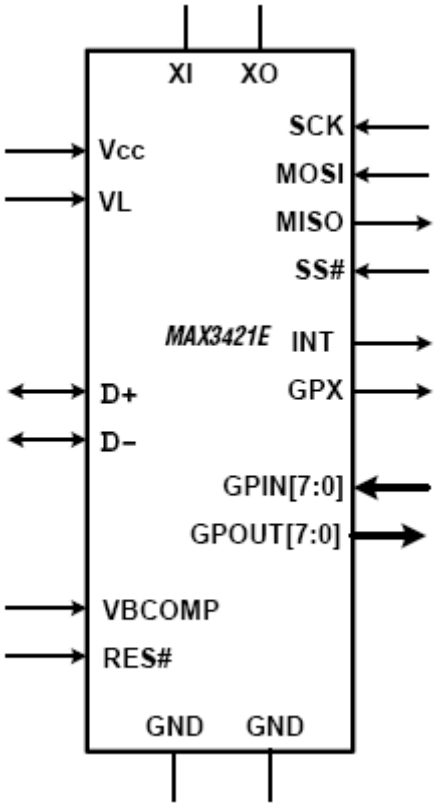
MAX3421E

编程指南

带SPI接口的USB外设/主机控制器

MAX3421E 可为任何带 SPI 接口的系统增加 USB 主机或外设功能。编程指南详细说明了主机工作模式所涉及到的每一个寄存器及其每一位的功能。有关 MAX3421E 外设工作模式的编程问题，请参考 MAX3420E 编程指南。



有关MAX3421E的更多信息，请访问：www.maxim-ic.com.cn/max3421e。
有关USB及Maxim公司USB产品的更多信息，请访问：www.maxim-ic.com.cn/usb。
SPI是Motorola, Inc.的商标。
Maxim标志是Maxim Integrated Products, Inc.的注册商标。
Dallas Semiconductor标志是Dallas Semiconductor Corp.的注册商标。

December 2006

关于编程指南

MAX3421E是具有双重作用的USB控制器，既可以设置为USB外设，又可设置为主机。作为外设时，MAX3421E的工作机制与只能作为外设的MAX3420E完全相同。如果只打算将MAX3421E用作USB外设，可运行现有的MAX3420E代码，并将HOST位置0 (缺省值)，编程细节可参考*MAX3420E编程指南*。

如果只打算将MAX3421E用作USB外设，但又想充分利用MAX3421E新提供的外设特性，请参阅本*编程指南*中新增的寄存器R21至R24，以及新增的寄存器位PULSEWID1/0、SEPIRQ和HOST的说明，同时参考*MAX3420E编程指南*。

大多数 MAX3421E 的用户希望了解主机工作模式的详细编程信息，这也正是编写该指南的目的所在。因此，列出主机操作所用到的位和寄存器(**表 1**)是理所当然的。但为了便于参考，**表 2** 给出了主机和外设工作模式用到的所有寄存器位。

如果浏览的是电子文档，那么表 1 可以作为导航工具。表中的链接可跳转到文档中详细说明相应寄存器和位的页面。

表 1. MAX3421E 主机寄存器(Host 位 = 1)

Reg	Name	b7	b6	b5	b4	b3	b2	b1	b0	acc
R0	—	0	0	0	0	0	0	0	0	—
R1	RCVFIFO	b7	b6	b5	b4	b3	b2	b1	b0	RSC
R2	SNDFIFO	b7	b6	b5	b4	b3	b2	b1	b0	RSC
R3	—	0	0	0	0	0	0	0	0	—
R4	SUDFIFO	b7	b6	b5	b4	b3	b2	b1	b0	RSC
R5	—	0	0	0	0	0	0	0	0	RSC
R6	RCVBC	0	b6	b5	b4	b3	b2	b1	b0	RSC
R7	SNDBC	0	b6	b5	b4	b3	b2	b1	b0	—
R8	—	0	0	0	0	0	0	0	0	—
R9	—	0	0	0	0	0	0	0	0	—
R10	—	0	0	0	0	0	0	0	0	—
R11	—	0	0	0	0	0	0	0	0	—
R12	—	0	0	0	0	0	0	0	0	RSC
R13	USBIRQ	0	VBUSIRQ	NOVBUSIRQ	0	0	0	0	OSCOKIRQ	RC
R14	USBIEN	0	VBUSIE	NOVBUSIE	0	0	0	0	OSCOKIE	RSC
R15	USBCTL	0	0	CHIPRES	PWRDOWN	0	0	0	0	RSC
R16	CPUCTL	PULSEWID1	PULSEWID0	0	0	0	0	0	IE	RSC
R17	PINCTL	0	0	0	FDUPSPI	INTLEVEL	POSINT	GPXB	GPXA	RSC
R18	REVISION	b7	b6	b5	b4	b3	b2	b1	b0	R
R19	—	0	0	0	0	0	0	0	0	—
R20	IOPINS1	GPIN3	GPIN2	GPIN1	GPIN0	GPOUT3	GPOUT2	GPOUT1	GPOUT0	RSC
R21	IOPINS2	GPIN7	GPIN6	GPIN5	GPIN4	GPOUT7	GPOUT6	GPOUT5	GPOUT4	RSC
R22	GPINIRQ	GPINIRQ7	GPINIRQ6	GPINIRQ5	GPINIRQ4	GPINIRQ3	GPINIRQ2	GPINIRQ1	GPINIRQ0	RC
R23	GPINIEN	GPINIEN7	GPINIEN6	GPINIEN5	GPINIEN4	GPINIEN3	GPINIEN2	GPINIEN1	GPINIEN0	RSC
R24	GPINPOL	GPINPOL7	GPINPOL6	GPINPOL5	GPINPOL4	GPINPOL3	GPINPOL2	GPINPOL1	GPINPOL0	RSC
R25	HIRQ	HXFRDNIRQ	FRAMEIRQ	CONDETIRQ	SUSDNIRQ	SNDBAVIRQ	RCVDAVIRQ	RWUIRQ	BUSEVENTIRQ	RC
R26	HIEN	HXFRDNIE	FRAMEIE	CONDETIE	SUSDNIE	SNDBAVIE	RCVDAVIE	RWUIE	BUSEVENTIE	RSC
R27	MODE	DPPULLDN	DMPULLDN	DELAYISO	SEPIRQ	SOFKAENAB	HUBPRE	LOWSPEED	HOST	RSC
R28	PERADDR	0	b6	b5	b4	b3	b2	b1	b0	RSC
R29	HCTL	SNDTOG1	SNDTOG0	RCVTOG1	RCVTOG0	SIGRSM	SAMPLEBUS	FRMRST	BUSRST	LS
R30	HXFR	HS	ISO	OUTNIN	SETUP	EP3	EP2	EP1	EP0	LS
R31	HRSL	JSTATUS	KSTATUS	SNDTOGRD	RCVTOGRD	HRSLT3	HRSLT2	HRSLT1	HRSLT0	R

将HOST位(R27的第0位)置为1时，将在以下三方面改变MAX3420E的寄存器映射表。重新定义寄存器R0至R7；清除了与主机工作无关的一些位；增加了寄存器25、26以及28至31。**表1**中标识为0的位在主机模式下必须写为0。为了清楚起见，没有用到的外设寄存器和位都标识为0。

表 2. MAX3421E 工作在外设和主机模式下的所有寄存器位

(阴影标识的位不随模式(HOST 位)的改变而改变。)

Reg	Perip.Name	Host Name	b7	b6	b5	b4	b3	b2	b1	b0	acc
R0	**EP0FIFO**	—	b7	b6	b5	b4	b3	b2	b1	b0	—
R1	**EP1OUTFIFO**	**RCVFIFO**	b7	b6	b5	b4	b3	b2	b1	b0	RSC
R2	**EP2INFIFO**	**SNDFIFO**	b7	b6	b5	b4	b3	b2	b1	b0	RSC
R3	**EP3INFIFO**		b7	b6	b5	b4	b3	b2	b1	b0	—
R4	**SUDFIFO**	**SUDFIFO**	b7	b6	b5	b4	b3	b2	b1	b0	RSC
R5	**EP0BC**	—	0	b6	b5	b4	b3	b2	b1	b0	RSC
R6	**EP1OUTBC**	**RCVBC**	0	b6	b5	b4	b3	b2	b1	b0	RSC
R7	**EP2INBC**	**SNDBC**	0	b6	b5	b4	b3	b2	b1	b0	—
R8	**EP3INBC**	—	0	b6	b5	b4	b3	b2	b1	b0	—
R9	**EPSTALLS**	—	0	ACKSTAT	STLSTAT	STLEP3IN	STLEP2IN	STLEP1OUT	STLEP0OUT	STLEP0IN	—
R10	**CLRTOGS**	—	EP3DISAB	EP2DISAB	EP1DISAB	CTGEP3IN	CTGEP2IN	CTGEP1OUT	0	0	—
R11	**EPIRQ**	—	0	0	SUDAVIRQ	IN3BAVIRQ	IN2BAVIRQ	OUT1DAVIRQ	OUT0DAVIRQ	IN0BAVIRQ	—
R12	**EPIEN**	—	0	0	SUDAVIE	IN3BAVIE	IN2BAVIE	OUT1DAVIE	OUT0DAVIE	IN0BAVIE	—
R13	**USBIRQ**	**USBIRQ**	URESDNIRQ	VBUSIRQ	NOVBUSIRQ	SUSPIRQ	URESIRQ	BUSACTIRQ	RWUDNIRQ	OSCOKIRQ	RC
R14	**USBIEN**	**USBIEN**	URESDNIE	VBUSIE	NOVBUSIE	SUSPIE	URESIE	BUSACTIE	RWUDNIE	OSCOKIE	RSC
R15	**USBCTL**	**USBCTL**	HOSCSTEN	VBGATE	CHIPRES	PWRDOWN	CONNECT	SIGRWU	0	0	RSC
R16	**CPUCTL**	**CPUCTL**	PULSEWID1	PULSEWID0	0	0	0	0	0	IE	RSC
R17	**PINCTL**	**PINCTL**	EP3INAK	EP2INAK	EP1INAK	FDUPSPI	INTLEVEL	POSINT	GPXB	GPXA	RSC
R18	**REVISION**	**REVISION**	b7	b6	b5	b4	b3	b2	b1	b0	R
R19	**FNADDR**	—	0	b6	b5	b4	b3	b2	b1	b0	—
R20	**IOPINS1**	**IOPINS1**	GPIN3	GPIN2	GPIN1	GPIN0	GPOUT3	GPOUT2	GPOUT1	GPOUT0	RSC
R21	**IOPINS2**	**IOPINS2**	GPIN7	GPIN6	GPIN5	GPIN4	GPOUT7	GPOUT6	GPOUT5	GPOUT4	RSC
R22	**GPINIRQ**	**GPINIRQ**	GPINIRQ7	GPINIRQ6	GPINIRQ5	GPINIRQ4	GPINIRQ3	GPINIRQ2	GPINIRQ1	GPINIRQ0	RC
R23	**GPINIEN**	**GPINIEN**	GPINIE7	GPINIE6	GPINIE5	GPINIE4	GPINIE3	GPINIE2	GPINIE1	GPINIE0	RSC
R24	**GPINPOL**	**GPINPOL**	GPINPOL7	GPINPOL6	GPINPOL5	GPINPOL4	GPINPOL3	GPINPOL2	GPINPOL1	GPINPOL0	RSC
R25		**HIRQ**	HXFRDNIRQ	FRAMEIRQ	CONDETIRQ	SUSDNIRQ	SNDBAVIRQ	RCVDAVIRQ	RWUIRQ	BUSEVENTIRQ	RC
R26		**HIEN**	HXFRDNIE	FRAMEIE	CONDETIE	SUSDNIE	SNDBAVIE	RCVDAVIE	RWUIE	BUSEVENTIE	RSC
R27	**MODE**	**MODE**	DPPULLDN	DMPULLDN	DELAYISO	SEPIRQ	SOFKAENAB	HUBPRE	LOWSPEED	HOST	RSC
R28		**PERADDR**	0	b6	b5	b4	b3	b2	b1	b0	RSC
R29		**HCTL**	SNDTOG1	SNDTOG0	RCVTOG1	RCVTOG0	SIGRSM	SAMPLEBUS	FRMRST	BUSRST	LS
R30		**HXFR**	HS	ISO	OUTNIN	SETUP	EP3	EP2	EP1	EP0	LS
R31		**HRSL**	JSTATUS	KSTATUS	SNDTOGRD	RCVTOGRD	HRSLT3	HRSLT2	HRSLT1	HRSLT0	R

术语

CPU

MAX3421E的SPI™接口可由任意SPI主控制器驱动，如微控制器、DSP或ASIC。为简便起见，*编程指南*将与MAX3421E SPI端口相连的SPI主控制器称作CPU。

SIE

SIE 是指串行接口引擎，该逻辑单元位于 MAX3421E 内部，负责处理所有的 USB 操作。为明确说明是哪个主体(CPU 还是 MAX3421E)对寄存器位进行置位或清除，在本文档中均使用了一致的专用术语。SIE 表示由 MAX3421E 来置位或清除寄存器位；CPU 表示由连接到 MAX3421E 的 SPI 主控制器来置位或清除寄存器位。

Mode

MAX3421E有两种工作模式：外设模式和主机模式。

- *MAX3420E编程指南*详细阐述了适用于外设模式的寄存器位。
- 标记为**外设模式和主机模式**的位在两种模式下均有效。
- 标记为**仅用于主机模式**的位只在主机模式下有效。

ISO

ISO是ISOCHRONOUS的缩写，是USB四种传输类型之一。MAX3421E作为USB主机时支持全速ISO传输。但是作为外设时不支持ISO传输。

复位

MAX3421E具有三个复位源：

1. 内部上电复位(POR)电路
2. RES#引脚
3. CHIPRES寄存器位

还有第四种方法可使器件复位。当HOST位改变状态(切换到主机或外设模式)时，SIE会清除一些特定的寄存器位。

接下来说明每种复位源产生的效果。

MAX3421E的寄存器位通过两种时钟源同步：

1. 12MHz内部振荡器
2. CPU发送给SPI接口的SCLK信号

触发RES#或者PWRDOWN复位会停止12MHz在的内部时钟。大多数寄存器位被异步清除。然而，通过SPI接口同步的寄存器位仍有效，因此CPU仍然可以控制SPI配置(如FDUPSPI 位)、USB总线下拉电阻以及PWRDOWN位的状态。

上电复位

SIE清除每一个寄存器位。一旦完成复位后，SIE设置以下各位以指示可用的缓冲区：

- IN3BAVIRQ (外设)
- IN2BAVIRQ (外设)
- IN0BAVIRQ (外设)
- SNDBAVIRQ (主机)

注意：POR 设置 HOST = 0，即缺省为外设工作模式。

触发 RES#引脚或设置 CHIPRES = 1

这些复位方式会停止12MHz内部振荡器，并清除大多数寄存器位，但SPI同步的寄存器位不受影响，CPU仍然可访问这些位。SPI同步的寄存器位包括：

- HOSCSTEN
- VBGATE
- CHIPRES
- PWRDOWN
- CONNECT
- SIGRWU
- FDUPSPI

- INTLEVEL
- POSINT
- GPX[B:A]
- GPOUT[7:0]
- DPPULLUP/DN

注意：这些复位方式设置 HOST = 0，选择外设工作模式。

改变模式位

外设模式切换到主机模式

SIE清除主机工作模式中不用的寄存器位。当HOST = 1时，CPU对这些位只能进行写0操作，不能写任何其他信息。

- EPSTALLS寄存器
- CLRTOGS寄存器
- EPIRQ寄存器
- EPIEN寄存器
- USBIRQ寄存器，只有VBUSIRQ位和NOVBUS IRQ位在主机模式下仍然有效
- USBIEN寄存器，只有VBUSIRQ位和NOVBUS IRQ位在主机模式下仍然有效
- PINCTL寄存器中的EP3/2/1INAK位

主机模式切换到外设模式

通过设置 CHIPRES 位，CPU 可从主机模式返回到外设模式，并根据上述规则清除寄存器位，将 HOST 位置为 0。

访问 MAX3421E 寄存器

SPI 主控制器通过读、写 R0 至 R31 (表 1)中的 21 个内部寄存器来控制 MAX3421E。SPI 主控制器通过触发 MAX3421E 的 SS# (从器件选择，低电平有效)引脚启动每次寄存器访问，同步输入 8 位 SPI 命令字节。命令字节的格式如**图 1** 所示。

b7	b6	b5	b4	b3	b2	b1	b0
Reg4	Reg3	Reg2	Reg1	Reg0	0	DIR 1=wr 0=rd	ACKSTAT

图 1. SPI 命令字节。所有 SPI 传输都是先发送第 7 位。ACKSTAT 位只在外设模式下有效；主机模式下，MAX3421E 忽略该位。

Reg4:Reg0设置寄存器地址，有效值为0至31。MAX3421E忽略大于31的Reg值。方向位(DIR)设置传输中剩余字节的方向。ACKSTAT位设置USB控制位(R9的第6位)的状态，并且仅当MAX3421E作为USB外设时才有效。当HOST = 1时，SIE忽略ACKSTAT位。

发送完命令字节之后，SPI主控制器按DIR位指示的方向传输一个或者多个字节。保持SS#为低，SPI主控制器为每个字节传输提供8个SCK脉冲。所有字节传送完成后，SPI主控器解除SS# (驱动为高)信号，终止传输过程。

MAX3421E提供两种寄存器类型：FIFO和控制寄存器。根据寄存器类型的不同，反复读写寄存器会产生不同的效果。

寄存器R1，R2和R4用来访问内部FIFO。使用命令字节选定寄存器地址后，SPI主控制器在同一个SPI传输过程中(保持SS#为低)通过重复读或写操作来装载或卸载连续的FIFO字节。例如，要向SENDFIFO中写入45个字节，SPI主控制器可执行以下步骤：

1. 设置SS# = 0，启动传输过程。
2. 发8个SCLK脉冲，发送命令字节00010010。该命令字节选择R2 (SNDFIFO)进行写操作(DIR = 1)。
3. 发8个SCK脉冲，每个脉冲向SNDFIFO寄存器同步写入一个数据位，每个SCK上升沿写一位。对应每个字节，SIE写入FIFO字节，并递增一个内部FIFO地址指针。
4. 第3步重复44次，总共向SNDFIFO写入45个字节。
5. 设置SS# = 1，终止传输。

寄存器R13至R31是MAX3421E的控制寄存器。如果SPI主控制器在同一个SPI传输过程中(SS#为低)反复读或写R13至R20，每个字节的读或写操作将自动递增内部寄存器地址。这种操作方式允许读或写地址连续的寄存器，而不必写新的命令字节来设置新的寄存器地址。寄存器地址采用这种方式持续递增，直到达到寄存器R20，此后寄存器地址将“保持”在R20。该特性使微处理器能够通过R20快速访问IO引脚。例如，要在一个GPIO引脚上输出预存的波形，SPI主控制器可发送命令字节10100010 (R20，写操作)，然后向R20发送多个数据字节，即可输出波形。

如果SPI主控制器指定寻址R21或更高地址的寄存器，那么在相同的SS#传输过程中，寄存器地址再次自动递增，直到达到寄存器R31为止，此后寄存器地址再次“保持”在R31。

全双工模式下 MISO 的前 8 位

当 FDUPSPI = 1 时，MAX3421E 的 SPI 端口工作在全双工模式下，即在每个 SCLK 上升沿，同时从 MOSI 引脚同步输入数据和从 MISO 引脚同步输出数据。SPI 通信的前 8 位组成同步输入到 MOSI 引脚的命令字节。该命令字节同步输入到 MOSI 引脚的同时，MAX3421E 向 MISO 引脚发送状态数据。在主机模式下，这些状态位如**图 2** 所示。

b7	b6	b5	b4	b3	b2	b1	b0
HXFRDN IRQ	FRAME IRQ	CONN IRQ	SUSDN IRQ	SNDBAV IRQ	RCVDAV IRQ	RSMREQ IRQ	BUSEVENT IRQ

图 2. FDUPSPI (HOST = 1)时的 MISO 数据

主机传输编程

当HOST = 1时，需要把它*当作*主机。这时，你需要发送主机请求，而不是响应主机请求。这就意味着所发送的每个包需包括以下部分：

- 存放在寄存器FNADDR中的功能地址
- 存放在寄存器HXFR中EP[3:0]的端点号
- 存放在寄存器HXFR中的包标识符(PID)
- 其他数据：SNDFIFO (确切的说，由SNDBC给出字节数目)中的OUT数据，或者RCVFIFO (由RCVBC给出字节数目)中的IN数据

功能地址和端点FIFO数据始终保存在MAX3421E中，直到被CPU更改为止。例如，要重发OUT包(由于外设错误)，CPU不需要重新装载SNDFIFO。CPU只需通过重写寄存器HXFR就能启动相同的传输。

MAX3421E处理USB的管理事务。设置上述各位，通过写寄存器HXFR启动传输，等待传输完成中断，检查HRSLT位(主机操作结果)确认传输结果。尽管MAX3421E有16种主机结果，您通常只会看到如**表3**所示的结果。

表 3. 正常 USB 操作的主机结果代码

HSRLT	Label	Meaning
0x00	hrSUCCESS	Successful Transfer
0x01	hrBUSY	SIE is busy, transfer pending
0x04	hrNAK	Peripheral returned NAK
0x05	hrSTALL	Perpheral returned STALL

在第36页给出了全部16个HRSL代码。**表3**中没有列出由电气或信号错误引起的结果；这些结果都是USB工作正常时的情况。hrBUSY结果针对以下应用场合：不使用HXFRDN中断请求来确定何时完成传输，而是采用轮询HRSL寄存器的方式检查是否完成传输。

主机能启动7种传输类型，如**表4**所示。

表 4. 不同传输类型的 HXFR 寄存器位设置

Xfr Type	HS	ISO	OUTNIN	SETUP	hex
SETUP	0	0	0	1	0x10
BULK-IN	0	0	0	0	0-ep
BULK-OUT	0	0	1	0	2-ep
HS-IN	1	0	0	0	0x80
HS-OUT	1	0	1	0	0xA0
ISO-IN	0	1	0	0	4-ep
ISO-OUT	0	1	1	0	6-ep

当SIE没有进行其他传输时，CPU可以随时写入寄存器HXFR。SIE会主动避免与其自动产生的SOF/KA帧标记发生冲突。

如果CPU在一帧内比较靠后的时刻进行寄存器HXFR写操作，以至于将要进行的传输可能会与帧标记发生冲突，那么SIE会自动将包的发送时间推迟到产生下一个帧标记之后。

注意：USB 主机通过 BULK 端点和 INTERRUPT 端点传输数据时都使用相同的方法。这两种传输类型的唯一不同在于*何时*安排数据包，即取决于控制固件函数。考虑到任何 BULK 的讨论同样适用于 INTERRUPT 传输，本文对 BULK 传输进行了阐述

关于 MAX3421E 的数据触发

USB协议使用两个PID (包标识符)之一给每个数据包添加了一个标签，这两个PID称为DATA0和DATA1。这些PID有助于检查USB错误。每个端点都有一个*数据触发*值与之关联，而这个数据触发值决定了使用哪一个DATA PID。发送至一个端点或由一个端点发送的第一个数据包(复位后)采用DATA0 PID。当收发双方均确认数据准确无误(通过发送/接收ACK握手包)时，双方均将触发值取反。因此正常情况下，从一个端点发送或接收的连续数据包具有不断转换的PID值—DATA0，DATA1，DATA0等等。

MAX3421E提供4位来实现数据触发功能。数据传输到一个端点后，SIE更新RCVTOGRD和SNDTOGRD位(第36页)，以指示所选端点的触发值。CPU可读取并存储这些位，以便在必要时重新初始化该端点的触发值。若要在某个端点传输之前初始化其触发值状态，CPU需要设置用于OUT数据的SNDTOG1和SNDTOG0位，或者设置用于IN数据的RCVTOG1和RCVTOG0位(第59页)。一次只能设置其中的一对。

对于**同一个**端点的多次**连续**传输，CPU 不需要初始化端点的触发值。MAX3421E 进行该端点的传输操作时，会同时更新其数据触发值。只有当 CPU **切换**端点时，才需要将端点的数据触发位恢复为先前保存的触发值。

BULK-OUT 传输编程

MAX3421E通过OUT包、SNDFIFO数据和握手包向外设发送BULK数据。

CPU首先检查是否SNDBAVIRQ = 1 (第56页)，以判断发送缓冲区是否可以用来装载数据。(只要OUT传输成功完成，SIE将置位中断请求位。) 如果缓冲区可用，CPU通过重复写寄存器R2，向SNDFIFO (第58页)写入多达64字节的数据。然后，CPU将字节计数值(装载到SNDFIFO的字节数)写入寄存器SNDBC (第57页)。装载SNDBC寄存器将使SIE清除SNDBAVAIRQ位。

注意：如果双缓冲 SNDFIFO 的第二个缓冲区为空，SIE 会立即重新触发 SNDBAVIRQ 位。

CPU *可能*需要为目标端点初始化数据触发值：

- 如果要传输到一个新的端点，那么CPU使用该端点最近一次保存的值来初始化其数据触发。
- 如果该端点是上一次传输的端点，那么CPU不需要初始化数据触发值。在每次传输完成后，SIE会自动更新数据触发。

最后，CPU把0010eeee (表4)装载到寄存器HXFR中，其中eeee是要发送数据的端点号。寄存器HXFR对装载操作敏感，也就是当CPU向寄存器HXFR装入数据时，SIE即刻启动传输过程。

SIE发送OUT令牌，PERADDR寄存器中的地址，EP[3:0]中的端点号，CRC5和EOP。紧接着SIE发送DATA0或者DATA1 PID (取决于触发位的状态)，由SNDBC给出字节数目的SNDFIFO数据以及CRC16。然后，SIE用6.5位次的时间等待外设响应。

注意：如果 SNDBC = 0，尽管 SIE 并不发送数据字节，但依然发送 DATA0/1 PID

当SIE收到握手或总线超时信号，将设置HXFRDNIRQ = 1，并使用HRSLT[3:0]位指示操作结果(第36页)。

如果外设返回ACK握手包，SIE同样也对数据触发取反，表明传输成功(HRSLT[3:0] = 0000)，并且置位SNDBAVIRQ，指示“发送缓冲区可用”。

如果外设没有应答(ACK)传输，数据可能需要重发。例如，外设通常使用NAK握手包应答来表明外设还没有准备好接收数据。CPU通过检查HRSLT位来查明OUT传输未被正确应答(ACK)的原因，只需通过重写HXFR = 0010eeee，CPU即可重新启动OUT传输。SIE使用同样的PERADDR和SNDBC数值发送SNDFIFO中的数据。

BULK-IN 传输编程

CPU发送一个IN令牌请求外设向它发送BULK数据。接着SIE将数据传输至RCVFIFO，并对该传输产生应答。

CPU写入HXFR = 0000eeee (表4)，启动IN传输，其中eeee表示所要求的端点地址。

SIE发送一个IN令牌，PERADDR寄存器中的地址，EP[3:0]中的端点号以及CRC5。接着SIE用6.5位次的时间等待外设响应。如果外设以DATA0或DATA1 PID以及紧随其后的数据响应，则SIE将接收到的数据字节装入RCVFIFO，并计算字节数。SIE在包的末尾检查包是否出错，更新寄存器RCVBC和HRSLT位，随之置位HXFRDNIRQ位。SIE是否触发RCVDAVIRQ取决于传输结果。

如果IN数据没有错误(HRSLT = 0000)，则SIE发送ACK令牌，取反数据触发值，并置位RCVDAVIRQ以指示新的IN数据有效。

如果IN数据没有错误，但数据触发不匹配(外设发送的DATA0或DATA1 PID与端点的触发值不匹配)，那么SIE发送ACK握手包，但不对数据触发取反，也不触发RCVDAVIRQ。在这种情况下，SIE设置HRSL = 0110 (触发错误)。如果外设在前一次IN传输中收到错误的ACK握手包，就会发生这种情况。在这种情况下，主机忽略RCVDATA FIFO中的数据，因为该数据是外设丢失了最近一个ACK握手包而错误重发的数据。SIE应答该传输过程，但不更新自身的触发位，这样可使外设对其触发位取反，从而强制数据触发机制重新同步。

如果HRSLT各位指示数据错误，则SIE不发送ACK，也不对数据触发取反或置位RCVDAVIRQ位。

CPU检查返回的HRSLT[3:0]位，据此来响应HXFRDNIRQ。如果结果为 0000 (成功)，CPU读取RCVBC，确定字节数，然后利用重复读RCVFIFO寄存器操作来读取该数目的字节(第48页)。CPU获得数据后，应清除RCVDAVIRQ位(通过向该位写 1 实现)。如果在双缓冲RCVFIFO中有另一个缓冲的IN数据包，SIE将立即重新触发RCVDAVIRQ位。

注意：如果 IN 传输出错，SIE 不会自动重试 IN 传输。相反，SIE 会将这一错误通知 CPU (通过 HXFRDNIRQ 位和 HRSLT 位)，并产生正确的 USB 响应。通常 CPU 会重新装载寄存器 HXFR 以重发 IN 包。

CONTROL传输编程

主机在以下两个或三个阶段会发送CONTROL传输：

1. SETUP包，向外设发送8个字节的“操作码”
2. 可选的数据阶段，通常是BULK IN请求
3. 状态阶段

主机向外设的缺省控制端点0发送CONTROL传输。

1. 设置

CPU向SUDFIFO中写入8字节的SETUP数据。由于SETUP包的有效载荷总是8个字节，因此没有与SUDFIFO关联的字节计数寄存器。随后CPU将00010000 (表4)写入寄存器HXFR，指示SIE将SETUP包发送到端点0。

然后，SIE将发送由SETUP PID、PERADDR寄存器中的地址、端点0000、CRC5以及EOP组成的SETUP包，接着发送SUDFIFO中包含8个字节的DATA0包。SIE用18位次时间等待设备响应或超时，最后通过触发HXFRDNIRQ位和更新HSRLT各位(第36页)来终止传输。USB规范要求外设必须应答SETUP包。

注意：SIE 针对 CONTROL 传输的不同阶段发送固定的 DATA0 和 DATA1 PID 令牌，而与内部数据触发的设置无关。

---

2. 数据 (可选)

如果需要数据阶段，传输设置为BULK-IN或BULK-OUT传输。某些CONTROL传输(如Set_Address)并不需要数据阶段，因为命令数据内嵌于SETUP包的8个字节中。

3. 状态

状态阶段是CONTROL传输独有的阶段，该阶段为这些关键任务的传输提供额外的保护措施。状态阶段是由与前一个阶段方向相反的IN包或OUT包构成。

STATUS包采用BULK-OUT或BULK-IN传输，其不包含数据，并且总是采用DATA1 PID。编程人员可以通过设置标准的BULK传输来发送这些传输。但为了方便起见，MAX3421E为握手包提供了HS-OUT和HS-IN代码，因此SIE即可完成该任务。

HS-OUT

主机发送OUT握手包来终止CONTROL-READ请求(如Get_Descriptor)。若要发送一个OUT握手包，CPU只需将值0xA0 (表4)装入寄存器HXFR。

SIE 发送 OUT PID、寄存器 PERADDR 中的地址、端点 0 和 CRC5。然后 SIE 发送 DATA1 PID，并用 6.5 位次时间等待外设响应或超时。

注意：这与采用固定DATA1 PID的无数据BULK-OUT包是相同的。当SIE接收到握手包或总线超时后，将设置HXFRDNIRQ = 1，并通过HRSLT[3:0]位指示结果(第36页)。

HS-IN

主机发送IN握手包来终止CONTROL-WRITE请求(如Set_Address)。若要发送一个IN握手包，CPU只需将值0x80 (表4)装入寄存器HXFR。

SIE将发送IN PID、寄存器PERADDR中的地址、端点0和CRC5的值，然后用6.5位次时间等待外设响应。接着SIE更新HRSLT各位，触发HXFRDNIRQ位。如果外设返回DATA1 PID (无数据)，SIE会自动发送ACK握手包，表明成功终止CONTROL传输。

关于ISO传输

USB ISOCHRONOUS传输的特点在于实时传输，而无需其他USB传输类型用到的握手信号。一旦USB主机进行了设备枚举和给出了ISO带宽要求(带宽代表每个1ms帧内分配给ISO端点的字节数)，USB规范要求主机每帧要提供或读取该数目的字节。

使用MAX3421E时需要特别注意这种传输类型，因为MAX3421E可能与低速SPI接口或CPU相接。控制器必须能够满足预定的要求。如果速度不匹配，MAX3421E采用两种方式处理该问题，具体通过DELAYISO位的设置实现。

双缓冲

MAX3421E的SNDFIFO和RCVFIFO都采用双缓冲结构，即每个缓冲区都提供两组64字节FIFO和字节计数寄存器。对于最大包尺寸超过64个字节的ISO传输来说，这种双缓冲结构至关重要。对编程人员来说，双缓冲特性是不可见的：

- **IN:** 如果CPU在卸载RCVFIFO之后清除了RCVDAVIRQ (接收数据有效IRQ)位，而此时有另一个数据包在另一个缓冲区等待处理，那么SIE会立即重新触发RCVDAVIRQ位。

- **OUT:** 如果CPU通过将字节数(装入SNDFIFO的字节数)装入寄存器SNDBC，来清除SNDBAVIRQ (发送缓冲区可用IRQ)，而此时另一个缓冲区可以用来装载数据，那么 SIE会立即触发SNDBAVIRQ位。

尽管 SNDFIFO 和 RCVFIFO 的容量都是 64 字节，但是只要 CPU 能及时地提供或读取数据，通过把连续装载/读取这些 FIFO 的数据拼接成更大的数据包，SIE 就能发送和接收任意大小(可高达 USB 规范所限制的 1023 字节)的 ISO 数据包。.

通过检测 EOP (End-Of-Packet，包结束)总线状态，SIE 检测 ISO IN 数据包的结尾。对于 OUT 传输，当 CPU 向 FIFO 中装载少于 64 字节的数据时，SIE 通常可检测到数据有效载荷的结尾。这既适用于单个包(63 字节或者更少)的数据载荷，又适用于包含多次 64 字节缓冲装载并且其最后一个缓冲区包含 63 个字节或者更少的数据载荷。然而，还存在一种特殊情况，就是最后一个数据包大小恰好是 64 个字节。SIE 需要确定这仅仅是在多缓冲区装载过程中的一个 64 个字节，还是数据长度为 64 字节整数倍的传输的最后 64 个字节。CPU 通过装载 SNDBC = 0 通知 SIE 这个 64 字节的包表示 OUT 传输的结束。这是 SIE 的一个特殊信号，当装载寄存器 SNDBC 时，SIE 不会像往常一样切换缓冲区。

ISO-IN 传输编程

CPU通过写入HXFR = 0100eeee，启动ISO IN传输。

SIE 利用寄存器 PERADDR 中的地址、EP[3:0]中的端点号、CRC5 和 EOP，发送一个 OUT 令牌。然后使用 6.5 位次时间等待 DATA0 PID 或总线超时。如果 SIE 接收到 DATA0 PID，则开始将数据载入寄存器 RCVFIFO。当 FIFO 中写满 64 个字节(或者总线上出现包结束信号)时，SIE 将字节计数写入寄存器 RCVBC，并触发 SNDBAVIRQ 位，以表明 CPU 可以根据需要向 FIFO 装载更多的数据。在 ISO IN 传输(EOP)结束时，SIE 更新 HRSLT 各位并触发 HXFRDNIRQ。

ISO-OUT 传输编程

SIE主机向外设发送一个OUT包，紧跟其后发送一个采用固定DATA0 PID的数据包。需考虑以下三种情况：

1. OUT数据有效载荷少于64个字节
2. 正好64个字节
3. 多于64个字节。

1. OUT 包有效载荷少于 64 个字节

CPU对这种情况的编程方式类似于BULK传输。CPU把字节载入寄存器SNDFIFO，并将字节计数写入寄存器SNDBC。然后将0110eeee (表4)写入寄存器HXFR，开始传输。

SIE 利用寄存器 PERADDR 中的地址，EP[3:0]中的端点号，CRC5 和 EOP 发送一个 OUT 令牌。SIE 接着立即发送 DATA0 PID，由 SNDBC 给出字节数目的 SNDFIFO 数据的和 CRC16。识别出 SNDBC<64 时，则 SIE 终止传输，并触发 SNDBAVIRQ。外设不会向 SIE 发送任何握手来进行确认。如果 CPU 在帧内调度传输进行得太迟，则无法避免与自动生成的 SOF 包发生冲突，因此突出的错误情况可能是一个包被截掉或丢失。DELAYISO 位用来控制 SIE 在这种错误情况下的处理方法。

2. OUT包有效载荷恰好为64个字节

这种情况适用于单个64字节包，或者多缓冲传输中的最后数据有效荷载正好是64个字节的情形。CPU把64个字节装入寄存器SNDFIFO，并写入SNDBC = 64，等待SNDBAVIRQ = 1，然后写入SNDBC = 0，指示传输结束。

3. OUT包有效载荷多于64个字节

CPU向寄存器SNDFIFO中装载64个数据字节，并写入SNDBC = 64。如果SNDBAVIRQ = 1, 则可将数据包的第二部分装入寄存器SNDFIFO，然后把字节数写入寄存器SNDBC。将0110eeee装入寄存器HXFR，开始传输。

当FIFO可用时，SIE继续触发SNDBAVIRQ，请求装载下一块ISO OUT数据。双缓冲机制允许SIE发送一个FIFO中的OUT数据，同时CPU向另一个缓冲区装载数据。

SIE 通过更新 HRSLT 位和触发 HXFRDNIRQ 来终止 ISO OUT 传输。在 SIE 发送完 SNDBC<64 的 FIFO 的最后一个字节后，或是当 CPU 写入 SNDBC = 0 时，ISO OUT 传输终止。

外设和主机模式下的有效位

MAX3421E的一些特性在外设和主机模式下都是有效的。这些特性涉及不属于USB系统的一些功能，诸如SPI主控制器如何配置，中断引脚如何工作以及IO引脚的状态等。这些特性可划分为以下各类：

- 接口配置
  - FDUPSPI
  - GPX[B:A]
  - INTLEVEL
  - POSINT
- IO引脚配置及其输出值
  - GPIN 引脚—中断极性和使能
  - GPOUT 引脚状态
  - VBCOMP 引脚检测电平
- 整体芯片操作
  - CHIPRES
  - PWRDOWN
  - OSCOK (振荡器就绪)
- 一些IRQ位和IE位

当SPI主控制器命令MAX3421E改变工作模式时(从外设模式切换到主机模式，或从主机模式切换到外设模式)，**表2**列出了取值保持不变的寄存器和寄存器位。可见，模式的改变并不影响SPI接口的配置以及GPOUT的引脚状态。IE位和GPIN中断位(GPINIRQ和GPINIE)在模式改变时仍保持不变。也就是说，如果在模式改变之前GPIN中断处于悬挂状态，那么在模式改变之后中断仍处于悬挂状态。

注意：有两种方法可使 MAX3421E 从主机模式切换到外设模式。第一种方法，SPI 主控制器可将 MODE 寄存器 R27 中的 HOST 位写为 0。第二种方法，SPI 主控制器可以先设置 USBCTL 寄存器 R16 中的 CHIPRES 位，然后将其清除。如果想要使用一个“干净”的外设，首选 CHIPRES 这种方法。

BUSEVENTIRQ, BUSEVENTIE

含义： **BUSEVENTIRQ**：已发生了两个总线事件中的一个。
BUSEVENTIE： 使能 BUSEVENTIRQ。

模式： **仅用于主机模式**

当 SIE 完成以下两个总线事件之一时，置位 BUSEVENTIRQ 位：

- 总线复位(当BUSRST从1变为0)
- 总线恢复(当BUSRSM从1变为0)

CPU 通过向 BUSEVENTIRQ 位写 1 清除该位。

CPU 可设置和清除 BUSEVENTIE 位。当 BUSEVENTIE = 1 时，BUSEVENTIRQ 被使能为可激活 INT 引脚的中断源。

编程要点
该中断由两个中断源共享：完成总线复位或总线恢复。因为这是一个共享中断，在启动任一个总线事件之前(在设置 BUSRST = 1 或 BUSREM = 1 之前)，最好清除 BUSEVENTIRQ 位。

BUSRST

含义： 向USB外设发送总线复位信号。

模式： **仅用于主机模式**

CPU设置该位以向外设发送一个50ms总线复位(SE0)信号。

总线复位信号结束时，SIE清除该位。

CPU可以通过清除该位提前终止50ms SE0总线信号。以这种方式终止复位信号会触发BUSEVENTIRQ。

编程要点

CPU通过设置该位指示SIE在D+和D-线路上发送一个总线复位信号。该位设置后，CPU可以通过轮询BUSRST位是否为0，或通过响应BUSEVENTIRQ，来检测50ms时间结束。依次按照下列步骤来设置总线复位：

1. 设置BUSRST = 1。
2. 测试是否BUSRST = 0，或响应BUSEVENTIRQ。
3. 设置SOFKAENAB = 1开启帧标记。
4. 至少等待一个FRAMEIRQ。

第4步确保主机逻辑电路为第一个主机处理操作做好准备。

CHIPRES

含义： 芯片复位

模式： **外设模式和主机模式**

CPU置位该位，使器件复位。其作用等效于拉低RES#引脚。

CPU清除该位可使芯片退出复位状态。

编程要点

CPU可在置位该位后立即将它清除。上电复位或通过设置CHIPRES位复位MAX3421E时，将清除大多数的寄存器位，其中包括HOST位，因此这可以将MAX3421E设置为USB外设模式。

设置 CHIPRES = 1 将停止内部振荡器。在设置 CHIPRES = 0 使芯片退出复位状态后，CPU 应检查 OSCOK 中断请求。只有在该中断置位后，表明振荡器和 PLL 已经稳定下来，CPU 才能继续工作。

CONDETIRQ, CONDETIE

含义： **CONDETIRQ：** 外设连接/断开中断请求
CONDETIE： 外设连接/断开中断使能

模式： **仅用于主机模式**

SIE设置CONDETIRQ位，用于指示外设的插入或拔出情况。CPU通过向CONDETIRQ位写1来清除该位。下列情况将置位CONDETIRQ位：

- 总线由 J 状态或 K 状态转换为 25µs 的 SE0。这表明外设已经断开。
- 总线由 SE0 的 8 位次时间转换为 25µs 的 J 状态或 K 状态。这表明外设已经连接。

CPU可置位和清除CONDETIE位。当CONDETIE = 1时，CONDETIRQ使能为可激活INT引脚的中断源。

编程要点
要检测外设的连接和断开情况，CPU应首先设置HOST = 1，将MAX3421E置为主机模式。然后分别设置DPPULLDN = 1和DMPULLDN = 1，将D+和D-信号设置为逻辑低电平。

若要确定外设是否连接，CPU 读取 HSRL 寄存器，并检查 JSTATUS 位和 KSTATUS 位。当 CPU 设置 SAMPLEBUS 位时，通常这些状态位会进行更新，但是当触发 CONDETIRQ 中断时，这些状态位也会自动更新。

DELAYISO

含义： 延迟数据传输到ISOCHRONOUS端点，直到下一帧(直到发出下一个SOF包)再进行传输。通常如果数据包安排在帧内较迟的时刻传输，就无法避免与下一个自动生成的SOF包发生冲突。

模式： **仅用于主机模式**

CPU设置和清除该位。

编程要点

USB规范确保主机为所枚举的ISO端点分配足够的总线带宽(帧时间)，以保证每个1ms帧内的ISO数据传输。也就是说，主机不仅要在每一帧内，而且要在每一帧内*尽可能早地*安排IN包或OUT包(通过写寄存器HXFR)，以保证该传输在MAX3421E自动生成下一个 SOF包之前结束。

由于 MAX3421E 可以与任何速度的 SPI 主控制器相连，并且不能确保控制固件在每一帧内都尽可能早地写寄存器 HXFR，因而不能保证数据包的传输。当包安排在帧内较晚时间传输时，SIE 提供的 DELAYISO 位可控制相应的操作。当存在包/SOF 时序冲突时，DELAYISO 位决定是先对数据包进行操作还是先对 SOF 包进行操作。

如果 DELAYISO = 1，当 CPU 通过装载寄存器 HXFR 传送 ISO 包时，SIE 将检查 1ms 帧内可用的剩余时间。如果 SIE 发现一帧内没有足够的时间来发送或接收 256 字节的 ISO 包，则将 ISO 包的发送操作推迟到下一帧再进行。

注意：选取 256 字节窗口可支持常见的 ISO 应用，如 CD 音频。在此应用中，采样速率为 44.1kps，每次采样双通道，每个通道 16 位分辨率，可知每 ms 帧需要传输 176 个字节。256 字节窗口增加了一定裕量。

ISO OUT-SOF 冲突

SIE提供两种方法来处理发送ISO OUT包数据与自动生成的SOF包之间的冲突问题。SIE既可以先发送包，然后提前终止数据发送以产生SOF，也可以延迟发送整个包直到产生下一个SOF包。

注意：另一个 ISO 错误状态是数据空载，通过结果代码 HRSLT = 0x06 指示该错误。如果在发送 ISO 数据包的同时，SIE 需要 SNDFIFO 数据，而 CPU 却没有及时装载 SNDFIFO，这时就会发生这种数据空载错误。

DELAYISO = 0 时的 ISO OUT

如果正在进行ISO-OUT数据包传输时，SIE到了生成下一个SOF包的时刻，则SIE截断数据包传输，并产生SOF。SIE通过设置HRSLT[3:0] = 0x03以指示数据丢失错误，紧接着触发FRAMEIRQ。这种情况下，并不触发HXFRDNIRQ。

DELAYISO = 1 时的 ISO OUT

如果CPU对寄存器HXFR的写操作时间临近该帧结束，则不允许SIE传输256个字节数据，SIE将延迟发送OUT数据包。这种情况下，SIE发送下一个SOF包(并触发FRAMEIRQ)；然后发送延迟的ISO包，终止传输，触发HXFRDNIRQ并提供结果代码0x00 (hrSUCCESS)。CPU在FRAMEIRQ触发之后通过采样SNDBAVIRQ位，来检测一个延迟的包。如果SNDBAVIRQ为0，表明缓冲区SNDFIFO仍然被USB传输所占用，因此可判断包被延迟发送。

ISO IN 与 SOF 冲突

SIE 处理 IN 传输冲突的方法与 OUT 传输有所不同，因为主机不可能命令外设停止发送数据，进而避免与下一个 SOF 包发生冲突。因此，在发生冲突的情况下，SIE 要么读取剩余的 ISO IN 数据而不生成 SOF 包，要么 SIE 延迟发送 IN 请求直到生成下一个 SOF 包为止。

注意：另一个 ISO 错误状态是数据过载，并通过代码 HRSLT = 0x06 指示这种错误情况。当 SIE 具有可用的 ISO IN 数据却没有空间存放时，就会产生此类错误。这是因为没有可用的 RCVFIFO (CPU 没有及时卸载 RCVFIFO)造成的。

DELAYISO = 0 的 ISO IN

如果由于ISO IN请求而导致外设发送的数据包时间跨越SOF生成时刻，那么SIE继续接收IN数据而阻止发送该帧的SOF包。这种情况下，SIE设置HRSLT[3:0] = 0x03，当IN传输结束时并不触发HXFRDNIRQ，但要为该帧触发FRAMEIRQ。CPU识别到没有触发HXFRDNIRQ而发生FRAMEIRQ的情况时，即检测到了这种错误。

注意：不管实际上是否发送了 SOF 包，SIE 总要增加帧计数值。

DELAYISO = 1 的 ISO IN

如果 CPU 启动 ISO-IN 传输的时刻会导致接收 256 字节数据包时与下一个 SOF 产生冲突，那么 SIE 将推迟发送 IN 包，直到下一帧到来。在这种情况下，SIE 为下一帧触发 FRAMEIRQ，发送 ISO-IN 包，传输 IN 数据，触发 HXFRDNIRQ 并产生 HRSLT[3:0] = 0x00 (hrSUCCESS)。

注意：如果检测到 ISO 调度时间有问题，则必须修改系统，从而在每帧内尽可能早地分配 ISO 包。这可以通过提高 SPI 接口的速度、调整固件或者同时采用两种方法。

DPPULLDN, DMPULLDN

含义： 在D+、D-与地之间连接15kΩ的内部电阻。

模式： **仅用于主机模式**

CPU置位和清除这两位。

编程要点

USB主机通过连接这两个下拉电阻将总线置为静止状态。低速外设在D-和3.3V之间连接一只1500kΩ电阻；全速外设在D+与3.3V之间连接一只1500kΩ电阻。因为总线通过这些15kΩ电阻弱下拉，因此SIE不仅能够检测出外设何时插入，而且还可检测其速度。

如果MAX3421E工作于外设模式(HOST = 0)，这些位都应设置为0 (缺省值)。

第63页阐述了DPPULLDN位和DMPULLDN位在设计中的作用，它们实现了两个USB连接器的功能，并且能够自动检测主机和外设连接方式。

FDUPSPI

含义： 全双工SPI端口

模式： **外设模式和主机模式**

CPU置位该位使SPI端口工作于全双工模式。

CPU清除该位使其工作于半双工模式。

POR： FDUPSPI = 0 (半双工)
芯片复位: *不变*
总线复位: *不变*
掉电: 读-写

编程要点
半双工SPI
在半双工模式下(FDUPSPI = 0)，MOSI (主控制器输出，从机输入)引脚变为双向IO引脚，并且MISO (主控制器输入，从机输出)引脚为三态。

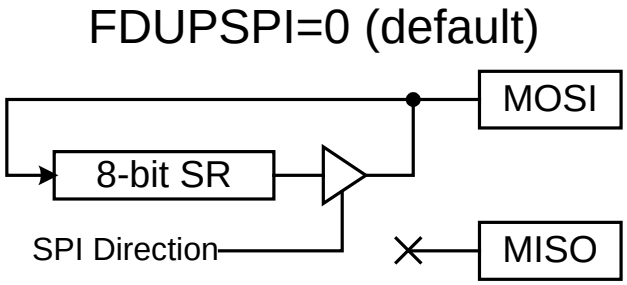


图3. 半双工SPI接口

全双工SPI
全双工模式(FDUPSPI = 1)具有独立的MOSI引脚和MISO引脚。该配置结构额外提供了以下特性：在同步输入每次传输的第一个字节(命令字节)时，同时输出8位状态信息。

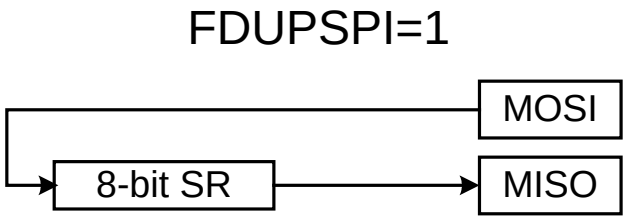


图4. 全双工SPI接口

SPI 命令字节

同步输入SPI接口的第一个字节是命令字节，用来设置寄存器地址、传输方向以及作为UBS外设时可直接设置ACKSTAT位的1位。在所有SPI传输中，无论输入还是输出，都是先发送b7，最后发送b0。

MOSI Bit	Signal
7	REG4
6	REG3
5	REG2
4	REG1
3	REG0
2	0
1	Direction (1= Wr, 0 = Rd)
0	ACKSTAT (peripheral)

图5. SPI命令字节

一个SPI传输过程从SPI主控制器驱动CS#引脚为低开始，然后提供8个SPI时钟，在时钟的上升沿同步输入命令字节。REG[4:0]用来设置寄存器地址，方向位用来设置SPI传输的读或写操作。工作在USB外设模式时，ACKSTAT位写入寄存器EPSTALLS的相应位。

发送完命令字节之后，SPI 主控制器发出一组或多组 8-SCK 时钟，向 MAX3421E 同步输入或输出字节数据。访问 FIFO 时，只要 CS#保持低电平，随命令字节输入的寄存器地址始终有效。当对该端点 FIFO 进行读或写操作时，这种适合多字节操作的功能非常方便。例如，要向 EP0FIFO (外设模式)中装载 37 个数据字节，SPI 主控制器写入命令字节 00000010，即选择 R0 (EP0FIFO)进行写操作(方向位为 1)。然后向 SPI 端口写入 37 个字节，最后驱动 CS#为高，则完成了 SPI 传输过程。

注意：MOSI 和 MISO 数据都是在 SCK 的上升沿采样。数据在 SCK 的下降沿发生变化。

当SPI主控制器驱动CS#返回高电平时，SPI传输过程结束。

SPI模式

SPI标准定义了4种时钟模式，表现为两种信号模式：CPOL (时钟极性)和CPHA (时钟相位)。这些信号表示为(CPOL，CPHA)的形式。要求上升沿SCK，并且要求MOSI数据在第一个时钟上升沿之前就有效的接口可以不加调整地工作在模式(0,0)和(1,1)下。这一特性使MAX3420E无需模式引脚即可工作在模式(0,0)和(1,1)下。

下面的示波器迹线展示了在微处理器和MAX3420E之间传输相同数据的过程。图 6采用SPI模式(0,0)，图 7采用SPI模式(1,1)。两者的区别在于SCK信号的空闲电平，模式(0,0)时为低，模式(1,1)时为高。两种模式下，在SCK上升沿采样的MOSI和MISO数据是相同的。

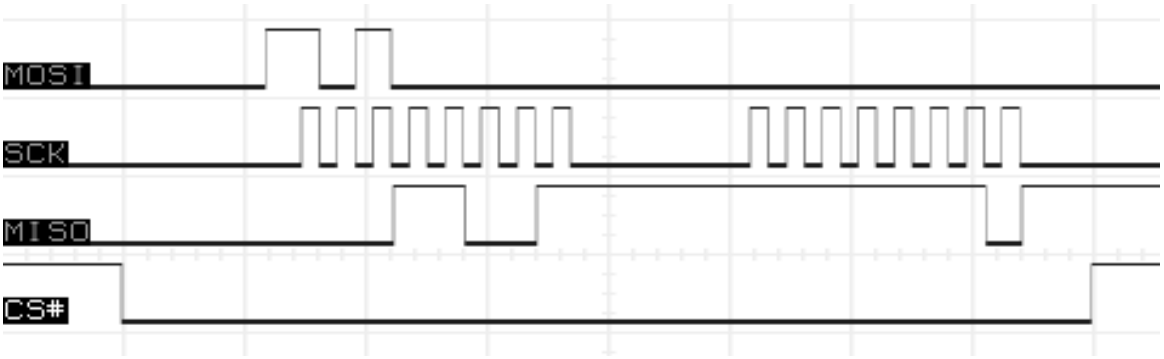


图6. 工作在(0,0)模式下的SPI接口

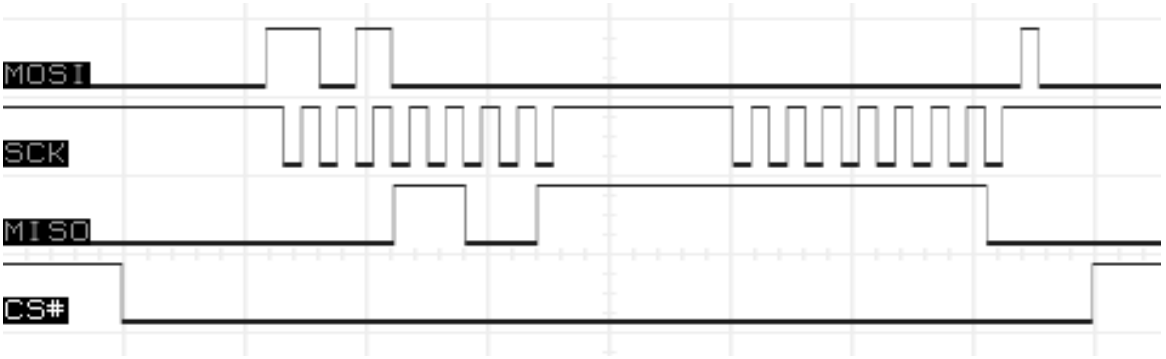


图7. 工作在(1,1)模式下的SPI接口

FRAMEIRQ, FRAMEIE

含义： **FRAMEIRQ：** 帧发生器中断请求
FRAMEIE： 帧发生器中断使能

模式： **仅用于主机模式**

只要SIE产生1ms帧标记，就会置位FRAMEIRQ位。帧标记由一个全速的SOF包或一个低速持续有效(keep-alive)脉冲组成。CPU通过向FRAMEIRQ位写1来清除该位。

CPU刻置位和清除FRAMEIE位。当FRAMEIE = 1时，FRAMEIRQ使能为可激活INT引脚的中断源。

编程要点
在全速模式下，SIE 在 SOF 包的开头设置 FRAMEIRQ 中断。

FRMRST

含义： 复位SOF帧计数器

模式： **仅用于主机模式**

CPU置位该位将清除帧计数器。

SIE清除该位表示完成操作。

编程要点

只有当MAX3421E作为全速主机时，该位才有意义。(当MAX3421E作为低速主机时，不存在SOF包或帧计数器。)设置FRMRST = 1后，下一个SOF包的帧计数器为0。

MAX3421E使用内部预分频器为帧计数器提供时钟。设置FRMRST = 1并不复位预分频器，只复位帧计数器，因此可保证每ms提供稳定的SOF包。

GPIN(0-7), GPINPOL(0-7) GPINIRQ(0-7), GPINIE (0-7)

含义： **GPIN** 通用输入引脚0至7
GPINPOL 通用IN中断极性0至7
GPINIRQ 通用IN中断请求0至7
GPINIE 通用IN中断使能0至7

模式： **外设模式和主机模式**

GPIN

CPU通过读取GPIN寄存器的各个位，获得GPIN引脚状态。只有外部电路可以控制这些寄存器位的状态。所有 8 个GPIN引脚均由内部电阻(约 20kΩ)弱上拉到V_{CC}。

GPINPOL

CPU 通过设置 GPINPOL 位的状态，设置每一个 GPIN 引脚中断请求的边沿极性(**表 5**)。上电时将清除这些位。无论是在 CHIPRES = 1 还是模式改变的情况下，这些位都保持其原有状态。

表 5. GPIN 中断的边沿极性

GPINPOL	Polarity
0	neg edge
1	pos edge

GPINIRQ

当 GPIN 引脚具有正向或负向跳变时，MAX3421E 置位 GPIN 中断请求位。GPINPOL 位(表 5)控制有效的边沿极性。无论各 GPIN 中断是否使能或 IE 位是否置位，GPINIRQ 位均有效。

GPINIE

CPU 设置 GPIN 中断使能位，从而可将对应的 IRQ 位传输到中断逻辑电路，最终驱动 MAX3421E 的 INT 引脚。如果 IE = 1，那么当 SEPIRQ = 0 时，在 INT 引脚上产生使能的 IRQ，当 SEPIRQ = 1 并且 GPX[B:A] = 10 时，则在 GPX 引脚上产生使能的 IRQ。

注意：在正常工作中，GPIN中断与MAX3421E其它所有中断源相“或”的结果出现在INT引脚上。可以将 8 个GPIN中断请求位从INT引脚的中断源中分离出来，使这 8 个中断请求位作为一组单独作用于GPX引脚。在该模式下，GPX引脚作为MAX3421E的第二个INT 输出引脚。对于需要最大限度减小外部事件检测时间的系统来说，该模式尤其适合。SEPIRQ位用于使能这组单独的中断源。

GPOUT(0 至 7)

含义： 通用输出引脚0至7

模式： **外设模式和主机模式**

CPU置位和清除这些位。

编程要点

CPU对这些位进行写操作以控制GPOUT引脚的状态。输出电平参考VL引脚上的电压。CPU还可对这些位进行读操作。读取的位值反映了用于驱动输出缓冲器的输出触发器状态。因此，如果输出引脚驱动大负载(例如LED)，逻辑电平可能会受到影响，但是CPU仍然能够读取输出引脚的正确逻辑状态。

GPXB, GPXA

含义： GPXB和GPXA两位决定GPX引脚的输出。

模式： **外设模式和主机模式**

CPU可置位和清除这些位。

编程要点

GPX引脚可输出下表所示的4个内部信号之一：

GPXB	GPXA	GPX 引脚
0	0	OPERATE (内部 POR 的反码)
0	1	VBUS 检测
1	0	BUSACT 或 INIRQ*
1	1	SOF (当 SOF 包到达时从 0 跳变至 1，信号的占空比为 50%)

** 如果SEPIRQ = 1*

当GPXB = 1，GPXA = 0 时，GPX输出缺省为BUSACT。然而，如果SEPIRQ位被置为 1，那么中断请求信号将取代BUSACT信号，只要 8 个GPIN引脚中有一个发生了从 0 到 1 或从 1 到 0 的跳变，就会在该引脚上产生有效的中断请求信号。这种情况下，GPX引脚将作为第二个中断引脚(还有INT引脚)，它与INT引脚的配置(电平或边沿，边沿极性)相同。关于SEPIRQ位的更多信息，请参阅第53页。

HOST

含义： MAX3421E的主机或外设模式

模式： **外设模式和主机模式**

CPU设置HOST = 1时，MAX3421E工作在USB主机模式。

CPU设置HOST = 0时，MAX3421E工作在USB外设模式(缺省)。

编程要点

上电复位或通过RES#引脚复位后，MAX3421E的HOST位置为0，缺省为外设模式。在该模式下，MAX3421E的工作类似于MAX3420E外设控制器。

CPU设置HOST = 1时，MAX3421E作为主机工作。其寄存器组如表1所示。

当HOST = 0时的编程信息可查阅*MAX3420E编程指南*。为便于参考，表2列出了外设和主机模式下所有的MAX3421E寄存器位。

HRSL 寄存器
HRSLT[3:0]
SNDTOGRD, RCVTOGRD

含义： 这些位指示主机传输的结果：

- **HRSLT[3:0]** 表示结果代码。
- **SNDTOGRD** 和 **RCVTOGRD** 分别表示 OUT 传输和 IN 传输产生的数据触发值。

模式： **仅用于主机模式**

SIE置位和清除这些位。

编程要点

SIE在完成主机传输时更新这些位。在触发HXFRDNIRQ之后，或在轮询HRSL寄存器并读取一个非hrBUSY值的HRSLT值之后，CPU读取寄存器HRSL。HRSLT各个位指示主机传输的结果，如**表6**所示。

表 6. HRSLT[3:0]代码

HSRLT	Label	Meaning
0x00	hrSUCCESS	Successful Transfer
0x01	hrBUSY	SIE is busy, transfer pending
0x02	hrBADREQ	Bad value in HXFR reg
0x03	hrUNDEF	(reserved)
0x04	hrNAK	Peripheral returned NAK
0x05	hrSTALL	Perpheral returned STALL
0x06	hrTOGERR	Toggle error/ISO over-underrun
0x07	hrWRONGPID	Received the wrong PID
0x08	hrBADBC	Bad byte count
0x09	hrPIDERR	Receive PID is corrupted
0x0A	hrPKTERR	Packet error (stuff, EOP)
0x0B	hrCRCERR	CRC error
0x0C	hrKERR	K-state instead of response
0x0D	hrJERR	J-state instead of response
0x0E	hrTIMEOUT	Device did not respond in time
0x0F	hrBABBLE	Device talked too long

SNDTOGRD (针对OUT传输)和RCVTOGRD (针对IN传输)表示传输的数据触发值。只要CPU完成了对同一端点的连续传输操作，那么CPU就应该读取并存储这些值。以便下次向同一端点传输数据时，CPU可恢复其触发值。CPU使用寄存器HCTL中的SNDTOG0/1 和RCVTOG0/1 位来初始化端点触发值(第59页)。

HUBPRE

含义： 通过USB集线器向LS设备发送PRE PID。

模式： **仅用于主机模式**

CPU置位和清除该位。

编程要点

如果主机固件(在枚举期间)检测到它正通过USB集线器与低速外设会话，将设置HUBPRE = 1。这会使SIE在发送每个低速包之前先发送全速的PRE PID，并且按照USB规范完成必要的信号传输。

HXFR 寄存器
EP[3:0]
HS, ISO, OUTNIN, SETUP

含义： CPU对该寄存器进行写操作以启动主机传输。

模式： **仅用于主机模式**

CPU置位和清除这些位。

编程要点

该寄存器对装载操作敏感。也就是说，当CPU对其进行写操作时，SIE会启动一个传输。CPU对其装载的数值如**表7**所示，用于启动不同的USB传输类型。

表 7. HXFR 寄存器位设置不同的传输类型

Xfr Type	HS	ISO	OUTNIN	SETUP	hex
SETUP	0	0	0	1	10
BULK-IN	0	0	0	0	0-ep
BULK-OUT	0	0	1	0	2-ep
HS-IN	1	0	0	0	8-ep
HS-OUT	1	0	1	0	A-ep
ISO-IN	0	1	0	0	4-ep
ISO-OUT	0	1	1	0	6-ep

BULK-IN和BULK-OUT适用于BULK或INTERRUPT端点的传输。这两种传输类型相同，唯一的区别是主机固件调度的方式不同。

“hex”列的“-ep”字段代表EP[3:0]的值。

关于每种传输类型的更多信息，请参阅第10页的“主机传输编程”。

HXFRDNIRQ, HXFRDNIE

含义： **HXFRDNIRQ：** 主机传输完成中断请求
HXFRDNIE： 主机传输完成中断使能

模式： **仅用于主机模式**

SIE完成主机传输时，置位HXFRDNIRQ位。CPU通过向HXFRDNIRQ位写1，来清除该位。

CPU可置位和清除HXFRDNIE位。HXFRDNIE = 1时，HXFRDNIRQ使能为可激活INT引脚的中断源。

编程要点
当触发HXFRDNIRQ中断时，CPU读取寄存器HRSL (第36页) 中的HSRLT位，以确定主机传输的结果。

IE

含义： 使能INT引脚。

模式： **外设模式和主机模式**

CPU将该位置位以激活INT输出引脚。INT输出引脚的特性通过INTLEVEL、POSINT和PULSEWID[1:0]位确定(第41页)。

CPU清除该位，则禁止INT输出引脚。

编程要点

IE = 0时，INT引脚的状态无效(电平模式时为高阻，下降沿时为高，上升沿时为低)。

内部IRQ位的工作独立于IE位的状态。IE位仅控制INT输出引脚是否激活。

INTLEVEL
POSINT
PULSEWID1, PULSEWID0

含义： **INTLEVEL**： 将INT输出引脚设置为电平有效(1)或边沿有效(0)。
POSINT： 边沿有效INT引脚的边沿极性。
PULSEWID： 边沿模式下，这两位设置IRQ的无效时间(如下所示)。

模式： 外设模式和主机模式

INTLEVEL
CPU置位INTLEVEL位时，设置INT输出引脚为电平有效。INTLEVEL = 1时，如果有一个或多个使能的中断处于悬挂状态，则驱动INT引脚为低电平。在该模式下，INT引脚为开漏极输出，因此系统必须接上拉电阻至VL。

CPU清除INTLEVEL位时，INT引脚为边沿有效。INTLEVEL = 0时，通过POSINT位设置边沿极性。在边沿输出模式下，以推挽方式驱动INT引脚，因此无需接上拉电阻到VL。

POSINT
只有当CPU设置INT引脚为边沿有效(INTLEVEL = 0)时，该位才有效。当POSINT = 1时，INT引脚采用上升沿指示悬挂的中断请求，当POSINT = 0时，INT引脚采用下降沿指示悬挂的中断请求。

PULSEWID1[0]
只有当CPU设置INT引脚为边沿有效(INTLEVEL = 0)时，这些位才有效。当CPU清除一个IRQ位时，如果仍然有一个或多个悬挂的中断(**图8**)，则会重新触发因前一次清除IRQ位而变为无效状态的INT引脚，INT引脚重新触发前的无效时间间隔是由这些位控制的。

编程要点

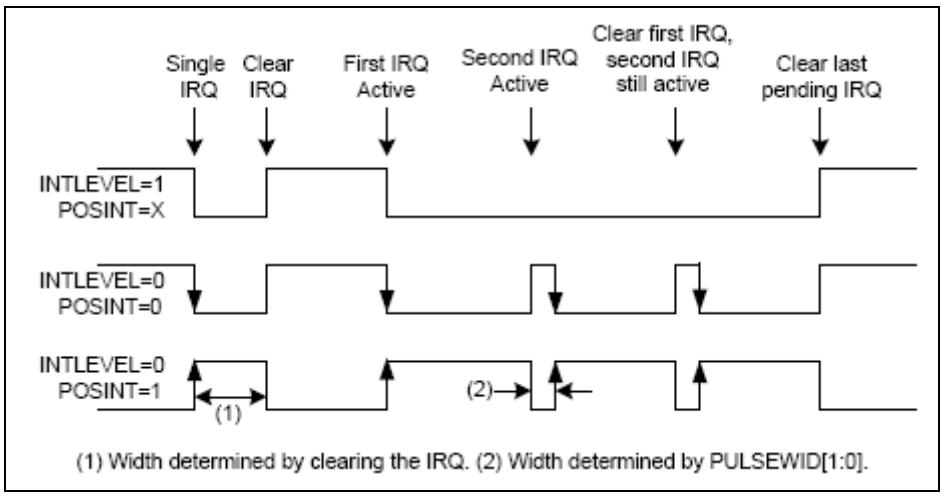


图 8. INT 引脚的动作取决于 INTLEVEL 位和 POSINT 位

图8给出的波形说明了INT引脚在不同INTLEVEL和POSINT设置下的动作。

在电平模式下(最上方的图)，INT引脚保持低电平直到没有悬挂的中断请求。INTLEVEL模式下INT是开漏极输出，因此INT引脚需要接上拉电阻至VL。

在边沿模式下，只要有新的中断请求产生，或者有其他悬挂的中断时清除了某个中断请求位，INT引脚都会产生一个跳沿。边沿模式下，如果一个中断处于悬挂状态，而另一个中断被清除，INT引脚会暂时变为无效状态，然后再次变为有效状态并产生跳沿。如***图8***中的(2)所示，无效时间可编程设置为下表所列的4个值。

PULSEWID1	PULSEWID0	INT Pulse Width uS
0	0	10.6
0	1	5.3
1	0	2.6
1	1	1.3

注意：MAX3420E 没有 PULSEWID[1:0]位。MAX3420E 的时间固定为 10.6µS

SEPIRQ = 1 时，MAX3421E与GPIN引脚相关的 8 个中断从INT引脚脱离出来。然后当GPX[B:A] = 10 时，MAX3421E将这些中断切换至GPX引脚，该引脚作为第二个中断输出引脚。GPX引脚工作于这种方式时，其作为INT输出引脚的特性同样是由INTLEVEL、POSINT和PULSEWID[1:0]的设置决定的。

LOWSPEED

含义： 设置主机为低速工作模式

模式： **仅用于主机模式**

CPU置位和清除该位。

编程要点
CPU置位该位，则使能低速USB主机。当CPU发现有外设插入(激活CONDETIRQ)并且静态总线处于D+ = 0、D- = 1的状态时，CPU通常将该位置位。

总线状态如下所示：

- LOWSPEED = 1 并检测到 J 状态。
- LOWSPEED = 0 并检测到 K 状态。

有关检测总线状态的信息，请参考第50页。

OSCOKIRQ, OSCOKIE

含义: **OSCOKIRQ**：振荡器正常中断请求
OSCOKIE：振荡器正常中断使能

模式：外设模式和主机模式

内部OSCOK信号用于指示内部12MHz振荡器和48MHz PLL处于稳定状态并且芯片准备就绪。当OSCOK信号发生从0到1的跳变时，SIE置位OSCOKIRQ位，指示芯片已经做好准备。CPU通过写1来清除OSCOKIRQ位。

CPU可置位和清除OSCOKIE位。OSCOKIE = 1时，OSCOKIRQ使能为可激活INT引脚的中断源。

编程要点
一旦CPU通过复位芯片(设置CHIPRES = 1，然后设置CHIPRES = 0)停止了MAX3421E的振荡器，CPU在继续工作之前需要等待OSCOKIRQ置位。

PERADDR 寄存器

含义： 包要发送的外设地址。

模式： **仅用于主机模式**

CPU写该寄存器；SIE读取该寄存器。

编程要点

CPU装载寄存器HXFR后，SIE从该寄存器提取外设地址来发送一个令牌包。如果CPU只与一个外设地址会话，SIE可初始化一次该寄存器，实现所有与该外设进行的传输。在枚举过程中，CPU通常向设备发送Set_Address请求，然后向该寄存器加载所需的地址。

PWRDOWN

含义: 使MAX3421E掉电

模式: **外设模式和主机模式**

CPU置位PWRDOWN位将使器件进入低功耗状态，清除PWRDOWN位可恢复工作。

编程要点

尽管在主机模式下可以访问该位，但是该位只是为外设模式而设计。当作为主机工作时，CPU决不能设置POWERDOWN = 1。

RCVBC 寄存器

含义： 接收FIFO字节计数寄存器

模式: **仅用于主机模式**

编程要点

从总线向寄存器RCVFIFO装载数据包后，SIE用接收到的字节数更新该寄存器，并触发INDAVIRQ位。

CPU读取完由寄存器RCVBC给出数目的字节数据后，向RCVDAVIRQ位写1以清除该位，为USB访问准备好空缓冲区。

RCVDAVIRQ, RCVDAVIE

含义： **RCVDAVIRQ：** 接收FIFO数据有效中断请求
RCVDAVIE: 接收FIFO数据有效中断使能

模式： 仅用于主机模式

当新的外设数据作为主机IN请求的结果出现在寄存器RCVFIFO中时，SIE对RCVDAVIRQ位置位。CPU收到中断请求后，清除RCVDAVIRQ位，读取寄存器RCVBC中的字节数，并连续读取寄存器RCVFIFO (R1)以提取数据。只有当SIE向外设发送ACK握手时才中断，否则SIE会不停地重试(由于PID，CRC，数据触发或超时错误)。

CPU可置位和清除RCVDAVIE位。当RCVDAVIE = 1时，RCVDAVIRQ使能为可激活INT引脚的中断源。

编程要点
一旦发生数据传输错误，将触发 HXVRDNIRQ 报警，但不触发 RCVDAVIRQ。

RCVFIFO 寄存器

含义: 接收FIFO。

模式: **仅用于主机模式**

外设在总线上发送数据以响应主机IN请求时，SIE向内部FIFO中载入数据。CPU通过重复读取寄存器RCVFIFO从FIFO中读取数据字节。

CPU决不能对寄存器RCVFIFO进行写操作，因为这样会损坏接收的数据。

编程要点

当数据包无差错地到达后，SIE向寄存器RCVBC中装载接收到的字节数，并触发RCVDAVIRQ位(接收数据有效IRQ)。响应中断时，CPU首先读取寄存器RCVBC以确定RCVFIFO中的字节数，然后清除RCVDAVIRQ位，最后通过反复读取寄存器RCVFIFO获取字节数据。

寄存器RCVFIFO与两个内部64字节的FIFO相连。当CPU读取一个FIFO的同时，双FIFO允许SIE并发地向另一个FIFO装入外设发送的IN数据。如果CPU清除RCVDAVIRQ位的同时，另一个FIFO中也有收到的包，SIE将立即重新触发RCVDAVIRQ位。

只有当RCVDAVIRQ = 1指示USB接收到有效数据时，CPU才可以读取RCVFIFO中的字节。

REVISION 寄存器

含义： MAX3421E版本号

模式： **外设模式和主机模式**

该只读存储器指示芯片的版本号，访问Maxim网址可获得最新的版本信息，对该寄存器进行写操作无效。

RWUIRQ, RWUIE

含义： **RWUIRQ：** 远程唤醒中断请求。
RWUIE： 远程唤醒中断使能。

模式： **仅用于主机模式**

当SIE接收到来自外设的远程唤醒信号时，将置位RWUIRQ位。CPU通过对RWUIRQ位写1来清除该位。

CPU可置位和清除RWUIE位。当RWUIE = 1时，RWUIRQ使能为可激活INT引脚的中断源。

编程要点
在CPU通过设置SOFKAEN = 0挂起总线信令后，具备远程唤醒(RWU)功能的外设可触发RWU总线状态，以请求主机恢复总线工作。当SIE检测到对应RWU信号的10ms K状态，以及随后的EOP时，它将触发RWUIRQ位。

SAMPLEBUS
JSTATUS, KSTATUS

含义： 采样USB总线的状态。

SAMPLEBUS： 采样总线状态。
JSTATUS, KSTATUS： 指示总线状态。

模式： **仅用于主机模式**

CPU置位SAMPLEBUS位，命令SIE对D+和D-线路状态进行采样。完成该操作后，SIE清除SAMPLEBUS位。

JSTATUS和KSTATUS是只读位，由SIE置位和清除该位。

编程要点
在两种情况下更新JSTATUS位和KSTATUS位：

1. CPU 设置 SAMPLEBUS = 1。
2. 触发 CONDETIRQ。

第二种情况表明器件的插入和拔出事件。响应CONDETIRQ中断时，CPU应该读取JSTATUS位和KSTATUS位以确定发生了哪个事件。

JSTATUS 位和 KSTATUS 位的编码如表 8 所示。

表 8. JSTATUS 位和 KSTATUS 位的编码

JSTATUS	KSTATUS	Meaning
0	0	SE0
0	1	K
1	0	J
1	1	N/A

表 8 中的第 4 项是未定义的 USB 总线情况。

注意：J、K 状态的含义取决于 LOWSPEED 位的设置。当 LOWSPEED = 0 时，J 表示 D+为高，而 D-为低；当 LOWSPEED = 1 时，J 表示 D+为低，而 D-为高。

SEPIRQ

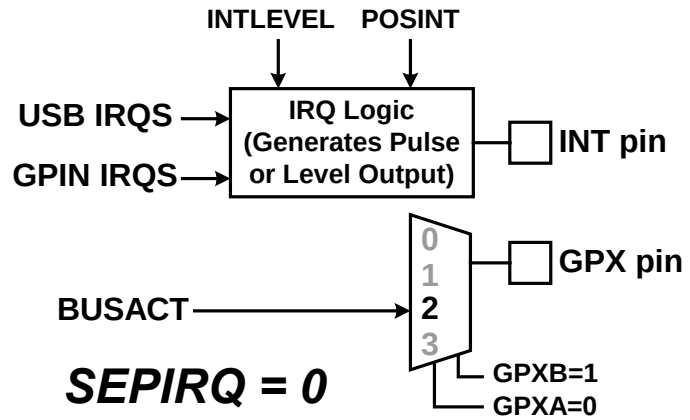
含义： 在单独引脚(GPX)上提供GPIN IRQ。

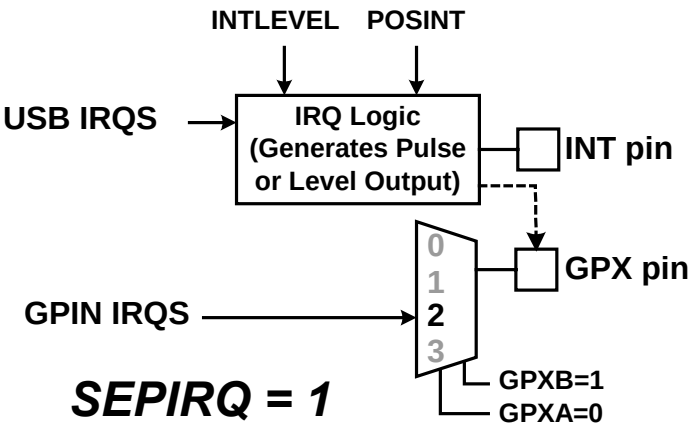
模式： **外设模式和主机模式**

CPU可置位和清除这些位。

编程要点

只要内部中断、USB或GPIN中有任一中断触发(假定相应的IRQ被使能，并且IE = 1)，MAX3421E的INT引脚将变为有效状态。下图中的第一个图给出了当SEPIRQ = 0时的缺省情况。





当 CPU 设置 SEPIRQ = 1 时，8 个 GPIN IRQ 信号将从 INT 引脚的中断源中分离出来，而被切换至引脚 GPX，作为 MAX3421E 的第二个中断引脚(下图)。GPX[B:A]设置为 2 时通常将 BUSACT 信号连接至 GPX 引脚。

设置 SEPIRQ = 1 时，GPX 引脚的 BUSACT 信号由指示 8 个 GPIN 中断的一路信号取代。下图中的虚线表明，当 GPX 引脚作为第二个中断引脚时，其特性(由 INTLEVEL 和 POSINT 确定)与 INT 引脚是一样的。

SIGRSM

含义： 为总线恢复事件发送信号。

模式： **仅用于主机模式**

CPU置位该位，为总线恢复事件发送信号。

SIE清除该位表明已经完成恢复信令。

编程要点

通过停止总线活动，USB主机挂起外设，并使其进入低功耗状态。CPU通过设置SOFKAENAB = 0来实现该过程，即阻止MAX3421E在每毫秒内自动生成全速的SOF包或低速的持续有效脉冲。

主机通过发送恢复信号来恢复总线活动，该信号由20ms的K状态组成，其后紧跟一个低速的EOP包。CPU先设置SIGRSM = 1，随后检查SIGRSM = 0，即可实现此功能。然后CPU设置SOFKAENAB = 1，重启毫秒帧标记。

CPU也可使用BUSEVENT的中断请求位BUSEVENTIRQ (第19页)，来检测MAX3421E的恢复信令是否结束。当SIGRSM位发生从 1 到 0 的跳变时，SIE将置位该中断请求位。

SNDBAVIRQ, SNDBAVIE

含义: 发送缓冲区可用中断请求
发送缓冲区可用中断使能

模式: **仅用于主机模式**

SIE向外设发送SNDFIFO中的数据后，并产生以下两种情况时将置位SNDBAVIRQ：

1. 外设以 ACK 握手包应答。
2. 无底层错误(有效 PID，CRC，EOP 和填充)发生。

CPU通过写寄存器SNDBC来清除SNDBAVIRQ位。

CPU可置位和清除SNDBAVIE位。当SNDBAVIE = 1时，SNDBAVIRQ使能为可激活INT引脚的中断源。

编程要点
在任何主机传输结束时，SIE清除其FIFO指针。因此，如果在IN传输之后主机读到一个非零的HRSLT (例如，当外设返回一个NAK握手时，HRSLT[3:0] = 0x04)，CPU只需向寄存器HXFR中重载适当的值，即可重试IN传输。由于FIFO中的数据一直保留，直到数据被重写，并且SIE FIFO指针每次传输结束时自行复位，因此主机无需每次重载FIFO数据，就能以这种方式重复发送相同的SNDFIFO数据。

CPU通过对寄存器SNDBC进行写操作，来清除SNDBAVIRQ。**CPU决不能直接清除SNDBAVIRQ位**。

SNDBC 寄存器

含义： 发送FIFO字节计数寄存器。

模式： **仅用于主机模式**

CPU装载该寄存器以指明载入SNDFIFO的字节数，并使FIFO在总线上发送OUT数据。

编程要点
CPU加载寄存器SNDBC时，SIE清除SNDBAVIRQ位。如果第二个FIFO可用，SIE将立即重新触发SNDBAVIRQ位。

不同于RCVDAVIRQ，RCVDAVIRQ是在CPU卸载完FIFO后由CPU直接清除，而SNDBAVIRQ必须通过CPU写寄存器SNDBC来清除。**CPU决不能直接清除SNDBAVIRQ位。**

SNDFIFO 寄存器

含义： 发送FIFO。

模式： **仅用于主机模式**

通过重复写寄存器SNDFIFO，CPU向内部FIFO加载OUT传输的数据。

编程要点
CPU向FIFO写入64个字节数据后，如果再读取寄存器SNDFIFO，那么可回读其数据。

在加载SNDFIFO之后，CPU将加载的字节数写入寄存器SNDBC (发送字节数)。当CPU向字节计数寄存器写入数据时，SIE清除SNDBAVIRQ (发送缓冲区有效IRQ) ，并且FIFO投入USB传输。

寄存器SNDFIFO连接至两个内部64字节的FIFO。SIE在USB总线上发送一个FIFO数据时，双FIFO机制允许CPU同时向另一个FIFO加载OUT数据。如果CPU写寄存器SNDBC时，而另一个缓冲区可用，则SIE清除SNDBAVIRQ位后，随即再次触发该位，以表明第二个缓冲区可用。

只有当SEND缓冲区可用时，即SNDBAVIRQ = 1时，CPU才可加载SNDFIFO。

SNDTOG1, SNDTOG0
RCVTOG1, RCVTOG0

含义： 设置或清除数据传输的数据触发。

模式： **仅用于主机模式**

CPU向这些位对中的一位写入1，来初始化数据传输的数据触发值。

向这些位写0无效，向任一位对中的两位同时写1也无效。

编程要点

MAX3421E包含两个数据触发器，在SNDFIFO和RCVFIFO数据传输期间，用来实现USB信令协议。向一个端点传输数据之前，CPU使用这些位为端点初始化数据触发值。该端点的数据传输完成之后，CPU读取SNDTOGRD或RCVTOGRD位(第36页)的触发值，并将该值存储到CPU的本地存储器。对于多个端点，CPU保留一组触发值，每一位对应一个端点。

当CPU返回到某个端点，传输更多数据时，首先，CPU必须用该端点上次所保存的值初始化数据触发值。CPU通过SNDTOG1/0和RCVTOG1/0位将数据触发设置为该端点上次所保存的值。

对同一端点的连续传输中，SIE维持数据触发值。只有当切换端点时，CPU才需要保存和重新初始化触发值。

欲了解更多数据触发的信息，请参考**关于MAX3421E的数据触发**(第11页)。

SOFKAENAB

含义： 使能自动生成全速SOF包或低速持续有效脉冲。

模式: **仅用于主机模式**

CPU可置位和清除该位。

编程要点

CPU设置SOFKAENAB = 1时，SIE自动生成1ms帧标记。如果LOWSPEED = 0，SIE生成SOF 包。如果LOWSPEED = 1，SIE生成持续有效脉冲。

SOFKAENAB位置位1ms后，启动SOF或KA脉冲。当SIE正在生成帧标记时，CPU设置SOFKAENAB = 0，则SIE在关闭帧标记之前结束信令。

SOF 包

SIE每毫秒发送一个SOF PID、内部11位帧计数器的数值以及CRC5。包发送完后，帧计数器递增并触发FRAMEIRQ。SIE并不更新寄存器HRSL的任何HRSLT位，因此CPU有足够的时间去读取上一次传输的结果。

帧计数器的上电缺省值为 0。通过设置FRMRST = 1 (第31页)，CPU可在任意时刻复位帧计数器。

持续有效脉冲

当SOFKAENAB = 1，并且MAX3421E作为低速主机工作时，SIE每毫秒产生一个持续有效脉冲，该脉冲由一个低速EOP组成。

SUDFIFO 寄存器

含义： 设置数据FIFO寄存器。

模式： **仅用于主机模式**

CPU对寄存器SUDFIFO进行8次写操作，向内部8字节FIFO加载SETUP包要包含的数据。

编程要点
SUDFIFO没有相关的字节计数寄存器，因为其有效载荷总是8个字节。

如果CPU向这个寄存器写入8个字节后，读取该寄存器可回读写入的FIFO字节。

SUSDNIRQ, SUSDNIE

含义： **SUSDNIRQ：** 挂起操作完成IRQ。
SUSDNIE： 挂起操作完成IE。

模式： 仅用于主机模式

SIE触发SUSDNIRQ位指示3ms的总线挂起时间。这通常是在CPU设置SOFKAENAB = 0之后的3ms产生的。CPU通过向SUSDNIRQ位写入1来清除该位。

CPU可置位和清除SUSDNIE位。SUSDNIE = 1时，SUSDNIRQ使能为可激活INT引脚的中断源。

VBUSIRQ, VBUSIE
NOVBUSIRQ, NOVBUSIE

含义：

VBUSIRQ：	V_{BUS}存在中断请求
VBUSIE:	V_{BUS}存在中断使能
NOVBUSIRQ:	无V_{BUS}中断请求
NOVBUSIE:	无V_{BUS}中断使能

模式： **外设模式和主机模式**

当内部V_{BUS}比较器的输出发生从0到1 (VBUSIRQ)或从1到0 (NOVBUSIRQ)的跳变时，SIE置位VBUSIRQ和NOVBUSIRQ位。该比较器代表了VBCOMP (V_{BUS}比较器)引脚上的电压。CPU通过向VBUSIRQ位和NOVBUSIRQ位写入1来清除这些位。

CPU可置位和清除VBUSIE和NOVBUSIE位，置位后使能对应的IRQ位，以激活INT引脚。

编程要点

这些位在外设和主机模式下均有效。外设模式下，这些位可用于自供电设计，以检测插入和拔出USB主机的事件。主机模式下，通常不需要V_{BUS}检测位，因为主机向USB连接器提供(并控制) V_{BUS}电源。这种情况下，VBCOMP引脚可用作通用输入，对应不同的跳沿极性，即从0到1的跳变(VBUSIRQ)或从1到0的跳变(NOVBUSIRQ)，分别提供独立的中断。VBCOMP引脚通过内部下拉电阻(约100kΩ)弱下拉到地，因此当采用VBCOMP引脚作为通用输入引脚时，不需要终端电阻。

在一些MAX3421E的应用中，可能同时提供A型(主机)和B型(外设) USB连接器，根据哪个连接器接有电缆(A型连接器至USB外设，B型连接器至USB主机)，要自动配置MAX3421E。这种情况下，可由本地5V电源给A型连接器的V_{BUS}引脚供电，并且接通MAX3421E的D+和D-下拉电阻，以检测外设是否插入A型连接器。主机设计应在A型连接器的V_{BUS}引脚连接诸如MAX4793的限流器/电流检测器/电流开关。B型连接器的V_{BUS}引脚可连接至MAX3421E的VBCOMP引脚。这样电路就能检测V_{BUS}是否存在，进而检测出MAX3421E何时插入USB主机。采用这样的处理方法，系统即可将自身配置为外设或主机。

USB总线复位时，由于大部分的中断使能位均被清除，因此在处理USB总线复位时，作为处理任务的一部分，应该调用一个打开中断使能位的初始化程序。