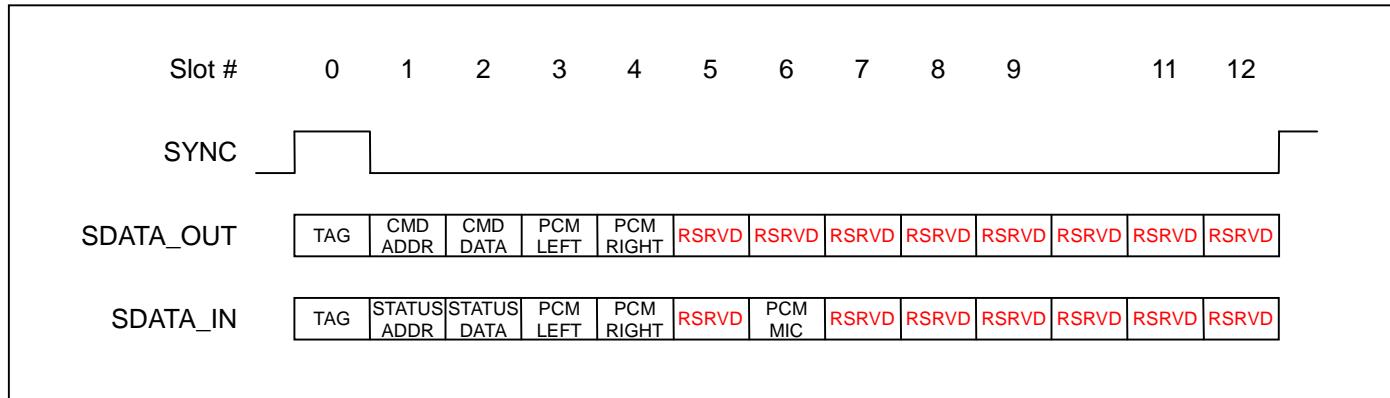


图 24-4. 时隙分配的双向 AC-link 帧

图 24-4 显示了 S3C2440A 的 AC'97 控制器支持的时隙分配。S3C2440A 的 AC'97 控制器为 AC-link 上所有数据传输提供同步。

一次数据传输是由 256 位组成，这些信息被分到 13 组时隙并且称之为 1 帧。时隙 0 被称为标记期并且它为 6 位长度。剩余的 12 次时隙被称为数据期。标记期包含了识别一个有效帧的位，并且数据期中的 12 位识别的时隙包含一个有效的数据。数据期的每个时隙为 20 位长度。当 SYNC 变为高时帧开始。SYNC 为高的时间量等于标记期。

AC'97 帧发生固定为 48 kHz 的时间间隔并且同步于 12.288MHz 的比特率时钟 BITCLK。控制器和编解码器使用 SYNC 和 BITCLK 来决定什么时候要送出发送数据和什么时候要采样接收数据。发送器在每个 BITCLK 的上升沿转移串行数据流，接收器在每个 BITCLK 的下降沿采样串行数据流。发送器必须在其串行数据流中标记有效时隙。有效时隙被标记在时隙 0 中。AC-link 上的串行数据是从 MSB 到 LSB 的。标记期的第一位为位[15]并且数据期中的每个时隙的第一位为位[19]。任何时隙的最后一位都为位[0]。

AC-LINK 输出帧 (SDATA_OUT)

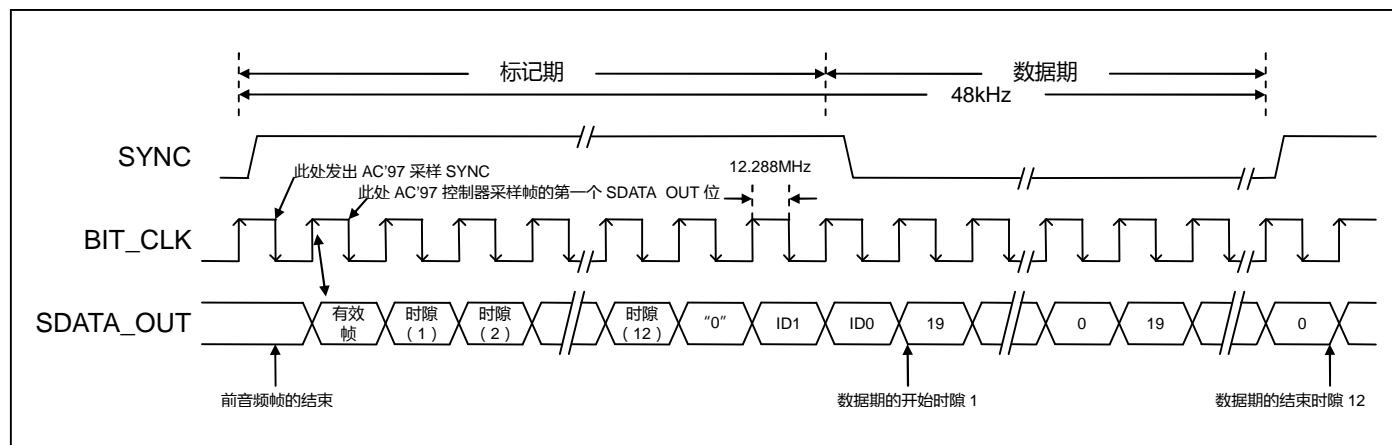


图 24-5. AC-link 输出帧

AC-LINK 输入帧 (SDATA_IN)

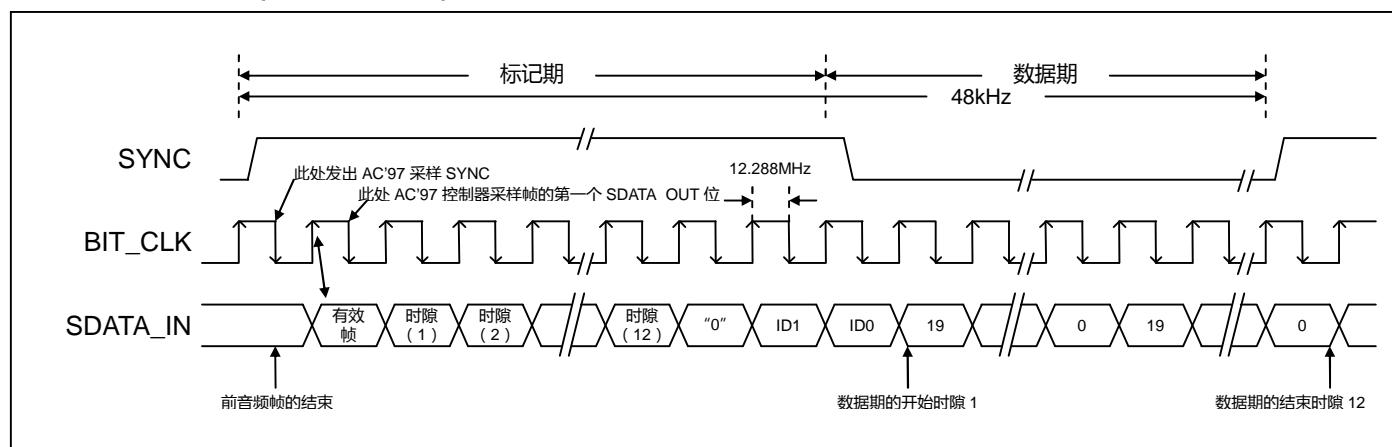


图 24-5. AC-link 输入帧

AC'97 掉电

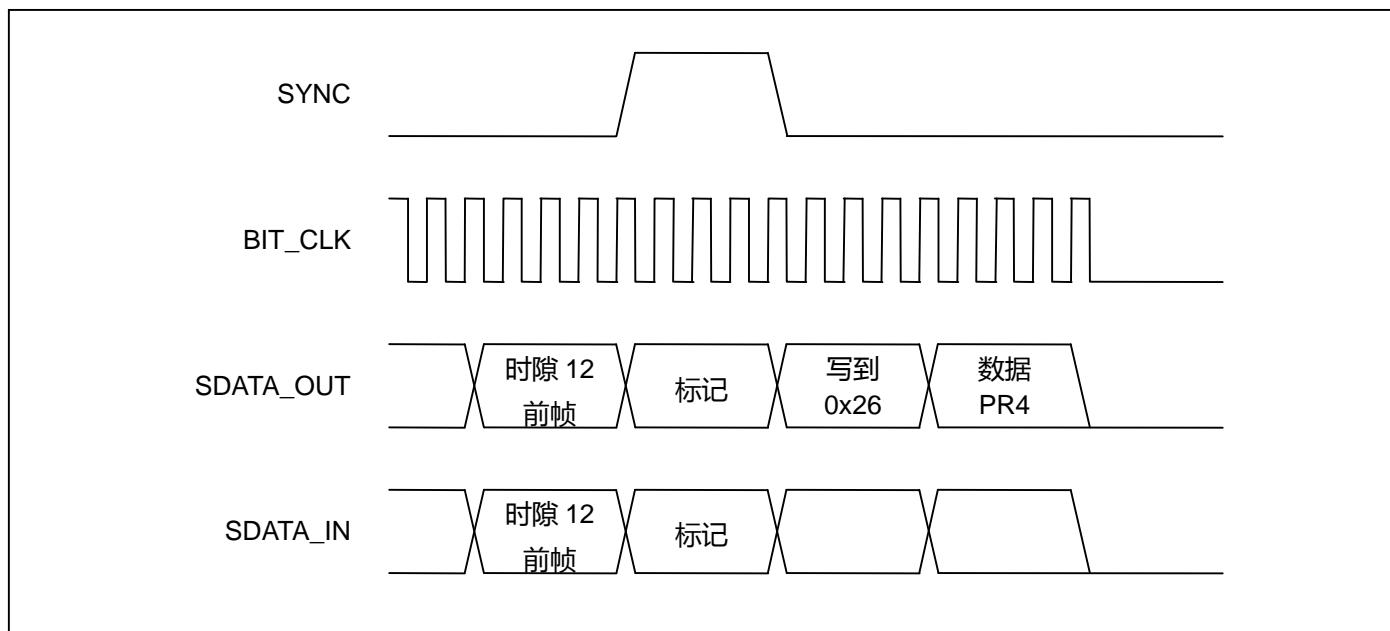


图 24-7. AC'97 掉电时序图

掉电 AC-link

当 AC'97 编解码器掉电寄存器 (0x26) 位 PR4 被设置为 1 时 (如通过写 0x1000) 则 AC-link 信号进入一个低功耗模式。然后主编解码器同时驱动 BITCLK 和 SDATA_IN 到逻辑低电平。图 27-7 中显示了时序图的顺序。

AC'97 控制器通过 AC-link 发送写掉电寄存器(0x26)。建立 AC'97 控制器以便当其写到掉电寄存器位 PR4(数据 0x1000) 时不能发送数据到时隙 3 至 12 , 并且当收到掉电请求时不需要编解码器来处理其它数据。当编解码器处理请求时 , 它立即转变 BITCLK 和 SDATA_IN 到逻辑低电平。AC'97 控制器在编程 AC_GLBCTRL 寄存器之后驱动 SYNC 和 SDATA_OUT 到逻辑低电平。

唤醒 AC-link——由 AC'97 控制器触发唤醒

AC-link 协议提供了一个冷 AC'97 复位和一个热 AC'97 复位。当前掉电状态最终决定了使用哪种 AC'97 复位。寄存器必须在所有掉电模式期间保持相同的状态，除非执行了冷 AC'97 复位。冷 AC'97 复位中，初始化 AC'97 寄存器到它们的默认值。掉电后，AC-link 必须在帧后等待至少 4 个音频帧时间，其中掉电发生在通过复位 SYNC 信号它可以被恢复活动之前。当 AC-link 上电时，通过编码就绪位（输入时隙 0，位[15]）指示就绪。

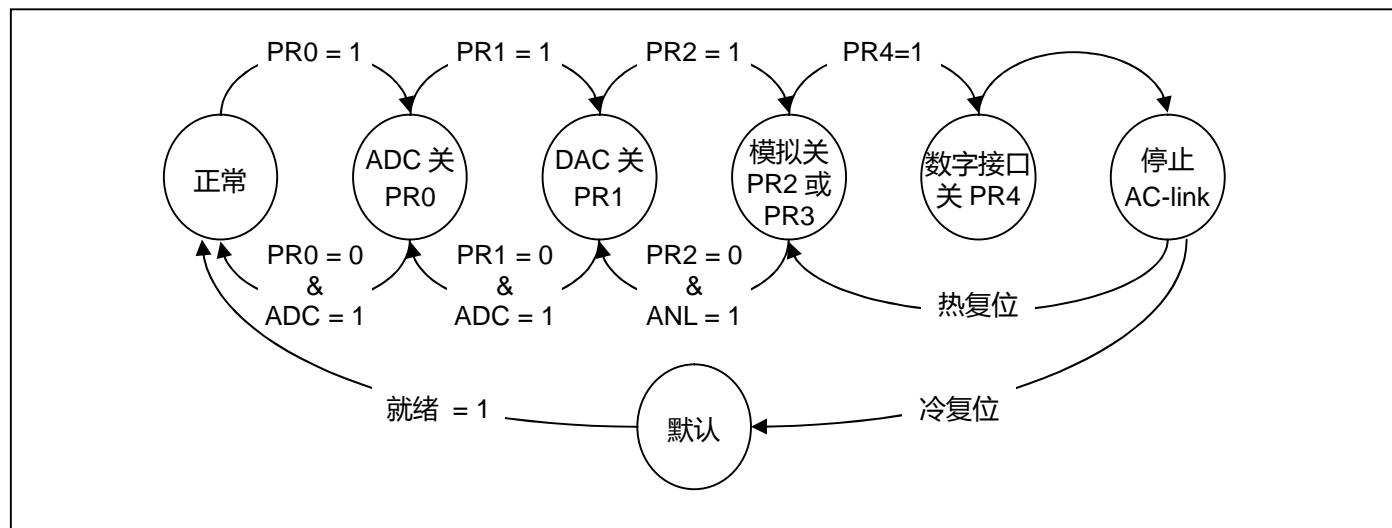


图 24-8. AC97 掉电/上电循环

冷 AC'97 复位

当通过 AC_GLBCTRL 生效了 nRESET 引脚则产生一个冷复位。生效和无效 nRESET 将触发 BITCLK 和 SDATA_OUT。初始化所有的 AC'97 控制寄存器到它们默认上电复位值

热 AC'97 复位

热 AC'97 复位恢复 AC-link 活动，无须变更当前 AC'97 寄存器值。当无 BITCLK 并且 SYNC 被驱动为高时产生一个热复位。正常音频帧中，SYNC 同步于 AC'97 输入。当无 BITCLK 时，SYNC 已经被处理为一个同步输入来用做产生一个热复位给 AC'97。AC'97 控制器在其再次采样到 SYNC 为低之前不能激活 BITCLK。这样避免错误察觉新音频帧。

AC'97 控制器特殊寄存器

AC'97 全局控制寄存器 (AC_GLBCTRL)

寄存器	地址	R/W	描述	复位值
AC_GLBCTRL	0x5B000000	R/W	AC'97 全局控制寄存器	0x000000

AC_GLBCTRL	位	描述		初始状态
保留	[31:23]	-		0x00
编解码器就绪 中断使能	[22]	0 : 禁止 1 : 使能		0
PCM 输出通道 欠载中断使能	[21]	0 : 禁止 1 : 使能 (FIFO 为空)		0
PCM 输入通道 过载中断使能	[20]	0 : 禁止 1 : 使能 (FIFO 为满)		0
MIC 输入通道 过载中断使能	[19]	0 : 禁止 1 : 使能 (FIFO 为满)		0
PCM 输出通道 阀值中断使能	[18]	0 : 禁止 1 : 使能 (FIFO 为半空)		0
PCM 输入通道 阀值中断使能	[17]	0 : 禁止 1 : 使能 (FIFO 为半满)		0
MIC 输入通道 阀值中断使能	[16]	0 : 禁止 1 : 使能 (FIFO 为半满)		0
保留	[15:14]	-		00
PCM 输出通道 传输模式	[13:12]	00 : 关 10 : DMA	01 : PIO 11 : 保留	00
PCM 输入通道 传输模式	[11:10]	00 : 关 10 : DMA	01 : PIO 11 : 保留	00
MIC 输入通道 传输模式	[9:8]	00 : 关 10 : DMA	01 : PIO 11 : 保留	00
保留	[7:4]	-		0000
使用 AC-link 传输数据使能	[3]	0 : 禁止 1 : 使能		0
AC-link 开	[2]	0 : 关 1 : SYNC 信号发送到编解码器		0
热复位	[1]	0 : 正常 1 : 从掉电唤醒编解码器		0
冷复位	[0]	0 : 正常 1 : 复位编解码器和控制器逻辑		0

AC'97 全局状态寄存器 (AC_GLBSTAT)

寄存器	地址	R/W	描述	复位值
AC_GLBSTAT	0x5B000004	R	AC'97 全局状态寄存器	0x0000000

AC_GLBSTAT	位	描述	初始状态
保留	[31:23]	-	0x00
编解码器就绪中断	[22]	0 : 无请求 1 : 请求	0
PCM 输出通道欠载中断	[21]	0 : 无请求 1 : 请求	0
PCM 输入通道过载中断	[20]	0 : 无请求 1 : 请求	0
MIC 输入通道过载中断	[19]	0 : 无请求 1 : 请求	0
PCM 输出通道阀值中断	[18]	0 : 无请求 1 : 请求	0
PCM 输入通道阀值中断	[17]	0 : 无请求 1 : 请求	0
MIC 输入通道阀值中断	[16]	0 : 无请求 1 : 请求	0
保留	[15:3]	-	0x000
控制器主状态	[2:0]	000 : 空闲 010 : 就绪 100 : LP 001 : 初始化 011 : 激活 101 : 热	000

AC'97 编解码器命令寄存器 (AC_CODEC_CMD)

寄存器	地址	R/W	描述	复位值
AC_CODEC_CMD	0x5B000008	R/W	AC'97 编解码器命令寄存器	0x00000000

AC_CODEC_CMD	位	描述	初始状态
保留	[31:24]	-	0x00
读使能	[23]	0 : 命令写 (注意) 1 : 状态读	0
地址	[22:16]	编解码器命令地址	0x00
数据	[15:0]	编解码器命令数据	0x0000

注意：

当写入命令到 AC_CODEC_CMD 寄存器中时，推荐在命令和下个命令之间多于延时 1/48 Hz 的延时。

AC'97 编解码器状态寄存器 (AC_CODEC_CMD)

寄存器	地址	R/W	描述	复位值
AC_CODEC_STAT	0x5B00000C	R	AC'97 编解码器状态寄存器	0x00000000

AC_CODEC_STAT	位	描述	初始状态
保留	[31:23]	-	0x00
地址	[22:16]	编解码器状态地址	
数据	[15:0]	编解码器状态数据	

注意：

如果希望经过 AC_CODEC_STAT 寄存器从 AC'97 编解码器寄存器读取数据，那么应该按以下步骤。

1. 写命令地址和数据到 Bit [23] =1 的 AC_CODEC_CMD 寄存器。
2. 延迟一段时间。
3. 从 AC_CODEC_STAT 寄存器中读取命令地址和数据。

AC'97 PCM 输入/输出通道 FIFO 地址寄存器 (AC_PCMADDR)

寄存器	地址	R/W	描述	复位值
AC_PCMADDR	0x5B000010	R	AC'97 PCM 输入/输出通道 FIFO 地址寄存器	0x00000000

AC_PCMADDR	位	描述	初始状态
保留	[31:28]	-	0000
输出读地址	[27:24]	PCM 输出通道 FIFO 读地址	0000
保留	[23:20]	-	0000
输入读地址	[19:16]	PCM 输入通道 FIFO 读地址	0000
保留	[15:12]	-	0000
输出写地址	[11:8]	PCM 输出通道 FIFO 写地址	0000
保留	[7:4]	-	0000
输入写地址	[3:0]	PCM 输入通道 FIFO 写地址	0000

AC'97 MIC 输入通道 FIFO 地址寄存器 (AC_MICADDR)

寄存器	地址	R/W	描述	复位值
AC_MICADDR	0x5B000014	R	AC'97 MIC 输入通道 FIFO 地址寄存器	0x00000000

AC_MICADDR	位	描述	初始状态
保留	[31:20]	-	0000
读地址	[19:16]	MIC 输入通道 FIFO 读地址	0000
保留	[15:4]	-	0x000
写地址	[3:0]	MIC 输入通道 FIFO 写地址	0000

AC'97 PCM 输入/输出通道 FIFO 数据寄存器 (AC_PCMDATA)

寄存器	地址	R/W	描述	复位值
AC_PCMDATA	0x5B000018	R/W	AC'97 PCM 输入/输出通道 FIFO 数据寄存器	0x00000000

AC_MICADDR	位	描述	初始状态
左数据	[31:20]	PCM 输入输出左通道 FIFO 数据 读 : PCM 输入左通道 写 : PCM 输出左通道	0x0000
右数据	[15:0]	PCM 输入输出右通道 FIFO 数据 读 : PCM 输入右通道 写 : PCM 输出右通道	0x0000

AC'97 MIC 输入通道 FIFO 数据寄存器 (AC_MICDATA)

寄存器	地址	R/W	描述	复位值
AC_MICDATA	0x5B00001C	R/W	AC'97 MIC 输入通道 FIFO 数据寄存器	0x00000000

AC_MICDATA	位	描述	初始状态
保留	[31:16]	-	0x0000
立体声数据	[15:0]	MIC 输入立体声通道 FIFO 数据	0x0000

25 总线优先级

概述

总线仲裁逻辑确定了总线主控的优先级。它支持循环优先级模式及固定优先级模式的组合。

总线优先级映射

S3C2440A 包含了 13 个总线主控。包括 DRAM 刷新控制器 ,LCD_DMA ,CAMIF DMA ,DMA0 ,DMA1 ,DMA2 ,DMA3 ,USB_HOST_DMA ,EXT_BUS_MASTER ,测试接口控制器 (TIC) 及 ARM920T。以下列表显示了其中复位后这些总线主控的优先级：

1. DRAM 刷新控制器
2. LCD_DMA
3. CAMIF 编码 DMA
4. CAMIF 预览 DMA
5. DMA0
6. DMA1
7. DMA2
8. DMA3
9. USB 主机 DMA
10. 外部总线主控
11. TIC
12. ARM920T
13. 保留

这些总线主控其中，4 个 DMA (DMA0 , DMA2 , DMA2 及 DMA3) 在循环优先级下运行，其他的在固定优先级下运行。

注 释

26 机械数据

封装尺寸

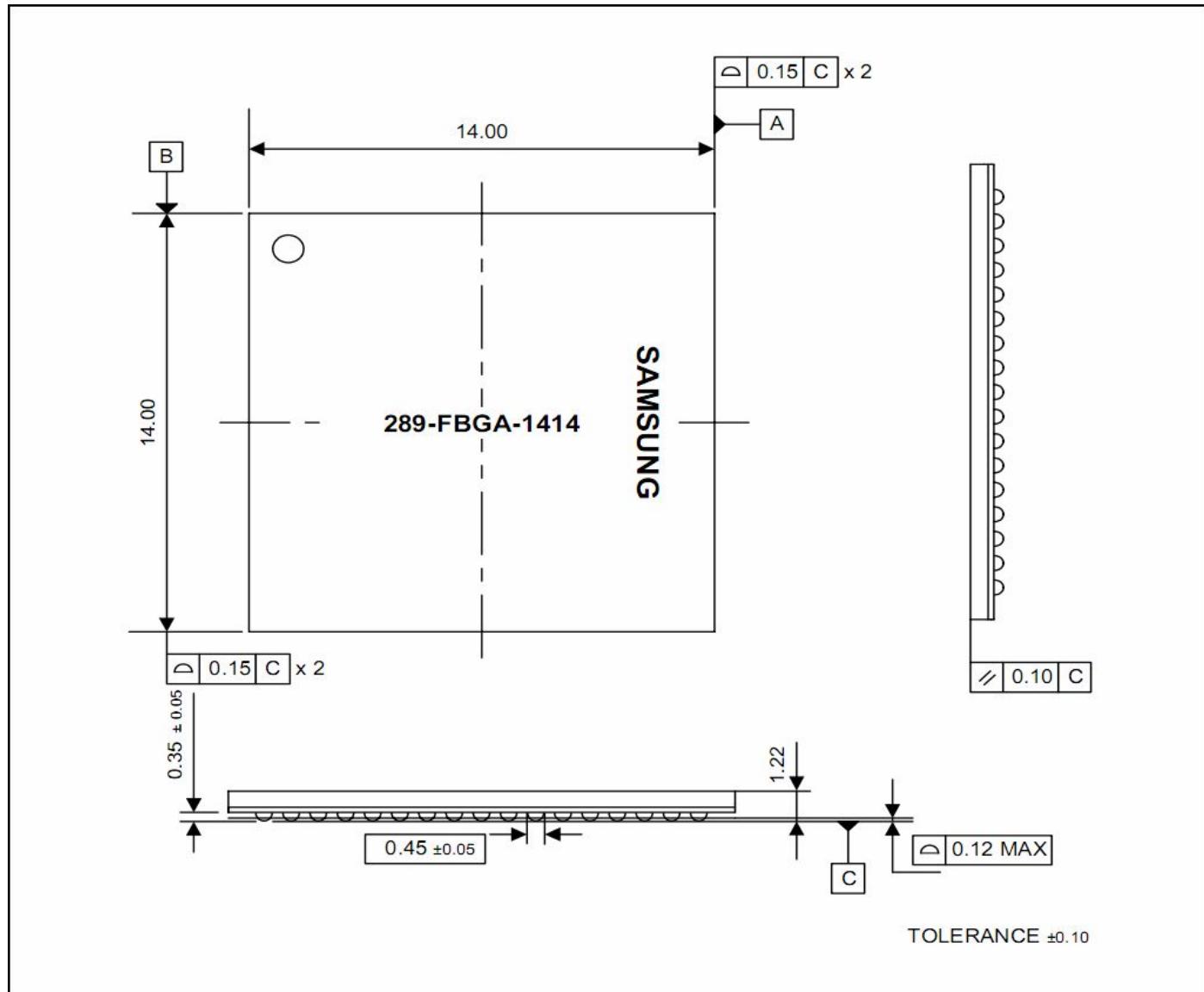


图 26-1 289-FBGA-1414 封装尺寸 1 (俯视)

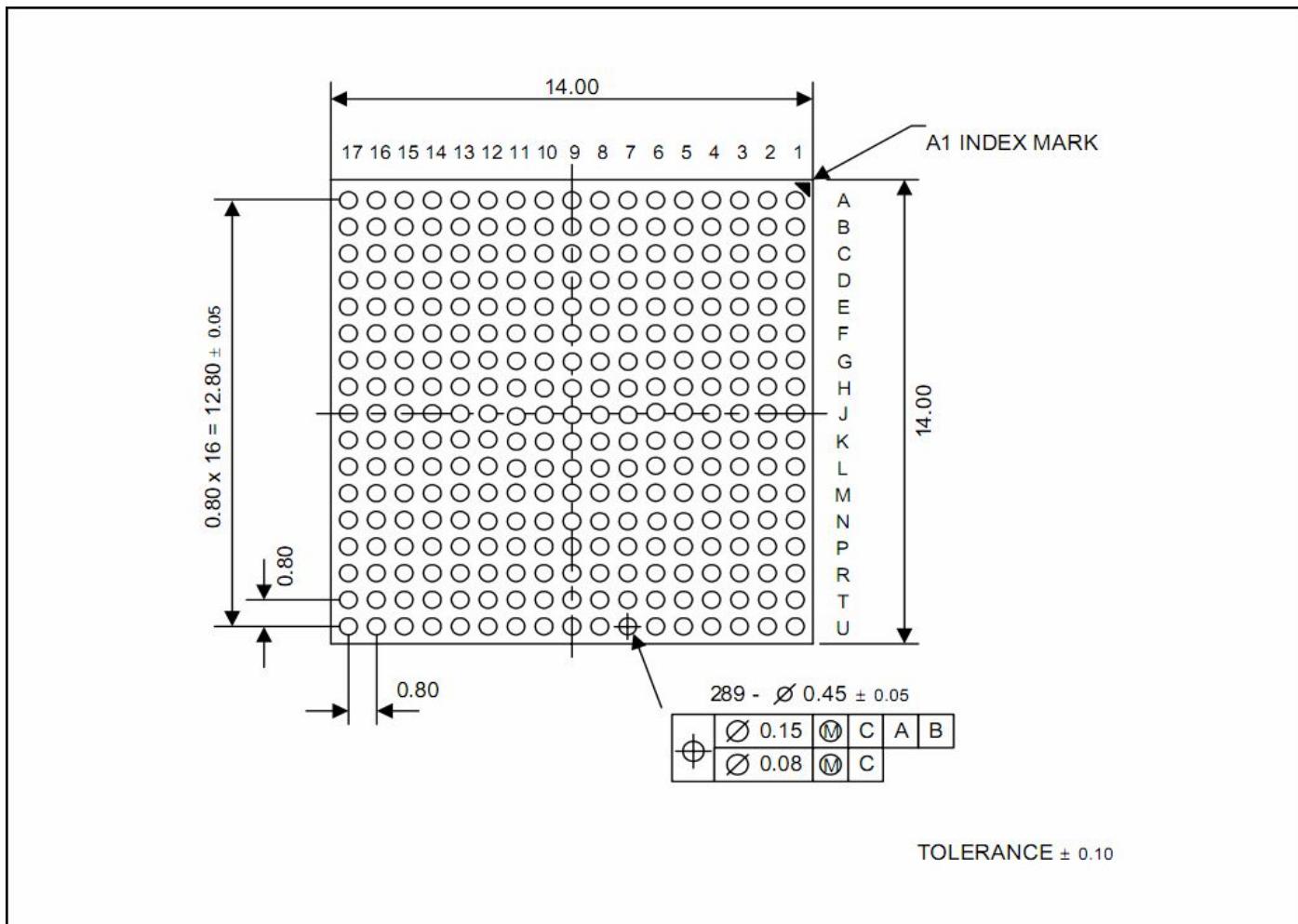


图 26-2 289-FBGA-1414 封装尺寸 2 (仰视)

推荐焊盘大小为 0.39 至 0.41mm 直径。

27 电气数据

绝对最大额定值

表 27-1. 绝对最大额定值

参数	符号	额定值	单位
直流供电电压	V_{DD_i}	1.2V V_{DD}	V
	V_{DDOP}	3.3V V_{DD}	
	V_{DDMOP}	1.8V/2.5V/3.0V/3.3V V_{DD}	
	V_{DDRTC}	1.8V/2.5V/3.0V/3.3V V_{DD}	
	V_{DDADC}	3.3V V_{DD}	
直流输入电压	V_{IN}	3.3V 输入缓冲	4.8
		3.3V 接口 / 5V 容忍输入缓冲	6.5
直流输出电压	V_{OUT}	3.3V 输出缓冲	4.8
直流输入 (锁存) 电流	I_{IN}	±200	mA
贮藏温度	T_{STG}	-65 至 150	°C

推荐工作条件

表 27-2. 推荐工作条件

参数	符号	额定值			单位
		典型值	最小值	最大值	
有效模块的直流供电电压	$V_{DDalive}$	300MHz : 1.2V V_{DD} 400MHz : 1.3V V_{DD}	1.15	1.25	V
内部的直流供电电压	$V_{DDi}^{(1)}$ $V_{DDiarm}^{(1)}$ $V_{DDMPPLL}$ $V_{DDUPPLL}$	133MHz : 1.1V V_{DD} 300MHz : 1.2V V_{DD} 400MHz : 1.3V V_{DD}	1.05 1.15	1.15 1.25	
I/O 模块的直流供电电压	V_{DDOP}	3.3V V_{DD}	3.0	3.6	
存储器接口的直流供电电压	V_{DDMOP}	1.8V/2.5V/3.0V/3.3V V_{DD}	1.7	3.6	
模拟内核的直流供电电压	V_{DD}	3.3V V_{DD}	3.0	3.6	
RTC 的直流供电电压	V_{DDRTC}	1.8V/2.5V/3.0V/3.3V V_{DD}	1.8	3.6	
直流输入电压	V_{IN}	3.3V 输入缓冲 3.3V 接口/5V 容忍输入缓冲	-0.3 -0.3	$V_{DDOP}+0.3$ 5.25	
直流输出电压	V_{OUT}	3.3V 输出缓冲	-0.3	$V_{DDOP}+0.3$	
工作温度	T_{OPR}	工业	-40 至 85		°C
		长期	-20 至 70		

注释：

DVS (动态电压调整) 中 V_{DDi} 和 V_{DDiarm} 可以在空闲模式中被供应为 1.0V。请参考应用手册以获取详细信息。

直流电气特性

表 27-3 和 27-4 定义了标准 LVCMS I/O 缓冲的直流电气特性。

表 27-3. 普通 I/O 引脚直流电气特性

存储器的普通 I/O 引脚直流电气特性 ($V_{DDMOP} = 2.5V \pm 0.2V$, $T_A = -40$ 至 85°C)

符号	参数	条件	最小值	典型值	最大值	单位
V_{IH}	高电平输入电压					V
	LVCMS 接口		1.7			
V_{IL}	低电平输入电压				0.7	V
	LVCMS 接口					
VT	开关门限			$0.5V_{DD}$		V
$VT+$	施密特触发器, 上升沿门限	CMOS			2.0	V
$VT-$	施密特触发器, 下降沿门限	CMOS	0.8			V
I_{IH}	高电平输入电流					μA
	输入缓冲	$V_{IN} = V_{DD}$	-10		10	
I_{IL}	低电平输入电流					μA
	输入缓冲	$V_{IN} = V_{SS}$	-10		10	
	带上拉的输入缓冲		-60	-33	-10	
V_{OH}	高电平输出电压					V
	类型 B4 至 B12	$I_{OH} = -1 \mu\text{A}$	$V_{DD}-0.05$			
	类型 B4	$I_{OH} = -4 \text{ mA}$	2.0			
	类型 B6	$I_{OH} = -6 \text{ mA}$				
	类型 B8	$I_{OH} = -8 \text{ mA}$				
	类型 B10	$I_{OH} = -10 \text{ mA}$				
	类型 B12	$I_{OH} = -12 \text{ mA}$				
V_{OL}	低电平输出电压					V
	类型 B4 至 B12	$I_{OL} = 1 \mu\text{A}$			0.05	
	类型 B4	$I_{OL} = 4 \text{ mA}$			0.4	
	类型 B6	$I_{OL} = 6 \text{ mA}$				
	类型 B8	$I_{OL} = 8 \text{ mA}$				
	类型 B10	$I_{OL} = 10 \text{ mA}$				
	类型 B12	$I_{OL} = 12 \text{ mA}$				

注释:

1. 类型 B6 意思为 6mA 输出驱动器单元
2. 类型 B8 意思为 8mA 输出驱动器单元

存储器的普通 I/O 引脚直流电气特性 ($V_{DDMOP} = 3.0V \pm 0.3V$, $T_A = -40$ 至 85°C)

符号	参数	条件	最小值	典型值	最大值	单位
V_{IH}	高电平输入电压 LVC MOS 接口		2.0			V
V_{IL}	低电平输入电压 LVC MOS 接口				0.8	V
VT	开关门限			0.5V _{DD}		V
VT+	施密特触发器, 上升沿门限	CMOS			2.0	V
VT-	施密特触发器, 下降沿门限	CMOS	0.8			V
I_{IH}	高电平输入电流 输入缓冲	$V_{IN} = V_{DD}$	-10		10	μA
I_{IL}	低电平输入电流 输入缓冲	$V_{IN} = V_{SS}$	-10		10	μA
	带上拉的输入缓冲		-60	-33	-10	
V_{OH}	高电平输出电压 类型 B4 至 B12	$I_{OH} = -1 \mu\text{A}$	$V_{DD} - 0.05$			V
	类型 B4	$I_{OH} = -4 \text{ mA}$	2.4			
	类型 B6	$I_{OH} = -6 \text{ mA}$				
	类型 B8	$I_{OH} = -8 \text{ mA}$				
	类型 B10	$I_{OH} = -10 \text{ mA}$				
	类型 B12	$I_{OH} = -12 \text{ mA}$				
V_{OL}	低电平输出电压 类型 B4 至 B12	$I_{OL} = 1 \mu\text{A}$			0.05	V
	类型 B4	$I_{OL} = 4 \text{ mA}$			0.4	
	类型 B6	$I_{OL} = 6 \text{ mA}$				
	类型 B8	$I_{OL} = 8 \text{ mA}$				
	类型 B10	$I_{OL} = 10 \text{ mA}$				
	类型 B12	$I_{OL} = 12 \text{ mA}$				

注释：

1. 类型 B6 意思为 6mA 输出驱动器单元
2. 类型 B8 意思为 8mA 输出驱动器单元
3. 类型 B12 意思为 12mA 输出驱动器单元

I/O 的普通 I/O 引脚直流电气特性 ($V_{DDMOP} = 3.0V \pm 0.3V$, $T_A = -40$ 至 85°C)

符号	参数	条件	最小值	典型值	最大值	单位
V_{IH}	高电平输入电压 LVC MOS 接口		2.0			V
V_{IL}	低电平输入电压 LVC MOS 接口				0.8	V
VT	开关门限			0.5V _{DD}		V
VT+	施密特触发器, 上升沿门限	CMOS			2.0	V
VT-	施密特触发器, 下降沿门限	CMOS	0.8			V
I_{IH}	高电平输入电流 输入缓冲	$V_{IN} = V_{DD}$	-10		10	μA
I_{IL}	低电平输入电流 输入缓冲	$V_{IN} = V_{SS}$	-10		10	μA
	带上拉的输入缓冲		-60	-33	-10	
V_{OH}	高电平输出电压 类型 B4 至 B12	$I_{OH} = -1 \mu\text{A}$	$V_{DD} - 0.05$			V
	类型 B4	$I_{OH} = -4 \text{ mA}$	2.4			
	类型 B6	$I_{OH} = -6 \text{ mA}$				
	类型 B8	$I_{OH} = -8 \text{ mA}$				
	类型 B10	$I_{OH} = -10 \text{ mA}$				
	类型 B12	$I_{OH} = -12 \text{ mA}$				
V_{OL}	低电平输出电压 类型 B4 至 B12	$I_{OL} = 1 \mu\text{A}$			0.05	V
	类型 B4	$I_{OL} = 4 \text{ mA}$			0.4	
	类型 B6	$I_{OL} = 6 \text{ mA}$				
	类型 B8	$I_{OL} = 8 \text{ mA}$				
	类型 B10	$I_{OL} = 10 \text{ mA}$				
	类型 B12	$I_{OL} = 12 \text{ mA}$				

注释：

4. 类型 B6 意思为 6mA 输出驱动器单元
5. 类型 B8 意思为 8mA 输出驱动器单元
6. 类型 B12 意思为 12mA 输出驱动器单元

表 27-4. USB 直流电气特性

符号	参数	条件	最小值	最大值	单位
V_{IH}	高电平输入电压	-	2.5	-	V
V_{IL}	低电平输入电压	-	-	0.8	V
I_{IH}	高电平输入电流	$V_{IN} = 3.3V$	-10	10	μA
I_{IL}	低电平输入电流	$V_{IN} = 0.0V$	-10	10	μA
V_{OH}	高电平输出电压	15K 至 GND	2.8	3.6	V
V_{OL}	低电平输出电压	1.5K 至 3.6V		0.3	V

表 27-5. S3C2440 供应电压和电流

参数	值	单位	条件
典型 V_{DDi} / V_{DDOP}	1.3 / 3.3	V	无 DVS
最大工作频率 (FCLK)	400	MHz	-
最大工作频率 (HCLK)	133	MHz	-
最大工作频率 (PCLK)	67	MHz	-
典型普通模式功耗 ⁽³⁾ (总计 $V_{DDi} + V_{IO}$)	368	mW	⁽¹⁾
典型普通模式功耗 ⁽³⁾ (总计 $V_{DDi} + V_{IO}$)	310	mW	⁽²⁾
典型空闲模式功耗 ⁽³⁾ (总计 $V_{DDi} + V_{IO}$)	213	mW	FCLK = 400MHz (F:H:P = 1:3:6)
典型慢速模式功耗 ⁽³⁾ (总计 $V_{DDi} + V_{IO}$)	97	mW	FCLK = 12MHz (F:H:P = 1:1:1)
典型睡眠模式功耗 ⁽³⁾	380	μA	在 1.2/3.3V , 室温 , 所有其它 I/O 静态
典型 RTC 功耗 ⁽³⁾	3	μA	在 3.0V , 室温 , X-tal = 32.768kHz 给 RTC

注释：

1. 指令/数据缓存：开； MMU：开； 代码在 SRAM； FCLK:HCLK:PCLK = 400MHz:133MHz:66.7MHz； LCD：开（320×240×16bpp×60Hz，彩色 TFT）； 13 kHz 定时器内部模式（5 通道运行）； 音频（IIS 和 DMA，CDCLK=16.9MHz，LRCK=44.1kHz）； 整型数据快速排序（65536 EA）
2. Pocket PC 2003 MPEG 显示
3. 上述功耗数据是在室温下随机采样测量的（Lot #: KZZ1FS）

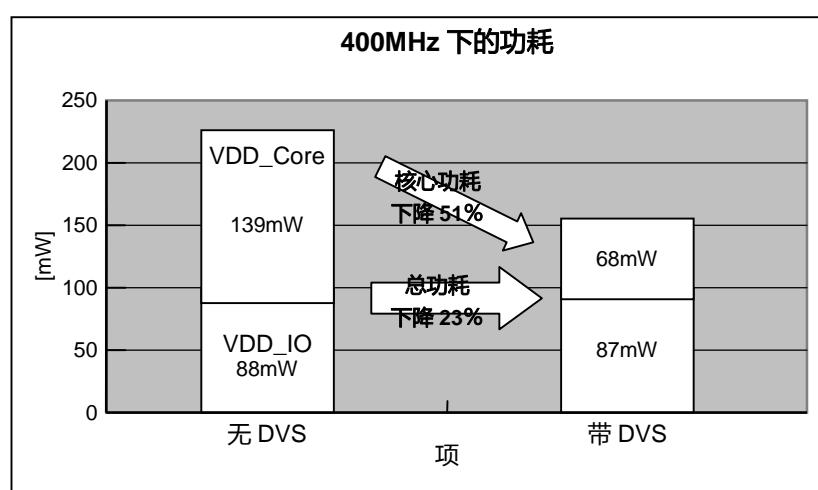


图 27-1. 当应用 DVS 设计时功耗例子对比

注释：

(条件) 当前测量条件：在 PPC2003 SMDK2440 上显示 Battlife.wma (比特率=64kbps)。

核心功耗：

无 DVS: $V_{DDalarm}/V_{DDi}/V_{DDupll}/V_{DDmpl}/V_{DDalive} = 1.3V$

使用 DVS: $V_{DDalarm}/V_{DDi} = 1.3V \Leftrightarrow 1.0V, V_{DDupll}/V_{DDmpl}/V_{DDalive} = 1.3V$

I/O 功耗: $V_{DDOP}/V_{DDMOP}/V_{DDRTC}/V_{DDADC} = 3.3V$

请参考应用手册以获取更多关于 DVS 的信息。

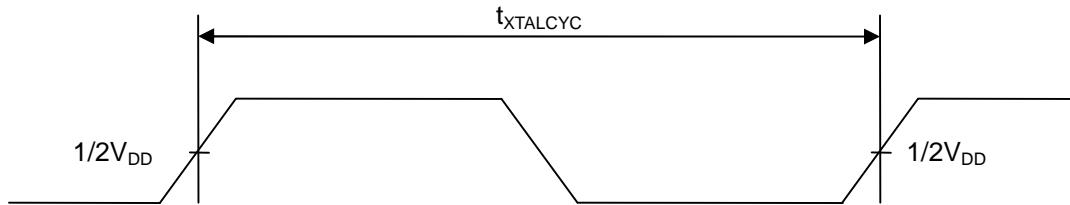
表 27-6. 由 CLKCON 寄存器典型电流减少量

外设	NFC	LCD	USBH	USBD	定时器	SDI	UART	RTC	ADC	IIC	IIS	SPI	摄像头	总计
电流	1.7	2.8	0.37	0.79	0.32	1.0	2.7	0.45	0.26	0.32	0.78	0.16	14.25	26.26

注释：

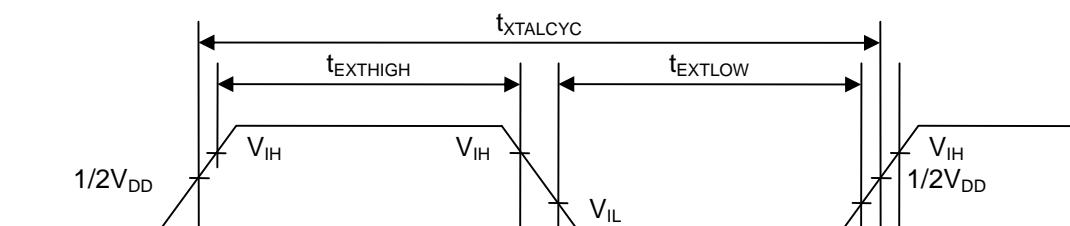
此表包括了每种外设的功耗。例如，如果不希望使用摄像头并通过 CLKCON 寄存器关闭了摄像头模块，则内部模块可以节约 14.25mA。

交流电气特性



注释：时钟输入是来自 XTIPLL 引脚。

图 27-2. XTIPLL 时钟时序图



注释：时钟输入是来自 EXTCLK 引脚。

图 27-3. EXTCLK 时钟输入时序图

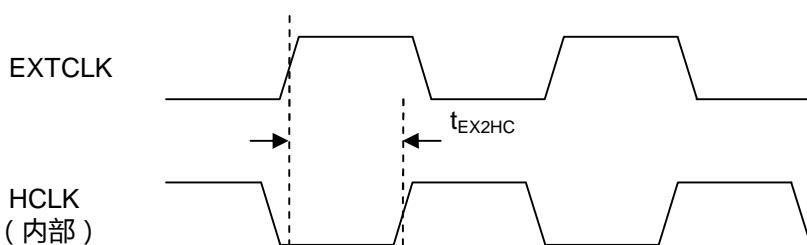


图 27-4. 当使用无 PLL 的 EXTCLK 时的 EXTCLK/HCLK

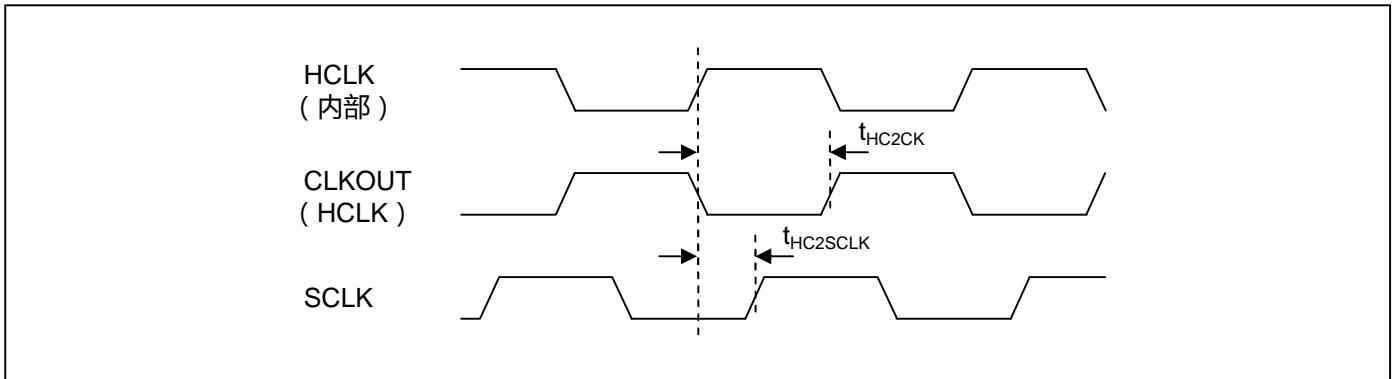


图 27-5. 当使用了 EXTCLK 的 HCLK/CLKOUT/SCLK

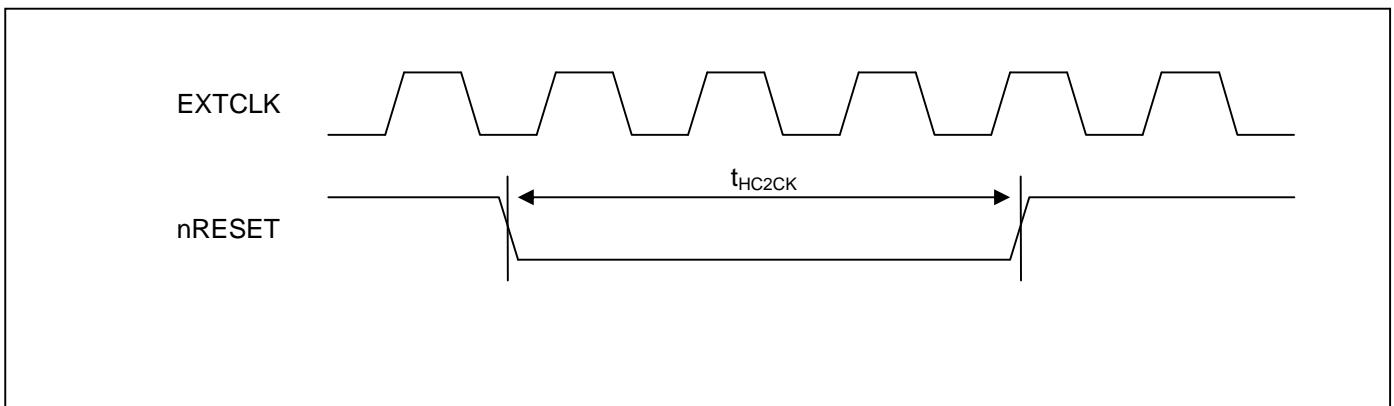


图 27-6. 手动复位输入时序图

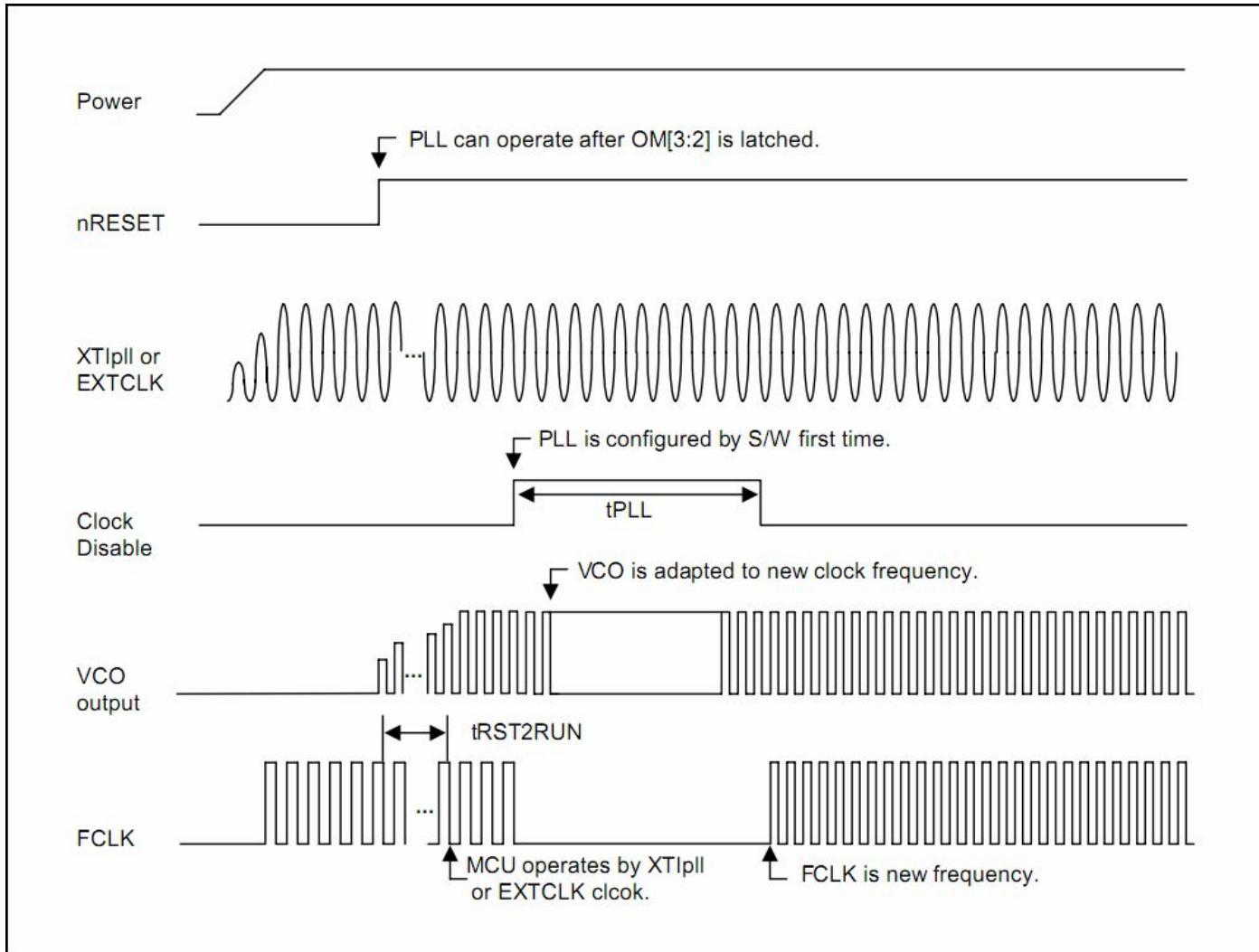


图 27-7. 上电振荡设置时序图

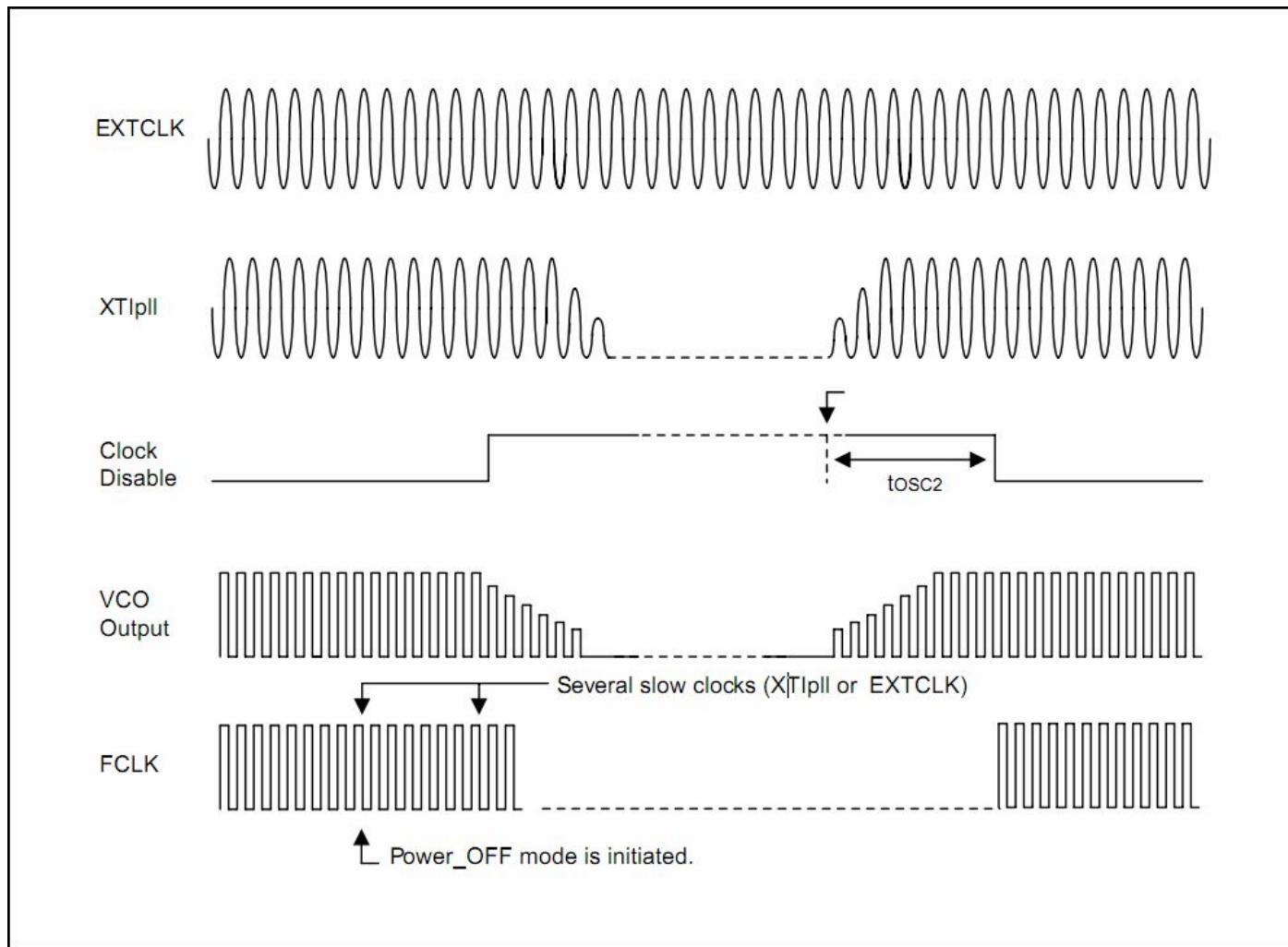


图 27-8. 睡眠模式恢复振荡设置时序图

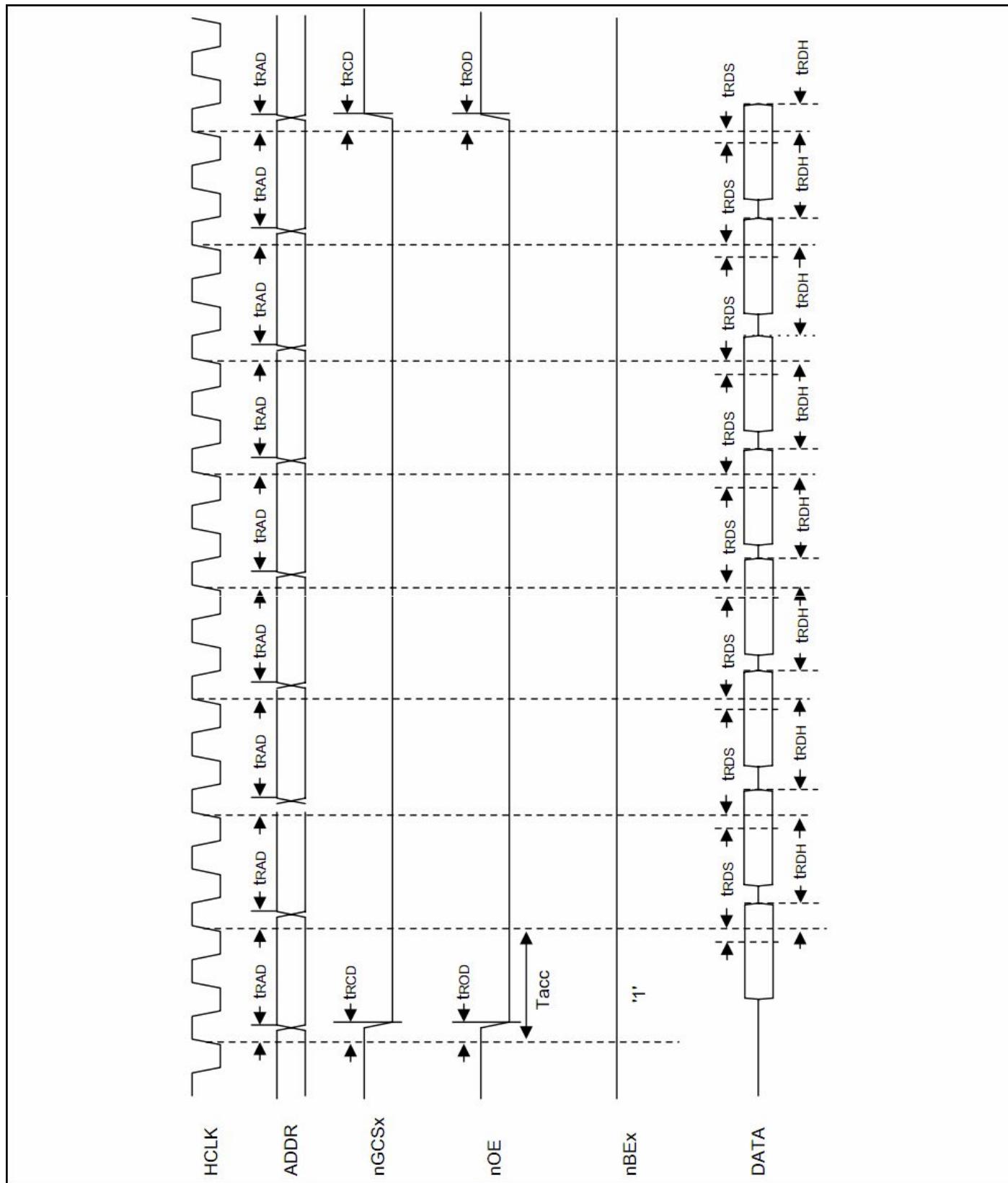


图 27-9. ROM/SRAM 突发读时序图 (I)

($T_{acs}=0$, $T_{cos}=0$, $T_{acc}=2$, $Toch=0$, $Tcah=0$, $PMC=0$, $ST=0$, $DW=16$ 位)

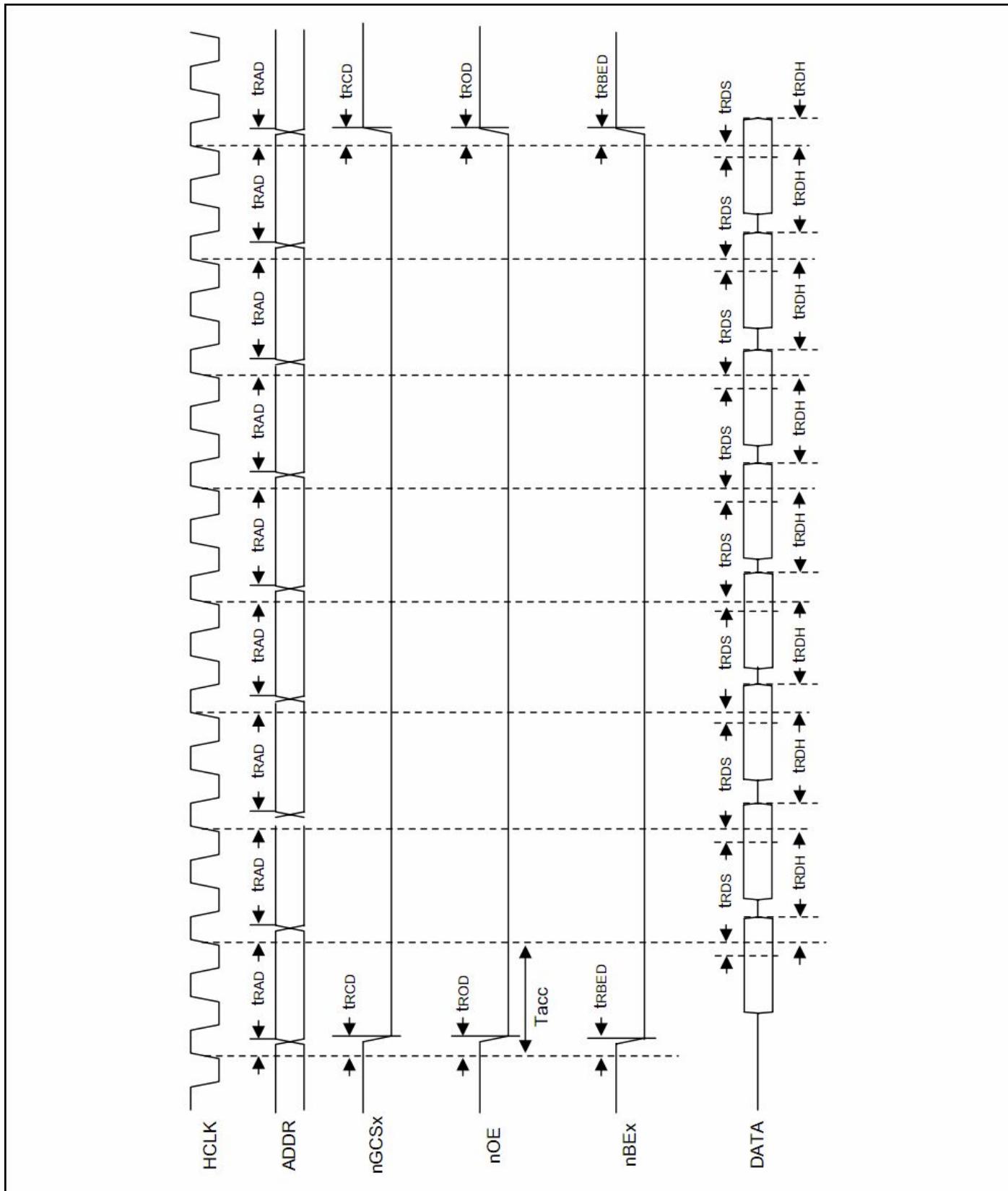


图 27-10. ROM/SRAM 突发读时序图 (II)

(T_{acs}=0 , T_{c0s}=0 , T_{acc}=2 , T_{och}=0 , T_{cah}=0 , PMC=0 , ST=1 , DW=16 位)

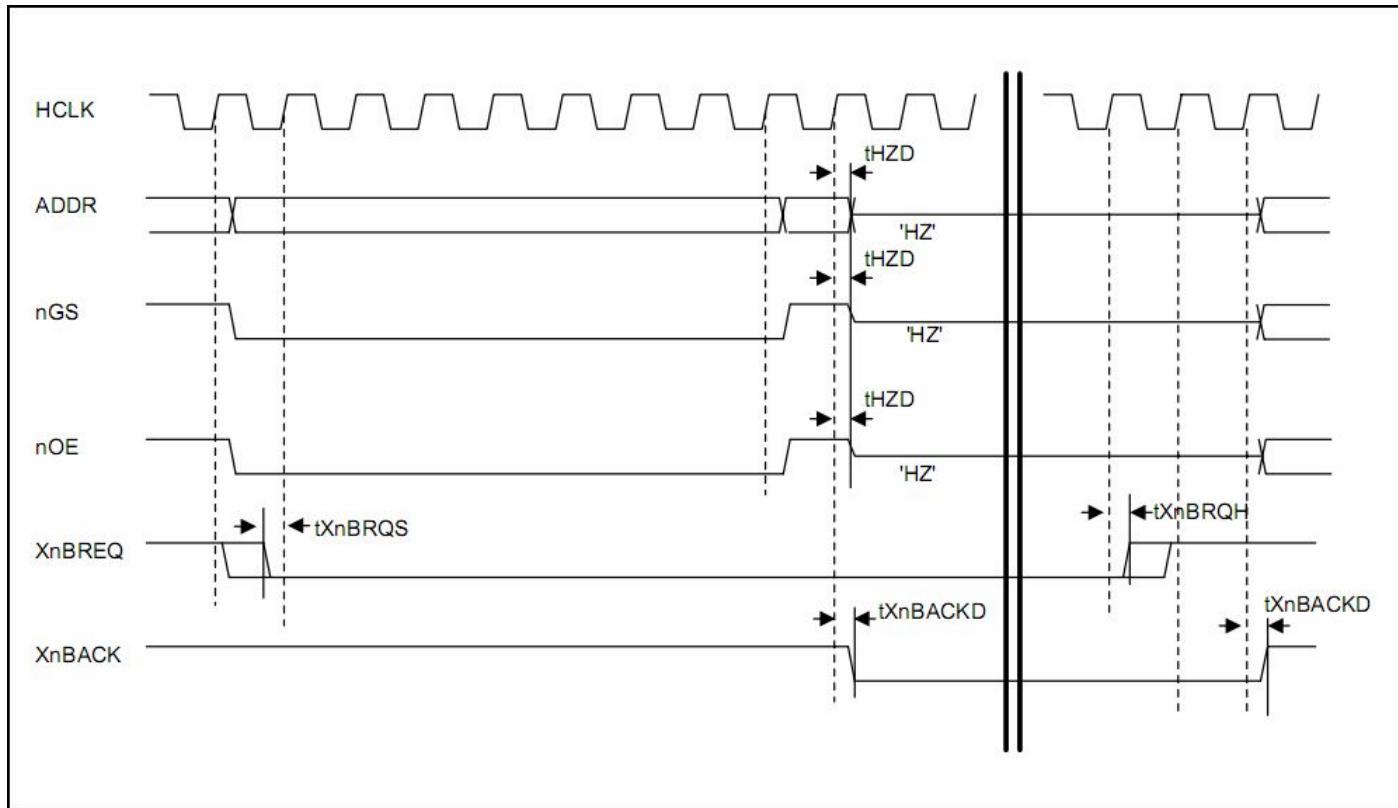


图 27-11. ROM/SRAM 周期中外部总线请求

(Tacs=0 , Tcos=0 , Tacc=8 , Toch=0 , Tcah=0 , PMC=0 , ST=0)

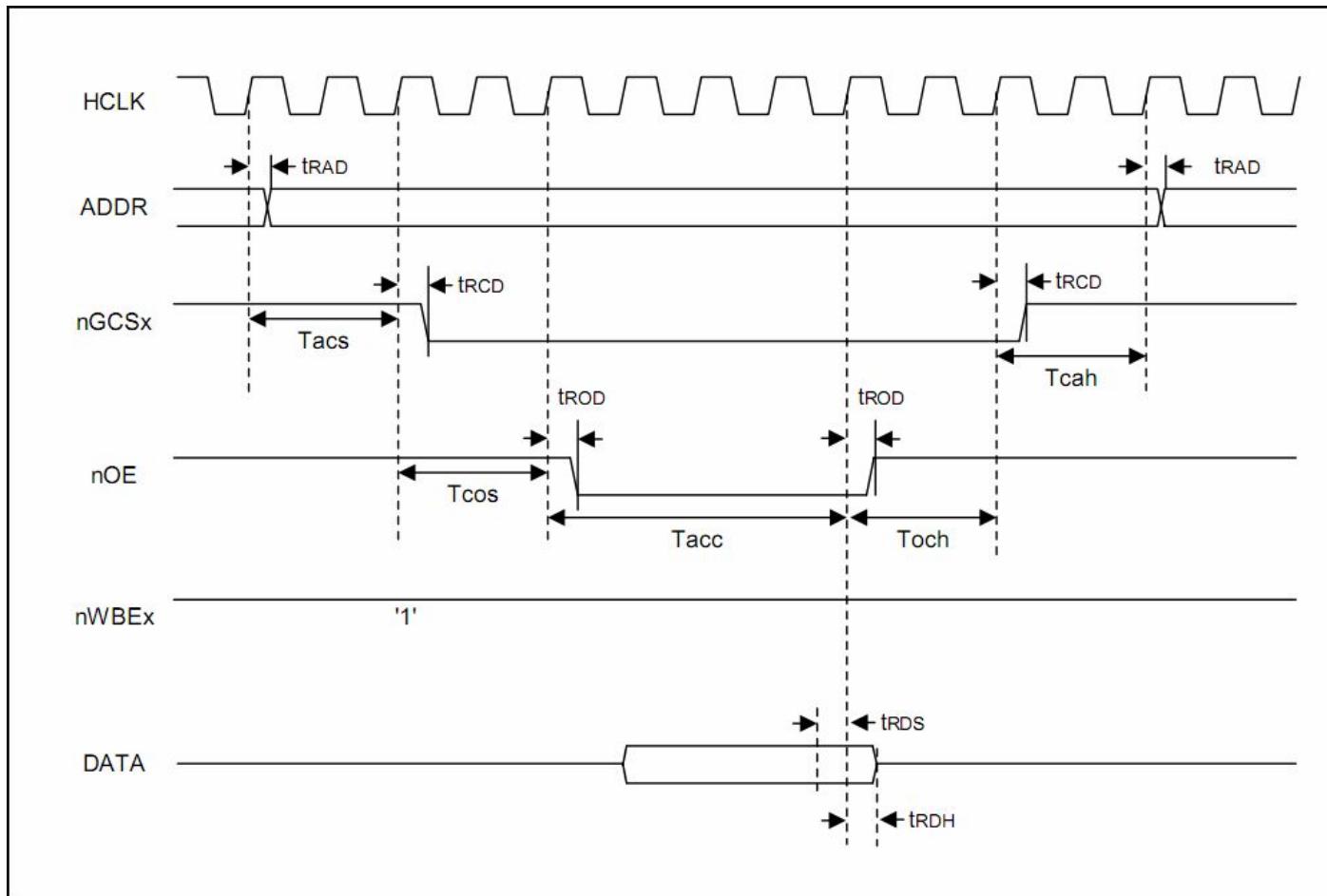


图 27-12. ROM/SRAM 读时序图 (I)

($T_{acs}=2$, $T_{cos}=2$, $T_{acc}=4$, $Toch=2$, $T_{cah}=2$, $PMC=0$, $ST=0$)

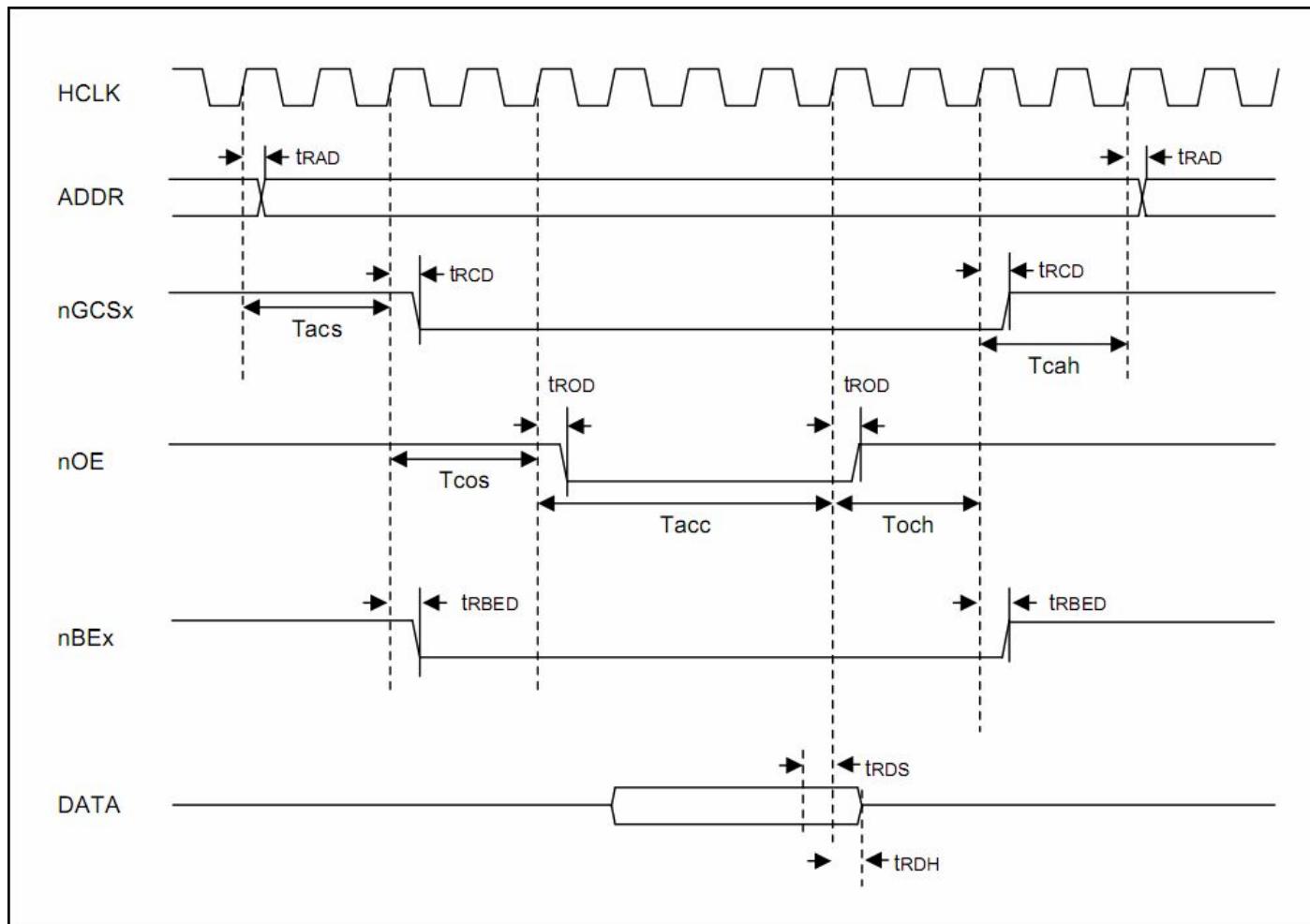


图 27-13. ROM/SRAM 读时序图 (II)

($T_{acs}=2$, $T_{cos}=2$, $T_{acc}=4$, $Toch=2$, $T_{crah}=2$ 周期 , $PMC=0$, $ST=1$)

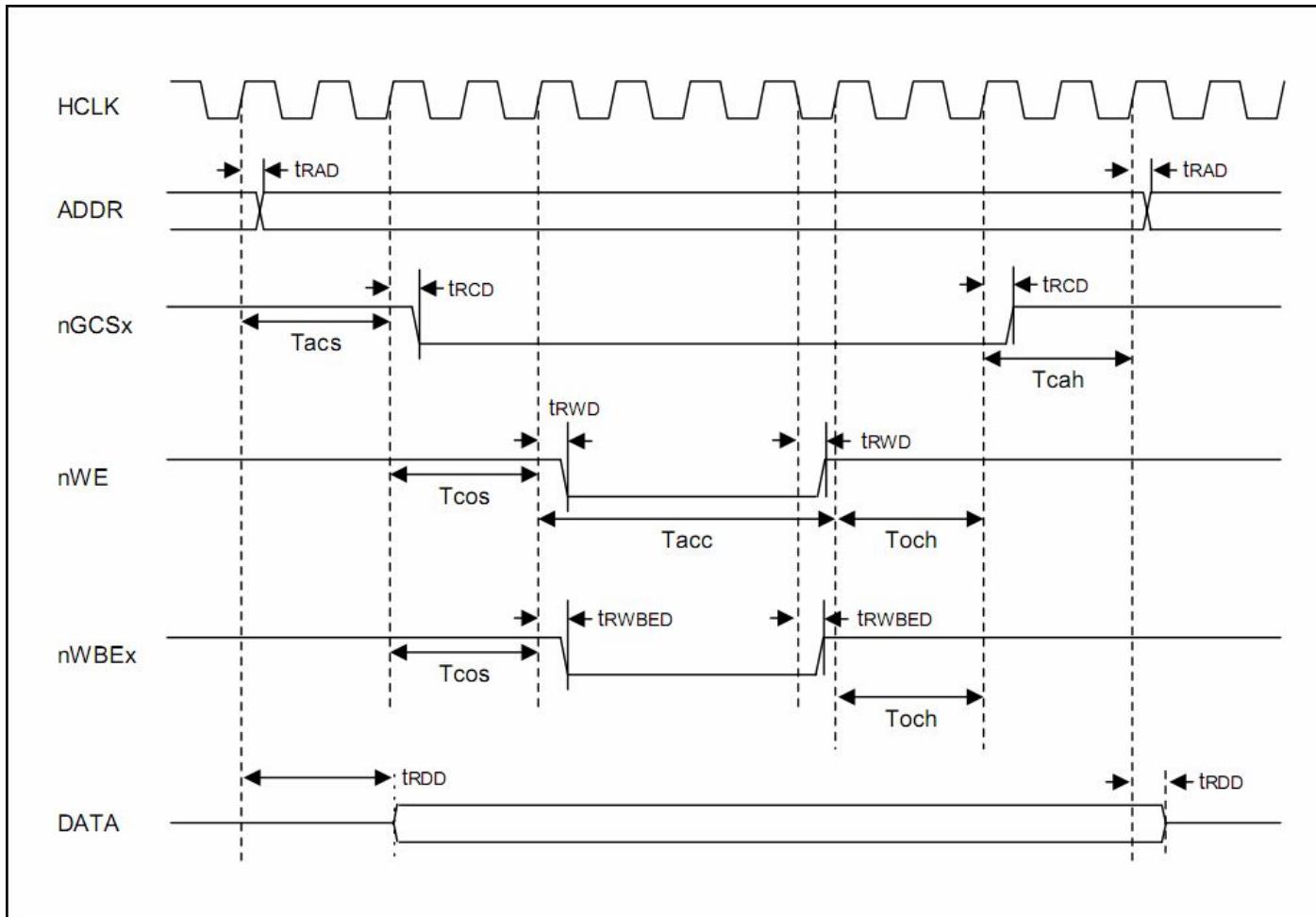


图 27-14. ROM/SRAM 写时序图 (I)

($T_{acs}=2$, $T_{cos}=2$, $T_{acc}=4$, $Toch=2$, $T_{cach}=2$, $PMC=0$, $ST=0$)

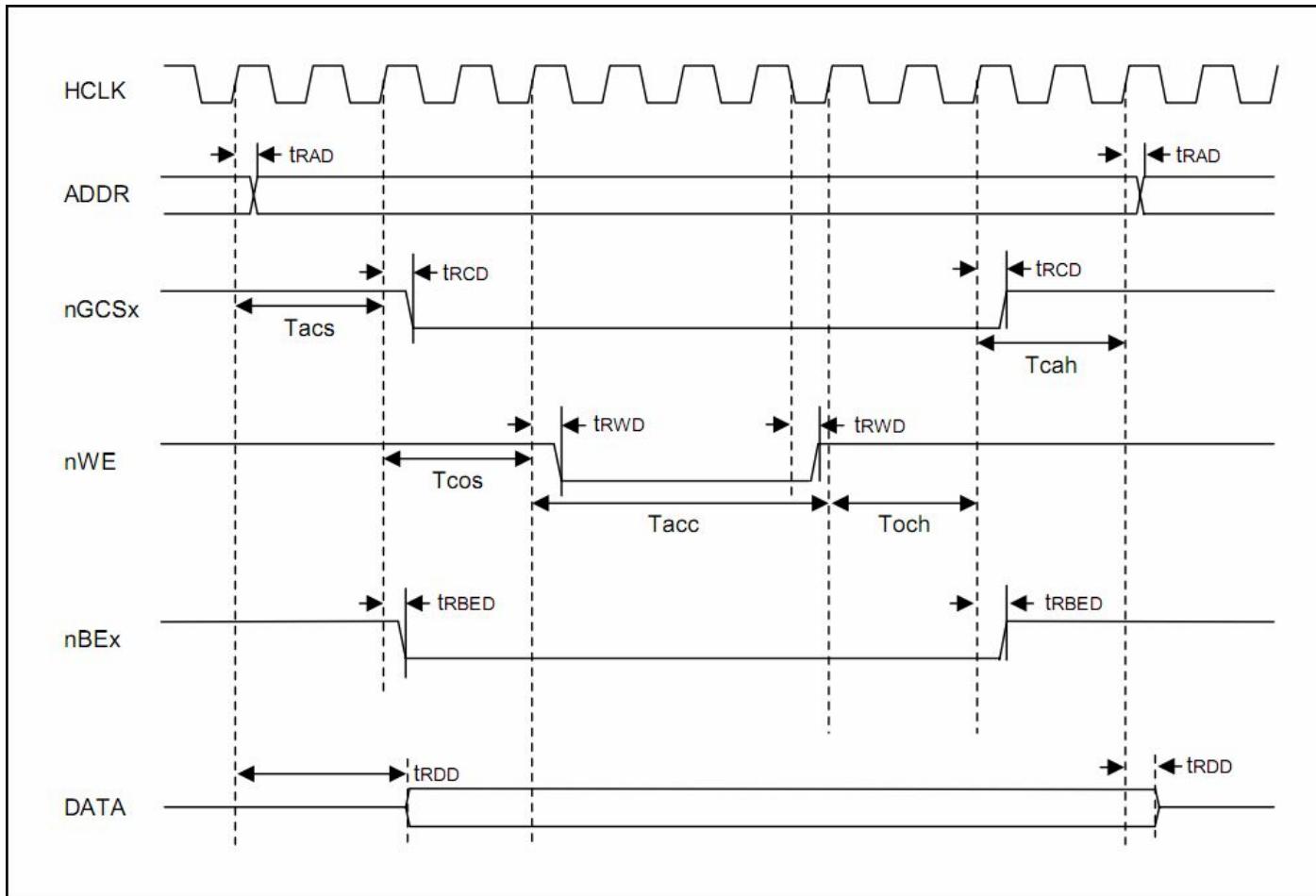


图 27-15. ROM/SRAM 写时序图 (II)

(Tacs=2 , Tcos=2 , Tacc=4 , Toch=2 , Tcah=2 , PMC=0 , ST=1)

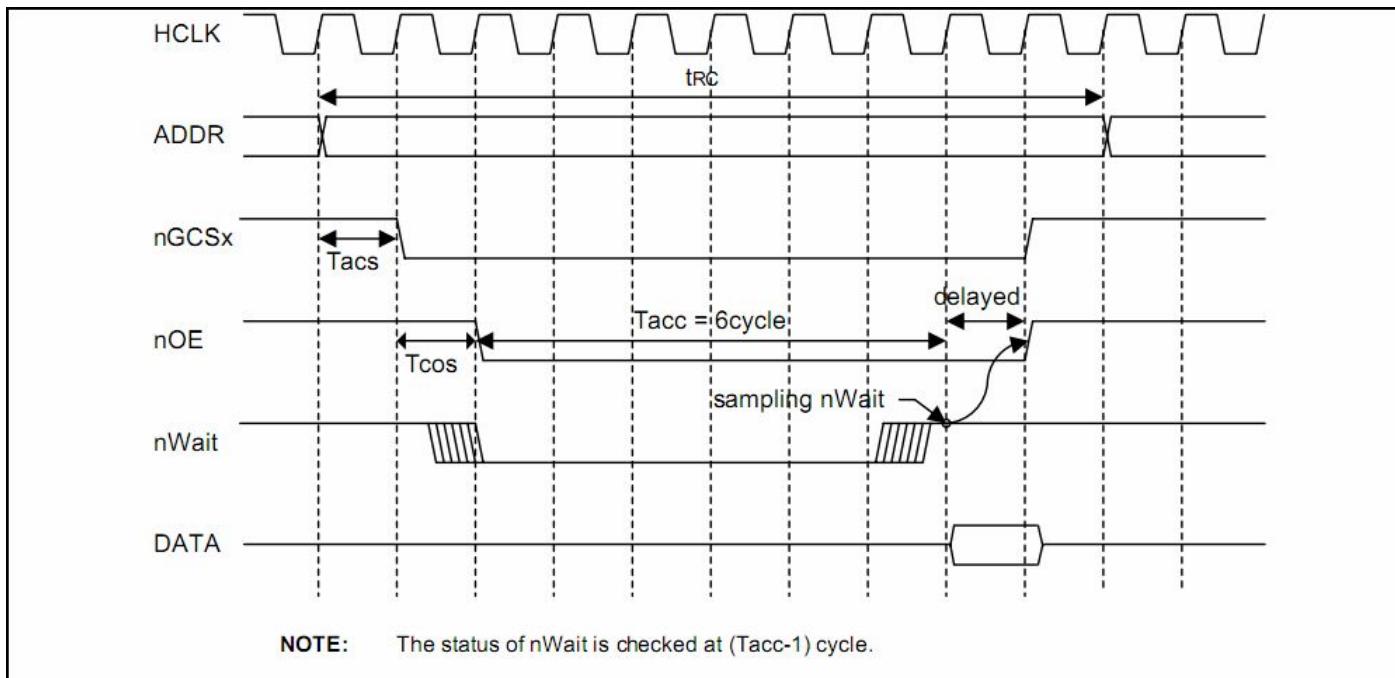


图 27-16. 外部 nWAIT 读时序图

(Tacs=1 , Tcos=1 , Tacc=4 (6 周期) , Toch=0 , Tcah=0 , PMC=0 , ST=0)

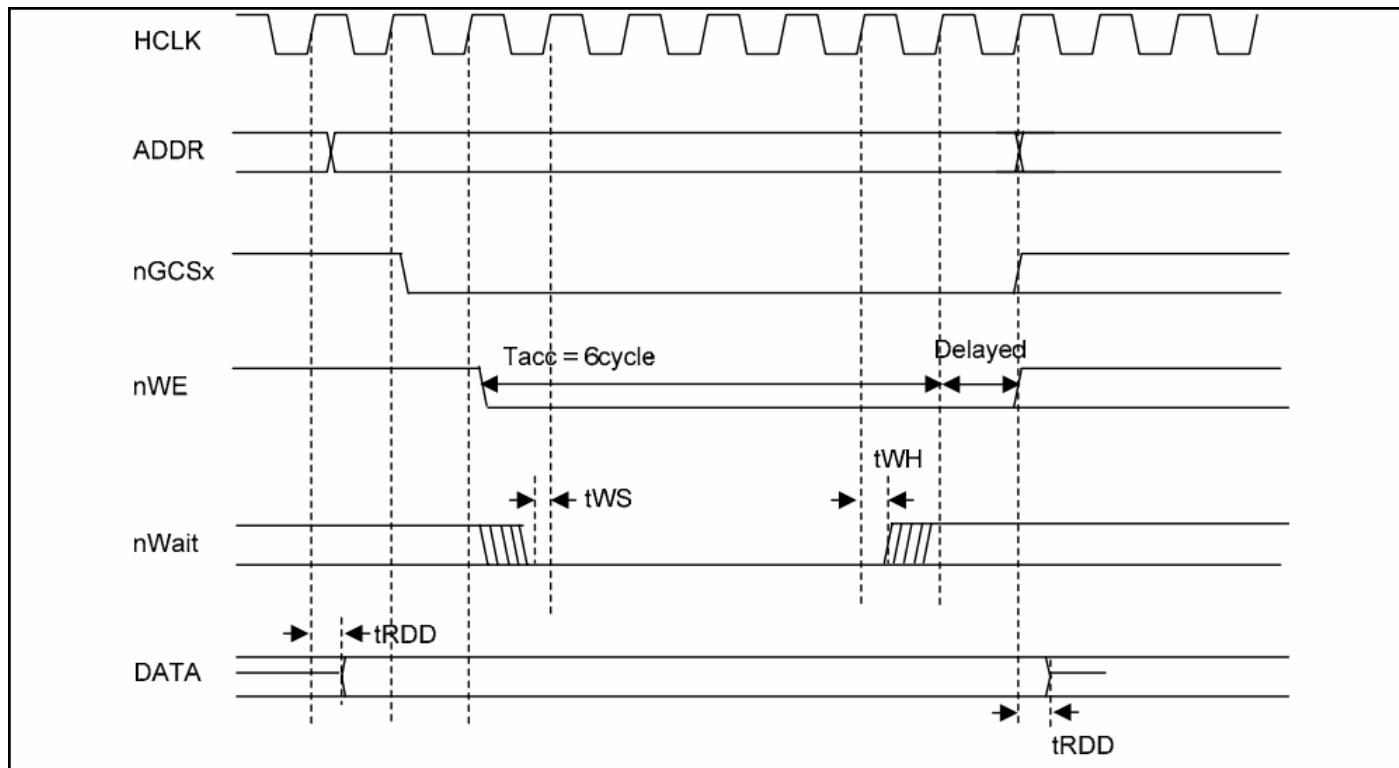


图 27-17. 外部 nWAIT 写时序图

(Tacs=1 , Tcos=1 , Tacc=4 (6 周期) , Toch=0 , Tcah=0 , PMC=0 , ST=0)

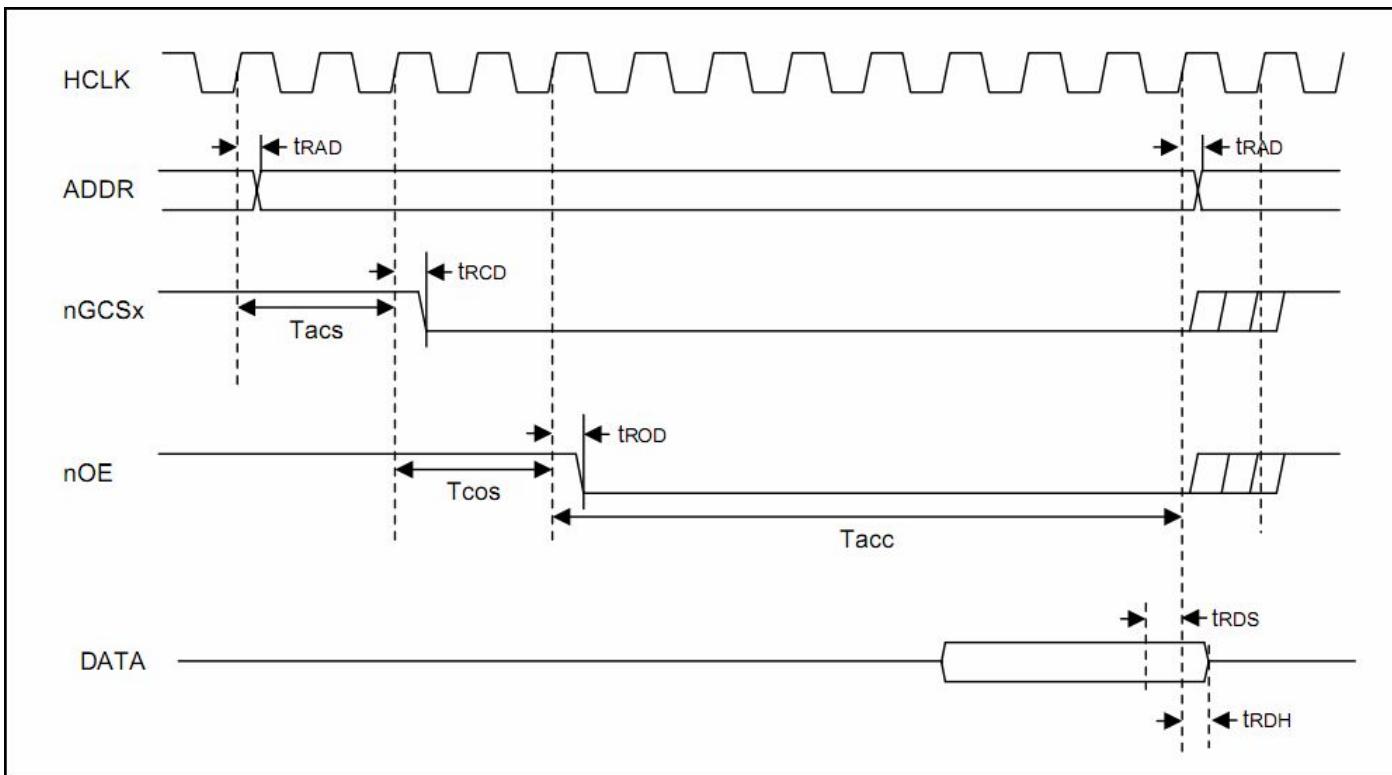


图 27-18. 掩模 ROM 单次读时序图

(Tacs=2 , Tcos=2 , Tacc=8 , PMC=01/10/11)

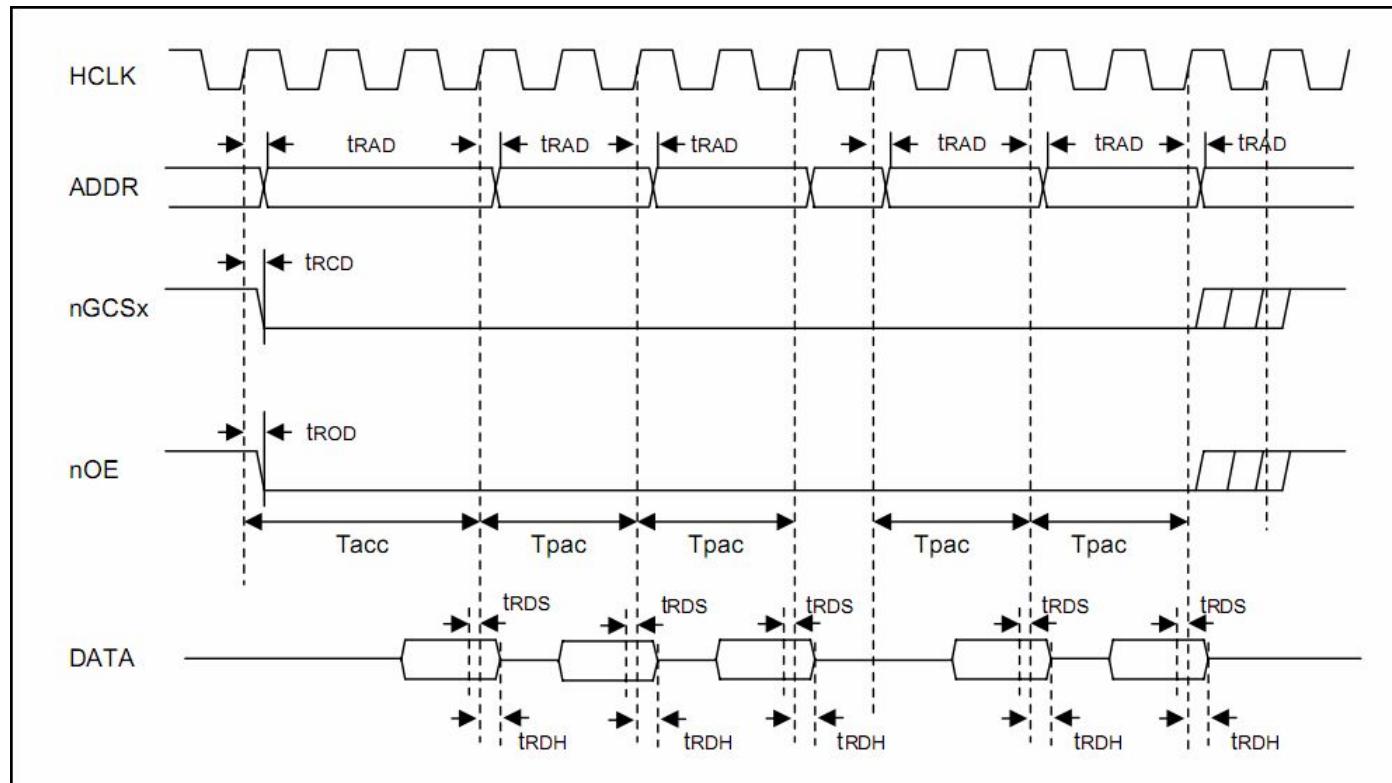


图 27-19. 掩模 ROM 连续读时序图

(Tacs=0 , Tcos=0 , Tacc=3 , Tpac=2 , PMC=01/10/11)

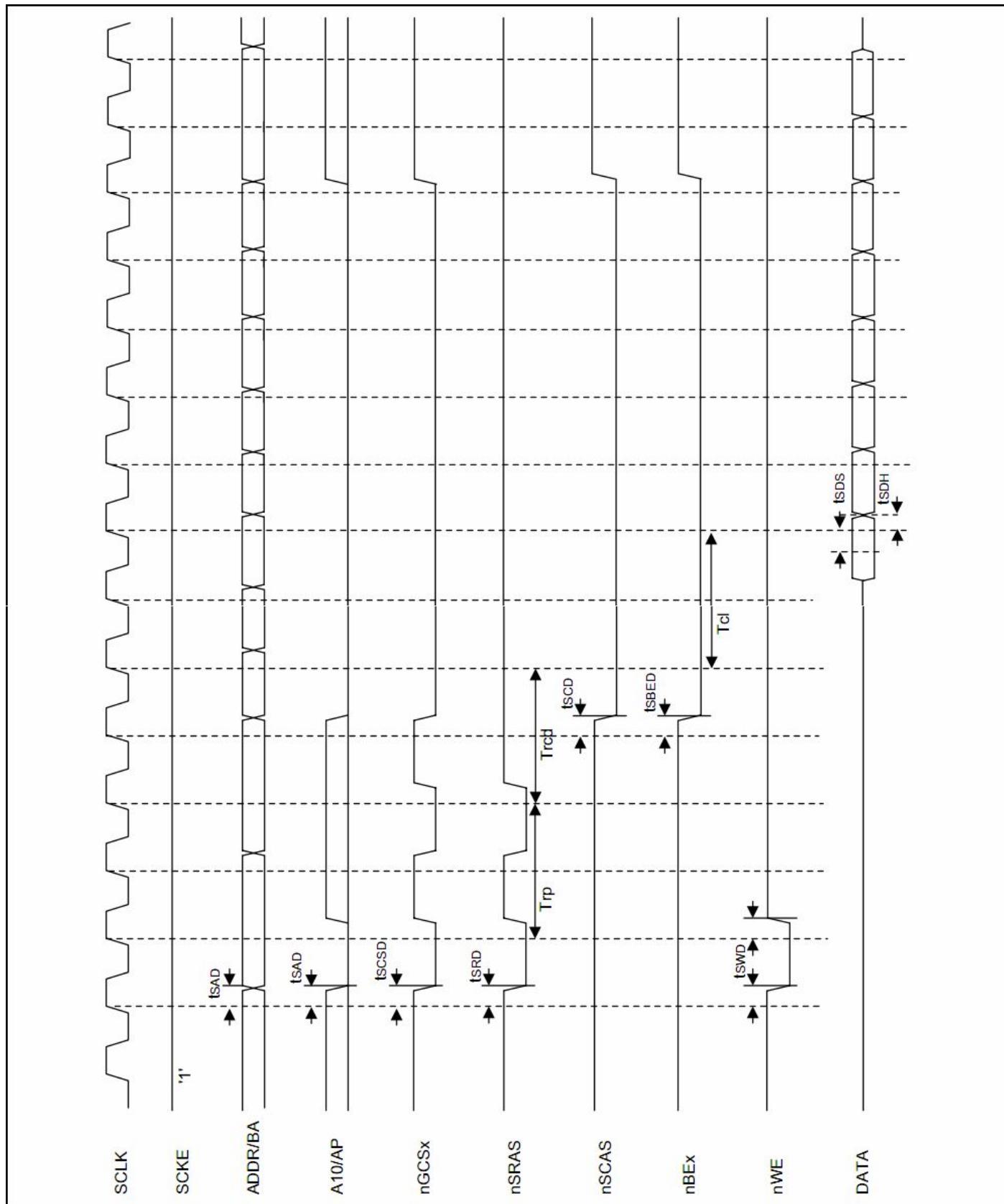


图 27-20. SDRAM 单次突发读时序图

($\text{Trp}=2$, $\text{Trcd}=2$, $\text{Tcl}=2$, $\text{DW}=16$ 位)

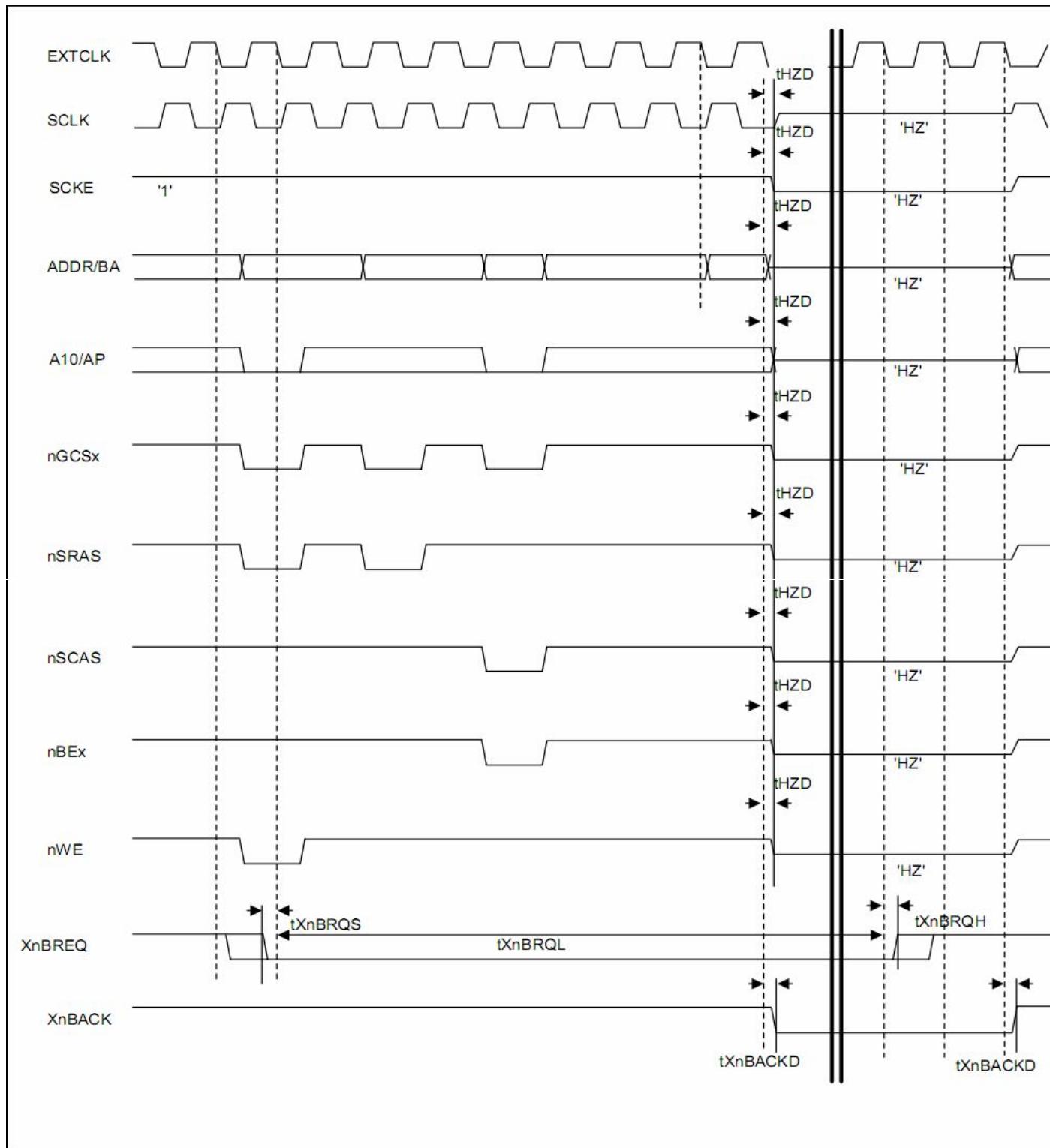


图 27-21. SDRAM 中外部总线请求时序图

($T_{RP}=2$, $T_{RC}=2$, $T_{CL}=2$)

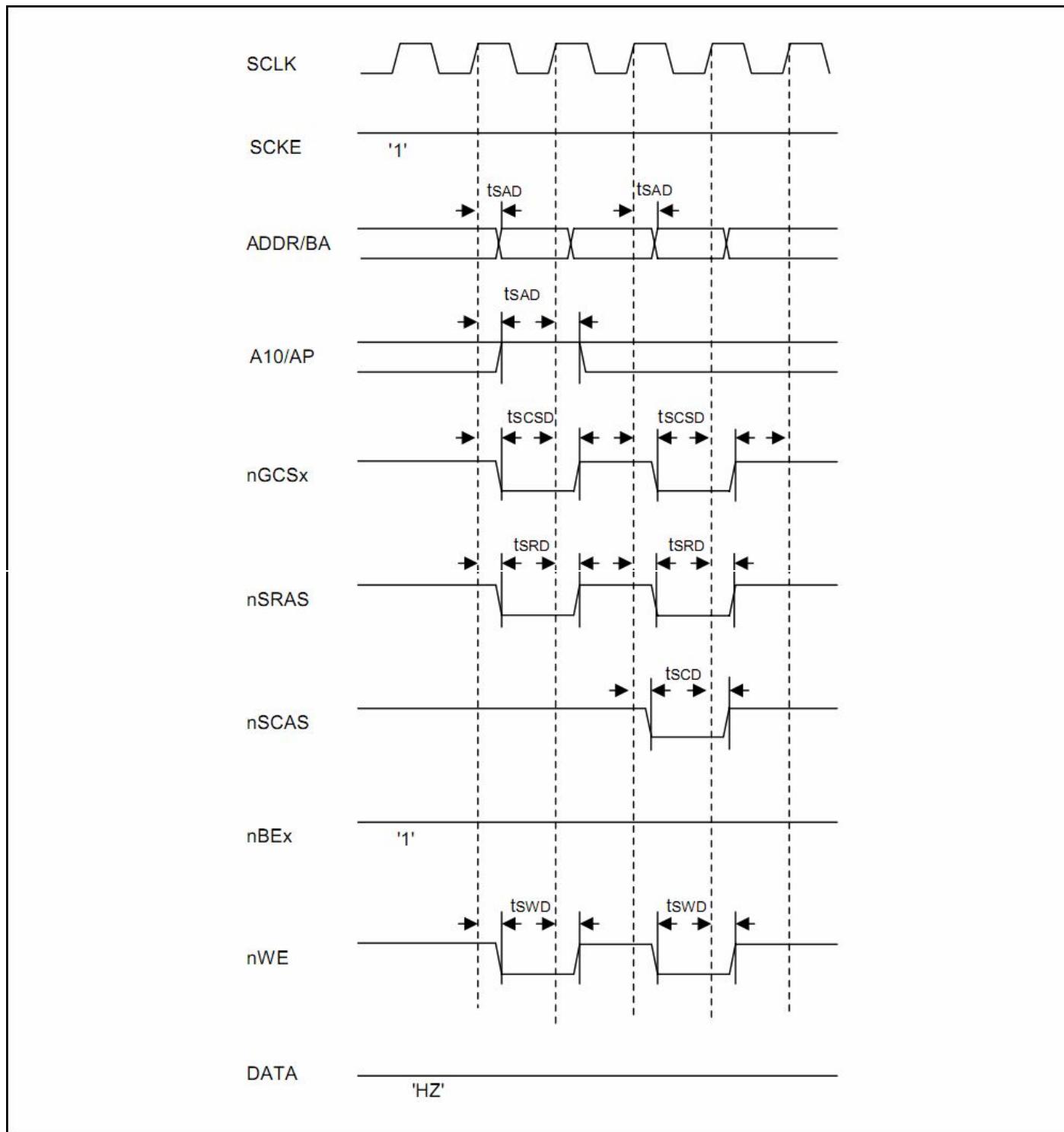


图 27-22. SDRAM MRS 时序图

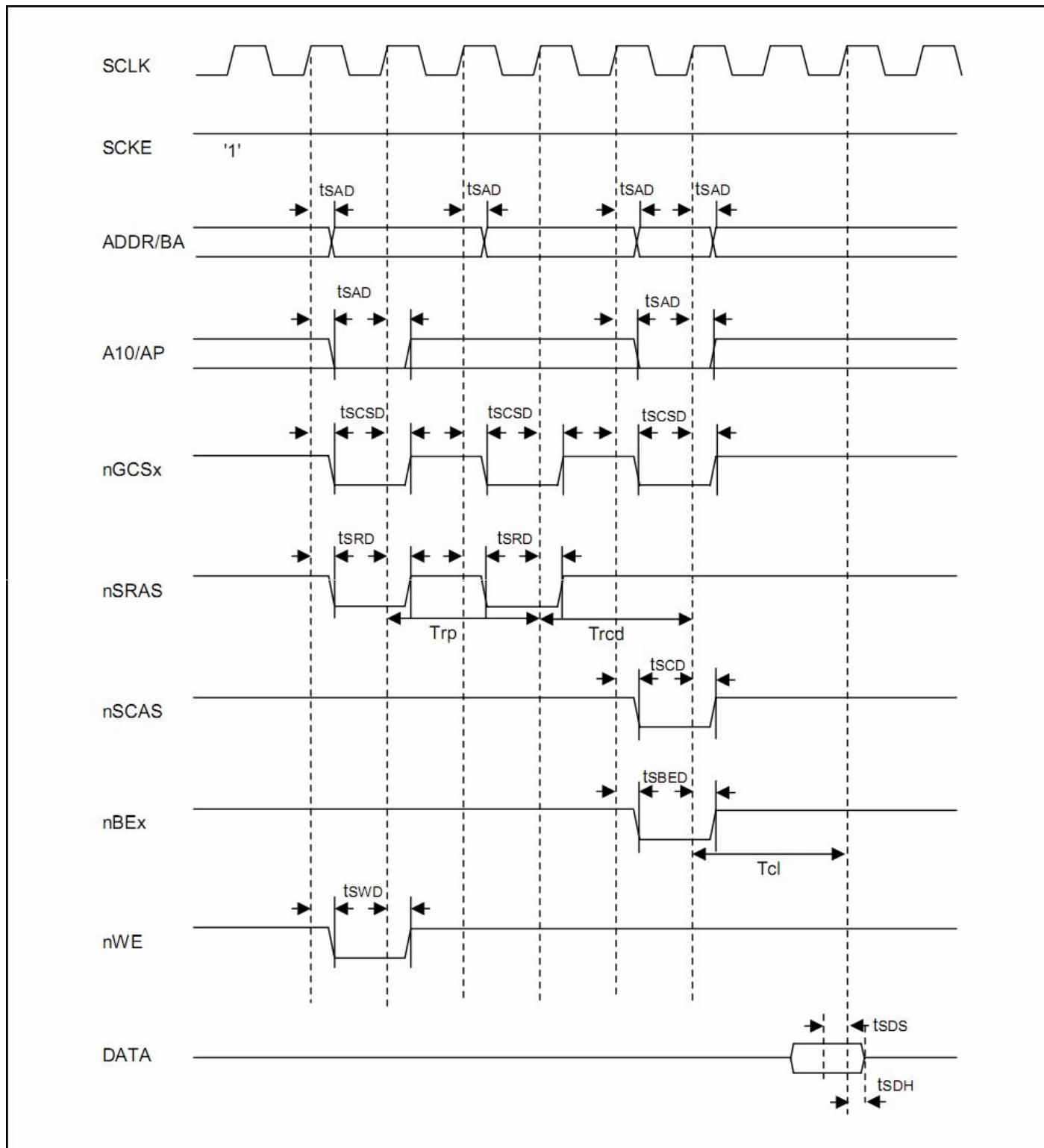


图 27-23. SDRAM 单次读时序图 (I)

(Trp=2 , Trcd=2 , Tcl=2)

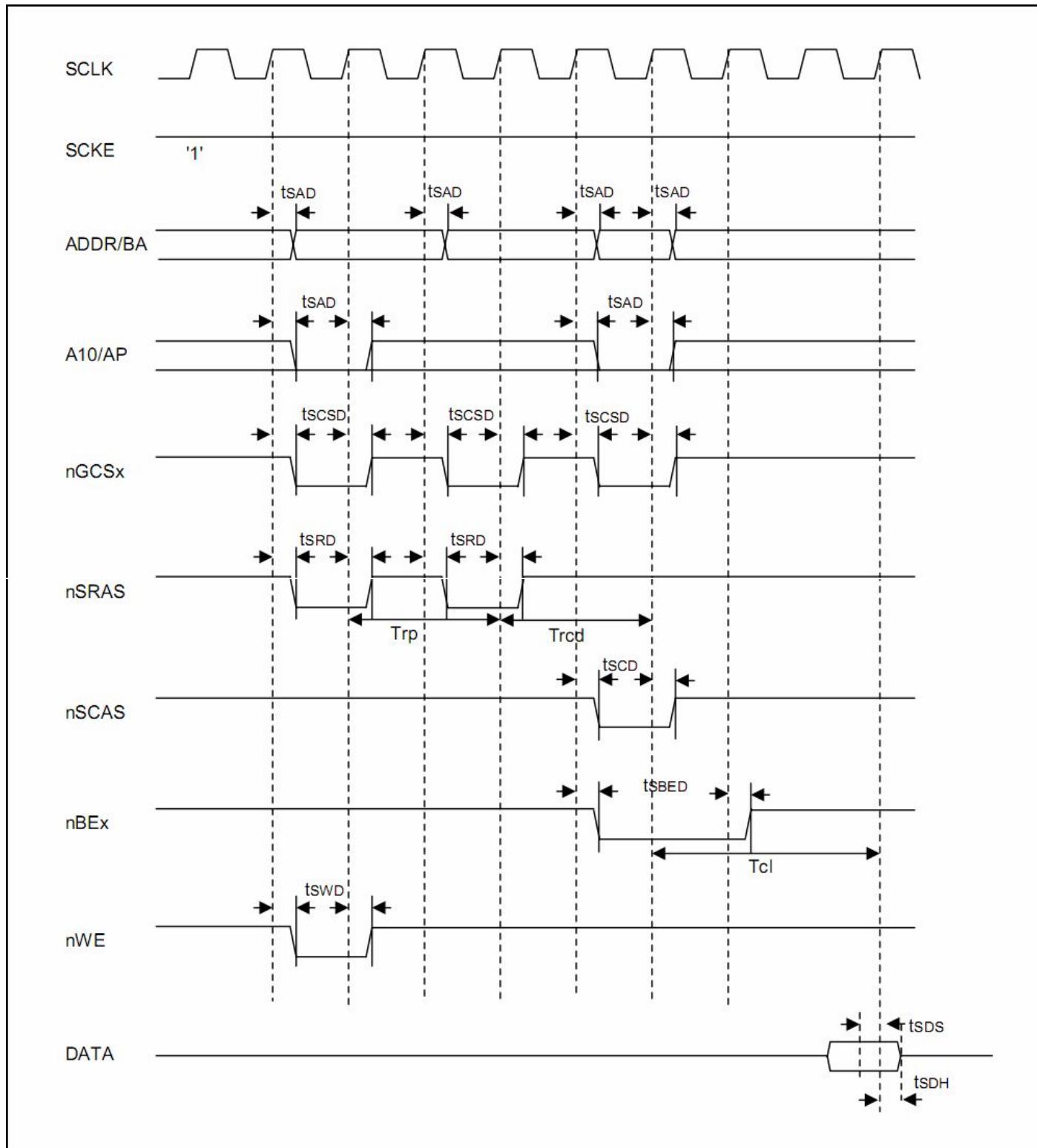


图 27-24. SDRAM 单次读时序图 (II)

($T_{RP}=2$, $T_{RC}=2$, $T_{CL}=3$)

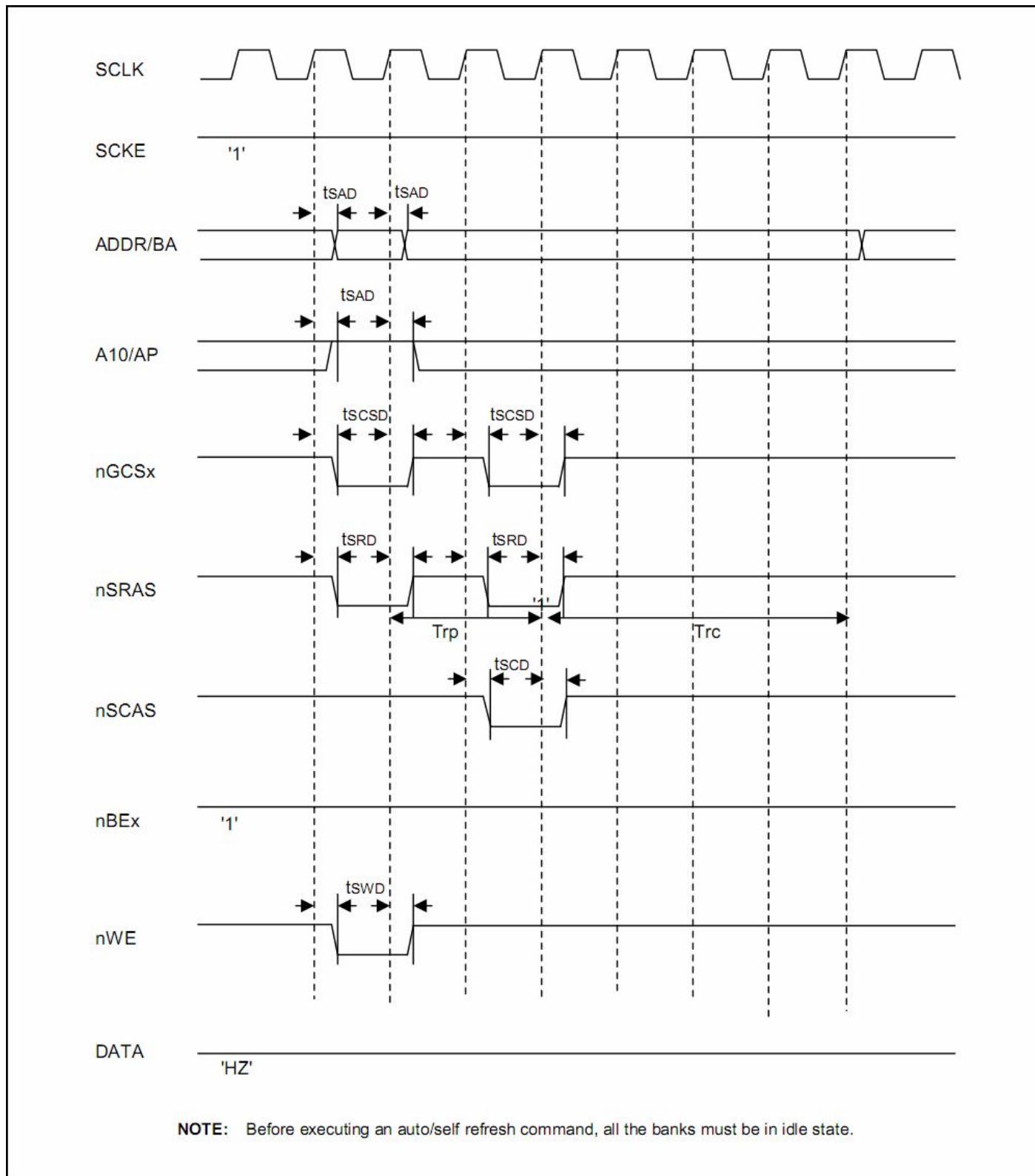


图 27-25. SDRAM 自动刷新时序图
(Trp=2 , Trc=4)

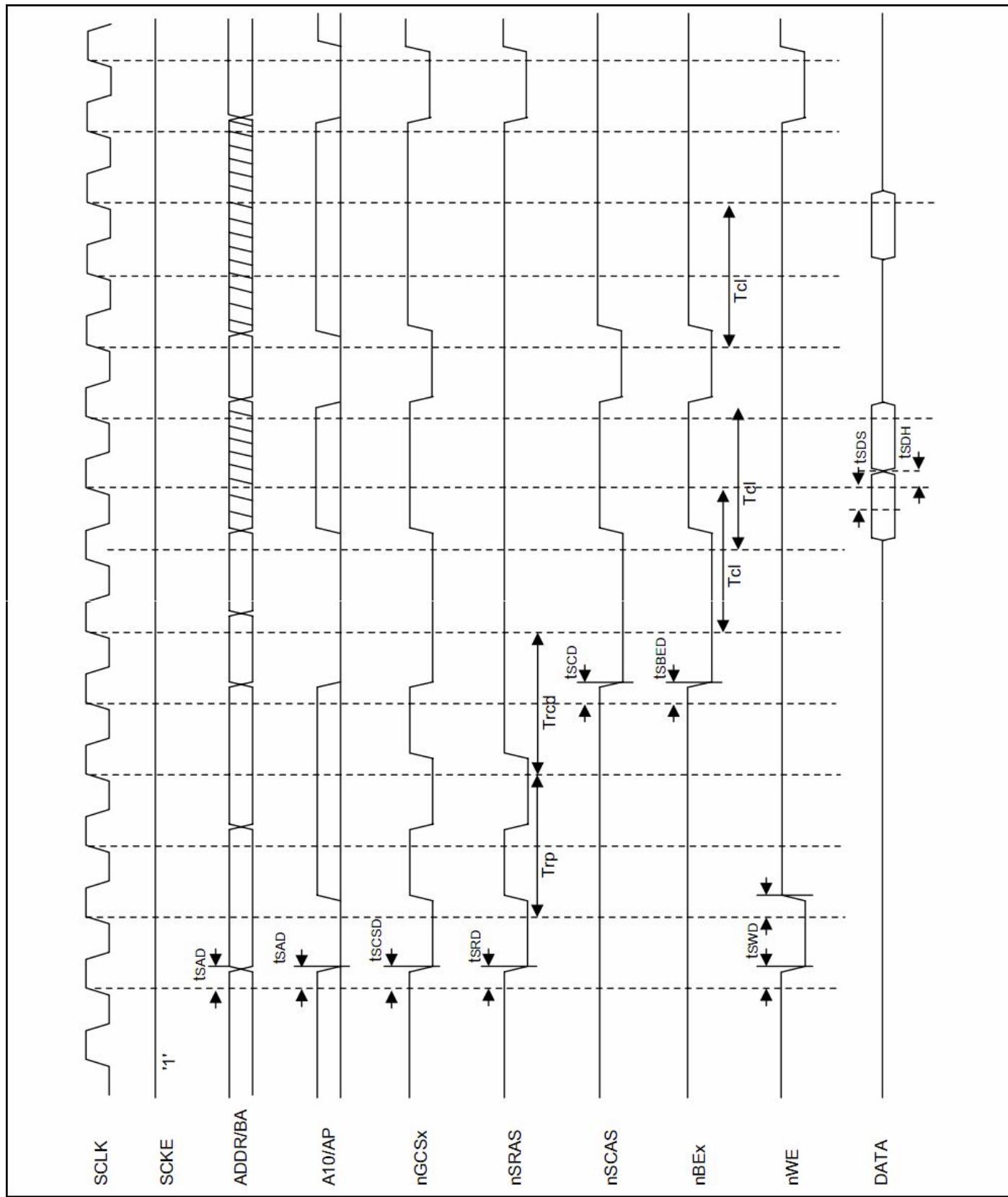


图 27-26. SDRAM 页 Hit-Miss 读时序图

($T_{RP}=2$, $T_{RCd}=2$, $T_{CL}=2$)

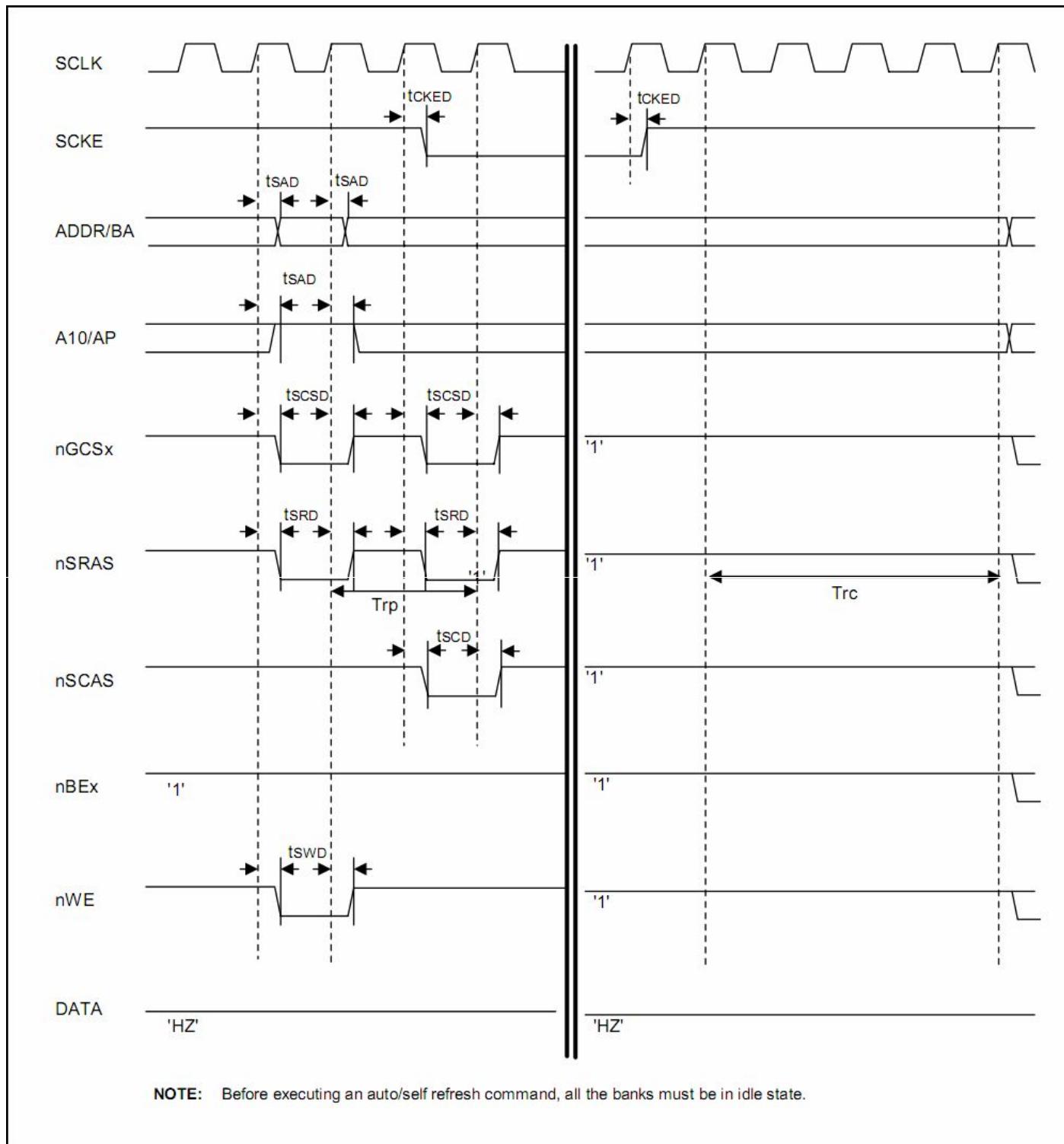


图 27-27. SDRAM 自刷新时序图
(Trp=2 , Trc=4)

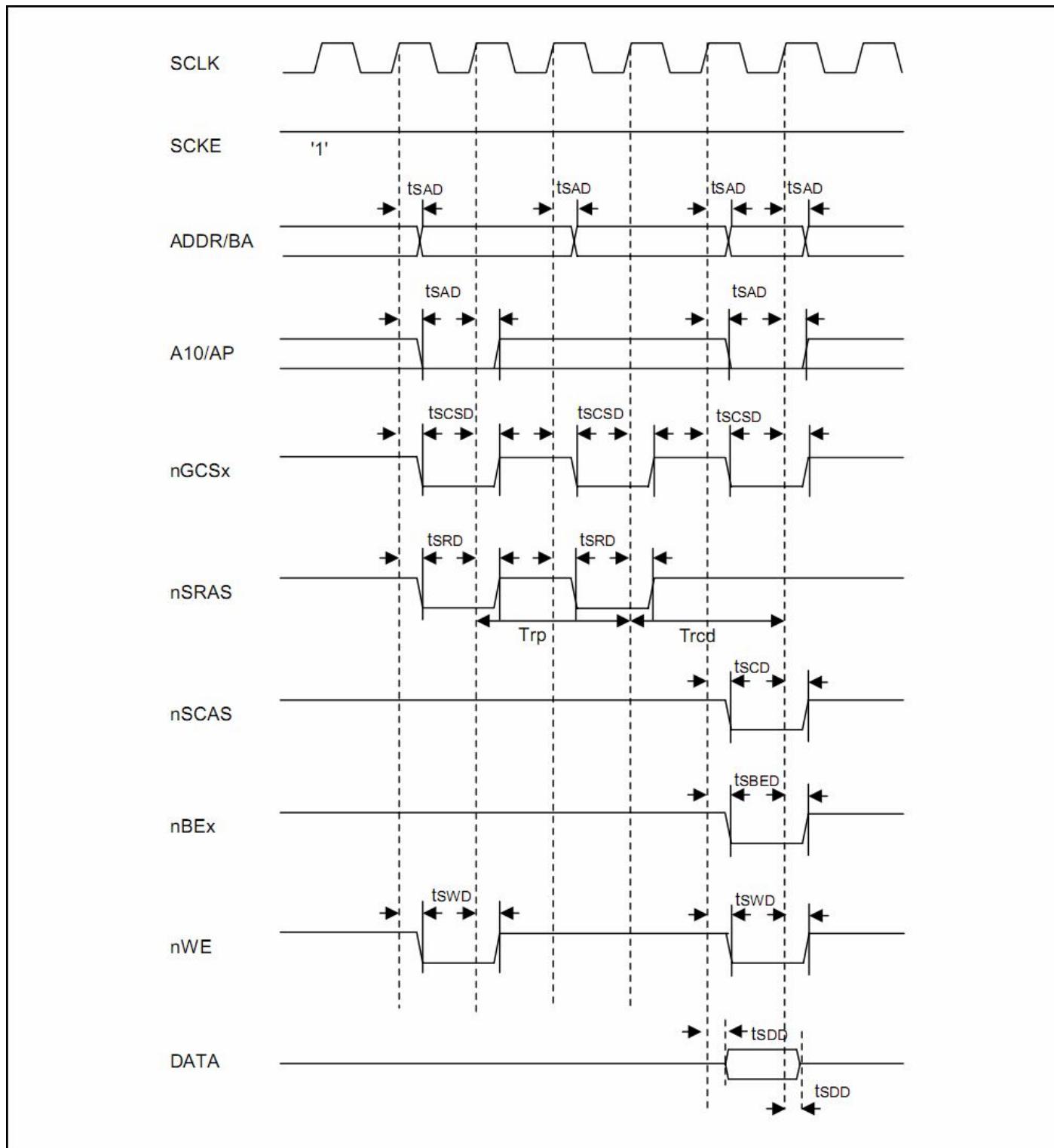


图 27-28. SDRAM 单次写时序图
(Trp=2 , Trcd=2)

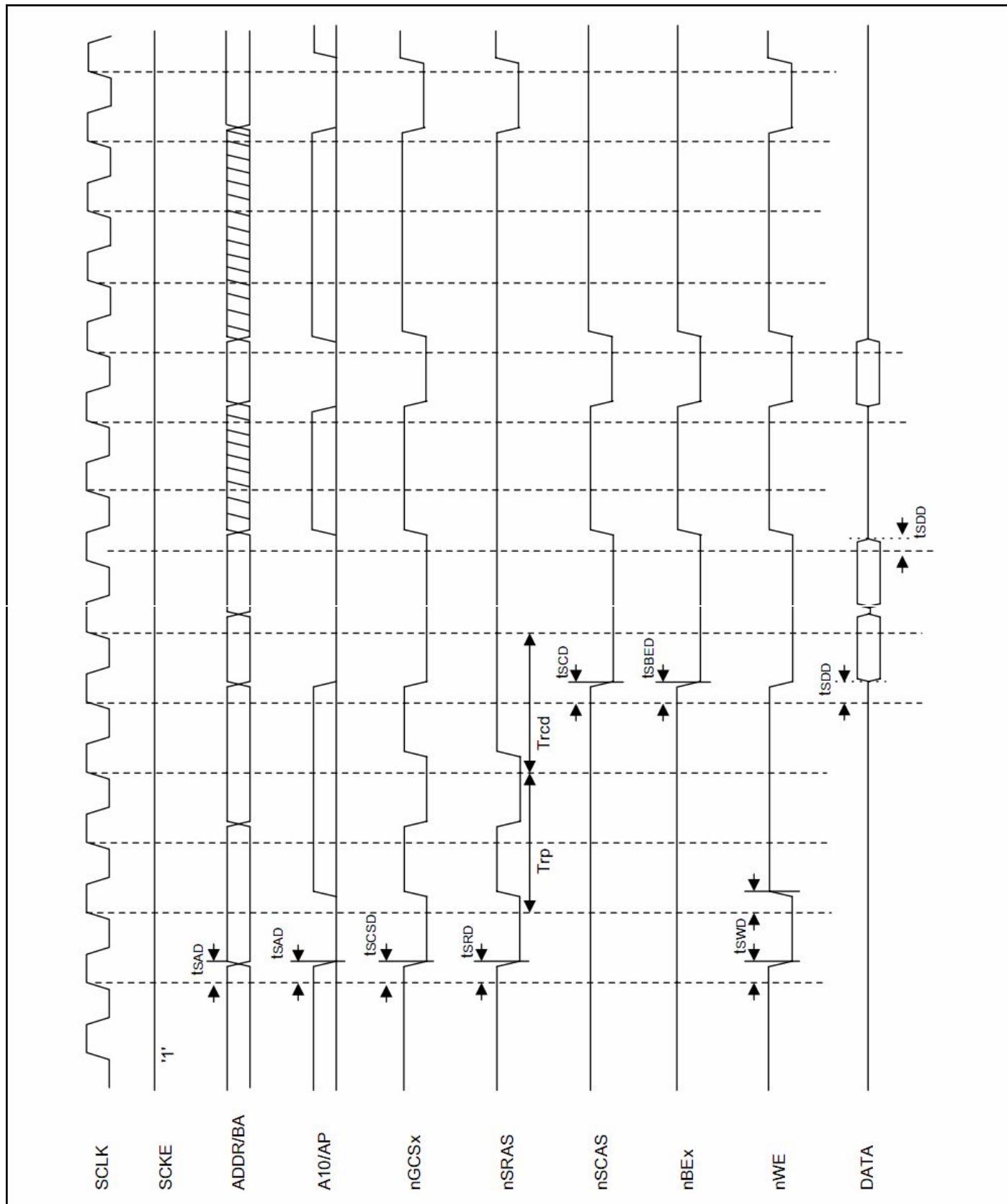


图 27-29. SDRAM 页 Hit-Miss 写时序图

(Trp=2 , Trcd=2 , Tcl=2)

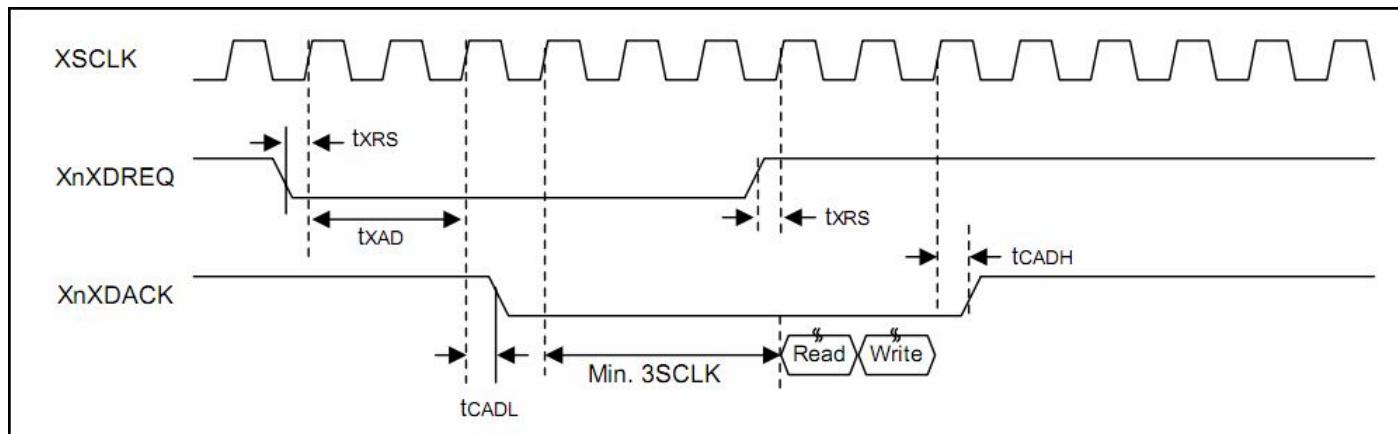


图 27-30. 外部 DMA 时序图
(握手, 单次传输)

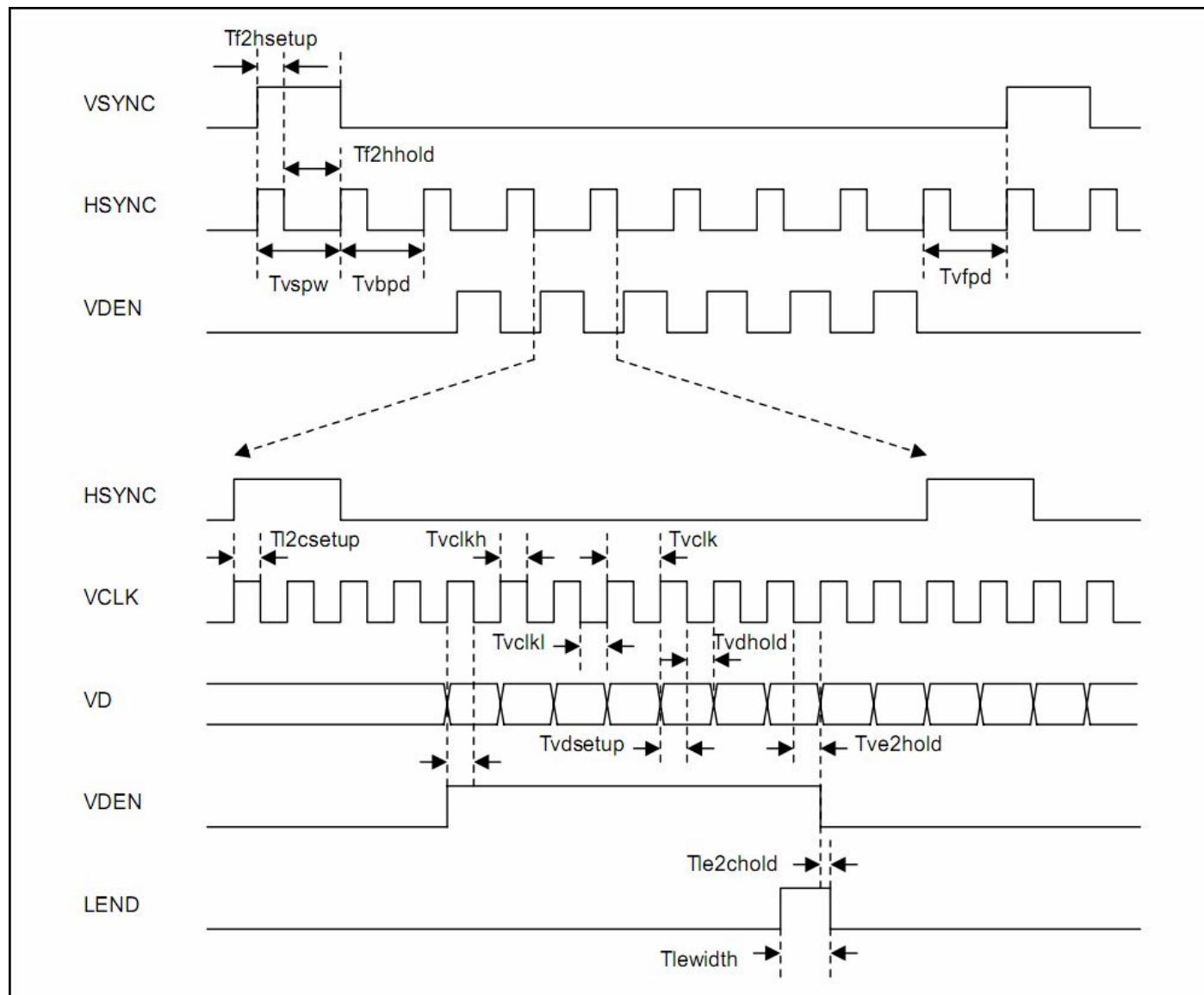


图 27-31. TFT LCD 控制器时序图

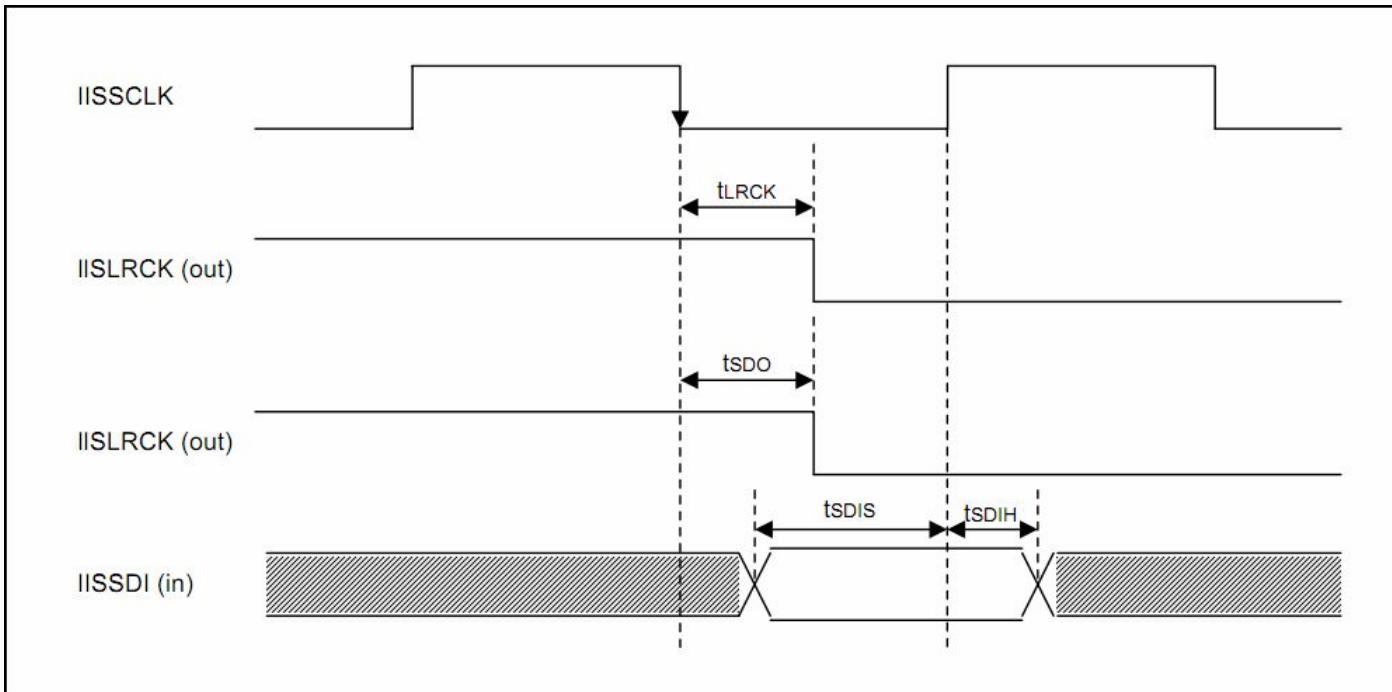


图 27-32. IIS 接口时序图

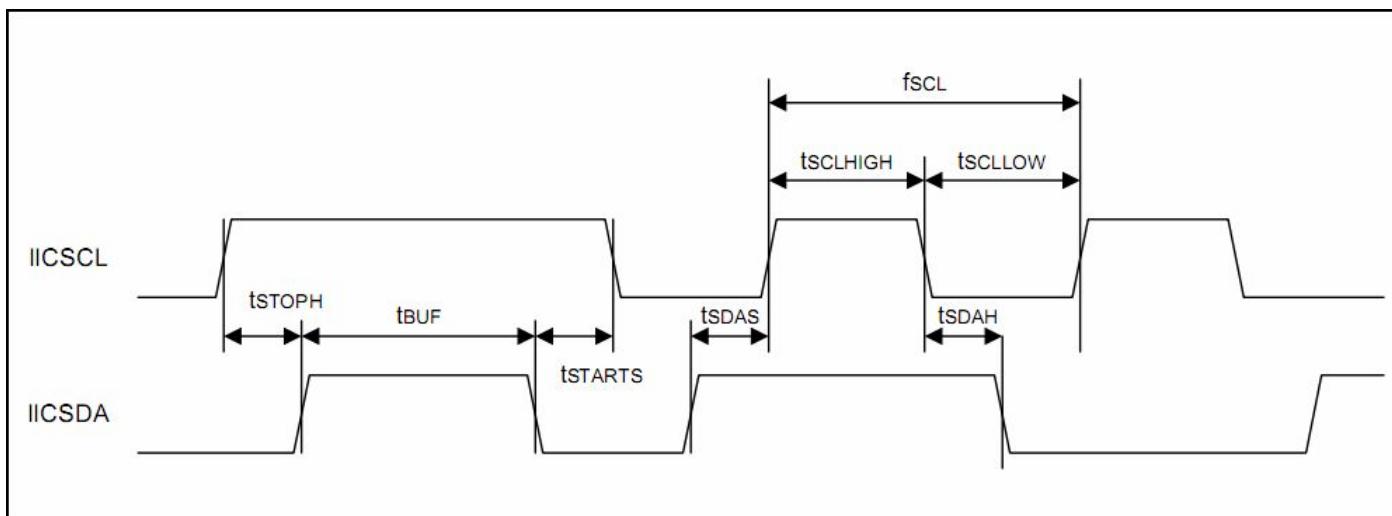


图 27-33. IIC 接口时序图

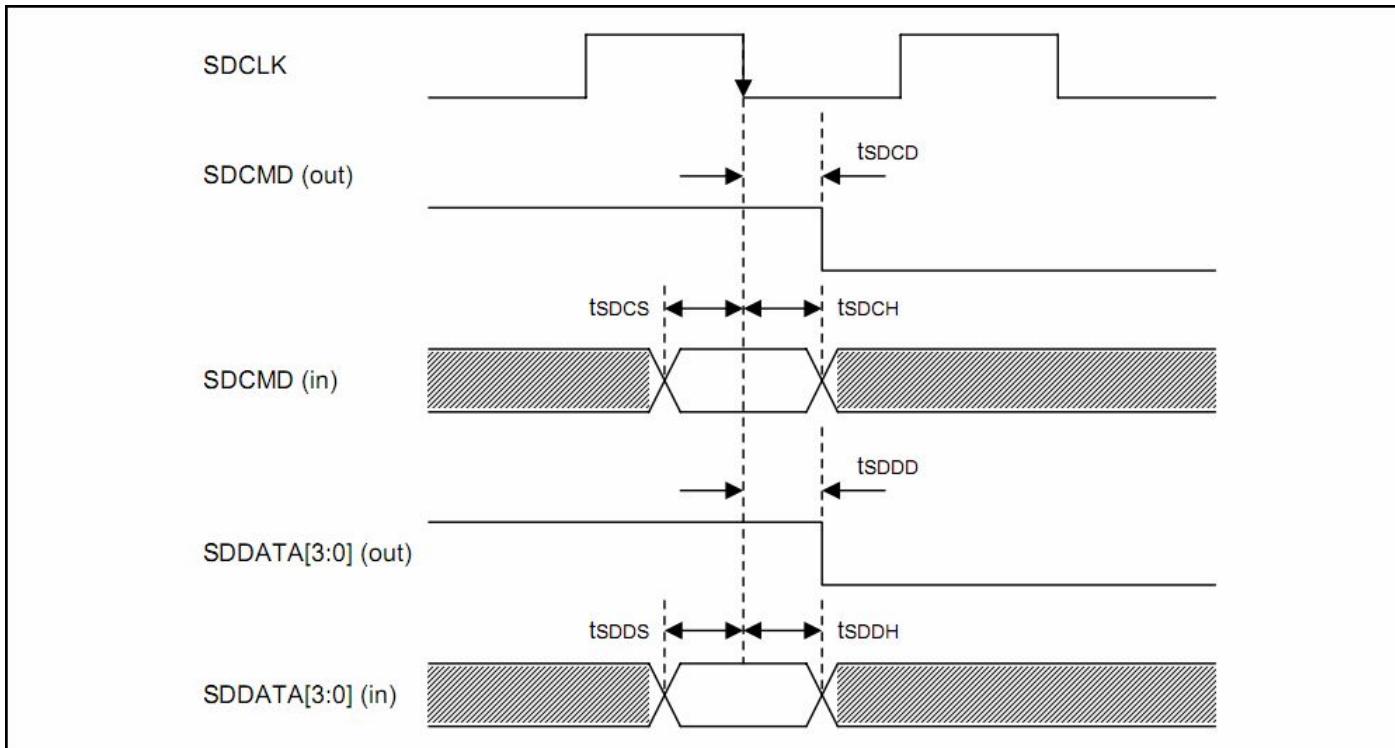


图 27-34. SD/MMC 接口时序图

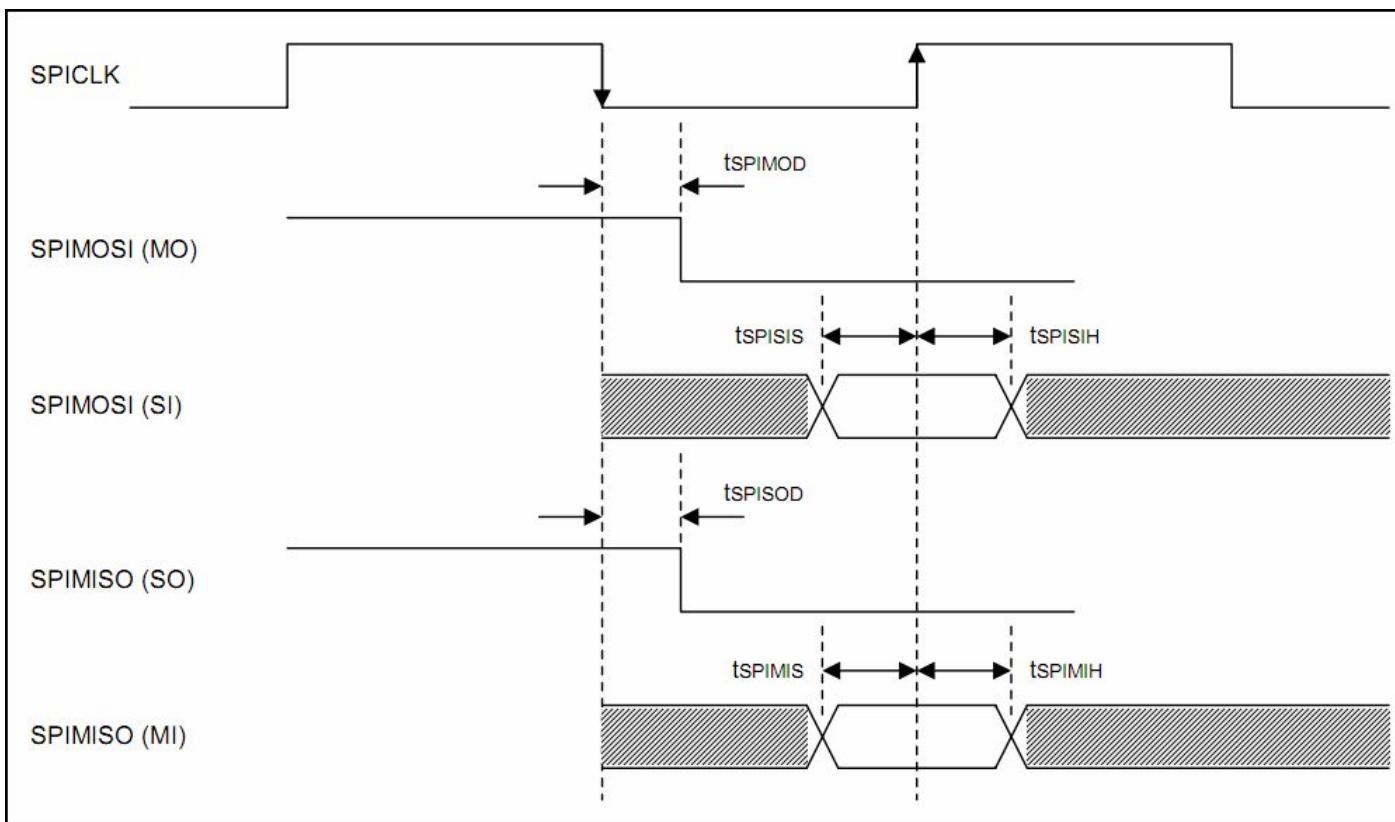


图 27-35. SPI 接口时序图

(CPHA=1 , CPOL=1)

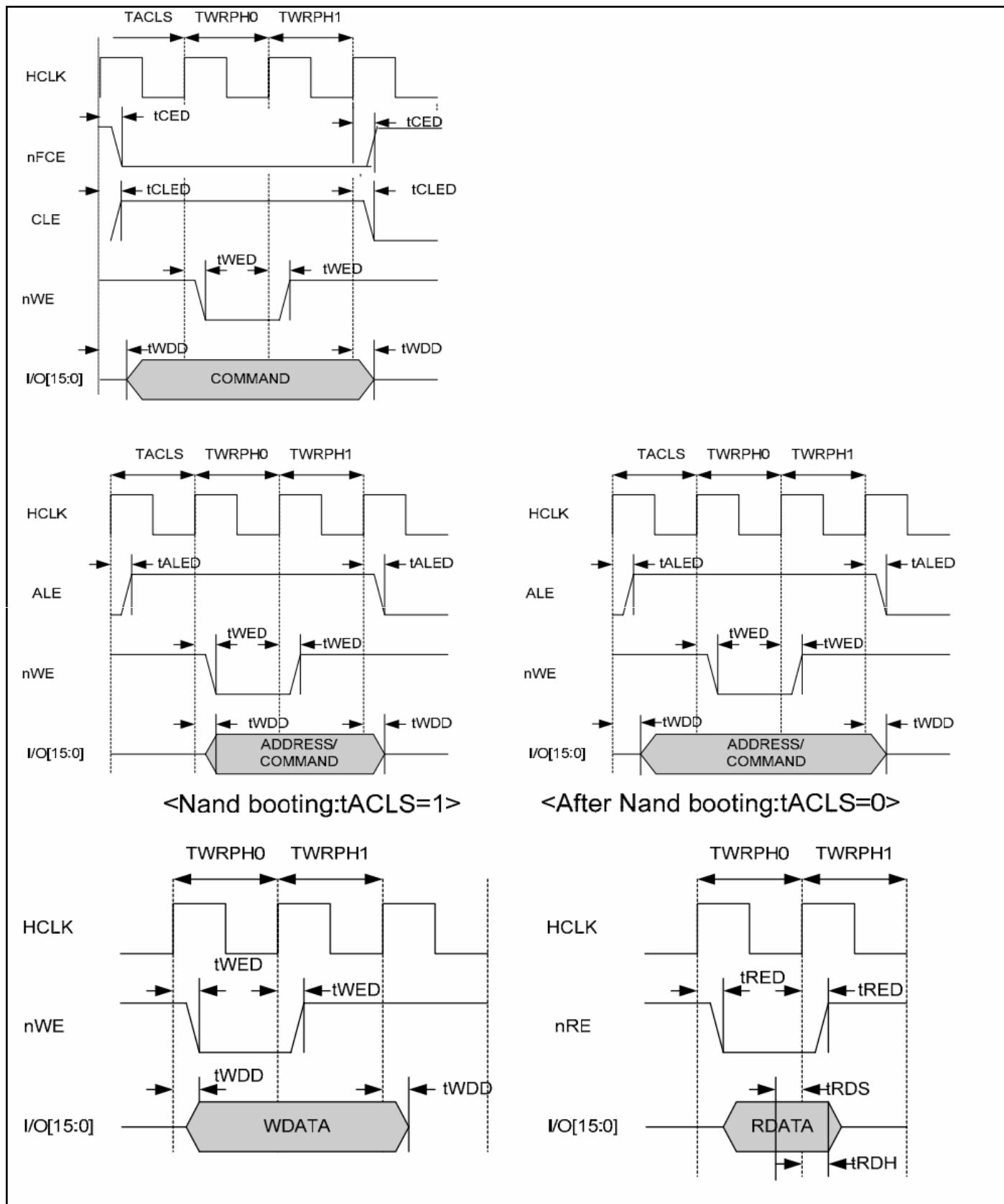


图 27-36 NAND Flash 地址/命令时序图

表 27-7. 时钟时序常数

($V_{DDi}, V_{DDalive}, V_{DDalarm} = 1.2 \text{ V} \pm 0.1 \text{ V}$, $T_A = -40$ 至 85°C , $V_{DDMOP} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
晶振时钟输入频率	f_{XTAL}	12	—	20	MHz
晶振时钟输入周期时间	$t_{XTALCYC}$	50	—	83.3	ns
外部时钟输入频率	f_{EXT}	—	—	66	MHz
外部时钟输入周期时间	t_{EXTCYC}	15.0	—	—	ns
外部时钟输入低电平脉冲宽度	t_{EXTLOW}	7	—	—	ns
外部时钟到 HCLK (无 PLL)	t_{EX2HC}	3	—	7	ns
HCLK (内部) 至 CLKOUT	t_{HC2CK}	3	—	9	ns
HCLK (内部) 至 SCLK	$t_{HC2SCLK}$	1	—	2	ns
外部时钟输入高电平脉冲宽度	$t_{EXTHIGH}$	4	—	—	ns
时钟稳定后复位生效时间	t_{RESW}	10	—	—	XTIpll 或 EXTCLK
PLL 锁定时间	t_{PLL}	300	—	—	μs
睡眠模式返回振荡设置时间	t_{OSC2}	—	—	65536	XTIpll 或 EXTCLK
CPU 运行前至 nRESET 生效后的间隔	$t_{RST2RUN}$	—	7	—	XTIpll 或 EXTCLK

表 27-8. ROM/SRAM 总线时序常数

(V_{DDi} , $V_{DDalive}$, $VDDiarm=1.2\text{ V}\pm0.1\text{ V}$, $T_A=-40$ 至 85°C , $V_{DDMOP}=3.3\text{V}\pm0.3\text{V}/3.0\text{V}\pm0.3\text{V}/2.5\text{V}\pm0.2\text{V}/1.8\text{V}\pm0.1\text{V}$)

参数	符号	最小值 ($V_{DDMOP}=3.3\text{V}/3.0\text{V}/2.5\text{V}/1.8\text{V}$)	典型值	最大值 ($V_{DDMOP}=3.3\text{V}/3.0\text{V}/2.5\text{V}/1.8\text{V}$)	单位
ROM/SRAM 地址延迟	t_{RAD}	2 / 2 / 2 / 3	—	6 / 6 / 7 / 8	ns
ROM/SRAM 片选延迟	t_{RCD}	2 / 2 / 3 / 3	—	6 / 6 / 6 / 7	ns
ROM/SRAM 输出使能延迟	t_{ROD}	2 / 2 / 2 / 3	—	5 / 5 / 5 / 6	ns
ROM/SRAM 读数据建立延迟	t_{RDS}	1 / 1 / 1 / 2	—	— / — / — / —	ns
ROM/SRAM 读数据保持延迟	t_{RDH}	0 / 0 / 0 / 0	—	— / — / — / —	ns
ROM/SRAM 字节使能延迟	t_{RBED}	2 / 2 / 2 / 3	—	5 / 5 / 5 / 7	ns
ROM/SRAM 写字节使能延迟	t_{RWBED}	2 / 2 / 2 / 3	—	5 / 5 / 6 / 7	ns
ROM/SRAM 输出数据延迟	t_{RDD}	2 / 2 / 2 / 2	—	6 / 6 / 6 / 7	ns
ROM/SRAM 外部等待建立时间	t_{WS}	3 / 3 / 4 / 4	—	— / — / — / —	ns
ROM/SRAM 外部等待保持延时间	t_{WH}	0 / 0 / 0 / 0	—	— / — / — / —	ns
ROM/SRAM 等待使能延迟	t_{RWD}	2 / 2 / 2 / 3	—	5 / 5 / 6 / 7	ns

表 27-9. 存储器接口时序常数

(V_{DDi} , $V_{DDalive}$, $VDDiarm=1.2\text{ V}\pm0.1\text{ V}$, $T_A=-40$ 至 85°C , $V_{DDMOP}=3.3\text{V}\pm0.3\text{V}/3.0\text{V}\pm0.3\text{V}/2.5\text{V}\pm0.2\text{V}/1.8\text{V}\pm0.1\text{V}$)

参数	符号	最小值	典型值	最大值	单位
SDRAM 地址延迟	t_{SAD}	1	—	4	ns
SDRAM 片选延迟	t_{SCSD}	1	—	3	ns
SDRAM 行有效延迟	t_{SRD}	1	—	3	ns
SDRAM 列有效延迟	t_{SCD}	1	—	3	ns
SDRAM 字节使能延迟	t_{SBED}	1	—	3	ns
SDRAM 写使能延迟	t_{SWD}	1	—	2	ns
SDRAM 读数据建立时间	t_{SDS}	2 / 3 / 3 / 5 *	—	—	ns
SDRAM 读数据保持时间	t_{SDH}	0	—	—	ns
SDRAM 输出数据延迟	t_{SDD}	1	—	4	ns
SDRAM 时钟使能延迟	T_{cked}	2	—	3	ns

注释：当 $V_{DDMOP} = 3.3\text{V} / 3.0\text{V} / 2.5\text{V} / 1.8\text{V}$ 时，分别最小 $t_{SDS} = 2\text{ns} / 3\text{ns} / 3\text{ns} / 5\text{ ns}$ 。

表 27-10. 外部总线请求时序常数

($V_{DD} = 1.2 \text{ V} \pm 0.1 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
外部总线建立时间	t_{XnBRQS}	4	—	—	ns
外部总线保持时间	t_{XnBRQH}	1	—	—	ns
外部总线应答延迟	$t_{XnBACKD}$	4	—	10	ns
HZ 延迟	t_{HZD}	2	—	6	ns

表 27-11. DMA 控制器模块信号时序常数

($V_{DD} = 1.2 \text{ V} \pm 0.1 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
外部请求建立	t_{XRS}	5	—	—	ns
邻近应答延迟直到变低	t_{CADL}	3	—	8	ns
邻近应答延迟直到变高	t_{CADH}	3	—	8	ns
外部请求延迟	t_{XAD}	2	—	—	SCLK

表 27-12. TFT LCD 控制器模块信号时序常数

($V_{DD} = 1.2 \text{ V} \pm 0.05 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
垂直脉冲宽度	T_{VSPW}	$VSPW + 1$	—	—	$\text{Phclk}^{(1)}$
垂直后沿延迟	T_{VBPD}	$VBPD+1$	—	—	Phclk
垂直前沿延迟	T_{VFPD}	$VFPD+1$	—	—	Phclk
VCLK 脉冲宽度	T_{VCLK}	1	—	—	$\text{Phclk}^{(2)}$
VCLK 脉冲高电平宽度	T_{VCLKH}	0.5	—	—	Phclk
VCLK 脉冲低电平宽度	T_{VCLKL}	0.5	—	—	Phclk
Hsync 建立至 VCLK 下降沿	$T_{I2CSETUP}$	0.5	—	—	Phclk
VDEN 建立至 VCLK 下降沿	$T_{DE2CSETUP}$	0.5	—	—	Phclk
VDEN 保持至 VCLK 下降沿	$T_{DE2CHOLD}$	0.5	—	—	Phclk
VD 建立至 VCLK 下降沿	$T_{VD2CSETUP}$	0.5	—	—	Phclk
VD 保持至 VCLK 下降沿	$T_{VD2CHOLD}$	0.5	—	—	Phclk
VSYNC 建立至 HSYNC 下降沿	$T_{F2HSETUP}$	$HSPW + 1$	—	—	Phclk
VSYNC 保持至 HSYNC 下降沿	$T_{F2HHOLD}$	$HBPD + HFDPD + HOZVAL + 3$	—	—	Phclk

注释：

1. HSYNC 周期
2. VCLK 周期

表 27-13. IIS 控制器模块信号时序常数

($V_{DD} = 1.2 \text{ V} \pm 0.1 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
IISLRCK 延迟时间	t_{LRCK}	0	—	3	ns
IISDO 延迟时间	t_{SDO}	1	—	2	ns
IISDI 输入建立时间	t_{SDIS}	13	—	—	ns
IISDI 输入保持时间	t_{SDIH}	1	—	—	ns
编解码器时钟频率	f_{CODEC}	1/16	—	1	f_{IIS_BLOCK}

表 27-14. IIC 总线控制器模块信号时序常数

($V_{DD} = 1.2 \text{ V} \pm 0.05 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
SCL 时钟频率	f_{SCL}	—	—	标准 100 快速 400	kHz
SCL 高电平脉冲宽度	$t_{SCLHIGH}$	标准 4.0 快速 0.6	—	2	μs
SCL 高电平脉冲宽度	t_{SCLLOW}	标准 4.7 快速 1.3	—	—	μs
起始和停止条件之间的总线空闲时间	t_{BUF}	标准 4.7 快速 1.3	—	—	μs
起始条件保持时间	t_{STARTS}	标准 4.0 快速 0.6	—	—	μs
SDA 保持时间	t_{SDAH}	标准 0 快速 0	—	标准— 快速 0.9	μs
SDA 建立时间	t_{SDAS}	标准 250 快速 100	—	—	ns
停止条件建立时间	t_{STOPH}	标准 4.0 快速 0.6	—	—	μs

注释：

标准意思为标准模式并且快速意思为快速模式。

1. IIC 数据保持时间 (t_{SDAH}) 最小为 0ns。(IIC 数据保持时间对于标准/快速总线模式为最低 0ns 是在 IIC 规范 v2.1 中) 请核对是否 IIC 器件发的数据保持时间为 0ns。
2. IIC 控制器只支持 IIC 总线器件 (标准/快速总线模式), 并不支持 C 总线器件。

表 27-15. SD/MMC 接口发送/接收时序常数

($V_{DD} = 1.2 \text{ V} \pm 0.1 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
SD 命令输出延迟时间	t_{SDCD}	0	—	1	ns
SD 命令输入建立时间	t_{SDCS}	14	—	—	ns
SD 命令输入保持时间	t_{SDCH}	1	—	—	ns
SD 数据输出延迟时间	t_{SDDD}	0	—	1	ns
SD 数据输入建立时间	t_{SDDS}	13	—	—	ns
SD 数据输入保持时间	t_{SDDH}	1	—	—	ns

表 27-16. SPI 接口发送/接收时序常数

($V_{DD} = 1.2 \text{ V} \pm 0.1 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	最小值	典型值	最大值	单位
SPI MOSI 主机输出延迟时间	t_{SPIMOD}	0	-	1	ns
SPI MOSI 从机建立时间	t_{SPISIS}	1	-	-	ns
SPI MOSI 从机保持时间	t_{SPISIH}	1	-	-	ns
SPI MISO 从机输出延迟时间	t_{SPISOD}	7	-	17	ns
SPI MISO 主机建立时间	t_{SPIMIS}	15	-	-	ns
SPI MISO 主机保持时间	t_{SPIMIH}	1	-	-	ns

表 27-17. USB 电气规范

($V_{DD} = 1.2 \text{ V} \pm 0.05 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	条件	最小值	最大值	单位
供电电流					
挂起设备	ICCS			10	μA
漏电电流					
高阻态输入漏电	ILO	$0\text{V} < V_{IN} < 3.3\text{V}$	-10	10	μA
输入电平					
差分输入灵敏度	VDI	$ (D+) - (D-) $	0.2		V
差分共模范围	VCM	包括 VDI 范围	0.8	2.5	
单端接收门限	VSE		0.8	2.0	
输出电平					
静态输出低	VOL	$1.5\text{k}\Omega$ 的 RL 到 3.6V		0.2	V
静态输出高	VOH	$1.5\text{k}\Omega$ 的 RL 到 GND	2.8	3.6	
电容					
收发器电容	CIN	引脚到 GND		20	pF

表 27-18. USB 全速输出缓冲电气规范

($V_{DD} = 1.2 \text{ V} \pm 0.05 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	条件	最小值	最大值	单位
设备特性					
翻转时间					
上升时间	TR	$CL = 50\text{pF}$	4.0	20	ns
下降时间	TF	$CL = 50\text{pF}$	4.0	20	
上升/下降时间匹配	Trfm	(TR / TF)	90	111.1	%
输出信号转换电压	Vcrs		1.3	2.0	V
设备输出阻抗	Zdrv	稳态驱动	28	44	Ω

表 27-19. USB 低速输出缓冲电气规范

($V_{DD} = 1.2 \text{ V} \pm 0.05 \text{ V}$, $T_A = -40$ 至 85°C , $V_{EXT} = 3.3\text{V} \pm 0.3\text{V}$)

参数	符号	条件	最小值	最大值	单位
设备特性					
上升时间	TR	$CL = 50\text{pF}$ $CL = 350\text{pF}$	75		ns
				300	
下降时间	TF	$CL = 50\text{pF}$ $CL = 350\text{pF}$	75		
				300	
上升/下降时间匹配	Trfm		80	125	%
输出信号转换电压	Vcrs	(Tr / Tf)	1.3	2.0	V

注释：

所有测量条件都是依照通用串行总线规范 1.1 最终修订草案。

表 27-20. NAND Flash 接口时序常数

(V_{DDi} , $V_{DDalive}$, $VDDiarm=1.2\text{ V}\pm0.1\text{ V}$, $T_A=-40$ 至 85°C , $V_{DDMOP}=3.3\text{V}\pm0.3\text{V}/3.0\text{V}\pm0.3\text{V}/2.5\text{V}\pm0.2\text{V}/1.8\text{V}\pm0.1\text{V}$)

参数	符号	最小值 ($V_{DDMOP}=3.3\text{V}/3.0\text{V}/2.5\text{V}/1.8\text{V}$)	典型值	最大值 ($V_{DDMOP}=3.3\text{V}/3.0\text{V}/2.5\text{V}/1.8\text{V}$)	单位
NFCON 片选使能延迟	t_{CED}	- / - / - / -	-	5.4/5.6/5.9/7.1	ns
NFCON CLE 延迟	t_{CLED}	- / - / - / -		5.3/5.5/5.8/7.0	ns
NFCON ALE 延迟	t_{ALED}	- / - / - / -		5.4/5.6/5.9/7.1	ns
NFCON 写使能延迟	t_{WED}	- / - / - / -		5.0/5.2/5.5/6.7	ns
NFCON 读使能延迟	t_{RED}	- / - / - / -		5.0/5.2/5.5/6.7	ns
NFCON 写数据建立时间	t_{WDS}	5.8/6.0/6.3/7.5		- / - / - / -	ns
NFCON 写数据保持时间	t_{WDH}	4.6/4.8/5.1/6.3		- / - / - / -	ns
NFCON 读数据建立请求时间	t_{RDS}	3/3.1/3.3/4		- / - / - / -	ns
NFCON 读数据保持请求时间	t_{RDH}	0.3/0.3/0.3/0.3		- / - / - / -	ns