AN5213

PCI1xxxx的配置和编程选项

作者： *Andrew Rogers、Josh Averyt和Sukanya Palanisami*
Microchip Technology Inc.

1.0 简介

Microchip PCI1XXXX是高度可定制的PCI Express®（PCIe®）交换芯片系列，集成了多种PCI Express端点器件，包括USB 3.2 10 Gbps主机、以太网MAC以及具有UART、I²C、SPI和GPIO外设的多功能器件。该器件系列包括PCI12000、PCI11010、PCI11101、PCI11400和PCI11414。

PCI1XXXX器件可通过以下方式进行配置：

- 外部EEPROM存储器
- 一次性编程（One Time Programmable，OTP）存储器
- I²C或SPI配置接口
- 通过PCIe主机进行运行时配置

1.1 章节

本应用笔记包含以下章节：

1.2 参考资料

使用本应用笔记时，应参考以下文档：

- 《PCI12000数据手册》（https://www.microchip.com.cn/newcommunity/Uploads/202405/6642dd6c1e178.pdf）
- *PCI11010 Data Sheet*（www.microchip.com/DS00003793）
- *PCI11101 Data Sheet*（www.microchip.com/DS00003791）
- 《PCI11400数据手册》（https://www.microchip.com.cn/newcommunity/Uploads/202504/68119a3b35f25.pdf）
- 《PCI11414数据手册》（https://www.microchip.com.cn/newcommunity/Uploads/202405/6642dd4de5d99.pdf）
- *PCI Express Base r3.1a*（可从PCI-SIG获取）
- *I²C规范*
- *SPI规范*
- *System Management Bus Specification*，版本1.0（http://smbus.org/specs）

2.0 器件编程和配置汇总

2.1 章节

- 第2.2节“配置选项”
- 第2.3节“必要配置一览”

PCI1xxxx系列器件具备多种特性和功能，如表1所示。

表1： 器件属性和编程选项

器件编号	封装	PCIe® 上行端口	PCIe 下行端口	USB2 PHY	USB3 PHY	以太网 MAC	UART 接口（注2）	I2C 主器件	SPI 主器件	GPIO（最大数量）
PCI12000	72 QFN	x2	2个x1	0	0	0	0	1	2	18
PCI11010	100 QFN	x2	1个x1	0	0	1	0	1	2	46
PCI11101	132 DQFN	x4	1个x2	1	1	0	1	1	2	48
PCI11400	132 DQFN	x4	1个x1	4	2	0	0	1	2	38
PCI11414	164 DQFN	x4	1个x1	4	2	1	4	1	2	64

注 1： 若要实现上述支持的最大GPIO数量，必须禁止多个主要器件接口，以便将相关的PROGx引脚重新配置为GPIO。

2： 默认配置。每个SKU最多可配置4个UART。

2.2 配置选项

PCI1xxxx系列器件具备多种特性和功能，很多需要经过配置后才能使用。大部分配置必须通过EEPROM、OTP存储器或SPI/I^2C完成。尽管器件支持同时通过这三种途径进行配置，但建议仅选择其中一种，以便简化和集中配置数据，请参见表2。

表2： 配置方法

配置方法	注
硬件配置脚选项	硬件配置脚仅限于少数关键项，必须在原理图设计阶段选择和处理所有设计。
EEPROM	EEPROM器件可在产品生命周期内重新编程，但会增加系统总BOM成本。
OTP存储器	OTP存储器无需额外成本，但未来升级能力受限。
SPI/I^2C配置	如果系统搭载MCU/SoC，则SPI/I^2C配置可能是首选方案。该方法还可以动态重新配置特定功能，而EEPROM和OTP方法无法实现这一点。
通过I^2C/SPI/PCIe®在运行时重新配置	可在运行时修改特定配置项。该方法需要应用层软件，许多配置项无法通过这种方式进行控制。

2.3 必要配置一览

许多功能需要完成一些基础配置才能正常运行。其中包括：

- **可编程功能引脚：**大多数可编程功能引脚默认用作GPIO。但是，大多数USB/以太网/外设功能需要I/O支持，这些I/O必须被使能并复用到最合适的PROGx引脚。
- **PCIe交换芯片：**默认配置可能已满足需求，但如果有任何PCIe端口未使用，应通过配置将其禁止。
- **USB端口：**USB端口应根据速度和连接类型（如Type-C®、Type-A或嵌入式设备）进行配置。
- **以太网MAC：**以太网MAC必须根据所选PHY、系统MAC地址以及任何特殊的速度或工作模式要求进行配置。
- **GPIO、UART、I^2C 和 SPI 控制器：**桥接控制器通常需要完成基础配置以符合预期操作。配置选项包括引脚复用选择、选择所需端口数以及引脚焊盘选项等。

3.0 常见问答

1. 是否必须进行EEPROM/OTP编程？

非必要。如果通过MCU/SOC使用I^2C或SPI处理配置，则无需EEPROM/OTP编程。

PCI1xxxx在设计上必须经过一定的配置才能实现正常工作模式。默认设置无法实现正常工作模式。

2. 是否提供预编程的EEPROM器件？

不提供。Microchip不提供预编程的EEPROM器件。

3. PCI1xxx是否执行固件？

不执行。PCI1xxxx不包含处理器，不执行固件。

4. 未使用的外设——是否可以以及是否应该禁止？

可以在配置数据中禁止未使用的外设。如果最终硬件设计中有任何未使用的外设，建议禁止这些功能，原因如下：

- 防止外设出现在主机计算机中
- 最大限度降低功耗
- 防止由于外设硬件引脚“悬空”引发意外行为

注： 请勿禁止GPIO子系统，因为该接口也用于编程OTP或EEPROM配置。

5. 是否可以修改或完全禁止USB主机端口速度？

若要禁止给定端口的USB 3.1接口，请修改以下两个配置参数：

- USB_PORT_ENABLE_REG.HOST_NUM_U3_PORT[3:0]
- USB_PORT_ENABLE_REG.USB_U3_PORT_ENABLE[3:0]

若要禁止给定端口的USB 2.0接口，请修改以下两个配置参数：

- USB_PORT_ENABLE_REG.HOST_NUM_U2_PORT[3:0]
- USB_PORT_ENABLE_REG.USB_U2_PORT_ENABLE[3:0]

6. 是否可以设置固定的以太网速度？

可以。如需特定的以太网速度，可将其设置为固定值。如果连接到以其他速度运行的网络，将导致连接失败，因此仅建议在特定场景或测试目的下使用该设置。

7. MAC地址是否已预编程？

否。MAC地址默认未经过预编程，终端系统集成商负责通过正规渠道申请MAC地址并对其进行编程。Microchip提供编程工具以简化和自动化编程过程。

若要设置MAC地址，请编程以下寄存器：

- MAC_RX_ADDRH
- MAC_RX_ADDRL

8. 主机计算机是否可以改写以太网MAC地址？

可以。主机计算机可通过运行时重新配置过程改写MAC地址。只需将新值写入MAC地址寄存器即可改写。

9. 如何为产品获取MAC地址段？

通常需向IEEE购买MAC地址段。更多详细信息，请访问IEEE注册管理机构官网（https://standards.ieee.org/products-programs/regauth/）。

4.0 引导/配置序列

在配置阶段，引脚复用功能将根据系统的PCB设计进行配置，各功能将按照系统要求进行配置。初始化配置可选择由外部配置源（SMBus/I^{2}C或SPI）提供。由于IC封装限制，PCI1xxxx包含的引脚功能数量超过了可同时支持的数量，因此必须明确规划引脚复用功能，并谨慎选择在系统中实现的并发功能。

必须通过配置脚和随后的引脚复用操作指明预期的外部配置源，以便暴露正确的配置接口。

4.1 章节

4.2 配置序列

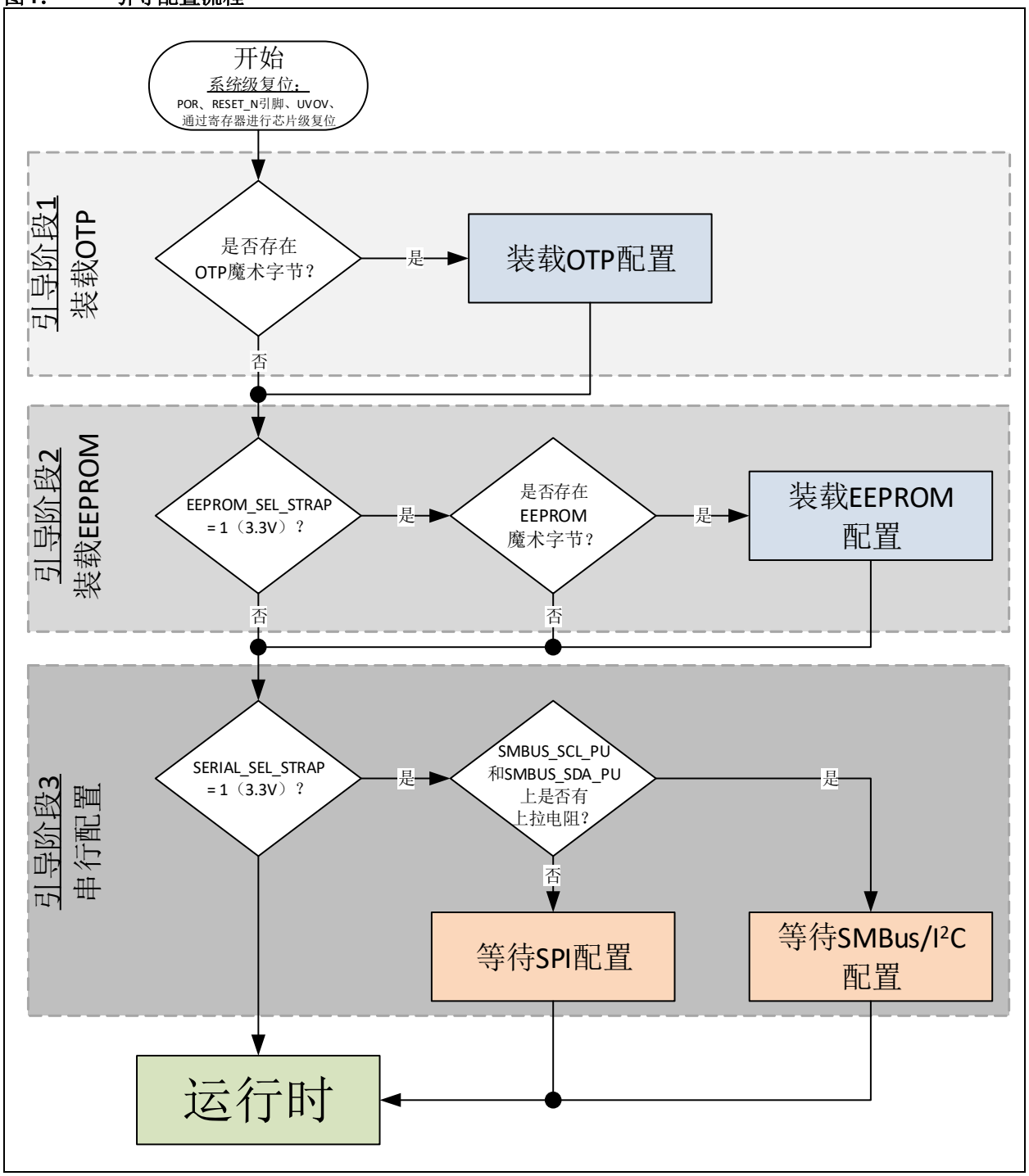
必须按顺序装载配置，以便正常执行初始化配置流程，在此期间寄存器中的内容会被替换为每个配置源的内容（如果改写）。有关配置流程，请参见图1。

支持的配置序列包括：

1. 硬件默认配置 > OTP配置
2. 硬件默认配置 > OTP配置 > EEPROM配置
3. 硬件默认配置 > OTP配置 > EEPROM配置 > SPI/I^{2}C配置
4. 硬件默认配置 > OTP配置 > SPI/I^{2}C配置
5. 硬件默认配置 > EEPROM配置
6. 硬件默认配置 > EEPROM配置 > SPI/I^{2}C配置
7. 硬件默认配置 > SPI/I^{2}C配置

完成初始化配置后，PCIe交换芯片即可在PCIe总线上完成枚举。建立PCIe连接后，配置即完成，此时PCI1xxxx进入运行时工作模式。

图1：　　引导配置流程



4.3　　引导阶段1：装载OTP存储器

引导时，PCI1xxxx会检查OTP存储单元0000h处的“魔术字节”（值为A5h）。如果读取值不是A5h，则判定OTP存储器无效，PCI1xxxx将进入引导阶段2。

如果值为A5h，则将从头到尾按顺序装载整个OTP存储器。有关OTP存储器装载过程，请参见第14.2节“引导时装载OTP存储器”。

有关EEPROM确认的更多详细信息，请参见第13.0章“配置文件格式”。

4.4 引导阶段2：装载EEPROM存储器

在引导阶段2中，会检查EEPROM_SEL_STRAP硬件电阻配置脚选项。这是一个简单的数字输入引脚。如果检测到电压为低电平（低于引脚的VIL规范），则PCI1xxxx将进入引导阶段3。

如果检测到EEPROM_SEL_STRAP上的电压为高电平（高于引脚的VIH规范），则PCI1xxxx将作为SMBus/I^2C控制器尝试读取连接的EEPROM器件。首先检查EEPROM存储单元0000h处的“魔术字节”（值为A5h）。如果从“魔术字节”存储单元读回值A5h，则将从头到尾按顺序装载整个EEPROM存储器。有关EEPROM存储器装载过程，请参见第15.2节“引导时的EEPROM存储器装载”。

EEPROM存储器中的配置数据与OTP存储器中的配置数据遵循相同的命令格式。

注： 如果某寄存器同时在OTP和EEPROM存储器中配置，则将以EEPROM配置数据中设置的值为准。

有关EEPROM确认的更多详细信息，请参见第15.0章“EEPROM存储器编程（读写操作）”。

4.5 引导阶段3：串行配置

在引导阶段3中，会检查SERIAL_SEL_STRAP硬件电阻配置脚选项。这是一个简单的数字输入引脚。如果检测到电压为低电平（低于引脚的VIL规范），则PCI1xxxx将直接进入运行时模式，并可将其上行PCIe接口连接到PCIe主机。

如果检测到SERIAL_SEL_STRAP上的电压为高电平（高于引脚的VIH规范），则PCI1xxxx将检查SMBUS_SCL_PU和SMBUS_SDA_PU硬件配置脚。

如果检测到SMBUS_SCL_PU和SMBUS_SDA_PU这两个引脚上的电压都为高电平（高于引脚的VIH规范），则使能SMBus/I^2C配置接口。否则，使能SPI配置接口。PIC1xxxx将在配置阶段无限期等待，直到EXT_SYS_CONFIG_DONE_REG中的配置完成位置1为止。

注： 如果某寄存器同时通过OTP和/或EEPROM以及串行配置进行配置，则以串行配置期间设置的值为准。

有关EEPROM确认的更多详细信息，请参见第16.0章“SMBus/I2C目标配置”和第17.0章“SPI配置”。

4.6 运行时

在运行时，可通过常规PCI存储器写操作来修改任何寄存器（可根据设计人员的喜好选择工具）。但是，为确保PCI1XXXX稳定工作，建议使用适用于所有标准用例的驱动程序和API。只能在测试和调试阶段直接操作PCI寄存器。

用户可参见*AN4255 PCI12000/PCI11XXXX Register Map*中的寄存器映射。

5.0 可编程引脚复用配置

5.1 可编程引脚复用

每个可编程引脚都有多个可选功能选项，可根据最终应用用例进行配置。产品数据手册包含可编程引脚及其默认功能选项的列表。

如需更改默认配置，必须修改相应的可编程功能寄存器。有关每个配置寄存器的偏移地址，请参见表3。

表3： 引脚复用配置

地址	寄存器	地址	寄存器	地址	寄存器
0x0024_0400	PF0_CTL_REG	0x0024_0480	PF32_CTL_REG	0x0024_0500	PF64_CTL_REG
0x0024_0404	PF1_CTL_REG	0x0024_0484	PF33_CTL_REG	0x0024_0504	PF65_CTL_REG
0x0024_0408	PF2_CTL_REG	0x0024_0488	PF34_CTL_REG	0x0024_0508	PF66_CTL_REG
0x0024_040C	PF3_CTL_REG	0x0024_048C	PF35_CTL_REG	0x0024_050C	PF67_CTL_REG
0x0024_0410	PF4_CTL_REG	0x0024_0490	PF36_CTL_REG	0x0024_0510	PF68_CTL_REG
0x0024_0414	PF5_CTL_REG	0x0024_0494	PF37_CTL_REG	0x0024_0514	PF69_CTL_REG
0x0024_0418	PF6_CTL_REG	0x0024_0498	PF38_CTL_REG	0x0024_0518	PF70_CTL_REG
0x0024_041C	PF7_CTL_REG	0x0024_049C	PF39_CTL_REG	0x0024_051C	PF71_CTL_REG
0x0024_0420	PF8_CTL_REG	0x0024_04A0	PF40_CTL_REG	0x0024_0520	PF72_CTL_REG
0x0024_0424	PF9_CTL_REG	0x0024_04A4	PF41_CTL_REG	0x0024_0524	PF73_CTL_REG
0x0024_0428	PF10_CTL_REG	0x0024_04A8	PF42_CTL_REG	0x0024_0528	PF74_CTL_REG
0x0024_042C	PF11_CTL_REG	0x0024_04AC	PF43_CTL_REG	0x0024_052C	PF75_CTL_REG
0x0024_0430	PF12_CTL_REG	0x0024_04B0	PF44_CTL_REG	0x0024_0530	PF76_CTL_REG
0x0024_0434	PF13_CTL_REG	0x0024_04B4	PF45_CTL_REG	0x0024_0534	PF77_CTL_REG
0x0024_0438	PF14_CTL_REG	0x0024_04B8	PF46_CTL_REG	0x0024_0538	PF78_CTL_REG
0x0024_043C	PF15_CTL_REG	0x0024_04BC	PF47_CTL_REG	0x0024_053C	PF79_CTL_REG
0x0024_0440	PF16_CTL_REG	0x0024_04C0	PF48_CTL_REG	0x0024_0540	PF80_CTL_REG
0x0024_0444	PF17_CTL_REG	0x0024_04C4	PF49_CTL_REG	0x0024_0544	PF81_CTL_REG
0x0024_0448	PF18_CTL_REG	0x0024_04C8	PF50_CTL_REG	0x0024_0548	PF82_CTL_REG
0x0024_044C	PF19_CTL_REG	0x0024_04CC	PF51_CTL_REG	0x0024_054C	PF83_CTL_REG
0x0024_0450	PF20_CTL_REG	0x0024_04D0	PF52_CTL_REG	0x0024_0550	PF84_CTL_REG
0x0024_0454	PF21_CTL_REG	0x0024_04D4	PF53_CTL_REG	0x0024_0554	PF85_CTL_REG
0x0024_0458	PF22_CTL_REG	0x0024_04D8	PF54_CTL_REG	0x0024_0558	PF86_CTL_REG
0x0024_045C	PF23_CTL_REG	0x0024_04DC	PF55_CTL_REG	0x0024_055C	PF87_CTL_REG
0x0024_0460	PF24_CTL_REG	0x0024_04E0	PF56_CTL_REG	0x0024_0560	PF88_CTL_REG
0x0024_0464	PF25_CTL_REG	0x0024_04E4	PF57_CTL_REG	0x0024_0564	PF89_CTL_REG
0x0024_0468	PF26_CTL_REG	0x0024_04E8	PF58_CTL_REG	0x0024_0568	PF90_CTL_REG
0x0024_046C	PF27_CTL_REG	0x0024_04EC	PF59_CTL_REG	0x0024_056C	PF91_CTL_REG
0x0024_0470	PF28_CTL_REG	0x0024_04F0	PF60_CTL_REG	0x0024_0570	PF92_CTL_REG
0x0024_0474	PF29_CTL_REG	0x0024_04F4	PF61_CTL_REG	0x0024_057C 至 0x0024_05FF	保留
0x0024_0478	PF30_CTL_REG	0x0024_04F8	PF62_CTL_REG		
0x0024_047C	PF31_CTL_REG	0x0024_04FC	PF63_CTL_REG		

每个可编程引脚寄存器都具有标准位映射，如表4所示。

表4：　　编程功能寄存器格式

PFX_CTL_REG		
Bit	**R/W**	**说明**
31:17	R	保留
16	R/W	选择RGMII/MII焊盘模式： 0b：GMII 1b：RGMII **注：**该寄存器仅在PAD_TYPE = 1b时生效。当PAD_TYPE = 0b时，该位为0b且为只读/保留。当SELECT[3:0]设置为FUNC1且PAD_TYPE = 1b时，该位域为只读并由内部硬件控制。
15:14	R	保留
12:13	R/W	选择引脚用作通用I/O时的驱动能力。 00b：2 mA 01b：4 mA 10b：8 mA 11b：10 mA **注：**该寄存器仅在PAD_TYPE = 0b时生效。当PAD_TYPE = 1b时，该位域为00b且为只读/保留。
11:9	R	保留
8	R	指示引脚焊盘的类型。 0b——通用3.3V I/O 1b——GMII/RGMII
7:4	R	保留
3:0	R/W	0000b——FUNC0——GPIO 0001b——FUNC1硬件定义功能 0010b——FUNC2硬件定义功能 0011b——FUNC3硬件定义功能 0100b——FUNC4硬件定义功能 0101b——FUNC5硬件定义功能 0110b——FUNC6硬件定义功能 0111b——FUNC7硬件定义功能 1000b——FUNC8硬件定义功能 1001b——FUNC9硬件定义功能 1010b——FUNC10硬件定义功能 1011b——FUNC11硬件定义功能 1100b——FUNC12硬件定义功能 1101b——FUNC13硬件定义功能 1110b——FUNC14硬件定义功能 1111b——FUNC15硬件定义功能 **注：**有关硬件定义功能，请参见相应产品数据手册中的可编程功能引脚映射表。

6.0 PCIe交换芯片实现

集成的PCIe交换芯片配置如表5所示。

表5： PCIe®交换芯片配置

端口编号	方向	通道数	比特率	说明和连接	器件支持
0	上行	2/4	8 GTps	上行端口	PCI12000和PCI11010 2个，其余型号4个
1	下行	1/2	8 GTps	用于连接外部器件的下行PCIe®端口	PCI11101 2个，其余型号1个
2	下行	4	8 GTps	用于连接外部器件的下行PCIe端口或USB 3.2 Gen 2主机控制器	PCI11101、PCI11400和PCI11414
3	下行	1	5 GTps	以太网控制器	PCI11010和PCI11414
4	下行	1	2.5 GTps	多功能器件端点（GPIO/SMBus/SPI/UART）	全部

PCI交换芯片ID如下：

- 供应商ID：0x1055（Microchip Technology, Inc/SMSC）
- 器件ID：
  - PCI12000：0xA008（端口0）、0xA009（端口1）、0xA00A（端口2）、0xA00B（端口2）和0xA00C（端口4）
  - PCI11010：0xA018（端口0）、0xA019（端口1）、0xA01A（端口2）、0xA01B（端口2）和0xA01C（端口4）
  - PCI11101：0xA028（端口0）、0xA029（端口1）、0xA02A（端口2）、0xA02B（端口2）和0xA02C（端口4）
  - PCI11400：0xA038（端口0）、0xA039（端口1）、0xA03A（端口2）、0xA03B（端口2）和0xA03C（端口4）
  - PCI11414：0xA048（端口0）、0xA049（端口1）、0xA04A（端口2）、0xA04B（端口2）和0xA04C（端口4）

6.1 PCIe交换芯片子系统寄存器

表6列出了PCIE_SWITCH_ADDR_BASE = 0x001C_0000时可用的寄存器。

小心： PCI1xxxx器件包含数千个寄存器，其中一部分寄存器保留供将来使用，另一部分寄存器的内容未纳入公开文档，因为其不涉及终端系统集成商用例（即：这些寄存器供硬件用于执行底层运行时任务，或在运行时由驱动程序直接使用/控制）。**请勿修改未纳入文档的寄存器的内容。**

表6： USB子系统寄存器地址相对于PCIE_SWITCH_ADDR_BASE的偏移量

寄存器名称	端口0	端口1	端口2	端口3	端口4
PCIE_SW_CONFIG_REG	0x0_0000				
PCIE_SW_CTL_REG	0x0_0004				
PCIE_SW_TESTIN_REG	0x0_0020				
PCIE_SW_PORT_HP_CAP_REG	n/a	0x1_1020	n/a	n/a	n/a
PCIE_SW_PORT_CTL_REG	0x1_0024	0x1_1024	0x1_2024	0x1_3024	0x1_4024
PCIE_SW_PORT_CFG_VID_REG	0x1_0040	0x1_1040	0x1_2040	0x1_3040	0x1_4040
PCIE_SW_PORT_CFG_DID_REG	0x1_0044	0x1_1044	0x1_2044	0x1_3044	0x1_4044
PCIE_SW_PORT_DBG_REG	0x1_0150	0x1_1150	0x1_2150	0x1_3150	0x1_4150
PCIE_SW_PORT_PCI_CONFIG_SPACE	0x1_0800	0x1_1800	0x1_2800	0x1_3800	0x1_4800

表7列出了PCIe交换芯片配置寄存器的内容。

表7： PCIE_SW_CONFIG_REG

PCIe®交换芯片配置寄存器			默认值： PCI12000：0x0000_0000 PCI11414/PCI11400/PCI11101/PCI11010：0x0000_0004
Bit	**名称**	**R/W**	**说明**
31:6	Reserved	R	始终为0
5	PCIE_SW_ EXT_OBFF_WIRE_OUT_EN	R/W	将来自端口1的pl1_wake_oen唤醒输出的WAKE#信号输出连接到PCIE_WAKE_N引脚，以使交换芯片能够通过线路向下行PCI器件发送OBFF信号： 0b：禁止外部OBFF输出。 1b：使能外部OBFF输出。
4	PCIE_SW_EXT_WAKE_EN	R/W	将来自交换芯片端口1的pl1_wake_in唤醒输入的WAKE#信号输入连接到PCIE_WAKE_N引脚，以使下行PCI器件能够通过线路发送WAKE#信号： 0b：禁止外部WAKE#输入。 1b：使能外部WAKE#输入。
3	Reserved	R	始终为0
2	PCIE_SW_P2_XHCI_CONN	R/W	将交换芯片的端口2连接到xHCI PCIe端点： 0b：端口2连接到外部PCIe端口。 1b：端口2连接到xHCI PCIe端点。 **注：**请勿修改该寄存器。
1	PCIE_SW_OBFF_WIRE_IN_EN	R/W	将来自PCIE_WAKE_N引脚的唤醒信号输出连接到PCIe端点的唤醒输入，以使能下行OBFF信号传输： 0b：禁止OBFF输入。 1b：使能OBFF输入。
0	PCIE_SW_BEACON_EN	R/W	将来自PCIe端点的唤醒信号输出连接到交换芯片下行端口的plx_wake_oen输入，以使能唤醒信号的上行信标： 0b：禁止信标。 1b：使能信标。

表8列出了PCIe交换芯片CTL寄存器的内容。

表8： PCIE_SW_CTL_REG

PCIe®交换芯片配置寄存器			默认值： 取决于器件
Bit	**名称**	**R/W**	**说明**
31:5	Reserved	R	始终为0
4:0	PCIE_SW_PORT_ENABLE[3:0]	R/W	由硬件设置的默认值，用于指示是否已禁止PCIe交换芯片的端口，其中bit 0对应端口1，bit 1对应端口2（无法禁止端口0），以此类推： 0b：禁止端口 1b：使能端口 **注：**PCIE_SW_PORT_ENABLE[3:0]的默认值由硬件根据器件编号自动设置，但可在系统启动前通过 OTP/EEPROM/SMBus/SPI重新配置进行改写。

表9列出了PCIe交换芯片测试寄存器的内容。

表9： PCIE_SW_TESTIN_REG

PCIe®交换芯片测试寄存器			默认值： 0x0000_0018
Bit	**名称**	**R/W**	**说明**
31:22	Reserved	R	始终读为0
21	PCIE_SW_TESTIN_DIS_ PARITY_CHK R/	R/W	禁止128b/130b SKP OS奇偶校验和报告。
20	PCIE_SW_TESTIN_EN_ DC_BAL_WARN	R/W	使能在训练集中接收到不正确的直流均衡符号时发出警告。
19	Masked	R	已屏蔽。仅供内部使用。
18:15	Reserved	R	始终读为0
14	PCIE_SW_TESTIN_FILTER_ COMP_EXIT	R/W	在速率变化期间滤除Polling.Compliance中的电气空闲退出。
13	PCIE_SW_TESTIN_DIS_EQ_REP	R/W	禁止报告均衡问题。
12	Reserved	R	始终读为0
11	PCIE_SW_TESTIN_FORCE_ RX_DET	R/W	强制从Polling.active进入Polling.Compliance。
10	PCIE_SW_TESTIN_FORCE_ COMPLIANCE	R/W	禁止从Polling.active进入Polling.Compliance（PCIE_SW_TESTIN_SET_COMP_RX_BIT置1时不适用）。
9	PCIE_SW_TESTIN_DIS_ COMPLIANCE	R/W	禁止从Polling.active进入Polling.Compliance（PCIE_SW_TESTIN_SET_COMP_RX_BIT置1时不适用）。
8	Reserved	R	始终读为0
7	PCIE_SW_TESTIN_SET_ COMP_RX_BIT	R/W	在发送的TS1有序集中设置合规接收位。
6	PCIE_SW_ TESTIN_DIS_SCRAMBLING	R/W	禁止加扰（仅限Gen 1/Gen 2模式）。

表9： PCIE_SW_TESTIN_REG（续）

PCIe®交换芯片测试寄存器			默认值：0x0000_0018
5	PCIE_SW_TESTIN_EN_RXTX_ASSRT	R/W	使能RX/TX性能吞吐率断言。
4	PCIE_SW_TESTIN_EN_INFO_ASSRT	R/W	使能信息断言（默认使能）。
3	PCIE_SW_TESTIN_EN_WARN_ASSRT	R/W	使能警告断言（默认使能）。
2	PCIE_SW_TESTIN_LOOPBACK_MST	R/W	环回请求模式：必须将该信号设置为1以指示链路进行环回（在环回主模式下）。当内核处于环回请求模式时，会在Loopback.Active状态下发送修改后的合规模式。
1	PCIE_SW_TESTIN_DIS_LPS_NEG	R/W	禁止低功耗状态协商：该信号置为有效时将禁止所有低功耗状态协商。
0	Masked	R	已屏蔽。仅供内部使用。

表10列出了PCIe交换芯片端口HP功能寄存器的内容。

表10： PCIE_SW_PORT_HP_CAP_REG

PCIe®交换芯片端口HP功能寄存器			默认值：0x0008_0000
Bit	**名称**	**R/W**	**说明**
31:19	HP_CAP_PHYS_SLOT_NUM[12:0]	R/W	物理插槽编号。仅PCI1xxxx器件的端口1支持物理插槽。
18	HP_CAP_NO_COMMAND_COMPLETE	R/W	支持无命令完成
17	HP_CAP_ELECTROMAG_INTERLOCK	R/W	支持机电互锁
16:15	HP_CAP_SLOT_PWR_LIM_SCL[1:0]	R/W	插槽功率限制调节
14:7	HP_CAP_SLOT_PWR_LIM_VAL[7:0]	R/W	插槽功率限值
6	HP_CAP_CAPABLE	R/W	热插拔功能： 0b——不支持 1b——支持
5	HP_CAP_SURPRISE	R/W	意外热插拔： 0b——不支持 1b——支持
4	HP_CAP_PWR_IND	R/W	支持电源指示灯 0b——不支持 1b——支持
3	HP_CAP_ATT_IND	R/W	支持注意指示灯 0b——不支持 1b——支持
2	HP_CAP_MRL	R/W	支持MRL传感器 0b——不支持 1b——支持
1	HP_CAP_PWR_CTL	R/W	支持电源控制器 0b——不支持 1b——支持

表10： PCIE_SW_PORT_HP_CAP_REG（续）

PCIe®交换芯片端口HP功能寄存器			默认值：0x0008_0000
Bit	**名称**	**R/W**	**说明**
0	HP_CAP_ATT_BUTN	R/W	支持注意按钮 0b——不支持 1b——支持

表11列出了PCIe交换芯片端口控制寄存器的内容。

表11： PCIE_SW_PORT_CTL_REG

PCIe®交换芯片端口控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:1	Reserved	R	始终读为0
0	PCIE_SW_PORT_RESET	R/W	对每个PCIe端口进行复位改写 该寄存器用于强制端口进入复位状态。 0b：端口复位遵循正常操作。 1b：端口复位信号置为有效，强制端口进入复位状态。 **注：**复位可用于配置存储空间PCIE_SWITCH_PORTx_ADDR_BASE（其中x为端口编号）中的寄存器。该功能适用于只能在交换芯片端口处于复位状态时进行配置的寄存器。使用方法是默认从OTP或EEPROM将该位置1，然后在相关寄存器配置完成后将该位清零。

表12列出了PCIe交换芯片端口供应商ID寄存器的内容。

表12： PCIE_SW_PORT_CFG_VID_REG

PCIe®交换芯片端口供应商ID寄存器			默认值：0x0000_1055
Bit	**名称**	**R/W**	**说明**
31:16	Reserved	R	始终读为0
15:0	PORT_PCI_VID[15:0]	R/W	PCI供应商ID。

表13列出了PCIe交换芯片端口器件ID寄存器的内容。

表13： PCIE_SW_PORT_CFG_DID_REG

PCIe®交换芯片端口器件ID寄存器			默认值：按端口
Bit	**名称**	**R/W**	**说明**
31:24	Reserved	R	始终读为0
23:16	PORT_PCI_RID[7:0]	R/W	PCI版本ID
15:0	PORT_PCI_DID[15:0]	R/W	PCI供应商ID

表14列出了PCIe交换芯片端口调试寄存器的内容。

表14： PCIE_SW_PORT_DBG_REG

PCIe®交换芯片端口调试寄存器			默认值： 0xXXXX_XXXX
Bit	**名称**	**R/W**	**说明**
31:18	Reserved	R	始终读为0
17:16	DBG_EQU_PHASE[1:0]	R	指示LTSSM处于恢复状态时的均衡阶段。 • 00b：阶段0 • 01b：阶段1 • 10b：阶段2 • 11b：阶段3
15:13	Reserved	R	始终读为0
12:8	DBG_LINK_SPEED[4:0]	R	报告PCIe链路速度： • 0_0001b：2.5 GTps • 0_0010b：5.0 GTps • 0_0011b：8.0 GTps **注：**该信号不指示链路接通。
7:5	Reserved	R	始终读为0
4:0	DBG_LTSSM_STATE[4:0]	R	报告当前LTSSM状态： • 00h：detect.quiet • 01h：detect.active • 02h：polling.active • 03h：polling.compliance • 04h：polling.configuration • 05h：config.linkwidthstart • 06h：config.linkwidthaccept • 07h：config.lanenumwait • 08h：config.lanenumaccept • 09h：config.complete • 0Ah：config.idle • 0Bh：recovery.receiverlock • 0Ch：recovery.equalization • 0Dh：recovery.speed • 0Eh：recovery.receiverconfig • 0Fh：recovery.idle • 10h：L0 • 11h：L0s • 12h：L1.entry • 13h：L1.idle • 14h：L2.idle • 15h：L2.transmitwake • 16h：disable • 17h：loopback.entry • 18h：loopback.active • 19h：loopback.exit • 1Ah：hotreset

表15列出了PCIe交换芯片端口PCI配置空间寄存器的内容。

表15： PCIE_SW_PORT_PCI_CONFIG_SPACE

PCIe®交换芯片端口PCI配置空间			默认值： 取决于功能
Bit	**名称**	**R/W**	**说明**
4095:0	PCIE_SW_PORT_PCI_CONFIG_SPACE	R/W	请参见附录B：“PCI配置空间”

7.0 USB主机实现

下列器件中提供USB主机控制器：

- PCI11400
- PCI11414
- PCI11101

USB主机控制器ID如下：

- 供应商ID：0x1055（Microchip Technology, Inc/SMSC）
- 器件ID：
  - PCI12000：N/A（无USB）
  - PCI11010：N/A（无USB）
  - PCI11101：0xA020
  - PCI11400：0xA030
  - PCI11414：0xA040

通常，以下各项应由终端系统集成商配置：

- 最大端口速度：10 Gb、5 Gb或480 Mb
- 端口1-4禁止
- 端口类型：Type-A或USB Type-C
- USB Type-C Rp vSafe5V电流能力
- 端口电源控制信号

7.1 章节

- 第7.2节“支持的端口组合”
- 第7.3节“USB端口所需的边带信号”
- 第7.4节“USB子系统寄存器”
- 第7.5节“USB端口配置示例”

7.2 支持的端口组合

PCI11400和PCI11414最多可实现两个USB Type-C端口，其余端口为USB Type-A。

当需要两个USB 3.2 Gen 2速度的USB Type-C端口时，必须在配置寄存器中重映射PCI11400/PCI11414收发器，以将USB 3.2收发器与正确的USB端口索引关联。有关支持的USB端口组合，请参见表16、表17和表18。

表16： PCI11400和PCI11414支持的USB端口组合

模式编号	物理端口1	物理端口2	物理端口3	物理端口4
1	最大速度： USB 3.2 Gen 2 连接器类型： Type-A USB 2.0 PHY编号： 1 USB 2.0逻辑端口映射： 端口1 USB 3.2 PHY编号： 1 USB 3.2逻辑端口映射： 端口1	最大速度： USB 3.2 Gen 2 连接器类型： Type-A USB 2.0 PHY编号： 2 USB 2.0逻辑端口映射： 端口2 USB 3.2 PHY编号： 2 USB 3.2逻辑端口映射： 端口2	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 3 USB 2.0逻辑端口映射： 端口3	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 4 USB 2.0逻辑端口映射： 端口4
2	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 1 USB 2.0逻辑端口映射： 端口1	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 2 USB 2.0逻辑端口映射： 端口2	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 3 USB 2.0逻辑端口映射： 端口3	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 4 USB 2.0逻辑端口映射： 端口4
3	最大速度： USB 3.2 Gen 2 连接器类型： Type-C® USB 2.0 PHY编号： 1 USB 3.2逻辑端口映射： 端口1 USB 3.2 PHY编号： 1 USB 3.2逻辑端口映射： 端口1	最大速度： USB 3.2 Gen 2 连接器类型： Type-C USB 2.0 PHY编号： 2 USB 2.0逻辑端口映射： 端口2 USB 3.2 PHY编号： 2 USB 3.2逻辑端口映射： 端口2	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 3 USB 2.0逻辑端口映射： 端口3	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 4 USB 2.0逻辑端口映射： 端口4

表16： PCI11400和PCI11414支持的USB端口组合（续）

模式编号	物理端口1	物理端口2	物理端口3	物理端口4
4	最大速度： USB 2.0 连接器类型： Type-C USB 2.0 PHY编号： 1 USB 2.0逻辑端口映射： 端口1	最大速度： USB 2.0 连接器类型： Type-C USB 2.0 PHY编号： 2 USB 2.0逻辑端口映射： 端口2	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 3 USB 2.0逻辑端口映射： 端口3	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 4 USB 2.0逻辑端口映射： 端口4
5	最大速度： USB 3.2 Gen 2 连接器类型： Type-C USB 2.0 PHY编号： 1 USB 2.0逻辑端口映射： 端口1 USB 3.2 PHY编号： 1 + 2 USB 3.2逻辑端口映射： 端口1	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 2 USB 2.0逻辑端口映射： 端口2	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 3 USB 2.0逻辑端口映射： 端口3	最大速度： USB 2.0 连接器类型： Type-A USB 2.0 PHY编号： 4 USB 2.0逻辑端口映射： 端口4

表17： PCI11101支持的USB端口组合

模式编号	物理端口1
1	**最大速度：** USB 3.2 Gen 2 **连接器类型：** Type-A **USB 2.0 PHY编号：** 1 **USB 2.0逻辑端口映射：** 端口1 **USB 3.2 PHY编号：** 1 **USB 3.2逻辑端口映射：** 端口2
2	**最大速度：** USB 2.0 **连接器类型：** Type-A **USB 2.0 PHY编号：** 1 **USB 2.0逻辑端口映射：** 端口1

表18： 支持的端口组合

模式	端口1	端口2	端口3	端口4
1	A			
	A	A		
	A	A		
	A	A	A	
	A	A	A	A
2	A			
	A	A		
	A	A		
	A	A		
3	C			
	C	C		
	C	A		
	C	A	A	
	C	A	A	A

图注：

绿色 = USB 2　　蓝色 = USB 3　　黄色 = 带外部多路开关的USB 3

表18： 支持的端口组合（续）

模式	端口1	端口2	端口3	端口4
4	C			
	C	C		
	C	A		
	C	A	A	
	C	A	A	A
5	C			
	C	A		
	C	A	A	
	C	A	A	A
	C	A		
	C	A	A	
	C	A	A	A
	C	C		
	C	C	A	
	C	C	A	A

图注：

绿色 = USB 2　　蓝色 = USB 3　　黄色 = 带外部多路开关的USB 3

7.3 USB端口所需的边带信号

7.3.1 USB TYPE-A端口

实现为USB Type-A的端口需要通过可用的固定功能和可编程（PROG）功能引脚来复用这些边带信号，以便连接到外部USB系统电路。表19列出了USB Type-A边带信号。

表19： USB TYPE-A边带信号

功能	引脚可用性	说明	外部连接
VB_PRT_CTL_P1/ VB_OCS_P1_N	**PROG70——功能8（PCI11400默认）** **PROG81——功能8（PCI11414默认）**	USB端口X的组合端口电源控制器/负载开关和过流故障检测引脚。使能内部上拉电阻。 这是PROG功能信号。	连接到外部5V USB端口电源开关的使能引脚，并与外部5V USB端口电源开关的低电平有效FAULT#或等效过流标志引脚进行线或连接。
VB_PRT_CTL_P2/ VB_OCS_P2_N	**PROG71——功能9（PCI11400默认）** **PROG82——功能9（PCI11414默认）**		
VB_PRT_CTL_P3/ VB_OCS_P3_N	**PROG80——功能5（PCI11400默认）** **PROG83——功能8（PCI11414默认）**		
VB_PRT_CTL_P4/ VB_OCS_P4_N	**PROG81——功能9（PCI11400默认）** **PROG84——功能1（PCI11414默认）**		

注： 默认情况下，PCI11101不会为有可用信号的USB 2.0端口配置VB_PRT_CTL_P1/VB_OCS_P1_N引脚。如果PCI11101应用中需要端口控制引脚，必须通过配置（EEPROM或OTP）选择PROG70或PROG81。

7.3.2 端口控制I/O模式

表20列出了VB_PRT_CTL_Px信号的不同工作模式。

表20： USB TYPE-C®边带信号

功能	端口处于禁止状态	端口处于使能状态
引脚方向	输出	输入
内部上拉使能	禁止	使能
推挽式置为有效	驱动为低电平	无法置为有效
系统目标	强制关闭VBUS电源。	通过内部上拉电阻使能VBUS电源，并监视来自VBUS电源电路的漏极开路故障指示输出的过流（OCS）事件输入状态。

7.3.3 过流检测

下面介绍了滤波器模块的操作。有关连接图，另请参见图2和图3。

- 当端口电源未使能时，忽略OCS事件。
- 当端口电源开启后，首次基于USB_OCS_REG.START_LOCKOUT_TIMER[7:0]值忽略OCS事件。请勿将START_LOCKOUT_TIMER[7:0]设置为零。
- 如果OCS事件的持续时间小于USB_OCS_REG.OC_TIMER[1:0]值，则将被忽略。
- 有效OCS的判定条件如下：
  - USB_OCS_REG.OCS_INACT[7:0] = 0。在USB_OCS_REG.OC_TIMER[1:0]超时后发生的单脉冲OCS事件将被判定为OCS状态。
  - USB_OCS_REG.OCS_MIN_WIDTH[6:0] = 0。在USB_OCS_REG.OC_TIMER[1:0]超时后发生的单脉冲OCS事件将被判定为OCS状态。
  - USB_OCS_REG.OCS_INACT[7:0] != 0。在同一非活动周期内发生两次或更多OCS事件。如果第二次OCS事件未发生在同一ocs_inact周期内，则将忽略第一次OCS事件。
  - 如果OCS引脚为低电平的持续时间超过USB_OCS_REG.OCS_MIN_WIDTH[6:0]值，则判定为OCS状态（单个长脉宽OCS事件）。

图2： TYPE-A USB 2.0连接图

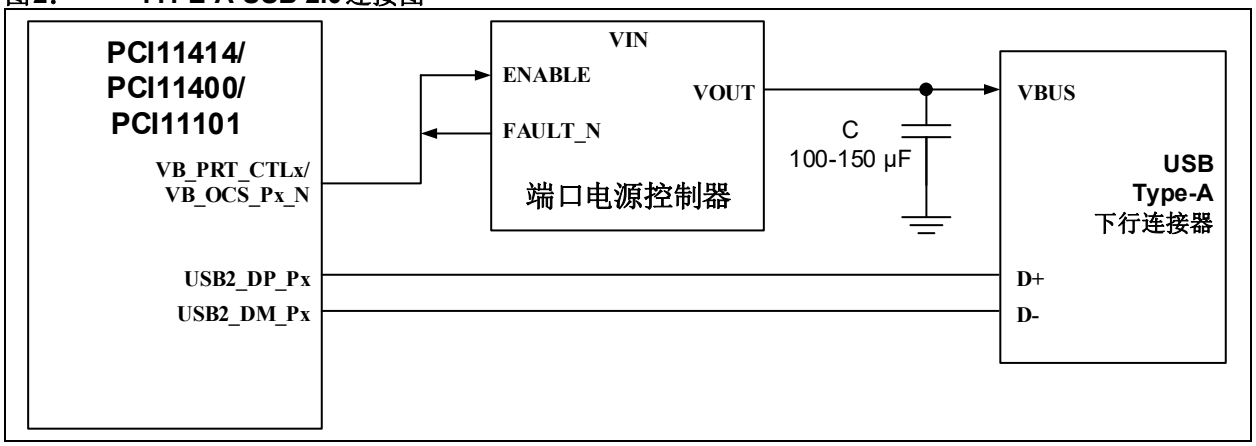
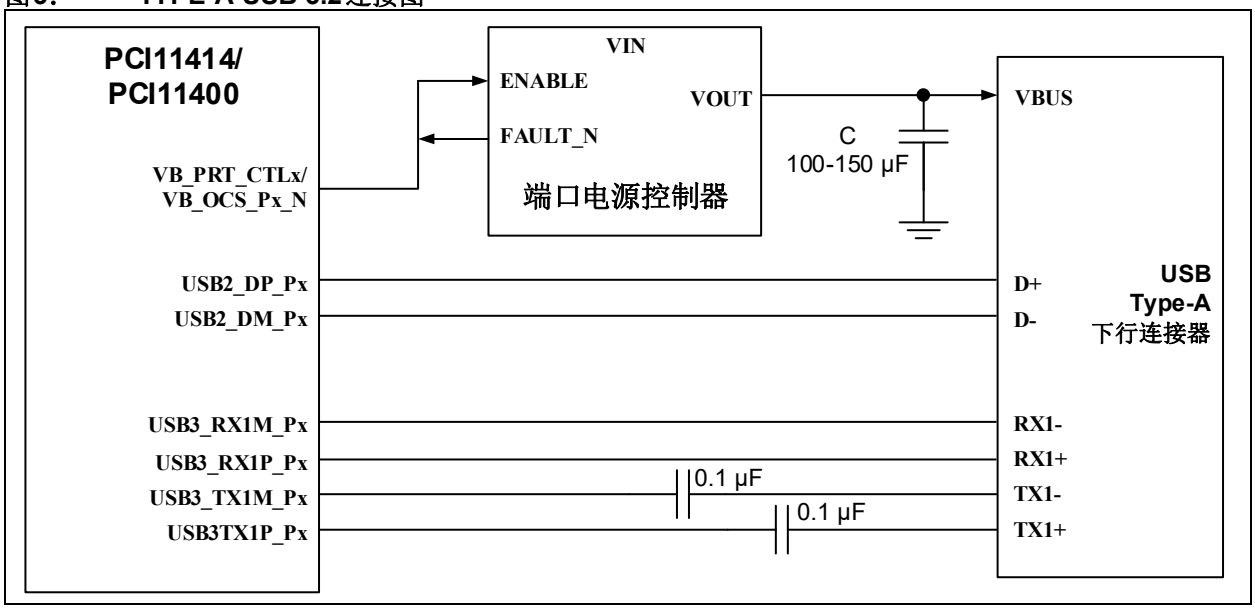


图3： TYPE-A USB 3.2连接图



7.3.4 USB TYPE-C端口所需的额外信号

PCI1xxxx最多支持两个USB Type-C端口。任何实现为USB Type-C的端口都需要通过其中一个可用的可编程（PROG）功能引脚来复用这些额外信号，以便连接到外部USB系统电路。有关边带信号的详细信息，请参见表21和表22；有关连接图，请参见图4、图5和图6。

USB Type-C端口还必须包括第7.3.1节中提到的VB_PRT_CTL_Px/VB_OCS_Px_N。

注：	PCI11101不支持USB Type-C端口。如果PCI11101应用需要Type-C端口，则需使用外部USB Type-C端口控制器。

表21： USB TYPE-C®固定功能边带信号

功能	引脚映射	说明	外部连接
CC1_P1	PCI11414——引脚114 PCI11400——引脚90	配置通道USB Type-C®信号，用于检测是否存在有效的Type-C端口端到端连接，以及向连接的器件通知端口上VBUS支持的最大电流。 CC1上存在连接表示Type-C处于“未翻转”方向，Type-C连接器上的有效USB 3.2通信引脚为TX1+/-和RX1+/-。	直接连接到USB Type-C连接器的CC1引脚。
CC1_P2	PCI11414——引脚103 PCI11400——引脚79		

表21： USB TYPE-C®固定功能边带信号（续）

功能	引脚映射	说明	外部连接
CC2_P1	PCI11414——引脚115 PCI11400——引脚91	配置通道Type-C信号，用于检测是否存在有效的Type-C端口端到端连接，以及向连接的器件通知端口上VBUS支持的最大电流。 CC2上存在连接表示Type-C处于“翻转”方向，Type-C连接器上的有效USB 3.2通信引脚为TX2+/-和RX2+/-。	直接连接到USB Type-C连接器的CC2引脚。
CC2_P2	PCI11414——引脚104 PCI11400——引脚80		
VBUS_MON_P1	PCI11414——引脚129 PCI11414——引脚105	用于监视VBUS的模拟输入引脚，实现符合USB Type-C规范的行为。 确保在VBUS已由其他电源供电时不会使能外部5V USB端口电源开关。 该信号为固定功能信号。	连接到外部VBUS分压电路（位于外部5V USB端口电源开关的输出端）。
VBUS_MON_P2	PCI11414——引脚132 PCI11400——引脚108		

表22： USB TYPE-C®可编程边带信号

功能	功能引脚可用性	说明	外部连接
VBUS_DIS_P1	**PROG76——功能7（PCI11414和PCI11400默认）**	数字输出引脚，用于根据USB Type-C®端口状态同步使能外部“VBUS放电”电路，并确保在下一次连接之前插口电源处于“冷”状态（<800 mV）。 该引脚为PROG功能信号。	连接到外部分立VBUS放电电路，如果外部5V USB端口电源开关与USB Type-C兼容，则不连接（在使能信号置为无效时自动放电）。
VBUS_DIS_P2	PROG51——功能4 PROG75——功能7 PROG86——功能2		
VCONN1_EN_P1	**PROG77——功能7（PCI11414默认）**	数字输出引脚，用于在USB Type-C“有源线缆”连接到端口时在USB Type-C插口的CC1上使能外部VCONN源。 该引脚为PROG功能信号。	连接到能为USB Type-C“有源线缆”提供至少1W功率的外部VCONN源电路（仅在USB Type-C CC1引脚上使能）。
VCONN1_EN_P2	PROG23——功能3 PROG48——功能6		
VCONN2_EN_P1	**PROG78——功能7（PCI11414默认）**	数字输出引脚，用于在USB Type-C“有源线缆”连接到端口时在USB Type-C插口的CC2上使能外部VCONN源。 该引脚为PROG功能信号。	连接到能为USB Type-C“有源线缆”提供至少1W功率的外部VCONN源电路（仅在USB Type-C CC2引脚上使能）。
VCONN2_EN_P2	PROG24——功能3 PROG49——功能6		
CC_ORIENT_P1	PROG6——功能3 PROG20——功能8 PROG68——功能7 PROG71——功能12	可选信号，用于指示在使用外部多路开关实现USB Type-C端口时检测到的Type-C方向。 该引脚为PROG功能信号。	连接到外部复用器件的方向选择输入。
CC_ORIENT_P2	PROG47——功能6 PROG86——功能1 PROG87——功能8		

表22： USB TYPE-C®可编程边带信号（续）

功能	功能引脚可用性	说明	外部连接
CC_ATTACH_P1	PROG5——功能3 PROG19——功能8 PROG69——功能11 PROG70——功能11	可选信号，用于指示在使用外部多路开关实现USB Type-C端口时检测到的Type-C端到端连接。该引脚为PROG功能信号。	连接到外部复用器件的使能引脚输入。
CC_ATTACH_P2	PROG46——功能6 PROG85——功能1 PROG86——功能8		

图4： TYPE-C® USB 2.0连接图

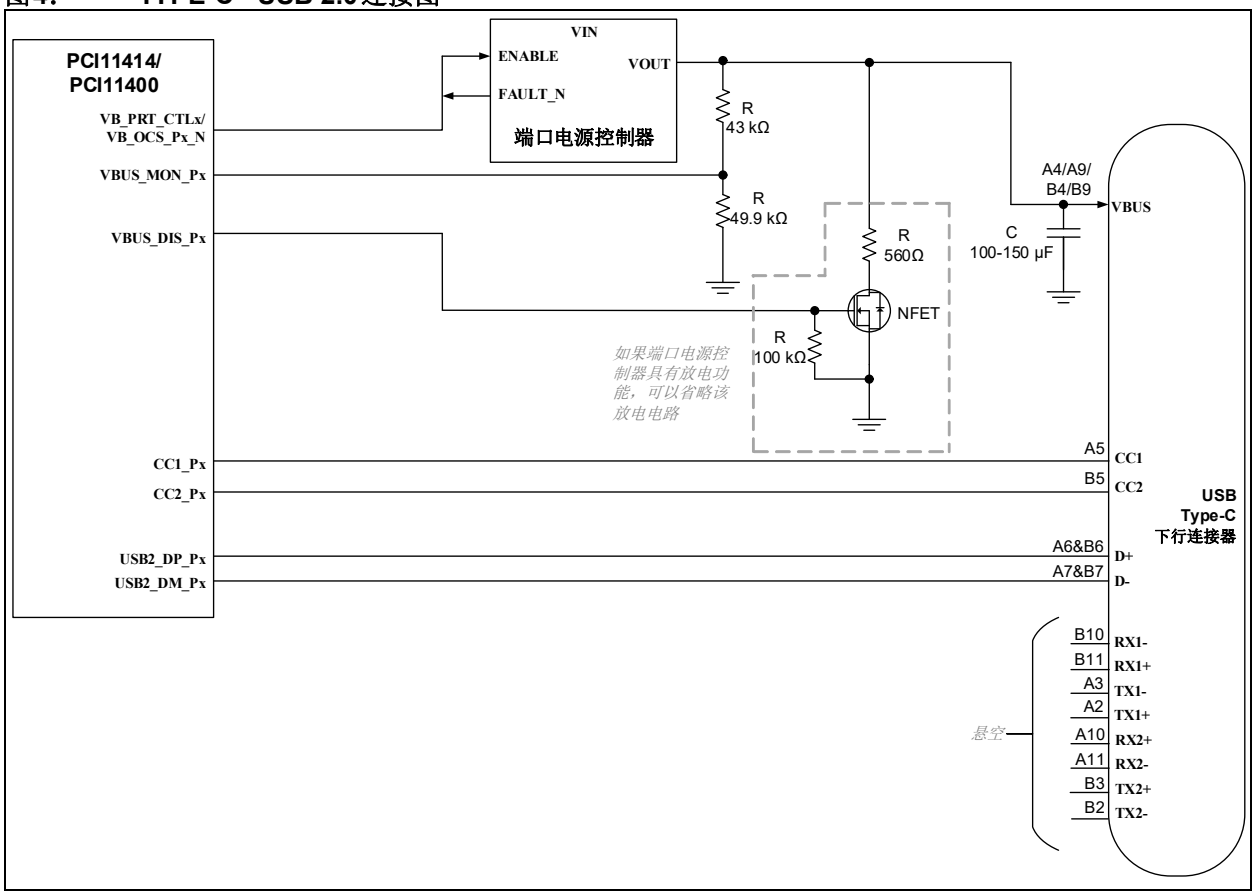


图5： TYPE-C® USB 3.2连接图（内部多路开关）

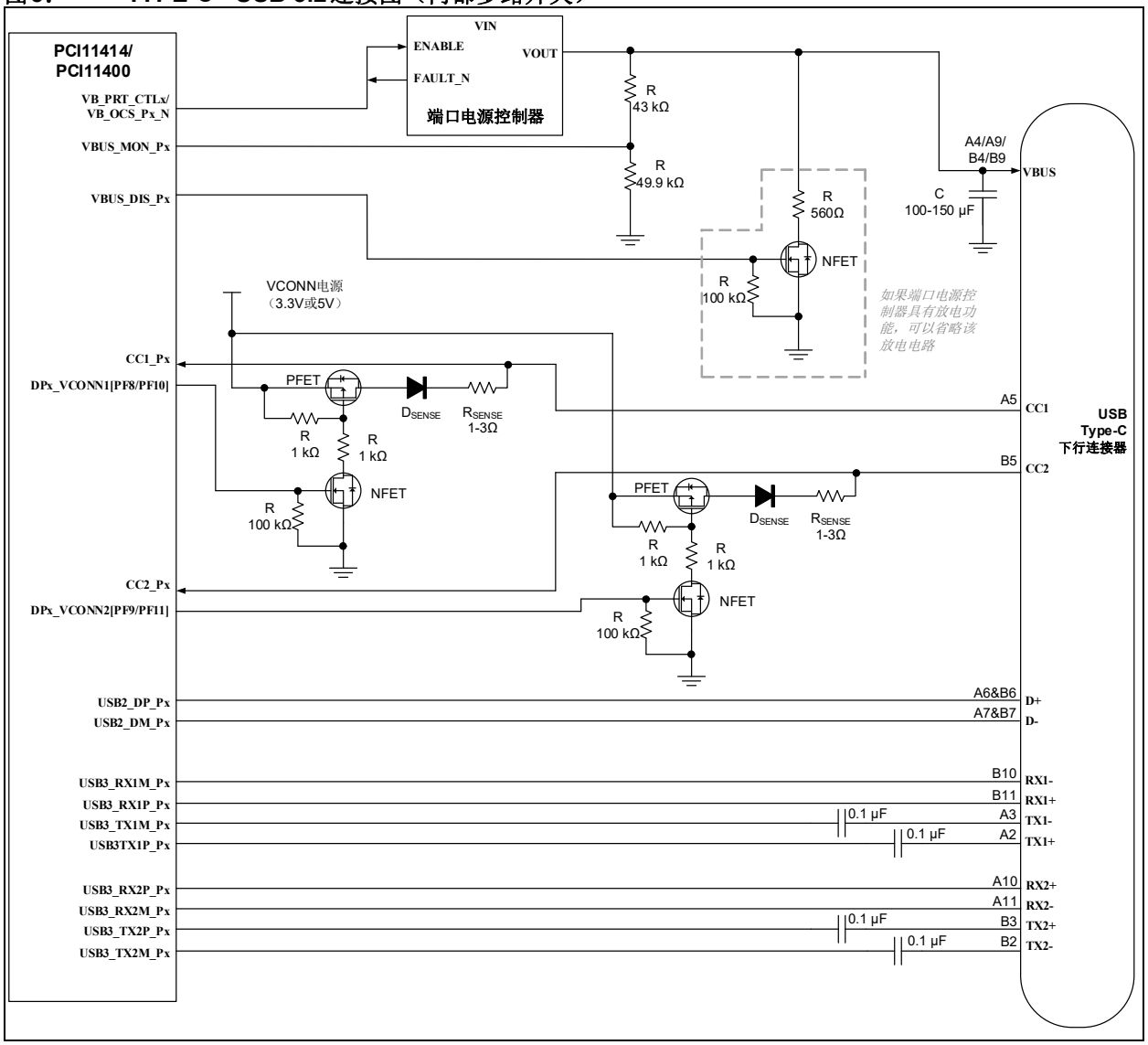
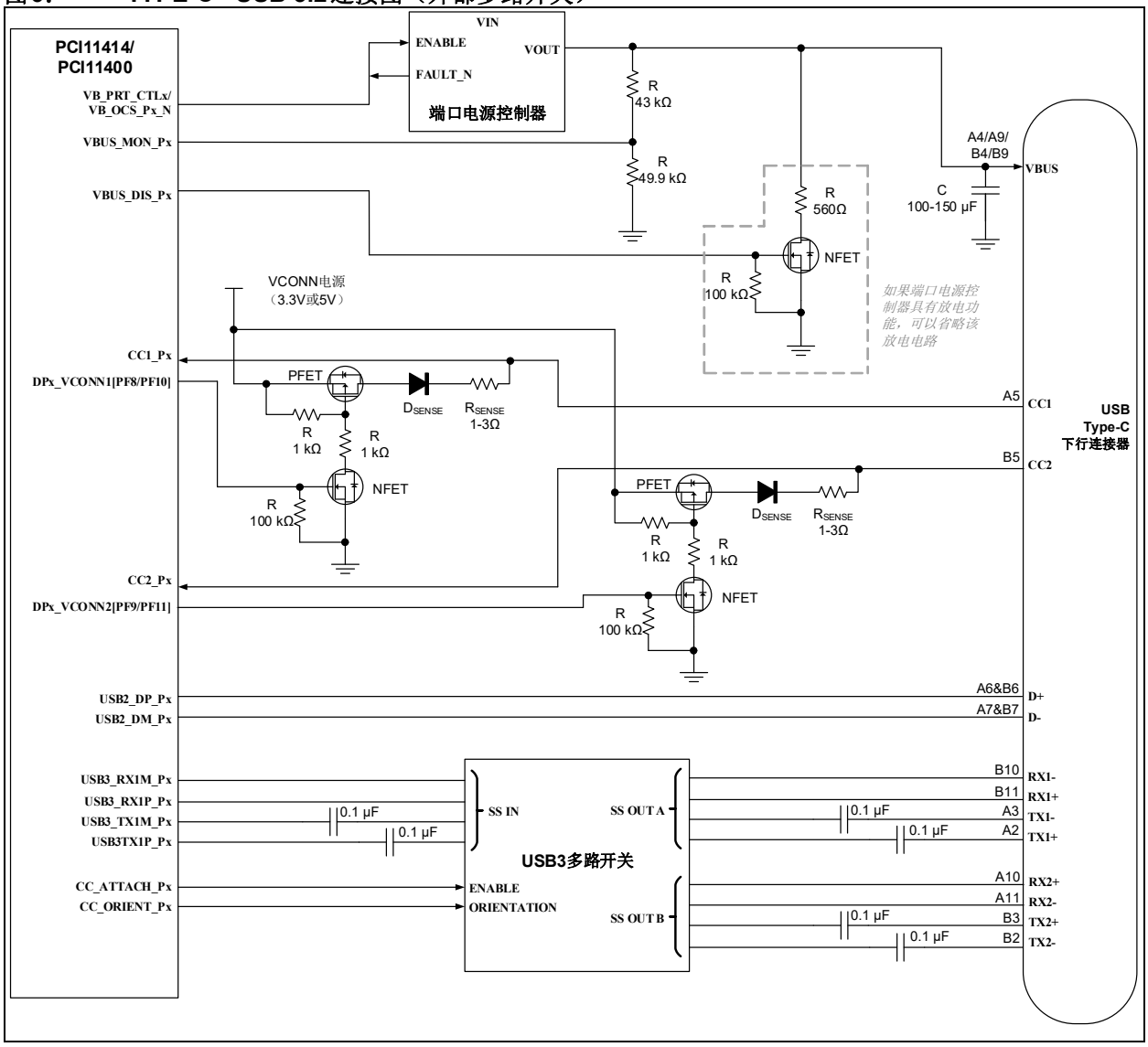


图6： TYPE-C® USB 3.2连接图（外部多路开关）



7.4 USB子系统寄存器

表23列出了USB_SUBSYSTEM_ADDR_BASE = 0x0004_0000时可用的寄存器。

注： PCI1xxxx器件包含数千个寄存器，其中一部分寄存器保留供将来使用，另一部分寄存器的内容未纳入公开文档，因为其不涉及终端系统集成商用例（即：这些寄存器供硬件用于执行底层运行时任务，或在运行时由驱动程序直接使用/控制）。**请勿修改未纳入文档的寄存器的内容。**

表23： USB子系统寄存器地址相对于USB_SUBSYSTEM_ADDR_BASE的偏移量

寄存器名称	端口1	端口2	端口3	端口4
USB2_TEST_MODE	0x1_6810			
USB2_TEST_PORT_SEL	0x1_6812			
USB2_AFE_CTRL	0x1_80C9	0x1_84C9	0x1_88C9	0x1_8CC9
USB2_HS_CTRL	0x1_80CA	0x1_84CA	0x1_88CA	0x1_8CCA
USB2_TEST_EN	0x1_80CB	0x1_84CB	0x1_88CB	0x1_8CCB
PHY_UP_REG	0x1_80CC	0x1_84CC	0x1_88CC	0x1_8CCC
LTSSM_STATE	0x1_81C0	0x1_85C0	**n/a**	**n/a**
PORT_CFG_SEL_x	0x1_6004	0x1_6008	0x1_600C	0x1_6010
USB_OCS_SEL_x	0x1_6020	0x1_6024	0x1_6028	0x1_602C
PORT_STAT_REG_x	0x1_6054	0x1_6058	0x1_605C	0x1_6060
USB_OCS_REG_x	0x1_6070			
CC_CONFIG_Px	0x1_A220	0x1_A620	n/a	n/a
VCONN1_PAD_CTL_Px	0x1_A228	0x1_A628	n/a	n/a
VCONN2_PAD_CTL_Px	0x1_A229	0x1_A629	n/a	n/a
DISCHARGE_PAD_CTL_Px	0x1_A22A	0x1_A62A	n/a	n/a
ATTACH_PAD_CTL_Px	0x1_A22B	0x1_A62B	n/a	n/a
ORIENT_PAD_CTL_Px	0x1_A22C	0x1_A62C	n/a	n/a
USB_RESET_REG	0x1_B000			
USB_PORT_ENABLE_REG	0x1_B004			

表24列出了USB测试模式寄存器的详细信息。

表24： USB2_TEST_MODE

USB 2测试模式寄存器			**默认值：** 0x00
Bit	**名称**	**R/W**	**说明**
7:4	Reserved	R	始终为0
3:0	TEST_MODE[3:0]	R/W	0000b：正常工作模式 0001b：USB 2 PHY特性模式 0010b：UTMI BIST模式 0011b - 0111b：保留

表25列出了USB测试模式端口选择寄存器的详细信息。

表25： USB2_TEST_PORT_SEL

USB测试模式端口选择寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:3	Reserved	R	始终为0
2:0	DFP_SEL[2:0]	R/W	000b：保留 001b：端口1为测试模式 010b：端口2为测试模式 011b：端口3为测试模式 100b：端口4为测试模式 101b - 111b：保留

表26列出了USB 2模拟前端控制寄存器的详细信息。

表26： USB2_AFE_CTRL

USB 2模拟前端控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7	Reserved	R	始终为0
6	USB2_FS_OEB	R/W	FS/LS驱动器输出使能 低电平有效。 1b0：使能驱动器 1b1：驱动器为三态
5	USB2_FS_VPO	R/W	FS/LS正输出数据 将数据驱动到DP输出
4	USB2_FS_VMO	R/W	FS/LS负输出数据 将数据驱动到DM输出
3	USB2_FS_DATA	R	FS/LS差分接收器输入状态
2:0	Reserved	R	始终为0

表27列出了USB 2高速AFE控制寄存器的详细信息。

表27： USB2_HS_CTRL

USB 2高速AFE控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:6	USB2_HS_TXVALID[1:0]	R/W	HS发送有效屏蔽。指示AFE_HS_TXVALID[1:0]总线的哪个位有效。 • 00b：两个位都无效 • 01b：LSb有效 • 10b：无效组合 • 11b：两个位都有效
5:4	USB2_HS_TXDATA[1:0]	R/W	HS发送数据。驱动器数据先发送LSb。
3	USB2_HS_CS_EN	R/W	HS电流源使能。 • 0b：驱动器掉电 • 1b：驱动器上电 只要端口处于高速模式，该信号就会置为有效。

表27： USB2_HS_CTRL（续）

USB 2高速AFE控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
2:0	USB2_HS_DRIVE[2:0]	R/W	HS输出电流（PHY升压） • 000b：标称值17.78 mA • 001b：减小5% • 010b：增大10% • 011b：增大5% • 100b：增大20% • 101b：增大15% • 110b：增大30% • 111b：增大25%

表28列出了USB 2测试寄存器的详细信息。

表28： USB2_TEST_EN

USB 2测试寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7	USB2_RPU_DP_EN	R/W	RPU DP终端电阻控制。该位有效时使能DP上的1.5 kΩ终端电阻。
6	USB2_RPU_DM_EN	R/W	RPU DM终端电阻控制。该位有效时使能DM上的1.5 kΩ终端电阻。
5	USB2_RPD_DP_EN	R/W	RPD DP终端电阻控制。该位有效时使能DP上的15 kΩ终端电阻。
4	USB2_RPD_DM_EN	R/W	RPD DM终端电阻控制。该位有效时使能DM上的15 kΩ终端电阻。
3:2	Reserved	R	始终为0
1	USB2_HS_TXACTIVE	R/W	HS驱动器有效。禁止该位时（在空闲模式期间），将HS驱动器置于低功耗模式。必须使能afe_hs_cs_en。 0b：驱动器处于低功耗模式 1b：驱动器处于有效发送模式（17.78 mA源激活）
0	Reserved	R	始终为0

表29列出了USB 2 PHY调整寄存器的详细信息。

表29： PHY_UP_REG

USB 2 PHY调整寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:6	USB2_HS_DISC_TUNE[1:0]	R/W	USB 2.0 AFE HS断开调整。连接到USB 2.0 PCS（UTMI）。 • 00b：标称值575 mV触发点 • 01b：增大50 mV • 10b：增大100 mV • 11b：增大125 mV

表29： PHY_UP_REG（续）

USB 2 PHY调整寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
5	USB2_HS_TERM_EN	R/W	HS终端电阻控制。该位有效时使能DP和DM上的45 kΩ终端电阻。
4	USB2_HS_SQUELCH_B	R	AFE高速抑制 指示HS过采样数据何时有效。低电平有效。 • 0b：数据有效 • 1b：数据无效
3	USB2_HS_DISC	R	AFE高速断开 指示在HS模式下何时断开线路。该信号应仅在第32个位时间的HS EOP期间选通。 • 0b：正常状态 • 1b：断开状态
2:0	USB2_SQ_TUNE[2:0]	R/W	抑制调整（Varisense） • 000b：标称值100 mV触发点 • 001b：减小12.5 mV • 010b：减小25 mV • 011b：减小37.5 mV • 100b：减小50 mV • 101b：减小62.5 mV • 110b：增大25 mV • 111b：增大12.5 mV

表30列出了USB 3 LTSSM状态寄存器的详细信息。

表30： LTSSM_STATE

USB 3 LTSSM状态寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:4	LTSSM_STATE[3:0]	R	LTSSM状态 0000b = U0（无子状态） 0001b = U1（无子状态） 0010b = U2（无子状态） 0011b = U3（无子状态） 0100b = SS.Disabled（子状态见下文） 0101b = Rx.Detect（子状态见下文） 0110b = SS.Inactive（子状态见下文） 0111b = Polling（子状态见下文） 1000b = Recovery（子状态见下文） 1001b = HotReset（子状态见下文） 1010b = Compliance（无子状态） 1011b = Loopback（无子状态）

表30： LTSSM_STATE（续）

USB 3 LTSSM状态寄存器			默认值： 0x00
Bit	**名称**	**R/W**	**说明**
3:0	LTSSM_SUB_STATE[3:0]	R	**U0子状态：**无 **U1子状态：** 0x0 U1 POWER = PHY电源状态P0 -> P1请求 0x1 U1_POWER_A = 等待PHY电源切换完成 0x2 U1_ACTIVE = 等待远程/本地U1退出 0x3 U1_EXIT_LOC_RESP = 启动U1退出并等待远程伙伴的最小响应 0x4 U1_EXIT_LOC_FIN = 本地发起的U1退出；发送最低要求的U1退出信号 0x5 U1_EXIT_REM = 远程发起的U1退出；发送最低要求的U1退出信号 0x6 U1_EXIT_P0 = PHY电源状态P1 -> P0请求 0x7 U1_EXIT_P0_A = 等待PHY电源切换P1 -> P0完成 0x8 U1_EXIT_DONE = U1退出成功；U1 -> Recovery **U2子状态：** 0x0 U2_POWER = PHY电源状态P0 -> P2请求 0x1 U2_POWER2 = 等待PHY电源切换完成 0x2 U2_ACTIVE = 等待远程/本地U2退出 0x3 U2_ACTIVE_0 = 请求启动接收器检测 0x4 U2_ACTIVE_1 = 等待接收器检测响应 0x5 U2_EXIT_LOC_RESP = 启动U2退出并等待远程伙伴的最小响应 0x6 U2_EXIT_LOC_FIN = 本地发起的U2退出；发送最低要求的U2退出信号 0x7 U2_EXIT_REM = 远程发起的U2退出；发送最低要求的U2退出信号 0x8 U2_EXIT_P0 = PHY电源状态P2 -> P0请求 0x9 U2_EXIT_P0_A = 等待PHY电源切换P1 -> P0完成 0xA U2_EXIT_DONE = U2退出成功；U2 -> Recovery **U3子状态：** 0x0 U3_POWER = PHY电源状态P0 -> P2/P3请求 0x1 U3_POWER3 = 等待PHY电压切换完成 0x2 U3_ACTIVE = 等待远程/本地U3退出 0x3 U3_ACTIVE_0 = 请求启动接收器检测 0x4 U3_ACTIVE_1 = 等待接收器检测响应 0x5 U3_EXIT_LOC_POW = PHY电源状态P3 -> P0 -> P2请求 0x6 U3_EXIT_LOC_POW_A = 等待PHY电压切换完成 0x7 U3_EXIT_LOC_RESP = 启动U3退出并等待远程伙伴的最小响应 0x8 U3_EXIT_LOC_FIN = 本地发起的U3退出；发送最低要求的U3退出信号 0x9 U3_EXIT_REM_POW = PHY电源状态P3 -> P0 -> P2请求 0xA U3_EXIT_REM_POW_A = 等待PHY电压切换完成 0xB U3_EXIT_REM = 远程发起的U3退出；发送最低要求的U3退出信号

表30： LTSSM_STATE（续）

USB 3 LTSSM状态寄存器			默认值： 0x00
Bit	**名称**	**R/W**	**说明**
			P2/P3 -> P0请求 0xD U3_EXIT_P0_A = 等待PHY电压切换P2/P3 -> P0完成 0xE U3_EXIT = 接收到远程/本地U3退出信号 0xF U3_EXIT_DONE U3 = 退出成功； U3 -> Recovery

表31列出了端口配置选择寄存器的详细信息。

表31： PORT_CFG_SEL_x

端口配置选择寄存器			默认值： 0x0000_0803
Bit	**名称**	**R/W**	**说明**
31:23	Reserved	R	始终为0
22	ORIENTATION_POLARITY	R/W	控制用于外部多路开关控制的CC_ORIENT_Px输出信号引脚的极性。 0b：当在CC1上检测到CC线时，CC_ORIENT_Px置为有效 1b：当在CC2上检测到CC线时，CC_ORIENT_Px置为有效
21	CC_ENABLE	R/W	使能USB Type-C®检测逻辑。 0b：禁止USB Type-C检测逻辑 1b：使能USB Type-C检测逻辑 **注：**仅适用于端口1和/或端口2。
20:18	Reserved	R	始终为0
17:16	VBUS_INPUT_SEL[1:0]	R/W	选择USB Type-C端口的VBUS检测逻辑。 00b：保留 01b：使能VBUS_MON_Px输入 10b：保留 11b：保留 **注：**仅在PORT_TYPE[1:0] = 2b'01（配置为USB Type-C）时有效。仅适用于端口1和/或端口2。
15	MUX_EN	R/W	使能内部USB 3.2 Gen 2多路开关。 0b：禁止内部USB 3.2 Gen 2多路开关 1b：使能内部USB 3.2 Gen 2多路开关 **注：**仅在PORT_TYPE[1:0] = 2b'01（配置为USB Type-C）时有效。仅适用于端口1和/或端口2。
14	Reserved	R	始终为0

表31： PORT_CFG_SEL_x（续）

端口配置选择寄存器			默认值： 0x0000_0803
Bit	**名称**	**R/W**	**说明**
13:12	C_MUX_SEL[1:0]	R/W	选择USB Type-C®检测逻辑对内部USB 3.2 Gen 2多路开关的控制方式。 00b：端口类型为USB Type-A（无内部复用） 01b：保留 10b：端口类型为仅支持USB 2.0的USB Type-C（无内部复用） 11b：端口类型为支持USB 3.2 Gen 2的USB Type-C（内部复用由USB Type-C连接和方向状态控制） **注：**仅在MUX_EN = 1和PORT_TYPE[1:0] = 2b'01（配置为USB Type-C）时有效。仅适用于端口1和/或端口2。
11	C_SEL_BnA	R/W	指示USB 3.2 Gen 2 PHY 0/1的内部多路开关选择状态 0b：选择USB 3.2 Gen 2 PHY 0 1b：选择USB 3.2 Gen 2 PHY 1 **注：**该位在C_MUX_SEL[1:0] = 2b'01或2b'11时由内部USB Type-C检测逻辑置1，在其他情况下则读为0。仅适用于端口1和/或端口2。
10	C_ATTACH	R/W	指示来自USB Type-C检测逻辑的连接状态 0b：USB Type-C端口处于未连接状态 1b：端口处于连接状态 **注：**仅在MUX_EN = 1和PORT_TYPE[1:0] = 2b'01（配置为USB Type-C）时有效。仅适用于端口1和/或端口2
9:8	PORT_TYPE[1:0]	R/W	选择USB端口类型（USB Type-A或USB Type-C） 00b：USB Type-A 01b：USB Type-C 10b：保留 11b：保留
7:4	Reserved	R	始终为0
3:0	PRT_SEL[1:0]	R/W	配置内部USB端口电源控制逻辑 0000b：禁止USB端口，无端口电源控制。 0001b：保留 0010b：保留 0011b：USB端口为USB 2.0或USB 3.2 Gen 2，使能端口电源控制逻辑。 0100b - 1111b：保留

表32列出了OCS配置选择寄存器的详细信息。

表32： USB_OCS_SEL_x

过流信号（OCS）配置选择寄存器			默认值：0x0000_0001
Bit	**名称**	**R/W**	**说明**
31:4	Reserved	R	始终为0
3:0	PRT_OCS_SEL[1:0]	R/W	选择USB端口过流（OCS）状态的信号源。 0000b：禁止USB端口 0001b：OCS由端口的VB_PRT_CTL_Px/VB_OCS_Px_N引脚指示 0010b - 1111b：保留

表33列出了端口状态寄存器的详细信息。

表33： PORT_STAT_REG_x

端口状态寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:2	Reserved	R	始终读为0
1	A_ATTACH_STATUS	R	Type-A端口连接状态的当前状态
0	C_ATTACH_STATUS	R	Type-C®端口连接状态的当前状态

表34列出了端口过流状态寄存器的详细信息。

表34： USB_OCS_REG_x

端口过流状态寄存器			默认值：0x0014_050A
Bit	**名称**	**R/W**	**说明**
31:26	Reserved	R	始终读为0
25:24	OC_TIMER[1:0]	R/W	过流定时器延时。用于判定脉冲有效的最小脉冲宽度。 00b = 3个时钟采样 01b = 6个时钟采样 10b = 12个时钟采样 11b = 24个时钟采样
23:16	OCS_INACT[7:0]	R/W	该定时器统计第二次OCS事件检测期间的非活动时长。 可配置值范围为0 - 255（0x00 - 0xFF） 默认值为20 ms。
15:8	OCS_MIN_WIDTH[6:0]	R/W	OCS脉冲窗口提供0 ms到5 ms的范围。
7:0	START_LOCKOUT_TIMER[7:0]	R/W	该定时器在端口电源开启后阻止检测OCS事件。该定时器值内的任何OCS事件都将忽略。 当前默认值为10 ms。

表35列出了Type-C CC配置寄存器的详细信息。

表35： CC_CONFIG_Px

TYPE-C® CC配置寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:8	Reserved	R	始终读为0
7	CONFIG_DONE	R	在CC逻辑的所有内部寄存器完成初始化后由硬件置1。
6	Reserved	R/W	始终为0
5	ERR_RECOVER	R/W	该位控制Type-C FSM是否应尝试从端口电源开关或VCONN FET引发的OCS状态中自动恢复。 不建议将该位置1，因为端口应直接通过USB主机驱动程序进行控制。
4:2	Reserved	R/W	始终为0
1:0	PWR_CAP[1:0]	R/W	在用作独立源信号时定义器件支持的充电电流。 00b：USB 2.0默认电流 01b：USB 3.0默认电流 10b：1.5A 11b：3.0A

表36列出了VCONN1焊盘控制寄存器的详细信息。

表36： VCONN1_PAD_CTL_Px

VCONN1焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:4	Reserved	R	始终读为0
3	CC1_VCONN_PF_POL	R/W	0b：使能VCONN1时将引脚设置为高电平，禁止VCONN1时将引脚设置为低电平。 1b：使能VCONN1时将引脚设置为低电平，禁止VCONN1时将引脚设置为高电平。
2	CC1_VCONN_PF_OD	R/W	0b：禁止漏极开路操作 1b：当相应输出为1时，禁止输出并使能上拉电阻。当输出为0时，使能输出并将其驱动为低电平。 **注：**必须将相应的VCONN1_EN_Px引脚配置为漏极开路。
1	CC1_VCONN_PF_PD	R/W	0b：禁止VCONN1_EN_Px引脚上的内部下拉电阻 1b：使能VCONN1_EN_Px引脚上的内部下拉电阻
0	CC1_VCONN_PF_PU	R/W	0b：禁止VCONN1_EN_Px引脚上的内部上拉电阻 1b：使能VCONN1_EN_Px引脚上的内部上拉电阻

表37列出了VCONN2焊盘控制寄存器的详细信息。

表37： VCONN2_PAD_CTL_PX

VCONN2焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:4	Reserved	R	始终读为0

表37： VCONN2_PAD_CTL_PX（续）

VCONN2焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
3	CC2_VCONN_PF_POL	R/W	0b：使能VCONN2时将引脚设置为高电平，禁止VCONN2时将引脚设置为低电平。 1b：使能VCONN2时将引脚设置为低电平，禁止VCONN2时将引脚设置为高电平。
2	CC2_VCONN_PF_OD	R/W	0b：禁止漏极开路操作 1b：当相应输出为1时，禁止输出并使能上拉电阻。当输出为0时，使能输出并将其驱动为低电平。 **注：**必须将相应的VCONN2_EN_Px引脚配置为漏极开路。
1	CC2_VCONN_PF_PD	R/W	0b：禁止VCONN2_EN_Px引脚上的内部下拉电阻 1b：使能VCONN2_EN_Px引脚上的内部下拉电阻
0	CC2_VCONN_PF_PU	R/W	0b：禁止VCONN2_EN_Px引脚上的内部上拉电阻 1b：使能VCONN2_EN_Px引脚上的内部上拉电阻

表38列出了VBUS放电焊盘控制寄存器的详细信息。

表38： DISCHARGE_PAD_CTL_PX

VBUS放电焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:4	Reserved	R	始终读为0
3	DISCHARGE_POL	R/W	0b：使能VBUS放电时将引脚设置为高电平，禁止VBUS放电时将引脚设置为低电平。 1b：使能VBUS放电时将引脚设置为低电平，禁止VBUS放电时将引脚设置为高电平。
2	DISCHARGE_OD	R/W	0b：禁止漏极开路操作 1b：当相应输出为1时，禁止输出并使能上拉电阻。当输出为0时，使能输出并将其驱动为低电平。 **注：**必须将相应的VBUS_DIS_Px引脚配置为漏极开路。
1	DISCHARGE_PD	R/W	0b：禁止VBUS_DIS_Px引脚上的内部下拉电阻 1b：使能VBUS_DIS_Px引脚上的内部下拉电阻
0	DISCHARGE_PU	R/W	0b：禁止VBUS_DIS_Px引脚上的内部上拉电阻 1b：使能VBUS_DIS_Px引脚上的内部上拉电阻

表39列出了USB-C连接焊盘控制寄存器的详细信息。

表39： ATTACH_PAD_CTL_PX

USB-C®连接焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:4	Reserved	R	始终读为0

表39： ATTACH_PAD_CTL_PX（续）

USB-C®连接焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
3	CC_ATTACH_POL	R/W	0b：检测到有效的端到端USB-C连接时将引脚设置为高电平，未检测到有效的USB-C连接时将引脚设置为低电平。 1b：检测到有效的端到端USB-C连接时将引脚设置为低电平，未检测到有效的USB-C连接时将引脚设置为高电平。
2	CC_ATTACH_OD	R/W	0b：禁止漏极开路操作 1b：当相应输出为1时，禁止输出并使能上拉电阻。当输出为0时，使能输出并将其驱动为低电平。 **注：**必须将相应的CC_ATTACH_Px引脚配置为漏极开路。
1	CC_ATTACH_PD	R/W	0b：禁止CC_ATTACH_Px引脚上的内部下拉电阻 1b：使能CC_ATTACH_Px引脚上的内部下拉电阻
0	CC_ATTACH_PU	R/W	0b：禁止CC_ATTACH_Px引脚上的内部上拉电阻 1b：使能CC_ATTACH_Px引脚上的内部上拉电阻

表40列出了USB-C方向焊盘控制寄存器的详细信息。

表40： ORIENT_PAD_CTL_PX

USB-C®连接焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:4	Reserved	R	始终读为0
3	CC_ORIENT_POL	R/W	0b：在CC2上检测到有效的端到端USB-C连接时将引脚设置为高电平，在CC1上检测到有效的端到端USB-C连接或未检测到有效的USB-C连接时将引脚设置为低电平。 1b：在CC2上检测到有效的端到端USB-C连接时将引脚设置为低电平，在CC1上检测到有效的端到端USB-C连接或未检测到有效的USB-C连接时将引脚设置为高电平。
2	CC_ORIENT_OD	R/W	0b：禁止漏极开路操作 1b：当相应输出为1时，禁止输出并使能上拉电阻。当输出为0时，使能输出并将其驱动为低电平。 **注：**必须将相应的CC_ORIENT_Px引脚配置为漏极开路。
1	CC_ORIENT_PD	R/W	0b：禁止CC_ORIENT_Px引脚上的内部下拉电阻 1b：使能CC_ORIENT_Px引脚上的内部下拉电阻
0	CC_ORIENT_PU	R/W	0b：禁止CC_ORIENT_Px引脚上的内部上拉电阻 1b：使能CC_ORIENT_Px引脚上的内部上拉电阻

表41列出了USB复位寄存器的详细信息。

表41： USB_RESET_REG

USB复位寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:10	Reserved	R	始终为0
9	USB_RELOAD_PCI	R/W	定义USB_RELOAD置1时在USB PCIe®端点上执行的配置；对应USB_PCIE_EP_ADDR_BASE。 0b：不会向USB_PCIE_EP_ADDR_BASE重载USB PCIe端点配置。系统配置硬件将尝试写入与特定配置相关的所有配置寄存器；子系统将阻止与USB_PCIE_EP_ADDR_BASE处的PCIe端点对应的任何写操作。 1b：将向USB_PCIE_EP_ADDR_BASE重载USB PCIe端点配置。
8	USB_RELOAD_EN	R/W	定义USB_RELOAD置1时在USB功能IP上执行的配置；USB功能IP对应USB_SUBSYSTEM_ADDR_BASE中的绝大多数子系统地址，只有对应USB_PCIE_EP_ADDR_BASE的地址除外，这些地址由USB_RELOAD_PCI位控制。 0b：不会重载USB功能IP配置。系统配置硬件将尝试写入与特定配置相关的所有配置寄存器；子系统将阻止与功能IP对应的任何写操作。 1b：将重载USB功能IP配置。
7:4	Reserved	R	始终为0
3	USB_HOST_RESET	R/W	使USB主机控制器保持复位状态。 0b：USB主机控制器未保持复位状态 1b：USB主机控制器保持复位状态
2	USB_RELOAD	R/W/SC	启动USB重载。
1	USB_LRST	R/W/SC	启动轻功能复位以复位除PCIe端点外的整个USB子系统。复位所有寄存器。
0	USB_SRST	R/W/SC	启动软复位以复位整个USB子系统，包括PCIe端点。复位所有寄存器。

表42列出了USB端口使能寄存器的详细信息。

表42： USB_PORT_ENABLE_REG

USB端口使能寄存器			默认值： PCI11400/PCI11414：0x0204_030F PCI11101：0x0001_0001 其他：N/A
Bit	**名称**	**R/W**	**说明**
31:28	Reserved	R	始终为0

表42： USB_PORT_ENABLE_REG（续）

USB端口使能寄存器			默认值： PCI11400/PCI11414：0x0204_030F PCI11101：0x0001_0001 其他：N/A
Bit	**名称**	**R/W**	**说明**
27:24	HOST_NUM_U3_PORT[3:0]	R/W	USB 3.2端口数 对于PCI11400和PCI11414，这些位的有效值范围从0000b（无端口）到0010b（2个端口）。 对于PCI11101，有效值为0000b（无端口）或0001b（1个端口）。 **注：**在对HOST_NUM_U3_PORT[3:0]位域进行任何更改后，USB主机控制器需要复位才能应用更改。
23:20	Reserved	R	始终为0
19:16	HOST_NUM_U2_PORT[3:0]	R/W	USB 2.0端口数 对于PCI11400和PCI11414，这些位的有效值范围从0001b（1个端口）到0100b（4个端口）。 对于PCI11101，只有0001b有效。 不得将这些位设置为0000b。至少需要一个USB 2.0端口。 **注：**在对HOST_NUM_U2_PORT[3:0]位域进行任何更改后，USB主机控制器需要复位才能应用更改。
15:10	Reserved	R	始终为0
9:8	USB_U3_PORT_ENABLE[1:0]	R/W	由硬件设置，用于指示是否已使能USB 3端口。必须在配置阶段更改这些位，不得在运行时动态更改。 Bit[0]对应端口1，Bit[1]对应端口2，Bit[2]对应端口3，Bit[3]对应端口4 0b：禁止端口 1b：使能端口 当该信号为1时，将停止报告端口上的连接/断开事件。该功能可用于安全场景，即硬件可强制禁止端口（无论xHCI驱动程序是否使能该端口）。 **注：**端口使能状态基于器件编号配置，但可通过配置设置进行改写。host_u2_port_disable随后用于禁止xHCI控制器上的相关端口。
7:4	Reserved	R	始终为0

表42： USB_PORT_ENABLE_REG（续）

USB端口使能寄存器			默认值： PCI11400/PCI11414：0x0204_030F PCI11101：0x0001_0001 其他：N/A
Bit	**名称**	**R/W**	**说明**
3:0	USB_U2_PORT_ENABLE[3:0]	R/W	由硬件设置，用于指示是否已使能USB 3端口。必须在配置阶段更改这些位，不得在运行时动态更改。 Bit[0]对应端口1，Bit[1]对应端口2，Bit[2]对应端口3，Bit[3]对应端口4 0b：禁止端口 1b：使能端口 当该信号为1时，将停止报告端口上的连接/断开事件。该功能可用于安全场景，即硬件可强制禁止端口（无论xHCI驱动程序是否使能该端口）。 **注：**端口使能状态基于器件编号配置，但可通过配置设置进行改写。host_u2_port_disable随后用于禁止xHCI控制器上的相关端口。

7.5 USB端口配置示例

7.5.1 禁止未使用的端口

表43-表46举例说明了PCI11101至PCI11414禁止USB端口的配置详细信息。

表43： PCI11414禁止USB端口4

编号	说明	注	寄存器地址	值
1	在 USB_PORT_ENABLE_REG 中禁止端口4	HOST_NUM_U3_PORT[3:0] = 10b HOST_NUM_U2_PORT[3:0] = 11b USB_U3_PORT_ENABLE[3:0] = 0011b USB_U2_PORT_ENABLE[3:0] = 0111b	基址 + 0x1_B004	0x0203_0307

表44： PCI11414禁止USB端口3和4

编号	说明	注	寄存器地址	值
1	在 USB_PORT_ENABLE_REG 中禁止端口3和4	HOST_NUM_U3_PORT[3:0] = 10b HOST_NUM_U2_PORT[3:0] = 10b USB_U3_PORT_ENABLE[3:0] = 0011b USB_U2_PORT_ENABLE[3:0] = 0011b	基址 + 0x1_B004	0x0202_0303

表45： PCI11414禁止USB端口2、3和4

编号	说明	注	寄存器地址	值
1	在 USB_PORT_ENABLE_REG 中禁止端口2、3和4	HOST_NUM_U3_PORT[3:0] = 01b HOST_NUM_U2_PORT[3:0] = 01b USB_U3_PORT_ENABLE[3:0] = 0001b USB_U2_PORT_ENABLE[3:0] = 0001b	基址 + 0x1_B004	0x0101_0101

表46： PCI11414禁止所有端口（禁止USB主机控制器）

编号	说明	注	寄存器地址	值
1	通过在PCIe交换芯片配置中禁止PCIe端口来禁止USB主机控制器	USB_Enable = 1'b0	基址 + 0x00E0	按CPN设置默认寄存器配置

7.5.2 支持的端口模式组合

1. PCI11414/PCI11400设置为端口配置模式1示例

有关支持的各种USB端口配置组合的完整说明，请参见第7.2节“支持的端口组合”。简而言之，端口配置模式1如下：

- 端口1：内置USB 3数据多路开关的USB 3.2 Gen 2 Type-C端口
- 端口2：USB 2.0 Type-A端口
- 端口3：USB 2.0 Type-A端口
- 端口4：USB 2.0 Type-A端口

表47： PCI11414/PCI11400设置为端口配置1

编号	说明	注	寄存器地址	值
1	在PORT_CFG_SEL_x中将端口1设置为内置USB-C数据多路开关的USB 3.2 Gen 2端口	CC_ENABLE = 1b，使能Type-C®逻辑 VBUS_INPUT_SEL[1:0] = 01b，使能VBUS电压监视引脚 MUX_EN = 1b，使能内部Type-C多路开关 C_MUX_SEL[1:0] = 11b，选择Type-C内部多路开关 PORT_TYPE[1:0] = 01b，选择USB Type-C端口模式	基址 + 0x5_6004	0x0000_0003
2	将端口1 PRT_CTL复用到引脚	GPIO70 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b1000	基址 + 0x0118	0x0000_0008

表47： PCI11414/PCI11400设置为端口配置1（续）

编号	说明	注	寄存器地址	值
3	将端口1 CC1复用到引脚	GPIO5 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b0011	基址 + 0x0014	0x0000_0003
4	将端口1 CC2复用到引脚	GPIO46 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b0110	基址 + 0x00B8	0x0000_0006
5	将端口1 VCONN1复用到引脚	GPIO48 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b0110	基址 + 0x00C0	0x0000_0006
6	将端口1 VCONN2复用到引脚	GPIO49 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b0110	基址 + 0x00C4	0x0000_0006
7	将端口1 VBUS_DISCHARGE复用到引脚	GPIO76 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b0111	基址 + 0x0130	0x0000_0007
8	将端口1 VBUS_MON复用到引脚	必须使能CC连接1。 如果CPN为11400或11414，该引脚将连接到CC1[2]	N/A	N/A
9	将端口2设置为Type-A USB 2.0端口		PCIE_SWITCH_ADDR_BASE + 0x0000_0008	0x0803
10	将端口2 PRT_CTL复用到引脚	GPIO71 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b1001	基址 + 0x011C	0x0000_0009
11	将端口3设置为Type-A USB 2.0端口		PCIE_SWITCH_ADDR_BASE + 0x0000_000C	0x0803
12	将端口3 PRT_CTL复用到引脚	GPIO83 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b1000	基址 + 0x014C	0x0000_0008
13	将端口4设置为Type-A USB 2.0端口		PCIE_SWITCH_ADDR_BASE + 0x0000_0010	0x0803
14	将端口4 PRT_CTL复用到引脚	GPIO81 Drive_Strength[1:0] = 2b00，以2 mA驱动 SELECT[3:0] = 4b1001	基址 + 0x0144	0x0000_0009

2. PCI11414/PCI11400设置为端口配置模式2

有关支持的各种USB端口配置组合的完整说明，请参见第7.2节“支持的端口组合”。简而言之，端口配置模式2如下：

- 端口1：带外部USB 3数据多路开关的USB 3.2 Gen 2 Type-C端口
- 端口2：带外部USB 3数据多路开关的USB 3.2 Gen 2 Type-C端口
- 端口3：USB 2.0 Type-A端口
- 端口4：USB 2.0 Type-A端口

3. PCI11414/PCI11400设置为端口配置模式3

有关支持的各种USB端口配置组合的完整说明，请参见第7.2节“支持的端口组合”。简而言之，端口配置模式3如下：

- 端口1：USB 3.2 Gen 2 Type-A端口
- 端口2：USB 3.2 Gen 2 Type-A端口
- 端口3：USB 2.0 Type-A端口
- 端口4：USB 2.0 Type-A端口

4. PCI11414/PCI11400设置为端口配置模式4

有关支持的各种USB端口配置组合的完整说明，请参见第7.2节“支持的端口组合”。简而言之，端口配置模式4如下：

- 端口1：USB 2.0 Type-C端口
- 端口2：USB 2.0 Type-C端口
- 端口3：USB 2.0 Type-A端口
- 端口4：USB 2.0 Type-A端口

5. PCI11414/PCI11400设置为端口模式5

有关支持的各种USB端口配置组合的完整说明，请参见第7.2节“支持的端口组合”。简而言之，端口配置模式5如下：

- 端口1：USB 2.0 Type-A端口
- 端口2：USB 2.0 Type-A端口
- 端口3：USB 2.0 Type-A端口
- 端口4：USB 2.0 Type-A端口

7.5.3 设置TYPE-C RP值

1. 将端口1 Type-C端口设置为3A

- 修改端口1 CC_CONFIG_Px寄存器的PWR_CAP[1:0]

基址 + 0x1_A220 = 0x0000_0003

2. 将端口2 Type-C端口设置为3A

- 修改端口2 CC_CONFIG_Px寄存器的PWR_CAP[1:0]

基址 + 0x1_A620 = 0x0000_0003

3. 将端口1 Type-C端口设置为1.5A

- 修改端口1 CC_CONFIG_Px寄存器的PWR_CAP[1:0]

基址 + 0x1_A220 = 0x0000_0002

4. 将端口2 Type-C端口设置为1.5A

- 修改端口2 CC_CONFIG_Px寄存器的PWR_CAP[1:0]

基址 + 0x1_A620 = 0x0000_0002

8.0 以太网实现

PCI1xxxx提供PCIe转以太网MAC功能，需连接外部以太网PHY。此外，PCI1xxxx还支持：

- PCIe 3.1a千兆位以太网控制器（单通道5 GTps）
- PCIe L1.1和L1.2子状态以及D3热和冷状态
- RGMII v1.3和v2.0
- SGMII
- 1000/2500BASE-X连接到外部以太网PHY。SGMII/1000/2500BASE-X引脚与RGMII RX引脚共用
- 过滤和LAN唤醒
- 多个TX和RX通道
- Microsoft RSS
- 1588v2和1588v2.1

PCI1xxxx以太网ID如下：

- 供应商ID：0x1055（Microchip Technology, Inc/SMSC）
- 器件ID：
  - PCI11010：0xA011
  - PCI11414：0xA041
- 分类ID：0x020000（“以太网控制器”）

8.1 章节

- 第8.2节“以太网MAC地址”
- 第8.3节“外部PHY复位”
- 第8.4节“接收过滤引擎（RFE）”
- 第8.5节“DMA控制器（DMAC）”
- 第8.6节“以太网接口寄存器”

8.2 以太网MAC地址

所有器件出厂时的MAC地址均为空。为确保在网络中正常工作，每个PCI1xxxx都需要配置惟一的MAC地址。

通常需向IEEE购买MAC地址段。这样可以确保MAC地址完全惟一。更多详细信息，请访问IEEE注册管理机构官网（https://standards.ieee.org/products-programs/regauth/）

另一个选项是通过驱动程序将MAC地址从主机PC传递给PCI1xxxx。

若要配置MAC地址，必须设置以下寄存器：

- MAC_RX_ADDRH
- MAC_RX_ADDRL

8.3 外部PHY复位

专用ENET_PHY_RESET_N引脚自动复位外部以太网PHY。该引脚作为复位序列的一部分自动置为有效。此外，PHY_RESET_N也可通过以太网PHY软件复位置为有效。

8.4 接收过滤引擎（RFE）

RFE从以太网MAC接收以太网帧、然后对这些帧进行处理，再将其传送给RX FCT。RFE负责过滤接收到的以太网帧、验证TCP/UDP/ICMP/IGMP和IP校验和以及移除VLAN标记。

当从MAC接收到帧时，RFE将获得帧数据和状态信息。帧处理完成后，RFE会用从MAC获取的状态信息封装其状态，并以RX命令A、RX命令B、RX命令C和RX命令D的形式将该信息（以及帧数据）传送给FCT。此外，RFE还会将接收时间戳从1588模块传送给FCT。

RFE使能时，可从帧中移除VLAN标记。VLAN标记的移除由接收过滤引擎控制寄存器（RFE_CTL）的使能VLAN标记移除位控制。如果该位置1，则标记会被移除。如果该位清零，则RFE不会以任何方式修改帧。

注： 如果一个帧中存在多个VLAN标记，则RFE仅会移除第一个标记（与MAC源地址相邻）。

RFE通过RX命令B提供第3层校验和（如果已使能）和VLAN ID，而RX命令A、RX命令C和RX命令D包含帧的状态。当RFE确定帧具有校验和错误时，它会将RX命令A的相应错误位置1以标识错误条件。

注： FCT不会回退FCT RX FIFO中校验和验证失败的帧。

RFE还将基于各种优先级方法、MAC源或目标地址或者基于Microsoft接收端缩放规范确定用来放置帧的正确RX FCT通道。为了通过RFE最大程度缩短延时，将对数据进行实时处理，然后并行存储到所有FIFO中。一旦确定了正确的目标FIFO，便会指示其他FIFO丢弃数据包。

8.5 DMA控制器（DMAC）

DMA控制器（DMAC）由独立的接收（RX）和发送（TX）DMAC、一系列仲裁器以及控制和状态寄存器（Control and Status Register，CSR）空间组成。TX DMAC将以太网帧从主机存储器传输到FIFO控制器（FCT），而RX DMAC将以太网帧从FCT传输到主机存储器。

RX和TX DMAC都分配有独立通道（4 RX和4 TX），可进行数据传输。RX和TX DMAC均利用描述符将数据从源地址高效移至目标地址，几乎不需要CPU干预。描述符是主机存储器中的数据结构，用于向DMAC通知数据缓冲区在主机存储器中的位置。对于RX DMAC，它还提供了一种机制，用于在DMA事务完成时将状态传送给CPU。主机负责设置描述符环并分配RX描述符缓冲区。系统根据需要分配TX描述符缓冲区并将其放入环中。每个通道都有自己的描述符环和数据缓冲区。描述符在芯片上高速缓存，以帮助减少主机总线延时。

可以将DMAC编程为在帧发送或接收传输完成等情况以及其他条件下将中断置为有效。

8.6 以太网接口寄存器

表48列出了ENET_SUBSYSTEM_ADDR_BASE = 0x000C_0000时可用的寄存器。

注： PCI1xxxx器件包含数千个寄存器，其中一部分寄存器保留供将来使用，另一部分寄存器的内容未纳入公开文档，因为其不涉及终端系统集成商用例（即：这些寄存器供硬件用于执行底层运行时任务，或在运行时由驱动程序直接使用/控制）。**请勿修改未纳入文档的寄存器的内容。**

表48： 以太网寄存器地址相对于ENET_SUBSYSTEM_ADDR_BASE的偏移量

寄存器名称	地址
ID_REV	0x0000
VLAN_TYPE	0x0008
HW_CFG	0x0010
PMT_CTL	0x0014
DP_SEL	0x0024
DP_CMD	0x0028
DP_ADDR	0x002C
DP_DATA_0	0x0030
DP_DATA_1	0x0034
DP_DATA_2	0x0038

表48： 以太网寄存器地址相对于ENET_SUBSYSTEM_ADDR_BASE的偏移量（续）

寄存器名称	地址
DP_DATA_3	0x003C
PCIE_CFG_ADDR	0x0048
PCIE_CFG_DATA	0x004C
MAC_CR	0x0100
MAC_RX	0x0104
MAC_TX	0x0108
MAC_FLOW	0x010C
MAC_RAND_SEED	0x0110
MAC_ERR_STS	0x0114
MAC_RX_ADDRH	0x0118
MAC_RX_ADDRL	0x011C
MII_ACCESS	0x0120
MII_DATA	0x0124
MII_RGMII_ID	0x0128
EEE_TX_LPI_REQUEST_DELAY_CNT	0x0130
WUCSR2	0x0600
RFE_CTL	0x0508
DMAC_CFG	0x0C00
DMAC_CMD	0x0C0C
DMAC_INT_SYS	0x0C10
SGMII_CTL	0x0728
INT_SYS	0x0780
INT_SET	0x0784
INT_EN_SET	0x0788
INT_EN_CLR	0x078C

表49列出了器件ID和版本寄存器的详细信息。

表49： ID_REV

器件ID和版本寄存器			默认值： 取决于器件/版本
Bit	**名称**	**R/W**	**说明**
31:16	CHIP_ID	R	器件ID。 以下是该寄存器的惟一有效值： PCI11010 = 0xA011 PCI11414 = 0xA041
15:0	CHIP_REV	R	器件版本 0x00A0对应版本A0硅片，以此类推。

表50列出了VLAN类型寄存器的详细信息。

表50： VLAN_TYPE

VLAN类型寄存器			默认值：0x0000_8100
Bit	**名称**	**R/W**	**说明**
31:16	Reserved	R	始终为0
15:0	VLAN_ETHERNET_TYPE	R/W	当使能VLAN标记插入时，该位域包含以太网类型的值。预期值为0x8100。通过更改该参数的值可支持专有VLAN标记。

表51列出了硬件配置寄存器的详细信息。

表51： HW_CFG

硬件配置寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:20	Reserved	R	始终为0
19	RST_PROTECT_PCIe	R/W	复位保护PCIe®配置 将该位置1可保护特定PCIe配置空间寄存器的选定位域不受POR_N以外的复位影响。 此外，复位保护还将禁止从器件级配置装载器装载默认值。 该位仅由POR_N复位。
18	FLR_DIS	R/W	功能级复位禁止 当收到功能级复位时，该位与D3 VAUX改写（D3_VAUX_OVR）位一起用于修改子系统复位操作。 当该位置1并且VAUX_DET引脚为高电平或D3_VAUX_OVR置1时，仅复位PCIe配置接口的功能部分。 其他情况下，复位子系统的整个功能部分。FLR永远不会复位PCIe接口的非功能部分（端点控制器、寄存器和链路等）。
17	EEE_PHY_LINK_CHANGE_SPEED_UP	R/W	EEE PHY链路接通加速 置1时，PHY链路需处于“接通”状态并持续200 µs才能请求和解码LPI。清零时，PHY链路需处于接通状态并持续1s。 **注：**软件只能在MAC控制寄存器（MAC_CR）中的节能以太网使能（EEEEN）位清零时更改该位。该位受复位保护（RST_PROTECT）位保护。

表51： HW_CFG（续）

硬件配置寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
16	EEE_TIMER_SPEED_UP	R/W	EEE定时器加速 置1时，EEE定时器和计数器（通常以µs为测量单位）将按照其时钟树的速度运行，如下所示： EEE_TX_LPI_REQUEST_DELAY_CNT——MAC TX时钟周期（3.2 ns/8 ns/40 ns） EEE_TX_LPI_AUTO_REMOVAL_DELAY_CNT——MAC TX时钟周期（3.2 ns/8 ns/40 ns） EEE RX LPI时间——晶振时钟周期 EEE TX LPI时间——晶振时钟周期 软件只能在MAC控制寄存器（MAC_CR）中的节能以太网使能（EEEEN）位清零时更改该位。
15	HOT_RESET_DIS	R/W	热复位禁止 当收到热复位时，该位与D3 VAUX改写（D3_VAUX_OVR）位一起用于修改子系统复位操作。 当该位置1并且VAUX_DET引脚为高电平或D3_VAUX_OVR置1时，仅复位PCIe接口及相关逻辑。 其他情况下，复位整个子系统。
14	D3_VAUX_OVR	R/W	D3 VAUX改写 当收到功能级复位时，该位与功能级复位禁止（FLR_DIS）位一起用于修改子系统复位操作。当收到热复位时，该位与热复位禁止（HOT_RESET_DIS）位一起用于修改子系统复位操作。 当FLR_DIS或HOST_RESET_DIS置1时，该位用于决定复位操作中是否涉及VAUX_DET。 当FLR_DIS和HOST_RESET_DIS清零时，该位无作用。
13	D3_RESET_DIS	R/W	D3Cold复位禁止 当退出D3Cold状态（PERST#置为无效）时，该位用于修改子系统复位操作。当该位置1且VAUX_DET引脚为高电平时，仅复位PCIe接口及相关逻辑。其他情况下，复位整个子系统。

表51： HW_CFG（续）

硬件配置寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
12	RST_PROTECT	R/W	复位保护 将该位置1可保护特定寄存器的选定位域不受POR_N以外的复位影响。 此外，复位保护还将禁止从器件级配置装载器装载或恢复默认值。 **注：**该位仅由POR_N复位。
11:6	RELOAD_TYPE	R/W	重载类型 这些位用于指定在执行CONFIG_RELOAD命令时使用哪部分配置数据。所有组合均有效。 bit 0——MAC地址 bit 1——PCIe配置寄存器和影子寄存器 bit 2——功能寄存器空间（不包括MAC地址）和系统（器件级）寄存器 bit 3——系统（器件级）寄存器 bit 4——配置脚寄存器 bit 5——保留
5	CONFIG_DL	R/W1C	已装载配置数据 该位置1时表示已找到有效配置，并且已正常完成寄存器编程。该位在上电后、各种复位后或CONFIG_RELOAD命令执行完毕后并成功装载数据之后置1。 **注：**该信号在配置过程中首次通过APB目标配置过滤器写入寄存器时即被置1。在使用CONFIG_RELOAD命令的情况下，该信号可能先于CONFIG_RELOAD位变为有效状态。 该信号基于APB目标配置过滤器过滤的APB写操作置1。即使随后的复位保护阻止写入实际寄存器位，该信号也会变为有效状态。

表51： HW_CFG（续）

硬件配置寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
4	CONFIG_RELOAD	R/W1S/SC	重载配置 将该位置1可指示器件配置重载器（EEPROM/OTP等）重载配置。 重载操作的目标模块由重载类型位域指定。 已装载配置数据（CONFIG_DL）位用于指示重载成功（至少成功编程了一个请求重载类型的寄存器）。该位将在重载命令完成后自清零。 **注：**如果设置了复位保护，则会阻止在复位后重载受保护的位。 在重载过程中，任何尝试读取CSR或器件级寄存器的操作都将返回全F值。这可用于指示正在进行重载。不得尝试写入CSR或器件级寄存器，否则可能会导致意外操作。
3:2	Reserved	R	始终为0
1	LRST	R/W1S/SC	轻复位 写入1会触发子系统轻软件复位。 轻复位不会影响PCIe接口或控制器。OTP/EEPROM的内容不会重载。该复位不会复位以太网PHY，不会重新调整IO焊盘或重新锁存配置脚。
0	SRST	R/W1S/SC	软复位 写入1会触发软件发起的子系统复位。如果使用外部以太网PHY，也将被复位。 软件复位将导致EEPROM的内容重载。复位序列执行期间，PCIe接口将断开连接。

表52列出了功耗管理控制控制寄存器的详细信息。

表52： PMT_CTL

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	D3_PLL_OVR	R/W	D3 PLL请求改写 置1时，PLL时钟请求（PLL_REQ）将在D3期间保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
30	D3_XTAL_OVR	R/W	D3晶振请求改写 置1时，晶振时钟请求（XTAL_REQ）将在D3期间保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
29:28	Reserved	R	始终为0
27	SGMII_ETH_ PHY_D3_COLD_OVR	R/W	SGMII和以太网PHY改写冷状态 置1时，如果子系统处于D3cold状态，SGMII/1000/2500BASE-X接口和外部以太网PHY保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
26	MAC_D3_TX_CLK_OVR	R/W	MAC TX时钟改写 置1时，如果子系统处于D3状态，以太网MAC和1588模块的发送时钟保持运行状态。 此外，晶振时钟请求（XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
25	MAC_D3_RX_CLK_OVR	R/W	MAC RX时钟改写 置1时，如果子系统处于D3状态，以太网MAC和1588模块的接收时钟保持运行状态。 此外，晶振时钟请求（XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
24	Reserved	R	始终为0

表52： PMT_CTL（续）

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
23	SGMII_ETH_PHY_D3_OVR	R/W	SGMII和以太网PHY改写 置1时，如果子系统处于D3状态，SGMII/1000/2500BASE-X接口和外部以太网PHY保持活动状态。 此外，晶振时钟请求（XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
22	INTC_D3_CLK_OVR	R/W	中断控制器时钟改写 置1时，如果子系统处于D3状态，中断控制器的系统时钟保持运行状态。 此外，PLL和晶振时钟请求（PLL_REQ和XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
21	DMAC_D3_CLK_OVR	R/W	DMA控制器时钟改写 置1时，如果子系统处于D3状态，DMA控制器的系统时钟保持运行状态。 此外，PLL和晶振时钟请求（PLL_REQ和XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
20	1588_D3_CLK_OVR	R/W	1588时钟改写 置1时，如果子系统处于D3状态，1588模块的系统时钟保持运行状态。 RX与TX网络时钟分开单独控制。 此外，PLL和晶振时钟请求（PLL_REQ和XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
19	MAC_D3_CLK_OVR	R/W	MAC时钟改写 置1时，如果子系统处于D3状态，以太网MAC的系统时钟保持运行状态。 RX与TX网络时钟分开单独控制。 此外，PLL和晶振时钟请求（PLL_REQ和XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。

表52： PMT_CTL（续）

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
18	RX_FCT_RFE_D3_CLK_OVR	R/W	RX FCT和RFE D3时钟改写 置1时，如果子系统处于D3状态，RX FCT和RFE的系统时钟保持运行状态。 此外，PLL和晶振时钟请求（PLL_REQ和XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
17	TX_FCT_LSO_D3_CLK_OVR	R/W	TX FCT和LSO D3时钟改写 置1时，如果子系统处于D3状态，TX FCT和LSO的系统时钟保持运行状态。 此外，PLL和晶振时钟请求（PLL_REQ和XTAL_REQ）在D3期间也保持活动状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
16:14	Reserved	R	始终为0
13	EEE_WAKEUP_EN	R/W	EEE唤醒使能 使能EEE唤醒状态（EEE_WUPS）作为唤醒事件。 如果恢复清零远程唤醒使能（RES_CLR_WKP_EN）位置1，当因唤醒事件发送PM_PME报文时，该位将自动清零。 **注：**该位受复位保护（RST_PROTECT）位保护。
12	EEE_WUPS	R/W1C	EEE唤醒状态 该位指示当前唤醒事件是否由EEE引发。该位与唤醒状态（WUPS）位一起使用。有关编码，请参见WUPS位域。 当节能以太网TX唤醒（EEE_TX_WAKE）或节能以太网RX唤醒（EEE_RX_WAKE）位置1时，该位置1。 无论EEE唤醒使能（EEE_WAKEUP_EN）位中的值为何，该位都将置1。 如果恢复清零远程唤醒状态（RES_CLR_WKP_STS）位置1，当因唤醒事件发送PM_PME报文时，该位将清零。 更多详细信息，请参见RES_CLR_WKP_STS位。
11:10	Reserved	R	始终为0

表52： PMT_CTL（续）

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
9	RES_CLR_WKP_STS	R/W	恢复清零远程唤醒状态 置1时，如果因唤醒事件发送PM_PME报文，以下状态信号将清零： 唤醒控制和状态寄存器1（WUCSR1）： • 已收到RFE唤醒帧（RFE_WAKE_FR） • 已收到完美DA帧（PFDA_FR） • 已收到远程唤醒帧（WUFR） • 已收到魔术包（MPR） • 已收到广播帧（BCAST_FR） • 节能以太网TX唤醒（EEE_TX_WAKE） • 节能以太网RX唤醒（EEE_RX_WAKE） 唤醒控制和状态寄存器2（WUCSR2）： • 已收到IPv6 TCP SYN数据包（IPV6_TCPSYN_RCD） • 已收到IPv4 TCP SYN数据包（IPV4_TCPSYN_RCD） 功耗管理控制寄存器（PMT_CTL）： • 唤醒状态（WUPS） • EEE唤醒状态（EEE_WUPS） 主机发起的唤醒不会清零唤醒状态信号。 清零时，即使主机清零PME_Status位，也不会清零唤醒状态信号。 **注：**该位受复位保护（RST_PROTECT）位保护。

表52： PMT_CTL（续）

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
8	RES_CLR_WKP_EN	R/W	恢复清零远程唤醒使能 置1时，如果因唤醒事件发送PM_PME报文，以下唤醒使能信号将清零： 唤醒控制和状态寄存器1（WUCSR1）： • RFE唤醒使能（RFE_WAKE_EN） • 完美DA唤醒使能（PFDA_EN） • 唤醒帧使能（WUEN） • 魔术包使能（MPEN） • 广播唤醒使能（BCAST_EN） • 节能以太网TX唤醒使能（EEE_TX_WAKE_EN） • 节能以太网RX唤醒使能（EEE_RX_WAKE_EN） 唤醒控制和状态寄存器2（WUCSR2）： • IPv6 TCP SYN唤醒使能（IPV6_TCPSYN_WAKE_EN） • IPv4 TCP SYN唤醒使能（IPV4_TCPSYN_WAKE_EN） 功耗管理控制寄存器（PMT_CTL） • LAN唤醒使能（WOL_EN） • 以太网PHY中断允许（ETH_PHY_WAKE_EN） • EEE唤醒使能（EEE_WAKEUP_EN） • GPIO唤醒使能（GPIO_WAKEUP_EN） 主机发起的唤醒不会清零唤醒使能信号。 清零时，即使主机清零PME_Status位，也不会清零唤醒使能信号。 **注：**该位受复位保护（RST_PROTECT）位保护。
7	READY	R	器件就绪 READY位用于指示子系统何时准备好正常工作。 READY位在以下情况下保持无效状态。 • 等待千兆位以太网PHY复位序列完成，PHY变为工作状态。对于外部PHY SKU，这可能需要超过125 ms的时间，具体取决于子系统的配置。 • 等待从器件级配置重载器重载内容。

表52： PMT_CTL（续）

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
6	RX_FIFO_L1_EXIT_EN	R/W	RX FIFO L1-ASPM退出使能 该位清零时，基于PCIe®活动退出L1-ASPM链路状态。 该位置1时，如果RX FIFO中存储了数据，也会退出L1-ASPM链路状态。 **注：**该位受复位保护（RST_PROTECT）位保护。
5	EXT_PHY_RDY_EN	R/W	外部PHY就绪延时使能 该位用于使能125 ms的额外等待时间，以允许外部以太网PHY做好准备，以便将器件就绪（READY）位置为有效。 **注：**该位受复位保护（RST_PROTECT）位保护。
4	ETH_PHY_RST	R/W1S/SC	以太网PHY复位 向该位写入1会复位以太网PHY。内部逻辑会自动使PHY保持复位状态并至少持续2 ms。当PHY从复位状态释放时，该位自动清零。该位为高电平时，将忽略对该位的所有写操作。 该位置1后，器件就绪（READY）位将立即清零。当PHY处于工作状态时，器件就绪（READY）位将置为有效。
3	WOL_EN	R/W	LAN唤醒使能 使能WOL作为唤醒事件，其中包含以下内容： • 唤醒帧 • 魔术包 • 完美DA匹配 • 广播帧 • IPv4 TCP SYN数据包 • IPv6 TCP SYN数据包 如果恢复清零远程唤醒使能（RES_CLR_WKP_EN）位置1，当因唤醒事件发送PM_PME报文时，该位将自动清零。 **注：**该位受复位保护（RST_PROTECT）位保护。

表52： PMT_CTL（续）

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
2	ETH_PHY_WAKE_EN	R/W	以太网PHY中断允许 允许以太网PHY中断和EDPD电能检测作为唤醒事件。 如果恢复清零远程唤醒使能（RES_CLR_WKP_EN）位置1，当因唤醒事件发送PM_PME报文时，该位将自动清零。 **注：**该位受复位保护（RST_PROTECT）位保护。

表52： PMT_CTL（续）

功耗管理控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
1:0	WUPS	R/W1C	唤醒状态 该位域与EEE唤醒状态（EEE_WUPS）位一起用于指示当前唤醒事件的原因。WUPS位通过向相应位写入1来清零。这些位的编码如下： EEE_WUPS, WUPS[1:0]： • 0,00：未检测到唤醒事件 • X,X1：以太网PHY唤醒事件 • X,1X：唤醒帧、魔术包、完美DA匹配广播帧、IPv4/IPv6 TCP SYN数据包或RFE帧 • 1,XX：EEE接收唤醒（EEE_RX_WAKE）/EEE发送唤醒（EEE_TX_WAKE） 可能有多个位同时置1以指示发生了多个事件。 如果以下任何一位置1，WUPS[1]都将置1： • 已收到RFE唤醒帧（RFE_WAKE_FR） • 已收到完美DA帧（PFDA_FR） • 已收到远程唤醒帧（WUFR） • 已收到魔术包（MPR） • 已收到广播帧（BCAST_FR） • 已收到IPv6 TCP SYN数据包（IPV6_TCPSYN_RCD_WK） • 已收到IPv4 TCP SYN数据包（IPV4_TCPSYN_RCD_WK） 无论LAN唤醒使能（WOL_EN）中的值为何，WUPS[1]都将置1。 无论以太网PHY中断允许（ETH_PHY_WAKE_EN）中的值为何，WUPS[0]都将置1。 如果恢复清零远程唤醒状态（RES_CLR_WKP_STS）位置1，当因唤醒事件发送PM_PME报文时，WUPS[1]和WUPS[0]将清零。 更多详细信息，请参见RES_CLR_WKP_STS位。

表53列出了数据端口选择寄存器的详细信息。

表53： DP_SEL

数据端口选择寄存器			默认值： 0x8000_0000
Bit	**名称**	**R/W**	**说明**
31	DPRDY	R	数据端口就绪 数据端口就绪位用于指示数据端口RAM访问何时完成。在读操作的情况下，该位指示读取的数据何时已存入数据端口数据寄存器。 1b = 数据端口就绪。 0b = 数据端口正忙于处理事务。
30	DP_EN	R/W	数据端口使能 该位置1时，由选择（SEL）位域指示的RAM将与其功能断开连接，随后可用于数据端口接口。
29:5	Reserved	R	始终为0
4:0	SEL	R/W	选择 选择要访问的RAM。 11100b——LSO_RAM_3 11011b——LSO_RAM_2 11010b——LSO_RAM_1 11001b——DMAC_REORDER_BUFFER_3 11000b——DMAC_REORDER_BUFFER_2 10111b——DMAC_REORDER_BUFFER_1 10110b——PCIE_DRCV_RAM 10101b——PCIE_HRCV_RAM 10100b——PCIE_SOT_RAM 10011b——PCIE_RETRY_RAM 10010b——DMAC_TX_RAM_3 10001b——DMAC_TX_RAM_2 10000b——DMAC_TX_RAM_1 01111b——DMAC_TX_RAM_0 01110b——DMAC_RX_RAM_3 01101b——DMAC_RX_RAM_2 01100b——DMAC_RX_RAM_1 01011b——DMAC_RX_RAM_0 01010b——DMAC_REORDER_BUFFER_0 01001b——FCT_TX_RAM_3 01000b——FCT_TX_RAM_2 00111b——FCT_TX_RAM_1 00110b——FCT_TX_RAM_0 00101b——FCT_RX_RAM_3 00100b——FCT_RX_RAM_2 00011b——FCT_RX_RAM_1 00010b——FCT_RX_RAM_0 00001b——RFE_RAM（VLAN和DA哈希表（VHF RAM）） 00000b——LSO_RAM_0（LSO报头RAM通道0） 访问所选RAM的数据端口会干扰该RAM的正常使用。对于PCIe® RAM，访问数据端口将导致PCIe功能异常。

表54列出了数据端口命令寄存器的详细信息。

表54： DP_CMD

数据端口命令寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:1	Reserved	R	始终为0
0	DATA_PORT_WRITE	R/W	写入该位会启动数据端口访问。 1b——写操作 0b——读操作

表55列出了数据端口地址寄存器的详细信息。

表55： DP_ADDR

数据端口地址寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:14	Reserved	R	始终为0
13:0	DATA_PORT_ADDR	R/W	数据端口地址

表56列出了数据端口数据0寄存器的详细信息。

表56： DP_DATA_0

数据端口数据0寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:0	DATA_PORT_DATA[31:0]	R/W	数据端口数据[31:0]

表57列出了数据端口数据1寄存器的详细信息。

表57： DP_DATA_1

数据端口数据1寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:0	DATA_PORT_DATA[63:32]	R/W	数据端口数据[63:32]

表58列出了数据端口数据2寄存器的详细信息。

表58： DP_DATA_2

数据端口数据2寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:0	DATA_PORT_DATA[95:64]	R/W	数据端口数据[95:64]

表59列出了数据端口数据3寄存器的详细信息。

表59： DP_DATA_3

数据端口数据3寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:0	DATA_PORT_DATA[127:96]	R/W	数据端口数据[127:96]

表60列出了PCIe配置地址寄存器的详细信息。

表60： PCIE_CFG_ADDR

PCIe®配置地址寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	PCIE_CONFIG_BUSY	R	PCIe配置繁忙 表示正在写入PCIe配置空间。 该位在写入PCIe配置数据寄存器（PCIE_CFG_DATA）时置1，在PCIe配置空间写操作完成后清零。
30:16	Reserved	R	始终为0
15	PCIE_CONFIG_SHADOW_SEL	R/W	PCIe配置影子选择 选择PCIe配置模块中的特殊影子寄存器。
14:12	Reserved	R	始终为0
11:0	PCIE_CONFIG_ADDR	R/W	PCIe配置地址[11:0] 指定PCIe配置寄存器字节地址。

表61列出了PCIe配置数据寄存器的详细信息。

表61： PCIE_CFG_DATA

PCIe®配置数据寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:0	PCIE_CFG_DATA	W	PCIe配置数据

表62列出了MAC控制寄存器的详细信息。

表62： MAC_CR

MAC控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:24	Reserved	R	始终为0

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表62： MAC_CR（续）

MAC控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
20	AUTO_CLR_EEEEN	R/W NASR	链路丢失时清零节能以太网使能位 该位置1时，链路丢失后会自动清零EEEEN位。 **注：**该位受复位保护（RST_PROTECT）位保护。
19	ASD_LS	R/W NASR	自动速度、双工和链路源 该位用于选择自动速度、双工和链路的来源 1b——PHY接口 0b——速度基于接收时钟计算得出，双工和链路基于引脚。 **注：**该位受复位保护（RST_PROTECT）位保护。
18	EEE_TX_CLK_STOP_EN	R/W NASR	节能以太网TX时钟停止使能 置1时，MAC将在TX LPI期间停止向PHY发送RGMII_TXC。 **注：**软件或OTP/EEPROM内容只能在PHY MMD寄存器3.1中的时钟停止功能位指示PHY能够允许停止TX时钟时才将该位置1。该位受复位保护（RST_PROTECT）位保护。
17	EEEEN	R/W NASR	节能以太网使能 置1时，使能MAC中的节能以太网操作。清零时，禁止节能以太网操作。 即使发送器使能（TXEN）位清零，MAC也会产生LPI请求；即使接收器使能（RXEN）位清零，MAC也会解码LPI指示。 尽管节能以太网使能（EEEEN）位负责生成向PHY发送的低功耗空闲信号和解码从PHY接收的低功耗空闲信号，但并不控制PHY内的自动协商EEE下一页报文传输。 该位不会立即启动LPI发送。LPI的延时取决于EEE PHY链路接通加速（EEE_PHY_LINK_CHANGE_SPEED_UP）位的设置。 **注：**该位受复位保护（RST_PROTECT）位保护。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表62： MAC_CR（续）

MAC控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
16	EEE_TX_LPI_AUTO_REMOVAL_EN	R/W NASR	节能以太网TX LPI自动移除使能 置1时，允许根据预期的周期性发送事件自动将LPI置为无效。等待时间在EEE TX LPI自动移除延时寄存器（EEE_TX_LPI_AUTO_REMOVAL_DELAY）中指定。 该时间从MAC启动LPI信号传输的时刻开始计时。 **注：**软件只能在节能以太网使能（EEEEN）位清零时更改该位。该位受复位保护（RST_PROTECT）位保护。
15	FLS	R/W NASR	强制链路状态 该位用于强制链路状态以指示PHY具有有效链路。 0b——链接状态通过第15.0章“EEPROM存储器编程（读写操作）”所述的正常自动方法确定。 1b——强制链接状态有效。
14	LSP	R/W NASR	链路状态极性 该位用于指示ENET_LINK引脚的极性。 0b——低电平有效——ENET_LINK置为低电平表示PHY具有有效链路。 1b——高电平有效——ENET_LINK置为高电平表示PHY具有有效链接。 **注：**当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1时，软件不得修改该位。该位受复位保护（RST_PROTECT）位保护。
13	ADP	R/W NASR	自动双工极性 该位用于指示ENET_DUPLEX引脚的极性。 0b——低电平有效——ENET_DUPLEX置为低电平表示PHY处于全双工模式。 1b——高电平有效——ENET_DUPLEX置为高电平表示PHY处于全双工模式。 **注：**当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1）时，软件不得修改该位。该位受复位保护（RST_PROTECT）位保护。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表62： MAC_CR（续）

MAC控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
12	ADD	R/W NASR	自动双工检测 置1时，MAC忽略双工模式（DPX）位的设置并自动确定双工工作模式。MAC使用ENET_DUPLEX引脚来完成模式检测并通过双工模式（DPX）位报告最后确定的状态。 复位时，双工模式（DPX）位的设置决定双工工作模式。 **注：**当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1）时，软件不得修改该位。该位受复位保护（RST_PROTECT）位保护。
11	ASD	R/W NASR	自动速度检测 置1时，MAC忽略MAC配置（CFG）位域的设置并自动确定工作速度。MAC采样接收时钟以完成速度检测，并通过MAC配置（CFG）位域报告最后确定的速度。 复位时，MAC配置（CFG）位域的设置决定工作速度。 **注：**当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1）时，软件不得修改该位。该位受复位保护（RST_PROTECT）位保护。
10	INT_LOOP	R/W NASR	内部环回工作模式 在TX数据路径接口与RX数据路径接口之间环回数据。这仅适用于全双工模式。在内部环回模式下，TX帧由内部GMII接口接收并发送回MAC，而不发送到PHY。 0b——正常模式 1b——使能内部环回 **注：**当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1）时，软件不得修改该位。
9:8	Reserved	R	始终为0

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表62： MAC_CR（续）

MAC控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
7:6	BOLMT	R/W NASR	后退限制 BOLMT位允许用户在宽松或激进模式下设置后退限制。根据IEEE 802.3，MAC在检测到冲突后必须等待随机数[r]个时隙，其中： （公式1）0 < r < 2K 指数K取决于当前待发送帧已重试的次数，具体如下： （公式2）K = min. (n, 10)，其中n是当前重试次数。 如果某个帧已重试3次，则K = 3，r = 8个时隙（最大值）。如果该帧已重试12次，则K = 10，r = 2(10) = 20个时隙（最大值）。 线性反馈移位寄存器（Linear Feedback Shift Register，LFSR）计数器仿真随机数发生器以从中获取r值。检测到冲突后，使用当前帧的当前重试次数获取K值（公式2）。该K值转换为从LFSR计数器中提取的位数。如果K值为3，MAC取LFSR计数器前三位的值，并使用该值针对时隙个数递减计数至零。这样最多可使MAC等待8个时隙。为了给用户提供更大的灵活性，BOLMT值强制将从LFSR计数器提取的位数设为下表中的预定值。 因此，如果K值 = 10，MAC将检查BOLMT，如果为00，则使用LFSR计数器的低十位进行等待倒计时。如果BOLMT为10，将仅使用前四位的值进行等待倒计时，以此类推。 时隙 = 512个位时间 **注：**当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1）时，软件不得修改该位。该位受复位保护（RST_PROTECT）位保护。
5	CNTR_RST	R/W1S/ SC	计数器复位 该位置1时，所有统计计数器和计满返回状态都将复位。该位自清零。
4	CNTR_WEN	R/W NASR	计数器写使能 该位置1时，所有统计计数器都可写，以便进行测试。清零时，统计计数器为只读。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表62： MAC_CR（续）

MAC控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
3	DPX	NASR	双工模式 自动双工检测（ADD）位复位时，该位决定MAC的双工工作模式。自动双工检测（ADD）位置1时，该位为只读并报告最后确定的双工工作模式。 0b——MAC处于半双工模式 1b——MAC处于全双工模式 **注：**当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1）时，软件不得修改该位。 如果检测到的速度或手动设置的速度为1000/2500 Mbps，则无论该位的设置为何，都禁止半双工模式。该位受复位保护（RST_PROTECT）位保护。
2:1	CFG	NASR	MAC配置 自动速度检测（ASD）位复位时，该位域决定MAC的工作速度。自动速度检测（ASD）位置1时，该位域为只读并报告最后确定的工作速度。 00b——10 Mbps 01b——100 Mbps 10b/11b——1000 Mbps/2500 Mbps **注：**自动速度检测（ASD）位复位时，值2和3等效，实际工作速度由SGMII/1000/2500BASE-X接口提供的时钟速度决定。软件可能会使用惟一值作为标志。自动速度检测（ASD）位置1时，对于1000 Mbps和2500 Mbps两种速率，报告的速度值均为2。当MAC的接收器或发送器使能（接收器使能（RXEN）或发送器使能（TXEN）位置1）时，软件不得修改该位。该位域受复位保护（RST_PROTECT）位保护。
0	MRST	R/W1S/SC	MAC复位 0b——使能MAC 1b——复位MAC 标记为NASR的寄存器位不会复位。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表63列出了MAC接收器寄存器的详细信息。

表63： MAC_RX

MAC接收器寄存器			默认值： 0x05EE_0068
Bit	**名称**	**R/W**	**说明**
31:30	Reserved	R	始终为0
29:16	MAX_SIZE	R/W NASR	最大帧大小 定义接收帧的最大大小。超过该大小的帧将被中止。 **注：**可保证支持的最大帧大小为9,220字节。如果该位域的值大于15360则无效，因为看门狗定时器会生效并在15361处截断帧。使能MAC的接收器（MAC接收寄存器（MAC_RX）中的接收使能（RXEN）位置1）时，硬件会阻止修改该位域。
15:7	Reserved	R	始终为0
6	LEN_FLD_LT_CHK	R/W NASR	长度字段小于校验值 0b = 允许大于64字节的帧具有小于接收帧长度（减去18字节）的长度/类型字段值。 1b = 中止大于64字节且长度/类型字段值小于接收帧长度（减去18字节）的帧并对其进行计数。 **注：**使能MAC的接收器（MAC接收寄存器（MAC_RX）中的接收使能（RXEN）位置1）时，硬件会阻止修改该位。该位受复位保护（RST_PROTECT）位保护。
5	WTL	R/W NASR	看门狗截断长度 0b = MAC在MAC_RX.MAX_SIZE+1或15361（以较小者为准）处截断RX帧。传递给FCT的截断接收帧的RxCmdA将具有MAC_RX.MAX_SIZE+1或15361（以较小者为准）的长度值。如果MAC_RX.MAX_SIZE设置为15360或更大值，RWT位将置1。如果MAC_RX.MAX_SIZE设置为15360或更小值，LONG位将置1。截断很可能导致FCS错误（FCS位置1）。 1b = MAC在15361处截断接收RX帧。传递给FCT的截断接收帧的RxCmdA将具有15361的长度值并将RWT位置1。 如果MAC_RX.MAX_SIZE设置为15360或更小值，LONG位将置1。截断很可能导致FCS错误（FCS位置1）。 **注：**使能MAC的接收器（MAC接收寄存器（MAC_RX）中的接收使能（RXEN）位置1）时，硬件会阻止修改该位。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表63： MAC_RX（续）

MAC接收器寄存器			默认值： 0x05EE_0068
Bit	**名称**	**R/W**	**说明**
4	FCS_STRIPPING	R/W NASR	FCS移除 置1时，MAC将移除所有接收帧的FC（最后4个字节）。 **注：**使能MAC的接收器（MAC接收寄存器（MAC_RX）中的接收使能（RXEN）位置1）时，硬件会阻止修改该位。
3	LFCD	R/W NASR	长度字段校验禁止 0b = 中止所有帧长度与长度/类型字段中指定的长度值不一致的帧。 1b = 忽略长度字段。 清零时，帧长度与长度/类型字段中指定的长度值不一致的帧被视为错误，将在RX FIFO处被丢弃（除非RX FIFO配置为存储错误帧）并进行计数。唤醒帧不会触发唤醒事件，ARP/NS卸载请求不会触发响应。 置1时，仍会报告帧长度错误并对错误帧进行计数，但不会丢弃帧。唤醒帧可能触发唤醒事件，ARP/NS卸载请求可能触发响应。 **注：**使能MAC的接收器（MAC接收寄存器（MAC_RX）中的接收使能（RXEN）位置1）时，硬件会阻止修改该位。该位受复位保护（RST_PROTECT）位保护。
2	FSE	R/W NASR	VLAN帧大小强制 0b = 中止所有大于最大帧的帧。 1b = 中止所有大于最大帧的非VLAN帧。中止所有带单个VLAN标记且大于最大帧 + 4的帧。中止所有带两个VLAN标记且大于最大帧 + 8的帧。 **注：**使能MAC的接收器（MAC接收寄存器（MAC_RX）中的接收使能（RXEN）位置1）时，硬件会阻止修改该位。
1	RXD	R/W NASR	已禁止接收器 该位用于指示已通过清零接收器使能（RXEN）位成功禁止MAC的接收器。当接收器使能（RXEN）位从1变为0（使能变为禁止）触发的硬件禁止过程完成时，该位置1。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表63： MAC_RX（续）

MAC接收器寄存器			默认值：0x05EE_0068
Bit	**名称**	**R/W**	**说明**
0	RXEN	R/W NASR	接收器使能 置1时，MAC的接收器处于使能状态，将从PHY接收帧。复位时，MAC的接收器处于禁止状态，不会从PHY接收任何帧。 如果该位在接收帧的过程中置为无效，接收中的帧仍将完成接收。完成后，禁止MAC的接收器并将接收器禁止（RXD）位置为有效。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表64列出了MAC发送寄存器的详细信息。

表64： MAC_TX

MAC发送寄存器			默认值：0x0000_0040
Bit	**名称**	**R/W**	**说明**
31:9	Reserved	R	始终为0
8:4	WRR_WTS	R/W NASR	加权循环权重表大小 该位域用于指定加权循环权重表的大小。有效值范围为1到16。 **注：**超出该范围的值可能导致意外结果。
3	TX_FIFO_SRV	R/W NASR	TX FIFO服务模式 置1时，MAC严格按照固定优先级顺序服务TX FIFO。清零时，按照加权循环（Weighted Round Robin，WRR）顺序。
2	BFCS	R/W NASR	错误FCS 置1时，MAC的发送器将在所有发送帧上附加错误FCS。该功能对于诊断来说很实用。 该功能只能与TX命令A的插入FCS和填充功能结合使用。
1	TXD	R/W1C NASR	已禁止发送器 该位用于指示已通过清零发送器使能（TXEN）位成功禁止MAC的发送器。当发送器使能（TXEN）位从1变为0（使能变为禁止）触发的硬件禁止过程完成时，该位置1。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表64： MAC_TX（续）

MAC发送寄存器			默认值：0x0000_0040
Bit	**名称**	**R/W**	**说明**
0	TXEN	R/W NASR	发送器使能 置1时，MAC的发送器处于使能状态，将从发送缓冲区向线缆发送帧。复位时，MAC的发送器处于禁止状态，不会发送任何帧。 如果在全双工模式下该位在发送帧的过程中清零，帧仍将完成发送。完成后，禁止MAC的发送器并将发送器禁止（TXD）位置为有效。在半双工模式下，如果在发送过程中发生冲突，应中止帧。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表65列出了流控制寄存器的详细信息。

表65： MAC_FLOW

流控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	FORCE_FC	R/W1S/SC	强制发送TX流控制帧 该位用于强制发送TX流控制帧。写入1会启动帧发送。生成的帧将包含暂停时间值。完成帧发送后，MAC将清零该位。
30	TX_FCEN	R/W NASR	TX流控制使能 置1时，根据RX FIFO中的高低阈值水印使能发送MAC流控制功能，如本节所述。 **注：**必须先编程FCT流控制x阈值寄存器（FCT_FLOW_x）中的阈值，然后再将该位置1。反之，在将FCT流控制x阈值寄存器（FCT_FLOW_x）中的阈值编程为0之前，必须先清零该位。
29	RX_FCEN	R/W NASR	RX流控制使能 置1时，使能接收MAC流控制功能。MAC解码所有传入帧以识别控制帧。如果收到有效的控制帧（PAUSE命令），将在指定时间（解码的暂停时间x时隙）内禁止发送器。 未置1时，禁止MAC流控制功能。MAC不解码识别控制帧。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表65： MAC_FLOW（续）

流控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
28	FPF	R/W NASR	转发暂停帧 使能将收到的暂停帧传递到RX数据路径接口。 0b = 丢弃收到的暂停帧。 1b = 将收到的暂停帧传递到RX数据路径接口。 **注：** 流控制适用于MAC设置为全双工模式时。
27:16	Reserved	R	始终为0
15:0	FCPT	R/W NASR	暂停时间 该位域用于指示在控制帧的暂停时间字段中使用的值。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表66列出了随机数种子值寄存器的详细信息。

表66： MAC_RAND_SEED

随机数种子值寄存器			默认值：0x0000_9876
Bit	**名称**	**R/W**	**说明**
31:16	Reserved	R	始终为0
15:0	RAND_SEED	R/W NASR	随机数种子 MAC随机数发生器种子值。该寄存器的内容是用于LFSR（线性反馈移位寄存器）计数器的种子值，该计数器用于仿真MAC TX后退定时器逻辑中的随机数发生器。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表67列出了错误状态寄存器的详细信息。

表67： MAC_ERR_STS

错误状态寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:11	Reserved	R	始终为0
10	LEN_ERR	R/W1C NASR	长度字段错误 指示帧长度与长度/类型字段中指定的长度值不一致。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表67： MAC_ERR_STS（续）

错误状态寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
9	RXERR	R/W1C NASR	RX错误 指示在帧接收期间（包括前导码和SFD期间）检测到接收错误（PHY RX错误信号置为有效）。 **注：**无论是否有其他错误条件（对齐错误、FCS错误和长度错误等），该位都将置1。
8	FERR	R/W1C NASR	FCS错误 已收到FCS错误帧。无论帧长度为何，该位都将置1。 **注：**如果ALERR位置1，该位不会置1。仅考虑长度为8位整数倍的帧。
7	LFERR	R/W1C NASR	帧过大错误 已收到超过最大帧大小的帧。
6	RFERR	R/W1C NASR	过短帧/短帧错误 已收到过短帧或短帧。无论FCS状态为何，该位都将置1。
5	RWTERR	R/W1C NASR	接收看门狗定时器超时 该位置1时表示收到的帧长度超过15,360字节，已被MAC截断。
4	ECERR	R/W1C NASR	冲突过量错误 因发生16次冲突而中止发送帧。
3	ALERR	R/W1C NASR	对齐错误 在收到的帧上检测到对齐错误（长度不是8位整数倍且存在FCS错误或RX符号错误（包括前导码和SFD期间））。
2	URERR	R/W1C NASR	数据不足错误 MAC的发送数据路径出现数据不足。
1:0	Reserved	R	始终为0

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表68列出了MAC接收器地址高字节寄存器的详细信息。

表68： MAC_RX_ADDRH

MAC接收器地址高字节寄存器			默认值：0x0000_FFFF
Bit	**名称**	**R/W**	**说明**
31:16	Reserved	R	始终为0
15:0	MAC_PHY_ADDR[47:32]	R/W NALR NASR	物理地址[47:32] 该位域包含MAC物理地址的高16位[47:32]。 **注：**该位域受复位保护（RST_PROTECT）位保护。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表69列出了MAC接收器地址低字节寄存器的详细信息。

表69： MAC_RX_ADDRL

MAC接收器地址低字节寄存器			默认值：0xFFFF_FFFF
Bit	**名称**	**R/W**	**说明**
31:0	MAC_PHY_ADDR[31:0]	R/W NALR NASR	物理地址[31:0] 该位域包含MAC物理地址的低32位[31:0]。 **注：**该位域受复位保护（RST_PROTECT）位保护。

注： NASR标识仅在MAC控制寄存器（MAC_CR）的MAC复位（MRST）位置1时适用。标记为NASR的寄存器位通过其他芯片级软件发起的复位进行复位。

表70列出了MII访问寄存器的详细信息。

表70： MII_ACCESS

MII访问寄存器			默认值：0x0000_0800
Bit	**名称**	**R/W**	**说明**
31:16	Reserved	R	始终为0
15:11	PHY_ADDR	R/W NASR	PHY地址 该位域用于指定PHY地址。该位域必须设置为外部PHY的地址。
10:6	MII_REGISTER_INDEX	R/W NASR	MII寄存器索引 对于Clause 22操作，这些位用于选择PHY中所需的MII寄存器。 对于Clause 45操作，这些位用于选择PHY中所需的MMD。

表70： MII_ACCESS（续）

MII访问寄存器			默认值： 0x0000_0800
Bit	**名称**	**R/W**	**说明**
5:4	MDC_CYCLE_TIME	R/W NASR	MDC周期时间 该位域用于选择MDC时钟速率 00b = 2.5 MHz（400 ns）——IEEE 802.3标准 01b = 5 MHz（200 ns） 10b = 12.5 MHz（80 ns） 11b = 25 MHz（40 ns）
3	MIICL45	R/W NASR	MII Clause 45 该位用于在802.3 Clause 22与Clause 45命令格式之间进行选择。 0b = Clause 22 1b = Clause 45
2:1	MIIWnR	R/W NASR	MII写 在Clause 22模式下，将低位置1会告知PHY本次为写操作（使用MII数据寄存器）。如果低位未置1，则为读操作（数据打包到MII数据寄存器中）。不使用高位。 在Clause 45模式下，这两个位按下列组合确定命令： 00b = 地址 01b = 写 10b = 读 11b = 读且后递增地址 **注：**与IEEE 802.3 Clause 45.3.4 OP（操作码）中给出的帧结构值相比，读命令与读且后递增地址命令的值互换。
0	MIIBZY	R/W NASR	MII繁忙 必须轮询该位以确定何时完成MII寄存器访问。写入该寄存器或MII数据寄存器之前，该位必须读为逻辑0。为使主机能够读取或写入任何MII PHY寄存器，LAN驱动程序软件必须将该位置1（1b）。 在访问MII寄存器期间，该位将置1以指示正在进行读/写访问。在PHY写操作期间，MII数据寄存器必须保持有效，直到MAC清零该位为止。在PHY读操作期间，MII数据寄存器保持无效，直到MAC清零该位为止。

表71列出了MII数据寄存器的详细信息。

表71： MII_DATA

MII数据寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:16	Reserved	R	始终为0
15:0	MII_DATA	R/W NASR	MII数据 该位域包含PHY读操作读取的16位值，或在MII写操作之前要写入PHY的16位数据值。 对于Clause 22访问或者Clause 45写命令、读命令和读且后递增地址命令，该寄存器包含要写入由MII访问寄存器（MII_ACCESS）指定的PHY寄存器的数据，或从由MII访问寄存器指定索引的PHY寄存器中读取的数据。 对于Clause 45地址命令，该寄存器提供随后将访问的寄存器地址。 **注：** 写入该寄存器之前，必须清零MII访问寄存器（MII_ACCESS）中的MII繁忙（MIIBZY）位。

表72列出了MII RGMII内部延时寄存器的详细信息。

表72： MII_RGMII_ID

MII RGMII内部延时寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:2	Reserved	R	始终为0
1	RGMII_TXC_DELAY_EN	R/W NASR	RGMII TXC延时使能 配置RGMII TXC延时模式： 0b = 禁止RGMII TXC延时模式 1b = 使能RGMII TXC延时模式 **注：** 该位仅在RGMII模式下有意义。该位受复位保护（RST_PROTECT）位保护。
0	RGMII_RXC_DELAY_EN	R/W NASR	RGMII RXC延时使能 配置RGMII RXC延时模式： 0 = 禁止RGMII RXC延时模式 1 = 使能RGMII RXC延时模式 **注：** 该位仅在RGMII模式下有意义。该位受复位保护（RST_PROTECT）位保护。

表73列出了节能以太网TX LPI请求延时计数寄存器的详细信息。

表73： EEE_TX_LPI_REQUEST_DELAY_CNT

节能以太网TX LPI请求延时计数寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:0	EEE_TX_LPI_REQUEST_ DELAY_CNT	R/W NASR	EEE TX LPI请求延时计数 包含与一段时间（以微秒为单位）对应的计数，MAC在TX FIFO为空后或在发送暂停帧或ARP/NS响应后必须等待这段时间结束才能调用LPI协议。 每当TX FIFO为空时，MAC会检查MAC控制寄存器（MAC_CR）中的节能以太网使能（EEEEN）位以确定节能以太网工作模式是否生效。如果该位清零，则不采取任何操作，否则MAC将等待该寄存器中指示的一段时间。在等待周期超时后，LPI协议将启动，MAC中断状态寄存器（MAC_INT_STS）中的节能以太网启动TX低功耗中断（EEE_START_TX_LPI_INT）位将置1。 如果TX FIFO变为非空或者MAC自主发送暂停帧或ARP/NS响应，将重启定时器。 主机软件只能在节能以太网使能（EEEEN）位清零时更改该位域。 **注：**由于存在1 µs预分频器，因此实际时间可能比指定时间长1 µs。可以设置为0，也不会因此触发延时。但是，零值可能会导致TX数据路径无法支持千兆位操作。当器件以千兆位速度工作时，使用值50 µs比较合理。可根据使能EEE后的性能测试结果适当增大该值。802.3 az的设计初衷是针对EEE链路大部分时间处于空闲状态、偶尔出现全带宽突发传输的场景，而非在数据包非活动期间积极优化功耗。 当EEE定时器加速（EEE_TIMER_SPEED_UP）位置1时，时间以MAC TX时钟周期递增。

表74列出了唤醒和控制寄存器的详细信息。

表74： WUCSR2

唤醒和控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	CSUM_DISABLE	R/W NASR	校验和禁止 清零时，将计算IP头校验和、TCP校验和、ICMP有效负载校验和与FCS，所有这些都必须与帧内容一致，以便将该帧（TCP_SYN或NS）纳入检测分析的考虑范围。 置1时，对于TCP_SYN和NS帧，仅计算并校验FCS。不计算IP头校验和、ICMP有效负载校验和和TCP校验和，因此忽略任何不匹配。 **注：**该位受复位保护（RST_PROTECT）位保护。
30	EN_ROUTE_HDR	R/W NASR	使能其他路由头 该位允许在验证TCP SYN帧的TCP校验和时使用类型0和2以外的IPv6路由头。 清零时，不支持类型0和2以外的IPv6路由头，并且不验证校验和。 置1时，如果报头中的剩余段数字段为零，则跳过类型0和2以外的IPv6路由头，否则不验证校验和。 **注：**该位受复位保护（RST_PROTECT）位保护。
29:11	Reserved	R	始终为0
10	FARP_FR	R/W NASR	转发ARP帧 以该MAC为目标且已被ARP卸载逻辑处理的接收ARP帧可传递到RX数据路径接口。 0b = 丢弃收到的ARP帧。 1b = 将收到的ARP帧传递到RX数据路径接口。
9	FNS_FR	R/W NASR	前向NS帧 以该MAC为目标且已被NS卸载逻辑处理的接收NS帧可传递到RX数据路径接口。 0b = 丢弃收到的NS帧。 1b = 将收到的NS帧传递到RX数据路径接口。

表74： WUCSR2（续）

唤醒和控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
8	NA_SA_SEL	R/W NASR	NA SA选择 用于选择NA报文中IPv6 SA的源。 置1时，使用NSx IPv6目标地址寄存器（NSx_IPV6_ADDR_DEST）值作为源。 清零时，使用NS数据包中的目标地址。
7	NS_RCD	R/W1C NASR	已收到NS数据包 MAC在收到有效的NS数据包时将该位置1。
6	ARP_RCD	R/W1C NASR	已收到ARP数据包 MAC在收到有效的ARP数据包时将该位置1。
5	IPV6_TCPSYN_RCD	R/W1C NASR	已收到IPv6 TCP SYN数据包 MAC在收到有效的IPv6 TCP SYN数据包时将该位置1。 如果IPv6 TCP SYN唤醒使能（IPV6_TCPSYN_WAKE_EN）位清零，则不会将该位置1。 如果以下任一位已置1，则不会将该位置1： 已收到RFE唤醒帧（RFE_WAKE_FR） 已收到完美DA帧（PFDA_FR） 已收到远程唤醒帧（WUFR） 已收到魔术包（MPR） 已收到广播帧（BCAST_FR） 已收到IPv4 TCP SYN数据包（IPV4_TCPSYN_RCD） 如果恢复清零远程唤醒状态（RES_CLR_WKP_STS）位置1，当端点因唤醒事件发送PM_PME报文时，该位将自动清零。

表74： WUCSR2（续）

唤醒和控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
4	IPV4_TCPSYN_RCD	R/W1C NASR	已收到IPv4 TCP SYN数据包 MAC在收到有效的IPv4 TCP SYN数据包时将该位置1。 如果IPv4 TCP SYN唤醒使能（IPV4_TCPSYN_WAKE_EN）位清零，则不会将该位置1。 如果以下任一位已置1，则不会将该位置1： 已收到RFE唤醒帧（RFE_WAKE_FR） 已收到完美DA帧（PFDA_FR） 已收到远程唤醒帧（WUFR） 已收到魔术包（MPR） 已收到广播帧（BCAST_FR） 已收到IPv6 TCP SYN数据包（IPV6_TCPSYN_RCD） 如果恢复清零远程唤醒状态（RES_CLR_WKP_STS）位置1，当端点因唤醒事件发送PM_PME报文时，该位将自动清零。
3	NS_OFFLOAD_EN	R/W NASR	NS卸载使能 置1时，使能对邻居请求数据包的响应。
2	ARP_OFFLOAD_EN	R/W NASR	ARP卸载使能 置1时，使能对ARP数据包的响应。
1	IPV6_TCPSYN_WAKE_EN	R/W NASR	IPv6 TCP SYN唤醒使能 置1时，使能在收到IPv6 TCP SYN数据包时唤醒。 如果恢复清零远程唤醒使能（RES_CLR_WKP_EN）位置1，当端点因唤醒事件发送PM_PME报文时，该位将自动清零。 **注：**该位受复位保护（RST_PROTECT）位保护。
0	IPV4_TCPSYN_WAKE_EN	R/W NASR	IPv4 TCP SYN唤醒使能 置1时，使能在收到IPv4 TCP SYN数据包时唤醒。 如果恢复清零远程唤醒使能（RES_CLR_WKP_EN）位置1，当端点因唤醒事件发送PM_PME报文时，该位将自动清零。 **注：**该位受复位保护（RST_PROTECT）位保护。

表75列出了接收过滤引擎控制寄存器的详细信息。

表75： RFE_CTL

接收过滤引擎控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:19	Reserved	R	始终为0
18	EN_OTHR_ROUTE_HDR	R/W NASR	使能其他路由头 该位允许在验证RX校验和卸载的UDP、TCP或ICMPv6校验和时使用类型0和2以外的IPv6路由头。清零时，不支持类型0和2以外的IPv6路由头，并且不验证校验和。 置1时，如果报头中的剩余段数字段为零，则跳过类型0和2以外的IPv6路由头，否则不验证校验和。 **注：**无论该位为何，都会计算原始校验和。
17:16	DFLT_REC_CHAN	R/W NASR	默认接收通道 该位域用于指定默认接收通道。
15	PASS_WKP	R/W NASR	始终传递唤醒帧 置1时，如果存储唤醒帧（STORE_WAKE）置1，RFE将永不丢弃在D3状态下唤醒器件的接收唤醒帧。
14	EN_IGMP_CHK_VAL	R/W NASR	使能IGMP校验和验证 置1时，RFE将验证IGMP校验和。此外，RFE还将计算L3原始校验和并将其插入RX命令B中。 **注：**即使帧不是IGMP，也将计算原始校验和。
13	EN_ICMP_CHK_VAL	R/W NASR	使能ICMP校验和验证 置1时，RFE将验证ICMP校验和。此外，RFE还将计算L3原始校验和并将其插入RX命令B中。 **注：**即使帧不是ICMP，也将计算原始校验和。
12	EN_TCPUDP_CHK_VAL	R/W NASR	使能TCP/UDP校验和验证 置1时，RFE将检查TCP或UDP校验和。此外，RFE还将计算L3原始校验和并将其插入RX命令B中。 **注：**即使帧不是TCP或UDP，也将计算原始校验和。

表75： RFE_CTL（续）

接收过滤引擎控制寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
11	EN_IP_CHK_VAL	R/W NASR	使能IP校验和验证 置1时，RFE将验证IP校验和。如果帧不是IPv4或IPv6，该位不起作用。
10	AB	R/W NASR	接受广播帧 置1时，接受所有广播帧。清零时，丢弃广播帧。
9	AM	R/W NASR	接受多播帧 置1时，接受所有多播帧。清零时，多播帧必须通过完美过滤或哈希过滤。 **注：**该位不适用于广播帧。
8	AU	R/W NASR	接受单播帧 置1时，接受所有单播帧。
7	EN_VLAN_TAG_STRIPPING	R/W NASR	使能VLAN标记移除 该位置1时，使能将接收帧的VLAN ID移除。
6	UF	R/W NASR	无标记帧过滤 置1时，丢弃所有无标记的接收帧。
5	VF	R/W NASR	使能VLAN过滤 该位置1时，使能对接收帧的VLAN ID进行过滤。
4	SPF	R/W NASR	使能源地址完美过滤 该位置1时，使能对接收帧的以太网源地址进行完美过滤。 **注：**如果使能目标地址过滤（完美或哈希），则帧必须同时通过源地址过滤和目标地址过滤才不会被丢弃。
3	MHF	R/W NASR	使能多播地址哈希过滤 置1时，将对多播目标地址进行哈希过滤。 **注：**永不对广播地址进行哈希过滤。
2	DHF	R/W NASR	使能目标地址哈希过滤 置1时，将对单播目标地址进行哈希过滤。
1	DPF	R/W NASR	使能目标地址完美过滤 该位置1时，使能对接收帧的以太网目标地址进行完美过滤。

表75： RFE_CTL（续）

接收过滤引擎控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
0	RST_RECV_FILT_ENG	SC	复位接收过滤引擎 该位置1时将复位RFE。标记为NASR的寄存器位不会复位。

表76列出了DMA控制器配置寄存器的详细信息。

表76： DMAC_CFG

DMA控制器配置寄存器			默认值：0x0000_0360
Bit	**名称**	**R/W**	**说明**
31:19	Reserved	R	始终为0
18	DMA_INTR_DSCR_RD_EN	R/W NASR	DMA描述符间空间读使能 置1时，DMAC将读取描述符之间的保留空间。 **注：**除非所有DMA通道都处于由开始接收x（STRT_R_x）/停止接收x（STP_R_x）和开始发送x（STRT_T_x）/停止发送x（STP_T_x）位指示的初始状态，否则软件不得修改该位。 该位控制所有RX和TX通道。
17	DMA_INTR_DSCR_WR_EN	R/W NASR	DMA描述符间空间写使能 置1时，DMAC将改写描述符之间的保留空间。 **注：**除非所有RX DMA通道都处于由开始接收x（STRT_R_x）/停止接收x（STP_R_x）位指示的初始状态，否则软件不得修改该位。 该位控制所有RX通道。
16	DMA_COAL_EN	R/W NASR	DMA合并模块 置1时，使能DMA合并。 **注：**该位控制所有RX和TX通道。每个通道都可以独立禁止合并。
15:13	DMA_CMPL_RETRY_CNT	R/W NASR	DMA完成重试计数 该位域用于指定DMA在遇到读请求导致的超时或中毒指示时将执行的重试次数。 **注：**除非所有DMA通道都处于由开始接收x（STRT_R_x）/停止接收x（STP_R_x）和开始发送x（STRT_T_x）/停止发送x（STP_T_x）位指示的初始状态，否则软件不得修改该位域。 该位域控制所有RX和TX通道。

表76： DMAC_CFG（续）

DMA控制器配置寄存器			默认值： 0x0000_0360
Bit	**名称**	**R/W**	**说明**
12	DMA_CMPL_RETRY_EN	R/W NASR	DMA完成重试使能 置1时，DMA将重试导致超时或存在中毒指示的读请求。 **注：**除非所有DMA通道都处于由开始接收x（STRT_R_x）/停止接收x（STP_R_x）和开始发送x（STRT_T_x）/停止发送x（STP_T_x）位指示的初始状态，否则软件不得修改该位。 该位控制所有RX和TX通道。
11:10	DMA_CH_ARB_SEL	R/W NASR	DMA通道仲裁选择 该位域用于控制DMA通道仲裁方案。00 = 固定优先级——RX高于TX；01 = 固定优先级——通道编号顺序（当通道编号相等时，RX高于TX）；10 = 固定优先级——RX高于TX；11 = 循环（通道内循环优先级）。 **注：**除非所有DMA通道都处于由开始接收x（STRT_R_x）/停止接收x（STP_R_x）和开始发送x（STRT_T_x）/停止发送x（STP_T_x）位指示的初始状态，否则软件不得修改该位域。 该位域控制所有RX和TX通道。
9	TX_RLS_ARB_DESC_REQ	R/W NASR	请求TX描述符时释放仲裁器 对于TX DMA描述符读操作，该位控制DMAC PCIe®仲裁器何时释放，以便能够为来自DMAC的其他PCIe请求提供服务。 0b = 仅在返回读取完成信号后释放仲裁器。该设置会拒绝服务所有后续的DMAC PCIe请求。 1b = 在发送读请求后释放仲裁器。该设置允许存在多个未完成的TX描述符读请求（每通道一个）以及其他并发PCIe读写请求。
8	RX_RLS_ARB_DESC_REQ	R/W NASR	请求RX描述符时释放仲裁器 对于RX DMA描述符读操作，该位控制DMAC PCIe仲裁器何时释放，以便能够为来自DMAC的其他PCIe请求提供服务。 0b = 仅在返回读取完成信号后释放仲裁器。该设置会拒绝服务所有后续的DMAC PCIe请求。 1b = 在发送读请求后释放仲裁器。该设置允许存在多个未完成的RX描述符读请求（每通道一个）以及其他并发PCIe读写请求。
7	Reserved	R	始终为0

表76： DMAC_CFG（续）

DMA控制器配置寄存器			默认值： 0x0000_0360
Bit	**名称**	**R/W**	**说明**
6:4	MAX_READ_REQ	R/W NASR	未完成数据读请求最大数量 该位域用于限制针对TX数据的未完成DMA读请求的最大数量。有效值为1至6。 **注：**除非TX DMA通道都处于由开始发送x（STRT_T_x）/停止发送x（STP_T_x）位指示的初始状态，否则软件不得修改该位域。 该位域控制所有TX通道。
3:2	Reserved	R	始终为0
1:0	DSPACE	R/W NASR	描述符间距 该位域用于指定存储器中为每个描述符保留的块大小（以字节为单位）。描述符本身将位于保留块的前16个字节中。该位域由主机编程以匹配主机CPU的高速缓存行大小。 00b = 16 01b = 32 10b = 64 11b = 128 **注：**除非所有DMA通道都处于由开始接收x（STRT_R_x）/停止接收x（STP_R_x）和开始发送x（STRT_T_x）/停止发送x（STP_T_x）位指示的初始状态，否则软件不得修改该位域。 该位域控制所有RX和TX通道。

表77列出了DMA控制器命令寄存器的详细信息。

表77： DMAC_CMD

DMA控制器命令寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	DMAC_SWR	R/W1S/SC	DMA软件复位 该位置1时，复位整个DMA引擎。标记为NASR的通道的寄存器位不会复位。 写入0不起作用。该位自清零。 **注：**这是一个即时复位，将中止任何正在处理的PCIe®请求。这可能导致PCIe数据包损坏。
30:29	Reserved	R	始终为0
28	DMA_COAL_EXIT	R/W1S/SC	DMA合并退出 向该位写入1将强制DMA控制器退出合并状态。 **注：**该位控制所有RX和TX通道。

表77： DMAC_CMD（续）

DMA控制器命令寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
27:24	TX_SWR_x	R/W1S/SC	TX DMA软件复位x 该位置1时，将复位通道x的发送DMA引擎，包括重排序缓冲区中针对该通道的待处理和已完成的请求。标记为NASR的通道的寄存器位不会复位。 写入0不起作用。该位自清零。 这是一个即时复位，将中止任何正在处理的PCIe请求。这可能导致PCIe数据包损坏。
23:20	STRT_T_x	R/W1S	开始发送x 向该位写入1时，通道x的TX DMAC将启动（如果处于初始状态）或恢复工作（如果处于停止状态）。 开始发送命令仅在TX DMAC未运行（初始或停止状态）时有效。在已启动或停止待处理状态下写入该位不起作用。 **注：**首次发出开始发送命令之前，必须将环的有效基址编程到TX通道x环基址高位寄存器（TX_BASE_ADDRHx）和TX通道x环基址低位寄存器（TX_BASE_ADDRLx）中；将环长度和其他参数编程到TX通道x配置A寄存器（TX_CFG_Ax）和TX通道x配置B寄存器（TX_CFG_Bx）中；如果使用头指针回写，还需编程TX通道x头回写地址高位寄存器（TX_HEAD_WRITEBACK_ADDRHx）和TX通道x头回写地址低位寄存器（TX_HEAD_WRITEBACK_ADDRLx），否则TX DMAC的行为将是未定义的。 该位将在停止发送请求完成后清零。 读取该位域以及停止发送x（STP_T_x）位域可指示TX DMAC通道的当前状态。 STRT_T_x/STP_T_x： 00b——初始 10b——已启动 11b——停止待处理 01b——已停止

表77： DMAC_CMD（续）

DMA控制器命令寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
19:16	STP_T_x	R/W1S	停止发送x 向该位写入1时，通道x的TX DMAC进入已停止状态。 停止发送命令仅在TX DMAC处于已启动状态时有效。在初始、停止待处理或已停止状态下写入该位不起作用。 **注：**除非另外说明，否则软件不得修改任何会影响已停止通道的DMA参数。更改参数之前，应通过复位命令将通道设置为初始状态。该位将在开始发送请求发出后清零。 读取该位域以及开始发送x（STRT_T_x）可指示TX DMAC通道的当前状态。 STRT_T_x/STP_T_x： 00b——初始 10b——已启动 11b——停止待处理 01b——已停止
15:12	Reserved	R	始终为0
11:8	RX_SWR_x	R/W1S/SC	RX DMA软件复位 该位置1时，复位通道x的接收DMA引擎。标记为NASR的通道的寄存器位不会复位。 写入0不起作用。该位自清零。 这是一个即时复位，将中止任何正在处理的PCIe请求。这可能导致PCIe数据包损坏。

表77： DMAC_CMD（续）

DMA控制器命令寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
7:4	STRT_R_x	R/W1S	开始接收 向该位写入1时，通道x的RX DMAC将启动（如果处于初始状态）或恢复工作（如果处于停止状态）。 开始接收命令仅在RX DMAC未运行（初始或停止状态）时有效。在已启动或停止待处理状态下写入该位不起作用。 **注：**首次发出开始接收命令之前，必须将环的有效基址编程到RX通道x环基址高位寄存器（RX_BASE_ADDRHx）和RX通道x环基址低位寄存器（RX_BASE_ADDRLx）中；将环长度和其他参数编程到RX通道x配置A寄存器（RX_CFG_Ax）和RX通道x配置B寄存器（RX_CFG_Bx）中；如果使用头指针回写，还需编程RX通道x头回写地址高位寄存器（RX_HEAD_WRITEBACK_ADDRHx）和RX通道x头回写地址低位寄存器（RX_HEAD_WRITEBACK_ADDRLx），否则RX DMAC的行为将是未定义的。该位将在停止接收请求完成后清零。 读取该位域以及停止接收x（STP_R_x）位域可指示RX DMAC通道的当前状态。 STRT_R_x/STP_R_x： 00b——初始 10b——已启动 11b——停止待处理 01b——已停止

表77： DMAC_CMD（续）

DMA控制器命令寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
3:0	STP_R_x	R/W1S	停止接收 向该位写入1时，通道x的RX DMAC进入已停止状态。 停止接收命令仅在RX DMAC处于已启动状态时有效。在初始、停止待处理或已停止状态下写入该位不起作用。 **注：**除非另外说明，否则软件不得修改任何会影响已停止通道的DMA参数。更改参数之前，应通过复位命令将通道设置为初始状态。该位将在开始接收请求发出后清零。 读取该位域以及开始接收x（STRT_R_x）可指示RX DMAC通道的当前状态。 STRT_R_x/STP_R_x： 00b——初始 10b——已启动 11b——停止待处理 01b——已停止

表78列出了DMA控制器中断状态寄存器的详细信息。

表78： DMAC_INT_SYS

DMA控制器中断状态寄存器			默认值： 0x0000_FF00
Bit	**名称**	**R/W**	**说明**
31:28	Reserved	R	始终为0
27:24	RXPRIx	R	RX优先级帧x状态 每个单独的状态位都用于指示对应的当前中断是由优先级帧触发。 每个位通常在对应的中断位置1时置1，但在对应中断位首次置1时可能保持未置1状态，等到后续出现优先级帧时再置1。即使后续出现非优先级帧，这些位也保持置1状态。 这些位在对应的中断位清零时清零（有关清零条件，请参见第13.0章“配置文件格式”）。 **注：**由于新的优先级帧造成的同时置1和清零将导致这些位（及对应的中断位）保持置1状态。这些位可能在发生首次中断时处于未置1状态，等到收到后续优先级帧时再置1。如果这两个事件与软件操作重叠，将响应首次中断，读取该寄存器并清零对应的中断位，优先级状态可能会丢失。软件应使用接收描述符中的PRI位来确定是否收到优先级帧。
23:22	Reserved	R	始终为0
21	DMA_ERR_INT	R	DMA错误中断 当RX通道x错误状态寄存器（RX_ERR_STSx）或TX通道x错误状态寄存器（TX_ERR_STSx）中的任何状态位置1时，该中断置为有效。 RX和TX尾指针错误以及TX序列错误需要使能才能通过该位报告。 该位级联到INT_STS寄存器中的DMA_GEN_INT位。 **注：**这是一个电平触发的中断事件，触发后将保持有效状态，直到RX通道x错误状态寄存器（RX_ERR_STSx）和TX通道x错误状态寄存器（TX_ERR_STSx）中的所有错误事件位都清零为止。
20	Reserved	R	始终为0

注： 该寄存器包含DMA中断源的当前状态。除非另外说明，否则向对应位写入1将应答并清除中断。该寄存器中的中断状态位始终反映中断源的状态，与对应的中断是否为允许状态无关。

表78： DMAC_INT_SYS（续）

DMA控制器中断状态寄存器			默认值：0x0000_FF00
Bit	**名称**	**R/W**	**说明**
19:16	RXFRMx_INT	R/W1C NASR	RX帧x中断 每个单独的中断在帧通过对应通道传输到主机存储器时触发。这些位级联到INT_STS寄存器中的DMA_RXx_INT位。 **注：**这些是边沿触发的中断事件。
15:12	RXx_STOP_INT	R/W1C NASR	RX DMA x已停止中断 每个单独的中断在RX DMA通道x由于处理停止命令而进入已停止状态时触发。这些位级联到INT_STS寄存器中的DMA_GEN_INT位。 **注：**这是一个边沿触发的中断。
11:8	TXx_STOP_INT	R/W1C NASR	TX DMA x已停止中断 每个单独的中断在TX DMA通道x由于处理停止命令而进入已停止状态时触发。该位级联到INT_STS寄存器中的DMA_GEN_INT位。 **注：**这是一个边沿触发的中断事件。
7:4	Reserved	R	始终为0
3:0	TXx_INT	R/W1C NASR	TX x中断 每个单独的中断在通道x的TX DMA引擎关闭IOC或LS位置1的发送描述符时触发，描述符是否置1取决于TX定时器和头指针回写选择（TX_TMR_HPWB_SEL）位域。 中断可能立即触发也可能延迟触发。如果发送描述符同时将DTI位置1，该中断将延迟触发。该位级联到INT_STS寄存器中的DMA_TXx_INT位。 **注：**这是一个边沿触发的中断事件。

注： 该寄存器包含DMA中断源的当前状态。除非另外说明，否则向对应位写入1将应答并清除中断。该寄存器中的中断状态位始终反映中断源的状态，与对应的中断是否为允许状态无关。

表79列出了SGMII控制寄存器的详细信息。

表79： SGMII_CTL

SGMII控制寄存器			默认值：0x0000_0080
Bit	**名称**	**R/W**	**说明**
31	SGMII_ENABLE	R/W	置1时，MAC将在SGMII/1000/2500BASE-X模式下工作。
30:8	Reserved	R	始终为0

表79： SGMII_CTL（续）

SGMII控制寄存器			默认值： 0x0000_0080
Bit	**名称**	**R/W**	**说明**
7:5	TX_VBOOST_LVL	R/W	连接到SGMII/1000/2500BASE-X PHY的TX_VBOOST_LVL[2:0]输入
4	TX_VBOOST_EN	R/W	连接到SGMII/1000/2500BASE-X PHY的TX0_VBOOST_EN输入
3	RBULK_SHORT	R/W	针对SGMII/RGMII焊盘多路开关使能时减小Bulk与GD间电阻 0b = 10 kΩ 1b = 1 kΩ
2	SGMII_CP_DISABLE	R/W	该位用于控制SGMII/RGMII焊盘多路开关中的电荷泵。 0b = SGMII/RGMII焊盘多路开关根据RGMII/SGMII模式和模拟电压控制电荷泵。 1b = 禁止电荷泵。 **注：**在实际使用中，由于SGMII/RGMII 引脚多路开关使用3.3V供电，因此永不使能电荷泵。
1	SGMII_PWR_DN	R/W	SGMII掉电 置1时，SGMII/1000/2500BASE-X接口被禁止并进入掉电状态。
0	SGMII_RESET	R/W	SGMII复位 置1时，SGMII/1000/2500BASE-X接口保持复位状态。

表80列出了中断状态寄存器的详细信息。

表80： INT_SYS

中断状态寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:28	Reserved	R	始终为0
27:24	DMA_RXx_INT	R/W1C	DMA RX通道3:0中断 指示来自RX通道的DMA控制器中断事件。只要DMA控制器中断状态寄存器（DMAC_INT_STS）中的以下位置1，该位就置1： • RXFRMx_INT 向该位写入1时，DMA控制器中断状态寄存器（DMAC_INT_STS）中的 RXFRMx_INT 和 RXPRIx 位将清零，此外还有多种其他清零方法。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到DMA控制器中的位清零或禁止为止。
23:20	Reserved	R	始终为0

表80： INT_SYS（续）

中断状态寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
19:16	DMA_TXx_INT	R	DMA TX通道x中断 指示来自TX通道的DMA控制器中断事件。只要DMA控制器中断状态寄存器（DMAC_INT_STS）中的以下位置1，该位就置1： • TXx_INT 向该寄存器写入1时，DMA控制器中断状态寄存器（DMAC_INT_STS）中的TXx_INT位将清零，此外还有多种其他清零方法。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到DMA控制器中的位清零或禁止为止。
15:11	Reserved	R	始终为0
10	DMA_GEN_INT	R	DMA通用中断 指示通用DMA控制器中断事件。只要DMA控制器中断状态寄存器（DMAC_INT_STS）中的以下位置1，该位就置1： • DMA_ERR_INT • RXx_STOP_INT • TXx_STOP_INT **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到DMA控制器中的位清零或禁止为止。
9	SW_GP_INT	R/W1C	软件通用中断 向中断置1寄存器（INT_SET）中的软件通用中断置1（SW_GP_INT_SET）位写入高电平时，将触发该中断。没有相关的硬件事件用于设置该位。 **注：**这是一个脉冲触发的中断。
8	PCIe_INT	R	PCIe®中断 指示PCIe中断事件。只要PCIe中断状态寄存器（PCIe_INT_STS）中的任何已使能位置1，该位就置1。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到PCIe中断状态寄存器（PCIe_INT_STS）中的位清零或禁止为止。

表80： INT_SYS（续）

中断状态寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
7	1588_INT	R	1588中断 指示1588 PTP中断事件。只要PTP中断状态寄存器（PTP_INT_STS）中的任何已使能位置1，该位就置1。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到1588模块中的位清零或禁止为止。
6	SGMII_AN_DONE_INT	R	SGMII/1000/2500BASE-X自动协商完成中断 指示 SGMII/1000/2500BASE-X 中断事件。当 SGMII/1000/2500BASE-X接口逻辑将其中断输出信号置为有效时，该位置1。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到SGMII/1000/2500BASE-X接口中的位清零或禁止为止。
5	ETH_PHY_INT	R	以太网PHY中断 指示以太网PHY中断事件。当ENET_PHY_INT_N引脚处于有效状态时，该位置1。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到PHY中的位清零或禁止或者ENET_PHY_INT_N引脚变为无效状态为止。
4	DP_INT	R/W1C	数据端口中断 指示待处理的数据端口操作已完成。 **注：**这是一个脉冲触发的中断。
3	MAC_INT	R	MAC中断 指示以太网MAC中断事件。只要MAC中断状态寄存器（MAC_INT_STS）中的任何已使能状态位置1，该位就置1。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到MAC模块中的位清零或禁止为止。
2	FCT_INT	R	FCT中断 指示FIFO控制器中断事件。只要FIFO控制器中断状态寄存器（FCT_INT_STS）中的任何已使能状态位置1，该位就置1。 **注：**这是一个电平触发的中断。该中断触发后会保持有效状态，直到FCT模块中的位清零或禁止为止。

表80： INT_SYS（续）

中断状态寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
1	GPT_INT	R/W1C	GP定时器中断 当通用定时器达到零时，触发该中断。 **注：**这是一个脉冲触发的中断。
0	MAS_INT	R	控制器中断 该位用于反映该寄存器中所有已使能位的或运算结果。已禁止的中断（通过各个允许位）不会影响该位。

表81列出了中断置1寄存器的详细信息。

表81： INT_SET

中断置1寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:28	Reserved	R	始终为0
27:24	DMA_RXx_INT_EN_SET	R/W1C	DMA RX通道3:0中断允许置1
23:20	Reserved	R	始终为0
19:16	DMA_TXx_INT_EN_SET	R/W1C	DMA TX通道x中断允许置1
15:11	Reserved	R	始终为0
10	DMA_GEN_INT_EN_SET	R/W1C	DMA通用中断允许置1
9	SW_GP_INT_EN_SET	R/W1C	软件通用中断允许置1
8	PCIe_INT_EN_SET	R/W1C	PCIe中断允许置1
7	1588_INT_EN_SET	R/W1C	1588中断允许置1
6	SGMII_AN_DONE_INT_EN_SET	R/W1C	SGMII/1000/2500BASE-X自动协商完成中断允许置1
5	ETH_PHY_INT_EN_SET	R/W1C	以太网PHY中断允许置1
4	DP_INT_EN_SET	R/W1C	数据端口中断允许置1
3	MAC_INT_EN_SET	R/W1C	MAC中断允许置1
2	FCT_INT_EN_SET	R/W1C	FCT中断允许置1
1	GPT_INT_EN_SET	R/W1C	GP定时器中断允许置1
0	MAS_INT_EN_SET	R/W1C	控制器中断允许置1 该位是影响所有主要中断源的额外使能位。

注： 软件可使用该寄存器设置中断状态寄存器（INT_STS）中的任何非保留位。

表82列出了中断允许置1寄存器的详细信息。

表82： INT_EN_SET

中断允许置1寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:28	Reserved	R	始终为0
27:24	MA_RXx_INT_EN_SET	R/W1C	DMA RX通道3:0中断允许置1
23:20	Reserved	R	始终为0
19:16	DMA_TXx_INT_EN_SET	R/W1C	DMA TX通道x中断允许置1
15:11	Reserved	R	始终为0
10	DMA_GEN_INT_EN_SET	R/W1C	DMA通用中断允许置1
9	SW_GP_INT_EN_SET	R/W1C	软件通用中断允许置1
8	PCIe_INT_EN_SET	R/W1C	PCIe中断允许置1
7	1588_INT_EN_SET	R/W1C	1588中断允许置1
6	SGMII_AN_DONE_INT_EN_SET	R/W1C	SGMII/1000/2500BASE-X自动协商完成中断允许置1
5	ETH_PHY_INT_EN_SET	R/W1C	以太网PHY中断允许置1
4	DP_INT_EN_SET	R/W1C	数据端口中断允许置1
3	MAC_INT_EN_SET	R/W1C	MAC中断允许置1
2	FCT_INT_EN_SET	R/W1C	FCT中断允许置1
1	GPT_INT_EN_SET	R/W1C	GP定时器中断允许置1
0	MAS_INT_EN_SET	R/W1C	控制器中断允许置1 该位是影响所有主要中断源的额外使能位。

注： 该寄存器用于将与中断状态寄存器（INT_STS）中的相应位关联的中断允许位置 1。向位中写入 1 会将相应的允许位置1，并将相应的中断配置为系统中断源。写入0不起作用。读取该寄存器将返回中断允许位的状态。

表83列出了中断允许清零寄存器的详细信息。

表83： INT_EN_CLR

中断允许清零寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:28	Reserved	R	始终为0
27:24	MA_RXx_INT_EN_CLR	R/W1C	DMA RX通道3:0中断允许清零
23:20	Reserved	R	始终为0
19:16	DMA_TXx_INT_EN_CLR	R/W1C	DMA TX通道x中断允许清零
15:11	Reserved	R	始终为0
10	DMA_GEN_INT_EN_CLR	R/W1C	DMA通用中断允许清零
9	SW_GP_INT_EN_CLR	R/W1C	软件通用中断允许清零
8	PCIe_INT_EN_CLR	R/W1C	PCIe中断允许清零
7	1588_INT_EN_CLR	R/W1C	1588中断允许清零

注： 该寄存器用于将与中断状态寄存器（INT_STS）中的相应位关联的中断允许位清零。向位中写 1 会将相应的允许位清零。写入0不起作用。读取该寄存器将返回中断允许位的状态。

表83： INT_EN_CLR（续）

中断允许清零寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
6	SGMII_AN_DONE_INT_EN_CLR	R/W1C	SGMII/1000/2500BASE-X自动协商完成中断允许清零
5	ETH_PHY_INT_EN_CLR	R/W1C	以太网PHY中断允许清零
4	DP_INT_EN_CLR	R/W1C	数据端口中断允许清零
3	MAC_INT_EN_CLR	R/W1C	MAC中断允许清零
2	FCT_INT_EN_CLR	R/W1C	FCT中断允许清零
1	GPT_INT_EN_CLR	R/W1C	GP定时器中断允许清零
0	MAS_INT_EN_CLR	R/W1C	控制器中断允许清零 该位是影响所有主要中断源的额外使能位。

注： 该寄存器用于将与中断状态寄存器（INT_STS）中的相应位关联的中断允许位清零。向位中写 1 会将相应的允许位清零。写入0不起作用。读取该寄存器将返回中断允许位的状态。

9.0 外设子系统实现——UART

UART外设子系统占用自己的PCI器件端点，需要主机系统提供独立的器件驱动程序才能正常工作。该子系统与所有其他外设（I^2C、SPI和GPIO）共用PCI总线。

子系统ID如下：

- 供应商ID：0x1055（Microchip Technology, Inc/SMSC）
- 器件ID：
  - PCI12000：0xA002
  - PCI11010：0xA012
  - PCI11400：0xA032
  - PCI11101：0xA022
  - PCI11414：0xA042
- 分类ID：0x070002（“16550串行端口”）

9.1 章节

- 第9.2节“特性汇总”
- 第9.3节“发送操作”
- 第9.4节“接收操作”
- 第9.5节“波特率时钟”
- 第9.6节“UART子系统寄存器”

9.2 特性汇总

- 最多支持四个UART实例，每个实例都配备完整线组：
  - TXD
  - RXD
  - nRTS
  - nCTS
  - nDTR
  - nDSR
  - nDCD
  - nRI
- 可编程字长、停止位和奇偶校验
- 可编程波特率发生器
- 中断发生器
- 环回模式
- 接口寄存器
- 256字节发送FIFO
- 256字节接收FIFO
- 多个时钟源
- 引脚极性控制
- 低功耗休眠模式

9.2.1 支持RS-232

典型应用使用最少四根线（称为RS-232）：

- TXD
- RXD
- nRTS
- nCTS

9.2.2 RS-485支持

对于工业RS-485用例，提供了自动方向功能。当没有其他数据可用时，该功能通过nRTS或nCTS线将发送器设置为三态。这样可以将共用线释放给其他节点使用。

9.3 发送操作

通过将要发送的数据写入UART_TX_DATA缓冲区来启动发送。然后将数据与起始位、奇偶校验位和停止位（由线路控制寄存器中的设置确定）一起传输到TX移位寄存器。随后使用波特率发生器的输出（16分频）作为时钟，将要发送的位以起始位、数据位（LSb在前）、奇偶校验位和停止位的顺序移出TX移位寄存器。如果允许TX保持寄存器为空中断，当UART_TX_DATA缓冲区中的数据发出后变为空时将产生该中断。

PCI1xxxx一次最多可存储256字节的发送数据。发送操作将持续至TX FIFO为空。FIFO是否准备好接受更多数据将由中断指示。

9.4 接收操作

可使用16分频（默认值）的接收时钟将数据采样到RX移位寄存器中。接收时钟由波特率发生器提供。使用滤波器将持续时间短于两个接收时钟周期的杂散输入滤除。当完整的字随时钟移入接收器时，数据位会传输到由PCIe主机读取的UART_RX_DATA（将待接收的数据的第一位放在该寄存器的bit 0）。接收器还会检查奇偶校验位和停止位是否与线路控制寄存器中指定的一致。

如果使能RX数据已接收中断，则在数据传输到RX缓冲寄存器时或达到RX触发电平时将产生该中断。此外，也可以通过产生中断信号来指示RX FIFO字符超时、奇偶校验错误、停止位丢失（帧错误）或其他线路状态错误。PCI1xxxx一次最多可存储256字节的接收数据。根据所选的RX触发电平，当RX FIFO包含2至250个字节的数据时，中断将激活以指示数据可用。具体字节数的计算如下：

((UART_FIFO_CTL.RECV_FIFO_TRIG[4:0] * 8) + 2)个字节

9.5 波特率时钟

注：	波特率可通过大多数通用UART/COM端口终端应用程序设置（如minicom/PuTTY）进行常规配置，PCI1xxxx UART驱动程序将设置相应的BAUD_CLOCK_DIV*寄存器值。在典型应用中，无需手动设置这些寄存器。

波特率时钟源自内部PLL，以确保在各种频率下保持所需的波特率精度；这通过CLK_SEL_REG.BAUD_CLOCK_SEL[1:0]进行选择，默认为62.5 MHz。

此外，还可以选择500 MHz和166.667 MHz时钟，但与62.5 MHz相比，既不能提供更高的精度，也无法支持更多波特率，因此本文档中未提供相关设置。

通过设置BAUD_CLK_SEL = 00b选择62.5 MHz时钟分频模式。然后从CLK_DIVISOR_REG.BAUD_CLOCK_DIV_INT[23:0]中选择一个分频值来配置最终的波特率。有关可用值，请参见表84。

表84： 62.5 MHz时钟分频模式波特率

波特率	实际波特率	百分比误差	采样数/UART位	BAUD_CLOCK_DIV_INT[23:0]值	BAUD_CLOCK_DIV_FRAC[7:0]值
50	50.0000	0.0	16	0x13_12D0	0x00
75	75.0000	0.0000	16	0x0C_B735	0x55
110	110.0000	0.0000	16	0x08_AB75	0xD1
134.5	134.5000	0.0000	16	0x07_172C	0xD4
150	150.0000	0.0000	16	0x06_5B9A	0xAA
300	300.0000	0.0000	16	0x03_2DCD	0x55
600	600.0000	0.0000	16	0x01_96E6	0xAA
1.2k	1,200.0000	0.0000	16	0x00_CB73	0x55
1.8k	1,799.9999	0.0000	16	0x00_87A2	0x39

表84： 62.5 MHz时钟分频模式波特率（续）

波特率	实际波特率	百分比误差	采样数/UART位	BAUD_CLOCK_DIV_INT[23:0]值	BAUD_CLOCK_DIV_FRAC[7:0]值
2k	2,000.0000	0.0000	16	0x00_7A12	0x00
2.4k	2,400.0000	0.0000	16	0x00_65B9	0xAA
3.6k	3,600.0003	0.0000	16	0x00_43D1	0x1C
4.8k	4,799.9993	0.0000	16	0x00_32DC	0xD5
7.2k	7,199.9989	0.0000	16	0x00_21E8	0x8E
9.6k	9,600.0014	0.0000	16	0x00_196E	0x6A
19.2k	19,200.0029	0.0000	16	0x00_0CB7	0x35
38.4k	38,400.0058	0.0000	16	0x00_065B	0x9A
57.6k	57,599.9393	-0.0001	16	0x00_043D	0x12
115.2k	115,200.2949	0.0003	16	0x00_021E	0x88
125k	125,000.0000	0.0000	16	0x00_01F4	0x00
136.4k	136,399.8151	-0.0001	16	0x00_01CA	0x36
150k	150,000.0000	0.0000	16	0x00_01A0	0xAA
166.7k	166,699.7887	-0.0001	16	0x00_0176	0xEC
187.5k	187,500.0000	0.0000	16	0x00_014D	0x55
214.3k	214,300.1210	0.0001	16	0x00_0123	0xA5
250k	250,000.0000	0.0000	16	0x00_00FA	0x00
300k	300,000.0000	0.0000	16	0x00_00D0	0x55
375k	375,000.0000	0.0000	16	0x00_00A6	0xAA
500k	500,000.0000	0.0000	16	0x00_007D	0x00
750k	750,000.0000	0.0000	16	0x00_0053	0x55
921.6k	921,615.6826	0.0017	16	0x00_0043	0xD0
1M	999,968.6284	-0.0031	16	0x00_003E	0x80
1.5M	1,500,000.0000	0.0000	16	0x00_0029	0xAA
2M	1,999,937.2569	-0.0031	16	0x00_001F	0x40
3M	2,999,717.6736	-0.0094	16	0x00_0014	0xD5
4M	3,999,874.5137	-0.0031	8	0x00_001F	0x40
5M	5,000,000.0000	0.0000	8	0x00_0019	0x00
6M	5,999,435.3473	-0.0094	8	0x00_0014	0xD5
7M	6,999,341.2385	-0.0094	8	0x00_0011	0xDB
8M	7,999,749.0275	-0.0031	4	0x00_001F	0x40
9M	9,000,423.5493	0.0047	4	0x00_001B	0xC6
10M	10,000,000.0000	0.0000	4	0x00_0019	0x00
11M	11,000,862.8128	0.0078	4	0x00_0016	0xB9
12M	11,998,870.6945	-0.0094	4	0x00_0014	0xD5
13M	12,999,592.1697	-0.0031	4	0x00_0013	0x3B
14M	13,998,682.5	-0.0094	4	0x00_0011	0xDB
15M	15,000,000.0	0.0000	4	0x00_0010	0xAA

9.6 UART子系统寄存器

表85列出了PERI_SUBSYSTEM_ADDR_BASE = 0x0014_0000时可用的寄存器。

注： PCI1xxxx器件包含数千个寄存器，其中一部分寄存器保留供将来使用，另一部分寄存器的内容未纳入公开文档，因为其不涉及终端系统集成商用例（即：这些寄存器供硬件用于执行底层运行时任务，或在运行时由驱动程序直接使用/控制）。**请勿修改未纳入文档的寄存器的内容。**

表85： UART子系统寄存器地址相对于PERI_SUBSYSTEM_ADDR_BASE的偏移量

寄存器名称	端口1	端口2	端口3	端口4
ADCL_CFG_REG	0x0_0040	0x0_0440	0x0_0840	0x0_0C40
CLK_SEL_REG	0x0_0050	0x0_0450	0x0_0850	0x0_0C50
CLK_DIVISOR_REG	0x0_0054	0x0_0454	0x0_0854	0x0_0C54
FRAC_DIV_CFG_REG	0x0_0058	0x0_0458	0x0_0858	0x0_0C58
UART_PAD_CTRL_PU_REG	0x0_0060	0x0_0460	0x0_0860	0x0_0C60
UART_PAD_CTRL_PD_REG	0x0_0064	0x0_0464	0x0_0864	0x0_0C64
UART_PAD_CTRL_OD_REG	0x0_0068	0x0_0468	0x0_0868	0x0_0C68
UART_PAD_CTRL_OPOL_REG	0x0_006C	0x0_046C	0x0_086C	0x0_0C6C
UART_PAD_CTRL_WAKE_DB	0x0_0070	0x0_0470	0x0_0870	0x0_0C70
UART_PAD_CTRL_CTS_WAKE_DB	0x0_0074	0x0_0474	0x0_0874	0x0_0C74
UART_PCI_CTRL_REG	0x0_0080			
UART_INT_REG	0x0_0084	0x0_0484	0x0_0884	0x0_0C84
UART_INT_MASK_REG	0x0_0088	0x0_0488	0x0_0888	0x0_0C88
UART_WAKE_REG	0x0_008C	0x0_048C	0x0_088C	0x0_0C8C
UART_WAKE_MASK_REG	0x0_0090	0x0_0490	0x0_0890	0x0_0C90
UART_RESET_REG	0x0_0094			
UART_LTR_VALUE_REG	0x0_0098	0x0_0498	0x0_0898	0x0_0C98

表86列出了自动方向控制寄存器的详细信息。

表86： ADCL_CFG_REG

自动方向控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:3	Reserved	R	始终为0
2	ADCL_POLARITY	R/W	选择自动方向控制（ADCL）期间相应串行端口的输出缓冲区为空或已满时所选信号被驱动的极性（高电平或低电平）。 0b：0表示为空，1表示已满。 1b：1表示为空，0表示已满。
1	ADCL_PIN_SEL	R/W	该使能位用于选择受自动方向控制（ADCL）影响的串行端口引脚（nRTS或nDTR）。 0b：ADCL使用nDTR。 1b：ADCL使用nRTS。

表86： ADCL_CFG_REG（续）

自动方向控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
0	ADCL_EN	R/W	使能或禁止自动方向控制（ADCL）。 0b：禁止ADCL。 1b：使能ADCL。

表87列出了时钟源选择寄存器的详细信息。

表87： CLK_SEL_REG

时钟源选择寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:2	Reserved	R	始终返回0
1:0	BAUD_CLOCK_SEL[1:0]	R/W	选择输入波特率时钟： 00b：62.5 MHz（默认值） 01b：166.667 MHz（500 MHz/3） 10b：500 MHz 11b：保留

表88列出了时钟分频比寄存器的详细信息。

表88： CLK_DIVISOR_REG

时钟分频比寄存器			默认值：0x0000_0015
Bit	**名称**	**R/W**	**说明**
31:8	BAUD_CLOCK_DIV_INT[23:0]	R/W	选择应用于波特率时钟（由CLK_SEL_REG.BAUD_CLOCK_SEL[1:0]选择）的分频比的整数部分，以生成正确的波特率。
7:0	BAUD_CLOCK_DIV_FRAC[7:0]	R/W	选择应用于波特率时钟（由CLK_SEL_REG.BAUD_CLOCK_SEL[1:0]选择）的分频比的小数部分，以生成正确的波特率。 **注：**BAUD_CLOCK_DIV_FRAC[7:0]仅在CLK_SEL_REG.BAUD_CLOCK_SEL[1:0]设置为0时有效。

表89列出了小数分频比配置寄存器的详细信息。

表89： FRAC_DIV_CFG_REG

小数分频器配置寄存器			默认值：0x6EF7_1000
Bit	**名称**	**R/W**	**说明**
31:28	TX_HALF_POINT[3:0]	R/W	发送UART停止位时，使用第N个采样脉冲发送UART后半部分停止位。对于16倍采样法，使用第6个采样。对于8倍采样法，可使用第2个采样。
27:24	TX_NEXT_END_POINT[3:0]	R/W	发送UART停止位时，使用第N个采样脉冲发送UART第二个完整停止位。对于16倍采样法，使用第14个采样。对于8倍采样法，可使用第6个采样。

表89： FRAC_DIV_CFG_REG（续）

小数分频器配置寄存器			默认值： 0x6EF7_1000
Bit	**名称**	**R/W**	**说明**
23:20	TX_END_POINT[3:0]	R/W	发送UART位时，使用第N个采样脉冲发送UART数据。对于16倍采样法，使用第15个采样。对于8倍采样法，可使用第7个采样。 此外，该值也供RX和TX引擎用于复位步进计数器，从而表示16倍采样法或8倍采样法。
19:16	RX_SAMPLE_POINT[3:0]	R/W	接收UART位时，将位的第N个采样的记录为接收位。对于16倍采样法，使用第7个采样。对于8倍采样法，可使用第3个采样。
15:0	SAMPLE_CLK_PERIOD[15:0]	R/W	该寄存器值表示波特率发生器计数器中的步进增量。

表90列出了UART焊盘控制上拉寄存器的详细信息。

表90： UART_PAD_CTRL_PU_REG

UART焊盘控制寄存器： 内部上拉电阻			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:6	Reserved	R	始终为0。
5	CTS_PU	R/W	0b：禁止焊盘上的内部上拉电阻 1b：使能焊盘上的内部上拉电阻
4	RTS_PU	R/W	0b：禁止焊盘上的内部上拉电阻 1b：使能焊盘上的内部上拉电阻
3	RI_PU	R/W	0b：禁止焊盘上的内部上拉电阻 1b：使能焊盘上的内部上拉电阻
2	DSR_PU	R/W	0b：禁止焊盘上的内部上拉电阻 1b：使能焊盘上的内部上拉电阻
1	DCD_PU	R/W	0b：禁止焊盘上的内部上拉电阻 1b：使能焊盘上的内部上拉电阻
0	DTR_PU	R/W	0b：禁止焊盘上的内部上拉电阻 1b：使能焊盘上的内部上拉电阻

表91列出了UART焊盘控制下拉寄存器的详细信息。

表91： UART_PAD_CTRL_PD_REG

UART焊盘控制寄存器： 内部下拉电阻			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:6	Reserved	R	始终为0。
5	CTS_PD	R/W	0b：禁止焊盘上的内部下拉电阻 1b：使能焊盘上的内部下拉电阻
4	RTS_PD	R/W	0b：禁止焊盘上的内部下拉电阻 1b：使能焊盘上的内部下拉电阻

表91： UART_PAD_CTRL_PD_REG（续）

UART焊盘控制寄存器：内部下拉电阻			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
3	RI_PD	R/W	0b：禁止焊盘上的内部下拉电阻 1b：使能焊盘上的内部下拉电阻
2	DSR_PD	R/W	0b：禁止焊盘上的内部下拉电阻 1b：使能焊盘上的内部下拉电阻
1	DCD_PD	R/W	0b：禁止焊盘上的内部下拉电阻 1b：使能焊盘上的内部下拉电阻
0	DTR_PD	R/W	0b：禁止焊盘上的内部下拉电阻 1b：使能焊盘上的内部下拉电阻

表92列出了UART焊盘控制漏极开路寄存器的详细信息。

表92： UART_PAD_CTRL_OD_REG

UART焊盘控制寄存器：漏极开路操作			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:2	Reserved	R	始终为0。
1	RTS_OD	R/W	0b：禁止漏极开路操作 1b：如果相应的UARTx_RTS_N引脚配置为漏极开路，当相应输出为1时，禁止输出并使能上拉电阻；当输出为0时，使能输出并将其驱动为0。
0	DTR_OD	R/W	0b：禁止漏极开路操作 1b：如果相应的UARTx_DTR_N引脚配置为漏极开路，当相应输出为1时，禁止输出并使能上拉电阻；当输出为0时，使能输出并将其驱动为0。

表93列出了UART焊盘控制极性寄存器的详细信息。

表93： UART_PAD_CTRL_OPOL_REG

UART焊盘控制寄存器：引脚极性			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:2	Reserved	R	始终为0。
1	RTS_OD	R/W	0b：默认输出极性 1b：输出极性反相
0	DTR_OD	R/W	0b：默认输出极性 1b：输出极性反相

表94列出了UART焊盘控制WAKE_N去抖寄存器的详细信息。

表94： UART_PAD_CTRL_WAKE_DB

UART焊盘控制寄存器：WAKE_N去抖			默认值：0x0000_000A
Bit	**名称**	**R/W**	**说明**
31:17	Reserved	R	始终为0。

表94： UART_PAD_CTRL_WAKE_DB（续）

UART焊盘控制寄存器： WAKE_N去抖			默认值： 0x0000_000A
Bit	**名称**	**R/W**	**说明**
16	UART_WAKE_DB_EN	R/W	UARTx_WAKE_N焊盘的去抖使能。 **注：**该引脚可配置为禁止去抖器。在这种情况下，直接传递原始输入。
15:12	Reserved	R	始终为0。
11:0	UART_CTS_WAKE_DB_ TIME[11:0]	R/W	该寄存器用于保存UARTx_WAKE_N焊盘的去抖定时器值。 当检测到信号跳变时，去抖器启动。如果引脚状态在配置的去抖时间内保持不变，则直接传递该跳变信号。如果在去抖时间内再次发生信号跳变，去抖器将重新启动。 每个计数对应1 ms，默认值为10 ms。由于计时基于自由运行时钟，因此会随边沿对齐情况而变化。计时精度限制为±一个计数。

表95列出了UART焊盘控制CTS去抖寄存器的详细信息。

表95： UART_PAD_CTRL_CTS_WAKE_DB

UART焊盘控制寄存器： CTS去抖			默认值： 0x0000_000A
Bit	**名称**	**R/W**	**说明**
31:17	Reserved	R	始终为0。
16	UART_CTS_WAKE_DB	R/W	唤醒期间nCTS焊盘的去抖使能。 **注：**使用nCTS唤醒UART时应用去抖；在正常工作期间不应用。
15:12	Reserved	R	始终为0。
11:0	UART_CTS_WAKE_DB_ TIME[11:0]	R/W	该寄存器用于保存唤醒期间nCTS焊盘的去抖定时器值。 当检测到信号跳变时，去抖器启动。如果引脚状态在配置的去抖时间内保持不变，则直接传递该跳变信号。如果在去抖时间内再次发生信号跳变，去抖器将重新启动。 每个计数对应1 ms，默认值为10 ms。由于计时基于自由运行时钟，因此会随边沿对齐情况而变化。计时精度限制为±一个计数。

表96列出了UART PCI控制寄存器的详细信息。

表96： UART_PCI_CTRL_REG

UART PCI控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:5	Reserved	R	始终为0
4	UART_MSI_VECTOR_SEL	R/W	选择每个实例使用不同的MSI/MSI-X向量，还是所有实例都使用相同的MSI/MSI-X向量： 0b：所有实例都使用第一个MSI/MSI-X向量（0） 1b：每个实例使用不同的MSI/MSI-X向量
3:1	Reserved	R	始终为0
0	UART_D3_CLK_EN	R/W	在PCI功能处于D3状态时使能62.5 MHz时钟： 0b：在PCI功能处于D3状态时禁止62.5 MHz时钟。 1b：在PCI功能处于D3状态时使能62.5 MHz时钟。

表97列出了UART WAKE_N引脚中断寄存器的详细信息。

表97： UART_INT_REG

UART WAKE_N引脚中断寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:1	Reserved	R	始终为0
0	UART_WAKE_N_PIN_INT	R/W1C	置1时表示UARTx_WAKE_N引脚置为有效。

表98列出了UART WAKE_N中断屏蔽寄存器的详细信息。

表98： UART_INT_MASK_REG

UART WAKE_N中断屏蔽寄存器			默认值：0x0000_0001
Bit	**名称**	**R/W**	**说明**
31:1	Reserved	R	始终为0
0	UART_WAKE_N_PIN_INT_MASK	R/W	UART_WAKE_N_PIN_INT状态位屏蔽： 0b：在UART_WAKE_N_PIN_INT状态位置1时生成UART_IRQ事件。 1b：在UART_WAKE_N_PIN_INT状态位置1时禁止生成UART_IRQ事件。

表99列出了UART WAKE寄存器的详细信息。

表99： UART_WAKE_REG

UART WAKE寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:7	Reserved	R	始终为0
2	UART_WAKE_N_PIN_WAKE	R/W1C	置1时表示UARTx_WAKE_N引脚置为有效

表99： UART_WAKE_REG（续）

UART WAKE寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
1	UART_NCTS_WAKE	R/W1C	置1时表示nCTS引脚在UART处于挂起状态时置为有效
0	UART_INT_WAKE	R/W1C	置1时表示至少一个UART_IIR状态位在UART未处于挂起状态时置为有效

表100列出了UART WAKE屏蔽寄存器的详细信息。

表100： UART_WAKE_MASK_REG

UART WAKE屏蔽寄存器			默认值：0x0000_00007
Bit	**名称**	**R/W**	**说明**
31:3	Reserved	R	始终为0
2	UART_WAKE_N_PIN_WAKE_MASK	R/W	UART_WAKE_N_PIN_WAKE状态位屏蔽： 0b：在UART_WAKE_N_PIN_WAKE状态位置1时生成UART_WAKE事件。 1b：在UART_WAKE_N_PIN_WAKE状态位置1时禁止生成UART_WAKE事件。
1	UART_NCTS_WAKE_MASK	R/W	UART_NCTS_WAKE状态位屏蔽： 0b：在UART_NCTS_WAKE状态位置1时生成UART_WAKE事件。 1b：在UART_NCTS_WAKE状态位置1时禁止生成UART_WAKE事件。
0	UART_INT_WAKE_MASK	R/W	UART_INT_WAKE状态位屏蔽： 0b：在UART_INT_WAKE状态位置1时生成UART_WAKE事件。 1b：在UART_INT_WAKE状态位置1时禁止生成UART_WAKE事件。

表101列出了UART复位寄存器的详细信息。

表101： UART_RESET_REG

UART复位寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:17	Reserved	R	始终为0
16	PERI_UART_D3_RESET_DIS	R/W	D3复位禁止 当退出D3Cold状态（PCIE_PERST_N置为无效）时，该位用于修改子系统复位操作。当该位置1且VAUX_DET引脚为高电平时，仅复位PCIe®接口及相关逻辑；不复位UART。 其他情况下，复位整个子系统。
15:10	Reserved	R	始终为0

表101： UART_RESET_REG（续）

UART复位寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
9	PERI_UART_RELOAD_PCI	R/W	定义PERI_UART_RELOAD置1时在外设UART PCIe端点上执行的配置；对应PERI_UART_PCIE_EP_ADDR_BASE。 设置为1时，外设UART PCIe端点配置将重载到PERI_UART_PCIE_EP_ADDR_BASE。 设置为0时，外设UART PCIe端点PF0配置不会重载到PERI_UART_PCIE_EP_ADDR_BASE。系统配置硬件将尝试写入与特定配置相关的所有配置寄存器；子系统将阻止与PERI_UART_PCIE_EP_ADDR_BASE处的PCIe端点对应的任何写操作。
8	PERI_UART_RELOAD_FN	R/W	定义PERI_UART_RELOAD置1时在外设UART功能IP上执行的配置；外设SMBus功能IP对应USB_SUBSYSTEM_ADDR_BASE中的绝大多数子系统地址，只有对应PERI_UART_PCIE_EP_ADDR_BASE的地址除外，这些地址由PERI_UART_RELOAD_PCI位控制。 设置为1时，将重载外设UART功能IP配置。 设置为0时，不会重载外设UART功能IP配置。系统配置硬件将尝试写入与特定配置相关的所有配置寄存器；子系统将阻止与功能IP对应的任何写操作。
7:3	Reserved	R	始终为0
2	PERI_UART_RELOAD	R/W/ SC	启动外设UART功能重载 **注：** 在将PERI_UART_RELOAD置1之前，应正确配置PERI_UART_RELOAD_PCI和PERI_UART_RELOAD_FN。该位在重载完成后自清零。
1	PERI_UART_LRST	R/W/ SC	启动外设UART功能LRST **注：**该位在LRST完成后自清零。
0	PERI_UART_SRST	R/W/ SC	启动外设子系统SRST **注：**该位在SRST完成后自清零。

表102列出了UART_LTR_VALUE_REG寄存器的详细信息。

表102： UART_LTR_VALUE_REG

UART_LTR_VALUE_REG			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	UART_NO_SNOOP_REQ	R/W	无侦听需求 0b：无延时需求，忽略UART_NO_SNOOP_LATENCY_VAL[9:0]和UART_NO_SNOOP_LATENCY_SCALE[2:0]位域。 1b：有延时需求，具体取决于UART_NO_SNOOP_LATENCY_VAL[9:0]和UART_NO_SNOOP_LATENCY_SCALE[2:0]位域。
30:29	Reserved	R	始终为0
28:26	UART_NO_SNOOP_LATENCY_SCALE[2:0]	R/W	UART_NO_SNOOP_LATENCY_VAL[9:0]位域调整： 000b：值乘以1 ns 001b：值乘以32 ns 010b：值乘以1024 ns 011b：值乘以32768 ns 100b：值乘以1048576 ns 101b：值乘以33554432 ns 110b：不允许值执行乘法 111b：不允许值执行乘法
25:16	UART_NO_SNOOP_LATENCY_VAL[9:0]	R/W	无侦听延时值
15	UART_SNOOP_REQ	R/W	侦听需求 0b：无延时需求，忽略UART_SNOOP_LATENCY_VAL[9:0]和UART_SNOOP_LATENCY_SCALE[2:0]位域。 1b：有延时需求，具体取决于UART_SNOOP_LATENCY_VAL[9:0]和UART_SNOOP_LATENCY_SCALE[2:0]位域。
14:13	Reserved	R	始终为0
12:10	UART_SNOOP_LATENCY_SCALE[2:0]	R/W	UART_SNOOP_LATENCY_VAL[9:0]位域调整： 000b：值减去1 ns 001b：值减去32 ns 010b：值减去1024 ns 011b：值减去32768 ns 100b：值减去1048576 ns 101b：值减去33554432 ns 110b：保留 111b：保留
9:0	UART_SNOOP_LATENCY_VAL[9:0]	R/W	侦听延时值

10.0 外设子系统实现——SMBus控制器

SMBus控制器子系统占用自己的PCI器件端点，需要主机系统提供独立的器件驱动程序才能正常工作。该子系统与所有其他外设（UART、SPI和GPIO）共用PCI总线。

子系统ID如下：

- 供应商ID：0x1055（Microchip Technology, Inc/SMSC）
- 器件ID：
  - PCI12000：0xA003
  - PCI11010：0xA013
  - PCI11101：0xA023
  - PCI11400：0xA033
  - PCI11414：0xA043
- 分类ID：0x0C0500（“SMBus控制器”）

SMBus接口具有3个引脚：

- SMBUS_CTLR_SCL——时钟信号
- SMBUS_CTLR_SDA——数据信号
- SMBUS_CTLR_ALERT_N——报警信号

支持三种速度：

- 100 kHz
- 400 kHz
- 1 MHz

注： PCI1xxxx符合SMBus 2.0规范，但包含SMBus 3.0规范中引入的部分功能。其中包括支持1 MHz和SMBus报警。

10.1 章节

10.2 接口/速度配置

注： I^2C/SMBus驱动程序将根据通过GPR0_REG所做的配置（写入PCI1xxxx配置（OTP/EEPROM）），自动配置表103中列出的寄存器。

必须正确设置SCL_PAD_CTRL和SDA_PAD_CTRL寄存器以使能I/O焊盘。这两个寄存器（表103）对来自SMBus控制器内核的焊盘控制信号进行门控。

部分AC时序参数需根据模块的内部延时进行调整。对于PCI1xxxx，必须使用正确的AC时序参数重新编程表103中的寄存器。

表103： 不同速度的SMBus控制器寄存器设置

寄存器	位域	默认值					
		100 kHz		400 kHz		1 MHz	
		时间	值	时间	值	时间	值
SR_HOLD_ TIME_REG	SR_HOLD_TIME[7:0]	4 µs	0x85	0.6 µs	0x14	0.26 µs	0x0B
IDLE_ SCALAING_REG	FAIR_BUS_IDLE_ MIN[11:0]	31 µs	0x3C9	5	0x09D	5 µs	0x09D
	FAIR_IDLE_DELAY[11:0]	32 µs	0x3E8	16	0x1F4	16 µs	0x1F4
BUS_CLOCK_REG	HIGH_PERIOD[7:0]/ LOW_PERIOD[7:0]	n/a	0x9A9C	n/a	0x2329	n/a	0x0E0F
CLKSYNC_ REGISTER	CLKSYNC[31:0]	100 ns	0x4	100	0x4	100 ns	0x4
DATA_TIMING_REG	FIRST_START_HOLD[7:0]	4 µs	0x16	0.6 µs	0x10	0.26 µs	0x06
	STOP_SETUP[7:0]	4 µs	0x9D	0.6 µs	0x14	0.26 µs	0x0C
	RESTART_SETUP[7:0]	4.7 µs	0x9D	0.6 µs	0x14	0.26 µs	0x0C
	DATA_HOLD[7:0]	0 ns	0x01	0 ns	0x01	0 ns	0x01
TO_SCALING_REG	BUS_IDLE_MIN[7:0]	4.7 µs	0xA7	1.3 µs	0x8B	0.5 µs	0x85
	CTLR_CUM_ TIME-OUT[7:0]	10 ms	0x9F	10 ms	0x9F	10 ms	0x9F
	TARGET_CUM_ TIME-OUT[7:0]	25 ms	0xC7	25 ms	0xC7	25 ms	0xC7
	CLOCK_HIGH_ TIME-OUT[7:0]	50 µs	0xCC	50 µs	0xCC	50 µs	0xCC

10.3 I²C/SMBus缓冲区

当主机软件向SMBus发送数据时，必须先将数据包存入缓冲区，然后使能SMBus控制器内核，从缓冲区向SMBus目标传输数据，每次一个数据字节。

从I²C总线接收数据时，SMBus控制器内核将向缓冲区传输数据，每次一个数据字节。当收到的数据包终止后，主机软件必须通过读取缓冲区中的所有字节来处理缓冲区。

缓冲区共有两个。每个缓冲区一次只能在一种模式（接收或发送）下工作。软件必须跟踪传输方向，并使用SMBUS_CONTROL.TRANSFER_DIR针对每次传输正确编程缓冲区。

当SMBUS_CONTROL.RUN设置为1b'1时，使能缓冲区。这样，缓冲区可通过端口在自身与SMBus控制器内核之间传输字节。当控制器的DMA_REQ信号置为有效时，SMBus控制器内核与缓冲区之间才真正发生字节传输。发生字节传输时，缓冲区会根据 SMBUS_CONTROL.TRANSFER_DIR 丢弃或获取来自控制器的字节。字节传输完成后，SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]位域会递增。

为终止缓冲区操作，SMBus控制器内核会通过SMBUS_STATUS.DMA_TERM生成相应的事件。这表示缓冲区在接收操作中已被填满，或在发送操作中已被清空。当缓冲区操作终止时，硬件会将SMBUS_CONTROL.RUN位域自动清除为0b。

软件只能在缓冲区未工作或SMBUS_CONTROL.RUN清除为0b时读写缓冲区。当缓冲区使能时，只有SMBus控制器内核可以访问缓冲区。

软件可通过两种方式访问缓冲区：

1. 在直接访问模式下，软件可通过处理器直接访问缓冲区内容（偏移地址范围为0x80至0xFF）。在直接访问模式下，SMBUS_CONTROL.HOST_FIFO_ENTRY清零。
2. 当SMBUS_CONTROL.HOST_FIFO_ENTRY置1时，将缓冲区视为FIFO进行访问。在该模式下，软件可通过SMBUS_FIFO_DATA寄存器间接访问缓冲区。缓冲区单元由SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]位域指示。任何读写SMBUS_FIFO_DATA.FIFO_DATA[7:0]位域的操作会导致SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]位域递增，使其指向下一个缓冲区单元。

10.3.1 缓冲区发送操作

按照以下步骤使能缓冲区以向SMBus发送数据：

1. 软件必须将SMBUS_MCU_COUNTER和SMBUS_DEV_COUNTER寄存器都复位为零。这可通过直接向这两个寄存器写入零或通过将SMBUS_CONTROL寄存器中的RESET_COUNTERS置1来完成。
2. 软件必须清零SMBUS_CONTROL寄存器中的TRANSFER_DIR位域，从而将缓冲区设置为发送模式。
3. 软件随后必须通过FIFO条目模式或直接访问模式将数据包存入缓冲区。使用FIFO条目模式时，每次写入SMBUS_FIFO_DATA.FIFO_DATA[7:0]位域都会自动递增SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]位域。使用直接访问模式时，软件必须将存入缓冲区的总字节数写入SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]位域。
4. 当数据包存入缓冲区后，通过将SMBUS_CONTROL寄存器中的RUN置1来使能缓冲区。发生该事件后，从缓冲区读取字节并传递给SMBus控制器内核进行发送。
5. 对于从缓冲区读取并传输到SMBus控制器内核的每个字节，SMBUS_DEV_COUNTER寄存器中的SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]位域都会递增。SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]位域会持续递增，直到达到数据包大小为止。
6. 当数据包发送完成后，BUF_EMPTY和/或DMA_TERM将置1。SMBUS_INTERRUPT_STATUS寄存器中相应的I^2C中断（若已配置）随后将置为有效。
7. 硬件自动清零RUN以结束缓冲区发送操作。

注： 数据包大小不能超过SMBUS_BUF_DEPTH.BUF_DEPTH[7:0]位域指定的量。

10.3.2 缓冲区接收操作

按照以下步骤使能缓冲区以向SMBus发送数据：

1. 软件必须将SMBUS_MCU_COUNTER和SMBUS_DEV_COUNTER寄存器都复位为零。这可通过直接向这两个寄存器写入零或通过将SMBUS_CONTROL寄存器中的RESET_COUNTERS置1来完成。
2. 软件必须将SMBUS_CONTROL寄存器中的TRANSFER_DIR位域置1，从而将缓冲区设置为接收模式。
3. 软件将SMBUS_CONTROL寄存器中的RUN置1。这样可以使用来自SMBus控制器内核的数据填充缓冲区。
4. 每当一个字节写入缓冲区时，SMBUS_DEV_COUNTER寄存器中的SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]位域都会递增。
5. SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]位域会持续递增，直到数据包接收完成为止。
6. 数据包接收完成后，硬件将RUN清零。
7. 相应的I^2C缓冲区中断（若已配置）将置为有效并通知MCU。
8. 软件通过读取SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]位域来确定接收的字节数。
9. 软件随后通过FIFO条目模式或直接访问模式清空缓冲区。使用FIFO条目模式时，每次读取SMBUS_FIFO_DATA.FIFO_DATA[7:0]位域都会自动递增SMBUS_MCU_COUNTER寄存器中的SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]位域。当固件清空缓冲区后，缓冲区操作结束。

10.3.3 缓冲区为空

当SMBus控制器内核在SMBus上的发送操作结束前已完全耗尽缓冲区时，会发生缓冲区为空情况。在从缓冲区读取最后一个字节并写入SMBus控制器内核后，SMBUS_INTERRUPT_STATUS寄存器中的BUF_EMPTY位立即置为有效，SMBUS_CONTROL寄存器中的RUN位立即清零。

如果SMBUS_INTERRUPT_MASK寄存器中的INT_STAT_BUF_EMPTY位清零，则触发相应的缓冲区中断信号。软件随后可处理中断，将下一组发送数据包存入缓冲区并再次将RUN置1。为了避免SMBus欠载，SMBus控制器内核可能会自动调用时钟延长。这取决于软件处理中断的速度。时钟延长在新数据存入缓冲区且RUN置1后终止。

10.3.4 缓冲区已满

当SMBus控制器内核在SMBus上的接收操作结束前已将缓冲区填充至SMBUS_BUF_DEPTH.BUF_DEPTH[7:0]大小时，会发生缓冲区已满情况。在从SMBus控制器内核读取最后一个字节并写入缓冲区且达到SMBUS_BUF_DEPTH.BUF_DEPTH[7:0]大小后，SMBUS_STATUS.BUF_FULL立即置为有效。SMBUS_CONTROL寄存器中的RUN位由硬件自动清零。

如果SMBUS_INTERRUPT_MASK寄存器中的INT_STAT_BUF_FULL位清零，则触发相应的缓冲区中断信号。软件随后可以处理中断，清空缓冲区并再次将RUN置1。为了避免SMBus欠载，SMBus控制器内核可能会自动调用时钟延长。这取决于软件处理中断的速度。时钟延长在缓冲区被清空且RUN置1后终止。

10.4 挂起

当SMBUS_PCI_CTRL_REG.SMB_D3_CLK_EN位设置为0时，如果SMBus控制器PCIe端点功能进入D3状态，将关闭提供给SMBus模块的62.5 MHz时钟。这被定义为SMBus模块处于挂起状态。当PCI功能进入D0状态时，将恢复时钟。

当SMBUS_PCI_CTRL_REG.SMB_D3_CLK_EN位设置为1时，即使SMBus控制器PCIe端点功能进入D3状态，也不会关闭提供给SMBus模块的62.5 MHz时钟；该模块将继续工作。这也意味着即使所有外设PCIe端点功能都处于D3状态，CLK_REQ_REG.PERI_CRYSTAL_REQ和CLK_REQ_REG.PERI_WR_PLL_REQ也都必须设置为1。这被定义为SMBus模块未处于挂起状态。

10.5 SMBus报警

SMBus报警信号为线与信号，就像SMBCLK和SMBDAT信号一样。SMBALERT#与SMBus报警响应地址（Alert Response Address，ARA）一起使用。经过配置后，PCI1xxxx通过SMBUS_CTLR_ALERT_N引脚提供该信号作为SMBus控制器的输入。

SMBus目标器件可通过SMBALERT#向SMBus主机发出信号以开始通信。主机将处理中断，并通过报警响应地址同时访问所有SMBALERT#器件。只有将SMBALERT#拉为低电平的器件才会应答报警响应地址。主机执行修改后的接收字节操作。SMBus目标发送器件提供的7位器件地址放入字节的高7位中。第8位可以是0或1。

当SMBUS_CTLR_ALERT_N引脚被拉为低电平时，表明有一个或多个SMBus目标需要与SMBus主机通信。当PCIe端点功能处于D3状态时，将触发SMBUS_CTLR_WAKE事件。

当PCIe端点功能处于D0状态时，将产生SMBUS_CTLR_WAKE中断。SMBUS_CTLR_ALERT_N引脚的焊盘控制在SMBALERT_CTLR_PAD_CTRL_REG寄存器中提供。用于触发中断的输入去抖时间可在SMBALERT_CTLR_DB_REG.SMBUS_CTLR_DB_TIME[11:0]中进行设置。SMBUS_CTLR_ALERT_N引脚的当前值和去抖后的值都在SMBALERT_CTLR_VAL_REG寄存器中提供。

注： 当SMBUS_CTLR_ALERT_N引脚用于唤醒系统且未使能去抖器（SMBALERT_CTLR_PAD_CTRL_REG.SMBALERT_CTLR_DB = 0）时，需通过SMBus目标将SMBUS_CTLR_ALERT_N置为有效并至少持续6 µs左右。请求环形振荡器时钟需要4 µs左右，向主机产生唤醒报文需要2 µs。

10.6 中断

SMBUS_CTLR_IRQ中断由集成的SMBus控制器控制。有关其使用的详细信息，请参见相应的规范。这用于通过PCIe生成MSI/MSI-X事件。

SMBus控制器具有两个中断源：

1. I2C_Intr —— 该中断由SMBus控制器内核置为有效。

   例如，SMBUS_STATUS寄存器中的CDONE、IDLE、AAS和PIN中断等。

2. I2C_buf_ctlr_intr —— 由SMBus控制器内核SMBus网络引擎操作置为有效。

   例如，SMBUS_INTERRUPT_STATUS寄存器中的BUF_FULL、BUF_EMPTY、SMBUS_STATUS.THRESHOLD_HIT和DMA_TERM中断等。

I2C_Intr和I2C_buf_ctlr_intr都产生SMBUS_CTLR_IRQ中断。可通过查看SMBUS_GEN_EVENT_REG寄存器找到中断源。触发I2C_Intr中断时，I2C_INT位置1；触发I2C_bus_ctlr_intr中断时，I2C_BUF_CTLR_INT位置1。

触发中断处理程序时，主机驱动程序软件检查这两个位以识别中断源并相应地处理对应的中断。

当SMBUS_CTLR_ALERT_N引脚被拉为低电平且去抖后的输入已稳定时，也会产生SMBUS_CTLR_IRQ中断。此时，SMBUS_GEN_EVENT_REG.SMBALERT_INT位也置1。

可通过将SMBUS_GEN_EVENT_MASK_REG寄存器中的相应位置1来屏蔽中断。

SMBUS_CTLR_WAKE事件存在的意义是允许检测唤醒事件。

10.7 支持唤醒

SMBus控制器支持异步唤醒。SMBus控制器通过SMBUS_CTLR_WAKE事件指示发生了异步唤醒。无论SMBus控制器是否处于挂起状态，都可能发生SMBUS_CTLR_WAKE事件。

当PCIe端点功能处于D3状态且SMBus控制器未处于挂起状态时：

- I2C_Intr和I2C_buf_ctlr_intr都生成SMBUS_CTLR_WAKE事件。可通过查看SMBUS_GEN_EVENT_REG寄存器找到中断源。触发I2C_Intr中断时，I2C_WAKE_EVENT位置1；触发I2C_bus_ctlr_intr中断时，I2C_BUF_CTLR_WAKE_EVENT位置1。
- 当SMBUS_CTLR_ALERT_N引脚被拉为低电平且去抖后的输入已稳定时，也会产生SMBUS_CTLR_WAKE中断。此时，SMBUS_GEN_EVENT_REG.SMBALERT_WAKE_EVENT位置1。

当PCIe端点功能处于D3状态且SMBus控制器处于挂起状态时，SMBUS_CTLR_ALERT_N引脚具有去抖能力（可通过SMBALERT_CTLR_PAD_CTRL_REG.SMBALERT_CTLR_DB使能），持续时间由SMBALERT_CTLR_DB_REG.SMBUS_CTLR_DB_TIME[11:0]定义。这有助于防止噪声引起的错误唤醒，这是I^2C规范的要求。经过滤波后，当SMBUS_CTLR_ALERT_N引脚被拉为低电平时，触发SMBUS_CTLR_WAKE事件并通过PCIe端点发送WAKE#。当62.5 MHz振荡器和BIAS稳定后，I^2C由新的时钟源提供时钟，并且SMBUS_GEN_EVENT_REG.SMBALERT_WAKE_EVENT位置1。

SMBUS_CTLR_WAKE事件会导致生成PCI唤醒信号。该唤醒请求应触发PCIe主机将SMBus控制器置于D0器件状态。此时，聚合的外设PLL请求CLK_REQ_REG.PERI_WR_PLL_REQ和聚合的外设晶振请求CLK_REQ_REG.PERI_CRYSTAL_REQ都将被外设子系统设置为true，因为有一个或多个物理功能现在处于D0状态。

唤醒事件通过SMBUS_GEN_EVENT_REG寄存器中的位来指示，并且可通过将SMBUS_GEN_EVENT_MASK_REG寄存器中的相应位置1来屏蔽。

10.8 SMBus子系统寄存器

表104列出了PERI_SUBSYSTEM_ADDR_BASE = 0x0015_0000时可用的寄存器。

注：	PCI1xxxx器件包含数千个寄存器，其中一部分寄存器保留供将来使用，另一部分寄存器的内容未纳入公开文档，因为其不涉及终端系统集成商用例（即：这些寄存器供硬件用于执行底层运行时任务，或在运行时由驱动程序直接使用/控制）。**请勿修改未纳入文档的寄存器的内容。**

表104： SMBus子系统寄存器地址相对于PERI_SUBSYSTEM_ADDR_BASE的偏移量

寄存器名称	偏移地址
CONTROL_REG	0x0_0000
STATUS_REG	0x0_0004
DATA_REG	0x0_0008
SMBUS_CTLR_COMMAND_REG	0x0_000C
PEC_REG	0x0_0014
SR_HOLD_TIME_REG	0x0_0018
COMPLETION_REG	0x0_0020
IDLE_SCALING_REG	0x0_0024
CONFIG_REG	0x0_0028
BUS_CLOCK_REG	0x0_002C
BLOCK_ID_REG	0x0_0030
REVISION_REG	0x0_0037
BB_CONTROL_REG	0x0_0038
CLKSYNC_REGISTER	0x0_003C
DATA_TIMING_REG	0x0_0040
TO_SCALING_REG	0x0_0044
CTLR_TX_BUFFER_REG	0x0_0050
CTLR_RX_BUFFER_REG	0x0_0054
DEBUG_FSM_I2C	0x0_0058
DEBUG_FSM_SMB	0x0_005C
SCL_PAD_CTL	0x0_0100
SDA_PAD_CTL	0x0_0101
SMBUS_CONTROL	0x0_0200
SMBUS_STATUS	0x0_0204
SMBUS_INTERRUPT_STATUS	0x0_0208
SMBUS_INTERRUPT_MASK	0x0_020C
SMBUS_FIFO_DATA	0x0_0210
SMBUS_MCU_COUNTER	0x0_0214
SMBUS_DEV_COUNTER	0x0_0218
SMBUS_THRESHOLD_VALUE	0x0_021C
SMBUS_BUFF_DEPTH	0x0_0220
SMBALERT_CTLR_PAD_CTRL_REG	0x0_0230
SMBALERT_CTLR_DB_REG	0x0_0234
SMBALERT_CTLR_VAL_REG	0x0_0238
SMBUS_GEN_EVENT_REG	0x0_023C
SMBUS_GEN_EVENT_MASK_REG	0x0_0240
SMBus_PCI_CTRL_REG	0x0_0244
SMBUS_RESET_REG	0x0_0248
SMBUS_LTR_VALUE_REG	0x0_024C
SMBUS_DATA_BUFFER	0x0_0280
GPR0_REG	0x0_1C00

表105列出了控制寄存器的详细信息。

表105： CONTROL_REG

SMBus控制寄存器			默认值： 0x81
Bit	**名称**	**R/W**	**说明**
7	PIN	W/SC	非待处理中断（Pending Interrupt Not，PIN）位用作软件复位功能。向PIN位写入1b会将除STATUS_REG.nBB位（不受PIN位的影响）之外的所有状态位置为无效。PIN位是自清零位。向该位写入0b不起作用。 **注：**在SMBus目标接收模式下，当SMBus目标检测到停止条件时，I^2C控制器内核中的PIN位会置为有效。在I^2C向后兼容模式下，如果SMBus目标主机在另一个有效的SMBus目标地址到达之前尚未将PIN位置为无效，则SMBus目标将以NACK响应SMBus目标地址，因为SMBus目标主机尚未处理原始中断。
6	ESO	W	使能串行输出（Enable Serial Output，ESO）位用于使能和禁止SMBus控制器内核串行数据输出（SDAT）。当ESO置为有效（1b'1）时，使能 SDAT。当 ESO 未置为有效（1b'0）时，禁止SDAT。ESO位不影响对寄存器接口中其他位的访问。 **注：**当SMBus控制器内核主动参与总线事务时，不得将ESO位置为无效。
5:4	Reserved	W	始终为0b
3	ENI	W	允许中断（ Enable Interrupt，ENI）位用于控制中断接口。
2	STA	W	STA和STO位用于控制生成I^2C启动条件、发送SMBus目标地址和R/W位（来自DATA_REG寄存器）、生成重复启动条件，以及生成停止条件。 置1时，发送启动条件+地址，如果数据寄存器bit 0（R/nW）= 1b'0，则保持CTRL/TRM。 **注：**STA与STO不得同时置1。
1	STO	W	置1时，发送停止条件转到TGT/REC模式。 **注：**STA与STO不得同时置1。
0	ACK	W	应答位（ACK）通常必须置为有效（1b）。这将导致控制器在每个字节后自动发送一个应答（在第 9 个时钟脉冲期间发生）。当控制器在SMBus控制器/接收器模式下工作并且不需要从SMBus目标发送器发送更多数据时，ACK位不得置为有效（0b）。这将导致I^2C总线上出现否定应答，从而停止SMBus目标器件的进一步发送操作。

表106列出了状态寄存器的详细信息。

表106： STATUS_REG

SMBus状态寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7	PIN	R	待处理中断位。
6	Masked	R	功能已屏蔽或未使用。
5	Masked	R	功能已屏蔽或未使用。
4	BER	R	总线错误（Bus Error，BER）位置为有效时，表示已检测到错误的启动/停止条件或已检测到总线超时。BER将nBB置为无效（1b'1）并立即将PIN位置为有效（1b'0）。当BER置为有效时，必须先使用配置寄存器的RESET位将SMBus控制器内核复位才能执行后续操作。
3	LRB_AD0	R	“最后一个接收位”或“地址0”（广播呼叫）位（LRB_AD0）具备双重功能，仅在PIN位置为有效（0b）时有效。 当AAS位未置为有效（0n）（即，未作为SMBus目标寻址）时，LRB_AD0保持通过总线接收的最后一个位的值。通常，该值将是SMBus目标应答的值；因此，通过测试LRB_AD0位来检查SMBus目标应答。
2	Masked	R	功能已屏蔽或未使用。
1	Masked	R	功能已屏蔽或未使用。
0	nRB	R	总线繁忙位（nBB）为只读标志，用于指示总线何时处于使用中。零表示总线繁忙状态，无法访问。该位由启动条件置为有效（0b），在停止条件后置为无效（1b）。此外，nBB位也会因总线超时而置为无效。

表107列出了数据寄存器的详细信息。

表107： DATA_REG

数据寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:0	Data[7:0]	R/W	数据

表108列出了控制器命令寄存器的详细信息。

表108： SMBUS_CTLR_COMMAND_REG

SMBus控制器命令寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:24	READCOUNT[7:0]	R/W	该位域用于对从SMBus读取到CTLR_RX_BUFFER_REG寄存器的字节进行计数，其值必须大于0，SMBus控制器状态机才能启动读阶段。每次从SMBus向CTLR_RX_BUFFER_REG寄存器读入一个字节，该位域都递减1。可以用从SMBus读入的第一个字节改写该位域。 **注：**读取计数是块读取的硬限制。超出该值的任何已接收数据都将被忽略。 ReadCount[7:0]不包括PEC字节，因此从SMBus目标读取的字节数为（ReadCount[7:0] + Read_PEC）。复制到存储器的字节数为（ReadCount[7:0] + 2XRead_PEC）。
23:16	WRITECOUNT[7:0]	R/W	该位域用于对从CTLR_TX_BUFFER_REG寄存器发送到SMBus的字节进行计数。每次从CTLR_TX_BUFFER_REG寄存器向SMBus写入一个字节，该位域都递减1。
15:14	Reserved	R	始终读为0
13	READ_PEC	R/W	如果该位为0b，当READCOUNT[7:0]达到0x00时，停止读取SMBus。如果该位为1b，当READCOUNT[7:0]为0x00时，继续再读取一个字节。这样SMBus控制器会在读取一个M字节的块后再读取从外部器件返回的PEC。读取PEC字节后，该位将清除为0b。
12	READC	R/W	如果该位为1b'1，则当ReadCount[7:0]为8h'01时，从SMBus读取的字节将替换ReadCount[7:0]位域。ReadCount[7:0]更新后，该位将清除为1b'0。
11	PEC_TERM	R/W	如果该位为1b'1，则当WriteCount[7:0]为1b'0时，将发送PEC_REG寄存器的副本。读取PEC_REG寄存器后，PEC_REG寄存器和PEC_TERM位都将清除为1b'0。
10	STOP	R/W	如果该位为1b'1，则在事务完成后发送停止位。
9	STARTN	R/W	如果该位为1b'1，则在WriteCount[7:0]的最后一个字节发送到SMBus发送器之前发送起始位。
8	START0	R/W	如果该位为1b'1，则在WriteCount[7:0]的第一个字节发送到SMBus发送器之前在SMBus上发送起始位。
7:2	Reserved	R	始终读为0b
1	CPROCEED	R/W	当该位为0b时，SMBus控制器状态机不会退出空闲或暂停状态。 当该位为1b时，SMBus控制器状态机立即切换到WAIT-BusBusy或CRUN-Receive状态。当SMBus控制器状态机进入暂停状态时，该位自动清零。

表108： SMBUS_CTLR_COMMAND_REG（续）

SMBus控制器命令寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
0	CRUN	R/W	当该位为1b时，通过SMBus传输字节。只要WRITECOUNT[7:0]非零，SMBus控制器发送缓冲区中的字节就会发送到SMBus目标器件，同时WRITECOUNT[7:0]递减。如果WRITECOUNT[7:0]为0x00，控制器会将COMPLETION_REG.CDONE置为有效，清零CPROCEED，并等待软件重启状态机。 当通过写入CPROCEED重启时，从SMBus目标器件接收的每个字节都会写入SMBus控制器接收缓冲区。在写阶段和读阶段，SMBus控制器都会驱动SMBus时钟。接收阶段完成后，状态机将COMPLETION_REG.CDONE置为有效。在以下情况下，该位清零且事务完成： • Read_PEC为0b且READCOUNT[7:0]和WRITECOUNT[7:0]都为0x00 • Read_PEC为1b、READCOUNT[7:0]为0x00且WRITECOUNT[7:0]为0xFF • 从SMBus目标器件收到NACK • SMBus控制器总线仲裁失败 • SMBus控制器超时或发生其他总线错误 当SMBus控制器状态机将WRITECOUNT[7:0]计数至8h'00且READCOUNT[7:0]大于8h'00时，该位不会清零。此时将触发COMPLETION_REG.CDONE中断，固件可通过向该寄存器中的CPROCEED位写入1b来重启SMBus控制器状态机。 若要退出空闲状态，软件必须将CRUN和CPROCEED都设置为1b。

表109列出了数据包错误寄存器的详细信息。

表109： PEC_REG

数据包错误寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
7:0	PEC[7:0]	R/W	如果使用PEC功能，则包含PEC值。

表110列出了保持时间寄存器的详细信息。

表110： SR_HOLD_TIME_REG

保持时间寄存器			默认值：0x85
Bit	**名称**	**R/W**	**说明**
7:0	SR_HOLD_TIME[7:0]	R/W	该位域用于控制时钟的保持时间，直到满足重复启动位的保持时间为止。

表111列出了完成寄存器的详细信息。

表111： COMPLETION_REG

完成寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	Masked	R	请勿修改
30	CDONE	R/W1C	如果该位为1b，则表示SMBus控制器状态机已完成操作并返回到空闲状态。写入1b时，该位清零。写入0b不起作用。
29	IDLE	R/W1C	当I^2C总线变为空闲状态（在STATUS_REG.nBB的上升沿）时，该位置1。如果允许空闲中断（在ENIDI中断的上升沿）或使能I^2C内核时总线处于空闲状态，则该位也会置1。写入1b时，该位清零。写入0b不起作用。
28:26	Reserved	R	始终读为0
25	CTR	R	SMBus控制器发送/接收。该位用于报告SMBus控制器状态机将CDONE置为有效时所处的阶段。 0b：SMBus控制器刚刚完成事务的接收阶段。 1b：SMBus控制器刚刚完成事务的发送阶段。
24	CNAKX	R/WC	如果该位为1b，则表示SMBus控制器状态机在通过SMBus接口发送数据时从接收SMBus目标收到NACK。
23:22	Reserved	R	始终读为0
21:19	Masked	R	请勿修改
18	Reserved	R	始终读为0
17:16	Masked	R	请勿修改
15	Reserved	R	始终读为0
14:13	Masked	R	请勿修改
12	CHDH	R/WC	CHDH为总线空闲超时检测位（时钟高电平数据高电平）。
11	CHDL	R/WC	CHDL为时钟高电平超时检测位（时钟高电平数据低电平）。
10	Masked	R	请勿修改
9	CCTO	R/WC	CCTO为SMBus控制器累积超时位。
8	DTO	R/WC	DTO为器件超时位。总线空闲周期应编程为TIDLE_WINDOW时间。该位用于定义满足公平协议所需的波特率时钟周期数。该位的默认值会将空闲窗口设置为31 µs，适用于100 kHz总线。
7	Reserved	R	始终读为0

表111： COMPLETION_REG（续）

完成寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
6	TIMERR	R	只要有已使能的超时错误检测状态位（CHDH、CHDL、TCTO、CCTO和DTO）置为有效，检测到超时错误位（TIMERR）就会置为有效（1b）。 **注：**超时错误检测状态位使能信号包括BIDEN、CCEN 和DTEN。当所有已使能的超时错误检测状态位都未置为有效时，TIMERR不会置为有效。如果所有超时错误检测状态位使能信号都为0b，则TIMERR无法置为有效。
5	BIDEN	R	BIDEN位用于使能总线空闲检测超时检查。当BIDEN置为有效（1b）时，使能总线空闲检测超时检查。当BIDEN未置为有效（0b）时，禁止总线空闲检测超时检查。
4	Masked	R	请勿修改
3	CCEN	R/W	CCEN位用于使能SMBus控制器累积超时检查。当CCEN置为有效（1b）时，使能SMBus控制器累积超时检查。当CCEN未置为有效（0b）时，禁止SMBus控制器累积超时检查。
2	DTEN	R/W	DTEN位用于使能器件超时检查。当DTEN置为有效（1b）时，使能器件超时检查。当DTEN未置为有效（0b）时，禁止器件超时检查。
1:0	Reserved	R	始终为0

表112列出了空闲调节寄存器的详细信息。

表112： IDLE_SCALING_REG

空闲调节寄存器			默认值：0x03E8_03C9
Bit	**名称**	**R/W**	**说明**
31:28	Reserved	R	始终读为0
27:16	FAIR_IDLE_DELAY[11:0]	R/W	该位域用于建立MCTP T IDLE_DELAY周期。使能MCTP公平协议时，该位域会在状态机检测到一个空闲周期（FAIRBUS IDLE MIN）后，使SMBus控制器事务的启动时间额外延长一个周期（TIDLE_DELAY）。该位域用于定义编程延时所需的波特率时钟周期数。
15:13	Reserved	R	始终读为0
11:0	FAIR_BUS_IDLE[11:0]	R/W	总线空闲周期应编程为TIDLE_WINDOW时间。该位域用于定义满足公平协议所需的波特率时钟周期数。

表113列出了配置寄存器的详细信息。

表113： CONFIG_REG

SMBus配置寄存器			默认值： 0x0000_2000
Bit	**名称**	**R/W**	**说明**
31	ENSI	R/W	如果该位为1，允许SMBus目标完成中断。如果该位为0，禁止SMBus目标完成中断。
30	ENMI	R/W	如果该位为1，允许SMBus控制器完成中断。如果该位为0，禁止SMBus控制器完成中断。
29	ENIDI	R/W	如果该位为1，允许空闲中断。如果该位为0，禁止空闲中断。 **注：**如果SMBus控制器状态机正在运行，不得允许空闲中断（SMBUS_CTLR_COMMAND_REG寄存器中的CRUN位设置为1）。
28	Masked	R	请勿修改
27:20	Reserved	R	始终为0
19	FLUSH_MRBUF	W	向该位写入1会强制将CTLR_RX_BUFFER_REG寄存器标记为空。写入0不起作用。该位自清零。
18	FLUSH_MXBUF	W	向该位写入1会强制将CTLR_TX_BUFFER_REG寄存器标记为空。写入0不起作用。该位自清零。
17	FLUSH_SRBUF	W	向该位写入1会强制将CTLR_RX_BUFFER_REG寄存器标记为空。写入0不起作用。该位自清零。
16	FLUSH_SXBUF	W	向该位写入1会强制将CTLR_TX_BUFFER_REG寄存器标记为空。写入0不起作用。该位自清零。
15:11	Masked	R	请勿修改
10	ENAB	R/W	当ENAB（使能）未置为有效（0b）（默认值）时，SMBus控制器内核被禁止并处于最低功耗状态（已禁止状态）。当ENAB未置为有效时，所有寄存器都可以正常访问。 为确保正常工作，必须将ENAB位置为有效（1b）。 当ENAB位置为有效时，不得将位拆裂控制寄存器中的BBEN位置为有效。 **注：**主机负责确保SMBus控制器在ENAB位置为无效之前不会被占用。

表113： CONFIG_REG（续）

SMBus配置寄存器			默认值： 0x0000_2000
Bit	**名称**	**R/W**	**说明**
9	RESET	R/W	当RESET置为有效（1b'1）时，RESET位自身以外的所有逻辑和寄存器都将初始化为上电默认状态。 RESET不会自清零，当输入主机时钟小于或等于I^2C波特率时钟周期时，可由主机在一个输入主机时钟周期后置为无效，否则会在一个I^2C波特率时钟周期后置为无效以恢复正常工作。 RESET置为有效时进行的寄存器读操作将返回默认寄存器值。
8	FEN	R/W	当输入滤波使能位（FEN）置为有效（1b）时，使能输入滤波器。当FEN未置为有效（0b）（默认值）时，旁路输入滤波器。 对于未在硬件接口上配置外部滤波的应用，需将FEN置为有效。
7	PECEN	R/W	当PEC使能位（PECEN）置为有效（1b）时，使能硬件PEC支持。
6	SS_DET_CLK_SEL	R	启动/停止检测时钟选择位： 0b：使用系统时钟进行检测。 1b：使用内核时钟进行检测。
5	CONFIG_SLOW_CLK	R	当该位为1b时，总线时钟寄存器的基本周期乘以4，因此频率除以4。这不会影响其他时序计算（例如数据时序寄存器或超时调节寄存器中的计算）。
4	TCEN	R/W	当时序校验使能位（TCEN）置为有效（1b）时，使能总线超时。
3:0	PORT_SEL[3:0]	R/W	PORT_SEL[3:0]位用于确定16个可能的总线端口中的哪一个应用于活动的双线SDAT和SCLK总线对。 例如，当PORT_SEL[3:0]位为4b'0000（默认值）时，SDAT和SCLK报文信号出现在SDAT_IN [0]、SDAT_OUT [0]、SDAT_EN_N [0]、SCLK_IN [0]、SCLK_OUT [0]和SCLK_EN_N [0]上。

表114列出了总线时钟寄存器的详细信息。

表114： BUS_CLOCK_REG

总线时钟寄存器			默认值：0x9A9C
Bit	**名称**	**R/W**	**说明**
15:8	HIGH_PERIOD[7:0]	R/W	该位域用于定义I²C/SMBus总线时钟的高电平阶段所包含的I²C波特率时钟周期数。时钟周期数比该位域的值大1。 例如，将该位域设置为0x27表示一个持续0x28个波特率时钟周期的阶段。 在I²C快速模式（400 kHz操作）下，HIGH_PERIOD[7:0]应定义一个至少0.6 µs的周期。 在I²C增强型快速模式（1 MHz操作）中，HIGH_PERIOD[7:0]应定义一个至少0.26 µs的周期。
7:0	LOW_PERIOD[7:0]	R/W	该位域用于定义I²C/SMBus总线时钟的低电平阶段所包含的I²C波特率时钟周期数。时钟周期数比该位域的值大1。 例如，将该位域设置为0x27表示一个持续0x28个波特率时钟周期的阶段。 **注：**在I²C快速模式（400 kHz操作）中，LOW_PERIOD[7:0]应定义一个至少1.3 µs的周期。 在I²C增强型快速模式（1 MHz操作）中，LOW_PERIOD[7:0]应定义一个至少0.5 µs的周期。

表115列出了模块ID的详细信息。

表115： BLOCK_ID_REG

SMBus模块ID			默认值：0x11
Bit	**名称**	**R/W**	**说明**
7:0	ID[7:0]	R/W	模块ID

表116列出了模块版本的详细信息。

表116： REVISION_REG

SMBus模块版本			默认值：0xXX
Bit	**名称**	**R/W**	**说明**
7:0	REVISION[7:0]	R/W	版本

表117列出了位拆裂控制寄存器的详细信息。

表117： BB_CONTROL_REG

位拆裂控制寄存器			默认值：0x60
Bit	**名称**	**R/W**	**说明**
7	Reserved	R	始终为0b。
6	BBDATI	R	位拆裂数据输入。BBDATI位始终返回SDAT的状态。
5	BBCLKI	R	位拆裂时钟输入。BBCLKI位始终返回SCLK的状态。
4	BBDAT	R/W	位拆裂数据。当BBEN = 1且DADIR = 1b'1时，BBDAT位控制SDAT的状态。
3	BBCLK	R/W	位拆裂时钟。当BBEN = 1且CLDIR = 1b'1时，BBCLK位控制SCLK的状态。
2	DADIR	R/W	位拆裂数据方向。DADIR位用于控制SDAT的方向。 0b：输入 1b：输出
1	CLDIR	R/W	位拆裂时钟方向。CLDIR位用于控制SCLK的方向。 0b：输入 1b：输出
0	BBEN	R/W	位拆裂模式使能。BBEN位用于使能和禁止位拆裂模式： 0b：禁止位拆裂模式 1b：使能位拆裂模式

表118列出了时钟同步寄存器的详细信息。

表118： CLKSYNC_REGISTER

时钟同步寄存器			默认值：0x0000_0004
Bit	**名称**	**R/W**	**说明**
31:0	CLKSYNC[31:0]	R/W	CLKSYNC[31:0]用于在开始比较内外时钟以进行时钟延长之前，提供n个时钟周期与外部时钟同步。

表119列出了数据时序寄存器的详细信息。

表119： DATA_TIMING_REG

数据时序寄存器			默认值：0x169D_9D01
Bit	**名称**	**R/W**	**说明**
31:24	FIRST_START_HOLD[7:0]	R/W	STOP_SETUP[7:0]定时器用于确定在第一个启动位传输期间SDAT驱动为低电平后的SCLK保持时间。重复启动保持时间由SR_HOLD_TIME_REG寄存器确定。
23:16	STOP_SETUP[7:0]	R/W	STOP_SETUP[7:0]定时器用于确定停止条件下自SCLK上升沿起的SDAT建立时间。

表119： DATA_TIMING_REG（续）

数据时序寄存器			默认值：0x169D_9D01
Bit	**名称**	**R/W**	**说明**
15:8	RESTART_SETUP[7:0]	R/W	RESTART_SETUP[7:0] 定时器用于确定重复启动条件下自SCLK上升沿起的SDAT建立时间。
7:0	DATA_HOLD[7:0]	R/W	DATA_HOLD[7:0] 定时器用于确定SCLK驱动为低电平后的SDAT保持时间。

表120列出了超时调节寄存器的详细信息。

表120： TO_SCALING_REG

超时调节寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:24	BUS_IDLE_MIN[7:0]	R/W	总线空闲最短时间。 在100 kHz I^2C总线速率下，总线空闲最短时间为4.7 µs或更长。
23:16	CTLR_CUM_TIME-OUT[7:0]	R/W	SMBus控制器累积超时持续时间 = CTLR_CUM_TIME-OUT[7:0] x Baud_Clock_Period x 2048。 在100 kHz I^2C总线速率下，SMBus控制器累积超时持续时间为10 ms（最大值）。
15:8	TARGET_CUM_TIME-OUT[7:0]	R/W	SMBus目标累积超时持续时间 = TARGET_CUM_TIME-OUT[7:0] x Baud_Clock_Period x 4096 在100 kHz I^2C总线速率下，SMBus目标累积超时持续时间为25 ms（最大值）。
7:0	CLOCK_HIGH_TIME-OUT[7:0]	R/W	时钟高电平超时周期 = CLOCK_HIGH_TIME-OUT[7:0] x Baud_Clock_Period x 8 在100 kHz I^2C总线速率下，时钟高电平超时周期为50 µs（最大值）。

表121列出了控制器发送缓冲区寄存器的详细信息。

表121： CTLR_TX_BUFFER_REG

SMBus控制器发送缓冲区寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:0	CTLR_TRANSMIT_BUFFER[7:0]	R/W	SMBus控制器发送缓冲区

表122列出了控制器接收缓冲区寄存器的详细信息。

表122： CTLR_RX_BUFFER_REG

SMBus控制器接收缓冲区寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:0	CTLR_RECEIVE_BUFFER[7:0]	R/W	SMBus控制器接收缓冲区

表123列出了调试I²C状态机寄存器的详细信息。

表123： DEBUG_FSM_I²C

调试I²C状态机寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:28	28 TIMER_MBI[3:0]	R	0000b：MBI_IDLE 0001b：MBI_COUNT 0010b-1111b：保留。这些值的操作未定义。
27:24	TIMER_SCTO[3:0]	R	0000b：SC_IDLE 0001b：SC_COUNT 0010b-1111b：保留。这些值的操作未定义
23:20	TIMER_MCTO[3:0]	R	定时器MCTO的状态： 0000b：MC_IDLE 0001b：MC_COUNT 0010b-1111b：保留。这些值的操作未定义。
19:16	PHY[3:0]	R	I²C PHY的状态： 0000b：IDLE 0001b：CLKHIGH 0010b：STARTSTOP 0011b：CLKLOW 0100b：SDATCTRL 0101b：ARBLOSS 0110b-1111b：保留。这些值的操作未定义。
15:8	Reserved	R	始终读为0
7:0	I2C_CTLR[7:0]	R	I²C控制器的状态： 0000b：IDLE 0001b：W4 START 0010b：ADDR PHASE 0011b：CHK ACK 0100b：RX DATA 0101b：ACK NACK DATA 0110b：TX DATA 0111b：TX DATA LD 1000b：WAIT ACK 1001b：W4 STOP 1010b：LOST ARB 1011b：LOST ARB REPSTART 1100b：LOST ARB REPSTART DLY1 1101b：LOST ARB REPSTART DLY2 1110b：W4 START HOLD 1111b：保留。该值的操作未定义。

表124列出了调试SMBus状态机寄存器的详细信息。

表124： DEBUG_FSM_SMB

调试SMBus状态机寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:24	Reserved	R	始终读为0
23:16	SMB_FAIR[7:0]	R	SMB公平协议的状态： 0000b：IDLE 0001b：BUSY 0010b：WINDOW 0011b：DELAY 0100b：WAIT 0101b：WAIT DONE 0110b：ACTIVE 0111b-1111b：保留。这些值的操作未定义。
15:8	Reserved	R	始终读为0
7:0	SMB_CTLR[7:0]	R	SMBus控制器的状态： 0000b：IDLE 0001b：SOP 0010b：START 0011b：START PIN 0100b：WDATA 0101b：WPEC 0110b：RSTART 0111b：RSTART PIN 1000b：RDATA N 1001b：RDATA PEC 1010b：RPEC 1011b：PAUSE 1100b：STOP 1101b：EOP 1110b-1111b：保留。这些值的操作未定义。

表125列出了时钟焊盘控制寄存器的详细信息。

表125： SCL_PAD_CTL

SMBus时钟焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:5	Reserved	R	始终读为0
4	FAST_OPEN_DRAIN	R/W	该位用于设置快速漏极开路操作。 0b：输出驱动为低电平 1b：从0跳变到1时，输出在进入三态之前驱动为高电平并持续一个系统时钟周期。
3	PULLUP_EN	R/W	设置为1b时使能上拉电阻。
2	PULLDOWN_EN	R/W	设置为1b时使能下拉电阻。
1	INPUT_EN	R/W	设置为1b时使能焊盘的输入缓冲区。 当INPUT_EN清除为0b时，输入缓冲区的输出默认为1b。
0	OUTPUT_EN	R/W	设置为1b时使能焊盘的输出缓冲区。

表126列出了数据焊盘控制寄存器的详细信息。

表126： SDA_PAD_CTL

SMBus数据焊盘控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:5	Reserved	R	始终读为0
4	FAST_OPEN_DRAIN	R/W	该位用于设置快速漏极开路操作。 0b：输出驱动为低电平 1b：从0跳变到1时，输出在进入三态之前驱动为高电平并持续一个系统时钟周期。
3	PULLUP_EN	R/W	设置为1b时使能上拉电阻。
2	PULLDOWN_EN	R/W	设置为1b时使能下拉电阻。
1	INPUT_EN	R/W	设置为1b时使能焊盘的输入缓冲区。 当INPUT_EN清除为0b时，输入缓冲区的输出默认为1b。
0	OUTPUT_EN	R/W	设置为1b时使能焊盘的输出缓冲区。

表127列出了SMBus控制寄存器的详细信息。

表127： SMBUS_CONTROL

SMBus控制寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:4	Reserved	R	始终读为0
3	RESET_COUNTERS	R/W/SC	将该位置1会将SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]和SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]复位为8'h00。该位在置1状态下持续一个时钟周期，之后自动清零。 清零该位不起作用。
2	TRANSFER_DIR	R/W	0b：数据从MCU传输到器件。这意味着MCU正在填充缓冲区，I^2C控制器正在消耗缓冲区。 1b：数据从器件传输到MCU。这意味着I^2C控制器正在填充缓冲区，MCU正在消耗缓冲区。
1	HOST_FIFO_ENTRY	R/W	0b：MCU可直接访问缓冲区。 1b：MCU可通过SMBUS_FIFO_DATA.FIFO_DATA[7:0]访问缓冲区。
0	RUN	R/W	将该位置1会使能DMA仿真逻辑，I^2C控制器将根据TRANSFER_DIR设置填充或消耗缓冲区。只有软件可将该位置1。 硬件和软件都可将该位清零。当SMBUS_STATUS.DMA_TERM为1b时，硬件将该位清零。如果在发送操作中SMBUS_STATUS.BUF_EMPTY设置为1b或者在接收操作中SMBUS_STATUS.BUF_FULL设置为1b，硬件也会将运行控制寄存器位清零。 当软件清零该位时，无论是否正在进行I^2C传输，都将中止DMA仿真逻辑。

表128列出了SMBus状态寄存器的详细信息。

表128： SMBUS_STATUS

SMBus状态寄存器			默认值： 0x00
Bit	**名称**	**R/W**	**说明**
7	DMA_TERM	R	DMA_TERM位直接与来自I²C/SMBus控制器的DMA_TERM信号相连。 如果DMA_TERM位设置为1b，仅在SMBUS_INTERRUPT_MASK.INT_MASK_DMA_TERM设置为0b时才能生成SMBUS_CTLR_IRQ事件。
6	DMA_REQ	R	寄存器位直接与来自I²C/SMBus控制器的SMB_MDMA_REQ信号相连。
5:3	Reserved	R	始终读为0
2	THRESHOLD_HIT	R	该寄存器位在SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]大于或等于 SMBUS_THRESHOLD_VALUE.THRESHOLD[7:0]时置1，在所有其他情况下均清零。 当SMBUS_INTERRUPT_MASK.INT_MASK_THRESHOLD 为0b时，该位可将SMBUS_CTLR_IRQ信号置为有效。
1	BUF_FULL	R	如果填充缓冲区时SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]或SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]大于或等于SMBUS_BUF_DEPTH.BUF_DEPTH[7:0]，该寄存器位置1，在所有其他情况下均清零。 TRANSFER_DIR用于确定跟踪填充操作的计数器。 当SMBUS_INTERRUPT_MASK.INT_MASK_BUF_FULL 为1b'0时，该位将SMBUS_CTLR_IRQ信号置为有效。
0	BUF_EMPTY	R	该寄存器位仅在SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]等于SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]且两个计数器的值均非零时置1，在所有其他情况下均清零。 当SMBUS_INTERRUPT_MASK.INT_MASK_BUF_EMPTY为1b'0时，该位可将SMBUS_CTLR_IRQ信号置为有效。

表129列出了SMBus中断状态寄存器的详细信息。

表129： SMBUS_INTERRUPT_STATUS

SMBus中断状态寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7	INT_STAT_DMA_TERM	R/W1C	DMA_TERM状态位中断。DMA_TERM状态位从0跳变到1会将该位设置为1b。软件只能通过写入1b来清零该位。
6:3	Reserved	R	始终读为0
2	INT_STAT_THRESHOLD	R/W1C	SMBUS_STATUS.THRESHOLD_HIT状态位中断。SMBUS_STATUS.THRESHOLD_HIT状态位从0跳变到1会将该位设置为1b'1。软件只能通过写入1b来清零该位。
1	INT_STAT_BUF_FULL	R/W1C	BUF_FULL状态位中断。BUF_FULL状态位从0跳变到1会将该位设置为1b。软件只能通过写入1b来清零该位。
0	INT_STAT_BUF_EMPTY	R/W1C	BUF_EMPTY状态位中断。BUF_EMPTY状态位从0跳变到1会将该位设置为1b。软件只能通过写入1b来清零该位。

表130列出了SMBus中断屏蔽寄存器的详细信息。

表130： SMBUS_INTERRUPT_MASK

SMBus中断屏蔽寄存器			默认值：0x87
Bit	**名称**	**R/W**	**说明**
7	INT_MASK_DMA_TERM	R/W	值为1b'1时，禁止在SMBUS_STATUS.DMA_TERM状态位 = 1b'1时产生中断。
6:3	Reserved	R	始终读为0
2	INT_MASK_THRESHOLD	R/W	值为1b时，禁止在SMBUS_STATUS.THRESHOLD_HIT状态位 = 1b时产生中断。
1	INT_MASK_BUF_FULL	R/W	值为1b时，禁止在SMBUS_STATUS.BUF_FULL状态位 = 1b时产生中断。
0	INT_MASK_BUF_EMPTY	R/W	值为1b时，禁止在SMBUS_STATUS.BUF_EMPTY状态位 = 1b时产生中断。

表131列出了SMBus FIFO数据寄存器的详细信息。

表131： SMBUS_FIFO_DATA

SMBus FIFO数据寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:0	FIFO_DATA[7:0]	R/W	写入FIFO_DATA[7:0]时会将数据传递给SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]所指向的缓冲区数组，之后SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]立即递增，但SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]达到最大缓冲区深度时除外。 **注：**读取FIFO_DATA[7:0]时会读取SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]所指向的缓冲区数组，之后SMBUS_MCU_COUNTER.MCU_COUNTER[7:0]立即递增，但 SMBUS_MCU_COUNTER.MCU_COUNTER[7:0] 达到最大缓冲区深度时除外。

表132列出了SMBus单片机计数器寄存器的详细信息。

表132： SMBUS_MCU_COUNTER

SMBus单片机计数器寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:0	MCU_COUNTER[7:0]	R/W	该寄存器用作缓冲区数组的地址。 软件可随时读写该寄存器。但是，建议软件仅在SMBUS_CONTROL.RUN位清除为1b'0（即，禁止DMA仿真逻辑）时写入该寄存器，否则DMA操作将受到影响。 每当写入或读取SMBUS_FIFO_DATA.FIFO_DATA[7:0]时，硬件都会递增该寄存器值，直至达到SMBUS_BUF_DEPTH.BUF_DEPTH[7:0]为止。 **注：**软件也可以复位MCU_COUNTER[7:0]。

表133列出了SMBus器件计数器寄存器的详细信息。

表133： SMBUS_DEV_COUNTER

SMBus器件计数器寄存器			默认值：0x00
Bit	**名称**	**R/W**	**说明**
7:0	DEV_COUNTER[7:0]	R/W	该寄存器用作缓冲区数组的地址。 软件可随时读写该寄存器。但是，建议软件仅在SMBUS_CONTROL.RUN位清除为0b时写入该寄存器。 每当I^2C控制器请求对缓冲区数组进行读或写操作时，硬件都会递增该寄存器值，直至达到SMBUS_BUF_DEPTH.BUF_DEPTH[7:0]为止。 此外，软件也可通过向SMBUS_CONTROL.RESET_COUNTERS写入1b来将DEV_COUNTER[7:0]复位为8h'00。

表134列出了SMBus阈值寄存器的详细信息。

表134： SMBUS_THRESHOLD_VALUE

SMBus阈值寄存器			默认值：0x7F
Bit	**名称**	**R/W**	**说明**
7:0	THRESHOLD[7:0]	R/W	该寄存器用于保存阈值。 当SMBUS_DEV_COUNTER.DEV_COUNTER[7:0]大于或等于SMBUS_THRESHOLD_VALUE.THRESHOLD[7:0]时，SMBUS_STATUS.THRESHOLD_HIT将置为有效。

表135列出了SMBus缓冲区深度寄存器的详细信息。

表135： SMBUS_BUFF_DEPTH

SMBus缓冲区深度寄存器			默认值：0x80
Bit	**名称**	**R/W**	**说明**
7:0	BUF_DEPTH[7:0]	R/W	返回配置的缓冲区深度。应始终为128字节。

表136列出了SMBus报警焊盘控制寄存器的详细信息。

表136： SMBALERT_CTLR_PAD_CTRL_REG

SMBus报警焊盘控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:3	Reserved	R	始终为0b。

表136： SMBALERT_CTLR_PAD_CTRL_REG（续）

SMBus报警焊盘控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
2	SMBALERT_CTLR_DB	R/W	SMBUS_CTLR_ALERT_N焊盘的去抖使能。 0b——禁止去抖 1b——使能去抖 **注：**该引脚可配置为禁止去抖器。在这种情况下，直接传递原始输入。
1	SMBALERT_CTRL_PD	R/W	SMBUS_CTLR_ALERT_N焊盘的下拉使能。 0b——禁止下拉 1b——使能下拉
0	SMBALERT_CTRL_PU	R/W	SMBUS_CTLR_ALERT_N焊盘的上拉使能。 0b——禁止上拉 1b——使能上拉 **注：**SMBALERT#为低电平有效信号。

表137列出了SMBus报警去抖寄存器的详细信息。

表137： SMBALERT_CTLR_DB_REG

SMBus报警去抖寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:12	Reserved	R	始终为0b。
11:0	SMBUS_CTLR_DB_TIME[11:0]	R/W	该寄存器用于保存SMBUS_CTLR_ALERT_N焊盘的去抖定时器值。 当检测到信号跳变时，去抖器启动。如果引脚状态在配置的去抖时间内保持不变，则直接传递该跳变信号。如果在去抖时间内再次发生信号跳变，去抖器将重新启动。 每个计数对应1 ms，默认值为10 ms。由于计时基于自由运行时钟，因此会随边沿对齐情况而变化。计时精度限制为±一个计数。

表138列出了SMBus报警值寄存器的详细信息。

表138： SMBALERT_CTLR_VAL_REG

SMBus报警值寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:2	Reserved	R	始终为0b。
1	SMBALERT_CTLR_DB_VAL	R	包含SMBUS_CTLR_ALERT_N引脚的去抖值。 **注：**去抖后的值用于产生SMBALERT_INT中断。
0	SMBALERT_CTLR_VAL	R	包含SMBUS_CTLR_ALERT_N引脚的当前值。

表139列出了SMBus事件生成寄存器的详细信息。

表139： SMBUS_GEN_EVENT_REG

SMBus事件生成寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:11	Reserved	R	始终为0b。
10	SMBALERT	R/W	设为1b时，表示SMBUS_CTLR_ALERT_N引脚已被拉为低电平。 写入1b可清零该位。
9	I2C_BUF_CTLR_INT	R/W	设为1b时，表示SMBus网络引擎已触发I2C_buf_ctlr_intr。有关该中断的详细信息，请参见SMBUS_INTERRUPT_STATUS寄存器。 写入1b可清零该位。
8	I2C_INT	R/W	设为1b时，表示SMBus控制器内核已触发I2C_intr。有关该中断的详细信息，请参见STATUS_REG寄存器。 写入1b可清零该位。
7:3	Reserved	R	始终为0b。
2	SMBALERT_WAKE	R/W	设为1b时，表示SMBUS_CTLR_ALERT_N引脚已被拉为低电平。 写入1b可清零该位。
1	I2C_BUF_CTLR_WAKE	R/W	设为1b时，表示SMBus网络引擎已触发I2C_buf_ctlr_intr。 写入1b可清零该位。
0	I2C_WAKE	R/W	设为1b时，表示SMBus控制器内核已触发I2C_intr。 写入1b可清零该位。

表140列出了SMBus事件生成屏蔽寄存器的详细信息。

表140： SMBUS_GEN_EVENT_MASK_REG

SMBus事件生成屏蔽寄存器			默认值：0x0000_0707
Bit	**名称**	**R/W**	**说明**
31:11	Reserved	R	始终为0b。
10	SMBALERT_INT_MASK	R/W	SMBALERT_INT状态位屏蔽： 0b：在SMBALERT_INT状态位置1时生成SMBUS_CTLR_IRQ事件。 1b：在SMBALERT_INT状态位置1时禁止生成SMBUS_CTLR_IRQ事件。

表140： SMBUS_GEN_EVENT_MASK_REG（续）

SMBus事件生成屏蔽寄存器			默认值：0x0000_0707
9	I2C_BUF_CTLR_INT_MASK	R/W	I2C_BUF_CTLR_INT状态位屏蔽： 0b：在I2C_BUF_CTLR_INT状态位置1时生成SMBUS_CTLR_IRQ事件。 1b：在I2C_BUF_CTLR_INT状态位置1时禁止生成SMBUS_CTLR_IRQ事件。
8	I2C_INT_MASK	R/W	I2C_INT状态位屏蔽： 0b：在I2C_INT状态位置1时生成SMBUS_CTLR_IRQ事件。 1b：在I2C_INT状态位置1时禁止生成SMBUS_CTLR_IRQ事件。
7:3	Reserved	R	始终为0b。
2	SMBALERT_WAKE_MASK	R/W	SMBALERT_WAKE_EVENT状态位屏蔽： 0b：在SMBALERT_WAKE_EVENT状态位置1时生成SMBUS_CTLR_WAKE事件。 1b：在SMBALERT_WAKE_EVENT状态位置1时禁止生成SMBUS_CTLR_WAKE事件。
1	I2C_BUF_CTLR_WAKE_MASK	R/W	I2C_BUF_CTLR_WAKE_EVENT状态位屏蔽： 0b：在I2C_BUF_CTLR_WAKE_EVENT状态位置1时生成SMBUS_CTLR_WAKE事件。 1b：在I2C_BUF_CTLR_WAKE_EVENT状态位置1时禁止生成SMBUS_CTLR_WAKE事件。
0	I2C_WAKE_MASK	R/W	I2C_WAKE_EVENT状态位屏蔽： 0b：在I2C_WAKE_EVENT状态位置1时生成SMBUS_CTLR_WAKE事件。 1b：在I2C_WAKE_EVENT状态位置1时禁止生成SMBUS_CTLR_WAKE事件。

表141列出了SMBus PCI控制寄存器的详细信息。

表141： SMBUS_PCI_CTRL_REG

SMBus PCI控制寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:0	Reserved	R	始终为0b。

表141： SMBUS_PCI_CTRL_REG（续）

SMBus PCI控制寄存器			默认值： 0x0000_0000
0	SMB_D3_CLK_EN	R/W	在PCI功能处于D3状态时使能62.5 MHz时钟： 0b：在PCI功能处于D3状态时禁止62.5 MHz时钟。 1b：在PCI功能处于D3状态时使能62.5 MHz时钟。 外设PCIe®端点使用聚合的时钟请求位CLK_REQ_REG.PERI_CRYSTAL_REQ和CLK_REQ_REG.PERI_WR_PLL_REQ来指示其对晶振时钟和AB WR PLL的需求。当SMB_D3_CLK_EN为1b时，即使所有物理功能都处于D3状态，CLK_REQ_REG.PERI_CRYSTAL_REQ和CLK_REQ_REG.PERI_WR_PLL_REQ也不得设置为0b。

表142列出了SMBus复位寄存器的详细信息。

表142： SMBUS_RESET_REG

SMBus复位寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:17	Reserved	R	始终为0b。
16	PERI_SMBUS_D3_RESET_DIS	R/W	D3复位禁止 当退出D3Cold状态（PCIE_PERST_N置为无效）时，该位用于修改子系统复位操作。 当该位置1且VAUX_DET引脚为高电平时，仅复位PCIe®接口及相关逻辑；不复位SMBus控制器。 其他情况下，复位整个子系统。
15:10	Reserved	R	始终为0b。
9	PERI_SMBUS_RELOAD_PCI	R/W	定义PERI_UART_RELOAD置1时在外设SMBus PCIe端点PF1上执行的配置；对应PERI_SMBUS_PCIE_EP_ADDR_BASE。 设置为1时，外设SMBus PCIe端点PF1配置将重载到PERI_SMBUS_PCIE_EP_ADDR_BASE。 设置为0时，外设SMBus PCIe端点PF1配置不会重载到PERI_SMBUS_PCIE_EP_ADDR_BASE。系统配置硬件将尝试写入与特定配置相关的所有配置寄存器；子系统将阻止与PERI_SMBUS_PCIE_EP_ADDR_BASE处的PCIe端点对应的任何写操作。

表142： SMBUS_RESET_REG（续）

SMBus复位寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
8	PERI_SMBUS_RELOAD_FN	R/W	定义PERI_SMBUS_RELOAD置1时在外设SMBus功能IP上执行的配置；外设SMBus功能IP对应SMBUS_PERI_ADDR_BASE中的绝大多数子系统地址，只有对应PERI_SMBUS_PCIE_EP_ADDR_BASE的地址除外，这些地址由PERI_SMBUS_RELOAD_PCI位控制。 设置为1时，将重载外设SMBus功能IP配置。 设置为0时，不会重载外设SMBus功能IP配置。系统配置硬件将尝试写入与特定配置相关的所有配置寄存器；子系统将阻止与功能IP对应的任何写操作。
7:3	Reserved	R	始终为0b。
2	PERI_SMBUS_RELOAD	R/W/SC	启动外设SMBus功能重载 在PERI_SMBUS_RELOAD置1之前，应正确配置PERI_SMBUS_RELOAD_PCI和PERI_SMBUS_RELOAD_FN。 该位在重载完成后自清零。
1	PERI_SMBUS_LRST	R/W/SC	启动外设SMBus功能LRST。 该位在LRST完成后自清零。
0	PERI_SMBUS_SRST	R/W/SC	启动外设SMBus功能SRST。 该位在SRST完成后自清零。 将该位置1会在PCI链路断开时复位整个外设子系统。

表143列出了SMBus无侦听延时值寄存器的详细信息。

表143： SMBUS_LTR_VALUE_REG

SMBus无侦听延时值寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	SMBUS_NO_SNOOP_REQ	R/W	无侦听需求 置1时，表示有延时需求，具体取决于SMBUS_NO_SNOOP_LATENCY_VAL[9:0]和SMBUS_NO_SNOOP_LATENCY_SCALE[2:0]位域。 清零时，表示无延时需求，忽略SMBUS_NO_SNOOP_LATENCY_VAL[9:0]和SMBUS_NO_SNOOP_LATENCY_SCALE[2:0]位域。

表143： SMBUS_LTR_VALUE_REG（续）

SMBus无侦听延时值寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
30:29	Reserved	R	始终为0b。
28:26	SMBUS_NO_SNOOP_LATENCY_SCALE[2:0]	R/W	无侦听延时调节 SMBUS_NO_SNOOP_LATENCY_VAL[9:0]位域调节： 000b——值乘以1 ns 001b——值乘以32 ns 010b——值乘以1024 ns 011b——值乘以32768 ns 100b——值乘以1048576 ns 101b——值乘以33554432 ns 110b——不允许值执行乘法 111b——不允许值执行乘法
25:16	SMBUS_NO_SNOOP_LATENCY_VAL[9:0]	R/W	无侦听延时值 延时值位域。
15	SMBUS_SNOOP_REQ	R/W	侦听需要延时 置1时，表示有延时需求，具体取决于SMBUS_SNOOP_LATENCY_VAL[9:0]和SMBUS_SNOOP_LATENCY_SCALE[2:0]位域。 清零时，表示无延时需求，忽略SMBUS_SNOOP_LATENCY_VAL[9:0]和SMBUS_SNOOP_LATENCY_SCALE[2:0]位域。
14:13	Reserved	R	始终为0b。
12:10	SMBUS_SNOOP_LATENCY_SCALE[2:0]	R/W	侦听延时调节 SMBUS_SNOOP_LATENCY_VAL[9:0]位域调节： 000b——值乘以1 ns 001b——值乘以32 ns 010b——值乘以1024 ns 011b——值乘以32768 ns 100b——值乘以1048576 ns 101b——值乘以33554432 ns 110b——不允许值执行乘法 111b——不允许值执行乘法
9:0	SMBUS_SNOOP_LATENCY_VAL[9:0]	R/W	侦听延时值位域。

表144列出了SMBus数据缓冲区寄存器的详细信息。

表144： SMBUS_DATA_BUFFER

SMBus数据缓冲区寄存器			默认值： 0x0000_0000
Bit	**名称**	**R/W**	**说明**
31	SMBUS_BUFFER	R/W	SMBus数据缓冲区

表145列出了通用寄存器的详细信息。

表145： GPR0_REG

通用寄存器			默认值：0x0000_0000
Bit	**名称**	**R/W**	**说明**
31:2	Reserved	R	始终为0
1:0	SMBUS_SPEED_SEL	R/W	选择SMBus接口的速度。 00b：400 kHz“快速模式” 01b：100 kHz“标准模式” 10b：1 MHz“增强型快速模式” 11b：用户自定义——所有SMBus控制器时序都在PCI1XXXX配置中定义，I²C/SMBus驱动程序不会改写。 该寄存器为软件定义的寄存器。I²C/SMBus器件驱动程序读取该寄存器。如果读取值为00b、01b或10b，驱动程序会根据表103自动将所有寄存器设置设为所选值。如果读取值为11b，驱动程序不会修改接口的任何速度设置。

10.9 SMBus配置示例

10.9.1 使用A0硅片设置PCI11414 SMBus

1. 选择可编程引脚作为SMBus。

在可编程引脚复用表（表146）中，确定系统应使用哪些PF引脚来暴露SMBUS_CTLR_SCL、SMBUS_CTLR_SDA和SMBUS_CTLR_ALERT_N（如果需要报警输入功能）。

在SYSTEM_REG_ADDR_BASE.PPCTL_ADDR_BASE中，找到用于暴露SMBus功能引脚的PFx引脚对应的偏移量，然后将相应的功能选择写入PPx_CTL_REG.SELECT[3:0]并将驱动能力写入PPx_CTL_REG.DRIVE_STRENGTH[1:0]。

表146： 可编程引脚复用

	时钟	数据	报警
引脚	PF32	PF33	PF34
寄存器	0x0024_0480	0x0024_0484	0x0024_0488
值	0x0000_2006	0x0000_2003	0x0000_0001
备注	选择功能6（SMBUS_CTLR_SCL）并设置DRIVE_STRENGTH = 8 mA	选择功能6（SMBUS_CTLR_SDA）并设置DRIVE_STRENGTH = 8 mA	选择功能1（SMBUS_CTLR_ALERT_N）并设置DRIVE_STRENGTH = 2 mA（**注：**驱动能力不适用于输入引脚）

2. 通过编程GPR0_REG（地址：0x0015_1C00）选择SCL速度，以便驱动程序正确编程SMBus控制器时序寄存器。对于A0硅片，必须始终设置为0x0000_00003。
3. 根据所需的速度写入表103中的各个值。
4. 必须将DATA_TIMING_REG.DATA_HOLD[7:0]的值改写为0x02，以防止SMBus控制器逻辑仲裁失败，否则SMBus控制器无法正常工作。

11.0 外设子系统实现——SPI

注： 本章所述的详细信息对大多数终端系统集成商而言并非必需。之所以提供是因为这些信息可能会对以下场景有所帮助：调试、通过直接访问PCIe寄存器手动验证功能的有效性，以及在检查PCIe协议跟踪时与物理硬件层面的实际情况进行关联对照。

SPI控制器子系统有自己的PCI器件端点，需要主机系统提供独立的器件驱动程序才能正常工作。该子系统与所有其他外设（UART、SMBus和和GPIO）共用PCI总线。

子系统ID如下：

- 供应商ID：0x1055（Microchip Technology, Inc/SMSC）
- 器件ID：
  - PCI12000：0xA004
  - PCI11010：0xA014
  - PCI11101：0xA024
  - PCI11400：0xA034
  - PCI11414：0xA044
- 分类ID：0x0C8000（“其他接口”）

SPI子系统通过7个单独的“芯片使能”信号支持最多7个器件，并且支持通过62.5 MHz时钟生成多个波特率时钟频率。表147列出了支持的频率。

表147： 编程功能寄存器格式

近似频率	实际频率	62.5 MHz分频比值
30 MHz	31.25 MHz	2
20 MHz	20.83 MHz	3
15 MHz	15.63 MHz	4
12 MHz	12.50 MHz	5
10 MHz	10.42 MHz	6
2 MHz	2.08 MHz	7

注： 62.5 MHz时钟的分频系数必须大于1。不支持62.5 MHz SPI操作。

11.1 章节

11.2 在未安装SPI器件驱动程序的情况下获取访问权限

必须执行以下操作才能允许主机直接访问PCIe寄存器：

1. 确保正确设置SPI外设通用PCI配置空间的COMMAND位域。值为0x0000时将阻止所有对PCI寄存器空间的访问。未针对SPI外设重载驱动程序时，COMMAND位域保持为0x00，必须手动改写。推荐值为0x0406。
2. 针对SPI子系统设置相应的SYS_LOCK寄存器。

11.3 操作概述

SPI接口使用命令缓冲区和响应缓冲区，并通过一些控制寄存器来控制SPI访问。

- **SPI_CMD_BUF[319:0]**：320字节命令缓冲区。与320字节响应缓冲区相对应。

- **SPI_RESP_BUF[319:0]**：320字节响应缓冲区。通过长度寄存器对字节进行计数。
- **SPI_CMD_LEN**：按字节计数的长度寄存器。
- **SPI_CTL**：控制寄存器，包含自清零的GO位，该位用于启动SPI访问。当GO位置1后，SPI硬件接口会立即将所选的芯片使能引脚（SPI_CEN）置为有效并开始振荡时钟，以便发送SPI_CMD_BUF缓冲区内容并将收到的数据存入SPI_RESP_BUF。时钟将振荡SPI_CMD_LEN * 8个时钟周期。

11.3.1 任意长度SPI访问示例

注： 在任意长度 SPI 访问期间，软件使用 SPI_CTLR_CTL_REG.FORCE_CE 位将芯片使能引脚置为有效，并连续向发送缓冲区写入数据。发送操作结束时，软件必须清除SPI_CTLR_CTL_REG.FORCE_CE位以释放芯片使能引脚。

对于任意长度的SPI访问：

1. 软件将SPI_CTLR_CTL_REG.MODE_SEL设置为所需模式。
2. 软件将SPI_CTLR_CTL_REG.DEV_SEL[2:0]设置为所需器件。
3. 软件将SPI_CTLR_CTL_REG.CMD_LEN[7:0]设置为所需长度。
4. 软件将SPI_CTLR_CTL_REG.FORCE_CE位置1。这会强制将芯片使能引脚设为低电平。
5. 软件将输出字节写入SPI_CTLR_CMD_BUF.SPI_CMD_BUF[0:319]。
6. 如果需要中断，软件应清零SPI_CTLR_CTL_REG.SPI_INT_MASK。
7. 软件将SPI_CTLR_CTL_REG.GO位置1。
8. PCI1xxxx从SPI_CTLR_CMD_BUF.SPI_CMD_BUF[0:319]发送数据，并将响应接收到SPI_CTLR_RSP_BUF.SPI_RSP_BUF[0:319]中。
9. 该过程完成时，PCI1xxxx清零SPI_CTLR_CTL_REG.GO位。
10. 如果SPI_CTLR_CTL_REG.SPI_INT_MASK清零，PCI1xxxx会发出SPI_CTLR_IRQ信号。
11. 如果SPI_CTLR_CTL_REG.GO位清零时SPI_CTLR_CTL_REG.SPI_INT_MASK置1，软件从SPI_CTLR_RSP_BUF.SPI_RSP_BUF[0:319]读取响应。
12. 如果SPI_CTLR_CTL_REG.SPI_INT_MASK置1，软件等待SPI_CTLR_IRQ后从SPI_CTLR_RSP_BUF.SPI_RSP_BUF[0:319]读取响应。
13. 如果还有更多数据，转到步骤5。
14. 当所有数据发送完成后，软件清零SPI_CTLR_CTL_REG.FORCE_CE位；这将释放芯片使能引脚。

11.3.2 固定长度SPI访问示例

注： 在固定长度 SPI 访问期间，软件向 SPI_CTLR_CMD_BUF.SPI_CMD_BUF[0:319] 中写入一个数据块进行发送。当软件将SPI_CTLR_CTL_REG.GO位置1时，PCI1xxxx在整个发送过程中将芯片使能引脚置为有效。发送完成后，PCI1xxxx将释放芯片使能引脚（硬件不会将SPI_CTLR_CTL_REG.FORCE_CE位置1）。

对于固定长度的SPI访问：

1. 软件将SPI_CTLR_CTL_REG.MODE_SEL设置为所需模式。
2. 软件将SPI_CTLR_CTL_REG.DEV_SEL[2:0]设置为所需器件。
3. 软件将SPI_CTLR_CTL_REG.CMD_LEN[7:0]设置为所需长度。
4. 软件将输出字节写入SPI_CTLR_CMD_BUF.SPI_CMD_BUF[0:319]。
5. 如果需要中断，软件应清零SPI_CTLR_CTL_REG.SPI_INT_MASK。
6. 软件将SPI_CTLR_CTL_REG.GO位置1。
7. PCI1xxxx强制将芯片使能引脚设为低电平。
8. PCI1xxxx从SPI_CTLR_CMD_BUF.SPI_CMD_BUF[0:319]发送数据，并将响应接收到SPI_CTLR_RSP_BUF.SPI_RSP_BUF[0:319]中。
9. 该过程完成时，硬件强制将芯片使能引脚设为高电平并清零SPI_CTLR_CTL_REG.GO位。
10. 如果SPI_CTLR_CTL_REG.SPI_INT_MASK清零，PCI1xxxx会发出SPI_CTLR_IRQ信号。
11. 如果SPI_CTLR_CTL_REG.GO位清零时SPI_CTLR_CTL_REG.SPI_INT_MASK置1，软件从SPI_CTLR_RSP_BUF.SPI_RSP_BUF[0:319]读取响应。

12. 如果SPI_CTLR_CTL_REG.SPI_INT_MASK置1，软件等待SPI_CTLR_IRQ后从SPI_CTLR_RSP_BUF.SPI_RSP_BUF[0:319]读取响应。

11.4 挂起

当SPI_PCI_CTRL_REG.SPI_D3_CLK_EN位设置为0时，如果SPI控制器PCIe端点功能进入D3状态，将关闭提供给SPI控制器实例的62.5 MHz时钟。这被定义为SPI模块处于挂起状态。当SPI控制器PCI功能进入D0状态时，将恢复时钟。

当SPI_PCI_CTRL_REG.SPI_D3_CLK_EN位设置为1时，即使SPI控制器PCIe端点功能进入D3状态，也不会关闭提供给SPI控制器实例的62.5 MHz时钟；该模块将继续工作。

这也意味着即使所有外设PCIe端点功能都处于D3状态，CLK_REQ_REG.PERI_CRYSTAL_REQ和CLK_REQ_REG.PERI_WR_PLL_REQ也都必须设置为1。这被定义为SPI模块未处于挂起状态。

11.5 中断和唤醒

SPI控制器模块将产生SPI_CTLR_IRQ（未屏蔽时）；随后会通过外设PCIe端点触发MSI/MSI-X/传统中断。SPI控制器模块将产生SPI_CTLR_WAKE（未屏蔽时）；随后会通过外设PCIe端点触发PCI唤醒。请参见表148。

注： 当通过相应的中断允许/屏蔽寄存器允许SPI_CTLR_IRQ和SPI_CTLR_WAKE时，无论PCIe外设器件或链路处于何种状态，这两个信号都将置为有效。置为有效时，PCIe端点将根据当前PCIe链路状态确定是否需要PCI Wake#/PME事件。如果使能PME且针对当前PMCSR D-状态配置了PME支持，则将该信号置为有效会导致控制器在必要的情况下从L1或L2状态唤醒。当控制器切换回到L0状态时，会发送PME报文并将PME_Status置1。收到PME报文后，PCIe根复合体应清零PME_Status并将D状态切换回D0。

表148： 中断和唤醒操作

SPI事件	已屏蔽	链路状态	器件状态	PCIe事件
SPI_CTLR_WAKE	是	全部	全部	无
	否	L0	D0	MSI/MSI-X/传统中断
	否	L1/L1ss/L2	D0	未定义
	否	L0	D3	未定义
SPI_CTLR_WAKE	是	全部	全部	无
	否	L0	D3	发送PME_Status
	否	L1/L1ss/L2	D3	在切换到L0时生成WAKE#，后跟PME_Status。
	否	全部	D0	未定义

11.6 中断

只要事务结束时SPI_CTLR_CTL_REG.GO位清零，就会触发SPI控制器中断。该中断通过SPI_CTLR_EVENT_REG.SPI_INT位置1来指示，并可通过将SPI_CTLR_EVENT_MASK_REG.SPI_INT_MASK位置1来屏蔽。

只要SPIALERT_CTLR_VAL_REG.SPIALERT_CTLR_VAL_REG位置1，就会触发SPI控制器报警中断。该中断通过SPI_CTLR_EVENT_REG.SPI_ALERT_INT位置1来指示，并可通过将SPI_CTLR_EVENT_MASK_REG.SPI_ALERT_INT_MASK位置1来屏蔽。

当SPI PCIe端点功能处于D0状态时，SPI_CTLR_IRQ输出指示是否有SPI中断待处理。其作用是通过PCIe产生MSI或MSI-X事件或传统PCI中断；如果在MSI/MSI-X中使用单独的向量，中断向量指示哪个SPI实例（0或1）产生了中断。

当SPI PCIe端点功能处于D3状态时，SPI_CTLR_WAKE输出指示是否有SPI控制器中断待处理。其作用是通过PCIe生成唤醒事件。

11.6.1 唤醒支持

SPI控制器支持异步唤醒。SPI控制器通过SPI_CTL事件指示发生了异步唤醒。无论SPI控制器是否处于挂起状态，都可能发生SPI_CTLR_WAKE事件。

当PCIe端点功能处于D3状态且SPI控制器未处于挂起状态时：

- 如果事务结束时SPI_CTLR_CTL_REG.GO位清零，将生成SPI_CTLR_WAKE事件。
- 如果SPI_CTLR*_ALERT_N引脚被拉为低电平且去抖后的输入已稳定，也会生成SPI_CTLR_WAKE事件。此时，SPI_CTLR_EVENT_REG.SPI_ALERT_WAKE位也置1。

当PCIe端点功能处于D3状态且SPI控制器处于挂起状态时，如果SPI_CTLR*_ALERT_N引脚被拉为低电平且去抖后的输入已稳定，将生成SPI_CTLR_WAKE事件。此时，SPI_CTLR_EVENT_REG.SPI_ALERT_WAKE位也置1。

SPI_CTLR_WAKE事件会导致生成PCI唤醒信号。该唤醒请求应触发PCIe主机将SPI控制器置于D0器件状态的操作。此时，聚合的外设PLL请求CLK_REQ_REG.PERI_WR_PLL_REQ和聚合的外设晶振请求CLK_REQ_REG.PERI_CRYSTAL_REQ都将被外设子系统设置为true，因为有一个或多个物理功能现在处于D0状态。

唤醒事件通过SPI_CTLR_EVENT_REG寄存器中的位来指示，并且可通过将SPI_CTLR_EVENT_MASK_REG寄存器中的相应位置1来屏蔽。

11.6.2 SPI报警

经过配置后，PCI1xxxx通过SPI_CTLRx_ALERT_N引脚提供该信号作为SPI外设的输入。

纯SPI外设器件可通过SPIALERT#向主机发出信号，以指示其想要通信。当SPI_CTLRx_ALERT_N引脚被拉为低电平时，表示SPI外设需要与SPI控制器通信。当SPI控制器处于挂起状态时，将触发SPI_CTLR_WAKE事件。当SPI控制器未处于挂起状态时，将产生SPI_CTLR_IRQ中断。

SPI_CTLRx_ALERT_N引脚的焊盘控制在SPI_CTLR_PAD_CTL_REG寄存器中提供；用于触发中断的输入去抖时间可在SPIALERT_CTLR_DB_REG.SPI_CTLR_DB_TIME[11:0]中设置。

SPI_CTLRx_ALERT_N引脚的当前值和去抖后的值都在SPIALERT_CTLR_VAL_REG寄存器中提供。

当SPI_CTLRx_ALERT_N引脚用于唤醒系统且去抖器未使能（SPI_CTLR_PAD_CTL_REG.SPIALERT_CTLR_DB = 0）时，SPI外设需将SPI_CTLRx_ALERT_N置为有效并至少持续6 µs左右。请求环形振荡器时钟启动需要4 µs左右，向主机产生唤醒报文需要2 µs。

11.6.3 退出D3COLD状态时保存功能现场

当器件使用辅助电源或复用主电源/辅助电源（已配置VAUX_DET引脚且为高电平）并且PCIE_PERST_N先置为有效后置为无效时，将发生暖复位。

通常，退出D3cold状态时将复位整个PCIe端点（不包括PME现场）。需要通过系统软件（通常为OS）重新初始化PCIe配置寄存器。需要通过驱动软件重新初始化PCIe端点功能。

尽管可在D3cold期间保存SMBus控制器功能现场，但后续的器件复位将导致接收FIFO和寄存器的内容清空。

可通过使能复位选项来禁止对SMBus控制器功能现场进行暖复位。当SPI_RESET_REG.PERI_SPI_D3_RESET_DIS位置1时，仅复位PCIe端点控制器、配置寄存器及相关逻辑；不会复位SMBus控制器的功能。当OS重新初始化PCIe配置寄存器并将PCIe端点置于D0a状态后，PCIe端点可在驱动程序的控制下继续正常工作，不会丢失功能现场。

11.7 寄存器映射

SPI控制器共有两个实例，每个实例都有自己的SPI控制器寄存器集。下面列出了这些存储器块相对于物理功能PF0的BAR 0/1的偏移量。这些偏移量还给出了通过JTAG/SPI/SMBus访问时，相对于SPI_CONTROLLER_ADDR_BASE的地址。

部分寄存器并非仅针对单个SPI实例，而是适用于所有实例。下列寄存器仅存在于SPI0寄存器集中；应用程序不得对SPI1寄存器集中具有相同偏移地址的寄存器进行读写操作：

- SPI_PCI_CTRL_REG
- SPI_RESET_REG

共有两个具有相同寄存器映射的SPI接口，请参见表149。

表149： SPI控制器寄存器

SPI0偏移地址	SPI1偏移地址	名称	R/W	说明
0x000-0x13Fh	0x800-0x93F	SPI_CTLR_CMD_BUF	R/W	命令输出缓冲区，最多320个字节。 *默认值 = 0x0000_0000*
0x140 - 0x1FF	0x840 - 0x8FF	Reserved	R	保留
0x200 - 0x23F	0x900 - 0x93F	SPI_CTLR_RSP_BUF	R/W	响应输入缓冲区，最多320个字节。 *默认值 = 0x0000_0000*
0x340 - 0x3FF	0xA40 - 0xAFF	Reserved	R	保留

表149： SPI控制器寄存器（续）

SPI0偏移地址	SPI1偏移地址	名称	R/W	说明
0x400	0xB00	SPI_CTLR_CTL_REG	R/W/SC	SPI模式控制 默认值 = 0x0000_0040 Bit [31:28]——保留（全0） Bit [27:25]——DEV_SEL[2:0] 选择要访问的器件 • 000b——SPIx_CEN0 • 001b——SPIx_CEN1 • 010b——SPIx_CEN2 • 011b——SPIx_CEN3 • 100b——SPIx_CEN4 • 101b——SPIx_CEN5 • 110b——SPIx_CEN6 • 111b——保留 Bit [24:17]——保留（全0） Bit [16:8]——CMD_LN[7:0] Bit [7:5]——SPI_SPEED[2:0] 通过62.5 MHz分频系数选择SPI速度 • 000b——关闭 • 001b——保留 • 010b——30 MHz • 011b——20 MHz • 100b——15 MHz • 101b——12 MHz • 110b——10 MHz • 111b——2 MHz Bit [4]——FORCE_CE 该位置1时，强制将SPI芯片使能引脚设为低电平。应仅在使用SPI命令缓冲区时才将该位置1。从总线直接访问时不得将该位置1。 Bit [3]——保留（0b） Bit [2]——MODE_SEL 选择SPI时钟模式 • 1b——模式0 • 0b——模式3 Bit [1]——保留（0b） Bit [0]——GO（自清零位） 将该位置1可启动SPI事务
0x404 - 0x41F	0xB04 - 0xB1F	Reserved	R	保留

表149： SPI控制器寄存器（续）

SPI0偏移地址	SPI1偏移地址	名称	R/W	说明
0x420	0xB20	SPI_CTLR_CTL_REG	R/W1C	事件寄存器 默认值 = 0x0000_0000 Bit [31:10]——保留（全0） Bit [2]——SPI_ALERT_INT 每次SPIALERT_CTLR_VAL_REG.SPIALERT_CTLR_DB_VAL位设置为1b'1时，该位都置1。该位可触发SPI_CTLR_IRQ中断。写入1b'1可清零该位。 Bit [2]——SPI_INT 每次事务结束后SPI_CTLR_CTL_REG.GO位清零时，该位都置1。该位可触发SPI_CTLR_IRQ中断。写入1b'1可清零该位。 Bit [7:2]——保留（全0） Bit [1]——SPI_ALERT_WAKE 当为1b'1时，表示唤醒事件是由于SPI报警置为有效而触发（见第11.6.2节“SPI报警”）。该位可触发SPI_CTLR_WAKE事件。写入1b'1可清零该位。 Bit [0]——SPI_INT_WAKE 当为1b'1时，表示唤醒事件是由于SPI_CTLR_CTL_REG.GO位在事务结束时清零而触发。该位可触发SPI_CTLR_WAKE事件。写入1b'1可清零该位。
0x424	0xB24	SPI_CTLR_EEVENT_REG	R/W	事件屏蔽寄存器 默认值 = 0x0000_0303
0x428 - 0x45F	0xB28 - 0xB5F	Reserved	R	保留
0x460	0xB60	SPI_CTLR_	R/W	SPI控制器焊盘控制寄存器 默认值 = 0x0000_0008
0x464	0xB64	SPI_CTLR_	R/W	SPIALERT#焊盘去抖寄存器 默认值 = 0x0000_000A
0x468	0xB68	SPI_CTLR_	R	SPI ALTER#值寄存器 默认值 = 0x0000_0003
0x470 - 0x47F	0xB70 - 0xB7F	Reserved	R	保留
0x480	n/a	SPI_PCI_CTRL_REG	R/W	SPI PCI控制寄存器 默认值 = 0x0000_0000
0x484	n/a	SPI_RESET_REG	R/W /SC	SPI复位寄存器 默认值 = 0x0000_0000
0x488	0xB88	SPI_LTR_VALUE_REG	R/W	SPI LTR值寄存器 默认值 = 0x0000_0000
0x48C - 0x7FF	0xB8C - 0xFFF	Reserved	R	保留

12.0 外设子系统实现——GPIO

GPIO子系统有自己的PCI器件端点，需要主机系统提供独立的器件驱动程序才能正常工作。该子系统与所有其他外设（UART、SMBus和和SPI）共用PCI总线。

子系统ID如下：

- 供应商ID：0x1055（Microchip Technology, Inc./SMSC）
- 器件ID：
  - PCI12000：0xA005
  - PCI11010：0xA015
  - PCI11101：0xA025
  - PCI11400：0xA035
  - PCI11414：0xA045
- 分类ID：0x130000（“非必要”）

GPIO寄存器仅在PROG引脚功能选择设置为GPIO（始终为FUNCTION 0）时才会影响PROG引脚。GPIO既可配置为输出也可配置为输入。

12.1 章节

12.2 基址

GPIO子系统的配置和控制需要根据用例正确使用不同的基址。通常，用户会用到函数调用（通过外设系统驱动程序使能）。只有在通过修改原始寄存器（而非通过MPLAB® Connect GUI界面选择）构建配置文件或直接访问PCI寄存器时，才需掌握这一知识点。请参见表150。

表150： 编程功能寄存器格式

访问途径	基址
配置（OTP或EEPROM）	0x3_0000
通过PCIe®进行运行时访问	在系统枚举期间通过CPU分配获取。使用PCI总线检查工具定位外设并获取“BAR0”地址。
通过SMBus/SPI进行运行时访问	0x3_0000

12.3 GPIO作为输出

任何GPIO都可以配置为推挽式输出或漏极开路输出。必须在PIO32_OUT_EN、PIO64_OUT_EN和PIO96_OUT_EN寄存器中设置输出使能。引脚模式默认设置为推挽式，但可通过PIO32_OD、PIO64_OD和PIO96_OD寄存器选择设置为漏极开路。

输出状态直接通过PIO32_OUT、PIO64_OUT和PIO96_OUT寄存器进行控制。

12.4 GPIO作为输入

当GPIO配置为输入时，其状态可通过轮询机制（即定期读取输入状态）进行监视，或者可将引脚配置为监视事件并产生中断。

12.4.1 GPIO输入事件

配置为输入的GPIO可配置为基于原始输入或去抖后的输入触发事件。来自PIO的事件（中断或唤醒事件）可编程为基于边沿或电平。

对于基于边沿的事件，触发信号可以是上升沿、下降沿或两个边沿。

对于基于电平的事件，可以选择任一电平。必须设置电平掩码和极性。

根据具体PIO的PIOxx_MODE、PIOxx_HI_TO_LO_EDGE_CONFIG、PIOxx_LO_TO_HI_EDGE_CONFIG、PIOxx_LEVEL_CONFIG和PIOxx_LEVEL_MSK寄存器设置，在PIOxx_STATUS寄存器中记录下降沿、上升沿或电平PIO事件。

PIOxx_STATUS寄存器必须由软件显式清零。向某位写入1b将清零相应位，并使能对下一个电平转换的检测。

注： 如果通过PIOxx_MODE将PIO配置为基于电平，并且引脚的电平保持有效，则PIOxx_STATUS在软件向其写入1b时仍将保持置1状态。

PIOxx_STATUS寄存器中的任一位为1b都将强制执行以下操作之一：

- 如果外设通用功能处于D0状态且中断未被PIOxx_INT_MASK寄存器中的相应位屏蔽，则产生PIO_IRQ中断事件。
- 如果外设通用功能处于D3状态且唤醒未被PIOxx_WAKE_MSK寄存器中的相应位屏蔽，则生成PIO_WAKE事件。

注： 当通过相应的中断允许/屏蔽寄存器允许PIO_IRQ和PIO_WAKE时，无论PCIe外设器件或链路处于何种状态，这两个信号都将置为有效。对于唤醒，PCIe端点将根据当前PCIe链路状态确定是否需要唤醒/PME事件。

12.4.2 支持GPIO输入唤醒

当PIO PCIe端点功能处于D3状态时，PIO支持异步唤醒。PIO通过PIO_WAKE事件指示发生了异步唤醒。无论PIO是否处于挂起状态，都可能发生PIO_WAKE事件。

PIO_WAKE事件会导致生成PCI唤醒信号。该唤醒请求应触发PCIe主机将PIO PCIe端点功能置于D0器件状态的操作。此时，聚合的外设PLL请求CLK_REQW_REG.PERI_WR_PLL_REQ和聚合的外设晶振请求CLK_REQ_REG.PERI_CRYSTAL_REQ都将被外设子系统设置为true，因为有一个或多个物理功能现在处于D0状态。

可通过将PIOx_WAKE_INT_MSK寄存器中的相应位置1来屏蔽唤醒事件。当引脚上发生PIO_WAKE事件且去抖器未使能（PIOxx_DEBOUNCE[xx] = 0）时，该信号需保持有效状态并至少持续6 µs。请求ROSC时钟需要4 µs，向主机产生唤醒报文需要2 µs。

12.5 挂起期间的GPIO操作

当PIO_PCI_CTRL_REG.PIO_D3_CLK_EN位设置为0且PIO PCIe端点功能进入PCIe D3状态时，将关闭提供给PIO的62.5 MHz时钟。发生这种情况时，PIO模块被视为“已挂起”。

当PIO PCI功能返回D0状态时，将恢复时钟。当PIO_PCI_CTRL_REG.PIO_D3_CLK_EN位设置为1时，如果PIO PCIe端点功能进入D3状态，将关闭提供给PIO的62.5 MHz时钟。该模块将继续工作。这也意味着即使所有外设PCIe端点功能都处于D3状态，CLK_REQ_REG.PERI_CRYSTAL_REQ和CLK_REQ_REG.PERI_WR_PLL_REQ也都必须设置为1。

12.6 GPIO配置寄存器

通过这些GPIO配置寄存器，可以设置属性以修改GPIO引脚的行为。每项设置对应三个32位寄存器，其中每个位对应96个GPIO引脚之一。设置包括输入/输出使能、去抖使能、引脚状态、中断允许和上拉/下拉使能等，如表151所示。

表151： 器件属性和编程选项

偏移地址	名称	R/W	说明	默认值
0x0003_4000	PIO32_OUT_EN	R/W	GPIO输出使能寄存器 0b——禁止输出 1b——使能输出 位映射： • Bit 0——PIO1 • Bit 1-30——PIO2 • Bit 31——PIO32	0x0000_0000
0x0003_4004	PIO64_OUT_EN	R/W	GPIO输出使能寄存器 0b——禁止输出 1b——使能输出 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000
0x0003_4008	PIO96_OUT_EN	R/W	GPIO输出使能寄存器 0b——禁止输出 1b——使能输出 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_400C - 0x0003_400F	Reserved	R	保留	全0
0x0003_4010	PIO32_INP_EN	R/W	GPIO输入使能寄存器 0b——禁止输入 1b——使能输入 位映射： • Bit 0——PIO1 • Bit 1-30——PIO2 • Bit 31——PIO32	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_4014	PIO64_INP_EN	R/W	GPIO输入使能寄存器 0b——禁止输入 1b——使能输入 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000
0x0003_4018	PIO96_INP_EN	R/W	GPIO输入使能寄存器 0b——禁止输入 1b——使能输入 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_401C - 0x0003_401F	Reserved	R	保留	全0
0x0003_4020	PIO32_OUT	R/W	GPIO输出状态寄存器 0b——驱动为低电平 1b——驱动为高电平 位映射： • Bit 0——PIO1 • Bit 1-30——PIO2 • Bit 31——PIO32	0x0000_0000
0x0003_4024	PIO64_OUT	R/W	GPIO输出状态寄存器 0b——驱动为低电平 1b——驱动为高电平 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000
0x0003_4028	PIO96_OUT	R/W	GPIO输出状态寄存器 0b——驱动为低电平 1b——驱动为高电平 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_402C - 0x0003_402F	Reserved	R	保留	全0

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_4040	PIO32_IN	R/W	GPIO输入状态寄存器 0b——输入低电平 1b——输入高电平 位映射： • Bit 0——PIO1 • Bit 1-30——PIO2 • Bit 31——PIO32	0x0000_0000
0x0003_4044	PIO64_IN	R/W	GPIO输入状态寄存器 0b——输入低电平 1b——输入高电平 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000
0x0003_4048	PIO96_IN	R/W	GPIO输入状态寄存器 0b——输入低电平 1b——输入高电平 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_403C - 0x0003_403F	Reserved	R	保留	全0
0x0003_4040	PIO32_PU	R/W	使能内部上拉电阻 0b——禁止上拉 1b——使能上拉 位映射： • Bit 0——PIO1 • Bit 1-30——PIO2 • Bit 31——PIO32	0x0000_0000
0x0003_4044	PIO64_PU	R/W	使能内部上拉电阻 0b——禁止上拉 1b——使能上拉 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_4048	PIO96_PU	R/W	使能内部上拉电阻 0b——禁止上拉 1b——使能上拉 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_404C - 0x0003_404F	Reserved	R	保留	全0
0x0003_4050	PIO32_PD	R/W	使能内部下拉电阻 0b——禁止下拉 1b——使能下拉 位映射： • Bit 0——PIO1 • Bit 1-30——PIO2 • Bit 31——PIO32	0x0000_0000
0x0003_4054	PIO64_PD	R/W	使能内部下拉电阻 0b——禁止下拉 1b——使能下拉 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000
0x0003_4058	PIO96_PD	R/W	使能内部下拉电阻 0b——禁止下拉 1b——使能下拉 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29 - 31——未使用	0x0000_0000
0x0003_405C - 0x0003_405F	Reserved	R	保留	全0
0x0003_4060	PIO32_OD	R/W	将引脚模式设置为漏极开路（如果配置为输出） 0b——推挽式 1b——漏极开路 位映射： • Bit 0——PIO1 • Bit 1-30——PIO2 • Bit 31——PIO32	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_4064	PIO64_OD	R/W	将引脚模式设置为漏极开路（如果配置为输出） 0b——推挽式 1b——漏极开路 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000
0x0003_4068	PIO96_OD	R/W	将引脚模式设置为漏极开路（如果配置为输出） 0b——推挽式 1b——漏极开路 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_406C - 0x0003_406F	Reserved	R	保留	全0
0x0003_4070	PIO32_WAKE_ MSK	R/W	0b：允许事件触发产生GPIO中断 1b：禁止事件触发产生GPIO中断 位映射： • Bit 0——PIO31 • Bit 1-30——PIO33 • Bit 31——PIO63	0xFFFF_FFFF
0x0003_4074	PIO64_WAKE_ MSK	R/W	0b：允许事件触发产生GPIO中断 1b：禁止事件触发产生GPIO中断 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0xFFFF_FFFF
0x0003_4078	PIO96_WAKE_ MSK	R/W	0b：允许事件触发产生GPIO中断 1b：禁止事件触发产生GPIO中断 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x1FFF_FFFF

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_407C - 0x0003_407F	Reserved	R	保留	全0
0x0003_4080	PIO32_MODE	R/W	0b——GPIO事件边沿触发模式 1b——GPIO事件电平触发模式 位映射： • Bit 0——PIO31 • Bit 1-30——PIO33 • Bit 31——PIO63	0x0000_0000
0x0003_4084	PIO64_MODE	R/W	0b——GPIO事件边沿触发模式 1b——GPIO事件电平触发模式 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000
0x0003_4088	PIO96_MODE	R/W	0b——GPIO事件边沿触发模式 1b——GPIO事件电平触发模式 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_408C - 0x0003_408F	Reserved	R	保留	全0
0x0003_4090	PIO32_LO_TO_HI_EDGE_CONFGIG	R/W	0b——允许在相应PIO线从低电平跳变到高电平时生成事件。 1b——禁止在相应PIO线从低电平跳变到高电平时生成事件。 位映射： • Bit 0——PIO31 • Bit 1-30——PIO33 • Bit 31——PIO63 **注：**对于边沿触发，如果高电平边沿和低电平边沿屏蔽位都置1，则不会发生PIO边沿事件。该事件仅发生在未屏蔽的方向上。	0xFFFF_FFFF

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_4094	PIO64_LO_TO_HI_EDGE_CONFGIG	R/W	0b——允许在相应PIO线从低电平跳变到高电平时生成事件。 1b——禁止在相应PIO线从低电平跳变到高电平时生成事件。 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64 **注：**对于边沿触发，如果高电平边沿和低电平边沿屏蔽位都置1，则不会发生PIO边沿事件。该事件仅发生在未屏蔽的方向上。	0xFFFF_FFFF
0x0003_4098	PIO96_LO_TO_HI_EDGE_CONFGIG	R/W	0b——允许在相应PIO线从低电平跳变到高电平时生成事件。 1b——禁止在相应PIO线从低电平跳变到高电平时生成事件。 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用 **注：**对于边沿触发，如果高电平边沿和低电平边沿屏蔽位都置1，则不会发生PIO边沿事件。该事件仅发生在未屏蔽的方向上。	0x1FFF_FFFF
0x0003_409C - 0x0003_409F	Reserved	R	保留	全0
0x0003_40A0	PIO32_HI_TO_LOW_EDGE_CONFIG	R/W	0b——允许在相应PIO线从高电平跳变到低电平时生成事件。 1b——禁止在相应PIO线从高电平跳变到低电平时生成事件。 位映射： • Bit 0——PIO1 • Bit 1-30——PIO33 • Bit 31——PIO32	0xFFFF_FFFF

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40A4	PIO64_HI_TO_LOW_EDGE_CONFIG	R/W	0b——允许在相应PIO线从高电平跳变到低电平时生成事件。 1b——禁止在相应PIO线从高电平跳变到低电平时生成事件。 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0xFFFF_FFFF
0x0003_40A8	PIO96_HI_TO_LOW_EDGE_CONFIG	R/W	0b——允许在相应PIO线从高电平跳变到低电平时生成事件。 1b——禁止在相应PIO线从高电平跳变到低电平时生成事件。 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x1FFF_FFFF
0x0003_40AC - 0x0003_40AF	Reserved	R	保留	全0
0x0003_40B0	PIO32_LEVEL_CONFIG	R/W	0b—— 电平模式下事件极性不变；当引脚为高电平时检测到事件。 1b—— 电平模式下事件极性反相；当引脚为低电平时检测到事件。 位映射： • Bit 0——PIO1 • Bit 1-30——PIO33 • Bit 31——PIO32	0x0000_0000
0x0003_40B4	PIO64_LEVEL_CONFIG	R/W	0b—— 电平模式下事件极性不变；当引脚为高电平时检测到事件。 1b—— 电平模式下事件极性反相；当引脚为低电平时检测到事件。 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40B8	PIO96_LEVEL_CONFIG	R/W	0b——电平模式下事件极性不变；当引脚为高电平时检测到事件。 1b——电平模式下事件极性反相；当引脚为低电平时检测到事件。 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x0000_0000
0x0003_40BC - 0x0003_40BF	Reserved	R	保留	全0
0x0003_40C0	PIO32_LEVEL_MSK	R/W	0b——不屏蔽电平模式事件。 1b——屏蔽电平模式事件。 位映射： • Bit 0——PIO1 • Bit 1-30——PIO33 • Bit 31——PIO32	0xFFFF_FFFF
0x0003_40C4	PIO64_LEVEL_MSK	R/W	0b——不屏蔽电平模式事件。 1b——屏蔽电平模式事件。 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0xFFFF_FFFF
0x0003_40C8	PIO96_LEVEL_MSK	R/W	0b——不屏蔽电平模式事件。 1b——屏蔽电平模式事件。 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x1FFF_FFFF
0x0003_40CC - 0x0003_40CF	Reserved	R	保留	全0

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40D0	PIO32_STATUS	R/W 1C	0b——未发生事件变化。 1b——已发生事件变化。 在边沿模式下，当输入引脚发生跳变（从低电平到高电平或从高电平到低电平）时，该位将置1。 在电平模式下，如果PIOxx_LEVEL_CONFIG = 0b，该位将在引脚状态为高电平时置1；如果PIOxx_LEVEL_CONFIG = 1b，该位将在引脚状态为低电平时置1。 可通过向相应位写入1b将事件清除。但是，在电平模式下，如果引脚仍处于有效状态，该状态位将保持置1。 位映射： • Bit 0——PIO1 • Bit 1-30——PIO33 • Bit 31——PIO32	0x0000_0000
0x0003_40D4	PIO64_STATUS	R/W 1C	0b——未发生事件变化。 1b——已发生事件变化。 在边沿模式下，当输入引脚发生跳变（从低电平到高电平或从高电平到低电平）时，该位将置1。 在电平模式下，如果PIOxx_LEVEL_CONFIG = 0b，该位将在引脚状态为高电平时置1；如果PIOxx_LEVEL_CONFIG = 1b，该位将在引脚状态为低电平时置1。 可通过向相应位写入1b将事件清除。但是，在电平模式下，如果引脚仍处于有效状态，该状态位将保持置1。 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40D8	PIO96_STATUS	R/W 1C	0b——未发生事件变化。 1b——已发生事件变化。 在边沿模式下，当输入引脚发生跳变（从低电平到高电平或从高电平到低电平）时，该位将置1。 在电平模式下，如果PIOxx_LEVEL_CONFIG = 0b，该位将在引脚状态为高电平时置1；如果PIOxx_LEVEL_CONFIG = 1b，该位将在引脚状态为低电平时置1。 可通过向相应位写入1b将事件清除。但是，在电平模式下，如果引脚仍处于有效状态，该状态位将保持置1。 位映射： • Bit 0——PIO65 • Bit 1-26——PIO66 • Bit 27——PIO92 • Bit 28-31——未使用	0x0000_0000
0x0003_40DC - 0x0003_40DF	Reserved	R	保留	全0
0x0003_40E0	PIO32_ DEBOUNCE	R/W	0b——不应用去抖，直接传递原始输入。 1b——如果置1且相应的PIO配置为输入，则PIO输入将按照PIO_GLOBAL_CONFIG.PIO_DEBOUNCE[11:0]中指定的量进行去抖。 位映射： • Bit 0——PIO1 • Bit 1-30——PIO33 • Bit 31——PIO32	0x0000_0000
0x0003_40E4	PIO64_ DEBOUNCE	R/W	0b——不应用去抖，直接传递原始输入。 1b——如果置1且相应的PIO配置为输入，则PIO输入将按照PIO_GLOBAL_CONFIG.PIO_DEBOUNCE[11:0]中指定的量进行去抖。 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40E8	PIO96_ DEBOUNCE	R/W	0b——不应用去抖，直接传递原始输入。 1b——如果置1且相应的PIO配置为输入，则PIO输入将按照PIO_GLOBAL_CONFIG.PIO_DEBOUNCE[11:0]中指定的量进行去抖。 位映射： • Bit 0——PIO65 • Bit 1-26——PIO66 • Bit 27——PIO92 • Bit 28-31——未使用	0x0000_0000
0x0003_40EC - 0x0003_40EF	Reserved	R	保留	全0
0x0003_40F0	PIO_GLOBAL_ CONFIG	R/W	Bit [31:18]——保留 Bit [17]——PIO_WAKE_MASTER_MASK 0b——不屏蔽PIO唤醒事件。 1b——屏蔽所有PIO唤醒事件。 这是对馈送到PCIe®端点的单一唤醒事件的最终门控（93个可能的单独PIO唤醒事件的逻辑或函数）。 Bit [16]——PIO_INT_MASTER_MASK 0b——不屏蔽PIO中断事件。 1b——屏蔽所有PIO中断事件。 这是对馈送到PCIe端点的单一唤醒事件的最终门控（93个可能的单独PIO唤醒事件的逻辑或函数）。 Bit [15:12]——保留 Bit [11:0]——PIO_DEBONCE[11:0] 该位域用于保存GPIO的去抖定时器值。 当检测到信号跳变时，去抖器启动。如果引脚状态在配置的去抖时间内保持不变，则直接传递该跳变信号。如果在去抖时间内再次发生信号跳变，去抖器将重新启动。 每个计数对应1 ms，默认值为10 ms。计时基于自由运行时钟，会随边沿对齐情况而变化。计时精度限制为±一个计数。	0x0003_000A

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40F4	PIO_PCI_CTRL_REG	R/W	Bit [31:1]——保留 Bit [0]——PIO_D3_CLK_EN 在PCI功能处于D3状态时使能62.5 MHz时钟： 0b——在PCI功能处于D3状态时禁止62.5 MHz时钟。 1b——在PCI功能处于D3状态时使能62.5 MHz时钟。	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40F8	PIO_LTR_ VALUE_REG	R/W	Bit [31]——PIO_NO_SNOOP_REQ 0b——无延时需求，忽略 PIO_NO_SNOOP_LATENCY_VAL[9:0]和 PIO_NO_SNOOP_LATENCY_SCALE[2:0] 位域。 1b——有延时需求，具体取决于 PIO_NO_SNOOP_LATENCY_VAL[9:0]和 PIO_NO_SNOOP_LATENCY_SCALE[2:0]位域 Bit [30:29]——保留 Bit [28:26]—— PIO_NO_SNOOP_LATENCY_SCALE[2:0] PIO_NO_SNOOP_LATENCY_VAL[9:0]位域 调节： 000b——值乘以1 ns 001b——值乘以32 ns 010b——值乘以1024 ns 011b——值乘以32768 ns 100b——值乘以1048576 ns 101b——值乘以33554432 ns 110b至111b——不允许值执行乘法 Bit [25:16]—— PIO_NO_SNOOP_LATENCY_VAL[9:0] 延时值位域 Bit [15]——PIO_SNOOP_REQ 0b——无延时需求，忽略 PIO_SNOOP_LATENCY_VAL[9:0]和 PIO_SNOOP_LATENCY_SCALE[2:0]位域。 1b——有延时需求，具体取决于 PIO_SNOOP_LATENCY_VAL[9:0]和 PIO_SNOOP_LATENCY_SCALE[2:0]位域。 Bit [14:13]——保留 Bit [12:10]——PIO_SNOOP_LATENCY_ SCALE[2:0] PIO_SNOOP_LATENCY_VAL[9:0]位域调节： 000b——值乘以1 ns 001b——值乘以32 ns 010b——值乘以1024 ns 011b——值乘以32768 ns 100b——值乘以1048576 ns 101b——值乘以33554432 ns 110b至111b——不允许值执行乘法 Bit [9:0]——PIO_SNOOP_LATENCY_VAL[9:0] 侦听延时值	0x0000_0000

表151： 器件属性和编程选项（续）

偏移地址	名称	R/W	说明	默认值
0x0003_40FC - 0x0003_40FF	Reserved	R	保留	全0
0x0003_4100	PIO32_INT_MSK	R/W	0b——不屏蔽中断。 1b——屏蔽中断。 位映射： • Bit 0——PIO1 • Bit 1-30——PIO33 • Bit 31——PIO32	0xFFFF_FFFF
0x0003_4104	PIO64_INT_MSK	R/W	0b——不屏蔽中断。 1b——屏蔽中断。 位映射： • Bit 0——PIO33 • Bit 1-30——PIO34 • Bit 31——PIO64	0xFFFF_FFFF
0x0003_4108	PIO96_INT_MSK	R/W	0b——不屏蔽中断。 1b——屏蔽中断。 位映射： • Bit 0——PIO65 • Bit 1-27——PIO66 • Bit 28——PIO93 • Bit 29-31——未使用	0x1FFF_FFFF
0x0003_410C - 0x0003_43FF	Reserved	R	保留	全0

13.0 配置文件格式

13.1 章节

- 第13.2节“配置存储器结构（适用于OTP和EEPROM）”
- 第13.3节“配置命令格式”

13.2 配置存储器结构（适用于OTP和EEPROM）

配置数据分为两个主要区域：

- MEM_STAT区域具有特定参数的固定地址，用于向PCI1xxxx指示哪些存储器区域包含配置数据块，以及哪些硬件子系统实际上具有修改默认设置的配置数据。
- 配置数据区域包含实际的配置数据块。每个配置数据块的长度可变，具体取决于特定应用所需的配置设置数量。

要配置OTP，应按照以下步骤操作：

1. 配置MEM_STAT存储器：
   - 写入MAGIC_BYTE以指示存储器已配置。
   - 写入OTP_PROG_COUNT中的下一个位以指示OTP已编程的次数。

注：	即使OTP_PROG_COUNT中的所有位都已置1，也仍可配置OTP存储器。惟一的影响是不能再使用该计数器跟踪额外的OTP编程事件。

   - 写入MEMORY_SIZE存储器以指示存储器大小。
   - 写入MEM_REGION_PROG存储器中的位以指示将要编程的存储器区域；每个要编程的子系统都需要一个存储器区域。
   - 如需使存储器区域失效，应将OTP_REGION_INV中的相应位置1。该操作通常用于为一个子系统替换新的配置块。在这种情况下，会将之前已编程的存储器区域设为无效，然后在MEM_REGION_PROG中设置新区域。
   - 对于每个存储器区域，配置存储器区域描述符 MEM_REGION_DESCRx。写入 MEM_TAG[7:0] 以指示存储器区域对应的子系统，并写入MEM_START_ADDRESS[12:0]以指示OTP中配置块的起始地址。

注：	选择MEM_START_ADDRESS[12:0]时应确保每个配置块都有足够的存储空间，并且各存储器区域之间互不重叠。

2. 按照第13.3节“配置命令格式”中定义的格式配置由MEM_REGION_DESCRx指向的每个存储器区域。
3. 对于任何需要写锁定的存储器区域，通过将 OTP_REGION_WR_PROT 中与存储器区域对应的位置 1 来进行配置。

注：	区域被标记为写保护后，将永远无法进行更改或扩展。因此，务必先确认将来不再需要额外的编程或扩展，然后再对存储器进行写保护。

对于任何需要读锁定的存储器区域，通过将OTP_REGION_RD_PROT中与存储器区域对应的位置1来进行配置，请参见表152。

表152： 配置存储器结构

偏移量	大小	名称	说明
0x00h	1	MAGIC_BYTE	编程为A5h时表示已配置OTP。所有其他值表示未编程或未正确编程OTP，应忽略。
0x01	1	Reserved	保留

表152： 配置存储器结构（续）

偏移量	大小	名称	说明
0x02 - 0x03	2	OTP_PROG_COUNT	OTP：OTP存储器已编程的次数。每个单独的位表示一次OTP编程实例，因此计数器最多只能跟踪16次OTP编程事件。首次对OTP编程后，应从LSB开始置1，后续每次编程操作依次向高位递进，直到第16次编程后将第16个位置1为止。如果存在跳位情况，计数器的含义将变得不确定/已损坏。 示例： • 0000_0000_0000_0001b：OTP已编程一次 • 0000_0000_0000_1111b：OTP已编程4次 • 1111_1111_1111_1111b：OTP已编程16次。
0x04 - 0x07	4	MEMORY_SIZE	MEMORY_SIZE用于给出OTP的最大大小（以字节为单位）；供硬件用于检测溢出。不得配置或访问超出MEMORY_SIZE的存储空间。
0x08 - 0x0B	4	MEM_REGION_PROG	32位存储器区域，用于指示32个存储器区域中哪些已编程；每个位对应32个存储器区域（0至31）之一。 例如，如果值为0x0000_9221，则表示存储器区域0、5、9、12和15已编程，应由硬件重载。
0x0C - 0x0F	4	OTP_REGION_INV	OTP特定位域，用于允许在编程配置块后使其无效。这是一个32位的存储器区域，用于指示32个存储器区域中的哪些已失效；每个位对应32个存储器区域（0至31）之一。 例如，要使区域12失效，设置OTP_REGION_INV = 0x0000_8000。
0x10 - 0x13	4	OTP_REGION_WR_PROT	OTP特定位域，用于允许对OTP存储器的特定区域进行写锁定。这是一个32位的存储器区域，用于指示32个存储器区域中的哪些已进行写锁定；每个位对应32个存储器区域（0至31）之一。 例如，要对区域0进行写锁定，设置OTP_REGION_WR_PROT = 0x0000_0001。
0x14 - 0x17	4	OTP_REGION_RD_PROT	OTP特定位域，用于允许对OTP存储器的特定区域进行读锁定。这是一个32位的存储器区域，用于指示32个存储器区域中的哪些已进行读锁定；每个位对应32个存储器区域（0至31）之一。 例如，要对区域0进行读锁定，设置OTP_REGION_RD_PROT = 0x0000_0001。
0x18 - 0x2F	17	Reserved	保留

表152： 配置存储器结构（续）

偏移量	大小	名称	说明
0x30 - 0x33	4	MEM_REGION_DESCR0	存储器区域描述符。硬件使用该描述符来获悉配置数据块的存储单元，以及配置数据块归属于哪个硬件子系统。 **Bit [31:29]**：保留 **Bit [28:16]**：MEM_START_ADDRESS：存储器区域的起始地址。 **Bit [15:8]**：保留 **Bit [7:0]**：MEM_TAG —— 指示存储器区域与哪个子系统相关，以及指向配置块的指针 MEM_TAG可以是下列值之一： • 0x00：保留 • 0x01：USB子系统标记 • 0x02：以太网子系统标记 • 0x03：UART物理功能子系统标记 • 0x04：SMBus物理功能子系统标记 • 0x05：SPI物理功能子系统标记 • 0x06：通用物理功能子系统标记 • 0x07：PCIe®交换芯片子系统主标记 • 0x08：PCIe交换芯片子系统端口0标记 • 0x09：PCIe交换芯片子系统端口1标记 • 0x0A：PCIe交换芯片子系统端口2标记 • 0x0B：PCIe交换芯片子系统端口3标记 • 0x0C：PCIe交换芯片子系统端口4标记 • 0x0D：系统寄存器基址 • 0x0E：PCIe PHY A标记 • 0x0F：PCIe PHY B标记 • 0x10：PCIe PHY C标记 • 0x11 - 0xFF：保留
0x34 - 0x37	4	MEM_REGION_DESCR1	
0x38 - 0x3B	4	MEM_REGION_DESCR2	
0x3C - 0x3F	4	MEM_REGION_DESCR3	
0x40 - 0x43	4	MEM_REGION_DESCR4	
0x44 - 0x47	4	MEM_REGION_DESCR5	
0x48 - 0x4B	4	MEM_REGION_DESCR6	
0x4C - 0x4F	4	MEM_REGION_DESCR7	
0x50 - 0x53	4	MEM_REGION_DESCR8	
0x54 - 0x57	4	MEM_REGION_DESCR9	
0x58 - 0x5B	4	MEM_REGION_DESCR10	
0x5C - 0x5F	4	MEM_REGION_DESCR11	
0x60 - 0x63	4	MEM_REGION_DESCR12	
0x64 - 0x67	4	MEM_REGION_DESCR13	
0x68 - 0x6B	4	MEM_REGION_DESCR14	
0x6C - 0x6F	4	MEM_REGION_DESCR15	
0x70 - 0x73	4	MEM_REGION_DESCR16	
0x74 - 0x77	4	MEM_REGION_DESCR17	
0x78 - 0x7B	4	MEM_REGION_DESCR18	
0x7C - 0x7F	4	MEM_REGION_DESCR19	
0x80 - 0x73	4	MEM_REGION_DESCR20	
0x84 - 0x77	4	MEM_REGION_DESCR21	
0x88 - 0x7B	4	MEM_REGION_DESCR22	
0x8C - 0x7F	4	MEM_REGION_DESCR23	
0x90 - 0x93	4	MEM_REGION_DESCR24	
0x94 - 0x97	4	MEM_REGION_DESCR25	
0x98 - 0x9B	4	MEM_REGION_DESCR26	
0x9C - 0x9F	4	MEM_REGION_DESCR27	
0xA0 - 0xA3	4	MEM_REGION_DESCR28	
0xA4 - 0xA7	4	MEM_REGION_DESCR29	
0xA8 - 0xAB	4	MEM_REGION_DESCR30	
0xAC - 0xAF	4	MEM_REGRION_DESC31	
0xB0 - 0xFF	79	Reserved	保留
0100h - 1F40h	4,096	配置数据块	所有配置数据块都编程到该存储器区域中。每个配置数据块的精确位置并不重要且不受强制要求，因为存储器起始区域已编程到MEM_REGION_DESCx存储器中。 通常，各配置区域采用连续编程（中间不留空白存储空间），但也可在各配置存储器区域之间预留缓冲空间，以便未来添加和升级功能。

13.3 配置命令格式

配置数据仅包含需更改默认值的寄存器修改。无需更改的寄存器不得包含在配置数据块中。配置数据通过一系列由用户定义长度的命令实现。每个配置数据块仅用于配置单个信号硬件模块。每个配置数据块都编程到32个可用存储器区域之一，请参见表153。

操作码主要有以下三个：

- STOP（0x00）：表示配置数据块结束。
- SET_MEMORY_ADRESS（0x80）：将存储器地址指针设置为所选地址，然后将可变长度的数据写入存储器以及从所选起始地址递增的所有连续存储器地址。
- SKIP_MEMORY（0x81-0xFF）：将存储器地址指针跳过最多127个存储单元，然后将可变长度的数据写入存储器以及从所选起始地址递增的所有连续存储器地址。

表153： OTP存储命令

命令	操作码	参数	说明
STOP	0x00	N/A	配置块中的最后一个操作码应为STOP。如需扩展配置块，可随时使用额外命令改写STOP，但扩展后的配置块不得与其他块发生重叠，并且命令本身后跟STOP以终止配置块。 **注：**OTP存储器中每个字节的默认值均为0x00。因此，STOP被定义为该值以使配置块具有自然的终止值（OTP默认值），从而在需要时便于进行扩展。
SET_MEMORY_ADDRESS	0x80	MEMORY_ADDRESS	将当前地址单元设置为指定地址。
		LENGTH	从MEMORY_ADDRESS参数中所选的存储空间开始写入存储器的连续DWORD数。 该值可以是0x01-0x7F（1到127个DWORD）。 0x00为非法值。
		DATA	要写入的DWORD。数据长度必须等于LENGTH参数。数据按小尾数格式排序。

表153： OTP存储命令（续）

命令	操作码	参数	说明
SKIP_MEMORY	0x81 - 0xFF	SKIP_DWORDS	将存储器地址指针跳过一定数量的地址，具体数量为所使用命令的值与0x80之差。该参数有助于OTP命令节省非易失性存储空间。 可指定为0x81 - 0xFF（跳过1到127个地址） 示例： 0x84将使地址指针跳过4个DWORD。如果起始地址为0x0FF0_0000，SKIP_DWORDS命令的操作码为0x84，则要配置的存储器地址将为0x0FF0_0010h
		LENGTH	从MEMORY_ADDRESS参数中所选的存储空间开始写入存储器的连续DWORD数。 该值可以是0x01-0x7F（1到127个DWORD）。 0x00为非法值。
		DATA	要写入的DWORD。数据长度必须等于LENGTH参数。数据按小尾数格式排序。

13.4 配置命令示例

1. 向单个存储器地址写入值。

要将0xAAAB_ACAD写入寄存器0xBF80_3000，请参见表154。

表154： 例1——OTP配置命令

字节偏移量	字节	说明
0	0x80	SET_MEMORY_ADDRESS的操作码
1	0x00	MEMORY_ADRESS参数
2	0x30	
3	0x80	
4	0xBF	
5	0x01	LENGTH
6	0xAD	要写入MEMORY_ADDRESS的数据
7	0xAC	
8	0xAB	
9	0xAA	
10	0x00	STOP

2. 向两个存储器地址写入值。

要将0xAAAB_ACAD写入位于0xBF80_3000处的寄存器并将0x5556_5758写入位于0xBF80_300C处的寄存器，请参见表155。

表155： 例2——OTP配置命令

字节偏移量	字节	说明
0	0x80	SET_MEMORY_ADDRESS的操作码
1	0x00	MEMORY_ADRESS参数
2	0x30	
3	0x80	
4	0xBF	
5	0x01	LENGTH
6	0xAD	要写入MEMORY_ADDRESS的数据
7	0xAC	
8	0xAB	
9	0xAA	
10	0x82	SKIP_MEMORY的操作码；跳过2个存储器地址（0x82-0x80 DWORD）。
11	0x01	LENGTH
12	0x58	要写入MEMORY_ADDRESS + 2的数据
13	0x57	
14	0x56	
15	0x55	
16	0x00	STOP

14.0 OTP编程（读写操作）

14.1 章节

14.2 引导时装载OTP存储器

PCI1XXXX在引导时从OTP装载的步骤如下：

1. 检查第一个字节是否为MAGIC_BYTE。如果存在MAGIC_BYTE，则继续。否则，不配置OTP并跳过从OTP装载（是否使用硬件默认值取决于是否存在其他配置选项）。
2. 锁存MEMORY_SIZE并将其用作校验值，以确保PCI1xxxx不会因配置不当而使读取范围超出已配置存储器的末端。当PCI1xxxx到达存储器末端时，将停止读取并假定配置已完成。
3. 锁存MEM_REGION_PROG以确定哪些存储器区域已编程。MEM_REGION_PROG中的每个位置1时都表示对应的存储器描述符已编程。
4. 锁存OTP_REGION_INV以确定哪些存储器区域有效。OTP_REGION_INV中的每个位置1时都表示对应的存储器描述符已失效。
5. PCI1xxxx遍历每个已编程（根据MEM_REGION_PROG）但尚未失效（根据OTP_REGION_INV）的存储器描述符MEM_REGION_DESRx。忽略其他存储器描述符。
6. PCI1xxxx读取MEM_TAG[7:0]存储器，该存储器指示要配置的子系统或功能。PCI1xxxx会验证该子系统对于所选器件编号是否有效，并通过检查GENERAL_SYS_CONFIG_REQ_REG [7:0]确认子系统或功能是否在请求配置，然后再继续配置子系统或功能。如果子系统不需要配置，则PCI1xxxx转到步骤10。
7. 对于要配置的子系统，PCI1xxxx读取MEM_REGION_DESRx中的MEM_START_ADDRESS[12:0]存储器，该存储器指向配置的存储器区域。硬件设置CFG_BYTE = MEM_START_ADDRESS[12:0]且MEM_DWORD = 0x0000_0000。
8. 硬件读取存储在CFG_BYTE中的字节：

   8.1 如果该字节为0x00，则为STOP操作码。配置块中无更多数据可读。硬件转到步骤10。

   8.2 如果该字节为0x80，则为SET_MEMORY_ADDRESS操作码。硬件将CFG_BYTE加1，然后读取MEMORY_ADDRESS并将其存入MEM_DWORD。硬件将CFG_BYTE加4。

   8.3 如果该字节为0x81至0xFF，则为SKIP_DWORDS操作码。如果MEMORY_ADDRESS = 0x0000_0000，表示SKIP_DWORDS操作码前面没有SET_MEMORY_ADDRESS操作码，硬件会将GENERAL_SYS_OTP_CONFIG_ERR_REG（对于OTP）中的相关错误位以及相应的MEM_TAG[7:0]置1，然后转到步骤10。硬件将MEM_DWORD中存储的值加4*(操作码 - 0x80)。硬件将CFG_BYTE加1。

   8.4 硬件读取LENGTH。

   8.5 如果LENGTH = 0，则为错误。硬件针对OTP以及相应的MEM_TAG[7:0]（bit 0至bit 7）将GENERAL_SYS_OTP_CONFIG_ERR_REG中的相关错误位置1，然后转到步骤10。

   8.6 硬件将CFG_BYTE加1并读取长度为LENGTH个DWORD的数据。

   8.7 硬件将CFG_BYTE加4*LENGTH。

   8.8 硬件将LENGTH个DWORD的数据写入MEMORY_ADDRESS。如果LENGTH = 1，硬件将使用MEMORY_ADDRESS的bit 0和bit 1位作为字节选择器（见表160）。

   8.9 硬件将MEM_DWORD中存储的值加4*LENGTH。

8.10 硬件返回步骤8。

9. 只要读取的CFG_BYTE超出MEMORY_SIZE指示的存储器的末端，硬件就会将GENERAL_SYS_OTP_CONFIG_ERR_REG.OTP_MEM_OVERSHOOT_ERR位置1，并立即转到步骤10。
10. 如果还有更多子系统需要配置，硬件返回步骤5。
11. 如果不存在指示需通过EEPROM或SPI/SMBus进行配置的配置脚，硬件将在此时启动子系统。

14.3 OTP读/写保护

OTP存储器的起始区域由硬件保留，用于存储控制JTAG访问权限和保护OTP存储器区域的值，使其永久不可修改。

OTP_REGION_WR_PROT和OTP_REGION_RD_PROT各有1个位对应由MEM_REGION_DESCRx描述符描述的32个存储器区域之一。当OTP_REGION_WR_PROT中与某个存储器区域对应的位置1时，表示该存储器区域受OTP硬件模块的写保护，不再支持写入。

当OTP_REGION_RD_PROT中与某个存储器区域对应的位置1时，表示该存储器区域受OTP硬件模块的读保护，不再支持芯片从外部读取。如果受读保护，则受读保护的OTP值只能由PCI1xxxx本身在从OTP装载系统配置期间读取。

14.4 手动OTP编程

所有OTP地址的未编程默认状态均为零。编程操作每次仅能写入1位。OTP_PROGRAM命令将OTP数据寄存器的内容编程到OTP地址指定的存储单元。

1. 通过清零OTP_PWR_DN寄存器中的OTP_PWRDN_N使OTP退出掉电状态。
2. 通过设置OTP_PRGM_MODE寄存器中的OTP_PGM_MODE_BYTE来相应地编程位模式或字节模式。
3. 编程各种脉冲宽度CSR。
4. 写入OTP_ADDR_HIGH/OTP_ADDR_LOW/OTP_ADDR_BITS寄存器。如果使用字节模式，则忽略OTP_ADDR_BITS寄存器。
5. 将要编程的数据模式写入OTP_PRGM_DATA寄存器。尽管OTP仅支持位写操作，但在选择字节编程时，硬件会进行相应的调整。硬件仅尝试编程值为1的位，因为OTP位单元默认为零。如果使能位编程，则该寄存器中只有bit 0有效。
6. 编程OTP_MAX_PRG寄存器中的OTP_MAX_PROG[4:0]以设置编程周期数。
7. 将OTP_FUNC_CMD寄存器中的OTP_PROGRAM命令位置1。该位将在操作完成后由硬件自清零。
8. 将OTP_CMD_GO寄存器中的OTP_GO命令位置1以启动OTP_PROGRAM命令。该位将在操作完成后由硬件自清零。
9. 当操作挂起时，OTP_STATUS寄存器中的OTP_BUSY位置为有效。硬件将自动驱动OTP的输入。
10. 编程周期完成后，OTP_RD_DATA寄存器将更新以包含实际编程到OTP中的值。硬件通过在CPUMPEN置为无效后采样OTP的D端口来获取该值。更新该寄存器时，硬件应考虑位写操作和字节写操作。
11. 轮询OTP_STATUS寄存器，直到OTP_BUSY位清零为止。
12. 操作完成，OTP_BUSY位置为无效。
13. 从OTP_RD_DATA寄存器读取OTP_RD_DATA_FLD[7:0]并确定OTP是否成功编程。该值应与OTP_PRGM_DATA寄存器的内容匹配。如果使能位编程，则该寄存器中只有bit 0有效。否则，OTP_RD_DATA寄存器中的每个位对应OTP_PRGM_DATA寄存器中的相关位。
14. 如果某个存储单元编程失败，则必须重新编程。IP提供商建议，如果连续失败11次，则可判定该存储单元存在故障，表明OTP存在制造缺陷。

注： OTP字节的默认值为0x00。如果目标字节为0x00，则无需对其进行编程。

14.5 SMART OTP编程

SMART模式编程用于缩短编程时间，应当用于编程所有需设置为逻辑1的位。

1. 通过清零OTP_PWR_DN寄存器中的值OTP_PWRDN_N位使OTP退出掉电状态。
2. 通过将OTP_PGM_MODE_BYTE设置为0来选择位模式编程。
3. 编程各种脉冲宽度CSR。
4. 写入OTP_ADDR_HIGH/OTP_ADDR_LOW/OTP_ADDR_BITS和OTP_PRGM_DATA寄存器。
5. 向OTP_FUNC_CMD寄存器中的OTP_FW_SMART_PGM_EN位写入1。
6. 如果下一条PROGRAM/TESTDEC命令是SMART编程序列中针对已编程地址的第一条命令，则向OTP_FW_SMART_PGM_INIT_ACCESS位写入1。否则，该位应设置为0。
7. 将OTP功能命令寄存器中的OTP_PROGRAM命令位置1。该位将在操作完成后由硬件自清零。
8. 将OTP_CMD_GO寄存器中的OTP_GO命令位置1以启动PROGRAM命令。该位在操作完成后自清零。
9. 检查OTP_STATUS寄存器中的OTP_BUSY位是否置为无效。
10. 从OTP_RD_DATA寄存器读取OTP_RD_DATA_FLD[7:0]并确定OTP是否成功编程。该值应与OTP_PRGM_DATA寄存器的内容匹配。如果使能位编程，则该寄存器中只有bit 0有效。否则，OTP_RD_DATA寄存器中的每个位对应OTP_PRGM_DATA寄存器中的相关位。
11. 如果OTP_RD_DATA与OTP_PRGM_DATA匹配，则继续使用新地址进行编程操作。否则，重复执行上述步骤5及后续步骤（地址保持不变），重复次数不得超过SMART编程流程中提及的最大编程次数。

14.6 OTP存储器的外部回读

读命令从OTP内部DATA寄存器读取数据。该数据随后存入OTP_RD_DATA寄存器。下面给出了典型的读操作序列。硬件将在访问期间自动驱动READEN信号。

1. 通过清零OTP_PWR_DN寄存器中的值OTP_PWRDN_N位使OTP退出掉电状态。
2. 编程OTP_RSTB_PW_HIGH/OTP_RSTB_PW_LOW和OTP_READ_PW_HIGH/OTP_READ_PW_LOW寄存器。
3. 写入OTP_ADDR_HIGH/OTP_ADDR_LOW/OTP_ADDR_BITS寄存器。
4. 将OTP_FUNC_CMD寄存器中的OTP_READ命令位置1。该位将在命令被OTP控制器接受后由硬件自清零。
5. 将OTP_CMD_GO寄存器中的OTP_GO命令位置1以启动读命令。该位将在命令被OTP控制器接受后由硬件自清零。
6. 当操作挂起时，OTP_BUSY位置为有效。
7. 轮询OTP_STATUS寄存器，直到OTP_BUSY位清零为止。
8. 操作完成，OTP_BUSY位置为无效。
9. 读取OTP_RD_DATA寄存器。

始终按字节宽度进行读操作。OTP的读操作按字节边界寻址。OTP会忽略OTP_ADDR_BITS寄存器。在所有计划的操作完成后，应将OTP置于掉电状态。

14.7 通过Linux存储器映射器件进行OTP编程

在Linux上，PCI1xxx通过GPIO子系统驱动程序将OTP和EEPROM枚举为两个独立的8 kB磁盘。

这样，可以使用Linux dd命令对OTP和EEPROM进行编程。

该驱动程序还提供了一个自定义IOCTL接口，允许从任意偏移量开始编程可变长度的字节流。该驱动程序始终会枚举OTP，而EEPROM只有在电路板上实际存在时才会被枚举。

OTP被枚举为：

```
/dev/PCI1xxxxOTPn
```

EEPROM被枚举为：

```
/dev/PCI1xxxxE2Pn
```

要通过dd命令将配置文件（在本例中称为“input.bin”）编程到OTP，请参见以下示例：

```
sudo dd if=input.bin of=/dev/PCI1xxxxOTPn bs=512 count=16 oflag=direct
```

要通过dd命令回读OTP，请参见以下示例：

```
sudo dd if=/dev/PCI1xxxxOTPn of=output.bin bs=512 count=16 iflag=direct
```

15.0 EEPROM存储器编程（读写操作）

15.1 章节

- 第15.2节“引导时的EEPROM存储器装载”
- 第15.3节“通过Linux存储器映射器件进行EEPROM编程”

15.2 引导时的EEPROM存储器装载

在所有情况下，PCI1xxxx都会先从OTP（如果存在）读取配置。如果EEPROM可用且已配置，PCI1xxxx随后从EEPROM获取额外设置。

PCI1xxxx按照以下步骤读取EEPROM：

1. 检查第一个字节是否为MAGIC_BYTE。如果存在MAGIC_BYTE，则继续。否则，不配置EEPROM并跳过EEPROM装载。
2. 锁存 MEMORY_SIZE 并将其用作校验值，以确保硬件不会因配置不当而使读取范围超出已配置存储器的末端。当PCI1xxxx到达存储器末端时，将停止读取并假定配置已完成。
3. 锁存MEM_REGION_PROG以确定哪些存储器区域已编程。MEM_REGION_PROG中的每个位置1时都表示对应的存储器描述符已编程。
4. PCI1xxxx 遍历每个已编程（根据 MEM_REGION_PROG）的存储器描述符 MEM_REGION_DESRx。忽略其他存储器描述符。
5. PCI1xxxx读取MEM_TAG[7:0]存储器，该存储器指示要配置的子系统。PCI1xxxx会验证该子系统对于由CPN_SEL[2:0]定义的CPN是否有效，并通过检查GENERAL_SYS_CONFIG_REQ_REG的bit [7:0]确认子系统是否在请求配置，然后再继续配置子系统。如果子系统不需要配置，则PCI1xxxx转到步骤10。
6. 对于要配置的子系统，PCI1xxxx读取MEM_REGION_DESRx中的MEM_START_ADDRESS[12:0]存储器，该存储器指向配置的存储器区域。
7. PCI1xxxx读取存储在CFG_BYTE中的字节：

   7.1 如果该字节为0x00，则为STOP操作码。配置块中无更多数据可读。硬件转到步骤9。

   7.2 如果该字节为0x80，则为SET_MEMORY_ADDRESS操作码。PCI1xxxx将CFG_BYTE加1，然后读取MEMORY_ADDRESS并将其存入MEM_DWORD。然后，PCI1xxxx将CFG_BYTE加4。

   7.3 如果该字节为0x81至0xFF，则为SKIP_DWORDS操作码。如果MEMORY_ADDRESS = 0x0000_0000，表示SKIP_DWORDS操作码前面没有SET_MEMORY_ADDRESS操作码，硬件会针对EEPROM以及相应的MEM_TAG[7:0]（bit 15至bit 21）将GENERAL_SYS_OTP_CONFIG_ERR_REG中的相关错误位置1，然后转到步骤9。PCI1xxxx将MEM_DWORD中存储的值加4*(操作码-0x80)，并将CFG_BYTE加1。

   7.4 PCI1xxxx读取LENGTH。

   7.5 如果LENGTH = 0，则为错误。PCI1xxxx会将GENERAL_SYS_EEPROM_CONFIG_ERR_REG（对于EEPROM）中的相关错误位以及相应的MEM_TAG[7:0]（bit 0至bit 7）置1，然后转到步骤9。

   7.6 PCI1xxxx将CFG_BYTE加1并读取长度为LENGTH个DWORD的数据。

   7.7 PCI1xxxx将CFG_BYTE加4*LENGTH。

   7.8 PCI1xxxx将LENGTH个DWORD的数据写入MEMORY_ADDRESS。如果LENGTH = 1，硬件将使用MEMORY_ADDRESS的bit 0和bit 1作为字节选择器。

   7.9 PCI1xxxx将MEM_DWORD中存储的值加4*LENGTH。

   7.10 PCI1xxxx返回步骤7。
8. 只要读取的CFG_BYTE超出MEMORY_SIZE指示的存储器的末端，PCI1xxxx就会将GENERAL_SYS_EEPROM_CONFIG_ERR_REG.EEPROM_MEM_OVERSHOOT_ERR位置1，并立即转到步骤9。
9. 如果还有更多子系统需要配置，PCI1xxxx返回步骤6。
10. 如果不存在指示需通过SPI/SMBus进行配置的配置脚，PCI1xxx将继续引导过程并激活硬件子系统。

15.3 通过Linux存储器映射器件进行EEPROM编程

在Linux上，PCI1xxx通过GPIO子系统驱动程序将OTP和EEPROM枚举为两个独立的8 kB磁盘。

这样，可以使用Linux dd命令对OTP和EEPROM进行编程。

该驱动程序还提供了一个自定义IOCTL接口，允许从任意偏移量开始编程可变长度的字节流。该驱动程序始终会枚举OTP，而EEPROM只有在电路板上实际存在时才会被枚举。

OTP被枚举为：

```
/dev/PCI1xxxxOTPn
```

EEPROM被枚举为：

```
/dev/PCI1xxxxE2Pn
```

要通过dd命令将配置文件（在本例中称为“input.bin”）编程到EEPROM，请参见以下示例：

```
sudo dd if=input.bin of=/dev/PCI1xxxxE2Pn bs=512 count=16 oflag=direct
```

要通过dd命令回读EEPROM，请参见以下示例：

```
sudo dd if=/dev/PCI1xxxxE2Pn of=output.bin bs=512 count=16 iflag=direct
```

16.0 SMBUS/I²C目标配置

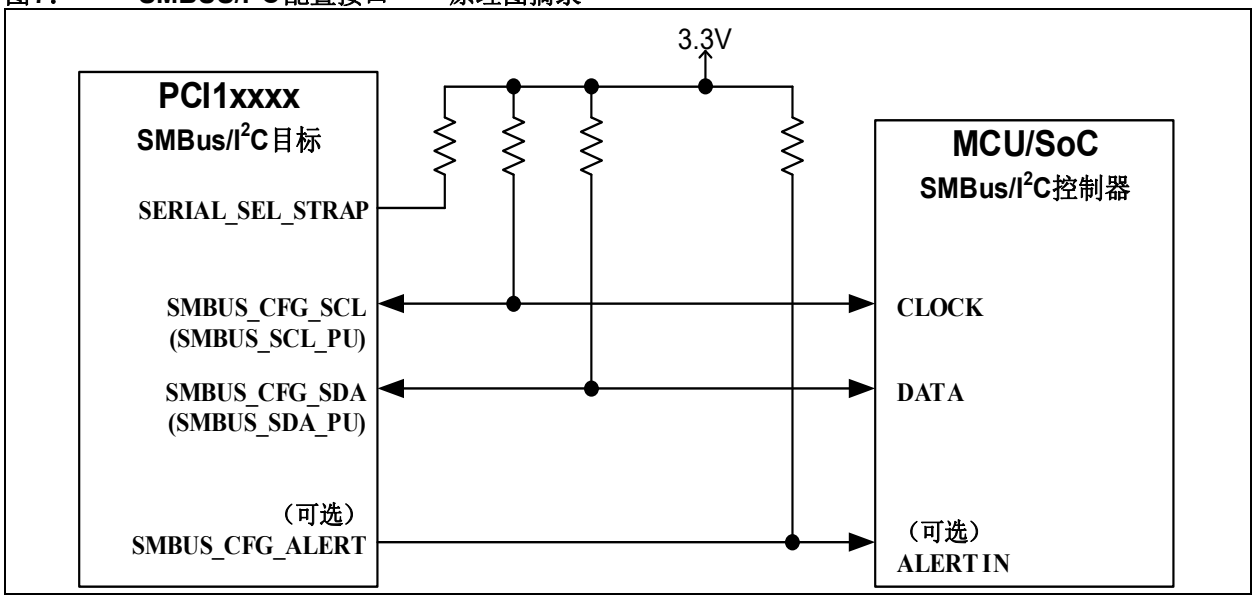
16.1 章节

16.2 PCI1XXXX的SMBus/I²C配置

若要对PIC1XXXX进行SMBus/I²C配置，必须正确配置硬件配置脚。

有关正确的硬件配置脚和接口处理，请参见图7。

图7： SMBUS/I²C配置接口——原理图摘录



这些硬件配置脚因器件而异。有关不同器件的引脚分配，请参见表156。

表156： 使能SPI配置接口所需的硬件配置脚——不同器件的引脚映射

器件	SERIAL SEL_STRAP	SMBUS_SCL_PU	SMBUS_SDA_PU
PCI12000	引脚24（PROG33）	引脚38（PROG64）	引脚39（PROG65）
PCI11010	引脚64（PROG68）	引脚60（PROG64）	引脚61（PROG65）
PCI11101	引脚78（PROG68）	引脚74（PROG64）	引脚75（PROG65）
PCI11400	引脚70（PROG68）	引脚66（PROG64）	引脚67（PROG65）
PCI11414	引脚93（PROG68）	引脚89（PROG64）	引脚90（PROG65）

SMBus/I²C接口引脚映射因器件而异。有关不同器件的引脚分配，请参见表157。

表157： 使能SPI配置接口所需的SMBus/I²C接口——不同器件的引脚映射

器件	SMBUS_CFG_SCL	SMBUS_CFG_SCL	SMBUS_CFG_ALERT_N
PCI12000	引脚38（PROG64）	引脚39（PROG65）	引脚40（PROG66）
PCI11010	引脚60（PROG64）	引脚61（PROG65）	引脚62（PROG66）
PCI11101	引脚74（PROG64）	引脚75（PROG65）	引脚76（PROG66）
PCI11400	引脚66（PROG64）	引脚67（PROG65）	引脚68（PROG66）
PCI11414	引脚89（PROG64）	引脚90（PROG65）	引脚91（PROG66）

16.3 SMBus/I²C报警配置

SMBUS_CFG_ALERT_N引脚为可选报警功能，使用前必须专门使能并配置。

SMBUS_CFG_ALERT_N与SMBus报警响应地址（ARA）一起使用。经过配置后，PCI1xxxx通过SMBUS_SLV_ALERT_N引脚提供该信号作为SMBus目标接口的输出。

PCI1xxxx可通过SMBUS_SLV_ALERT_N通知控制器其有更新需要传达给控制器。SMBus/I²C控制器处理中断，并通过报警响应地址同时访问所有SMBUS_SLV_ALERT_N器件。

只有将SMBALERT#拉为低电平的器件才会应答报警响应地址。主机执行修改后的接收字节操作。

目标发送器件提供的7位器件地址放入字节的高7位中。第8位可以是0或1。

在下列情况下，应将SMBUS_SLV_ALERT_N引脚拉为低电平：

注 1： 任何EXT_SYS_CONFIG_REQ_REG.EXT_*CONFIG_REQ位置1（从0跳变为1）。

2： 任何EXT_SYS_CONFIG_REQ_REG.EXT_*CONFIG_REQ位为1且相应的模块变为未就绪（GENERAL_SYS_READY_REG.*_RDY从0跳变为1）。

当I2C_ALERT_SC位设置为1时，SMBUS_CFG_ALERT_N引脚（置为有效时）将在收到控制器对其地址的应答（ACK）响应后自动清零。

当I2C_ALERT_SC位设置为0时，SMBUS_CFG_ALERT_N引脚（置为有效时）必须通过主机软件向I2C_SLV_ALERT写入1来手动清零。

16.4 SMBus/I²C指令

SMBus/I²C为双线数据协议。发送数据的器件被定义为发送器，接收数据的器件被定义为接收器。总线由控制器控制，控制器可生成SCL时钟、控制总线访问并产生启动和停止条件。控制器和目标都可以作为发送器或接收器，具体由控制器决定。
时钟（SCL）和数据（SDA）信号都具有数字输入滤波器，可抑制小于100 ns的脉冲。

存在以下总线状态：

- **空闲：**当总线空闲时，SDA和SCL均为高电平。
- **启动和停止条件：**启动条件被定义为SCL为高电平时SDA线从高电平变为低电平。停止条件被定义为SCL为高电平时SDA线从低电平变为高电平。在启动条件之后将总线视为繁忙状态；在停止条件之后将总线视为空闲4.7 µs/1.3 µs（分别对应于100 kHz和400 kHz操作）。在重复启动条件（而非停止条件）后，总线保持繁忙状态。启动和重复启动在功能上是相同的。
- **数据有效：**在启动条件之后，当SCL为高电平时并且SDA保持稳定时数据有效。只能在时钟为低电平时更改数据。每个时钟脉冲都有一个有效位。每个字节必须为8位长，且先发送MSb。

- **应答：**数据的每个字节后都有一个应答位。控制器生成第九个时钟脉冲作为应答位。发送器释放SDA（高电平）。接收器将SDA驱动为低电平，使其在时钟的高电平周期内保持有效，其中包括建立和保持时间。接收器可为控制器，也可为目标，具体取决于数据的方向。通常，接收器会应答每个字节。如果控制器为接收器，则不会在传输最后一个字节后产生应答。这将通知目标不要驱动下一个数据字节，以便控制器能够生成停止或重复启动条件。

16.5 SMBus/I²C目标操作

在SMBus/I²C管理模式下，SMBus/I²C目标接口用于器件的CPU/MCU/SoC管理。在这些模式下，CPU/MCU/SoC可访问所有器件寄存器。SMBus/I²C目标控制器可实现低电平SMBus/I²C目标串行接口（启动和停止条件检测、数据位发送和接收以及应答产生和接收）、处理目标命令协议并执行系统寄存器读写操作。SMBus/I²C目标控制器符合SMBus/I²C规范。

SMBus/I²C目标串行接口由一个数据线（SMBUS_CFG_SDA）和一个串行时钟（SMBUS_CFG_SCL）组成。串行时钟由控制器驱动，而数据线是双向的。两个信号均为漏极开路信号，需要外部上拉电阻。

SMBus/I²C目标串行接口支持标准模式速度（最高100 kHz）和快速模式速度（400 kHz）。

有关时序的详细信息，请参见SMBus/I²C规范，但需注意以下变化：

- $t_{VD;DAT}$最大值（从SCL下降到SDA数据输出有效的时间）在标准模式和快速模式下分别为3000 ns和700 ns。
- $t_{VD;ACK}$最大值（从SCL下降到SDA应答输出有效的时间）在标准模式和快速模式下分别为3000 ns和700 ns。
- t_{SP}最大值（SCL和SDA上的输入尖峰抑制）为100 ns。

注：	可通过I2C_SLV_CONFIG_REG.I2C_PULSE_FILTER[3:0]位域编程最小脉冲滤波器宽度。

- $t_{HD;DAT}$最小值（SCL下降后的SDA数据和应答输出保持的时间）为100 ns。

注：	$t_{HD;DAT}$参数比I²C规范中的要求大100 ns。t_{HD}参数从SCL的$V_{IL(max)}$点开始算起。SCL的下降时间（从$V_{IH(min)}$到$V_{IL(max)}$）可能为300 ns。由于SCL低于$V_{IH(min)}$时SDA输出可能在任何时刻发生变化，因此它有可能在SCL为有效低电平之前就发生变化。为了对此进行补偿，器件必须提供额外的300 ns保持时间（总共400 ns）以满足所需的100 ns规范要求。

16.5.1 SMBUS/I²C目标命令格式

SMBus/I²C目标串行接口支持单寄存器和多寄存器读写命令。控制器先发送一个启动条件，然后发送控制字节，以此来启动读命令或写命令。控制字节由一个7位目标地址和一个1位读/写指示（R/~W）组成。器件使用的默认目标地址为0001010b，以SA6（线路上的第一位）至SA0（线路上的最后一位）的格式写入。或者，也可将I2C_CFG_CONFIG_REG.I2C_CFG_ADDR[6:0]位域配置为所需值，从而将SMBus/I²C目标地址配置为其他地址。假设控制字节中的目标地址与该地址匹配，器件将会应答控制字节。否则，在下一个启动条件之前，整个序列都将被忽略。

如果控制字节中的读/写指示（R/~W）为0（指示可能的写操作），则控制器随后发送的32位为寄存器地址。在目标确认地址DWORD后，控制器既可以发送要写入的4个数据字节，也可以发送另一个启动条件（以开始读取数据）或停止条件。后两种情况将终止当前写操作（未写入任何数据），但会影响用于后续读取的内部寄存器地址的设置。如果控制字节中的读/写指示为1（指示读操作），则目标将在控制字节应答之后开始发送数据。

在器件初始化到各种配置输入均有效的程度之前，SMBus/I²C目标接口不得响应任何外部引脚活动，也不会受其影响。如果器件初始化在有效周期内完成，应忽略周期的尾端。内部寄存器不应受到影响，并且I²C目标接口的状态不能发生变化。

16.5.1.1 通过读取轮询的方式确认SMBus/I²C目标是否完成初始化

在器件初始化之前，SMBus/I²C接口不会返回有效数据。要确定I²C目标接口何时正常工作，应轮询字节顺序测试寄存器（BYTE_TEST_REG）。一旦读取到正确模式，即可认为接口正常工作。

注： 轮询BYTE_TEST_REG寄存器时，SMBus/I²C控制器应仅使用单次寄存器读操作（每个启动/停止条件对应一个数据周期）。

16.5.2 功耗管理期间和功耗管理之后的访问

在低功耗模式下，将忽略读写操作。SMBus/I²C目标接口将不会响应任何外部引脚活动，也不会受其影响。要确定I²C目标接口何时正常工作，应轮询字节顺序测试寄存器（BYTE_TEST_REG）。一旦读取到正确模式，即可认为接口正常工作。

注： 轮询BYTE_TEST_REG寄存器时，SMBus/I²C控制器应仅使用单次寄存器读操作（每个启动/停止条件对应一个数据周期）。

16.5.3 读命令详细信息

器件寻址后，当控制器发送一个启动条件和R/~W位置1的控制字节时，会从目标读取寄存器。假设控制字节中的地址与目标地址匹配，则控制字节会由目标应答。否则，在下一个启动条件之前，整个序列都将被忽略。应答后，器件将发送4个字节的数据。控制器将对前3个字节进行应答，而在收到第4个字节时，控制器将发送无应答信号，后跟停止条件。无应答信号用于通知器件不要发送后续的4个字节（与多次读操作类似）。在单次读操作后，内部寄存器地址保持不变。

如果控制器在收到第4个字节时发送应答信号，则将执行多次读操作。内部地址将递增，并将移出下一个寄存器的内容。当内部地址达到其最大值后，将计满返回0。

当控制器发送无应答信号后跟停止条件时，多次读操作结束。无应答信号用于通知目标不要发送后续的4个字节。每次读操作（包括最后一次）都会递增内部寄存器地址。

无论是单次读操作还是多次读操作，只要控制器在收到寄存器的前三个字节后发送无应答信号，器件就会停止发送后续字节。如果控制器发送意外的启动或停止条件，则目标将立即停止发送，并根据需要响应下一个序列。

SMBus/I²C读取未使用的寄存器地址时将返回全零。

例1： 读命令示例

表158给出了读取地址0000_0120h处的BYTE_TEST_REG寄存器的示例。在本例中，返回了预期值8765_4321h。

注： 由于寄存器读操作始终以4字节的块执行，因此丢弃寄存器地址的最后两位，仅使用bit 9:2。最高两位同样不是必需的，因为PCI1xxxx的总寄存器空间没有使用这两位。在本例中，目标地址为120h = 0001_0010_0000b。丢弃最高两位和最低两位后，最终值为01001000b。

表158： SMBus/I²C寄存器写操作示例

操作	控制	写控制字节（含目标SMBus/I²C地址）								状态	寄存器地址字节（Bit 9:2）								状态
位	START	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	WRITE	ACK	REG ADDR9	REG ADDR8	REG ADDR7	REG ADDR6	REG ADDR5	REG ADDR4	REG ADDR3	REG ADDR2	ACK
控制器	S	0	0	0	1	0	1	0	0	z	0	1	0	0	1	0	0	0	
目标	z	z	z	z	z	z	z	z	z	0	z	z	z	z	z	z	z	z	0

操作	控制	读控制字节（含目标SMBus/I²C地址）								状态	数据字节（Bit 31:24）								状态
位	START	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	READ	ACK	D31	D30	D29	D28	D27	D26	D25	D24	ACK
控制器	S	0	0	0	1	0	1	0	1	z	z	z	z	z	z	z	z	z	0
目标	z	z	z	z	z	z	z	z	z	0	1	0	0	0	0	1	1	1	z

操作	数据字节（Bit 23:16）								状态	数据字节（Bit 15:8）								状态
位	D23	D22	D21	D20	D19	D18	D17	D16	ACK	D15	D14	D13	D12	D11	D10	D9	D8	ACK
控制器	z	z	z	z	z	z	z	z	0	z	z	z	z	z	z	z	z	0
目标	0	1	1	0	0	1	0	1	z	0	1	0	0	0	0	1	1	z

操作	数据字节（Bit 7:0）								状态	控制
位	D7	D6	D5	D4	D3	D2	D1	D0	NOT ACK	STOP
控制器	z	z	z	z	z	z	z	z	z	P
目标	0	0	1	0	0	0	0	1	z	z

16.5.4 写命令详细信息

目标寻址后，当控制器继续发送数据字节时，会将寄存器的内容写入器件。目标将应答每个字节。在序列的第4个字节后，控制器可发送另一个启动条件或用停止条件暂停序列。在单次写操作后，内部寄存器地址保持不变。如果控制器在第4个应答信号之后发送额外的字节，则将执行多次写操作。内部地址将自动递增，并会写入下一个寄存器。内部地址达到其最大值后，会计满返回0。当控制器发送另一个启动或停止条件时，多次写操作将结束。每次写操作（包括最后一次）都会递增内部寄存器地址。这与后续写操作无关，因为新的写周期会包含新的寄存器地址。但是，如果不先复位寄存器地址而直接用于读操作，则会影响内部寄存器地址。

对于单次和多次写操作，如果控制器发送意外启动或停止条件，则器件将立即停止，并根据需要响应下一个序列。

寄存器的数据写操作在32位数据输入之后发生。如果未写入32位（控制器意外发送启动或停止条件），则会认为写操作无效，寄存器不受影响。可在一个多次写周期中写入多个寄存器，每个寄存器在其对应的32位数据输入之后写入。对于未使用的寄存器地址，不能执行SMBus/I2C写操作。

例2： 写命令示例

表159给出了写入地址058h处的GENERAL_SYS_CONFIG_DONE_REG寄存器的示例。在本例中，每个硬件组件的配置更改标志都设置为1，以确保每个系统组件的配置更改都能生效。

注： 由于寄存器写操作始终以4字节的块执行，因此丢弃寄存器地址的最后两位，仅使用bit 9:2。最高两位同样不是必需的，因为PCI1xxxx的总寄存器空间没有使用这两位。在本例中，目标地址为058h = 0000_0101_0100b。丢弃最高两位和最低两位后，最终值为00010101b。

表159： SMBus/I2C寄存器写操作示例

操作	控制	写控制字节（含目标SMBus/I2C地址）								状态	寄存器地址字节（Bit 9:2）								状态
位	START	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	WRITE	ACK	REG ADDR9	REG ADDR8	REG ADDR7	REG ADDR6	REG ADDR5	REG ADDR4	REG ADDR3	REG ADDR2	ACK
控制器	S	0	0	0	1	0	1	0	0	z	0	0	0	1	0	1	0	1	
目标	z	z	z	z	z	z	z	z	z	0	z	z	z	z	z	z	z	z	0

操作	数据字节（Bit 31:24）								状态	数据字节（Bit 23:16）								状态
位	D31	D30	D29	D28	D27	D26	D25	D24	ACK	D23	D22	D21	D20	D19	D18	D17	D16	ACK
控制器	0	0	0	0	0	0	0	1	z	0	0	1	1	1	1	1	1	z
目标	z	z	z	z	z	z	z	z	0	z	z	z	z	z	z	z	z	0

操作	数据字节（Bit 15:8）								状态	数据字节（Bit 7:0）								状态	控制
位	D15	D14	D13	D12	D11	D10	D9	D8	ACK	D7	D6	D5	D4	D3	D2	D1	D0	ACK	STOP
控制器	0	0	1	1	1	1	1	1	z	0	0	1	1	1	1	1	1	z	P
目标	z	z	z	z	z	z	z	z	0	z	z	z	z	z	z	z	z	0	z

16.6 配置完成

如果对PCI1XXXX进行I2C配置，则将在配置阶段无限期等待，直到EXT_SYS_CONFIG_DONE_REG位全部置1为止。

17.0 SPI配置

17.1 章节

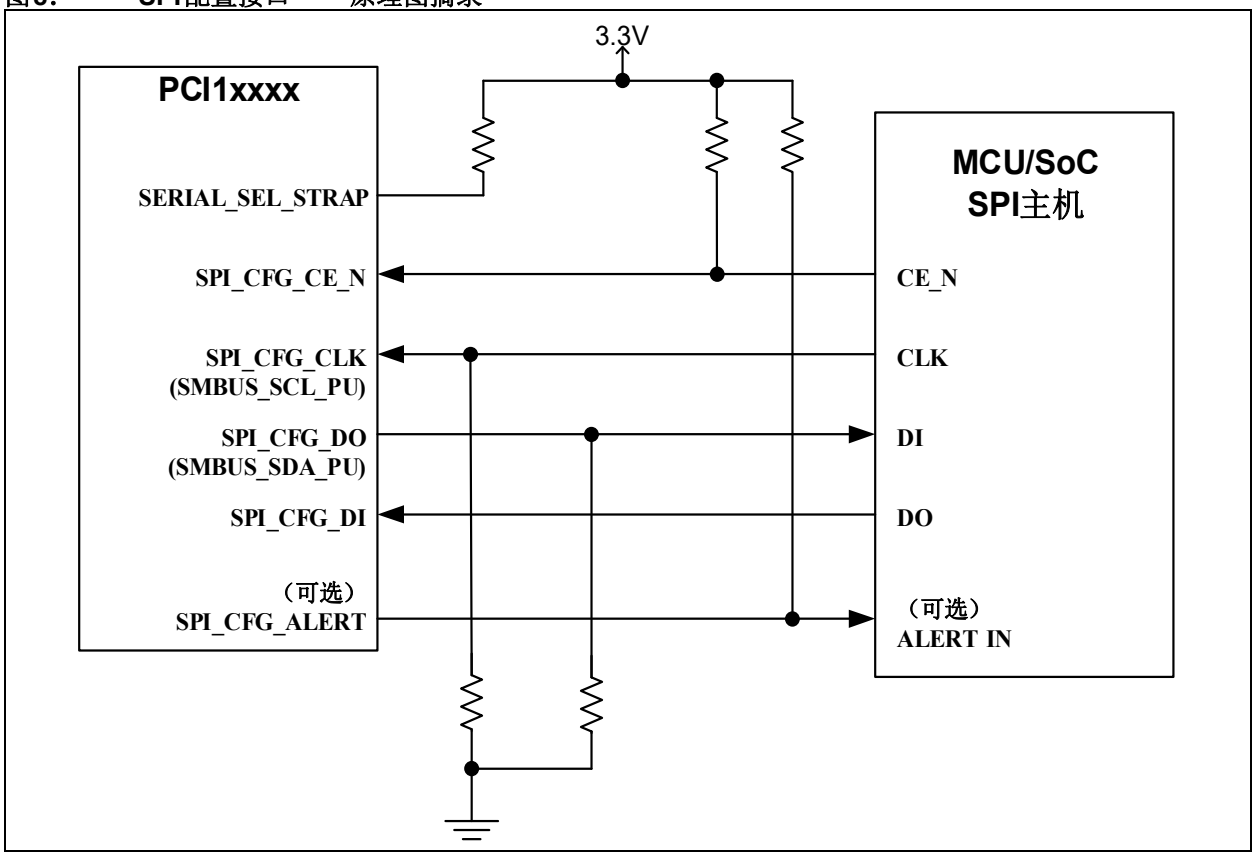
- 第17.2节“PCI1XXXX的SPI配置”
- 第17.3节“SPI报警配置”
- 第17.4节“SPI指令”
- 第17.5节“配置完成”

17.2 PCI1XXXX的SPI配置

若要对PIC1XXXX进行SPI配置，必须正确配置硬件配置脚。

有关正确的硬件配置脚和接口处理，请参见图8。

图8： SPI配置接口——原理图摘录



这些硬件配置脚因器件而异。有关不同器件的引脚分配，请参见表160。

表160： 使能SPI配置接口所需的硬件配置脚——不同器件的引脚映射

器件	SERIAL SEL_STRAP	SMBUS_SCL_PU	SMBUS_SDA_PU
PCI12000	引脚24（PROG33）	引脚38（PROG64）	引脚39（PROG65）
PCI11010	引脚64（PROG68）	引脚60（PROG64）	引脚61（PROG65）
PCI11101	引脚78（PROG68）	引脚74（PROG64）	引脚75（PROG65）
PCI11400	引脚70（PROG68）	引脚66（PROG64）	引脚67（PROG65）
PCI11414	引脚93（PROG68）	引脚89（PROG64）	引脚90（PROG65）

SPI接口引脚映射因器件而异。有关不同器件的引脚分配，请参见表161。

表161： 使能SPI配置接口所需的SPI接口——不同器件的引脚映射

器件	SPI_CFG_CLK	SPI_CFG_DO	SPI_CFG_DI	SPI_CFG_CE_N	SPI_CFG_ALERT_N
PCI12000	引脚38（PROG64）	引脚39（PROG65）	引脚40（PROG66）	引脚41（PROG67）	可配置
PCI11010	引脚60（PROG64）	引脚61（PROG65）	引脚62（PROG66）	引脚63（PROG67）	可配置
PCI11101	引脚74（PROG64）	引脚75（PROG65）	引脚76（PROG66）	引脚78（PROG67）	可配置
PCI11400	引脚66（PROG64）	引脚67（PROG65）	引脚68（PROG66）	引脚69（PROG67）	可配置
PCI11414	引脚89（PROG64）	引脚90（PROG65）	引脚91（PROG66）	引脚92（PROG67）	可配置

17.3 SPI报警配置

SPI_CFG_ALERT_N引脚是为可选报警功能，必须映射到可用的PROGx引脚。该引脚可映射到表162中列出的任何备用PROGx引脚。

表162： 不同器件的SPI配置接口报警引脚映射

器件	PROG0	PROG1	PROG2	PROG3	PROG4	PROG5	PROG6	PROG7	PROG8	PROG9	PROG10	PROG17	PROG18	PROG19	PROG20	PROG21	PROG22	PROG23	PROG24
PCI12000																			
PCI11010	X	X	X	X								X	X						
PCI11101	X	X	X		X	X	X					X	X	X	X	X			
PCI11400	X	X	X	X	X	X	X	X				X	X	X	X				
PCI11414	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

器件	PROG46	PROG47	PROG48	PROG49	PROG50	PROG51	PROG67	PROG68	PROG73	PROG79	PROG81	PROG86	PROG87
PCI12000							X	X					
PCI11010	X	X	X	X	X	X	X	X		X	X		
PCI11101	X	X	X	X	X	X	X	X	X	X	X		
PCI11400	X	X	X	X			X	X		X	X		
PCI11414	X	X	X	X	X	X	X	X		X	X	X	X

经过专门配置后，PCI1xxxx通过SPI_CFG_ALERT_N引脚提供该信号作为SPI配置通道的输出。
在下列情况下，应将SPI_CFG_ALERT_N引脚拉为低电平：

- 任何EXT_SYS_CONFIG_REQ_REG.EXT_*CONFIG_REQ位置1（从0跳变为1）。
- 任何EXT_SYS_CONFIG_REQ_REG.EXT_*CONFIG_REQ位为1且相应的模块变为未就绪（GENERAL_SYS_READY_REG.*_RDY从0跳变为1）。

如果SPI_CFG_CONFIG_REG.SPI_ALERT_SC位设置为1，当SPI_CFG_CE_N引脚从高电平变为低电平时，SPI_CFG_CE_N引脚（置为有效时）将自动清零。在SPI配置主机将SPI_CFG_CE_N引脚置为有效（从0变为1）后，目标器件必须停止将SMBUS_CFG_ALERT_N引脚拉为低电平。

当SPI_CFG_CONFIG_REG.SPI_ALERT_SC位设置为0时，SPI_CFG_CE_N引脚（置为有效时）必须通过主机软件向SPI_CFG_STATUS_REG.SPI_CFG_ALERT写入1来手动清零。当主机软件将SPI_CFG_STATUS_REG.SPI_CFG_ALERT清零后，目标器件必须停止将SMBUS_CFG_ALERT_N引脚拉为低电平。

17.4 SPI指令

SPI接口仅支持基本的读命令和写命令。

在PCI1xxxx初始化之前，SPI接口不会返回有效数据。要确定SPI接口何时正常工作，应轮询字节顺序测试寄存器（BYTE_TEST_REG）。一旦读取到正确模式，即可认为接口正常工作。在此轮询期间，仅使用单次寄存器读操作。

SPI_CFG_DI引脚上的输入数据在SCK输入时钟的上升沿采样。输出数据在时钟的下降沿通过SPI_CFG_DO引脚提供。

SPI接口仅支持基本的读命令和写命令，请参见表163。

表163： 支持SPI指令

指令	代码	字节			最大频率
		地址	空	数据	
读	03h	4	2	4+	30 MHz
写	02h	4	0	4+	80 MHz

注： 不支持下列命令：FASTREAD、SDOR、SDIOR、SQOR、SQIOR、SDDW、SDADW、SQDW和SQADW。

17.4.1 读命令详细信息

READ指令每个时钟输入一位指令代码和地址字节，每个时钟输出一位数据。仅当时钟频率达30 MHz时，SPI总线协议才支持该指令。

首先将SPI_CFG_CE_N置为有效，选择SPI目标接口。然后向SPI_CFG_DO引脚中输入8位READ指令（03h），接着是四个地址字节和两个空字节。地址字节用于指定器件内的字节地址。两个空字节用于延长等待状态，以对应内部寄存器的最长读取时间，即512 ns（16*33 ns）。

在最后一个空位的上升沿之后的时钟下降沿，SPI_CFG_DI引脚从所选寄存器的最低有效字节（Least-Significant Byte，LSB）的最高有效位（Most-Significant Bit，MSb）开始驱动。剩余的寄存器位在随后的时钟下降沿移出。SPI_CFG_CE_N输入变为无效以结束周期。此时，SPI_CFG_DI引脚处于三态（高阻态）。

例3： SPI读命令示例

表164给出了读取地址0000_0120h处的BYTE_TEST_REG寄存器的示例。在本例中，返回了预期值8765_4321h。

表164： SPI单次寄存器读操作示例

名称	指令	地址				空字节		返回数据			
字节	1	2	3	4	5	6	7	8	9	10	11
SI	02h	00h	00h	01h	20h	00h	00h	x	x	x	x
SO	z	z	z	z	z	z	z	21	43	65	87

17.4.2 写命令详细信息

WRITE指令输入指令代码、地址和数据字节，每个时钟一位。当时钟频率达30 MHz时，SPI总线协议支持该指令。

首先将SPI_CFG_CE_N置为有效，选择SPI目标接口。对于SPI模式，8位WRITE指令（02h）输入SI/SIO[0]引脚，后跟4个地址字节。地址字节用于指定器件内的字节地址。

数据跟在地址字节之后。数据从LSB的MSb开始输入SPI_CFG_DI引脚。寄存器的数据写操作在32位数据输入之后发生。如果SPI_CFG_CE_N恢复高电平时未写入32位，则会认为写操作无效，寄存器不受影响。

SPI_CFG_CE_N输入变为无效以结束周期。

例4： SPI写命令示例

表165给出了写入地址0058h处的GENERAL_SYS_CONFIG_DONE_REG寄存器的示例。在本例中，每个硬件组件的配置更改标志都设置为1，以确保每个系统组件的配置更改都能生效。

表165： SPI单次寄存器写操作示例

名称	指令	地址				空字节		写入数据			
字节	1	2	3	4	5	6	7	8	9	10	11
SI	02h	00h	00h	00h	58h	00h	00h	3F	3F	3F	01
SO	z	z	z	z	z	z	z	z	z	z	z

17.5 配置完成

如果对PCI1xxxx进行SPI配置，则将在配置阶段无限期等待，直到EXT_SYS_CONFIG_DONE_REG位全部置1为止。有关写入该寄存器的示例，请参见表165。

18.0 运行时配置

在运行时期间，系统可能在有限的寄存器集内异步重新配置PCI1xxxx。运行时可能重新配置的通用功能包括：

- 以太网MAC地址
- 以太网PHY配置参数（如全/半双工和自动协商）
- USB端口禁止/使能
- UART速度（波特率）
- I^2C/SPI速度

必须通过将相应的运行时置为有效来指示重新配置已完成。可通过下列三种途径管理配置：

- PCIe主机
- I^2C配置通道
- SPI配置通道

附录A： 系统锁定寄存器（SYS_LOCK_REG）

PCI1XXXX允许多个驱动程序从不同的PCIe存储空间访问公共系统寄存器。SYS_LOCK寄存器提供了一种安全机制，可防止因不同驱动程序的多重并发访问而导致冲突。

大多数PCI1XXXX器件用户不需要了解SYS_LOCK_REG寄存器及其操作，但在某些调试场景（例如，在未装载驱动程序的情况下测试硬件）可能需要理解并手动访问这些寄存器。

A.1 SYS_LOCK_REG的驱动程序使用模型

1. 读取SYS_LOCK_REG。
2. 如果任一位置1，等待一段时间后返回步骤1。
3. 如果没有其他驱动程序锁定系统组件，则向SYS_LOCK_REG的相应功能位写入1b。
4. 读取SYS_LOCK_REG以确保已锁定。
5. 如果预期的锁定位未置1，则再次返回步骤1（其他驱动程序在此之前已锁定组件，因此锁定失败）。
6. 如果预期的锁定位已置1，则表示已锁定。对系统寄存器执行任何所需操作，直到不再需要访问或软件处于即使其他软件介入也不会引发问题的状态。
7. 向SYS_LOCK_REG的相应功能位写入0b（解锁）。

小心：	如果不按照上述步骤，将导致系统寄存器写操作被忽略，系统寄存器读操作始终返回0x00000000。驱动程序的作用是防止因未清零位而导致锁定。

A.2 硬件操作

SYS_LOCK_REG遵循以下规则：

- SYS_LOCK_REG寄存器的每个位只能从其对应功能的存储空间写入（置1（锁定）或清零（解锁））。
- 如果SYS_LOCK_REG的一个位已被某个功能置1，则其他功能无法将其对应的位置1，直到该位清零为止。
- SYS_LOCK_REG寄存器始终可供任何功能读取，无论其对应的位是否置1。
- 若要通过某个功能的存储空间访问系统寄存器（SYS_LOCK_REG除外），该功能必须已预先锁定。如果没有预先锁定，则该功能对系统寄存器（SYS_LOCK_REG除外）的访问将按如下方式处理：
  - 将丢弃所有写操作
  - 任何读操作都将返回0x00000000。
- 在正确配置器件的通用PCI配置空间的COMMAND字节（不保持默认值0x0000）之前，无法写入SYS_LOCK_REG。通常由正常运行的驱动程序来实现，在无驱动程序的情况下则需要手动设置。

A.3 SYS_LOCK_REG寄存器定义

SYS_LOCK_REG可从任何PCIe功能的存储空间访问（即：USB子系统、以太网系统或外设子系统之一），请参见表166和表167。

若要通过PCI寄存器访问定位该寄存器，必须先通过读取器件配置空间来定位所选外设的BAR0地址（见**附录B：“PCI配置空间”**）。

表166： SYS_LOCK_REG

基址		外设基址		SYS_LOCK 偏移量
获取所选地址的BAR0	+	USB_SYSTEM _REG_ADDR_BASE = 0x0001_2000 ENET_SYSTEM_REG_ADDR_BASE = 0x0000_4000 PERI_UART_SYSTEM_REG_ADDR_BASE = 0x0000_1000 PERI_SMBUS_SYSTEM_REG_ADDR_BASE = 0x0001_1000 PERI_SPI_SYSTEM_REG_ADDR_BASE = 0x0002_1000 PERI_GEN_SYSTEM_REG_ADDR_BASE = 0x0003_1000	+	0x0A0

表167： SYS_LOCK_REG

SYS_LOCK_REG 偏移量：0x0A0			系统锁定寄存器 默认值 = 0x0000_0000
BIT	**名称**	**R/W**	**说明**
31:8	Reserved	R	始终读为0
7	MAIN_LOCK	R/W	表示通过SYSTEM_REG_ADDR_BASE实现锁定的位 0b = 未锁定 1b = 已锁定
6	Reserved	R	始终读为0
5	GEN_PERI_LOCK	R/W	表示通过GENERAL_PERI_ADDR_BASE实现锁定的位 0b = 未锁定 1b = 已锁定
4	SPI_PERI_LOCK	R/W	表示通过SPI_PERI_ADDR_BASE实现锁定的位 0b = 未锁定 1b = 已锁定
3	SMBUS_PERI_LOCK	R/W	表示通过SMBUS_PERI_ADDR_BASE实现锁定的位 0b = 未锁定 1b = 已锁定
2	UAR_PERI_LOCK	R/W	表示通过UART_PERI_ADDR_BASE实现锁定的位 0b = 未锁定 1b = 已锁定
1	ENET_SS_LOCK	R/W	表示通过ENET_SUBSYSTEM_ADDR_BASE实现锁定的位 0b = 未锁定 1b = 已锁定
0	USB_SS_LOCK	R/W	表示通过USB_SUBSYSTEM_ADDR_BASE实现锁定的位 0b = 未锁定 1b = 已锁定

附录B： PCI配置空间

除了主机总线桥之外，所有PCI器件都提供256字节的配置寄存器（称为配置地址空间）。这样，可从软件层面完全初始化和配置每个器件。

配置空间存储器映射因器件类型而异。PCI1xxxx上除PCIe交换芯片外的所有可用外设都使用头类型0x0。

配置空间的映射如表168所示。有关低级位的详细信息，请参见相关PCI规范。字节顺序始终使用小尾数法。对于占用多个相邻字节的位域，其最低有效值位于低地址。

表168： 配置空间存储器映射

偏移量	Bit [31:24]	Bit [23:16]	Bit [15:8]	Bit [7:0]
0x00	器件ID		供应商ID	
0x04	状态		命令	
0x08	分类代码	子类	编程接口	版本ID
0x0C	BIST	头类型	延时定时器	高速缓存大小
0x10	BAR0（基址0）			
0x14	BAR1			
0x18	BAR2			
0x1C	BAR3			
0x20	BAR4			
0x24	BAR5			
0x28	Cardbus CIS指针			
0x2C	子系统器件ID		子系统供应商ID	
0x30	扩展ROM基址			
0x34	保留			功能指针
0x38	保留			
0x3C	最大延时	最小授权	中断引脚	中断线

B.1 Linux中的配置空间读访问

“lspci”是一种灵活的实用程序，用于显示PCI总线上所有器件的信息。使用该工具可以轻松读取PCI总线上所有器件的配置空间。若要读取PCI总线上所有器件的配置空间，可在控制台终端中输入以下命令：

```
lspci -xxx
```

该命令将返回PCI总线上所有器件的完整配置空间转储，并将在终端中显示为如下形式：

```
29:00.0 Serial controller: Microchip Technology / SMSC Device a042 (rev a0)
00: 55 10 42 a0 06 04 10 00 a0 02 00 07 10 00 80 00
10: 04 80 01 fc 00 00 00 00 04 70 02 fc 00 00 00 00
20: 04 60 02 fc 00 00 00 00 00 00 00 00 55 10 42 a0
30: 00 00 00 00 40 00 00 00 00 00 00 00 0b 01 00 00
```

此外，还可使用“-d [<vendor>]:[<device>]”选项筛选仅匹配Microchip供应商ID的器件：

```
lspci -xxx -d 1055:*
```

B.2 Linux中的配置空间写访问

“setpci”实用程序用于读取和设置PCI器件配置空间。可通过domain:bus:slot.function或供应商ID和器件ID来选择器件：

```
setpci -s [[[[<domain>]:]<bus>]:][<slot>][.[<func>]]


setpci -d [<vendor>]:[<device>]
```

最后，可选择存储器偏移量和要在配置空间中设置的数据宽度，.b表示1个字节，.w表示2个字节，.L表示4个字节。

例如，要将COMMAND寄存器设置为0x0406，并且已知器件的domain:bus:slot.function为29:00.3：

```
sudo setpci -s 29:00.3 04.w=0406
```

附录C： 应用笔记版本历史

表C-1： 版本历史

版本与日期	节/图/条目	更正
DS00005213B （2024年11月6日）	表16	更新了PHY和逻辑端口映射的详细信息。
	表47	将“BASE ADDRESS + 0x1_A220”更改为“BASE ADDRESS + 0x5_6004”。
	全部	从页脚中删除了“机密”标识。 更新了寄存器表中R/W列的数据。 对文本和格式进行了少量更改。
DS00005213A （2024年4月1日）	文档初始版本	

注：

MICROCHIP 网站

Microchip 网站（www.microchip.com）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的变更通知客户服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 www.microchip.com。在“支持”（Support）下，点击“产品变更通知服务”（Customer Change Notification）服务后按照注册说明完成注册。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（FAE）
- 技术支持

客户应联系其代理商、代表或应用工程师（FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http://microchip.com/support 获得网上技术支持。

Microchip 信息

商标

“Microchip”的名称和徽标组合、“M”徽标及其他名称、徽标和品牌均为Microchip Technology Incorporated或其关联公司和/或子公司在美国和/或其他国家或地区的注册商标或商标（“Microchip商标”）。有关Microchip商标的信息，可访问https://www.microchip.com/en-us/about/legal-information/microchip-trademarks。

ISBN：979-8-3371-2755-2

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关Microchip产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考Microchip Technology Inc.的英文原版文档。

本出版物及其提供的信息仅适用于Microchip产品，包括设计、测试以及将Microchip产品集成到您的应用中。以其他任何方式使用这些信息都将被视为违反条款。本出版物中的器件应用信息仅为您提供便利，将来可能会发生更新。您须自行确保应用符合您的规范。如需额外的支持，请联系当地的Microchip销售办事处，或访问https://www.microchip.com/en-us/support/design-help/client-support-services。

Microchip“按原样”提供这些信息。Microchip对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对非侵权性、适销性和特定用途的适用性的暗示担保，或针对其使用情况、质量或性能的担保。

在任何情况下，对于因这些信息或使用这些信息而产生的任何间接的、特殊的、惩罚性的、偶然的或附带的损失、损害或任何类型的开销，Microchip概不承担任何责任，即使Microchip已被告知可能发生损害或损害可以预见。在法律允许的最大范围内，对于因这些信息或使用这些信息而产生的所有索赔，Microchip在任何情况下所承担的全部责任均不超出您为获得这些信息向Microchip直接支付的金额（如有）。

如果将Microchip器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切损害、索赔、诉讼或费用时，会维护和保障Microchip免于承担法律责任。除非另外声明，在Microchip知识产权保护下，不得暗中或以其他方式转让任何许可证。

Microchip器件代码保护功能

请注意以下有关Microchip产品代码保护功能的要点：

- Microchip的产品均达到Microchip数据手册中所述的技术规范。
- Microchip确信：在正常使用且符合工作规范的情况下，Microchip系列产品非常安全。
- Microchip注重并积极保护其知识产权。严禁任何试图破坏Microchip产品代码保护功能的行为，这种行为可能会违反《数字千年版权法案》（Digital Millennium Copyright Act）
- Microchip或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。代码保护功能处于持续发展中。Microchip承诺将不断改进产品的代码保护功能。