ENT-AN1015

Application Note

Using the SparX-III Serial GPIO/LED Controller

Microsemi
Power Matters.™

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

VPPD-03049

Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

Revision 1.0 was the first publication of this document.

Contents

Figures

Tables

2 Using the SparX-III Serial GPIO/LED Controller

The Microsemi SparX-III family of devices features a power serial GPIO controller. By using a serial interface, the serial GPIO controller significantly extends the number of available GPIOs with a minimum number of additional pins on the device. The serial GPIO controller can be used for serial LEDs, status and control of SFP modules, or provide additional customized inputs and outputs.

This application note describes the use of the serial GPIO controller along with examples of how to implement the serial LED feature as well as control and monitor SFP modules.

The name “SparX-III” is used throughout the document to represent the Microsemi Ethernet switch family, which includes VSC7420, VSC7421, VSC7422, VSC7424, VSC7425, VSC7426, and VSC7427.

2.1 SparX-III Serial GPIO

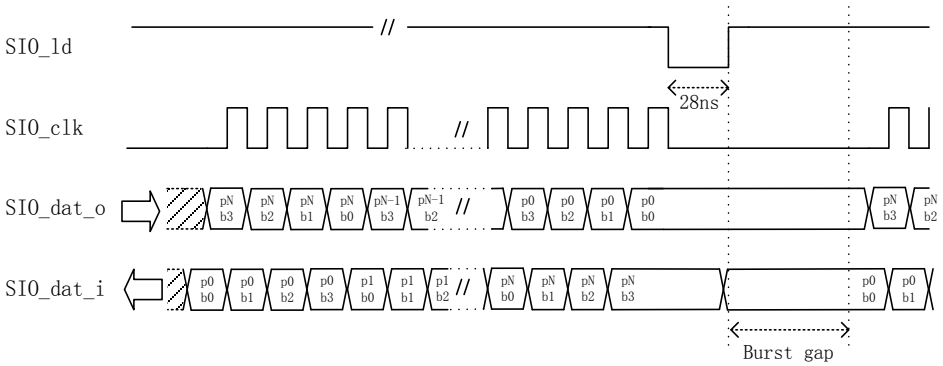
The serial GPIO controller uses a 4-pin serial interface to significantly extend the number of available general purpose I/O pins. The serial GPIO controller has both serial input and serial output functions.

Figure 1 shows the input/output timing of the serial GPIO controller. Serial data is output on the SG_DO pin which is clocked by the SG_CLK pin in bursts. After each burst, there is an assertion of the SG_LD signal. Serial data can be output as a single burst or as continuous bursts separated by a configurable burst gap. The burst gap is configured to between 0 and 33 ms in steps of approximately 1 ms. At the same time, as shifting out serial data on SG_DO, the serial GPIO controller also samples the SG_DI input. The values sampled on SG_DI are made available to software, or forwarded to SparX-III port modules if LOS operation is configured.

When the controller is used for serial LEDs, each single bit in the SG_DO stream corresponds to a different LED. A single burst of data is a frame of data for all the LEDs at a certain moment. External serial to parallel shift registers are needed to convert the serial data to parallel signals for driving LEDs. External parallel to serial shift registers should be used for serial input. SG_LD latches parallel input values to the shift registers while SG_CLK shifts them to the SparX-III device bit by bit.

SG_LD can be used to assure that outputs are stable when serial data is being shifted through the shift registers. This can be done by using the SG_LD signal to load the serial data onto the parallel output pins after the burst has completed for load capable shift registers. If the serial GPIO controller is only used for serial LED output, then SG_LD is optional because it is usually acceptable for outputs to toggle when serial data is updated (for instance, shifted through the chain). When the shift frequency is fast enough, the shifts through the chain are not perceptible to the human eye. It is very easy to make the shift frequency fast enough because the clock frequency of SG_CLK is programmable.

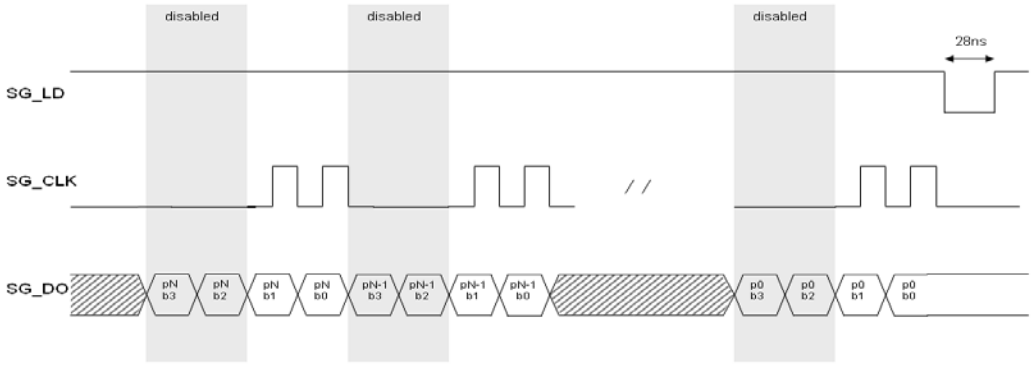
Figure 1 Serial GPIO Timing



The maximum length of a burst is 128 bits long. By default, it is organized into 32 ports each with 4 bits per port. However, each port can be enabled or disabled individually. The port width (number of bits per port) is also centrally configurable, so all enabled ports will have the same width. When a port and/or a bit is disabled, the corresponding SG_CLK is omitted.

The following illustration shows the timing of the serial GPIO output stream when port width is set to 2.

Figure 2 Serial GPIO Output Timing with Disabled Bits



2.2 Using Serial GPIO for Serial LED

Ethernet switches generally use LEDs to indicate link status for all their ports. Normally, more than one LED per port is used because there is a lot of information to be shown (link status, speed, duplex, activity, and so on). Thus, one can imagine there are a lot of LEDs on the front panel of a 24 port Ethernet switch. Traditional designs use a parallel LED scheme which means there is a dedicated pin for each LED provided from Ethernet chipset. The large number of LED pins not only increases the cost of the chipset, but may also increase the cost of the printed circuit board (PCB), as more layers may be required to route all the LED traces to the front panel. If the LEDs are placed on a separate LED board, such that all the parallel LED signals must be bridged over from the main board containing the Ethernet switch to the LED board through ribbon wires, it may reduce the reliability of the system.

To solve the problem, a serial LED scheme has recently been adopted by customers. Serial LEDs use only two pins, a serial data line, and a serial clock line to send out a serial stream of LED status bits. This is then converted to parallel signals externally through a chain of serial to parallel shift registers. The shift registers can be placed at the appropriate locations of the board to ease the layout of the board. The number of wires between the main board and LED board are reduced significantly for use of a LED board.

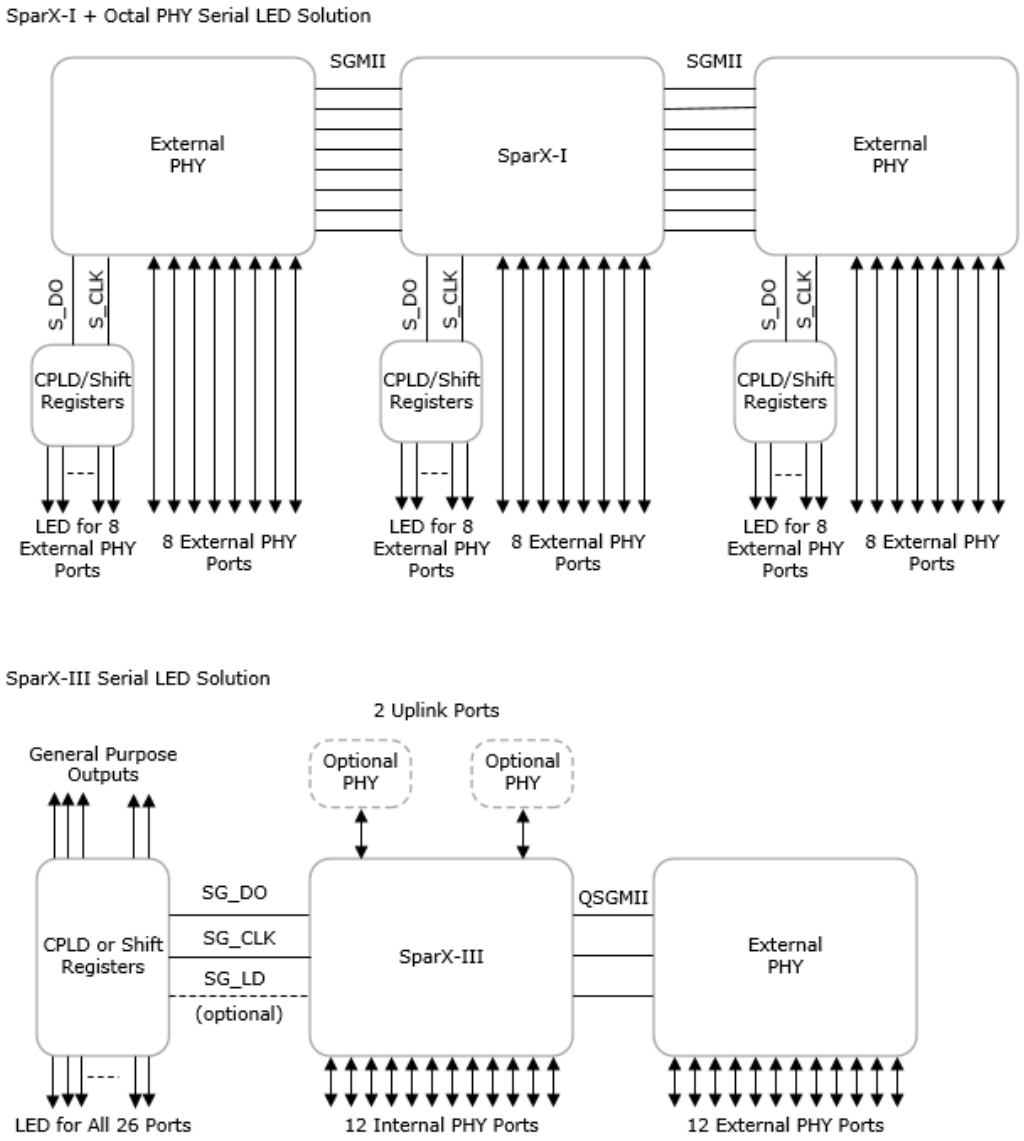
2.2.1 SparX-III Serial LED Benefits

Serial LED is not a new feature for Microsemi Ethernet products. Serial LED had already been introduced years ago, dating back to SparX-I and most Microsemi Ethernet PHYs. However, by using the serial GPIO controller, SparX-III does improve serial LED support in several aspects and offers the following benefits for customers:

- For the old generation SparX-I, and all Microsemi PHY products, the serial LED interface only supported the internal PHY ports (ports 8 thru 15 for VSC7390/89/91). Thus for a 24 port switch application consisting of a 24-port switch with 8 integrated PHYs and two additional 8-port PHY devices, there would be three separate serial LED chains, one from each switch/PHY device. There is a downside when SGMII/SerDes interfaces on SparX-I are used for driving SFP modules directly without a PHY, no serial LEDs can be supported for these ports and there are limited parallel GPIO pins on SparX-I available for that purpose. In contrast, SparX-III supports up to 128 serial outputs from the serial GPIO controller which can be used for serial LEDs for both internal PHY ports and external PHY ports. In addition, customized LEDs such as a temperature alarm, fan speed indication, software alive indications, etc., can all be supported by the 2-pin serial LED interface offered from the serial GPIO controller.
- For SparX-I switches and all Microsemi PHYs, the bit number and order of the serial LED stream is fixed. They support 8 ports with 12 LED status bits per port resulting in a 96 bit serial LED stream for SparX-I and Microsemi 8 port PHYs. That means no matter how many ports are enabled and how many LEDs per port are needed the hardware must have 96 bits of shift registers for each serial LED chain. This is obviously not cost optimized. With SparX-III, the number of ports enabled and the number of bits per port in the serial LED bit stream is programmable. So for a 24 port switch application with 2 LEDs per port there are only 48 (24 times 2) bit shift registers needed. Contrast that with the SparX-I solution which required 288 (96 times 3) bit shift registers.

The following illustration shows the serial LED comparison between SparX-I and SparX-III for a 24/26 port switch.

Figure 3 SparX-III vs. SparX-I on Serial LEDs for a 24/26 Port Switch



2.2.2 SparX-III Serial Output/LED Example

Serial LEDs only make use of serial outputs from the serial GPIO controller. If serial LEDs use less than the total 128 bits, the unassigned bits can be used for other general purpose outputs such as output signals for controlling TxDisable signals for SFP modules or driving a 7-segment display for showing the die temperature of the chipset.

The external circuits for the SparX-III serial LED implementation are as simple as a single chain of serial to parallel shift registers. The number of shift registers must equal the number of enabled ports times the port width.

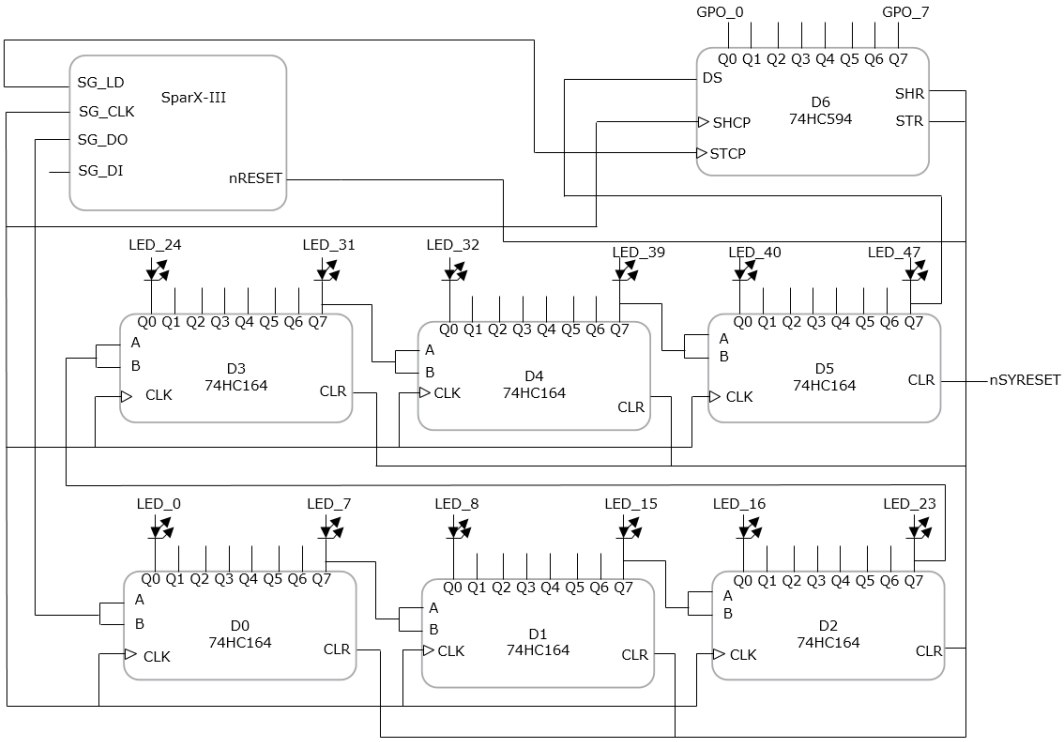
There are different 74 series serial to parallel shift registers with different features to choose from to meet customers' needs. A common requirement is that the shift register should shift in data on the rising edge of the clock. The following lists commonly used serial to parallel shift registers.

- 74164—an 8-bit serial to parallel shift register without load capability and tri-state output. It can be used for serial LED.
- 74594—an 8-bit serial to parallel shift register with load capability. It uses a separate load signal to latch shift register values to the storage register. The storage register holds the values for the parallel output pins. The 74594 can be used for outputs which are not tolerant of output toggling during shifting (for instance, TxDisable output for SFP modules) so SG_LD can be used to load data after a complete burst. The 74594 has independent shift and storage register clear inputs. These clear inputs can be connected to the system reset signal to make sure the parallel outputs are in a known state after reset. The 74594 does not support tri-state outputs as the outputs are always enabled.
- 74595—an 8-bit serial to parallel shift register with load capability and tri-state outputs. The 74595 also uses a separate load signal to latch shift register values to the storage register similar to the 74594. The difference is that tri-state outputs are controlled by an output enable (/OE) signal. When the 74595 is used for serial LEDs, it is possible to pulse gate the parallel LED outputs by using a pulse width modulation (PWM) control signal. There is a PWM output on SparX-III with programmable clock frequency and duty cycle. The signal is initially for the purpose of controlling fan speed. If the PWM output is not used for controlling fan speed, the user may connect it to the /OE input of the 74595 to adjust the brightness of the LEDs. By changing the duty cycle of the signal, the LED brightness will change. Please note that the clear input for the 74595 only clears the shift register, but not the storage register. So the storage register is undefined until the first load pulse is applied, and the shift register values are loaded to the storage register.

The following illustration shows a system with 48 serial LEDs and 8 general purpose outputs. Six 74HC164 (D0 through D5) are used for the 48 LEDs, and a 74HC594 (D6) is used for the 8 general purpose outputs.

28 serial GPIO ports are enabled, and the port width is set to 2. Serial GPIO port0 through port23 are used for serial LEDs (port0 bit0 for LED_0, port0 bit1 for LED_1, port1 bit0 for LED_2, port1 bit1 for LED3, …, port 23 bit 0 for LED_46, and port 23 bit1 for LED_47). Serial GPIO port24 through port27 are used for general purpose outputs (port 24 bit0 for GPO_0, port_24 bit1 for GPO_1, …, port 27 bit0 for GPO_6, and port27 bit1 for GPO_7).

Figure 4 48 LEDs plus 8 Serial Outputs

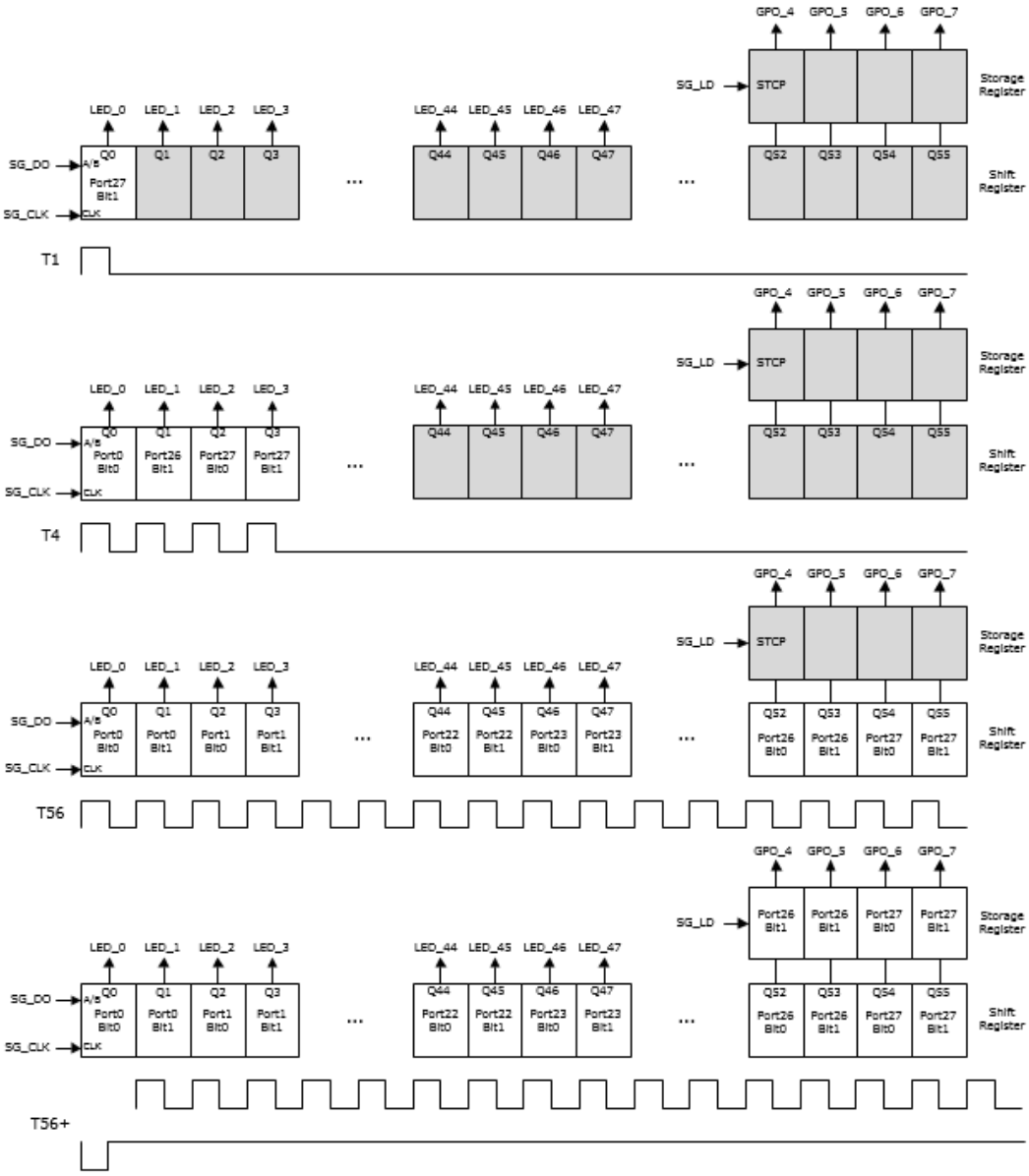


The following illustration shows a burst of serial output data being shifted through the shift register chain. The bit order within the serial stream is in default order, the highest bit (port27 bit1) is shifted out at the first clock, followed by port27 bit0, port26 bit1, …, port0 bit1, and port0 bit0. The register bits in grey hold values from the last burst of data, or default values following reset.

T1 represents the time when only one clock has been sent out. Port27 bit1 is sent out on SG_DO along with the first clock. It will be latched into LED_0, but the destination of port27 bit1 is GPO_7 (the last bit in the shift register chain). It is only at T56 that port27 bit1, together with all other bits, have finally reached their destination through the whole chain. After the assertion of SG_LD, port27 bit1 can then be latched from the shift register to the storage register and output on pin GPO_7. All shift registers hold their values during the burst gap until the next burst of clock and data.

LED_0 thru LED47 are not stable from T1 to T56. However, the human eye cannot detect the toggling if the serial clock is fast enough, and the burst gap is long enough.

Figure 5 Serial Output Bits Shifting Through the Serial Register Chain



2.2.3 Configuring for Serial Output/LED Timing

Several timing parameters must be configured to customize the serial GPIO controller to meet the customer's requirements and match the customer's hardware. The parameters include the number of ports (the enabled/disabled state for each of the 32 ports), port width, serial clock frequency, burst gap, signal polarity, and bit ordering. The relevant registers for configuring the timing of the serial GPIO output are as follows:

- SIO_PORT_CONFIG
- SIO_PORT_ENABLE
- SIO_CONFIG
- SIO_CLOCK

Please refer to the SparX-III datasheet for detailed descriptions of each register. Only the serial LED timing-related register fields and their usages are as follows:

- Clock frequency of the serial GPIO output clock (signal SG_CLK)—SG_CLK is used for both serial input and output. The clock frequency of that signal is configured through register SIO_CLOCK::SIO_CLK_FREQ (bits 11:0). The resulting SG_CLK frequency is the system clock divided by the register value. Given that the system clock of SparX-III is 250 MHz, and if SIO_CLK_FREQ is set to 0xA, the output frequency will be 25 MHz. A faster SIO_CLK makes LED toggling along the shift register chain less noticeable. The customer must ensure that the configured SG_CLK frequency is within the specification of the selected shift register devices. Setting SIO_CLK_FREQ to 0 disables clock division, resulting in SIO_CLK being 250 MHz. Setting SIO_CLK_FREQ to 1 is illegal.
- Serial GPIO burst gap—the burst gap length is programmable through register SIO_CONFIG::SIO_BURST_GAP (bits 11:7) in approximately 1 ms steps. Setting the register field to 0 results in a 1 ms burst gap and 0x1F results in a 33 ms burst gap. Burst gap can also be disabled by setting register bit SIO_CONFIG::SIO_BURST_GAP_DIS. In that case, frames of serial output data follow one by one without a gap between them. A shorter burst gap results in serial outputs being updated more frequently and thus a better visual effect for a blinking LED, especially when the LED is blinking at a fast frequency (such as, 20 Hz). The cost is a little more power consumption.
- Serial GPIO burst mode—SparX-III serial GPIO supports both single burst and continuous bursts. In single burst mode, a frame of serial data is only sent out once on demand (after register bit SIO_CONFIG::SIO_SINGLE_SHOT is set), which is not very useful for dynamic serial LED applications with blinks. Continuous bursts mode is usually used for serial LED applications, so register SIO_CONFIG::SIO_AUTO_REPEAT should be set to enable that mode.
- Port_number and port width—these parameters decide how many bits will be present in every frame of serial data. Port number is configured through a 32-bit register, SIO_PORT_ENABLE, with each bit corresponding to a port. Ports can be individually enabled/disabled by setting/clearing the corresponding bit in the register. Port width is configured centrally for all ports through register SIO_CONFIG::SIO_PORT_WIDTH (bits 3:2). When set to 0, only bit0 for each enabled port is enabled. When SIO_CONFIG::SIO_PORT_WIDTH is set to 2, all 4 bits (bit3, bit2, bit1, and bit0) are enabled.
- Serial LED output bit order—the default order of the output bit stream is in the order of portN bit3, portN bit2, …, port0 bit1, port0 bit0 (see Figure 1). This can be reversed by setting register SIO_CONFIG::SIO_REVERSE_OUTPUT, and the bit order changes to port0 bit0, port0 bit1, …, portN bit2, portN bit3.
- SG_LD polarity—the polarity of the serial data load signal, SG_LD, can be programmed to be active high or low through register bit SIO_CONFIG::SIO_LD_POLARITY. The purpose is for adapting to different types of load capable shift registers.

2.2.4 Serial Output Modes

Serial output modes are the supported behaviors for each individual LED. Serial output mode is programmed for each serial output bit through a group of registers–SIO_PORT_CONFIG[n]. There are 32 registers in total. SIO_PORT_CONFIG[n] corresponds to a serial output portn. Only bits 11:0 of a SIO_PORT_CONFIG are used for programming serial output mode for 4 bits. For example, SIO_PORT_CONFIG[0]::BIT_SOURCE_3(bits 11:9) are used to configure the serial output mode for port0 bit3, SIO_PORT_CONFIG[0]::BIT_SOURCE_2(bits 8:6) are used to configure the serial output mode for port0 bit2, SIO_PORT_CONFIG[0]::BIT_SOURCE_1(bits 5:3) are used to configure the serial output mode for port0 bit1 and SIO_PORT_CONFIG[0]::BIT_SOURCE_1(bits 2:0) are used to configure the serial output mode for port0 bit0.

The three bits for each serial output bit are coded for the following 8 serial output modes:

- Force “0”
- Force “1”
- Blinking mode 0
- Blinking mode 1
- Link activity blinking mode 0
- Link activity blinking mode 1
- Link activity blinking mode 0 inversed polarity
- Link activity blinking mode 1 inversed polarity

Mode 1 and mode 2 are forced modes. They are used to assign a fixed value to the corresponding output bit in the serial output stream. The fixed value when converted to parallel by the external shift registers will result in a fixed voltage level for turning on/off the connected LED according to the polarity of the external LED. The forced modes are used to display static events like link status (linkup or linkdown), speed (10M, 100M, 1000M), and duplex of an Ethernet port.

Note: The hardware does not collect link status, speed, and duplex information from any Ethernet port and map the serial output bits automatically. Software must get that information and program the output mode field. For example, when serial output port 0 bit1 is intended to be used as link 1000M indication for Ethernet port module0, software must check link status for Ethernet port module0 periodically. Only when software detects Ethernet port module0 establishes a link in 1000M speed, it will then update SIO_PORT_CONFIG[0]::BIT_SOURCE_1 to 0 to turn on the LED, assuming the external LED circuit is active low.

Mode 3 and mode 4 are two blink modes. Blinking modes are used to make the serial output bits to blink at a fixed rate. SparX-III serial GPIO controller features two blink modes that can be set independently. The following table shows the supported blinking frequencies. Two blink modes are able to convey different information and can be programmed at different blinking frequencies so that some LEDs can blink faster than the others, which is useful to give different levels of alarm by using different blinking frequency, for example.

Table 1 Blinking Modes

Mode	Description
Blink mode 0	0: 20 Hz blink frequency 1: 10 Hz blink frequency 2: 5 Hz blink frequency 3: 2.5 Hz blink frequency
Blink mode 1	0: 20 Hz blink frequency 1: 10 Hz blink frequency 2: 5 Hz blink frequency 3: Burst toggle

Blinking frequencies for blink mode 0 and 1 can be programmed through register SIO_CONFIG::SIO_BMODE_0 (bits 19:18) and SIO_CONFIG::SIO_BMODE_1 (bits 21:20).

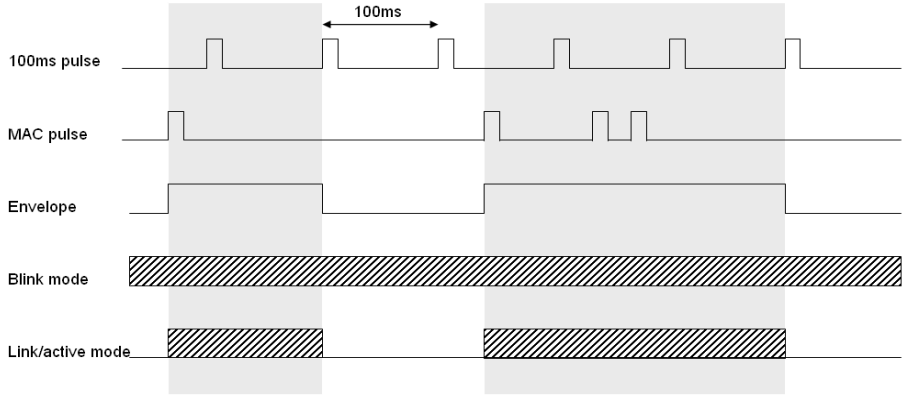
If more than one Ethernet switch/PHY are used in a system, the LED blinking pattern from between these devices can go out of sync, and this can sometimes be annoying. SparX-III offers a solution to the problem by providing a blink counter reset bit (register SIO_CONFIG::SIO_BLINK_RESET) to synchronize the blink patterns between different devices. This is more effective when the different devices run from the same clock source; otherwise the devices will slowly drift apart, thus a blinking counter reset needs to be done periodically.

Modes 5 through 8 are link active modes. For serial output mode 1 through 4, the serial output ports have no direct relation to the Ethernet port modules. Bits in serial output port 0 can be used for LEDs for any port module as they are basically controlled by software. However, it is not true for the link active modes. Link active modes are used to make the output blink when there is activity on the corresponding port module (both Tx or Rx). The mapping between serial output port and Ethernet port module is 1:1 (static mapping). Serial output port 0 is connected to port module 0, serial output 1 is connected to port module 1, and so on.

The link active modes use an envelope signal to gate the selected blinking pattern (blink mode 0 or blink mode 1). When the envelope signal is asserted, the output will blink. When the envelope signal is de-asserted, the output will be fixed at 0 (blink mode 5 and 6) or 1 (blink mode 7 and 8). Blink modes 5 and 6 can be used for a Link/activity LED when the hardware LED is active low. Blink modes 7 and 8 can be used for Link/activity LED when the hardware LED is active high.

To ensure that even a single packet makes a visual blink, an activity pulse from the port module will be extended to a minimum of 100 ms. If another packet is sent while the envelope signal is asserted, it will be extended by another 100 ms, as shown in the following illustration.

Figure 6 Link Active Mode Timing



2.2.5 Software Example for Serial Output/LED

A code example is given for supporting a 24 port switch. With two LEDs per port are used in the example. LED0 (corresponds to bit0 in the serial stream) is 10/100link/activity and LED1 (corresponds to bit1 in the serial stream) is 1000link/activity. The blink frequency for LED0 and LED1 are at 5 Hz. Additionally, a software alive LED and a die temperature alarm LED are also presented. The software alive LED blinks at 0.5 Hz to indicate that the software is running while the die temperature alarm LED has 3 states:

- UnAlarm (Alarm LED off)—temperature below alarm_threshold_0
- Alarm_level_1 (LED blink at 2.5 Hz)—temperature between alarm_threshold_0 and alarm_threshold_1
- Alarm_level_2 (LED blink at 5 Hz)—temperature above alarm_threshold_1

In this example, assume all LEDs have been designed as active low by hardware. The following code initializes the serial GPIO and sets the serial output mode for all the LEDs dynamically:

```
Write(SIO_CLOCK::SIO_CLK_FREQ, 0x64);    /* Set SG_CLK to be 2.5M,250M/100 = 2.5M */
Write(SIO_CONFIG::SIO_BURST_GAP, 0x1f);  /* set length of burst gap to be 33ms    */
Write(SIO_PORT_ENABLE,0x1ffff);          /* enable 25 serial output ports, the 25th port is for
                                                 software alive and alarm LED    */
```

```
Write(SIO_CONFIG::SIO_PORT_WIDTH, 1);    /* set port width to 2 */
Write(SIO_CONFIG::SIO_BMODE_1, 2);       /* set blink mode 1 frequency at 5Hz, used for
                          10/100link/activity, 1000Link/activity and temperature Alarm_level_2 */
Write(SIO_CONFIG::SIO_BMODE_0, 3);       /* set blinking mode 0 frequency at 2.5Hz used for
                                                    temperature Alarm_level_1   */
/* Turn off all LED, assume the hardware design for all LEDs are active low */
for (n = 0, n <25, n++) {
       Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_0, 1);    /*  force off for LED0  */
       Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_1, 1);    /*  force off for  LED1  */
}
Write(SIO_CONFIG::SIO_AUTO_REPEAT, 1);           /* enable continuous bursts mode   */

/* Checking port status and setting link/activity LED mode every 100ms */
While(timer_100ms expired){
for (n = 0, n <24, n++) {
  if(link status for port[n] changed) {
    if(port[n] linkup && port[n] speed == 10M/100M){
     Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_0, 5);  /* Activate LED0 for 10/100Link/activity */
     Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_1, 1);  /* turn off LED1 */
    }
    else if(port[n] linkup && port[n] speed == 1000M){
      Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_1, 5);  /* Activate LED1 for 1000Link/activity */
      Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_0, 1);  /* turn off LED0 */
    }
    else if(port[n] linkdown)){
    Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_0, 1);  /* turn off LED0 */
    Write(SIO_PORT_CONFIG[n]::BIT_SOURCE_1, 1);  /* turn off LED1 */
    }
  else{ /* No link status change, no action  */ }
  }
 }
}
/* Invert alive LED and update temperature Alarm status once per second  */
While(timer_1s expired){
   Write(SIO_PORT_CONFIG[24]::BIT_SOURCE_0,(SIO_PORT_CONFIG[24]::BIT_SOURCE_0 + 1)&1);
                                            /* invert alive LED   */
   If(T<alarm_threshold_0){
      Write(SIO_PORT_CONFIG[24]::BIT_SOURCE_1, 1);      /* turn off temperature alarm LED  */
   }
   else if(alarm_threshold_0<=T<alarm_threshold_1){
      Write(SIO_PORT_CONFIG[24]::BIT_SOURCE_1, 2); /* set alarm LED in blinking mode 0 -- 2.5Hz */
   }
   else{
      Write(SIO_PORT_CONFIG[24]::BIT_SOURCE_1, 3); /* set alarm LED in blinking mode 1 -- 5Hz */
   }
 }
```

2.3 Using Serial GPIO Input

As previously described, the SparX-III serial GPIO controller has both output and input function blocks. Serial input uses a serial interface to extend the number of general purpose input pins. The extended input pins can be used to monitor status from SFP modules. There are three status outputs from each SFP module, Tx_fault, Module_detect, and LOS. The required number of general purpose

input pins could be large if customers support a lot of SFP modules in their design. Besides SFP module status monitoring, the serial GPIO input can also be used for push buttons, DIP switch configurations, and more. Additional inputs make the switch more configurable and intelligent.

2.3.1 Common Settings for Serial Input and Output

The serial input block works simultaneously with the output block. When serial output data is being shifted out on SG_DO, serial input data is at the same time being shifted into the switch through SG_DI. The serial input data will be stored in register S_IN0 thru S_IN3 for software to read.

Serial input and output of the serial GPIO controller share several common settings:

- Same configurations for serial GPIO clock frequency and burst gap. Please note SG_LD assertion is followed immediately after a burst of serial clock (SG_CLK). It is not until the next burst of serial clock that the latched serial inputs can be shifted from the external shift registers into the switch. Longer burst gaps result in longer input latency. The maximum input latency could be 33ms (maximum burst gap). Although it is possible to disable burst gap by setting register bit SIO_CONFIG::SIO_BURST_GAP_DIS to reduce input latency, but that would cause problems for the serial LEDs (outputs) if shift registers without load capability are used (e.g., 74164). Burst gap optimization must address requirements of both serial inputs and outputs. Refer to Section 3.3 on configurations for serial GPIO clock and burst gap.
- Same configurations for serial GPIO port number (serial ports enabled/disabled) and port width. The serial port/bit enabling/disabling is effective for both input and output. Refer to Section 3.3 for configurations on port number and port width.
- Serial GPIO burst mode: Once enabled single burst and continuous bursts are effective for both serial input and output. Single burst mode enables the software to read serial input at any time. But due to the input latency described earlier in order to sample the input values by single burst mode, two consecutive single burst commands must be issued. The first burst will result in the input values being latched by the external serial to parallel registers, while the second burst will shift the input values into the switch. Set register bit SIO_CONFIG::SIO_SINGLE_SHOT to trigger a single burst. This register is self cleared.
- Polarity of SG_LD: Both serial input and serial output use the same load signal, SG_LD. For serial output SG_LD is used to latch shift register values to the parallel output pins. For serial input SG_LD is used to load parallel input values from parallel input pins to the shift register. The polarity of SG_LD is configurable through register bit SIO_CONFIG::SIO_LD_POLARITY.

2.3.2 Serial Input Bit Order and Format

As shown in Figure 1, the default order of the input bit stream is in the order of port0 bit0, port0 bit 1, …, portN bit 2, portN bit 3. The lowest bit is shifted into the switch first. This can be reversed by setting SIO_CONFIG::SIO_REVERSE_INPUT, after which the bit order changes to portN bit 3, portN bit2, …, port0 bit1, port0, port0 bit0. The bit order for serial input can be reversed independent of the bit order for serial output.

The sampled serial input data is stored in four 32-bit registers, S_IN0 through S_IN3. The following illustration shows the data format of the four registers when only serial GPIO port 0 through port 15 are enabled with a port width of 2 bits. The bits for disabled ports/bits are undefined. In the following illustration, the bits in grey represent the undefined/disabled bits.

Figure 7 Serial Input Data Register

			…						…					…
S_IN3	P31 b3	P30 b3	…	P17 b3	P16 b3	P15 b3	P14 b3	P13 b3	P12 b3	…	P3 b3	P2 b3	P1 b3	P0 b3
S_IN2	P31 b2	P30 b2	…	P17 b2	P16 b2	P15 b2	P14 b2	P13 b2	P12 b2	…	P3 b2	P2 b2	P1 b2	P0 b2
S_IN1	P31 b1	P30 b1	…	P17 b1	P16 b1	P15 b1	P14 b1	P12 b1	P12 b1	…	P3 b1	P2 b1	P1 b1	P0 b1
S_IN0	P31 b0	P30 b0	…	P17 b0	P16 b0	P15 b0	P14 b0	P13 b0	P12 b0	…	P3 b0	P2 b0	P1 b0	P0 b0

2.3.3 Serial Input and Interrupts

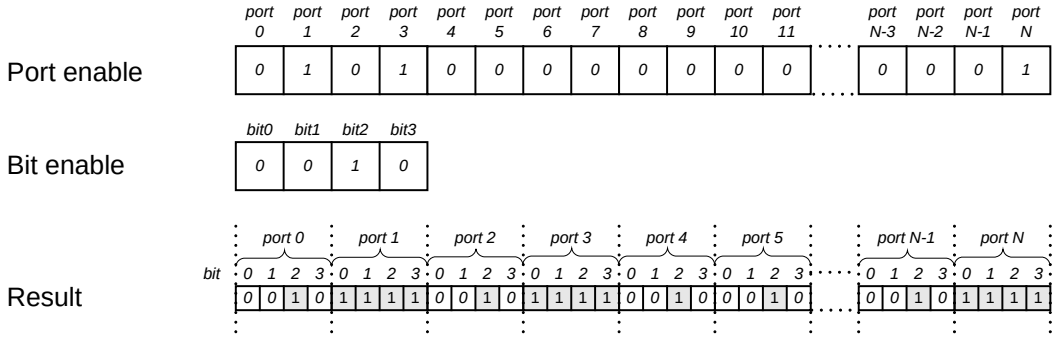
The serial GPIO input is able to generate interrupts based on the serial input data values. All interrupts are level-sensitive and can be enabled per port and per bit position.

The 32-bit register SIO_PORT_INT_ENA enables interrupts per serial input port. When a bit is set in this register, interrupts for all four input bits of the corresponding serial port are enabled.

The 4-bit register SIO_CONFIG::SIO_INT_ENA enables serial input interrupts per bit position. The lowest bit in SIO_CONFIG::SIO_INT_ENA represents bit0 of all the serial GPIO ports, while the highest bit in SIO_CONFIG::SIO_INT_ENA represents bit3 of all ports. Setting a bit in SIO_CONFIG::SIO_INT_ENA will enable interrupts for the corresponding input bit for all ports. The result of the per port and per bit position interrupts are shown in the following illustration.

Interrupt polarity is configured for each serial input bit in SIO_INT_POL_0 thru SIO_INT_POL_3. Bit ordering is the same as shown in the illustration. Setting a bit configures an interrupt at logic value “0” for the serial input bit. Clearing a bit configures it to an interrupt at logic value “1”.

Figure 8 Serial Input Interrupt Masks



The serial GPIO controller has one interrupt output connected to the main interrupt controller that gets asserted when one or more interrupts are active. To determine which input bit is causing the interrupt, the CPU must read the “sticky-bit” interrupt registers SIO_INT_REG_0 through SIO_INT_REG_3. The registers have one bit per serial input bit and can only be cleared by software. A bit is cleared by writing a “1” to the bit position. The interrupt output will remain high until all interrupts in SIO_INT_REG_x are cleared.

2.3.4 Serial GPIO Input for LOS

The serial GPIO controller is able to propagate loss of signal detection inputs directly to the port module’s signal detection input. This is useful when, for example, SFP modules are used in the design. The mapping between serial GPIO ports and port modules is the same as for link activity outputs (port 0 is connected to port module 0, port 1 is connected to port module 1, and so forth).

Only bit 0 of each serial input port can be configured to forward directly to the corresponding port module's loss of signal input. Enabling serial input for LOS can be configured per port or per port module. The following signal detect configuration bits for the port module should be configured:

- PCS1G_SD_CFG::SD_ENA (bit 0) must be set to enable Signal Detect for the PCS of the port module. When this bit is cleared, the PCS assumes an active Signal Detect at all times.
- PCS1G_SD_CFG::SD_SEL (bit 8) must be set to enable serial input bit 0 as the signal detect input to the PCS. Otherwise, the signal detect signal for the PCS is generated internally.
- PCS1G_SD_CFG::SD_POL (bit 4) configures the polarity of the signal detect input. Because LOS from an SFP module is active low, this bit must be cleared. Additionally, the corresponding interrupt polarity registers of the serial input bit in SIO_INT_POL_0 through SIO_INT_POL_3 must also be cleared.

2.3.5 SparX-III Serial Input Examples

External parallel to serial shift registers must be used to sample and store the parallel input values before they are shifted into the switch. The 8-bit shift register 74165 can be used for that purpose. It latches parallel values into the shift register on the rising edge of /PL (typically connected to SG_LD) and shifts data out on the rising edge of CP (typically connected to SG_CLK).

Because the same serial ports/bits must be enabled at the same time for serial input and output, the shift register width for input must theoretically be equal to the shift register width for output. As the number of serial outputs is generally more than the number of serial input required in a design, this does not necessarily mean that the design must contain the same number of physical shift registers for serial input as it does for serial output. Several techniques are described in the following sections to reduce the number of shift registers for serial input.

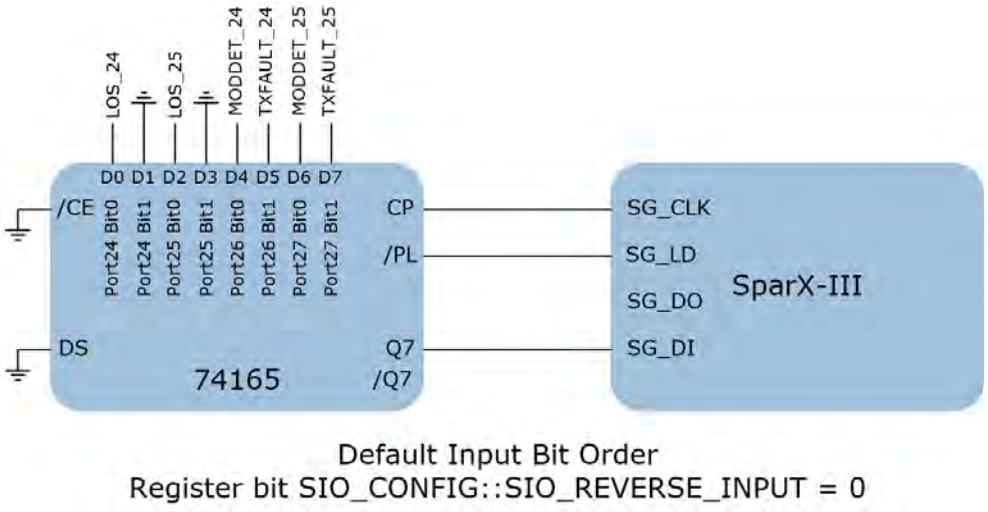
A general recommendation is to use the lowest enabled ports/bits for serial input when serial input is in default bit order (register bit SIO_CONFIG::SIO_REVERSE_INPUT cleared) or use the highest ports/bits for serial input when serial input is in reverse bit order (register bit SIO_CONFIG::SIO_REVERSE_INPUT set). In that case, the shift register width can be limited to the number of actually required serial input bits. The idea is to only have external shift registers for the required serial inputs by arranging the required serial inputs to be shifted into the switch earlier than other "don't care" serial inputs, which do not necessarily have shift registers for them because the values are not used.

As an example, assume we have the same serial output/LED requirements as in SparX-III Serial Output/LED Example such that 28 serial ports with 2-bit port width must be enabled. Now we have the serial input requirement of 6 inputs for two SFP modules on port module 24 and 25. The signals Tx_fault, Module_detect, and LOS from the two SFP modules must be monitored through serial input. Because the two LOS signals must be mapped to serial port 24 bit 0 and serial port 25 bit 0, we have to use the highest ports/bits for the six inputs with reversed serial input bit order.

The following illustration shows how this can be realized by using only one 8-bit parallel to serial shift register.

In this example, the six input signals from the two SFP modules will be correctly input to the switch while all the other serial input bits will all get a value "0" because signal DS is grounded. LOS_24 and LOS_25 must be mapped to serial port 24, bit 0 and port 25 bit 0 correspondingly.

Figure 9 Serial Input, Example 1

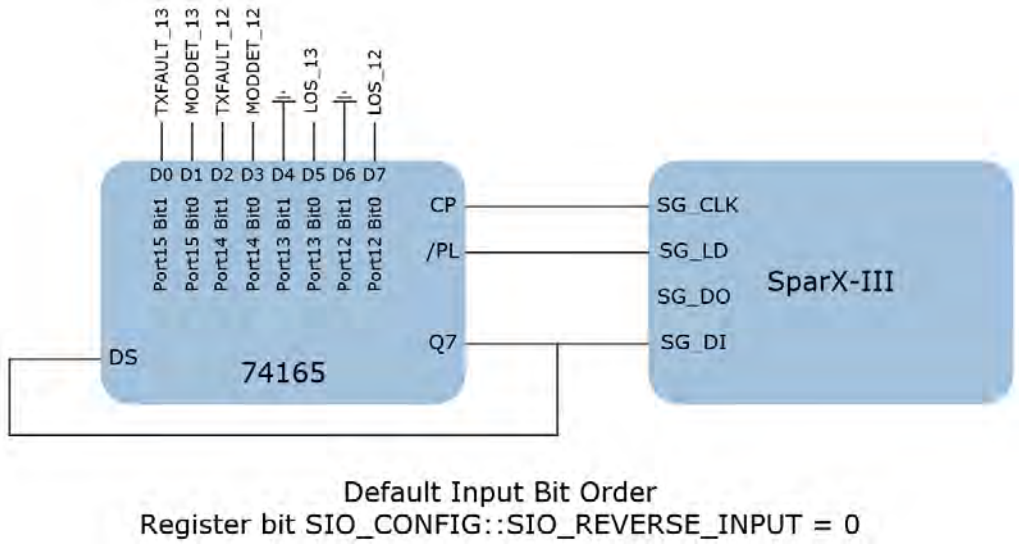


Serial inputs such as Module_detect and Tx_fault from SFP modules or push button inputs are only for software to read (we call them soft inputs) so they can be mapped to any serial port/bit. However, the LOS signal from an SFP module is forwarded directly to the PCS block of the corresponding port module so there is no freedom to place these in any port/bit position at will.

In the preceding example, the two SFP modules are on port 24 and 25 so we can use the highest ports/bits for all the serial input signals. If the two SFP modules must be supported on port module 12 and 13, a technique called shift register loopback can be used.

The following illustration shows an example with the same serial output/LED requirements as the previous example, but the six serial inputs from two SFP modules are from port module 12 and 13. Serial output Q7 of the shift register is looped back to its serial input DS, so that the same 8-bit values will be repeated on the serial input stream and get duplicated in the internal serial input registers. In this way, we have successfully copied the 8 serial inputs into the correct port/bit positions.

Figure 10 Serial Input, Example 2



The following illustration shows the resulting serial input values in the switch after a complete burst. Bits in grey are disabled bits. Bits in dashed boxes are don't care bits.

Figure 11 Input Data with Loopback

S_IN3	P31 b3	P30 b3	P29 b3	P28 b3	P27 b3	P26 b3	P25 b3	P24 b3	P23 b3	P22 b3	P21 b3	P20 b3	P19 b3	P18 b3	P17 b3	P16 b3
	P15 b3	P14 b3	P13 b3	P12 b3	P11 b3	P10 b3	P9 b3	P8 b3	P7 b3	P6 b3	P5 b3	P4 b3	P3 b3	P2 b3	P1 b3	P0 b3
S_IN2	P31 b2	P30 b2	P29 b2	P28 b2	P27 b2	P26 b2	P25 b2	P24 b2	P23 b2	P22 b2	P21 b2	P20 b2	P19 b2	P18 b2	P17 b2	P16 b2
	P15 b2	P14 b2	P13 b2	P12 b2	P11 b2	P10 b2	P9 b2	P8 b2	P7 b2	P6 b2	P5 b2	P4 b2	P3 b2	P2 b2	P1 b2	P0 b2
S_IN1	P31 b1	P30 b1	P29 b1	P28 b1	P27b1 TXFAULT13	P26b1 TXFAULT12	P25b1 0	P24b1 0	P23b1 TXFAULT13	P22b1 TXFAULT12	P21b1 0	P20b1 0	P19b1 TXFAULT13	P18b1 TXFAULT12	P17b1 0	P16b1 0
	P15b1 TXFAULT13	P14b1 TXFAULT12	P13b1 0	P12b1 0	P11b1 TXFAULT13	P10b1 TXFAULT12	P9b1 0	P8b1 0	P7b1 TXFAULT13	P6b1 TXFAULT12	P5b1 0	P4b1 0	P3b1 TXFAULT13	P2b1 TXFAULT12	P1b1 0	P0b1 0
S_IN0	P31 b0	P30 b0	P29 b0	P28 b0	P27b0 MODDET13	P26b0 MODDET12	P25b0 LOS13	P24b0 LOS12	P23b0 MODDET13	P22b0 MODDET12	P21b0 LOS13	P20b0 LOS12	P19b0 MODDET13	P18b0 MODDET12	P17b0 LOS13	P16b0 LOS12
	P15b0 MODDET13	P14b0 MODDET12	P13b0 LOS13	P12b0 LOS12	P11b0 MODDET13	P10b0 MODDET12	P9b0 LOS13	P8b0 LOS12	P7b0 MODDET13	P6b0 MODDET12	P5b0 LOS13	P4b0 LOS12	P3b0 MODDET13	P2b0 MODDET12	P1b0 LOS13	P0b0 LOS12

The last example is a special case where only one serial input needs to be read. No matter if it is a soft input or a LOS signal from a single SFP for any port module, this signal can be connected directly to SG_DI without a shift register. This single input bit will be read repeatedly by the serial GPIO controller, and it will finally reach the correct bit position where it is needed.