Chipsmall
深圳市津利帝科技有限公司
Geelead Technology Limited
Chipsmall
采购部

CY7C68001

EZ-USB SX2™ High Speed USB Interface Device

1. Features

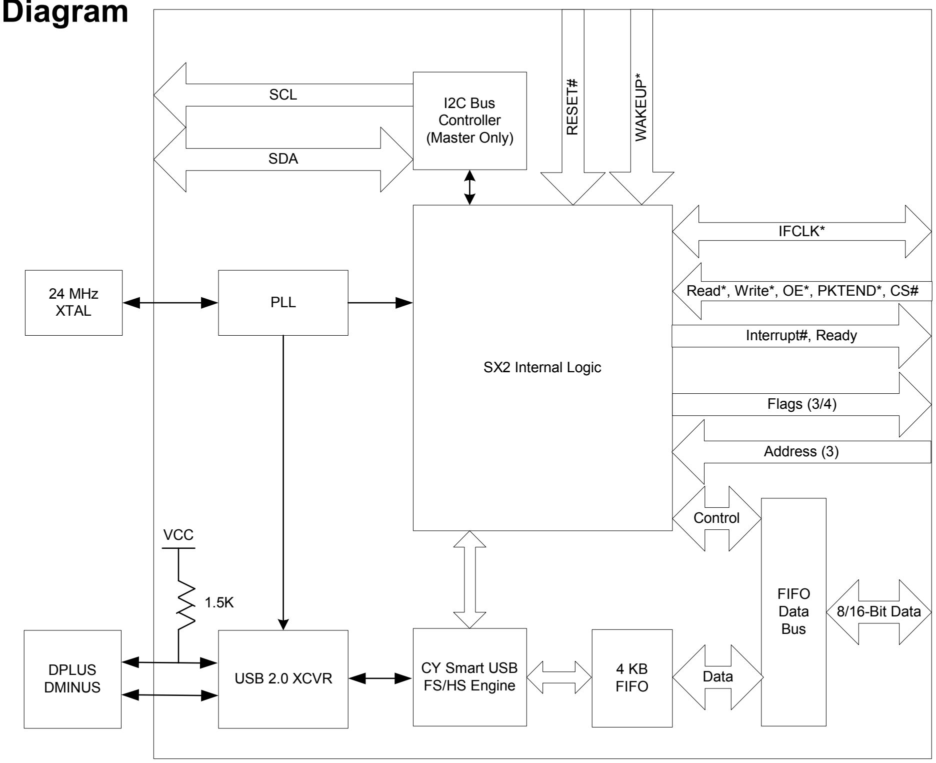
- USB 2.0-Certified Compliant
  - ❐ On the USB-IF Integrators List: Test ID Number 40000713
- Operates at High (480 Mbps) or Full (12 Mbps) Speed
- Supports Control Endpoint 0:
  - ❐ Used for handling USB device requests
- Supports Four Configurable Endpoints that share a 4-KB FIFO Space
  - ❐ Endpoints 2, 4, 6, 8 for application-specific control and data
- Standard 8- or 16-bit External Master Interface
  - ❐ Glueless interface to most standard microprocessors DSPs, ASICs, and FPGAs
  - ❐ Synchronous or Asynchronous interface
- Integrated Phase-locked Loop (PLL)
- 3.3V Operation, 5V Tolerant I/Os
- 56-pin SSOP and QFN Package
- Complies with most Device Class Specifications

2. Applications

- DSL modems
- ATA interface
- Memory card readers
- Legacy conversion devices
- Cameras
- Scanners
- Home PNA
- Wireless LAN
- MP3 players
- Networking
- Printers

The "Reference Designs" section of the Cypress web site, www.cypress.com, provides additional tools for typical USB applications. Each reference design comes complete with firmware source code and object code, schematics, and documentation.

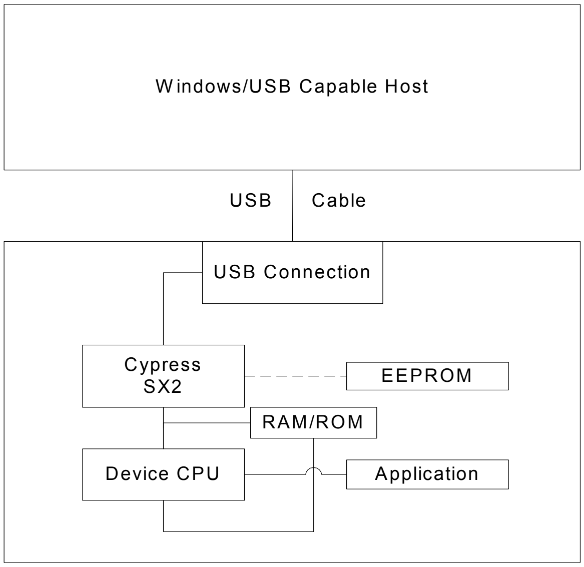
3. Logic Block Diagram



4. Introduction

The EZ-USB *SX2*™ USB interface device is designed to work with any external master, such as standard microprocessors, DSPs, ASICs, and FPGAs to enable USB 2.0 support for any peripheral design. SX2 has a built in USB transceiver and Serial Interface Engine (SIE), along with a command decoder for sending and receiving USB data. The controller has four endpoints that share a 4 KB FIFO space for maximum flexibility and throughput, and Control Endpoint 0. SX2 has three address pins and a selectable 8- or 16- bit data bus for command and data input or output.

Figure 4-1. Example USB System Diagram



5. Functional Overview

5.1 USB Signaling Speed

SX2 operates at two of the three rates defined in the *Universal Serial Bus Specification Revision 2.0*, dated April 27, 2000:

- Full speed, with a signaling bit rate of 12 Mbits/s
- High speed, with a signaling bit rate of 480 Mbits/s.

SX2 does not support the low speed signaling rate of 1.5 Mbits/s.

5.2 Buses

SX2 features:

- A selectable 8- or 16-bit bidirectional data bus
- An address bus for selecting the FIFO or Command Interface.

5.3 Boot Methods

During the power up sequence, internal logic of the *SX2* checks for the presence of an I^{2}C EEPROM.[1,2] If it finds an EEPROM, it boots off the EEPROM. When the presence of an EEPROM is detected, the *SX2* checks the value of first byte. If the first byte is found to be a 0xC4, the *SX2* loads the next two bytes into the IFCONFIG and POLAR registers, respectively. If the fourth byte is also 0xC4, the *SX2* enumerates using the descriptor in the EEPROM, then signals to the external master when enumeration is complete through an ENUMOK interrupt (See "Interrupt System" on page 3.). If no EEPROM is detected, the *SX2* relies on the external master for the descriptors. After this descriptor information is received from the external master, the *SX2* connects to the USB and enumerates.

5.3.1 EEPROM Organization

The valid sequence of bytes in the EEPROM are displayed in the following table.

Table 5-1. Descriptor Length Set to 0x06: Default Enumeration

Byte Index	Description
0	0xC4
1	IFCONFIG
2	POLAR
3	0xC4
4	Descriptor Length (LSB):0x06
5	Descriptor Length (MSB): 0x00
6	VID (LSB)
7	VID (MSB)
8	PID (LSB)
9	PID (MSB)
10	DID (LSB)
11	DID (MSB)

Table 5-2. Descriptor Length Not Set to 0x06

Byte Index	Description
0	0xC4
1	IFCONFIG
2	POLAR
3	0xC4
4	Descriptor Length (LSB)
5	Descriptor Length (MSB
6	Descriptor[0]
7	Descriptor[1]
8	Descriptor[2]

Notes

[1] Because there is no direct way to detect which EEPROM type (single or double address) is connected, *SX2* uses the EEPROM address pins A2, A1, and A0 to determine whether to send out one or two bytes of address. Single-byte address EEPROMs (24LC01, etc.) should be strapped to address 000 and double-byte EEPROMs (24LC64, etc.) should be strapped to address 001.

[2] The SCL and SDA pins must be pulled up for this detection method to work properly, even if an EEPROM is not connected. Typical pull up values are 2.2K–10K Ohms.

- **IFCONFIG**: The IFCONFIG byte contains the settings for the IFCONFIG register. The IFCONFIG register bits are defined in IFCONFIG Register 0x01 on page 17. If the external master requires an interface configuration different from the default, that interface can be specified by this byte.
- **POLAR**: The Polar byte contains the polarity of the FIFO flag pin signals. The POLAR register bits are defined in POLAR Register 0x04 on page 18. If the external master requires signal polarity different from the default, the polarity can be specified by this byte.
- **Descriptor**: The Descriptor byte determines if the *SX2* loads the descriptor from the EEPROM. If this byte = 0xC4, the *SX2* loads the descriptor starting with the next byte. If this byte does not equal 0xC4, the *SX2* waits for descriptor information from the external master.
- **Descriptor Length**: The Descriptor length is within the next two bytes and indicate the length of the descriptor contained within the EEPROM. The length is loaded least significant byte (LSB) first, then most significant byte (MSB).
- **Byte Index 6 Starts Descriptor Information**: The descriptor can be a maximum of 500 bytes.

5.3.2 Default Enumeration

An optional default descriptor can be used to simplify enumeration. Only the Vendor ID (VID), Product ID (PID), and Device ID (DID) need to be loaded by the *SX2* for it to enumerate with this default setup. This information is either loaded from an EEPROM in the case when the presence of an EEPROM (Table 5-1) is detected, or the external master may simply load a VID, PID, and DID when no EEPROM is present. In this default enumeration, the *SX2* uses the in-built default descriptor (refer to Default Descriptor on page 37).

If the descriptor length loaded from the EEPROM is 6, *SX2* loads a VID, PID, and DID from the EEPROM and enumerate. The VID, PID, and DID are loaded LSB, then MSB. For example, if the VID, PID, and DID are 0x0547, 0x1002, and 0x0001, respectively, then the bytes should be stored as:

- 0x47, 0x05, 0x02, 0x10, 0x01, 0x00.

If there is no EEPROM, *SX2* waits for the external master to provide the descriptor information. To use the default descriptor, the external master must write to the appropriate register (0x30) with descriptor length equal to 6 followed by the VID, PID, and DID. Refer to Default Enumeration on page 8 for further information on how the external master may load the values.

The default descriptor enumerates the following endpoints:

- Endpoint 2: Bulk out, 512 bytes in high speed mode, 64 bytes in full speed mode
- Endpoint 4: Bulk out, 512 bytes in high speed mode, 64 bytes in full speed mode
- Endpoint 6: Bulk in, 512 bytes in high speed mode, 64 bytes in full speed mode
- Endpoint 8: Bulk in, 512 bytes in high speed mode, 64 bytes in full speed mode.

The entire default descriptor is listed in Default Descriptor on page 37 of this data sheet.

5.4 Interrupt System

5.4.1 Architecture

The *SX2* provides an output signal that indicates to the external master that the *SX2* has an interrupt condition, or that the data from a register read request is available. The *SX2* has six interrupt sources: SETUP, EP0BUF, FLAGS, ENUMOK, BUSACTIVITY, and READY. Each interrupt can be enabled or disabled by setting or clearing the corresponding bit in the INTENABLE register.

When an interrupt occurs, the INT# pin is asserted, and the corresponding bit is set in the Interrupt Status Byte. The external master reads the Interrupt Status Byte by strobing SLRD/SLOE. This presents the Interrupt Status Byte on the lower portion of the data bus (FD[7:0]). Reading the Interrupt Status Byte automatically clears the interrupt. Only one interrupt request occurs at a time; the *SX2* buffers multiple pending interrupts.

If the external master has initiated a register read request, the *SX2* buffers interrupts until the external master has read the data. This insures that after a read sequence has begun, the next interrupt that is received from the *SX2* indicates that the corresponding data is available. Following is a description of this INTENABLE register.

5.4.2 INTENABLE Register Bit Definition

Bit 7: SETUP

If this interrupt is enabled, and the *SX2* receives a setup packet from the USB host, the *SX2* asserts the INT# pin and sets bit 7 in the Interrupt Status Byte. This interrupt only occurs if the setup request is not one that the *SX2* automatically handles. For complete details on how to handle the SETUP interrupt, refer to Endpoint 0 on page 8 of this data sheet.

Bit 6: EP0BUF

If this interrupt is enabled, and the Endpoint 0 buffer becomes available to the external master for read or write operations, the *SX2* asserts the INT# pin and sets bit 6 in the Interrupt Status Byte. This interrupt is used for handling the data phase of a setup request. For complete details on how to handle the EP0BUF interrupt, refer to Endpoint 0 on page 8 of this data sheet.

Bit 5: FLAGS

If this interrupt is enabled, and any OUT endpoint FIFO's state changes from empty to not empty and from not empty to empty, the *SX2* asserts the INT# pin and sets bit 5 in the Interrupt Status Byte. This is an alternate way to monitor the status of OUT endpoint FIFOs instead of using the FLAGA-FLAGD pins, and can be used to indicate when an OUT packet has been received from the host.

Bit 2: ENUMOK

If this interrupt is enabled and the *SX2* receives a SET_CONFIGURATION request from the USB host, the *SX2* asserts the INT# pin and sets bit 2 in the Interrupt Status Byte. This event signals the completion of the *SX2* enumeration process.

Bit 1: BUSACTIVITY

If this interrupt is enabled, and the *SX2* detects either an absence or resumption of activity on the USB bus, the *SX2* asserts the INT# pin and sets bit 1 in the Interrupt Status Byte. This usually indicates that the USB host is either suspending or resuming or that a self-powered device has been plugged in or unplugged. If the *SX2* is bus-powered, the external master must put the *SX2* into a low power mode after detecting a USB suspend condition to be USB-compliant.

Bit 0: READY

If this interrupt is enabled, bit 0 in the Interrupt Status Byte is set when the *SX2* has powered up and performed a self-test. The external master should always wait for this interrupt before trying to read or write to the *SX2*, unless an external EEPROM with a valid descriptor is present. If an external EEPROM with a valid descriptor is present, the ENUMOK interrupt occurs instead of the READY interrupt after power up. A READY interrupt also occurs if the *SX2* is awakened from a low power mode via the WAKEUP pin. This READY interrupt indicates that the *SX2* is ready for commands or data.

5.4.3 Qualify with READY Pin on Register Reads

It is true that all interrupts are buffered after a command read request has been initiated. However, in very rare conditions, there might be a situation when there is a pending interrupt already, when a read request is initiated by the external master. In this case it is the interrupt status byte that is output when the external master asserts the SLRD. So, a condition exists where the Interrupt Status Data Byte can be mistaken for the result of a command register read request. In order to get around this possible race condition, the first thing that the external master must do on getting an interrupt from the *SX2* is check the status of the READY pin. If the READY is low at the time the INT# was asserted, the data that is output when the external master strobes the SLRD is the interrupt status byte (not the actual data requested). If the READY pin is high at the time when the interrupt is asserted, the data output on strobing the SLRD is the actual data byte requested by the external master. So it is important that the state of the READY pin be checked at the time the INT# is asserted to ascertain the cause of the interrupt.

5.5 Resets and Wakeup

5.5.1 Reset

An input pin (RESET#) resets the chip. The internal PLL stabilizes after V_{CC} has reached 3.3V. Typically, an external RC network (R = 100 KOhms, C = 0.1 μF) is used to provide the RESET# signal. The Clock must be in a stable state for at least 200 μs before the RESET is released.

5.5.2 USB Reset

When the *SX2* detects a USB Reset condition on the USB bus, *SX2* handles it like any other enumeration sequence. This means that *SX2* enumerates again and assert the ENUMOK interrupt to let the external master know that it has enumerated. The external master is then responsible for configuring the *SX2* for the application. The external master should also check whether *SX2* enumerated at High or Full speed in order to adjust the EPxPKTLENH/L register values accordingly. The last initialization task is for the external master to flush all of the *SX2* FIFOs.

5.5.3 Wakeup

The *SX2* exits its low power state when one of the following events occur:

- USB bus signals a resume. The *SX2* asserts a BUSACTIVITY interrupt.
- The external master asserts the WAKEUP pin. The *SX2* asserts a READY interrupt[3].

5.6 Endpoint RAM

5.6.1 Size

- Control endpoint: 64 Bytes: 1 × 64 bytes (Endpoint 0).
- FIFO Endpoints: 4096 Bytes: 8 × 512 bytes (Endpoint 2, 4, 6, 8).

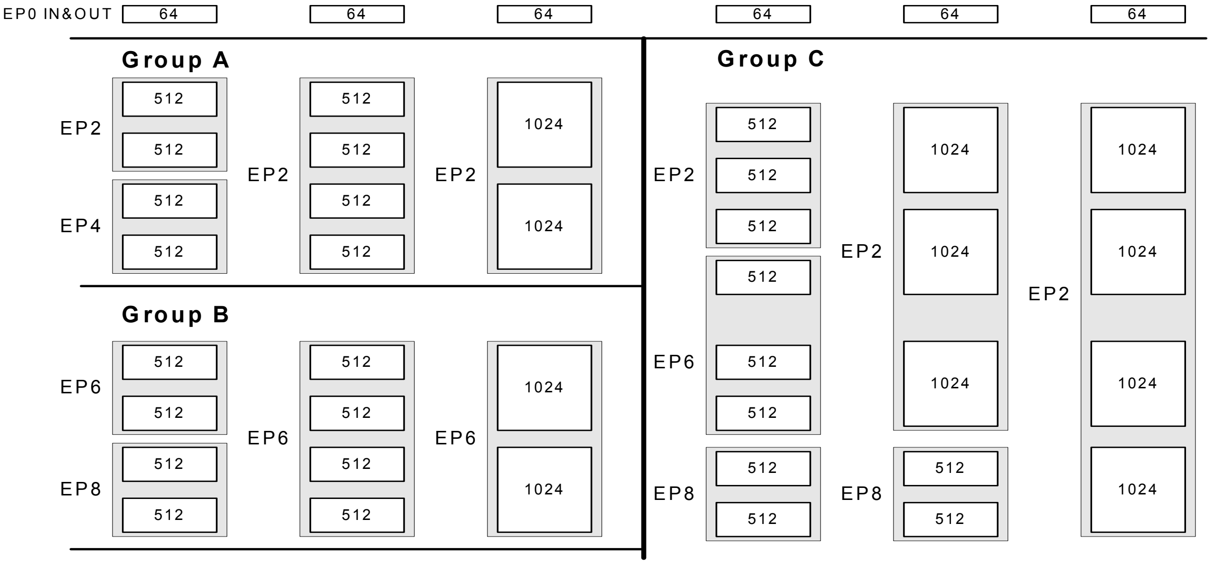
Organization

- EP0–Bidirectional Endpoint 0, 64-byte buffer.
- EP2, 4, 6, 8–Eight 512-byte buffers, bulk, interrupt, or isochronous. EP2 and EP6 can be either double-, triple-, or quad-buffered. EP4 and EP8 can only be double-buffered. For high speed endpoint configuration options, see Figure 8-1. on page 11.

Note

[3] If the descriptor loaded is set for remote wakeup enabled and the host does a set feature remote wakeup enabled, then the SX2 logic performs RESUME signalling after a WAKEUP interrupt.

Figure 5-1. Endpoint Configurations (High Speed Mode)



Endpoint 0 is the same for every configuration as it serves as the CONTROL endpoint. For Endpoints 2, 4, 6, and 8, refer to Figure 8-1. on page 11. Endpoints 2, 4, 6, and 8 may be configured by choosing either:

- One configuration from Group A and one from Group B
- One configuration from Group C.

Some example endpoint configurations are as follows.

- EP2: 1024 bytes double-buffered, EP6: 512 bytes quad-buffered.
- EP2: 512 bytes double-buffered, EP4: 512 bytes double-buffered, EP6: 512 bytes double-buffered, EP8: 512 bytes double buffered.
- EP2: 1024 bytes quad-buffered.

5.6.2 Default Endpoint Memory Configuration

At power-on-reset, the endpoint memories are configured as follows:

- EP2: Bulk OUT, 512 bytes/packet, 2x buffered.
- EP4: Bulk OUT, 512 bytes/packet, 2x buffered.
- EP6: Bulk IN, 512 bytes/packet, 2x buffered.
- EP8: Bulk IN, 512 bytes/packet, 2x buffered.

5.7 External Interface

The *SX2* presents two interfaces to the external master.

1. A FIFO interface through which EP2, 4, 6, and 8 data flows.
2. A command interface, which is used to set up the *SX2*, read status, load descriptors, and access Endpoint 0.

5.7.1 Architecture

The *SX2* slave FIFO architecture has eight 512-byte blocks in the endpoint RAM that directly serve as FIFO memories and are controlled by FIFO control signals (IFCLK, CS#, SLRD, SLWR, SLOE, PKTEND, and FIFOADR[2:0]).

The *SX2* command interface is used to set up the *SX2*, read status, load descriptors, and access Endpoint 0. The command interface has its own READY signal for gating writes, and an INT# signal to indicate that the *SX2* has data to be read, or that an interrupt event has occurred. The command interface uses the same control signals (IFCLK, CS#, SLRD, SLWR, SLOE, and FIFOADR[2:0]) as the FIFO interface, except for PKTEND.

5.7.2 Control Signals

FIFOADDR Lines

The *SX2* has three address pins that are used to select either the FIFOs or the command interface. The addresses correspond to the following table.

Table 5-3. FIFO Address Lines Setting

Address/Selection	FIFOADR2	FIFOADR1	FIFOADR0
FIFO2	0	0	0
FIFO4	0	0	1
FIFO6	0	1	0
FIFO8	0	1	1
COMMAND	1	0	0
RESERVED	1	0	1
RESERVED	1	1	0
RESERVED	1	1	1

The *SX2* accepts either an internally derived clock (30 MHz or 48 MHz) or externally supplied clock (IFCLK, 5 to 50 MHz), and SLRD, SLWR, SLOE, PKTEND, CS#, FIFOADR[2:0] signals from an external master. The interface can be selected for 8- or 16- bit operation by an internal configuration bit, and an Output Enable signal SLOE enables the data bus driver of the selected width. The external master must ensure that the output enable signal is inactive when writing data to the *SX2*. The interface can operate either asynchronously where the SLRD and SLWR signals act directly as strobes, or synchronously where the SLRD and SLWR act as clock qualifiers. The optional CS# signal tristates the data bus and ignore SLRD, SLWR, PKTEND.

The external master reads from OUT endpoints and writes to IN endpoints, and reads from or writes to the command interface.

Read: SLOE and SLRD

In synchronous mode, the FIFO pointer is incremented on each rising edge of IFCLK while SLRD is asserted. In asynchronous mode, the FIFO pointer is incremented on each asserted-to-deasserted transition of SLRD.

SLOE is a data bus driver enable. When SLOE is asserted, the data bus is driven by the *SX2*.

Write: SLWR

In synchronous mode, data on the FD bus is written to the FIFO (and the FIFO pointer is incremented) on each rising edge of IFCLK while SLWR is asserted. In asynchronous mode, data on the FD bus is written to the FIFO (and the FIFO pointer is incremented) on each asserted-to-deasserted transition of SLWR.

PKTEND

PKTEND commits the current buffer to USB. To send a short IN packet (one which has not been filled to max packet size determined by the value of PL[X:0] in EPxPKTLENH/L), the external master strobes the PKTEND pin.

All these interface signals have a default polarity of low. In order to change the polarity of PKTEND pin, the master may write to the POLAR register anytime. In order to switch the polarity of the SLWR/SLRD/SLOE, the master must set the appropriate bits 2, 3 and 4 respectively in the FIFOPINPOLAR register located at XDATA space 0xE609. Please note that the *SX2* powers up with the polarities set to low. POLAR Register 0x04 on page 18 provides further information on how to access this register located at XDATA space.

5.7.3 IFCLK

The IFCLK pin can be configured to be either an input (default) or an output interface clock. Bits IFCONFIG[7:4] define the behavior of the interface clock. To use the *SX2*'s internally-derived 30- or 48 MHz clock, set IFCONFIG.7 to 1 and set IFCONFIG.6 to 0 (30 MHz) or to 1 (48 MHz). To use an externally supplied clock, set IFCONFIG.7=0 and drive the IFCLK pin (5 MHz to 50 MHz). The input or output IFCLK signal can be inverted by setting IFCONFIG.4=1.

5.7.4 FIFO Access

An external master can access the slave FIFOs either asynchronously or synchronously:

- Asynchronous–SLRD, SLWR, and PKTEND pins are strobes.
- Synchronous–SLRD, SLWR, and PKTEND pins are enables for the IFCLK clock pin.

An external master accesses the FIFOs through the data bus, FD [15:0]. This bus can be either 8- or 16-bits wide; the width is selected via the WORDWIDE bit in the EPxPKTLENH/L registers. The data bus is bidirectional, with its output drivers controlled by the SLOE pin. The FIFOADR[2:0] pins select which of the four FIFOs is connected to the FD [15:0] bus, or if the command interface is selected.

5.7.5 FIFO Flag Pins Configuration

The FIFO flags are FLAGA, FLAGB, FLAGC, and FLAGD. These FLAGx pins report the status of the FIFO selected by the FIFOADR[2:0] pins. At reset, these pins are configured to report the status of the following:

- FLAGA reports the status of the programmable flag.
- FLAGB reports the status of the full flag.
- FLAGC reports the status of the empty flag.
- FLAGD defaults to the CS# function.

The FIFO flags can either be indexed or fixed. Fixed flags report the status of a particular FIFO regardless of the value on the FIFOADR [2:0] pins. Indexed flags report the status of the FIFO selected by the FIFOADR [2:0]pins.[4]

5.7.6 Default FIFO Programmable Flag Setup

By default, FLAGA is the Programmable Flag (PF) for the endpoint being pointed to by the FIFOADR[2:0] pins. For EP2 and EP4, the default endpoint configuration is BULK, OUT, 512, 2x; the PF pin asserts when the entire FIFO has ≥ 512 bytes. For EP6 and EP8, the default endpoint configuration is BULK, IN, 512, 2x, and the PF pin asserts when the entire FIFO has less than/equal to 512 bytes. In other words, EP6/8 report a half-empty state, and EP2/4 report a half-full state.

5.7.7 FIFO Programmable Flag (PF) Setup

Each FIFO's programmable-level flag (PF) asserts when the FIFO reaches a user-defined fullness threshold. That threshold is configured as follows:

1. For OUT packets: The threshold is stored in PFC12:0. The PF is asserted when the number of bytes *in the entire FIFO* is less than/equal to (DECIS = 0) or greater than/equal to (DECIS = 1) the threshold.
2. For IN packets, with PKTSTAT = 1: The threshold is stored in PFC9:0. The PF is asserted when the number of bytes written into *the current packet in the FIFO* is less than/equal to (DECIS = 0) or greater than/equal to (DECIS = 1) the threshold.
3. For IN packets, with PKTSTAT = 0: The threshold is stored in two parts: PKTS2:0 holds the number of committed packets, and PFC9:0 holds the number of bytes in the current packet. The PF is asserted when the FIFO is at or less full than (DECIS = 0), or at or more full than (DECIS = 1), the threshold.

Note

[4] In indexed mode, the value of the FLAGx pins is indeterminate except when addressing a FIFO (FIFOADR[2:0]={000,001,010,011}).

5.7.8 Command Protocol

An address of [1 0 0] on FIFOADR [2:0] selects the command interface. The command interface is used to write to and read from the *SX2* registers and the Endpoint 0 buffer, as well as the descriptor RAM. Command read and write transactions occur over FD[7:0] only. Each byte written to the *SX2* is either an address or a data byte, as determined by bit7. If bit7 = 1, then the byte is considered an address byte. If bit7 = 0, then the byte is considered a data byte. If bit7 = 1, then bit6 determines whether the address byte is a read request or a write request. If bit6 = 1, then the byte is considered a read request. If bit6 = 0 then the byte is considered a write request. Bits [5:0] hold the register address of the request. The format of the command address byte is shown in Table 5-4.

Table 5-4. Command Address Byte

Address/ Data#	Read/ Write#	A5	A4	A3	A2	A1	A0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Each Write request is followed by two or more data bytes. If another address byte is received before both data bytes are received, the *SX2* ignores the first address and any incomplete data transfers. The format for the data bytes is shown in Table 5-5 and Table 5-6. Some registers take a series of bytes. Each byte is transferred using the same protocol.

Table 5-5. Command Data Byte One

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	X	X	X	D7	D6	D5	D4

Table 5-6. Command Data Byte Two

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	X	X	X	D3	D2	D1	D0

The first command data byte contains the upper nibble of data, and the second command byte contains the lower nibble of data.

Write Request Example

Prior to writing to a register, two conditions must be met: FIFOADR[2:0] must hold [1 0 0], and the Ready line must be HIGH. The external master should not initiate a command if the READY pin is not in a HIgh state.

Example: to write the byte <10110000> into the IFCONFIG register (0x01), first send a command address byte as follows.

Table 5-7. Command Address Write Byte

Ad-dress/Data#	Read/ Write#	A5	A4	A3	A2	A1	A0
1	0	0	0	0	0	0	1

- The first bit signifies an address transfer.
- The second bit signifies that this is a write command.
- The next six bits represent the register address (000001 binary = 0x01 hex).

After the byte has been received the *SX2* pulls the READY pin low to inform the external master not to send any more information. When the *SX2* is ready to receive the next byte, the *SX2* pulls the READY pin high again. This next byte, the upper nibble of the data byte, is written to the *SX2* as follows.

Table 5-8. Command Data Write Byte One

Ad-dress/Data#	Don't Care	Don't Care	Don't Care	D7	D6	D5	D4
0	X	X	X	1	0	1	1

- The first bit signifies that this is a data transfer.
- The next three are don't care bits.
- The next four bits hold the upper nibble of the transferred byte.

After the byte has been received the *SX2* pulls the READY pin low to inform the external master not to send any more information. When the *SX2* is ready to receive the next byte, the *SX2* pulls the READY pin high again. This next byte, the lower nibble of the data byte is written to the *SX2*.

Table 5-9. Command Data Write Byte Two

Address/ Data#	Don't Care	Don't Care	Don't Care	D3	D2	D1	D0
0	X	X	X	0	0	0	0

At this point the entire byte <10110000> is transferred to register 0x01 and the write sequence is complete.

Read Request Example

The Read cycle is simpler than the write cycle. The Read cycle consists of a read request from the external master to the *SX2*. For example, to read the contents of register 0x01, a command address byte is written to the *SX2* as follows.

Table 5-10. Command Address Read Byte

Ad-dress/Data#	Read/ Write#	A5	A4	A3	A2	A1	A0
1	1	0	0	0	0	0	1

When the data is ready to be read, the *SX2* asserts the INT# pin to tell the external master that the data it requested is waiting on FD[7:0].[5]

Note

[5] An important note: Once the *SX2* receives a Read request, the *SX2* allocates the interrupt line solely for the read request. If one of the six interrupt sources described in Interrupt System on page 3 is asserted, the *SX2* buffers that interrupt until the read request completes.

6. Enumeration

The *SX2* has two modes of enumeration. The first mode is automatic through EEPROM boot load, as described in Boot Methods on page 2. The second method is a manual load of the descriptor or VID, PID, and DID as described in the following section.

6.1 Standard Enumeration

The *SX2* has 500 bytes of descriptor RAM into which the external master may write its descriptor. The descriptor RAM is accessed through register 0x30. To load a descriptor, the external master does the following:

- Initiate a Write Request to register 0x30.
- Write two bytes (four command data transfers) that define the length of the entire descriptor about to be transferred. The LSB is written first, followed by the MSB.[6]
- Write the descriptor, one byte at a time until complete.[6] Note: the register address is only written once.

After the entire descriptor has been transferred, the *SX2* floats the pull up resistor connected to D+, and parse through the descriptor to locate the individual descriptors. After the *SX2* has parsed the entire descriptor, the *SX2* connects the pull up resistor and enumerate automatically. When enumeration is complete, the *SX2* notifies the external master with an ENUMOK interrupt.

The format and order of the descriptor should be as follows (see Default Descriptor on page 37 for an example):

- Device.
- Device qualifier.
- High speed configuration, high speed interface, high speed endpoints.
- Full speed configuration, full speed interface, full speed endpoints.
- String.

The SX2 can be set to run in full speed only mode. To force full speed only enumeration write a 0x02 to the unindexed register CT1 at address 0xE6FB before downloading the descriptors. This disables the chirp mechanism forcing the SX2 to come up in full speed only mode after the descriptors are loaded. The CT1 register can be accessed using the unindexed register mechanism. Examples of writing to unindexed registers are shown in Resetting Data Toggle on page 9. Each write consists of a command write with the target register followed by the write of the upper nibble of the value followed by the write of the lower nibble of the value.

6.2 Default Enumeration

The external master may simply load a VID, PID, and DID and use the default descriptor built into the *SX2*. To use the default descriptor, the descriptor length described in the previous section must equal 6. After the external master has written the length, the VID, PID, and DID must be written LSB, then MSB. For example, if the VID, PID, and DID are 0x04B4, 0x1002, and 0x0001 respectively, then the external master does the following:

- Initiates a Write Request to register 0x30.
- Writes two bytes (four command data transfers) that define the length of the entire descriptor about to be transferred. In this case, the length is always six.
- Writes the VID, PID, and DID bytes: 0xB4, 0x04, 0x02, 0x10, 0x01, 0x00 (in nibble format per the command protocol).

The default descriptor is listed in Default Descriptor on page 37. The default descriptor can be used as a starting point for a custom descriptor.

7. Endpoint 0

The *SX2* automatically responds to USB chapter 9 requests without any external master intervention. If the *SX2* receives a request to which it cannot respond automatically, the *SX2* notifies the external master. The external master then has the choice of responding to the request or stalling.

After the *SX2* receives a setup packet to which it cannot respond automatically, the *SX2* asserts a SETUP interrupt. After the external master reads the Interrupt Status Byte to determine that the interrupt source was the SETUP interrupt, it can initiate a read request to the SETUP register, 0x32. When the *SX2* sees a read request for the SETUP register, it presents the first byte of setup data to the external master. Each additional read request presents the next byte of setup data, until all eight bytes have been read.

The external master can stall this request at this or any other time. To stall a request, the external master initiates a write request for the SETUP register, 0x32, and writes any non-zero value to the register.

If this setup request has a data phase, the *SX2* then interrupts the external master with an EP0BUF interrupt when the buffer becomes available. The *SX2* determines the direction of the setup request and interrupts when either:

- IN: the Endpoint 0 buffer becomes available to write to, or
- OUT: the Endpoint 0 buffer receives a packet from the USB host.

For an IN setup transaction, the external master can write up to 64 bytes at a time for the data phase. The steps to write a packet are as follows:

1. Wait for an EP0BUF interrupt, indicating that the buffer is available.
2. Initiate a write request for register 0x31.
3. Write one data byte.
4. Repeat steps 2 and 3 until either all the data or 64 bytes have been written, whichever is less.
5. Write the number of bytes in this packet to the byte count register, 0x33.

Note

[6] These and all other data bytes must conform to the command protocol.

To send more than 64 bytes, the process is repeated. The *SX2* internally stores the length of the data phase that was specified in the wLength field (bytes 6,7) of the setup packet. To send less than the requested amount of data, the external master writes a packet that is less than 64 bytes, or if a multiple of 64, the external master follows the data with a zero-length packet. When the *SX2* sees a short or zero-length packet, it completes the setup transfer by automatically completing the handshake phase. The *SX2* does not enable more data than the wLength field specified in the setup packet. Note: the PKTEND pin does not apply to Endpoint 0. The only way to send a short or zero length packet is by writing to the byte count register with the appropriate value.

For an OUT setup transaction, the external master can read each packet received from the USB host during the data phase. The steps to read a packet are as follows:

1. Wait for an EP0BUF interrupt, indicating that a packet was received from the USB host into the buffer.
2. Initiate a read request for the byte count register, 0x33. This indicates the amount of data received from the host.
3. Initiate a read request for register 0x31.
4. Read one byte.
5. Repeat steps 3 and 4 until the number of bytes specified in the byte count register has been read.

To receive more than 64 bytes, the process is repeated. The *SX2* internally stores the length of the data phase that was specified in the wLength field of the setup packet (bytes 6,7). When the *SX2* sees that the specified number of bytes have been received, it completes the set up transfer by automatically completing the handshake phase. If the external master does not wish to receive the entire transfer, it can stall the transfer.

If the *SX2* receives another setup packet before the current transfer has completed, it interrupts the external master with another SETUP interrupt. If the *SX2* receives a setup packet with no data phase, the external master can accept the packet and complete the handshake phase by writing zero to the byte count register.

The *SX2* automatically responds to all USB standard requests covered in chapter 9 of the USB 2.0 specification except the Set/Clear Feature Endpoint requests. When the host issues a Set Feature or a Clear feature request, the *SX2* triggers a SETUP interrupt to the external master. The USB spec requires that the device respond to the Set endpoint feature request by doing the following:

- Set the STALL condition on that endpoint.

The USB spec requires that the device respond to the Clear endpoint feature request by doing the following:

- Reset the Data Toggle for that endpoint
- Clear the STALL condition of that endpoint.

The register that is used to reset the data toggle TOGCTL (located at XDATA location 0xE683) is not an index register that can be addressed by the command protocol presented in Command Protocol on page 7. The following section provides further information on this register bits and how to reset the data toggle accordingly using a different set of command protocol sequence.

7.1 Resetting Data Toggle

Table 7-1. Bit definition of the TOGCTL register

TOGCTL							0xE683	
Bit #	7	6	5	4	3	2	1	0
Bit Name	Q	S	R	I/O	EP3	EP2	EP1	EP0
Read/Write	R	W	W	R/W	R/W	R/W	R/W	R/W
Default	0	0	1	1	0	0	1	0

Bit 7: Q, *Data Toggle Value*

Q=0 indicates DATA0 and Q=1 indicates DATA1, for the endpoint selected by the I/O and EP3:0 bits. Write the endpoint select bits (IO and EP3:0), before reading this value.

Bit 6: S, *Set Data Toggle to DATA1*

After selecting the desired endpoint by writing the endpoint select bits (IO and EP3:0), set S=1 to set the data toggle to DATA1. The endpoint selection bits should not be changed while this bit is written.

Bit 5: R, *Set Data Toggle to DATA0*

Set R=1 to set the data toggle to DATA0. The endpoint selection bits should not be changed while this bit is written.

Bit 4: IO, *Select IN or OUT Endpoint*

Set this bit to select an endpoint direction prior to setting its R or S bit. IO=0 selects an OUT endpoint, IO = 1 selects an IN endpoint.

Bit 3-0: EP3:0, *Select Endpoint*

Set these bits to select an endpoint prior to setting its R or S bit. Valid values are 0, 1, 2, 6, and 8.

A two-step process is employed to clear an endpoint data toggle bit to 0. First, write to the TOGCTL register with an endpoint address (EP3:EP0) plus a direction bit (IO). Keeping the endpoint and direction bits the same, write a “1” to the R (reset) bit. For example, to clear the data toggle for EP6 configured as an “IN” endpoint, write the following values sequentially to TOGCTL:

00010110b

00110110b

Following is the sequence of events that the master should perform to set this register to 0x16:

- Send Low Byte of the Register (0x83)
    - ❐ Command **address** write of address 0x3A
    - ❐ Command **data** write of upper nibble of the Low Byte of Register Address (0x08)
    - ❐ Command **data** write of lower nibble of the Low Byte of Register Address (0x03)
- Send High Byte of the Register (0xE6)
    - ❐ Command **address** write of address 0x3B
    - ❐ Command **data** write of upper nibble of the High Byte of Register Address (0x0E)
    - ❐ Command **data** write of lower nibble of the High Byte of Register Address (0x06)
- Send the actual value to write to the register Register (in this case 0x16)
    - ❐ Command **address** write of address0x3C
    - ❐ Command **data** write of upper nibble of the register value (0x01)
    - ❐ Command **data** write of lower nibble of the register value (0x06)

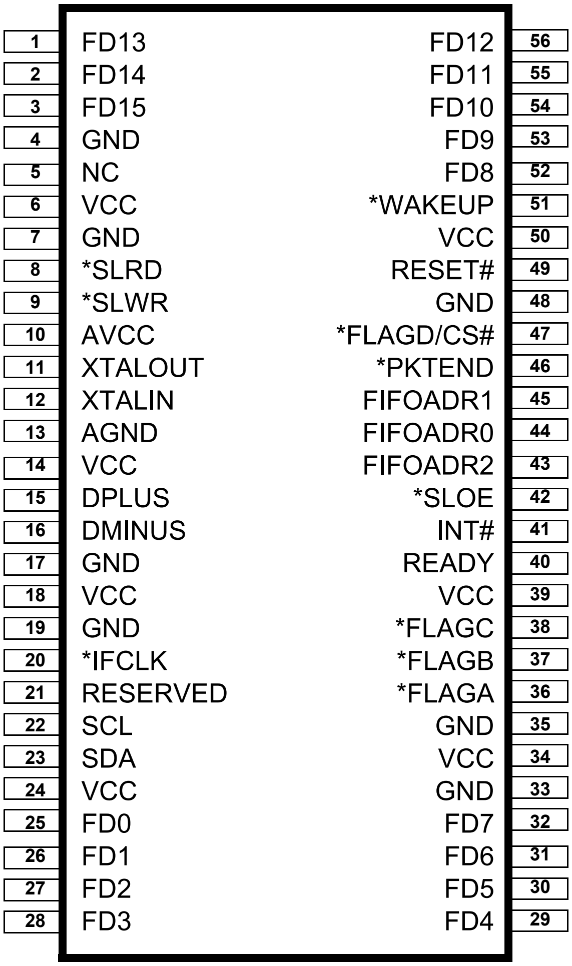
The same command sequence needs to be followed to set TOGCTL register to 0x36. The same command protocol sequence can be used to reset the data toggle for the other endpoints.

In order to read the status of this register, the external master must do the following sequence of events:

- Send Low Byte of the Register (0x83)
    - ❐ Command **address** write of 0x3A
    - ❐ Command **data** write of upper nibble of the Low Byte of Register Address (0x08)
    - ❐ Command **data** write of lower nibble of the Low Byte of Register Address (0x03)
- Send High Byte of the Register (0xE6)
    - ❐ Command **address** write of address 0x3B
    - ❐ Command **data** write of upper nibble of the High Byte of Register Address (0x0E)
    - ❐ Command **data** write of lower nibble of the High Byte of Register Address (0x06)
- Get the actual value from the TOGCTL register (0x16)
    - ❐ Command **address READ** of 0x3C

8. Pin Configurations

Figure 8-1. CY7C68001 56-Pin SSOP Pin Assignment[7]



Note

[7] A * denotes programmable polarity.

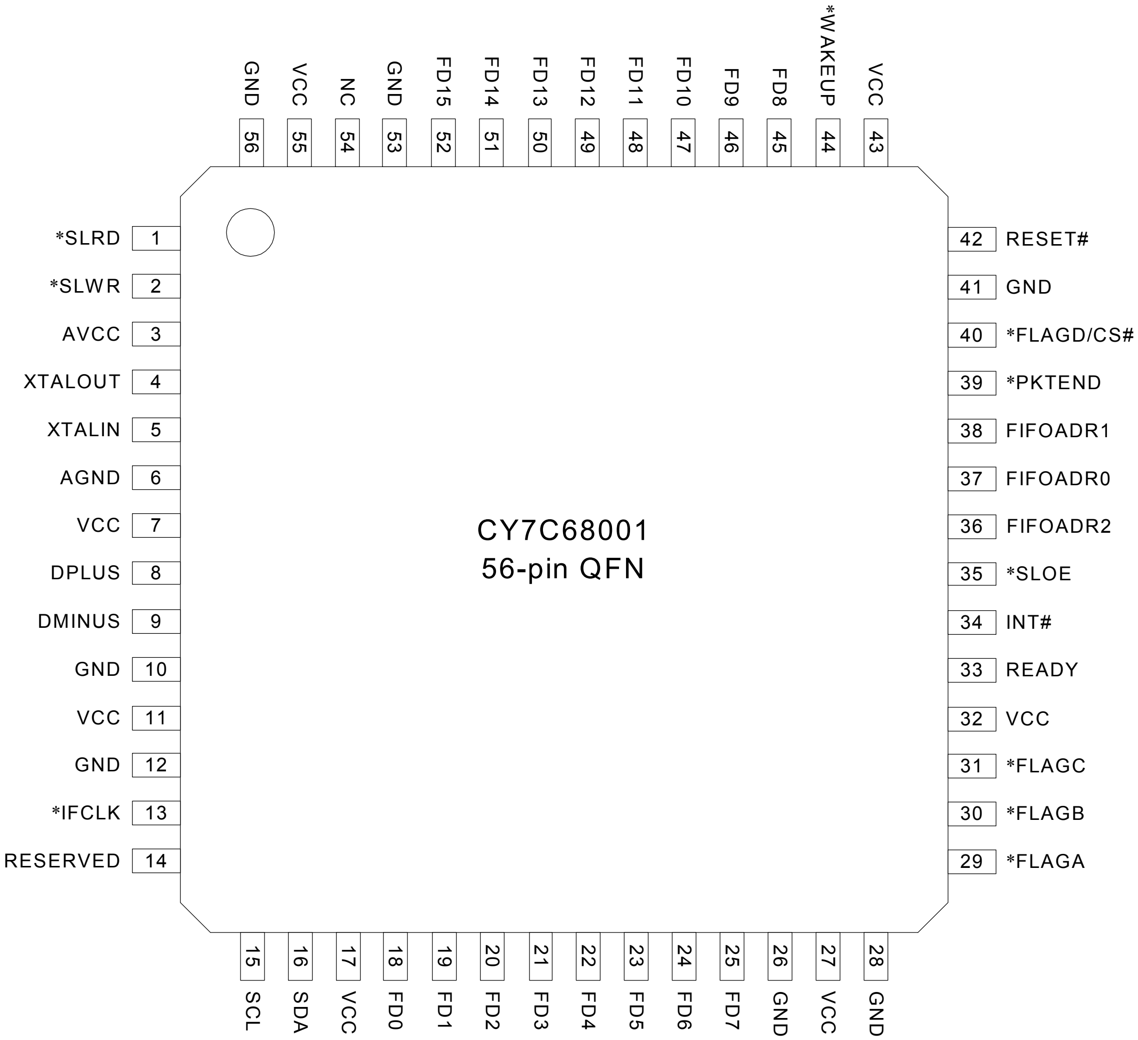


Figure 8-2. CY7C68001 56-pin QFN Assignment[7]

8.1 CY7C68001 Pin Definitions

Table 8-1. ***SX2*** **Pin Definitions**

QFN Pin	SSOP Pin	Name	Type	Default	Description
3	10	AVCC	Power	N/A	**Analog V_{CC}**. This signal provides power to the analog section of the chip.
6	13	AGND	Power	N/A	**Analog Ground**. Connect to ground with as short a path as possible.
9	16	DMINUS	I/O/Z	Z	**USB D– Signal**. Connect to the USB D– signal.
8	15	DPLUS	I/O/Z	Z	**USB D+ Signal**. Connect to the USB D+ signal.
42	49	RESET#	Input	N/A	**Active LOW Reset**. Resets the entire chip. This pin is normally tied to V_{CC} through a 100K resistor, and to GND through a 0.1-µF capacitor.
5	12	XTALIN	Input	N/A	**Crystal Input**. Connect this signal to a 24 MHz parallel-resonant, fundamental mode crystal and 20 pF capacitor to GND. It is also correct to drive XTALIN with an external 24 MHz square wave derived from another clock source.
4	11	XTALOUT	Output	N/A	**Crystal Output**. Connect this signal to a 24 MHz parallel-resonant, fundamental mode crystal and 20 pF capacitor to GND. If an external clock is used to drive XTALIN, leave this pin open.
54	5	NC	Output	O	**No Connect**. This pin must be left unconnected.
33	40	READY	Output	L	**READY** is an output-only ready that gates external command reads and writes. Active High.
34	41	INT#	Output	H	**INT#** is an output-only external interrupt signal. Active Low.
35	42	SLOE	Input	I	**SLOE** is an input-only output enable with programmable polarity (POLAR.4) for the slave FIFOs connected to FD[7:0] or FD[15:0].
36	43	FIFOADR2	Input	I	**FIFOADR2** is an input-only address select for the slave FIFOs connected to FD[7:0] or FD[15:0].
37	44	FIFOADR0	Input	I	**FIFOADR0** is an input-only address select for the slave FIFOs connected to FD[7:0] or FD[15:0].
38	45	FIFOADR1	Input	I	**FIFOADR1** is an input-only address select for the slave FIFOs connected to FD[7:0] or FD[15:0].
39	46	PKTEND	Input	I	**PKTEND** is an input-only packet end with programmable polarity (POLAR.5) for the slave FIFOs connected to FD[7:0] or FD[15:0].
40	47	FLAGD/CS#	CS#:I FLAGD:O	I	**FLAGD** is a programmable slave-FIFO output status flag signal. CS# is a master chip select (default).
18	25	FD[0]	I/O/Z	I	**FD[0]** is the bidirectional FIFO/Command data bus.
19	26	FD[1]	I/O/Z	I	**FD[1]** is the bidirectional FIFO/Command data bus.
20	27	FD[2]	I/O/Z	I	**FD[2]** is the bidirectional FIFO/Command data bus.
21	28	FD[3]	I/O/Z	I	**FD[3]** is the bidirectional FIFO/Command data bus.
22	29	FD[4]	I/O/Z	I	**FD[4]** is the bidirectional FIFO/Command data bus.
23	30	FD[5]	I/O/Z	I	**FD[5]** is the bidirectional FIFO/Command data bus.
24	31	FD[6]	I/O/Z	I	**FD[6]** is the bidirectional FIFO/Command data bus.
25	32	FD[7]	I/O/Z	I	**FD[7]** is the bidirectional FIFO/Command data bus.
45	52	FD[8]	I/O/Z	I	**FD[8]** is the bidirectional FIFO data bus.
46	53	FD[9]	I/O/Z	I	**FD[9]** is the bidirectional FIFO data bus.
47	54	FD[10]	I/O/Z	I	**FD[10]** is the bidirectional FIFO data bus.
48	55	FD[11]	I/O/Z	I	**FD[11]** is the bidirectional FIFO data bus.
49	56	FD[12]	I/O/Z	I	**FD[12]** is the bidirectional FIFO data bus.
50	1	FD[13]	I/O/Z	I	**FD[13]** is the bidirectional FIFO data bus.
51	2	FD[14]	I/O/Z	I	**FD[14]** is the bidirectional FIFO data bus.

Table 8-1. ***SX2*** **Pin Definitions** (continued)

QFN Pin	SSOP Pin	Name	Type	Default	Description
52	3	FD[15]	I/O/Z	I	**FD[15]** is the bidirectional FIFO data bus.
1	8	SLRD	Input	N/A	**SLRD** is the input-only read strobe with programmable polarity (POLAR.3) for the slave FIFOs connected to FD[7:0] or FD[15:0].
2	9	SLWR	Input	N/A	**SLWR** is the input-only write strobe with programmable polarity (POLAR.2) for the slave FIFOs connected to FD[7:0] or FD[15:0].
29	36	FLAGA	Output	H	**FLAGA** is a programmable slave-FIFO output status flag signal. Defaults to PF for the FIFO selected by the FIFOADR[2:0] pins.
30	37	FLAGB	Output	H	**FLAGB** is a programmable slave-FIFO output status flag signal. Defaults to FULL for the FIFO selected by the FIFOADR[2:0] pins.
31	38	FLAGC	Output	H	**FLAGC** is a programmable slave-FIFO output status flag signal. Defaults to EMPTY for the FIFO selected by the FIFOADR[2:0] pins.
13	20	IFCLK	I/O/Z	Z	**Interface Clock**, used for synchronously clocking data into or out of the slave FIFOs. IFCLK also serves as a timing reference for all slave FIFO control signals. When using the internal clock reference (IFCONFIG.7=1) the IFCLK pin can be configured to output 30/48 MHz by setting bits IFCONFIG.5 and IFCONFIG.6. IFCLK may be inverted by setting the bit IFCONFIG.4=1. Programmable polarity.
14	21	Reserved	Input	N/A	**Reserved**. Must be connected to ground.
44	51	WAKEUP	Input	N/A	**USB Wakeup**. If the *SX2* is in suspend, asserting this pin starts up the oscillator and interrupts the *SX2* to allow it to exit the suspend mode. During normal operation, holding WAKEUP asserted inhibits the *SX2* chip from suspending. This pin has programmable polarity (POLAR.7).
15	22	SCL	OD	Z	**I^{2}C Clock**. Connect to V_{CC} with a 2.2K-10 KOhms resistor, even if no I^{2}C EEPROM is attached.
16	23	SDA	OD	Z	**I^{2}C Data**. Connect to V_{CC} with a 2.2K-10 KOhms resistor, even if no I^{2}C EEPROM is attached.
55	6	V_{CC}	Power	N/A	**V_{CC}**. Connect to 3.3V power source.
7	14	V_{CC}	Power	N/A	**V_{CC}**. Connect to 3.3V power source.
11	18	V_{CC}	Power	N/A	**V_{CC}**. Connect to 3.3V power source.
17	24	V_{CC}	Power	N/A	**V_{CC}**. Connect to 3.3V power source.
27	34	V_{CC}	Power	N/A	**V_{CC}**. Connect to 3.3V power source.
32	39	V_{CC}	Power	N/A	**V_{CC}**. Connect to 3.3V power source.
43	50	V_{CC}	Power	N/A	**V_{CC}**. Connect to 3.3V power source.
53	4	GND	Ground	N/A	**Connect to ground**.
56	7	GND	Ground	N/A	**Connect to ground**.
10	17	GND	Ground	N/A	**Connect to ground**.
12	19	GND	Ground	N/A	**Connect to ground**.
26	33	GND	Ground	N/A	**Connect to ground**.
28	35	GND	Ground	N/A	**Connect to ground**.
41	48	GND	Ground	N/A	**Connect to ground**.

9. Register Summary

Table 9-1. *SX2* Register Summary

Hex	Size	Name	Description	D7	D6	D5	D4	D3	D2	D1	D0	Default	Access
		General Configuration											
01	1	IFCONFIG	Interface Configuration	IFCLKSRC	3048MHZ	IFCLKOE	IFCLKPOL	ASYNC	STANDBY	FLAGD/CS#	DISCON	11001001	bbbbbbbb
02	1	FLAGSAB	FIFO FLAGA and FLAGB Assignments	FLAGB3	FLAGB2	FLAGB1	FLAGB0	FLAGA3	FLAGA2	FLAGA1	FLAGA0	00000000	bbbbbbbb
03	1	FLAGSCD	FIFO FLAGC and FLAGD Assignments	FLAGD3	FLAGD2	FLAGD1	FLAGD0	FLAGC3	FLAGC2	FLAGC1	FLAGC0	00000000	bbbbbbbb
04	1	POLAR	FIFO polarities	WUPOL	0	PKTEND	SLOE	SLRD	SLWR	EF	FF	00000000	bbbrrrbb
05	1	REVID	Chip Revision	Major	Major	Major	Major	minor	minor	minor	minor	xxxxxxxx	rrrrrrrr
		Endpoint Configuration[9]											
06	1	EP2CFG	Endpoint 2 Configuration	VALID	dir	TYPE1	TYPE0	SIZE	STALL	BUF1	BUF0	10100010	bbbbbbbb
07	1	EP4CFG	Endpoint 4 Configuration	VALID	dir	TYPE1	TYPE0	0	STALL	0	0	10100000	bbbbrbrr
08	1	EP6CFG	Endpoint 6 Configuration	VALID	dir	TYPE1	TYPE0	SIZE	STALL	BUF1	BUF0	11100010	bbbbbbbb
09	1	EP8CFG	Endpoint 8 Configuration	VALID	dir	TYPE1	TYPE0	0	STALL	0	0	11100000	bbbbrbrr
0A	1	EP2PKTLENH	Endpoint 2 Packet Length H	INFM1	OEP1	ZEROLEN	WORDWIDE	0	PL10	PL9	PL8	00110010	bbbbbbbb
0B	1	EP2PKTLENL	Endpoint 2 Packet Length L (IN only)	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0	00000000	bbbbbbbb
0C	1	EP4PKTLENH	Endpoint 4 Packet Length H	INFM1	OEP1	ZEROLEN	WORDWIDE	0	0	PL9	PL8	00110010	bbbbbbbb
0D	1	EP4PKTLENL	Endpoint 4 Packet Length L (IN only)	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0	00000000	bbbbbbbb
0E	1	EP6PKTLENH	Endpoint 6 Packet Length H	INFM1	OEP1	ZEROLEN	WORDWIDE	0	PL10	PL9	PL8	00110010	bbbbbbbb
0F	1	EP6PKTLENL	Endpoint 6 Packet Length L (IN only)	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0	00000000	bbbbbbbb
10	1	EP8PKTLENH	Endpoint 8 Packet Length H	INFM1	OEP1	ZEROLEN	WORDWIDE	0	0	PL9	PL8	00110010	bbbbbbbb
11	1	EP8PKTLENL	Endpoint 8 Packet Length L (IN only)	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0	00000000	bbbbbbbb
12	1	EP2PFH	EP2 Programmable Flag H	DECIS	PKTSTAT	IN: PKTS[2] OUT:PFC12	IN: PKTS[1] OUT:PFC11	IN: PKTS[0] OUT:PFC10	0	PFC9	PFC8	10001000	bbbbbbbb
13	1	EP2PFL	EP2 Programmable Flag L	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0	00000000	bbbbbbbb
14	1	EP4PFH	EP4 Programmable Flag H	DECIS	PKTSTAT	0	IN: PKTS[1] OUT:PFC10	IN: PKTS[0] OUT:PFC9	0	0	PFC8	10001000	bbbbbbbb
15	1	EP4PFL	EP4 Programmable Flag L	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0	00000000	bbbbbbbb
16	1	EP6PFH	EP6 Programmable Flag H	DECIS	PKTSTAT	IN: PKTS[2] OUT:PFC12	IN: PKTS[1] OUT:PFC11	IN: PKTS[0] OUT:PFC10	0	PFC9	PFC8	00001000	bbbbbbbb
17	1	EP6PFL	EP6 Programmable Flag L	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0	00000000	bbbbbbbb
18	1	EP8PFH	EP8 Programmable Flag H	DECIS	PKTSTAT	0	IN: PKTS[1] OUT:PFC10	IN: PKTS[0] OUT:PFC9	0	0	PFC8	00001000	bbbbbbbb
19	1	EP8PFL	EP8 Programmable Flag L	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0	00000000	bbbbbbbb
1A	1	EP2ISOINPKTS	EP2 (if ISO) IN Packets per frame (1-3)	0	0	0	0	0	0	INPPF1	INPPF0	00000001	bbbbbbbb
1B	1	EP4ISOINPKTS	EP4 (if ISO) IN Packets per frame (1-3)	0	0	0	0	0	0	INPPF1	INPPF0	00000001	bbbbbbbb
1C	1	EP6ISOINPKTS	EP6 (if ISO) IN Packets per frame (1-3)	0	0	0	0	0	0	INPPF1	INPPF0	00000001	bbbbbbbb
1D	1	EP8ISOINPKTS	EP8 (if ISO) IN Packets per frame (1-3)	0	0	0	0	0	0	INPPF1	INPPF0	00000001	bbbbbbbb
		FLAGS											
1E	1	EP24FLAGS	Endpoints 2,4 FIFO Flags	0	EP4PF	EP4EF	EP4FF	0	EP2PF	EP2EF	EP2FF	00100010	rrrrrrrr
1F	1	EP68FLAGS	Endpoints 6,8 FIFO Flags	0	EP8PF	EP8EF	EP8FF	0	EP6PF	EP6EF	EP6FF	01100110	rrrrrrrr
		INPKTEND/FLUSH[10]											
20	1	INPKTEND/FLUSH	Force Packet End / Flush FIFOs	FIFO8	FIFO6	FIFO4	FIFO2	EP3	EP2	EP1	EP0	00000000	wwwwwwww
		USB Configuration											
2A	1	USBFRAMEH	USB Frame count H	0	0	0	0	0	FC10	FC9	FC8	xxxxxxxx	rrrrrrrr
2B	1	USBFRAMEL	USB Frame count L	FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0	xxxxxxxx	rrrrrrrr
2C	1	MICROFRAME	Microframe count, 0-7	0	0	0	0	0	MF2	MF1	MF0	xxxxxxxx	rrrrrrrr
2D	1	FNADDR	USB Function address	HSGRANT	FA6	FA5	FA4	FA3	FA2	FA1	FA0	00000000	rrrrrrrr
		Interrupts											

Table 9-1. ***SX2*** **Register Summary** (continued)

Hex	Size	Name	Description	D7	D6	D5	D4	D3	D2	D1	D0	Default	Access
2E	1	INTENABLE	Interrupt Enable	SETUP	EP0BUF	FLAGS	1	1	ENUMOK	BUSACTIVITY	READY	11111111	bbbbbbbb
		Descriptor											
30	500	DESC	Descriptor RAM	d7	d6	d5	d4	d3	d2	d1	d0	xxxxxxxx	wwwwwwww
		Endpoint 0											
31	64	EP0BUF	Endpoint 0 Buffer	d7	d6	d5	d4	d3	d2	d1	d0	xxxxxxxx	bbbbbbbb
32	8/1	SETUP	Endpoint 0 Setup Data / Stall	d7	d6	d5	d4	d3	d2	d1	d0	xxxxxxxx	bbbbbbbb
33	1	EP0BC	Endpoint 0 Byte Count	d7	d6	d5	d4	d3	d2	d1	d0	xxxxxxxx	bbbbbbbb
		Un-Indexed Register control											
3A	1		Un-Indexed Register Low Byte pointer	a7	a6	a5	a4	a3	a2	a1	a0		
3B	1		Un-Indexed Register High Byte pointer	a7	a6	a5	a4	a3	a2	a1	a0		
3C	1		Un-Indexed Register Data	d7	d6	d5	d4	d3	d2	d1	d0		
Address		Un-Indexed Registers in XDATA Space											
0xE609		FIFOPIN-POLAR	FIFO Interface Pins Polarity	0	0	PKTEND	SLOE	SLRD	SLWR	EF	FF	00000000	rrbbbbbb
0xE683		TOGCTL	Data Toggle Control	Q	S	R	IO	EP3	EP2	EP1	EP0	xxxxxxxx	rbbbbbbb

8.

Notes

9. Please note that the *SX2* was not designed to support dynamic modification of these endpoint configuration registers. If your applications need the ability to change endpoint configurations after the device has already enumerated with a specific configuration, please expect some delay in being able to access the FIFOs after changing the configuration. For example, after writing to EP2PKTLENH, you must wait for at least 35 µs measured from the time the READY signal is asserted before writing to the FIFO. This delay time varies for different registers and is not characterized, because the *SX2* was not designed for this dynamic change of endpoint configuration registers.
10. Please note that the *SX2* was not designed to support dynamic modification of the INPKTEND/FLUSH register. If your applications need the ability to change endpoint configurations or access the INPKTEND register after the device has already enumerated with a specific configuration, please expect some delay in being able to access the FIFOs after changing this register. After writing to INPKTEND/FLUSH, you must wait for at least 85 µs measured from the time the READY signal is asserted before writing to the FIFO. This delay time varies for different registers and is not characterized, because the *SX2* was not designed for this dynamic change of endpoint configuration registers.

9.1 IFCONFIG Register 0x01

IFCONFIG							0x01
Bit #	7	6	5	4	3	2	1	0
Bit Name	IFCLKSRC	3048 MHZ	IFCLKOE	IFCLKPOL	ASYNC	STANDBY	FLAGD/CS#	DISCON
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	0	0	1	0	0	1

9.1.1 Bit 7: IFCLKSRC

This bit selects the clock source for the FIFOs. If IFCLKSRC = 0, the external clock on the IFCLK pin is selected. If IFCLKSRC = 1 (default), an internal 30 or 48 MHz clock is used.

9.1.2 Bit 6: 3048 MHZ

This bit selects the internal FIFO clock frequency. If 3048 MHZ = 0, the internal clock frequency is 30 MHz. If 3048 MHZ = 1 (default), the internal clock frequency is 48 MHz.

9.1.3 Bit 5: IFCLKOE

This bit selects if the IFCLK pin is driven. If IFCLKOE = 0 (default), the IFCLK pin is floated. If IFCLKOE = 1, the IFCLK pin is driven.

9.1.4 Bit 4: IFCLKPOL

This bit controls the polarity of the IFCLK signal.

- When IFCLKPOL=0, the clock has the polarity shown in all the timing diagrams in this data sheet (rising edge is the activating edge).
- When IFCLKPOL=1, the clock is inverted (in some cases may help with satisfying data setup times).

9.1.5 Bit 3: ASYNC

This bit controls whether the FIFO interface is synchronous or asynchronous. When ASYNC = 0, the FIFOs operate synchronously. In synchronous mode, a clock is supplied either internally or externally on the IFCLK pin, and the FIFO control signals function as read and write enable signals for the clock signal.

When ASYNC = 1 (default), the FIFOs operate asynchronously. No clock signal input to IFCLK is required, and the FIFO control signals function directly as read and write strobes.

9.1.6 Bit 2: STANDBY

This bit instructs the *SX2* to enter a low power mode. When STANDBY=1, the *SX2* enters a low power mode by turning off its oscillator. The external master should write this bit after it receives a bus activity interrupt (indicating that the host has signaled a USB suspend condition). If *SX2* is disconnected from the USB bus, the external master can write this bit at any time to save power. Once suspended, the *SX2* is awakened either by resumption of USB bus activity or by assertion of its WAKEUP pin.

9.1.7 Bit 1: FLAGD/CS#

This bit controls the function of the FLAGD/CS# pin. When FLAGD/CS# = 0 (default), the pin operates as a slave chip select. If FLAGD/CS# = 1, the pin operates as FLAGD.

9.1.8 Bit 0: DISCON

This bit controls whether the internal pull up resistor connected to D+ is pulled high or floating. When DISCON = 1 (default), the pull up resistor is floating simulating a USB unplug. When DISCON=0, the pull up resistor is pulled high signaling a USB connection.

9.2 FLAGSAB/FLAGSCD Registers 0x02/0x03

The *SX2* has four FIFO flags output pins: FLAGA, FLAGB, FLAGC, FLAGD.

FLAGSAB								0x02
Bit #	7	6	5	4	3	2	1	0
Bit Name	FLAGB3	FLAGB2	FLAGB1	FLAGB0	FLAGA3	FLAGA2	FLAGA1	FLAGA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

FLAGSCD								0x03
Bit #	7	6	5	4	3	2	1	0
Bit Name	FLAGD3	FLAGD2	FLAGD1	FLAGD0	FLAGC3	FLAGC2	FLAGC1	FLAGC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

These flags can be programmed to represent various FIFO flags using four select bits for each FIFO. The 4-bit coding for all four flags is the same, as shown in Table 9-2.

Table 9-2. FIFO Flag 4-bit Coding

FLAGx3	FLAGx2	FLAGx1	FLAGx0	Pin Function
0	0	0	0	FLAGA = PF, FLAGB = FF, FLAGC = EF, FLAGD = CS# (actual FIFO is selected by FIFOADR[2:0] pins)
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	Reserved
0	1	0	0	EP2 PF
0	1	0	1	EP4 PF
0	1	1	0	EP6 PF
0	1	1	1	EP8 PF
1	0	0	0	EP2 EF
1	0	0	1	EP4 EF
1	0	1	0	EP6 EF
1	0	1	1	EP8 EF
1	1	0	0	EP2 FF
1	1	0	1	EP4 FF
1	1	1	0	EP6 FF
1	1	1	1	EP8 FF

For the default (0000) selection, the four FIFO flags are fixed-function as shown in the first table entry; the input pins FIFOADR[2:0] select to which of the four FIFOs the flags correspond. These pins are decoded as shown in Table 5-3.

The other (non-zero) values of FLAGx[3:0] allow the designer to independently configure the four flag outputs FLAGA-FLAGD to correspond to any flag-Programmable, Full, or Empty-from any of the four endpoint FIFOs. This allows each flag to be assigned to any of the four FIFOs, including those not currently selected by the FIFOADR [2:0] pins. For example, the external master could be filling the EP2IN FIFO with data while also checking the empty flag for the EP4OUT FIFO.

9.3 POLAR Register 0x04

This register controls the polarities of FIFO pin signals and the WAKEUP pin.

POLAR							0x04	
Bit #	7	6	5	4	3	2	1	0
Bit Name	WUPOL	0	PKTEND	SLOE	SLRD	SLWR	EF	FF
Read/ Write	R/W	R/W	R/W	R	R	R	R/W	R/W
Default	0	0	0	0	0	0	0	0

9.3.1 Bit 7: WUPOL

This flag sets the polarity of the WAKEUP pin. If WUPOL = 0 (default), the polarity is active LOW. If WUPOL=1, the polarity is active HIGH.

9.3.2 Bit 5: PKTEND

This flag selects the polarity of the PKTEND pin. If PKTEND = 0 (default), the polarity is active LOW. If PKTEND = 1, the polarity is active HIGH.

9.3.3 Bit 4: SLOE

This flag selects the polarity of the SLOE pin. If SLOE = 0 (default), the polarity is active LOW. If SLOE = 1, the polarity is active HIGH. This bit can only be changed by using the EEPROM configuration load.

9.3.4 Bit 3: SLRD

This flag selects the polarity of the SLRD pin. If SLRD = 0 (default), the polarity is active LOW. If SLRD = 1, the polarity is active HIGH. This bit can only be changed by using the EEPROM configuration load.

9.3.5 SLWR Bit 2

This flag selects the polarity of the SLWR pin. If SLWR = 0 (default), the polarity is active LOW. If SLWR = 1, the polarity is active HIGH. This bit can only be changed by using the EEPROM configuration load.

9.3.6 EF Bit 1

This flag selects the polarity of the EF pin (FLAGA/B/C/D). If EF = 0 (default), the EF pin is pulled low when the FIFO is empty. If EF = 1, the EF pin is pulled HIGH when the FIFO is empty.

9.3.7 FF Bit 0

This flag selects the polarity of the FF pin (FLAGA/B/C/D). If FF = 0 (default), the FF pin is pulled low when the FIFO is full. If FF = 1, the FF pin is pulled HIGH when the FIFO is full.

Note that bits 2(SLWR), 3(SLRD) and 4 (SLOE) are READ only bits and cannot be set by the external master or the EEPROM. On power up, these bits are set to active low polarity. In order to change the polarity after the device is powered-up, the external master must access the previously undocumented (un-indexed) *SX2* register located at XDATA space at 0xE609. This register has exact same bit definition as the POLAR register except that bits 2, 3 and 4 defined as SLWR, SLRD and SLOE respectively are Read/Write bits. Following is the sequence of events that the master should perform for setting this register to 0x1C (setting bits 4, 3, and 2):

1. Send Low Byte of the Register (0x09)
   a. Command address write of address 0x3A
   b. Command data write of upper nibble of the Low Byte of Register Address (0x00)
   c. Command data write of lower nibble of the Low Byte of Register Address (0x09)
2. Send High Byte of the Register (0xE6)
   d. Command address write of address 0x3B
   e. Command data write of upper nibble of the High Byte of Register Address (0x0E)
   f. Command data write of lower nibble of the High Byte of Register Address (0x06)

3. Send the actual value to write to the register Register (in this case 0x1C)
   g. Command address write of address 0x3C
   h. Command data write of upper nibble of the register value (0x01)
   i. Command data write of lower nibble of the register value (0x0C)

In order to avoid altering any other bits of the FIFOPINPOLAR register (0xE609) inadvertently, the external master must do a read (from POLAR register), modify the value to set/clear appropriate bits and write the modified value to FIFOPINPOLAR register. The external master may read from the POLAR register using the command read protocol as stated in Command Protocol on page 7. Modify the value with the appropriate bit set to change the polarity as needed and write this modified value to the FIFOPINPOLAR register.

9.4 REVID Register 0x05

These register bits define the silicon revision.

REVID								0x05
Bit #	7	6	5	4	3	2	1	0
Bit Name	Major	Major	Major	Major	Minor	Minor	Minor	Minor
Read/ Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
De- fault	X	X	X	X	X	X	X	X

The upper nibble is the major revision. The lower nibble is the minor revision. For example: if REVID = 0x11, then the silicon revision is 1.1.

9.5 EPxCFG Register 0x06–0x09

These registers configure the large, data-handling *SX2* endpoints, EP2, 4, 6, and 8. Figure 8-1. on page 11 shows the configuration choices for these endpoints. Shaded blocks group endpoint buffers for double-, triple-, or quad-buffering. The endpoint direction is set independently—any shaded block can have any direction.

EPx- CFG								0x06, 0x08
Bit #	7	6	5	4	3	2	1	0
Bit Name	VALID	DIR	TYPE1	TYPE0	SIZE	STALL	BUF1	BUF0
Read/ Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
De- fault	1	0	1	0	0	0	1	0

9.5.1 Bit 7: VALID

The external master sets VALID = 1 to activate an endpoint, and VALID = 0 to deactivate it. All *SX2* endpoints default to valid. An endpoint whose VALID bit is 0 does not respond to any USB traffic. (Note: when setting VALID=0, use default values for all other bits.)

9.5.2 Bit 6: DIR

0 = OUT, 1 = IN. Defaults for EP2/4 are DIR = 0, OUT, and for EP6/8 are DIR = 1, IN.

9.5.3 Bit [5,4]: TYPE1, TYPE0

These bits define the endpoint type, as shown in Table 9-3. The TYPE bits apply to all of the endpoint configuration registers. All *SX2* endpoints except EP0 default to BULK.

Table 9-3. Endpoint Type

TYPE1	TYPE0	Endpoint Type
0	0	Invalid
0	1	Isochronous
1	0	Bulk (Default)
1	1	Interrupt

9.5.4 Bit 3: SIZE

0 = 512 bytes (default), 1 = 1024 bytes.

Endpoints 4 and 8 can only be 512 bytes and is a read only bit. The size of endpoints 2 and 6 is selectable.

9.5.5 Bit 2: STALL

Each bulk endpoint (IN or OUT) has a STALL bit (bit 2). If the external master sets this bit, any requests to the endpoint return a STALL handshake rather than ACK or NAK. The Get Status-Endpoint Request returns the STALL state for the endpoint indicated in byte 4 of the request. Note that bit 7 of the endpoint number EP (byte 4) specifies direction.

9.5.6 Bit [1,0]: BUF1, BUF0

For EP2 and EP6 the depth of endpoint buffering is selected via BUF1:0, as shown in Table 9-4. For EP4 and EP8 the buffer is internally set to double buffered and are read only bits.

Table 9-4. Endpoint Buffering

BUF1	BUF0	Buffering
0	0	Quad
0	1	Invalid[11]
1	0	Double
1	1	Triple

Note

[11] Setting the endpoint buffering to invalid causes improper buffer allocation

9.6 EPxPKTLENH/L Registers 0x0A–0x11

The external master can use these registers to set smaller packet sizes than the physical buffer size (refer to the previously described EPxCFG registers). The default packet size is 512 bytes for all endpoints. Note that EP2 and EP6 can have maximum sizes of 1024 bytes, and EP4 and EP8 can have maximum sizes of 512 bytes, to be consistent with the endpoint structure.

In addition, the EPxPKTLENH register has four other endpoint configuration bits.

EPxPK-TLENL							0x0B, 0x0D, 0x0F, 0x11
Bit #	7	6	5	4	3	2	1	0
Bit Name	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

EP2PKTLENH, EP6PKTLENH							0x0A, 0x0E	
Bit #	7	6	5	4	3	2	1	0
Bit Name	INFM1	OEP1	ZERO LEN	WORD WIDE	0	PL10	PL9	PL8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	1	1	0	0	1	0

EP4PKTLENH, EP8PKTLENH							0x0C, 0x10	
Bit #	7	6	5	4	3	2	1	0
Bit Name	INFM1	OEP1	ZERO LEN	WORD WIDE	0	0	PL9	PL8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	1	1	0	0	1	0

9.6.1 Bit 7: INFM1 EPxPKTLENH.7

When the external master sets INFM = 1 in an endpoint configuration register, the FIFO flags for that endpoint become valid one sample earlier than when the full condition occurs. These bits take effect only when the FIFOs are operating synchronously according to an internally or externally supplied clock. Having the FIFO flag indications one sample early simplifies some synchronous interfaces. This applies only to IN endpoints. Default is INFM1 = 0.

9.6.2 Bit 6: OEP1 EPxPKTLENH.6

When the external master sets an OEP = 1 in an endpoint configuration register, the FIFO flags for that endpoint become valid one sample earlier than when the empty condition occurs. These bits take effect only when the FIFOs are operating synchronously according to an internally or externally supplied clock. Having the FIFO flag indications one sample early simplifies some synchronous interfaces. This applies only to OUT endpoints. Default is OEP1 = 0.

9.6.3 Bit 5: ZEROLEN EPxPKTLENH.5

When ZEROLEN = 1 (default), a zero length packet is sent when the PKTEND pin is asserted and there are no bytes in the current packet. If ZEROLEN = 0, then a zero length packet is not sent under these conditions.

9.6.4 Bit 4: WORDWIDE EPxPKTLENH.4

This bit controls whether the data interface is 8 or 16 bits wide. If WORDWIDE = 0, the data interface is eight bits wide, and FD[15:8] have no function. If WORDWIDE = 1 (default), the data interface is 16 bits wide.

9.6.5 Bit [2..0]: PL[X:0] Packet Length Bits

The default packet size is 512 bytes for all endpoints.

9.7 EPxPFH/L Registers 0x12–0x19

The Programmable Flag registers control when the PF goes active for each of the four endpoint FIFOs: EP2, EP4, EP6, and EP8. The EPxPFH/L fields are interpreted differently for the high speed operation and full speed operation and for OUT and IN endpoints.

Following is the register bit definition for high speed operation and for full speed operation (when endpoint is configured as an isochronous endpoint).

Full Speed ISO and High Speed Mode: EP2PFL, EP4PFL, EP6PFL, EP8PFL							0x13, 0x15, 0x17, 0x19	
Bit #	7	6	5	4	3	2	1	0
Bit Name	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Full Speed ISO and High Speed Mode: EP4PFH, EP8PFH							0x14, 0x18	
Bit #	7	6	5	4	3	2	1	0
Bit Name	DECIS	PKTSTAT	0	IN: PKTS[1] OUT: PFC10	IN: PKTS[0] OUT: PFC9	0	0	PFC8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	1	0	0	0

Full Speed ISO and High Speed Mode: EP2PFH, EP6PFH							0x12, 0x16	
Bit #	7	6	5	4	3	2	1	0
Bit Name	DECIS	PKTSTAT	IN: PKTS[2] OUT: PFC12	IN: PKTS[1] OUT: PFC11	IN: PKTS[0] OUT: PFC10	0	PFC9	PFC8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	0	0	0	1	0	0	0

Following is the bit definition for the same register when the device is operating at full speed and the endpoint is not configured as isochronous endpoint.

Full Speed Non-ISO Mode: EP2PFL, EP4PFL, EP6PFL, EP8PFL							0x13, 0x15, 0x17, 0x19	
Bit #	7	6	5	4	3	2	1	0
Bit Name	IN: PKTS[1] OUT: PFC7	IN: PKTS[0] OUT: PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Full Speed Non-ISO Mode: EP2PFH, EP6PFH							0x12, 0x16	
Bit #	7	6	5	4	3	2	1	0
Bit Name	DECIS	PKTSTAT	OUT: PFC12	OUT: PFC11	OUT: PFC10	0	PFC9	IN: PKTS[2] OUT: PFC8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	0	0	0	1	0	0	0

Full Speed Non-ISO Mode: EP4PFH, EP8PFH							0x14, 0x18	
Bit #	7	6	5	4	3	2	1	0
Bit Name	DECIS	PKT-STAT	0	OUT: PFC10	OUT: PFC9	0	0	PFC8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	1	0	0	0

9.7.1 DECIS: EPxPFH.7

If DECIS = 0, then PF goes high when the byte count i is equal to or less than what is defined in the PF registers. If DECIS = 1 (default), then PF goes high when the byte count equal to or greater than what is set in the PF register. For OUT endpoints, the byte count is the total number of bytes in the FIFO that are available to the external master. For IN endpoints, the byte count is determined by the PKSTAT bit.

9.7.2 PKSTAT: EPxPFH.6

For IN endpoints, the PF can apply to either the entire FIFO, comprising multiple packets, or only to the current packet being filled. If PKTSTAT = 0 (default), the PF refers to the entire IN endpoint FIFO. If PKTSTAT = 1, the PF refers to the number of bytes in the current packet.

PKTSTAT	PF applies to	EPnPFH:L format
0	Number of committed packets + current packet bytes	PKTS[] and PFC[]
1	Current packet bytes only	PFC[ ]

9.7.3 IN: PKTS(2:0)/OUT: PFC[12:10]: EPxPFH[5:3]

These three bits have a different meaning, depending on whether this is an IN or OUT endpoint.

IN Endpoints

If IN endpoint, the meaning of this *EPxPFH[5:3]* bits depend on the PKTSTAT bit setting. When PKTSTAT = 0 (default), the PF considers when there are PKTS packets plus PFC bytes in the FIFO. PKTS[2:0] determines how many packets are considered, according to Table 9-5.

Table 9-5. PKTS Bits

PKTS2	PKTS1	PKTS0	Number of Packets
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4

When PKTSTAT = 1, the PF considers when there are PFC bytes in the FIFO, no matter how many packets are in the FIFO. The PKTS[2:0] bits are ignored.

OUT Endpoints

The PF considers when there are PFC bytes in the FIFO regardless of the PKTSTAT bit setting.

9.8 EPxISOINPKTS Registers 0x1A–0x1D

EP2ISOINOKTS, EP4ISOINPKTS, EP6ISOINPKTS, EP8ISOINPKTS							0x1A, 0x1B, 0x1C, 0x1D	
Bit #	7	6	5	4	3	2	1	0
Bit Name	0	0	0	0	0	INPPF2	INPPF1	INPPF0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	1

For ISOCHRONOUS IN endpoints only, these registers determine the number of packets per frame (only one per frame for full speed mode) or microframe (up to three per microframe for high speed mode), according to the following table.

Table 9-6. EPxISOINPKTS

INPPF1	INPPF0	Packets
0	0	Invalid
0	1	1 (default)
1	0	2
1	1	3

9.9 EPxxFLAGS Registers 0x1E–0x1F

The EPxxFLAGS provide an alternate way of checking the status of the endpoint FIFO flags. If enabled, the *SX2* can interrupt the external master when a flag is asserted, and the external master can read these two registers to determine the state of the FIFO flags. If the INFM1 and/or OEP1 bits are set, then the EPxEF and EPxFF bits are actually empty +1 and full –1.

EP24FLAGS								0x1E
Bit #	7	6	5	4	3	2	1	0
Bit Name	0	EP4PF	EP4EF	EP4FF	0	EP2PF	EP2EF	EP2FF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	1	0	0	0	1	0

EP68FLAGS								0x1F
Bit #	7	6	5	4	3	2	1	0
Bit Name	0	EP8PF	EP8EF	EP8FF	0	EP6PF	EP6EF	EP6FF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	1	0	0	0	1	0

9.9.1 EPxPF Bit 6, Bit 2

This bit is the current state of endpoint x's programmable flag.

9.9.2 EPxEF Bit 5, Bit 1

This bit is the current state of endpoint x's empty flag. EPxEF = 1 if the endpoint is empty.

9.9.3 EPxFF Bit 4, Bit 0

This bit is the current state of endpoint x's full flag. EPxFF = 1 if the endpoint is full.

9.10 INPKTEND/FLUSH Register 0x20

This register allows the external master to duplicate the function of the PKTEND pin. The register also allows the external master to selectively flush endpoint FIFO buffers.

INPKTEND/FLUSH								0x20
Bit #	7	6	5	4	3	2	1	0
Bit Name	FIFO8	FIFO6	FIFO4	FIFO2	EP3	EP2	EP1	EP0
Read/Write	W	W	W	W	W	W	W	W
Default	0	0	0	0	0	0	0	0

Bit [4..7]: FIFOx

These bits allows the external master to selectively flush any or all of the endpoint FIFOs. By writing the desired endpoint FIFO bit, *SX2* logic flushes the selected FIFO. For example setting bit 7 flushes endpoint 8 FIFO.

Bit [3..0]: EPx

These bits are is used only for IN transfers. By writing the desired endpoint number (2,4,6 or 8), *SX2* logic automatically commits an IN buffer to the USB host. For example, for committing a packet through endpoint 6, set the lower nibble to 6: set bits 1 and 2 high.

9.11 USBFRAMEH/L Registers 0x2A, 0x2B

Every millisecond, the USB host sends an SOF token indicating "Start Of Frame," along with an 11-bit incrementing frame count. The *SX2* copies the frame count into these registers at every SOF.

USBFRAMEH								0x2A
Bit #	7	6	5	4	3	2	1	0
Bit Name	0	0	0	0	0	FC10	FC9	FC8
Read/Write	R	R	R	R	R	R	R	R
Default	X	X	X	X	X	X	X	x

USBFRAMEL								0x2B
Bit #	7	6	5	4	3	2	1	0
Bit Name	FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0
Read/Write	R	R	R	R	R	R	R	R
Default	X	X	X	X	X	X	X	X

One use of the frame count is to respond to the USB SYNC_FRAME Request. If the *SX2* detects a missing or garbled SOF, the *SX2* generates an internal SOF and increments USBFRAMEL–USBRAMEH.

9.12 MICROFRAME Registers 0x2C

MICROFRAME								0x2C
Bit #	7	6	5	4	3	2	1	0
Bit Name	0	0	0	0	0	MF2	MF1	MF0
Read/Write	R	R	R	R	R	R	R	R
Default	X	X	X	X	X	X	X	x

MICROFRAME contains a count 0–7 that indicates which of the 125 microsecond microframes last occurred.

This register is active only when *SX2* is operating in high speed mode (480 Mbits/sec).

9.13 FNADDR Register 0x2D

During the USB enumeration process, the host sends a device a unique 7-bit address that the *SX2* copies into this register. There is normally no reason for the external master to know its USB device address because the *SX2* automatically responds only to its assigned address.

FNADDR								0x2D
Bit #	7	6	5	4	3	2	1	0
Bit Name	HSGRANT	FA6	FA5	FA4	FA3	FA2	FA1	FA0
Read/Write	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

Bit 7: HSGRANT, Set to 1 if the *SX2* enumerated at high speed. Set to 0 if the *SX2* enumerated at full speed.

Bit[6..0]: Address set by the host.

9.14 INTENABLE Register 0x2E

This register is used to enable/disable the various interrupt sources, and by default all interrupts are enabled.

INTENABLE							0x2E
Bit #	7	6	5	4	3	2	1	0
Bit Name	SETUP	EP0 BUF	FLAGS	1	1	ENUM OK	BUS ACTIVITY	READY
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	1	1	1	1	1	1

9.14.1 SETUP Bit 7

Setting this bit to a 1 enables an interrupt when a setup packet is received from the USB host.

9.14.2 EP0BUF Bit 6

Setting this bit to a 1 enables an interrupt when the Endpoint 0 buffer becomes available.

9.14.3 FLAGS Bit 5

Setting this bit to a 1 enables an interrupt when an OUT endpoint FIFO's state transitions from empty to not-empty.

9.14.4 ENUMOK Bit 2

Setting this bit to a 1 enables an interrupt when *SX2* enumeration is complete.

9.14.5 BUSACTIVITY Bit 1

Setting this bit to a 1 enables an interrupt when the *SX2* detects an absence or presence of bus activity.

9.14.6 READY Bit 0

Setting this bit to a 1 enables an interrupt when the *SX2* has powered on and performed an internal self-test.

9.15 DESC Register 0x30

This register address is used to write the 500-byte descriptor RAM. The external master writes two bytes (four command data transfers) to this address corresponding to the length of the descriptor or VID/PID/DID data to be written. The external master then consecutively writes that number of bytes into the descriptor RAM in nibble format. For complete details, refer to Enumeration on page 8.

9.16 EP0BUF Register 0x31

This register address is used to access the 64-byte Endpoint 0 buffer. The external master can read or write to this register to complete Endpoint 0 data transfers. For complete details, refer to Endpoint 0 on page 8.

9.17 SETUP Register 0x32

This register address is used to access the 8-byte setup packet received from the USB host. If the external master writes to this register, it can stall Endpoint 0. For complete details, refer to Endpoint 0 on page 8.

9.18 EP0BC Register 0x33

This register address is used to access the byte count of Endpoint 0. For Endpoint 0 OUT transfers, the external master can read this register to get the number of bytes transferred from the USB host. For Endpoint 0 IN transfers, the external master writes the number of bytes in the Endpoint 0 buffer to transfer the bytes to the USB host. For complete details, refer to Endpoint 0 on page 8.

10. Absolute Maximum Ratings

Storage Temperature .... –65°C to +150°C

Ambient Temperature with Power Supplied...... 0°C to +70°C

Supply Voltage to Ground Potential................–0.5V to +4.0V

DC Input Voltage to Any Pin 5.25V

DC Voltage Applied to
Outputs in High-Z State .... –0.5V to V_{CC} + 0.5V

Power Dissipation.. 936 mW

Static Discharge Voltage... > 2000V

11. Operating Conditions

T_A (Ambient Temperature Under Bias)............. 0°C to +70°C

Supply Voltage...+3.0V to +3.6V

Ground Voltage..0V

F_{OSC} (Oscillator or Crystal Frequency) .... 24 MHz
± 100-ppm Parallel Resonant

12. DC Electrical Characteristics

Table 12-1. DC Characteristics

Parameter	Description	Conditions[12]	Min	Typ	Max	Unit
V_{CC}	Supply Voltage		3.0	3.3	3.6	V
V_{IH}	Input High Voltage		2		5.25	V
V_{IL}	Input Low Voltage		–0.5		0.8	V
I_I	Input Leakage Current	0< V_{IN} < V_{CC}			±10	µA
V_{OH}	Output Voltage High	I_{OUT} = 4 mA	2.4			V
V_{OL}	Output Voltage Low	I_{OUT} = –4 mA			0.4	V
I_{OH}	Output Current High				4	mA
I_{OL}	Output Current Low				4	mA
C_{IN}	Input Pin Capacitance	Except D+/D–			10	pF
		D+/D–			15	pF
I_{SUSP}	Suspend Current	Includes 1.5k integrated pull up		250	400	µA
I_{SUSP}	Suspend Current	Excluding 1.5k integrated pull up		30	180	µA
I_{CC}	Supply Current	Connected to USB at high speed		200	260	mA
		Connected to USB at full speed		90	150	mA
T_{RESET}	RESET Time after valid power	V_{CC} min = 3.0V	1.91			mS

13. AC Electrical Characteristics

13.1 USB Transceiver

USB 2.0-certified compliant in full and high speed.

Note

[12] Specific conditions for I_{CC} measurements: HS typical 3.3V, 25°C, 48 MHz; FS typical 3.3V, 25°C, 48 MHz.

13.2 Command Interface

Figure 13-1. Command Synchronous Read Timing Diagram[13]

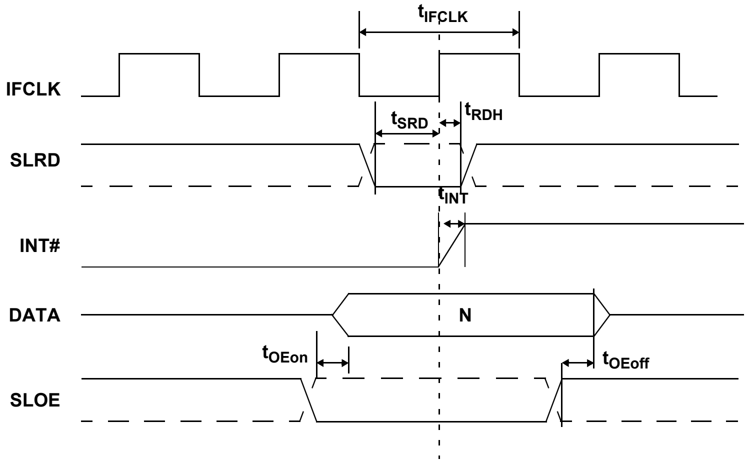


Table 13-1. Command Synchronous Read Parameters with Internally Sourced IFCLK

Parameter	Description	Min	Max	Unit
t_{IFCLK}	IFCLK period	20.83		ns
t_{SRD}	SLRD to Clock Setup Time	18.7		ns
t_{RDH}	Clock to SLRD Hold Time	0		ns
t_{OEon}	SLOE Turn on to FIFO Data Valid		10.5	ns
t_{OEoff}	SLOE Turn off to FIFO Data Hold		10.5	ns
t_{INT}	Clock to INT# Output Propagation Delay		9.5	ns

Table 13-2. Command Synchronous Read with Externally Sourced IFCLK[14]

Parameter	Description	Min	Max	Unit
t_{IFCLK}	IFCLK Period	20	200	ns
t_{SRD}	SLRD to Clock Setup Time	12.7		ns
t_{RDH}	Clock to SLRD Hold Time	3.7		ns
t_{OEon}	SLOE Turn on to FIFO Data Valid		10.5	ns
t_{OEoff}	SLOE Turn off to FIFO Data Hold		10.5	ns
t_{INT}	Clock to INT# Output Propagation Delay		13.5	ns

Notes

[13] Dashed lines denote signals with programmable polarity.

[14] Externally sourced IFCLK must not exceed 50 MHz.