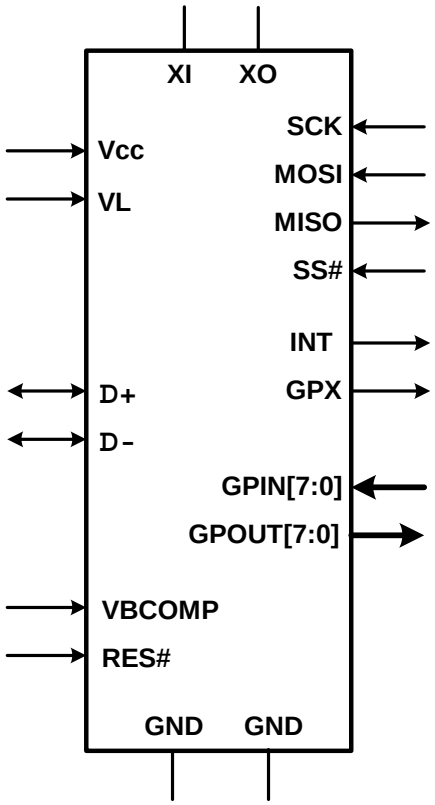
MAX3421E
Programming Guide

USB Peripheral and Host Controller with SPI Interface

Abstract: *The MAX3421E adds USB host or peripheral capability to any system with an SPI interface. This Programming Guide describes every register and bit for host operation. For operation of the MAX3421E as a peripheral, consult the MAX3420E Programming Guide.*



For more information on the MAX3421E, please visit: www.maxim-ic.com/max3421e
For more information on USB and Maxim's USB products, see: www.maxim-ic.com/usb
SPI is a trademark of Motorola, Inc.
The Maxim logo is a registered trademark of Maxim Integrated Products, Inc.
The Dallas Semiconductor logo is a registered trademark of Dallas Semiconductor Corp.

June 2006

About This Programming Guide

The MAX3421E is a dual-role USB controller, which can be programmed either as a USB peripheral or host. As a peripheral, the MAX3421E's operation is identical to the MAX3420E peripheral-only part. If you plan to use the MAX3421E only as a USB peripheral that runs existing MAX3420E code, keep the HOST bit set to 0 (the default), and consult the *MAX3420E Programming Guide* for programming details.

If you plan to use the MAX3421E only as a USB peripheral but want to take advantage of the new peripheral features, read about the added registers R21 to R24 and the added bits PULSEWID1/0, SEPIRQ, and HOST in this *Programming Guide*. Then consult the *MAX3420E Programming Guide*.

Most MAX3421E users will want programming details for host operation, which is the purpose of this guide. It is, therefore, quite logical to present only the bits and registers that apply to host operation (**Table 1**). For convenient reference, however, **Table 2** shows all the register bits for both host and peripheral operation.

Table 1 serves as a navigation tool if you are viewing this document in electronic form. This table contains links to document pages that describe the linked registers and bits.

Table 1. MAX3421E HOST Registers (Host Bit = 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

Setting the HOST bit (R27 bit 0) to 1 changes the MAX3420E register map in three ways. It redefines registers R0 through R7, clears certain bits that do not apply to host operation, and adds registers 25, 26, and 28 through 31. Bits shown with 0 in **Table 1** above should be written with 0 in host mode. For clarity, the unused peripheral registers and bits are shown with zeros.

Table 2. All MAX3421E Register Bits, for both PERIPHERAL and HOST Modes

(Shaded bits do not change during mode (HOST bit) changes.)

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

Terminology

CPU

The MAX3421E SPI™ interface can be driven by any SPI master, for example a microcontroller, DSP, or ASIC. For simplicity, this *Programming Guide* refers to the SPI master attached to the MAX3421E SPI port as a CPU.

SIE

SIE refers to the Serial Interface Engine, a logic unit inside the MAX3421E that manages all USB activities. To clarify which entity—CPU or MAX3421E—sets or clears a register bit, this document uses the terms quite specifically and consistently. SIE indicates that the MAX3421E sets or clears the bit; CPU indicates that the SPI master attached to the MAX3421E sets or clears the bit.

Mode

The MAX3421E operates in two modes, peripheral or host.

- Peripheral-only bits are documented in the *MAX3420E Programming Guide*.
- Bits marked **host and peripheral** are active in both modes.
- Bits marked **host only** are valid only in host mode.

ISO

ISO is an abbreviation for ISOCHRONOUS, one of the four USB transfer types. When acting as a USB host, the MAX3421E supports full-speed ISO transfers. It does not support ISO transfers as a peripheral.

Resets

The MAX3421E has three reset sources:

1. An internal Power-On-Reset (POR) circuit
2. A RES# pin
3. A register bit called CHIPRES

There is a fourth way in which a reset can occur. When the HOST bit changes state (switching to host or peripheral operation), the SIE clears certain register bits.

The following section explains the effect of each reset source.

MAX3421E register bits are clocked from two sources:

1. The internal 12MHz oscillator
2. The SCLK signal that the CPU supplies to the SPI interface.

Asserting the RES# or PWRDOWN reset stops the internal 12MH clock. Most register bits are asynchronously cleared. However, the register bits that are clocked from the SPI interface remain active so the CPU can control the SPI configuration (e.g., the FDUPSPI bit), USB bus pulldown resistors, and the state of the PWRDOWN bit.

Power-On Reset

The SIE clears every register bit. Once out of reset, the SIE sets the following bits to indicate available buffers:

- IN3BAVIRQ (peripheral)
- IN2BAVIRQ (peripheral)
- IN0BAVIRQ (peripheral)
- SNDBAVIRQ (host)

Note: POR sets HOST = 0, defaulting to peripheral operation.

Asserting the RES# Pin or Setting CHIPRES = 1

These resets stop the internal 12MHz oscillator and clear most register bits, but leave the SPI-clocked register bits undisturbed and still accessible by the CPU. The SPI-clocked register bits are:

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

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

Note: These resets set HOST = 0, selecting peripheral operation.

Changing the MODE Bit

Peripheral to Host

The SIE clears register bits that do not apply to host operation. The CPU should not write these bits with anything but 0 when HOST = 1.

- EPSTALLS register
- CLRTOGS register
- EPIRQ register
- EPIEN register
- USBIRQ register, except the VBUSIRQ and NOVBUS IRQ bits, which remain active as a host
- USBIEN register, except VBUSIRQ and NOVBUS IRQ bits, which remain active as a host
- EP3/2/1INAK bits in the PINCTL register

Host to Peripheral

The CPU should revert from Host to Peripheral mode by asserting the CHIPRES bit, which clears bits according to the rules listed above and sets the HOST bit to zero.

Accessing the MAX3421E Registers

An SPI master controls the MAX3421E by writing and reading 21 internal registers, R0 through R20. The SPI master begins every register access by asserting the MAX3421E SS# (slave select, active low) pin, and clocking in eight bits that comprise the SPI command byte. **Figure 1** shows the command byte format.

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

Figure 1. SPI command byte. As for all SPI transfers, bit 7 is sent first. The ACKSTAT bit is valid only in peripheral mode; the MAX3421E ignores it in host mode.

Reg4:Reg0 set the register address, with valid values 0 to 31. The MAX3421E ignores Reg values above 31. The direction bit sets the direction for subsequent bytes in the transfer. The ACKSTAT bit duplicates a USB control bit (R9 bit 6), and is valid only when the MAX3421E operates as a USB peripheral. The SIE ignores the ACKSTAT bit when HOST = 1.

After sending the command byte, the SPI master transfers one or more bytes in the direction indicated by the DIR bit. Keeping SS# low, the SPI master provides additional bursts of eight SCK pulses for each byte. When the byte transfers are complete, the SPI master de-asserts SS# (drives high) and the transfer terminates.

The MAX3421E has two register types, FIFOS and control registers. Repeated reads or writes to a register have different effects, depending on the register type.

Registers R1, R2, and R4 access internal FIFOS. After selecting the register number with the command byte, the SPI master loads or unloads consecutive FIFO bytes by repeating reads or writes during the same SPI transfer (maintaining SS# low). To write 45 bytes into the SENDFIFO, for example, the SPI master would perform the following steps:

1. Set SS# = 0 to start the transfer.
2. Issue eight SCLK pulses and send the command byte 00010010. This command byte selects R2 (SNDFIFO) for a write operation (DIR = 1).
3. Issue eight SCK pulses, each clocking a data bit into the SNDFIFO register, one bit per SCK rising edge. Every byte the SIE writes the FIFO byte and advances an internal FIFO address pointer.
4. Repeat Step 3 for 44 more times, clocking in a total of 45 bytes into the SNDFIFO.
5. Set SS# = 1 to terminate the transfer.

Registers R13 through R31 are MAX3421E control registers. If the SPI master repeatedly reads or writes R13 through R20 during the same SPI transfer (SS# low), every byte read or write automatically increments the internal register address. This action allows reading or writing consecutive registers without writing a new command byte to set each new register address. The register address continues to increment in this manner until R20 is reached, at which point the

register address “sticks” at R20. This feature gives the microprocessor quick access to the IO pins in R20. For example, to output a prestored waveform on a GPIO pin, the SPI master can write the command byte 10100010 (R20, Write) and then send multiple data bytes to R20 to output the waveform.

If the SPI master explicitly addresses R21 or above, the register addresses again automatically increment during the same SS# transfer until R31 is reached, at which point it “sticks” again at R31.

MISO Values for the First Eight Bits in Full-Duplex Mode

When FDUPSPI = 1, the MAX3421E operates its SPI port in full-duplex mode, meaning that data is simultaneously clocked in from the MOSI pin and out to the MISO pin on each SCLK rising edge. The first eight bits of an SPI access constitute the command byte, clocked into the MOSI pin. The MAX3421E sends status data on the MISO pin while the command byte is clocked into the MOSI pin. The status bits ,while operating in HOST mode, are shown in **Figure 2.**

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

Figure 2. MISO data when FDUPSPI (HOST = 1).

Programming Host Transfers

When HOST = 1, you need to *think* like a host. Instead of responding to host requests, you generate them. This means every packet you send out requires:

- A function address in the FNADDR register
- An endpoint number in EP[3:0] of the HXFR register
- A Packet ID (PID) in the HXFR register
- Some data: OUT data in the SNDFIFO (SNDBC bytes, to be exact), or IN data in the RCVFIFO (RCVBC bytes)

The function address and endpoint FIFO data persist in the MAX3421E until the CPU changes them. This means, for example, to resend an OUT packet (due to an error by the peripheral) the CPU does not need to load the SNDFIFO again. It just launches the same transfer by rewriting the HXFR register.

The MAX3421E does the USB housekeeping work. Set up the above bits, launch the transfer by writing the HXFR register, wait for a completion interrupt, and check the HRSLT bits (Host Result) to see transfer results. Although the MAX3421E reports 16 host result conditions, you will usually see one of the results shown in **Table 3**.

Table 3. Host Result Codes for Normal USB Operation

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

All 16 HRSL codes are shown on page36. None of the conditions in Table 3 represents electrical or signaling errors; they occur in normal USB operations. The hrBUSY result is for applications that do not use the HXFRDN interrupt request to determine when a transfer completes, but instead poll the HRSL register to check for transfer completion.

The host can launch seven transfer types, shown in **Table 4**.

Table 4. HXFR Register Bit Settings for Different Transfer Types

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

The CPU can write the HXFR register any time when the SIE is not busy with another transfer. The SIE takes care to avoid collisions with its automatically generated SOF/KA frame markers.

If the CPU writes the HXFR register late enough in a frame that the ensuing transfer would collide with the frame marker, the SIE automatically defers sending the packet until after it generates the next frame marker.

Note: A USB host transfers data over BULK and INTERRUPT endpoints using identical methods. The only difference between these two transfer types is *when* the packets are scheduled, a function of the controlling firmware. This document refers to BULK transfers with the understanding that any BULK discussion also applies for INTERRUPT transfers.

About MAX3421E Data Toggles

USB protocol tags every data packet with one of two PIDs (Packet ID's) called DATA0 and DATA1. These PIDs help detect USB errors. Associated with every endpoint is a *data toggle* value, which determines which DATA PIDs to use. The first data packet (after reset) to or from an endpoint is sent using the DATA0 PID. When both sides, the sending and receiving ends, agree that the data is accurate (by generating/receiving the ACK handshake), they both complement their toggle values. Therefore, consecutive data packets sent to or received from an endpoint normally will have *toggling* PID values—DATA0, DATA1, DATA0, etc.

The MAX3421E provides four bits to maintain the data toggles. After a data transfer to an endpoint, the SIE updates bits RCVTOGRD and SNDTOGRD (page 36) to indicate the toggle values for the selected endpoint. The CPU reads and stores these bits to re-initialize the toggle value for the same endpoint when necessary. To initialize the toggle state before transferring to an endpoint, the CPU sets one of the bit pairs SNDTOG1-SNDTOG0 for OUT data, or RCVTOG1-RCVTOG0 for IN data (page 59). Only one of these bit pairs should be set at a time.

The CPU does not need to initialize an endpoint's toggle value for multiple **consecutive** transfers to the **same** endpoint. The MAX3421E updates the data toggle value as it performs transfers to the endpoint. Only when the CPU **switches** endpoints does it need to restore the data toggle bits to the values that it saved for the endpoint.

Programming BULK-OUT Transfers

The MAX3421E sends BULK data to a peripheral using OUT packets, SNDFIFO data, and handshakes.

The CPU first checks for SNDBAVIRQ = 1 (page 56) to see if the send buffer is available for loading. (The SIE sets this interrupt request bit whenever an OUT transfer successfully completes.) If the buffer is available, the CPU writes up to 64 data bytes into the SNDFIFO (page 58) by repeatedly writing R2. Then the CPU writes the byte count (the number of bytes it loaded into the SNDFIFO) into the SNDBC register (page 57). Loading the SNDBC register causes the SIE to negate the SNDBAVAIRQ bit.

Note: If the second buffer of the double-buffered SNDFIFO is free, the SIE immediately reasserts the SNDBAVIRQ bit.

The CPU *may* need to initialize the data toggle value for the desired endpoint:

- If the transfer is to a new endpoint, the CPU initializes the data toggle to the last saved value for the endpoint.
- If the endpoint is the same as for the last transfer, the CPU does not need to initialize the data toggle value. The SIE updates the data toggle after each transfer.

Finally, the CPU loads the HXFR register with the value 0010eeee (Table 4) where eeee is the endpoint number to which it is sending the data. The HXFR register is load-sensitive, which means that when the CPU loads the HXFR register, the SIE initiates the transfer.

The SIE sends an OUT token, the address in the PERADDR register, the endpoint number in EP[3:0], a CRC5, and EOP. The SIE immediately follows this by sending a DATA0 or DATA1 PID (depending on the state of the toggle bit), SNDBC bytes from the SNDFIFO, and a CRC16. Then the SIE waits 6.5 bit times for the peripheral to respond.

Note: If SNDBC = 0, the SIE sends no data bytes, but does send the DATA0/1 PID.

When the SIE receives a handshake or bus timeout, it sets HXFRDNIRQ = 1 and indicates the results in the HRSLT[3:0] bits (page 36).

If the peripheral returns the ACK handshake, the SIE also complements the data toggle, indicates a successful transfer (HRSLT[3:0] = 0000), and asserts the SNDBAVIRQ indicating “Send Buffer Available.”

If the peripheral did not ACK the transfer, the data may need to be resent. For example, a NAK handshake is a common response used by a peripheral to indicate that it is not ready to accept the data. The CPU discovers the cause of a non-ACK’d OUT transfer by examining the HRSLT bits. The CPU relaunches the OUT transfer simply by rewriting HXFR = 0010eeee. The SIE uses the same values of PERADDR and SNDBC to send the same data in the SNDFIFO.

Programming BULK-IN Transfers

The CPU issues an IN token to request a peripheral to send it BULK data . Then the SIE transfers data into its RCVFIFO, and ACKS the transfer.

The CPU writes HXFR = 0000eeee (Table 4) to initiate the IN transfer, where eeee is the desired endpoint address.

The SIE sends an IN token, the address in the PERADDR register, the endpoint number in EP[3:0], and a CRC5. It then waits 6.5 bit times for the peripheral to respond. If the peripheral responds with a DATA0 or DATA1 PID followed by data, the SIE loads the received data bytes into the RCVFIFO and counts the bytes. At the end of the packet the SIE checks the packet for errors, updates the RCVBC register and HRSLT bits, and then asserts the HXFRDNIRQ bit. Depending on the transfer outcome, the SIE may or may not assert the RCVDAVIRQ.

If the IN data was error-free (HRSLT = 0000), the SIE sends an ACK token, complements the data toggle, and asserts the RCVDAVIRQ to indicate that new IN data is valid.

If the IN data was error-free but there was a data toggle mismatch (the DATA0 or DATA1 PID send by the peripheral did not match the endpoint toggle value), the SIE sends the ACK handshake, but it does not complement the data toggle or assert the RCVDAVIRQ. The SIE sets HRSL = 0110 (Toggle Error) for this condition. This situation would happen if the peripheral received a corrupted ACK handshake from the previous IN transfer. In this case the host ignores the data in the RCVDATA FIFO, because it represents data that the peripheral mistakenly resent when it missed the last ACK handshake. By ACK-ing the transfer and not updating its own toggle bit, the SIE causes the peripheral to complement *its* toggle bit, thus forcing the data toggle mechanism back into sync.

If the HRSLT bits indicate a data error, the SIE does not send the ACK, complement the data toggle, or assert the RCVDAVIRQ bit.

The CPU responds to the HXFRDNIRQ indication by examining the result bits in HRSLT[3:0]. If the result is 0000 (success), the CPU reads RCVBC to determine the byte count, and then reads that number of bytes using repeated reads to the RCVFIFO register (page 48). After the CPU retrieves the data, it clears the RCVDAVIRQ bit (by writing 1 to it). If there is another buffer of IN data in the double-buffered RCVFIFO, the SIE immediately re-asserts the RCVDAVIRQ bit.

Note: The SIE does not automatically retry an IN transfer that indicates an error. Instead, it notifies the CPU (through the HXFRDNIRQ and HRSLT bits) of the error and generates the correct USB response. Usually the CPU will simply resend the IN packet by reloading the HXFR register.

Programming a CONTROL Transfer

A host sends a CONTROL transfer in two or three stages:

1. A SETUP packet, which sends 8 bytes of "op-code" to the peripheral
2. An optional data stage, usually a BULK IN request
3. A status stage

The host sends CONTROL transfers to a peripheral's default control endpoint zero.

1. Setup

The CPU writes the 8-byte SETUP data into the SUDFIFO. There is no byte count register associated with the SUDFIFO because the SETUP packet payload is always 8 bytes. Then the CPU loads the HXFR register with the value 00010000 (Table 4), which instructs the SIE to dispatch a SETUP packet to endpoint zero.

The SIE then sends a SETUP packet consisting of a SETUP PID, the address in the PERADDR register, endpoint 0000, a CRC5, and EOP, and follows with a DATA0 packet containing the 8 bytes in the SUDFIFO. The SIE waits 18 bit times for the device to respond or time out, and finally terminates the transfer by asserting the HXFRDNIRQ bit and updating the HSRLT bits (page 36). The USB spec says that a peripheral must always ACK a SETUP packet.

Note: The SIE sends fixed DATA0 and DATA1 PID tokens for the various stages of a CONTROL transfer, regardless of the setting of the internal data toggle.

---

2. Data (optional)

If a data stage is required, it is programmed as a BULK-IN or BULK-OUT transfer. Some CONTROL transfers, such as Set_Address, do not require a data stage because the command data fits into the 8 bytes in the SETUP packet.

3. Status

Unique to CONTROL transfers, the status stage provides an added measure of protection for these mission-critical transfers. A status stage consists of an IN or OUT packet in the opposite direction as the preceding stage.

A STATUS packet is a BULK-OUT or BULK-IN transfer that contains no data and which always uses the DATA1 PID. The programmer could send these transfers by programming standard BULK transfers. For convenience, however, the MAX3421E provides special HS-OUT and HS-IN launch codes for handshakes so the SIE can do the work.

HS-OUT

The host sends the OUT handshake to terminate a CONTROL-READ request such as Get_Descriptor. To send an OUT handshake packet, the CPU loads the HXFR register with the value 0xA0 (Table 4).

The SIE sends the OUT PID, the address in PERADDR, endpoint zero, and a CRC5 value. The SIE then sends a DATA1 PID, and waits 6.5 bit times for the peripheral to respond or time out.

Note that this is identical to a BULK-OUT packet with no data and a fixed DATA1 PID. When the SIE receives a handshake or bus timeout, it sets HXFRDNIRQ = 1 and indicates the results in the HRSLT[3:0] bits (page 36).

HS-IN

The host sends the IN handshake to terminate a CONTROL-WRITE request such as Set_Address. To send an IN handshake packet, the CPU loads the HXFR register with the value 0x80 (Table 4).

The SIE sends the IN PID, the address in PERADDR, endpoint zero, and a CRC5 value, then waits 6.5 bit times for the peripheral to respond. Then the SIE updates the HRSLT bits and asserts the HXFRDNIRQ bit. If the peripheral returns the DATA1 PID (with no data), the SIE automatically sends an ACK handshake indicating successful termination of the CONTROL transfer.

About ISO Transfers

USB ISOCHRONOUS transfers are characterized by on-time delivery without the handshakes that accompany the other USB transfer types. Once a USB host enumerates a device and grants its ISO bandwidth requirements (bandwidth indicates the number of bytes allocated to the ISO endpoint in every 1-millisecond frame), the USB spec requires the host to deliver or consume that number of bytes *every* frame.

This gives rise to special consideration for the MAX3421E, which may be connected to a slow SPI interface or CPU. The controller must keep pace with the scheduling requirements. If there is a speed mismatch, the MAX3421E has two ways to indicate the problem, as set by the DELAYISO bit.

Double Buffering

The MAX3421E SNDFIFO and RCVFIFO are double-buffered, meaning that each has two sets of 64-byte FIFOS and byte count registers. This double-buffering is essential for performing ISO transfers larger than one maximum packet size of 64 bytes. The double-buffer feature is invisible to the programmer:

- **INs:** If the CPU clears the RCVDAVIRQ (Receive Data Available IRQ) bit after unloading the RCVFIFO, and if there is another packet of data waiting in the other buffer, the SIE immediately re-asserts the RCVDAVIRQ bit.

- **OUTs:** If the CPU clears the SNDBAVIRQ (Send Buffer Available IRQ) by loading the SNDBC register with the number of bytes which it loaded into the SNDFIFO, and if the other buffer is available for loading, the SIE immediately re-asserts the SNDBAVIRQ bit.

Although the SNDFIFO and RCVFIFO are 64 bytes, the SIE can send and receive ISO data packets of any size (up to the USB spec limit of 1023 bytes) by concatenating data from consecutive loads/reads of these FIFOS into one large data packet, as long as the CPU supplies or consumes the data in time.

The SIE detects the end of an ISO IN data packet by detecting the EOP (End-Of-Packet) bus condition. For OUT transfers, the SIE normally detects the end of a data payload when the CPU loads a FIFO with fewer than 64 bytes. This works either for a single packet (63 or fewer bytes), or a data payload that spans multiple 64-byte buffer loads where the last buffer contains 63 or fewer bytes. However, there is a special case when the last packet of data is exactly 64 bytes. The SIE needs to know whether this represents another 64 bytes in a multibuffer load, or if it is the last 64 bytes in a transfer whose length is an exact multiple of 64 bytes. The CPU informs the SIE that a 64 byte packet represents the end of the OUT transfer by loading SNDBC = 0. This is a special signal to the SIE, which does not take its normal action of switching buffers when the SNDBC register is loaded.

Programming ISO-IN Transfers

The CPU writes HXFR = 0100eeee to initiate an ISO IN transfer.

The SIE sends an OUT token using the address in the PERADDR register, the endpoint number in EP[3:0], a CRC5, and EOP. It then waits 6.5 bit times for a DATA0 PID or bus timeout. If the SIE receives the DATA0 PID, it begins loading data into the RCVFIFO. When the FIFO fills with 64 bytes (or the end-of-packet signal is indicated on the bus), the SIE writes the byte count to RCVBC and asserts the SNDBAVIRQ, thereby indicating that a FIFO is available for the CPU to load more data if necessary. At the end of the ISO IN transfer (EOP), the SIE updates the HRSLT bits and asserts the HXFRDNIRQ.

Programming ISO-OUT Transfers

The SIE master sends an OUT packet to the peripheral, followed by a data packet using the fixed DATA0 PID. There are three cases to consider:

1. OUT data payload sizes of fewer than 64 bytes
2. Exactly 64 bytes
3. More than 64 bytes.

1. OUT Payload Fewer than 64 Bytes

The CPU programs this similarly to a BULK transfer. The CPU loads bytes into the SNDFIFO, and writes the byte count into SNDBC. Then it writes the HXFR register with 0110eeee (Table 4) to launch the transfer.

The SIE sends an OUT token using the address in the PERADDR register, the endpoint number in EP[3:0], a CRC5, and EOP. The SIE immediately follows by sending a DATA0 PID, SNDBC bytes from the SNDFIFO, and a CRC16. Then, recognizing SNDBC<64, the SIE terminates the transfer and asserts the SNDBAVIRQ. The peripheral sends back no handshake for the SIE to check. As a result, the significant error condition would be a truncated or missed packet because the CPU scheduled the transfer too late in the frame to avoid colliding with the automatically generated SOF packet. The DELAYISO bit controls SIE behavior for this error condition.

2. OUT Payload of Exactly 64 Bytes

This case applies for a single 64-byte packet, or if the final data payload of a multibuffer transfer is exactly 64 bytes. The CPU loads the SNDFIFO with 64 bytes and writes SNDBC = 64, waits for SNDBAVIRQ = 1, then writes SNDBC = 0 to indicate the end of the transfer.

OUT Payload Greater than 64 Bytes

The CPU loads the SNDFIFO with 64 bytes of data, then writes SNDBC = 64. If SNDBAVIRQ = 1, it can load the second portion of the data packet into the SNDFIFO, after which it again writes the SNDBC register with the byte count. To launch the transfer it loads HXFR register with 0110eeee.

As a FIFO becomes available, the SIE continues to assert SNDBAVIRQ, asking for the next chunk of ISO OUT data. The double-buffering allows the SIE to send OUT data from one FIFO while the CPU loads the other one.

The SIE terminates the ISO OUT transfer by updating the HRSLT bits and asserting the HXFRDNIRQ. An ISO OUT transfer terminates after the SIE transmits the last byte of a FIFO with an associated SNDBC<64, or when the CPU loads SNDBC = 0.

Active Bits in Peripheral and Host Modes

Some MAX3421E features are available for both peripheral and host operation. These features involve non-USB system aspects such as how the SPI master is configured, how the interrupt pin operates, and the state of the IO pins. These features can be grouped as follows:

- Interface Configuration
  - FDUPSPI
  - GPX[B:A]
  - INTLEVEL
  - POSINT
- IO Pin configuration and Output Values
  - GPIN pins—interrupt polarity and enables
  - GPOUT pin states
  - VBCOMP pin detected level
- Global Chip Operations
  - CHIPRES
  - PWRDOWN
  - OSCOK (Oscillator OK)
- Some IRQ Bits and the IE Bit

Table 2 shows the registers and bits that persist when the SPI master commands a MAX3421E mode change, either from peripheral to host or from host to peripheral. Therefore, a mode change does not disturb the SPI interface configuration or the GPOUT pin values. The IE bit and the GPIN interrupt bits (GPINIRQ and GPINIE) remain unchanged through a mode change. This means that if a GPIN interrupt is pending before the mode change, it will be pending after the mode change.

Note: There are two ways to command the MAX3421E to switch from HOST to PERIPHERAL operation. First, the SPI master can write a zero to the HOST bit in the MODE register R27. Second, the SPI master can set, then clear the CHIPRES bit in the USBCTL register R16. Using CHIPRES is the preferred way if you want to start with a “clean” peripheral.

BUSEVENTIRQ, BUSEVENTIE

Meaning: **BUSEVENTIRQ**: One of two bus events has occurred.
BUSEVENTIE: Enable the BUSEVENTIRQ.

Mode: **Host only**

The SIE sets the BUSEVENTIRQ bit when it completes signaling one of two USB bus events:

- Bus Reset (when BUSRST 1→0)
- Bus Resume (when BUSRSM 1→0)

The CPU clears the BUSEVENTIRQ bit by writing a 1 to it.

The CPU sets and clears the BUSEVENTIE bit. When BUSEVENTIE = 1, the BUSEVENTIRQ is enabled as a source to activate the INT pin.

Programming Notes

This interrupt is shared by two sources: completion of bus reset signaling, or bus resume signaling. As this is a shared interrupt, it is good practice to clear the BUSEVENTIRQ bit before initiating either of the two bus signals (before setting BUSRST = 1 or BUSREM = 1).

BUSRST

Meaning: Issue a Bus Reset to a USB peripheral.

Mode: **Host only**

The CPU sets this bit to initiate a 50ms bus reset (SE0) signal to the peripheral.

The SIE clears this bit at the conclusion of the bus reset signaling.
The CPU may also clear this bit to prematurely terminate the 50ms SE0 bus signal. Terminating the reset signal in this manner causes the BUSEVENTIRQ to assert.

Programming Notes

The CPU sets this bit to instruct the SIE to issue a bus reset on the D+ and D- lines. After setting this bit, the CPU can detect the end of the 50ms interval either by polling the BUSRST bit for zero, or by responding to the BUSEVENTIRQ. To program a bus reset, follow this sequence of steps:

1. Set BUSRST = 1.
2. Test for BUSRST = 0 or respond to the BUSEVENTIRQ.
3. Turn on frame markers by setting SOFKAENAB = 1.
4. Wait for at least one FRAMEIRQ.

Step 4 ensures that the host logic is ready to generate the first host transaction.

CHIPRES

Meaning: Chip Reset

Mode: **Peripheral and Host**

The CPU sets this bit to reset the chip. Its effect is identical to driving the RES# pin low.
The CPU clears this bit to take the chip out of reset.

Programming Notes

The CPU can clear this bit immediately after setting it. Resetting the MAX3421E either at power on or by setting the CHIPRES bit clears most register bits, including the HOST bit, and therefore sets up the MAX3421E to operate as a USB peripheral device.

Setting CHIPRES = 1 stops the internal oscillator. After removing the reset by setting CHIPRES = 0, the CPU should check the OSCOK interrupt request. The CPU should proceed only after this interrupt asserts, an indication that the oscillator and PLL have stabilized.

CONDETIRQ, CONDETIE

Meaning: **CONDETIRQ:** Peripheral Connect/Disconnect Interrupt Request
CONDETIE: Peripheral Connect/Disconnect Interrupt Enable

Mode: **Host only**

The SIE sets the CONDETIRQ bit to indicate that a peripheral has either plugged in or unplugged. The CPU clears the CONDETIRQ bit by writing a 1 to it. The conditions that activate the CONDETIRQ bit are:

- The bus makes a transition from the J-state or K-state to 25 microseconds of SE0. This indicates that a peripheral device has disconnected.
- The bus makes a transition from 8 SE0 bit times to 25 microseconds of the J-state or K-state. This indicates that a peripheral device has connected.

The CPU sets and clears the CONDETIE bit. When CONDETIE = 1 the CONDETIRQ is enabled as a source to activate the INT pin.

Programming Notes

To detect a peripheral connect or disconnect, the CPU should first set HOST = 1 to put the MAX3421E into host mode. It should then set DPPULLDN = 1 and DMPULLDN = 1 to establish logic low levels for the D+ and D- signals.

To determine whether connect or disconnect occurred, the CPU reads the HSRL register and examines the JSTATUS and KSTATUS bits. Normally these status bits update when the CPU sets the SAMPLEBUS bit, but they also automatically update when the CONDETIRQ interrupt asserts.

DELAYISO

Meaning: Delay data transfer to an ISOCHRONOUS endpoint until the next frame (until after the next SOF packet is transmitted). This will happen if the packet is scheduled too late in the frame to avoid colliding with the next automatically generated SOF packet.

Mode: **Host only**

The CPU sets and clears this bit.

Programming Notes

The USB spec guarantees that a host will grant an enumerated ISO endpoint enough bus bandwidth (frame time) to allow ISO data transfers in every 1ms frame. This means that the host must dispatch an IN or OUT packet (by writing the HXFR register) not only in every frame, but *early enough* in every frame to guarantee that the transfer can complete before the MAX3421E automatically generates the next SOF packet.

Since the MAX3421E can be connected to any speed SPI master, and since there is no guarantee that the controlling firmware will write the HXFR register early enough in every frame to guarantee delivery, the SIE provides the DELAYISO bit to control the behavior when a packet is scheduled too late in a frame. When there is a packet/SOF timing conflict, the DELAYISO bit determines whether the packet data or the SOF packet has priority.

If DELAYISO = 1, the SIE checks the available time left in the 1-millisecond frame whenever the CPU launches an ISO packet by loading the HXFR register. If the SIE finds that there is not enough time in the frame to send or receive a 256-byte ISO packet, it delays sending the ISO packet until the beginning of the next frame.

Note: The 256-byte window was chosen to accommodate a common ISO application, CD quality audio. In that application, 44.1 K-samples per second, times two channels, times 16 bits per channel requires 176 bytes per millisecond frame. A 256-byte window adds some margin.

ISO OUT-SOF Conflicts

The SIE has two ways to deal with a scheduling conflict between sending ISO OUT packet data and an automatically generated SOF packet. The SIE either starts to send the packet and then prematurely cuts off sending the data to generate the SOF, or it delays sending the entire packet until after it generates the next SOF packet.

Note: Another ISO error condition is a data under-run, indicated by the result code HRSLT = 0x06. A data under-run occurs if the SIE needs SNDFIFO data while sending an ISO data packet, but the CPU has not loaded the SNDFIFO in time.

ISO OUTS with DELAYISO = 0

If an ISO-OUT data packet transfer is underway when it is time for the SIE to generate the next SOF packet, the SIE truncates the data packet and generates the SOF. The SIE indicates this loss-of-data condition by setting HRSLT[3:0] = 0x03, followed by the FRAMEIRQ. It does not assert the HXFRDNIRQ in this case.

ISO OUTS with DELAYISO = 1

The SIE delays sending the OUT data packet if the CPU writes the HXFR register at a time that would not allow the SIE to transfer 256 bytes before the end of the frame. In this case the SIE sends the next SOF packet (and asserts FRAMEIRQ); it then sends the delayed ISO packet and terminates the transfer, asserting the HXFRDNIRQ with code 0x00 (hrSUCCESS). The CPU detects a deferred packet by sampling the SNDBAVIRQ bit just after the FRAMEIRQ. If SNDBAVIRQ is zero, the SNDFIFO buffer is still committed to USB transfer and the packet was therefore delayed.

ISO IN-SOF Conflicts

The SIE handles IN transfer conflicts somewhat differently than for OUTS, because the host cannot command a peripheral to stop sending data to avoid colliding with the next SOF packet. Therefore, in the conflict condition the SIE either reads the remaining ISO IN data and does not generate the SOF packet, or it delays sending the IN request until after it generates the next SOF packet.

Note: Another ISO error condition is a data over-run, indicated by the result code HRSLT = 0x06. A data over-run occurs when the SIE has ISO IN data available but nowhere to put it. This happens because no RCVFIFOS are available (the CPU has not unloaded the RCVFIFO in time.

ISO INS with DELAYISO = 0

If an ISO-IN request results in the peripheral sending a data packet that runs past the SOF generation time, the SIE continues to receive the IN data and suppresses sending the SOF packet for that frame. In this case the SIE sets HRSLT[3:0] = 0x03, does not assert the HXFRDNIRQ when the IN transfer completes, but does assert the FRAMEIRQ for that frame. The CPU can detect this error case by recognizing that a FRAMEIRQ occurred without an HXFRDNIRQ before it.

Note: The SIE always increments the frame count, whether or not an SOF packet is actually sent.

ISO INS with DELAYISO = 1

If the CPU initiates the ISO-IN transfer at a time that would not allow a 256 byte data packet to be received before colliding with the next SOF, the SIE defers sending the IN packet until after the next frame. In this case the SIE asserts the FRAMEIRQ for the next frame, sends the ISO-IN packet, transfers the IN data, and asserts the HXFRDNIRQ with HRSLT[3:0] = 0x00 (hrSUCCESS).

Note: If ISO scheduling problems are detected, the system must be modified to dispatch the ISO packets earlier in each frame, perhaps by speeding up the SPI interface, tuning the firmware, or both.

DPPULLDN, DMPULLDN

Meaning: Connect internal 15kΩ resistors from D+ and D- to ground.

Mode: **Host only**

The CPU sets and clears these bits.

Programming Notes

A USB host puts the bus into a quiescent state by connecting these two pulldown resistors. A low-speed peripheral connects a 1500kΩ resistor from D- to 3.3V; a full-speed peripheral connects a 1500kΩ resistor from D+ to 3.3V. Because the bus is weakly pulled down, the SIE can detect not only when a peripheral has plugged in, but also its speed.

If the MAX3421E is operating as a peripheral (HOST = 0), these bits should both be set to zero (their default values).

Page 63 describes the role of the DPPULLUP and DMPULLUP bits in designs that implement two USB connectors and automatically detect connection as a peripheral host.

FDUPSPI

Meaning: Full-Duplex SPI port operation

Mode: **Peripheral and Host**

The CPU sets this bit to operate the SPI port in full-duplex mode.
The CPU clears this bit to operate in half-duplex mode.

POR: FDUPSPI = 0 (half-duplex)
Chip Reset: *No change*
Bus Reset: *No change*
Pwr Down: Read-write

Programming Notes

Half-duplex SPI In half-duplex mode (FDUPSPI = 0), the MOSI (Master Out, Slave In) pin becomes a bidirectional IO pin, and the MISO (Master In, Slave Out) pin is tri-stated.

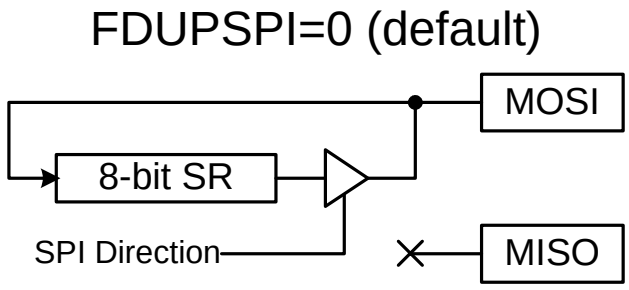


Figure 3. Half-duplex SPI interface.

Full-Duplex SPI

Full-duplex mode (FDUPSPI = 1) provides separate MOSI and MISO pins. This configuration has the added feature that while the first byte of every transfer (the command byte) is clocked in, eight bits of status information are simultaneously clocked out.

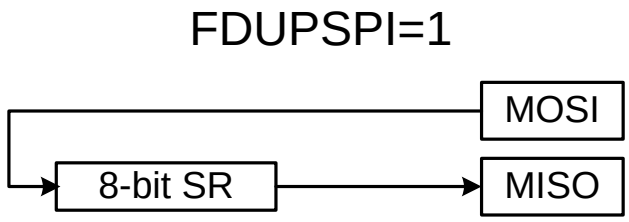


Figure 4. Full-duplex SPI interface.

The SPI Command Byte

The first byte clocked into the SPI interface is a command byte that sets: the register address; the direction; and when operating as a USB peripheral device, a bit that directly sets the ACKSTAT bit. In all SPI transactions, whether in or out, the bit ordering is b7 first, b0 last.

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

Figure 5. The SPI command byte.

An SPI cycle starts with the SPI master driving CS# low, then driving eight SPI clocks whose rising edges strobe in the command byte. REG[4:0] set the register address, and the direction bit sets the read or write direction for the SPI cycle. For USB peripheral operation, the ACKSTAT bit writes the corresponding bit in the EPSTALLS register.

Following the command byte, the SPI master issues one or more groups of 8-SCK clocks to clock byte data into or out of the MAX3421E. When accessing a FIFO, as long as CS# remains low, the register address clocked in with the command remains in effect. This ability to burst bytes is convenient when reading or writing the endpoint FIFOS. For example, to load 37 bytes into the EP0FIFO (peripheral mode), the SPI master writes the command byte 00000010 which selects R0 (EP0FIFO) for a write operation (direction bit is 1). Then it writes 37 bytes to the SPI port, and finally drives CS# high to complete the SPI cycle.

Note: Both MOSI and MISO data are sampled on the rising edge of SCK. Data changes on the falling edge of SCK.

The SPI cycle terminates when the SPI master return CS# to its high state.

SPI Modes

The SPI standard defines four clocking modes, reflecting two mode signals called CPOL (clock polarity) and CPHA (clock phase). These signals are represented in the form (CPOL, CPHA). An interface that expects both positive edge SCKS and the MOSI data to be available before the first positive clock edge, can operate in modes (0,0) and (1,1) without alteration. This property allows the MAX3421E to operate in mode (0,0) or (1,1) without requiring a mode pin.

The scope traces below illustrate identical data transfers between a microprocessor and the MAX3421E. **Figure 6** uses SPI mode (0,0) and **Figure 7** uses SPI mode (1,1). The difference is

the inactive level of the SCK signal, low for mode (0,0) and high for mode (1,1). In both modes the MOSI and MISO data sampled by the rising edge of SCK is the same.

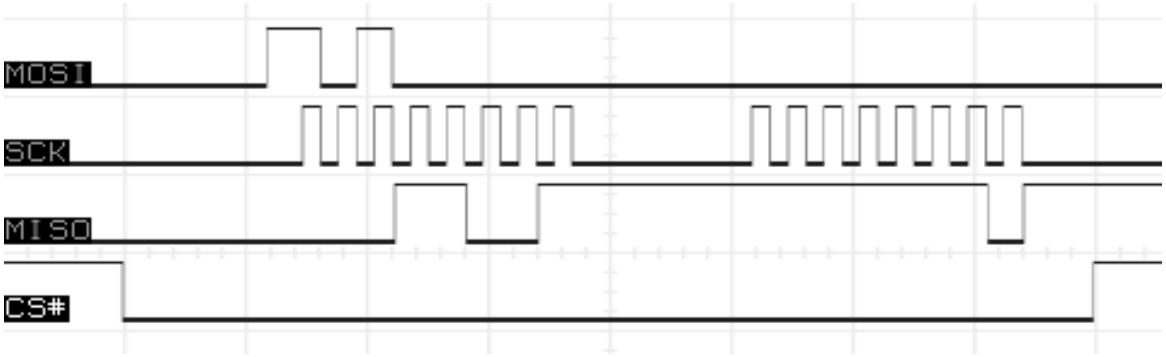


Figure 6. SPI interface operating in mode (0,0).

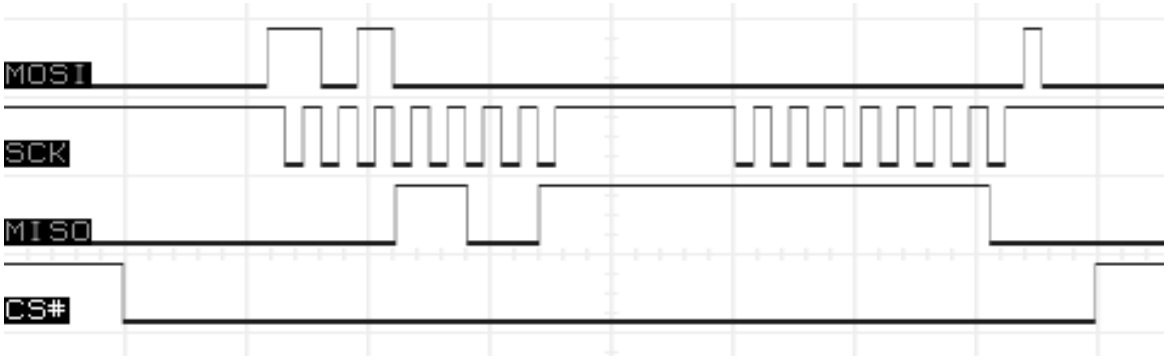


Figure 7. SPI interface operating in mode (1,1,).

FRAMEIRQ, FRAMEIE

Meaning: **FRAMEIRQ:** Frame Generator Interrupt Request
FRAMEIE: Frame Generator Interrupt Enable

Mode: **Host only**

The SIE sets the FRAMEIRQ bit whenever it generates a 1-millisecond frame marker, that consists of a full-speed SOF packet or a low-speed keep-alive pulse. The CPU clears the FRAMEIRQ bit by writing a 1 to it.

The CPU sets and clears the FRAMEIE bit. When FRAMEIE = 1, the FRAMEIRQ is enabled as a source to activate the INT pin.

Programming Notes

At full speed, the SIE sets the FRAMEIRQ interrupt at the beginning of the SOF packet.

FRMRST

Meaning: Reset the SOF frame counter

Mode: **Host only**

The CPU sets this bit to clear the frame counter.
The SIE clears this bit to indicate that it has completed the operation.

Programming Notes

This bit has meaning only when the MAX3421E is operating as a full-speed host. (When operating as a low-speed host, there is no SOF packet or frame count.) After setting FRMRST = 1, the next SOF packet will contain a frame count of zero.

The MAX3421E clocks its frame counter using an internal prescaler. Setting FRMRST = 1 resets the frame count but not the prescaler, thus ensuring a steady stream of SOF packets every millisecond.

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

Meaning:

GPIN	General-Purpose Input pins 0 through 7
GPINPOL	General-Purpose IN Interrupt Polarity 0 though 7
GPINIRQ	General-Purpose IN Interrupt Request 0 through 7
GPINIE	General-Purpose IN Interrupt Enable 0 through 7

Mode: **Peripheral and Host**

GPIN

The CPU reads the GPIN pin states by reading the GPIN register bits. Only the outside circuitry can control these register bit states. All eight GPIN pins are pulled up to V_{CC} with weak (~20 kΩ) internal resistors.

GPINPOL

The CPU sets the GPINPOL bits to set the edge polarity of the interrupt requests for each of the GPIN pins (**Table 5**). These bits are cleared at power-on. They retain their states either when CHIPRES = 1 or after a mode change.

Table 5. Edge Polarity for the GPIN Interrupts

GPINPOL	Polarity
0	neg edge
1	pos edge

GPINIRQ

The MAX3421E sets a GPIN Interrupt Request bit when a signal on the GPIN pin makes a positive or negative transition. The GPINPOL bit (Table 5) controls the active edge polarity. The GPINIRQ bits are active whether or not the individual GPIN interrupt enables or the IE bit is set.

GPINIE

The CPU sets a GPIN Interrupt Enable bit to pass the corresponding IRQ bit through to the interrupt logic feeding the MAX3421E INT pin. If IE = 1, an enabled IRQ appears either on the INT pin if SEPIRQ = 0, or on the GPX pin if SEPIRQ = 1 and GPX[B:A] = 10.

Note: During normal operation, the GPIN interrupts are OR'd with all the other MAX3421E interrupt sources indicated on the INT pin. It is also possible to detach the eight GPIN interrupt request bits as INT pin sources and make them separately available as a group on the GPX pin. In this mode, the GPX pin serves as a second MAX3421E INT output pin. This mode may be preferred in systems that need to minimize detection time for external events. The SEPIRQ bit enables this separation of interrupt sources.

GPOUT(0 through 7)

Meaning: General-Purpose Output pins 0 through 7

Mode: **Peripheral and Host**

The CPU sets and clears these bits.

Programming Notes

The CPU writes these bits to control the states of the GPOUT pins. The output voltages are referenced to the voltage on the VL pin. The CPU can also read these bits. Reading the bit indicates the state of the output flip-flop that drives the output buffer. Therefore, if the output pin is driving a large load (e.g., an LED) which compromises the logic level, the CPU will read the correct logic state of the output pin.

GPXB, GPXA

Meaning: Two bits, GPXB and GPXA, determine the output of the GPX pin.

Mode: **Peripheral and Host**

The CPU sets and clears these bits.

Programming Notes

The GPX pin can output one of four internal signals, as shown in the table below:

GPXB	GPXA	GPX Pin
0	0	OPERATE (complement of internal POR)
0	1	VBUS Detect
1	0	BUSACT or INIRQ*
1	1	SOF (0-1 transition when SOF packet arrives, 50% duty-cycle signal)

* *If SEPIRQ = 1*

The default setting for the GPX output when GPXB = 1 and GPXA = 0 is BUSACT. However, if the SEPIRQ bit is set to 1, the BUSACT signal is replaced by an interrupt request signal that is active whenever one of the eight GPIN pins makes a 0-1 or 1-0 transition. In this case the GPX pin serves as a second interrupt pin (along with INT), with the same configuration (level or edge, edge polarity) as the INT pin. See page 53 for more information about the SEPIRQ bit.

HOST

Meaning: MAX3421E host or peripheral operation

Mode: **Peripheral and Host**

The CPU sets HOST = 1 to operate the MAX3421E as a USB host.
The CPU sets HOST = 0 to operate the MAX3421E as a USB peripheral (default).

Programming Notes

At power-on or after a RES# pin reset, the MAX3421E defaults to peripheral operation with the HOST bit set to 0. In this mode the MAX3421E operates as a MAX3420E peripheral-only controller.

The CPU sets HOST = 1 to operate the MAX3421E as a host. It uses the register set shown in Table 1.

Programming information for operating when HOST = 0 is found in the *MAX3420E Programming Guide*. For reference, Table 2 shows all MAX3421E register bits, for both peripheral and host operation.

HRSL Register
HRSLT[3:0]
SNDTOGRD, RCVTOGRD

Meaning: These bits indicate the results of a host transfer:
- **HRSLT[3:0]** indicate the result code.
- **SNDTOGRD** and **RCVTOGRD** indicate the resulting data toggle values for OUT and IN transfers, respectively.

Mode: **Host only**

The SIE sets and clears these bits.

Programming Notes

The SIE updates these bits at completion of a host transfer. The CPU reads the HRSL register after receiving the HXFRDNIRQ or after polling the HRSL register and reading an HRSLT value other than hrBUSY. The HRSLT bits indicate the result of the host transfer, as shown in **Table 6.**

Table 6. HRSLT[3:0] Codes

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

The SNDTOGRD (for an OUT transfer) and RCVTOGRD (for an IN transfer) indicate the data toggle values resulting from the transfer. The CPU should read and store these values whenever it finishes a consecutive sequence of transfers to the same endpoint. Doing this allows the CPU to restore the toggle value the next time the CPU transfers data to the same endpoint. The CPU initializes the endpoint toggle value using the SNDTOG0/1 and RCVTOG0/1 bits in the HCTL register (page 59).

HUBPRE

Meaning: Send the PRE PID to a LS device operating through a USB hub.

Mode: **Host only**

The CPU sets and clears this bit.

Programming Notes

If the host firmware detects (during enumeration) that it is talking to a low-speed peripheral through a USB hub, it sets HUBPRE = 1. This instructs the SIE to precede every low-speed packet with a full-speed PRE PID, and to do the necessary signaling required by the USB specification.

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

Meaning: The CPU writes this register to launch a host transfer.

Mode: **Host only**

The CPU sets and clears these bits.

Programming Notes

This register is load-sensitive. This means that when the CPU writes it, the SIE launches a transfer. The CPU loads values shown in **Table 7** to launch the various USB transfer types.

Table 7. HXFR Register Bit Settings for Different Transfer Types

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

The BULK-IN and BULK-OUT entries apply for transfers either to a BULK or INTERRUPT endpoint. These two transfer types are identical, differing only by when the host firmware schedules them.

The '-ep' field in the 'hex' column indicates the value of EP[3:0].

For details about each of these transfer types, see "Programming Host Transfers" on page 10.

HXFRDNIRQ, HXFRDNIE

Meaning: **HXFRDNIRQ:** Host Transfer Done Interrupt Request
HXFRDNIE: Host Transfer Done Interrupt Enable

Mode: **Host only**

The SIE sets the HXFRDNIRQ bit when it has completed a host transfer. The CPU clears the HXFRDNIRQ by writing a 1 to it.

The CPU sets and clears the HXFRDNIE bit. When HXFRDNIE = 1, the HXFRDNIRQ is enabled as a source to activate the INT pin.

Programming Notes

When this interrupt asserts, the CPU determines the result of the host transfer by reading the HSRLT bits in the HRSL register (page 36).

IE

Meaning: Enable the INT pin.

Mode: **Peripheral and Host**

The CPU sets this bit to activate the INT output pin. The characteristics of the INT output pin are determined by the INTLEVEL, POSINT and PULSEWID[1:0] bits (page 41).

The CPU clears this bit to disable the INT output pin.

Programming Notes

When IE = 0, the state of the INT pin is inactive (open for level mode, high for negative edge, low for positive edge).

The internal IRQ bits operate independent of the state of the IE bit. The IE bit only controls activation of the INT output pin.

INTLEVEL
POSINT
PULSEWID1, PULSEWID0

Meaning: **INTLEVEL**: Sets the INT output pin to level-active (1) or edge-active (0).
POSINT: Edge polarity of the edge-active INT pin.
PULSEWID: These two bits set the IRQ inactive time in edge mode (see below).

Mode: **Peripheral and Host**

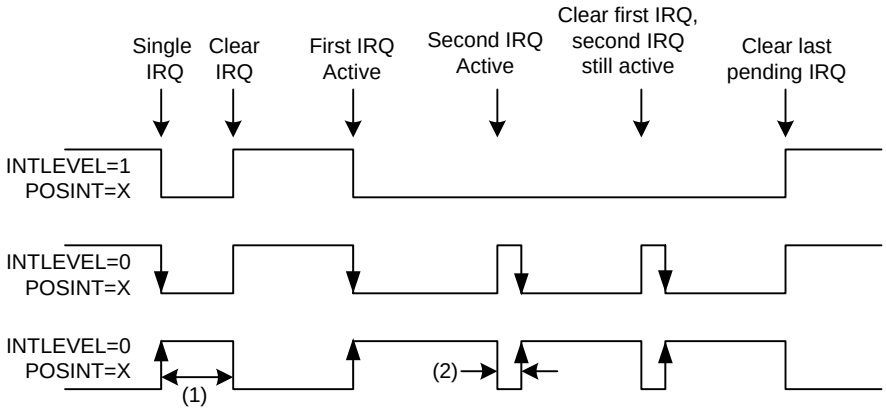
INTLEVEL
The CPU sets the INTLEVEL bit to make the INT output pin level-active. When INTLEVEL= 1 the INT pin drives low if one or more enabled interrupts are pending. In this mode the INT pin is open-drain, so the system must include a pullup resistor to VL.

The CPU clears the INTLEVEL bit to make the INT pin edge active. When INTLEVEL= 0, the edge polarity is set by the POSINT bit. In edge output mode the INT pin driver is push-pull, so no pullup resistor to VL is required.

POSINT
This bit has effect only when the CPU has programmed the INT pin to be edge-active (INTLEVEL= 0). When POSINT= 1 the INT pin signals pending interrupts with a positive edge, and when POSINT= 0, the INT pin signals pending interrupts with a negative edge.

PULSEWID1[0]
These bits have effect only when the CPU has programmed the INT pin to be edge-active (INTLEVEL = 0). They control the inactive INT pin time intervals between the CPU clearing one IRQ bit and the INT pin re-asserting due to one or more pending interrupts (**Figure 8**).

Programming Notes



(1) Width determined by clearing the IRQ. Width determined by PULSEWID[1:0].

Figure 8. INT pin behavior depending on INTLEVEL and POSINT bits.

The waveforms in ***Figure 8*** show the INT pin behavior for different settings of the INTLEVEL and POSINT bits.

In level mode (top diagram) the INT pin stays low until no interrupt requests are pending. The INTLEVEL mode is open-drain and requires a pullup resistor from the INT pin to VL.

In edge mode the INT pin delivers an edge whenever a new interrupt request occurs or an interrupt request bit is cleared while others are pending. If an interrupt is pending when another is cleared in edge mode, the INT pin momentarily goes inactive, then active again to deliver the edge. The inactive time, shown as (2) in the ***Figure 8***, is programmable to four values, shown in the table below.

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

Note: The MAX3420E does not have the PULSEWID[1:0] bits. The MAX3420E time is fixed at 10.6µS.

When SEPIRQ = 1, the MAX3421E removes the eight interrupts associated with the eight GPIN pins from the INT pin. The MAX3421E then routes them to the GPX pin, which serves as a second interrupt output pin when GPX[B:A] = 10. When the GPX pin operates in this manner its characteristics as an INT output pin are set by the INTLEVEL, POSINT, and PULSEWID[1:0] bits.

LOWSPEED

Meaning: Sets the host for low-speed operation

Mode: **Host only**

The CPU sets and clears this bit.

Programming Notes

The CPU sets this bit to enable operation as a low-speed USB host. The CPU will normally set this bit when it discovers that a peripheral has plugged in (activating CONDETIRQ) and that the quiescent bus state is D+ = 0, D- = 1.

This bus condition is indicated by:

- LOWSPEED = 1 and a J-state is detected.
- LOWSPEED = 0 and a K-state is detected.

See page 50 for information about detecting the bus state.

OSCOKIRQ, OSCOKIE

Meaning: **OSCOKIRQ:** Oscillator OK Interrupt Request
OSCOKIE: Oscillator OK Interrupt Enable

Mode: **Peripheral and Host**

An internal OSCOK signal indicates that the internal 12MHz oscillator and 48MHz PLL are stable and that the chip is ready to operate. The SIE sets the OSCOKIRQ bit when the OSCOK signal makes a 0-1 transition, indicating that the chip is ready to operate. The CPU clears the OSCOKIRQ bit by writing a 1 to it.

The CPU sets and clears the OSCOKIE bit. When OSCOKIE = 1 the OSCOKIRQ is enabled as a source to activate the INT pin.

Programming Notes

Whenever the CPU stops the MAX3421E oscillator by resetting the chip (setting CHIPRES = 1 then CHIPRES = 0), it should wait for the OSCOKIRQ to assert before continuing operation.

PERADDR Register

Meaning: Peripheral Address to which packets are to be sent.

Mode: **Host only**

The CPU writes this register; the SIE reads it.

Programming Notes

When the SIE sends a token packet after the CPU loads the HXFR register, it takes the peripheral address from this register. If the CPU talks only to one device address, the SIE can initialize this register once for all transfers to the device. The CPU normally issues a Set_Address request to the device during enumeration, and then loads the requested address into this register.

PWRDOWN

Meaning: Power Down the MAX3421E.

Mode: **Peripheral and Host**

The CPU sets the PWRDOWN bit to put the chip into a low-power state, and clears the PWRDOWN bit to resume operation.

Programming Notes

This bit is designed only for peripheral mode usage, although it is accessible in host mode. The CPU should never set POWERDOWN = 1 when operating as a host.

RCVBC Register

Meaning: Receive FIFO Byte Count Register
Mode: **Host only**

Programming Notes

After loading a data packet from the bus into the RCVFIFO, the SIE updates this register with the received byte count and asserts the INDAVIRQ bit.

After the CPU has read the number of bytes indicated in the RCVBC register, it clears the RCVDAVIRQ bit by writing a 1 to it. This readies the empty buffer for USB access.

RCVDAVIRQ, RCVDAVIE

Meaning: **RCVDAVIRQ:** Receive FIFO Data Available Interrupt Request
RCVDAVIE: Receive FIFO Data Available Interrupt Enable

Mode: **Host only**

The SIE sets the RCVDAVIRQ bit when new peripheral data is in the RCVFIFO as a result of a host IN request. After receiving this interrupt, the CPU clears the RCVDAVIRQ bit, reads the byte count in the RCVBC register, and does successive reads to the RCVFIFO register (R1) to retrieve the data. The SIE handles all retries (due to PID, CRC, data toggle, or timeout errors), and only interrupts when it generates the ACK handshake to the peripheral.

The CPU sets and clears the RCVDAVIE bit. When RCVDAVIE = 1, the RCVDAVIRQ is enabled as a source to activate the INT pin.

Programming Notes

1. The CPU **must** clear this IRQ bit (by writing a 1 to it) before reading the RCVFIFO data.
2. If any error occurs, the HXVRDNIRQ asserts, while the RCVDAVIRQ does not.

RCVFIFO Register

Meaning: Receive FIFO.

Mode: **Host only**

As a peripheral sends data over the bus in response to a host IN request, the SIE fills an internal FIFO with data. The CPU reads bytes from the FIFO by repeatedly reading the RCVFIFO register.

The CPU should never write the RCVFIFO, because it would corrupt the received data.

Programming Notes

After an error-free data packet arrives, the SIE loads the RCVBC register with the number of received bytes and asserts the RCVDAVIRQ bit (Receive Data Available IRQ). The CPU responds first by reading the RCVBC register to determine the number of bytes in the RCVFIFO, clearing the RCVDAVIRQ bit, and finally reading the bytes with repeated reads to the RCVFIFO register.

The RCVFIFO register connects to two internal 64-byte FIFOs. The two FIFOs allow the SIE to load IN data transmitted by a peripheral into one FIFO, while the CPU concurrently empties the other FIFO. If the CPU clears the RCVDAVIRQ bit when there is another packet waiting in the other FIFO, the SIE immediately re-asserts the RCVDAVIRQ bit.

The CPU should read RCVFIFO bytes only when USB received data is available, indicated by RCVDAVIRQ = 1.

REVISION Register

Meaning: MAX3421E Revision Number

Mode: **Peripheral and Host**

This read-only register indicates the chip revision code. Consult the Maxim website for current revision information. Writing this register has no effect.

RWUIRQ, RWUIE

Meaning: **RWUIRQ:** Remote Wakeup Interrupt Request
RWUIE: Remote Wakeup Interrupt Enable

Mode: **Host only**

The SIE sets the RWUIRQ bit when it receives a remote wakeup signal from a peripheral device. The CPU clears the RWUIRQ bit by writing a 1 to it.

The CPU sets and clears the RWUIE bit. When RWUIE = 1, the RWUIRQ is enabled as a source to activate the INT pin.

Programming Notes

After the CPU suspends bus signaling by setting SOFKAEN = 0, a peripheral device that is enabled for remote wakeup (RWU) can assert the RWU bus state to request the host to resume bus activity. When the SIE detects the RWU signal of 10 milliseconds of the bus K-state followed by an EOP, it asserts the RWUIRQ bit.

SAMPLEBUS
JSTATUS, KSTATUS

Meaning: Sample the state of the USB bus.

SAMPLEBUS:	Sample the bus.
JSTATUS, KSTATUS:	Indicate the state.

Mode: **Host only**

The CPU sets the SAMPLEBUS bit to instruct the SIE to sample the state of the D+ and D- lines**.** The SIE clears the SAMPLEBUS bit when it completes the operation.

The JSTATUS bits are read-only, set and cleared by the SIE.

Programming Notes

The JSTATUS and KSTATUS bits update under two conditions:

1. The CPU sets SAMPLEBUS = 1
2. The CONDETIRQ asserts.

The second case indicates either a device attach or detach condition. The CPU should respond to the CONDETIRQ by reading the JSTATUS and KSTATUS bits to determine which event occurred.

The JSTATUS and KSTATUS bits are encoded as shown in Table 8.

Table 8. Encoding of the JSTATUS and KSTATUS Bits

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

The fourth entry in Table 8 is a non-defined USB bus condition.

Note: The meaning of the J and K states depends on the setting of the LOWSPEED bit. When LOWSPEED = 0, then J means D+ high and D- low; when LOWSPEED = 1, J means D+ low and D- high.

SEPIRQ

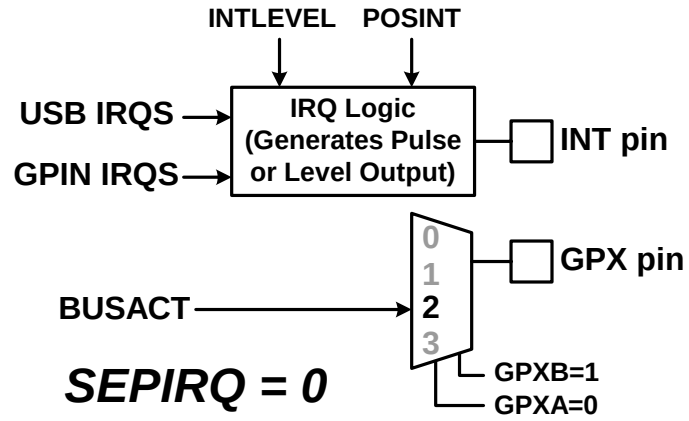
Meaning: Provides the GPIN IRQS on a separate pin (GPX).

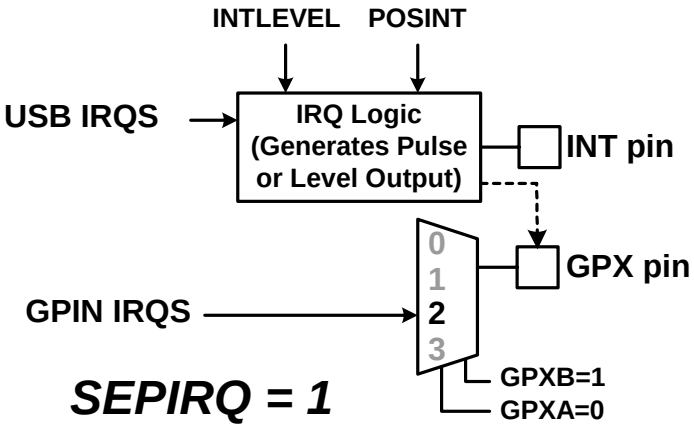
Mode: **Peripheral and Host**

The CPU sets and clears these bits.

Programming Notes

The MAX3421E INT pin goes active whenever any of the internal interrupts, USB or GPIN, asserts (assuming that the IRQ is enabled and IE = 1). This default case is shown in the top figure below, when SEPIRQ = 0.





When the CPU sets SEPIRQ = 1, the eight GPIN IRQ signals are removed as an interrupt source to the INT pin, and routed to the GPX pin to serve as a second MAX3421E interrupt pin (bottom figure). The GPX[B:A] setting of 2 normally connects the BUSACT signal to the GPX pin.

Setting SEPIRQ = 1 replaces this signal with a signal indicating any of the eight GPIN interrupts. The dotted line in the bottom figure indicates that when the GPX pin functions as this second interrupt pin, its characteristics (as determined by INTLEVEL and POSINT) are the same as for the INT pin.

SIGRSM

Meaning: Signal a bus resume event.

Mode: **Host only**

The CPU sets this bit to signal a bus resume event.
The SIE clears this bit to indicate that it has completed the resume signaling.

Programming Notes

A USB host suspends a peripheral, putting it into a low-power state, by stopping bus activity. The CPU does this by setting SOFKAENAB = 0, which stops the MAX23421E from automatically generating full-speed SOF packets or low-speed keep-alive pulses every millisecond.

To resume bus activity, the host sends a Resume signal consisting of a 20 millisecond K-state followed by a low-speed EOP. The CPU accomplishes this by setting SIGRSM = 1, and then checking for SIGRSM = 0. Then the CPU restarts the millisecond frame markers by setting SOFKAENAB = 1.

The CPU can also check for the end of MAX3421E resume signaling by using the BUSEVENT interrupt request bit BUSEVENTIRQ (page 19). The SIE sets this bit when the SIGRSM bit makes a 1-0 transition.

SNDBAVIRQ, SNDBAVIE

Meaning: Send Buffer Available Interrupt Request
Send Buffer Available Interrupt Enable

Mode: **Host only**

The SIE sets the SNDBAVIRQ bit after it sends the data in the SNDFIFO to a peripheral and two conditions result:

1. The peripheral answered with an ACK handshake.
2. No low-level errors (valid PID, CRC, EOP, Stuff) occurred.

The CPU clears the SNDBAVIRQ bit by writing the SNDBC register.

The CPU sets and clears the SNDBAVIE bit. When SNDBAVIE = 1 the SNDBAVIRQ is enabled as a source to activate the INT pin.

Programming Notes

The SIE clears its FIFO pointers at the termination of any host transfer. Therefore, if the host reads a nonzero HRSLT after an IN transfer (for example HRSLT[3:0] = 0x04 when the peripheral returns a NAK handshake), the CPU can retry the IN transfer simply by reloading the HXFR register with the appropriate value. Because the FIFO data persists until rewritten, and because the SIE FIFO pointer is reset, the host can repeatedly send the same SNDFIFO data in this manner without reloading the FIFO data every time.

The CPU clears the SNDBAVIRQ by writing the SNDBC register. **The CPU should never directly clear the SNDBAVIRQ bit.**

SNDBC Register

Meaning: Send FIFO Byte Count Register.

Mode: **Host only**

The CPU loads this register to indicate the number of bytes it loaded into the SNDFIFO, and to commit the FIFO to sending OUT data over the bus.

Programming Notes

When the CPU loads the SNDBC register, the SIE clears the SNDBAVIRQ bit. If the second FIFO is available, the SIE immediately re-asserts the SNDBAVIRQ bit.

Unlike the RCVDAVIRQ bit, which the CPU directly clears when finished unloading the FIFO, the CPU clears the SNDBAVIRQ by writing the SNDBC register. **The CPU should never directly clear the SNDBAVIRQ bit.**

SNDFIFO Register

Meaning: Send FIFO.

Mode: **Host only**

The CPU fills an internal FIFO with data for transmission as an OUT transfer by repeatedly writing the SNDFIFO register.

Programming Notes

If the CPU reads the SNDFIFO register after writing 64 bytes to the FIFO, the bytes are read back.

After loading the SNDFIFO, the CPU writes the SNDBC (Send Byte Count) register with the number of bytes loaded. When the CPU writes the byte count register, the SIE negates the SNDBAVIRQ (Send Buffer Available IRQ) and commits the FIFO to USB transmission.

The SNDFIFO register connects to two internal 64-byte FIFOs. The two FIFOs allow the CPU to load OUT data into one FIFO while the SIE concurrently sends data from the other FIFO over USB. If the CPU writes the SNDBC register and the other buffer is available, the SIE negates the SNDBAVIRQ bit and then immediately re-asserts it to indicate availability of the second buffer.

The CPU should load the SNDFIFO only when a SEND buffer is available, indicated by SNDBAVIRQ = 1.

SNDTOG1, SNDTOG0
RCVTOG1, RCVTOG0

Meaning: Set or clear the data toggle value for a data transfer.

Mode: **Host only**

The CPU writes a 1 to one of these bit pairs to initialize the data toggle value for a data transfer. Writing a 0 to these bits has no effect. Writing 1s to both bits of either bit pair has no effect.

Programming Notes

The MAX3421E contains two data toggle flip-flops, which it uses to implement USB signaling protocol during SNDFIFO and RCVFIFO data transfers. Before transferring data to an endpoint, the CPU initialized the data toggle value for the endpoint by using these bits. After transferring data to the endpoint, the CPU reads the toggle value in the SNDTOGRD or RCVTOGRD bit (page 36) and stores this value in CPU local storage. For multiple endpoints, the CPU maintains an array of toggle bit values, one per endpoint.

When the CPU returns to an endpoint to transfer more data, it must first initialize the data toggle value to the last value saved for the endpoint. The CPU uses the SNDTOG1/0 and RCVTOG1/0 bits to set the data toggle to the saved last value for the endpoint.

During consecutive transfers to the same endpoint, the SIE maintains the toggle values. The CPU needs to save and reinitialize toggle values only when switching endpoints.

See **About MAX3421E Data Toggles** (page 11) for more information about data toggles.

SOFKAENAB

Meaning: Enable automatic generation of full-speed SOF packets or low-speed Keep-Alive pulses.

Mode: **Host only**

The CPU sets and clears this bit.

Programming Notes

When the CPU sets SOFKAENAB = 1, the SIE automatically generates 1-millisecond frame markers. If the bit LOWSPEED = 0, the SIE generates SOF packets. If LOWSPEED = 1, the SIE generates keep-alive pulses.

The SOF or KA pulses start after the SOFKAENAB bit has asserted for 1 millisecond. If the CPU sets SOFKAENAB = 0 while the SIE is generating a frame marker, the SIE completes the signaling before shutting off the frame markers.

SOF Packets

Every millisecond the SIE sends the SOF PID, the contents of an internal 11-bit frame counter, and a CRC5 value. After sending the packet it increments the frame counter and asserts the FRAMEIRQ. The SIE does not update any HRSLT bits in the HRSL register, thus giving the CPU all the time needed to read the result of the last transfer.

The power-on default value of the frame counter is zero. The CPU can reset the frame counter at any time by setting the bit FRMRST = 1 (page 31).

Keep-Alive Pulses

When operating as a low-speed host and when SOFKAENAB = 1, the SIE generates a keep-alive pulse that consists of one low-speed EOP every millisecond

SUDFIFO Register

Meaning: Set Up Data FIFO Register.

Mode: **Host only**

The CPU writes the SUDFIFO register eight times to load an internal 8-byte FIFO with data to be included in a SETUP packet.

Programming Notes

The SUDFIFO has no associated byte-count register because the payload is always eight bytes.

If the CPU reads this register after writing eight bytes, the FIFO bytes are read back.

SUSDNIRQ, SUSDNIE

Meaning: **SUSDNIRQ:** Suspend operation Done IRQ
SUSDNIE: Suspend operation Done IE

Mode: **Host only**

The SIE asserts the SUSDNIRQ bit to indicate 3 milliseconds of bus activity. This usually occurs 3 milliseconds after the SPU sets SOFKAENAB = 0. The CPU clears the SISDNIRQ bit by writing a 1 to it.

The CPU sets and clears the SUSDNIE bit. When SUSDNIE = 1, the SUSDNIRQ is enabled as a source to activate the INT pin.

VBUSIRQ, VBUSIE NOVBUSIRQ, NOVBUSIE

Meaning:	**VBUSIRQ:**	V_{BUS} Detect Interrupt Request
	VBUSIE:	V_{BUS} Detect Interrupt Enable
	NOVBUSIRQ:	V_{BUS} Absence Interrupt Request
	NOVBUSIE:	V_{BUS} Absence Interrupt Enable

Mode: **Peripheral and Host**

The SIE sets the VBUSIRQ and NOVBUSIRQ bits when the output of the internal V_{BUS} comparator makes a 0-1 (VBUSIRQ) or 1-0 (NOVBUSIRQ) transition. This comparator indicates the voltage on the VBCOMP (V_{BUS} Comparator) pin. The CPU clears the VBUSIRQ and NOVBUSIRQ bits by writing a 1 to them.

The CPU sets and clears the VBUSIE and NOVBUSIE bits which, when set, enable the respective IRQ bits to activate the INT pin.

Programming Notes

These bits are active both in peripheral and host modes. In peripheral mode, they can be used in self-powered designs to detect attaching and detaching from a USB host. They are not normally required as V_{BUS} detect bits in host mode, because the host supplies (and controls) V_{BUS} power to the USB connector. In this case the VBCOMP pin can be used as a general-purpose input, with separate interrupts for each edge: 0-1 (VBUSIRQ) or 1-0 (NOVBUSIRQ). The VBCOMP pin has a weak (~100kΩ) pulldown resistor to ground, so no termination resistor is required when using the VBCOMP pin as a general-purpose input pin.

Some MAX3421E applications may implement both an A (host) and B (peripheral) USB connector, and automatically configure the MAX3421E to reflect which connector has an attached cable (the A-connector to a USB peripheral, or the B-connector to a USB host). In this case the A-connector V_{BUS} pin can be sourced by a local 5V supply, and the D+ and D- MAX3421E pulldown resistors can be switched on to detect a peripheral device plugging into the A connector. Host designs should use a current limiter/detector/switch such as the MAX4793 on the A connector V_{BUS} pin. The V_{BUS} pin on the B connector can connect to the MAX3421E VBCOMP pin. This lets the circuit sense the presence of V_{BUS} and thus detect when it is plugged into a USB host. In this manner the system can self-configure to either peripheral or host.

Because most of the Interrupt Enable bits are cleared during a USB bus reset, the initialization routine that turns on the interrupt enable bits should be called as part of servicing a USB bus reset.