香港中文大學
The Chinese University of Hong Kong

CENG3430 Rapid Prototyping of Digital Systems

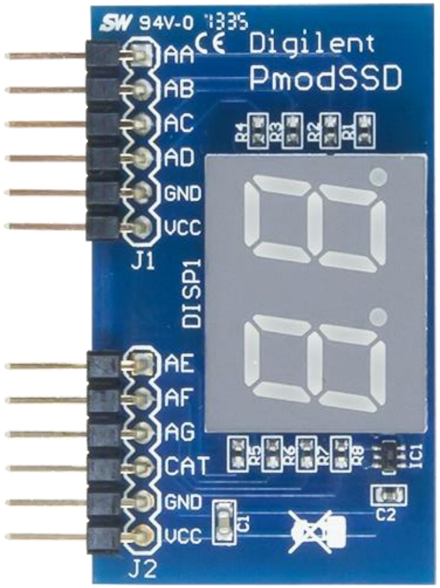
Lecture 06: Driving Peripheral Modules with ZedBoard

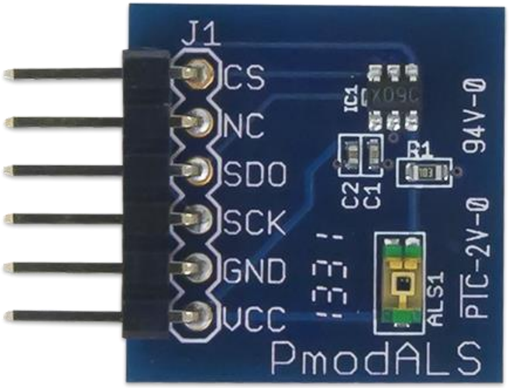
Ming-Chang YANG

mcyang@cse.cuhk.edu.hk

Outline

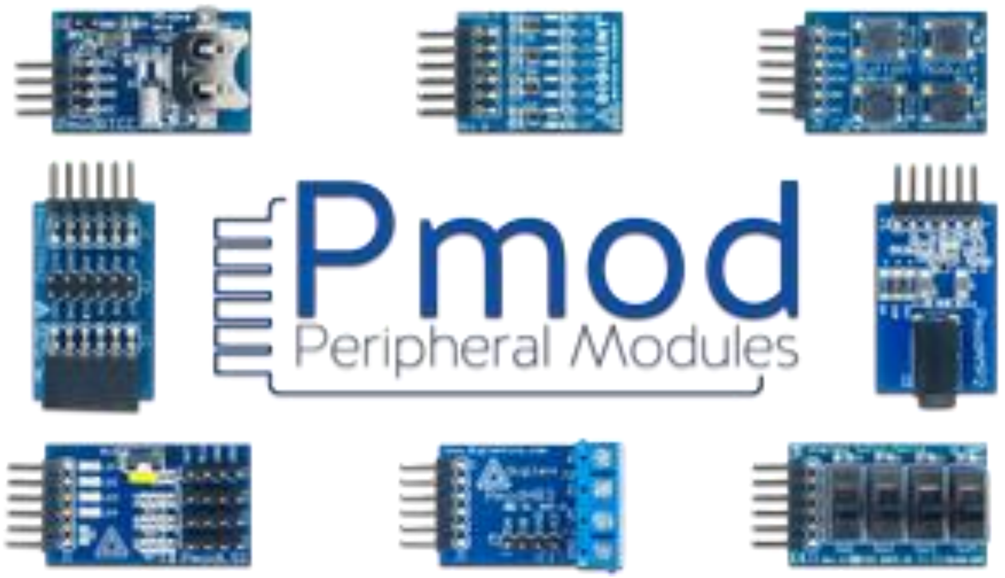
- Digilent Pmod™ Peripheral Modules
  - Communication Protocols
  - Pmod Ports on ZedBoard
- Case Study 1: Pmod SSD (GPIO)
- Case Study 2: Pmod ALS (SPI Protocol)

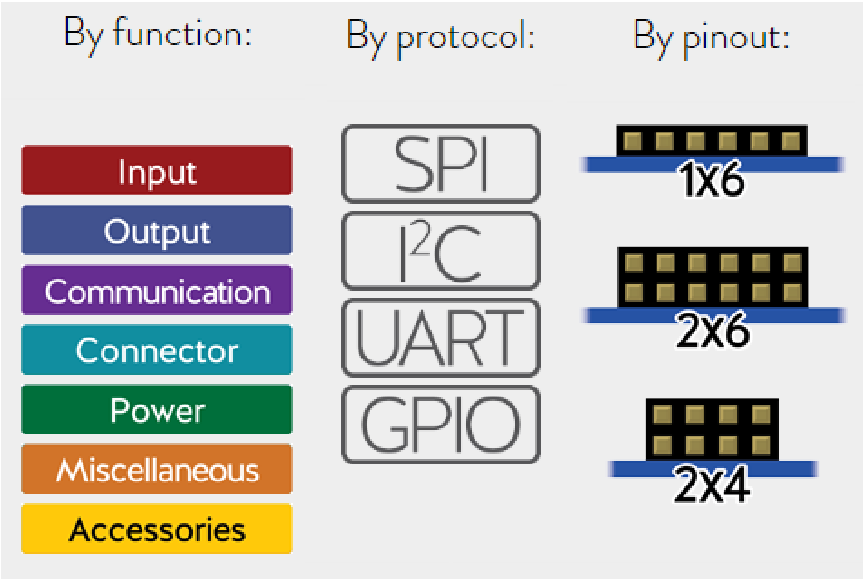




Digilent Pmod™ Peripheral Modules

- **Pmod™ devices** are Digilent's line of small I/O interface boards.
  - They offer an ideal way to extend the capabilities of programmable logic and embedded control boards.
  - They communicate with system boards, via different protocols, using 6, 8, or 12-pin connectors.





https://store.digilentinc.com/pmod-modules-connectors/

Communication Protocols (1/2)

- General Purpose Input/Output (GPIO)
  - GPIO does not follow any set of rules for communication.

※ We can **directly send/receive signals to/from the Pmod** via external I/O pins at any time.

- Serial Peripheral Interface (SPI)
  - SPI is a sync. serial communication scheme that uses four wires: slave select, serial clock, master-out/in-slave-in/out.
  - The host board (i.e., the SPI master) must strictly follow the SPI protocol regarding how to communicate with the Pmod (i.e., the SPI slave) through the four wires.

※ We may implement the SPI master via HDL (**harder**), use the third-party SPI master (**easier**), or use Digilent Pmod IP.

Communication Protocols (2/2)

- Inter-Integrated-Circuit (I²C)
  - I²C is also a sync. serial comm. scheme with only two wires: a serial data line and a serial clock line (drove by master).
  - The host board (i.e., the master) also needs to strictly follow the I²C protocol to interact with the Pmod (i.e., the slave).

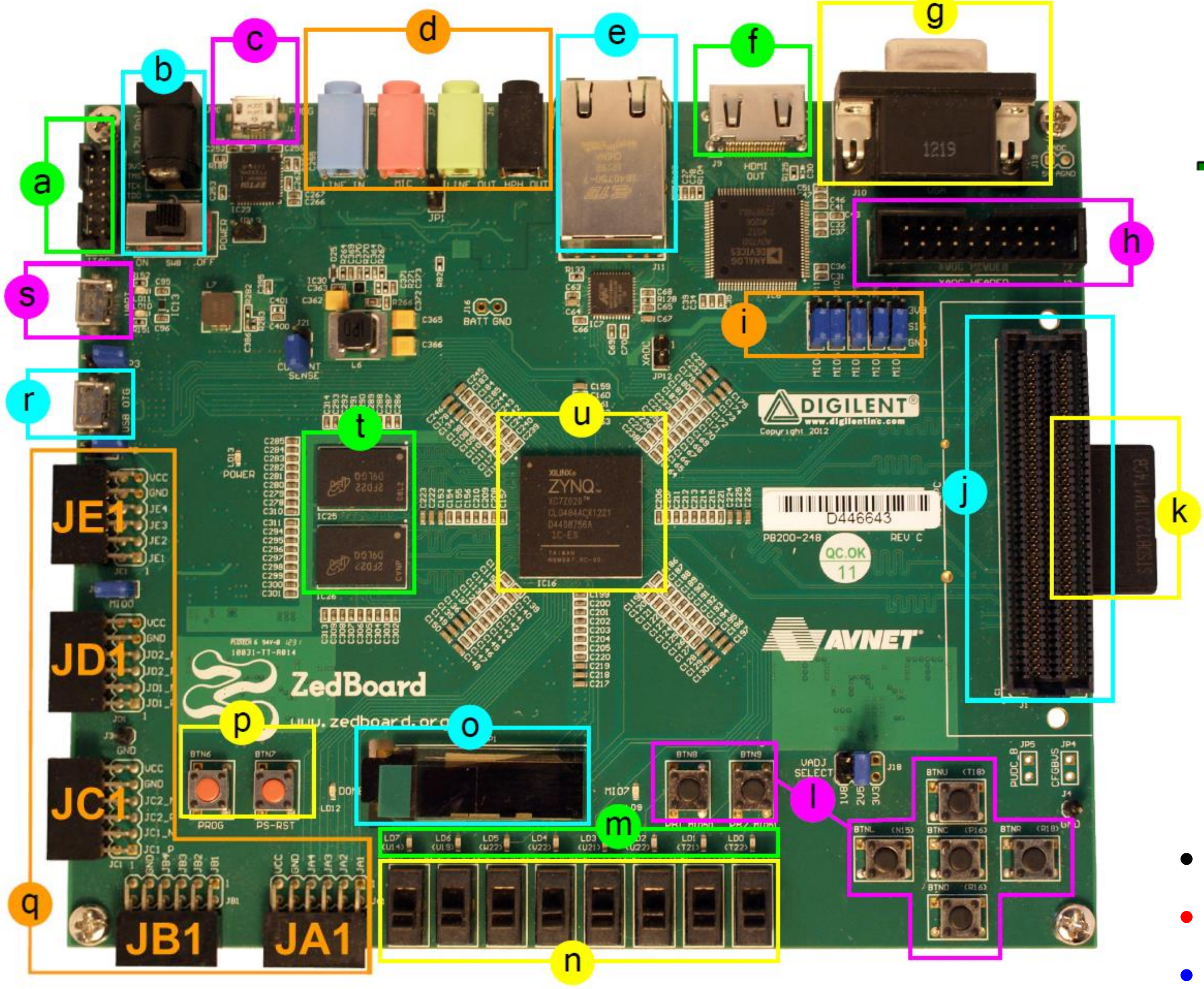
※ We may implement the I²C master via HDL (**harder**), use the third-party I²C master (**easier**), or use Digilent Pmod IP.

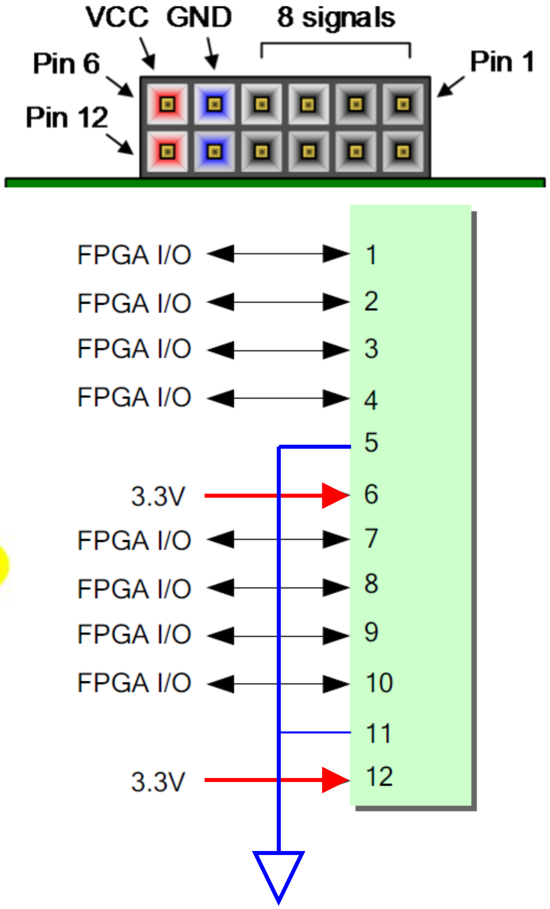
- Universal Asynchronous Receiver/Transmitter (UART)
  - UART is an async. serial comm. scheme that uses one transmit data line and one receive data line.
    - No clock is used but a baud rate (i.e., the number of bits that can be transmitted in a second) must be specified.

※ We may implement UART receiver/transmitter via HDL (**harder**), use the third-party UART transmitter/receiver (**easier**) or Digilent Pmod IP.

Pmod Ports on ZedBoard (1/3)

- ZedBoard has **five** Pmod™ 2x6 connectors (**JA1**~**JE1**).

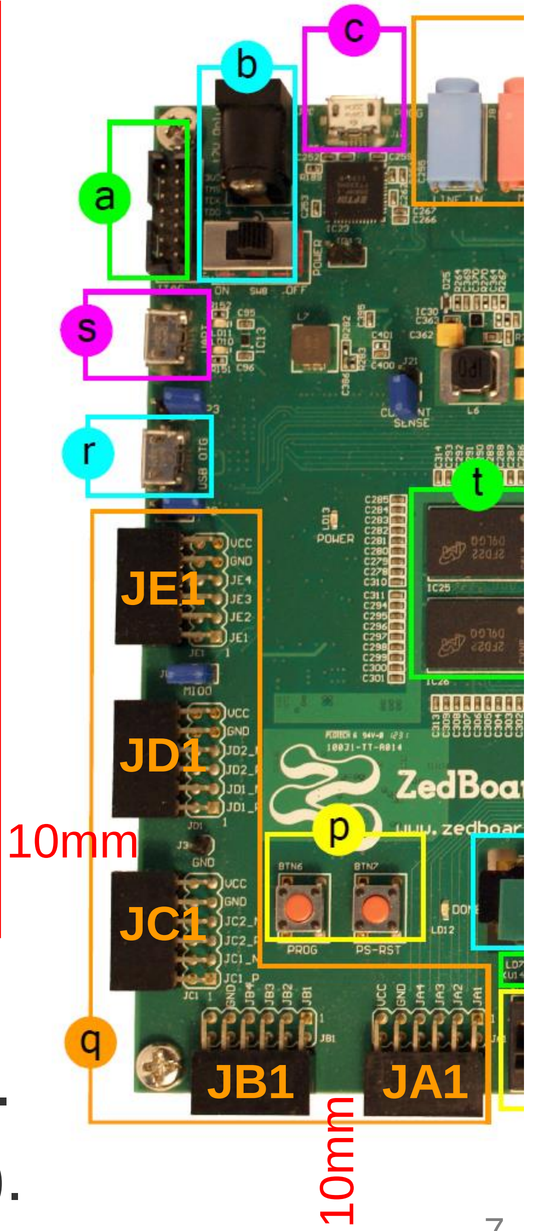




- **Eight** FPGA I/Os
- **Two** 3.3V signals
- **Two** ground signals

Pmod Ports on ZedBoard (2/3)

- **Four** Pmod connectors (**JA1~JD1**) interface to the **PL-side** of the Zynq-7000 AP SoC.
  - **JA1~JD1** connect to Bank 13 (3.3V).
  - **JA1~JD1** are placed in adjacent pairs on the board edge.
    - The clearance between JA1 and JB1 and between JC1 and JD1 are both 10mm.
  - *Note: **JC1** and **JD1** are aligned in a dual configuration and routed differentially to support LVDS running at 525Mbs.*
- **One** Pmod connect (**JE1**) interface to the **PS-side** of the Zynq-7000 AP SoC.
  - On MIO pins [0,9-15] in Bank 0/500 (3.3V).

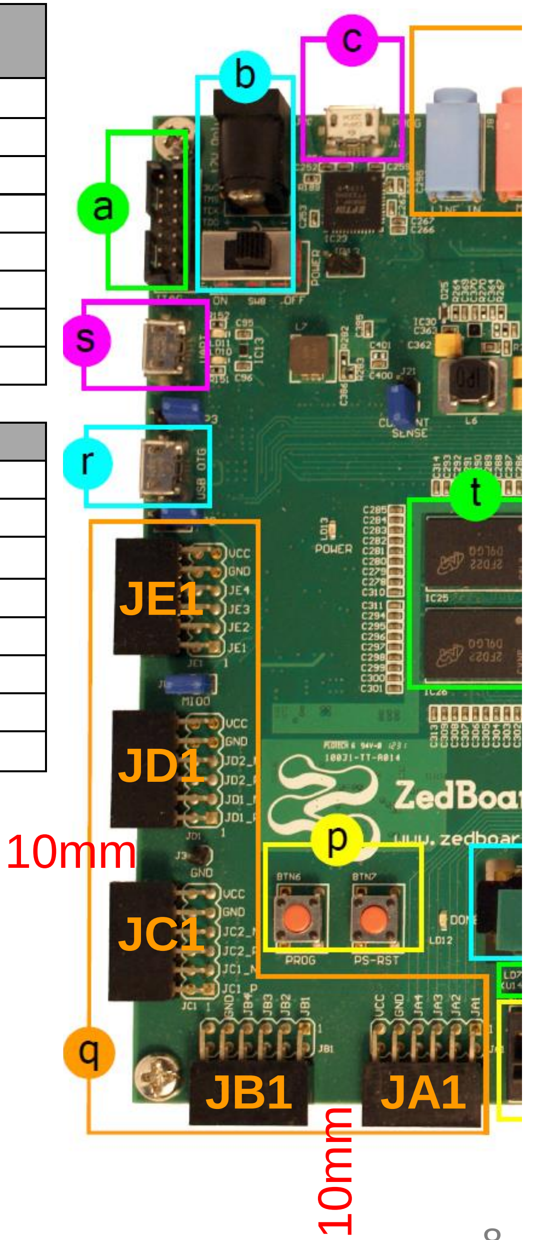


Pmod Ports on ZedBoard (3/3)

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JA1	JA1	Y11	JB1	JB1	W12
	JA2	AA11		JB2	W11
	JA3	Y10		JB3	V10
	JA4	AA9		JB4	W8
	JA7	AB11		JB7	V12
	JA8	AB10		JB8	W10
	JA9	AB9		JB9	V9
	JA10	AA8		JB10	V8

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JC1 Differential	JC1_N	AB6	JD1 Differential	JD1_N	W7
	JC1_P	AB7		JD1_P	V7
	JC2_N	AA4		JD2_N	V4
	JC2_P	Y4		JD2_P	V5
	JC3_N	T6		JD3_N	W5
	JC3_P	R6		JD3_P	W6
	JC4_N	U4		JD4_N	U5
	JC4_P	T4		JD4_P	U6

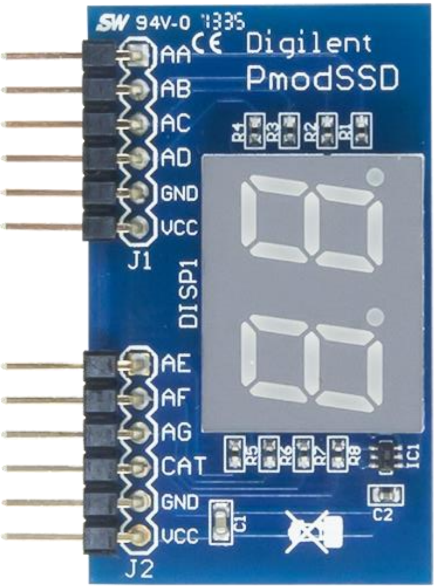
Pmod	Signal Name	Zynq pin	MIO
JE1 MIO Pmod	JE1	A6	MIO13
	JE2	G7	MIO10
	JE3	B4	MIO11
	JE4	C5	MIO12
	JE7	G6	MIO0
	JE8	C4	MIO9
	JE9	B6	MIO14
	JE10	E6	MIO15

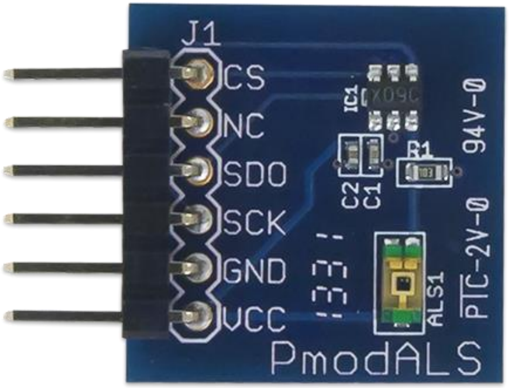


http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf

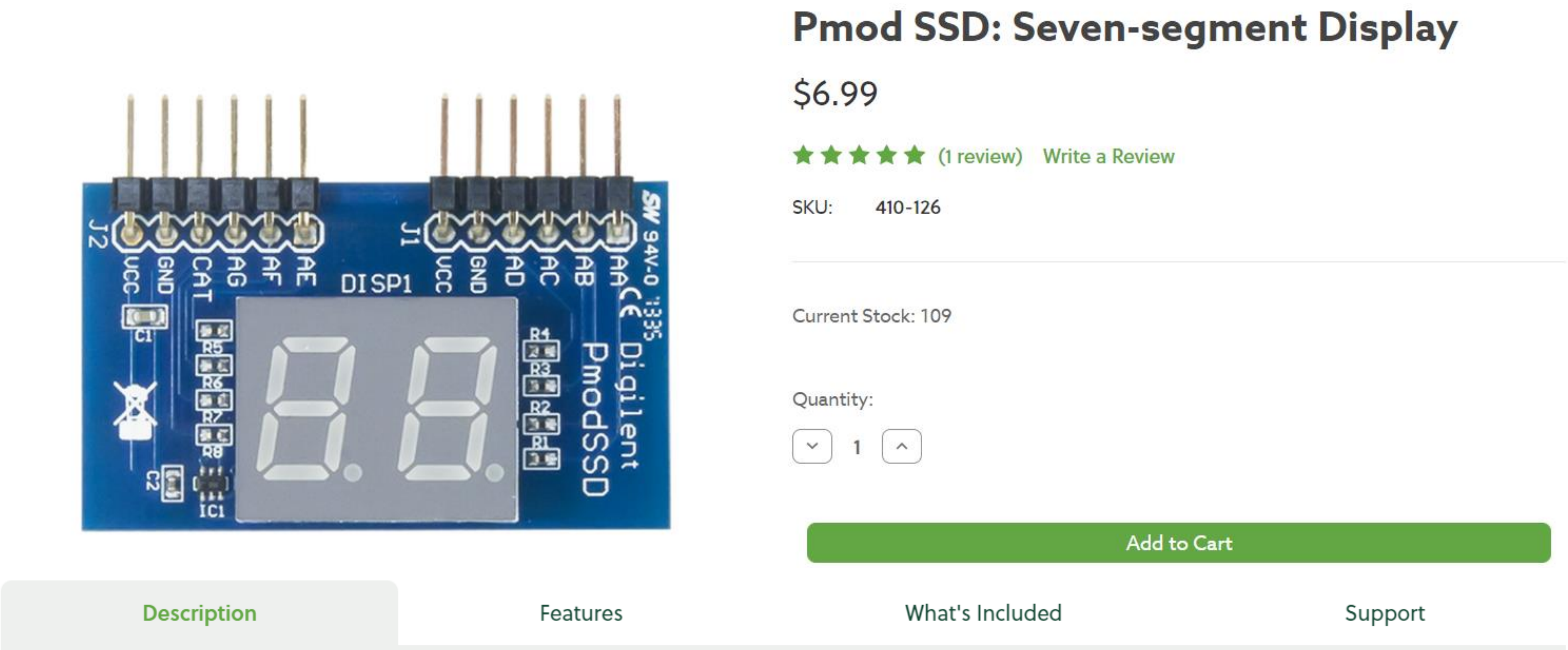
Outline

- Digilent Pmod™ Peripheral Modules
  - Communication Protocols
  - Pmod Ports on ZedBoard
- Case Study 1: Pmod SSD (GPIO)
- Case Study 2: Pmod ALS (SPI Protocol)





Case Study 1: Pmod SSD (1/2)



Pmod SSD: Seven-segment Display

$6.99

★★★★★ (1 review) Write a Review

SKU: 410-126

Current Stock: 109

Quantity:

1

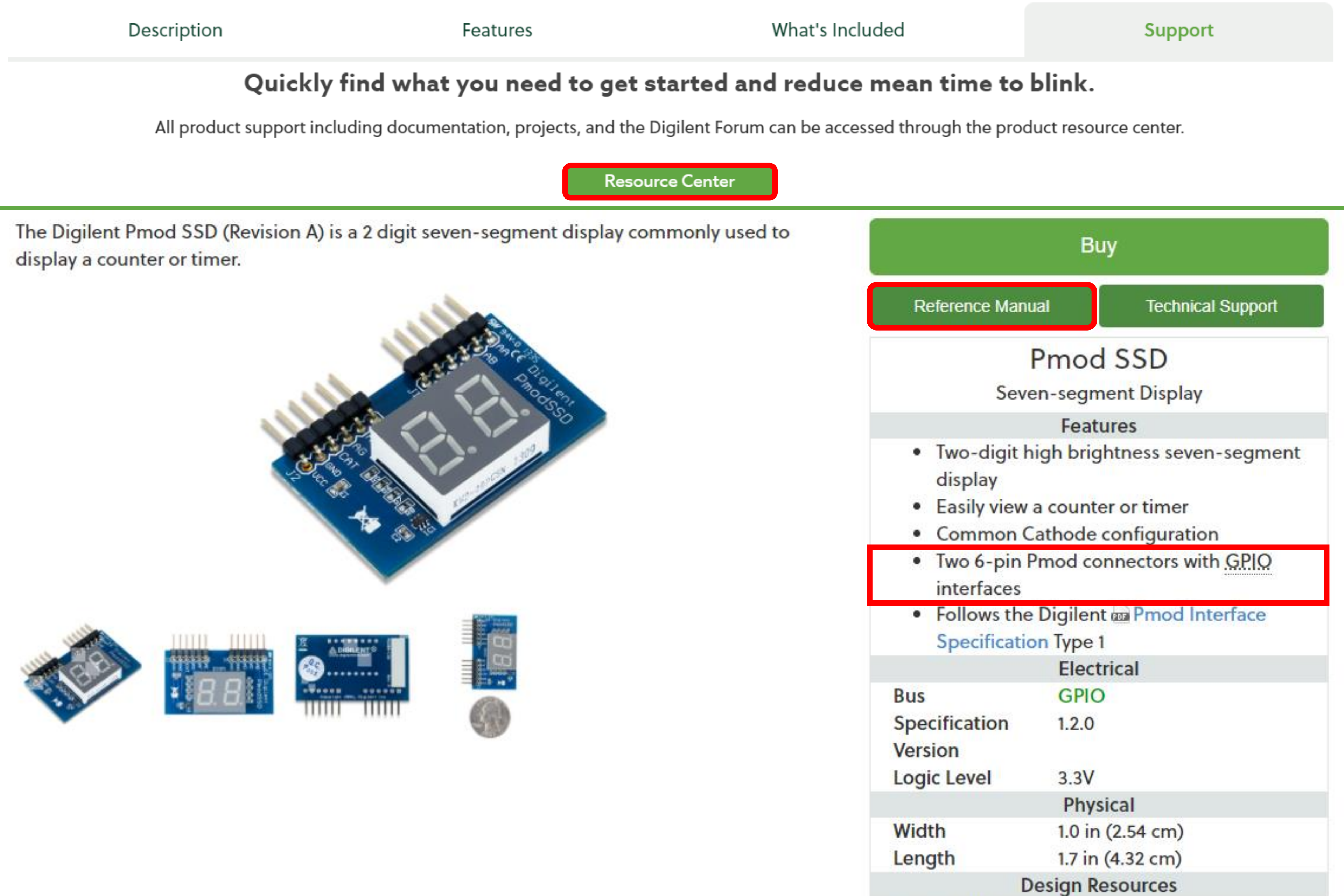
Add to Cart

Description Features What's Included Support

- The Pmod SSD is a two-digit seven-segment display.
- Users can toggle through GPIO signals which digit is currently on at a rate of 50 Hz or greater.
  - Why? To achieve persistence-of-vision to give the effect of both digits being lit up simultaneously.

https://digilent.com/shop/pmod-ssd-seven-segment-display/

Case Study 1: Pmod SSD (2/2)



Quickly find what you need to get started and reduce mean time to blink.

All product support including documentation, projects, and the Digilent Forum can be accessed through the product resource center.

The Digilent Pmod SSD (Revision A) is a 2 digit seven-segment display commonly used to display a counter or timer.

Pmod SSD

Seven-segment Display

Features

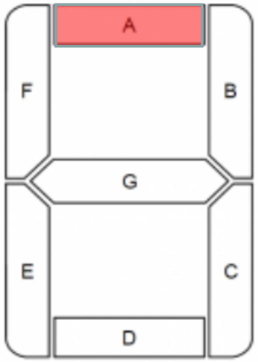
- Two-digit high brightness seven-segment display
- Easily view a counter or timer
- Common Cathode configuration
- Two 6-pin Pmod connectors with GPIO interfaces
- Follows the Digilent Pmod Interface Specification Type 1

Electrical	
Bus	GPIO
Specification Version	1.2.0
Logic Level	3.3V
Physical	
Width	1.0 in (2.54 cm)
Length	1.7 in (4.32 cm)

Design Resources

Pmod SSD: Interfacing via GPIO

- The Pmod SSD communicates with the host board via the General Purpose Input/Output (GPIO) pins.
  - This does not follow a strict set of rules for communication.
  - Rather, the host can send signals to the Pmod via output pins at any time (to have the Pmod immediately respond); the host can also receive signals from Pmod via input pins at any time (to take immediate actions).
    - Since the pins are used for communication, the capability to operate a large number of Pmods might be limited.
- A logic-level high signal on a particular anode will light up that respective segment (i.e., **`Segment A~G`**) immediately.
  - E.g., `ssd[0] <= 1; -- light up segment A`



https://digilent.com/reference/learn/fundamentals/communication-protocols/gpio/start?redirect=1

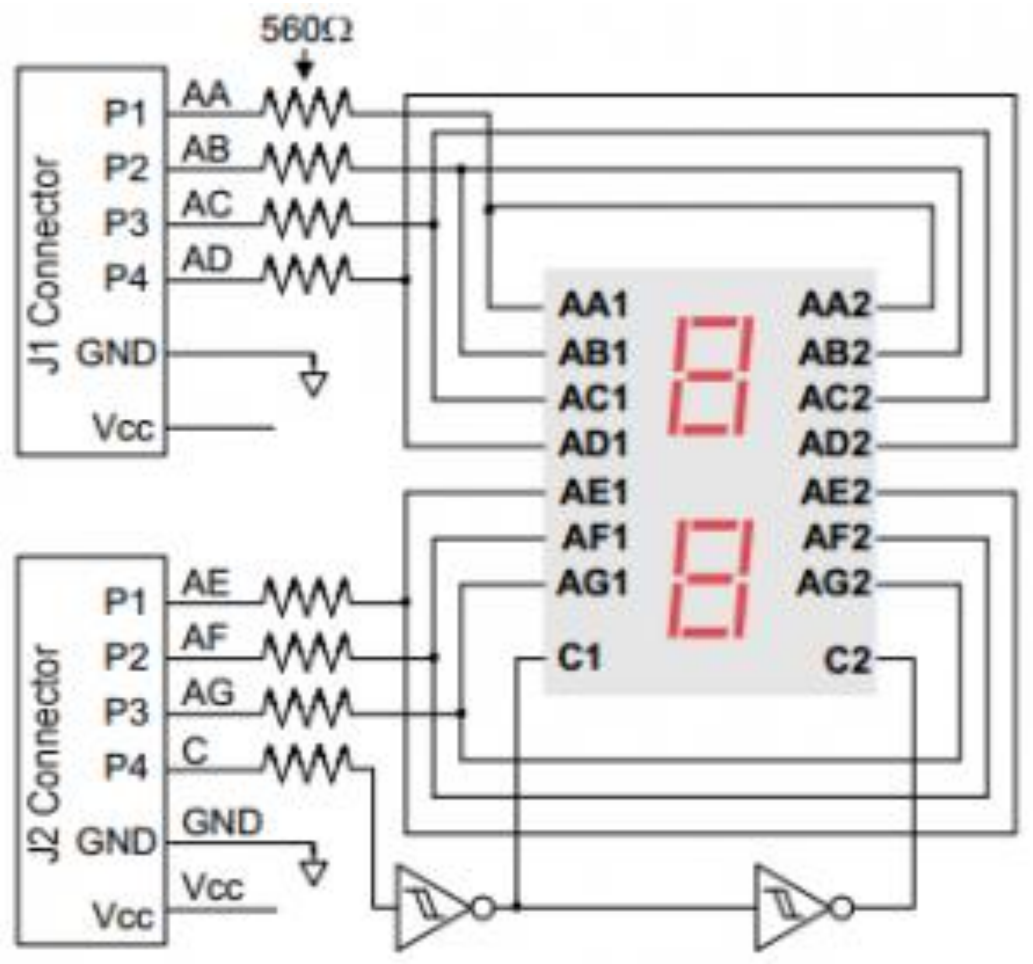
Pmod SSD: Pinout Description Table

Header J1

Pin	Signal	Description	
1	AA	Segment A	
2	AB	Segment B	ssd
3	AC	Segment C	
4	AD	Segment D	
5	GND	Power Supply Ground	
6	VCC	Positive Power Supply	

Header J2

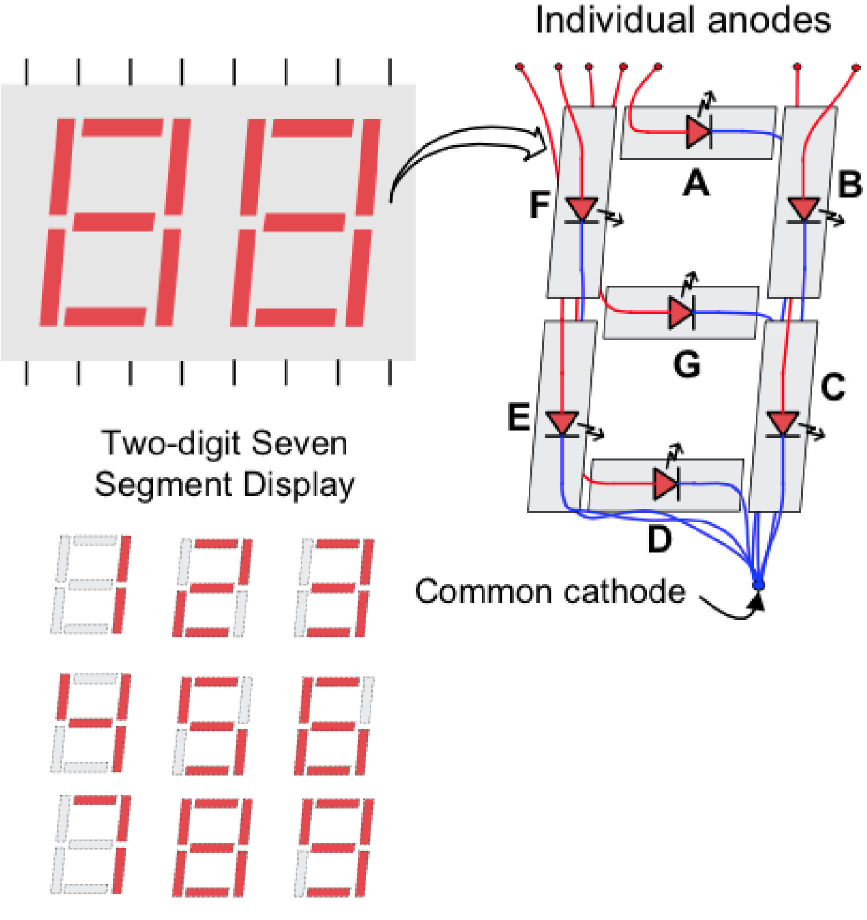
Pin	Signal	Description	
1	AE	Segment E	
2	AF	Segment F	ssd
3	AG	Segment G	
4	C	Digit Selection pin	sel
5	GND	Power Supply Ground	
6	VCC	Positive Power Supply	



https://store.digilentinc.com/pmod-ssd-seven-segment-display/

Pmod SSD: LED Mapping and Activation

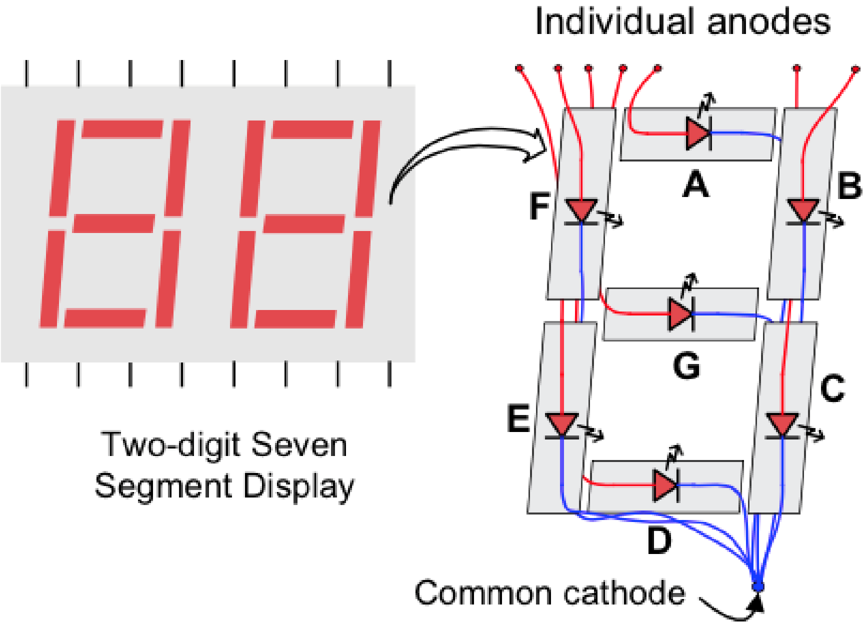
- Each digit has seven LEDs, labeled **A** through **G**.
- To make digits? We can light up segments as below:



Digit	Segments	Value (ssd)
0	A B C D E F	“1111110”
1	B C	“0110000”
2	A B D E G	“1101101”
3	A B C D G	“1111001”
4	B C F G	“0110011”
5	A C D F G	“1011011”
6	A C D E F G	“1011111”
7	A B C	“1110000”
8	A B C D E F G	“1111111”
9	A B C F G	“1110011”

Class Exercise 6.1

- Show how to activate the LED values (**ssd**) for hexadecimal digits: **A, b, C, d, E, F.**



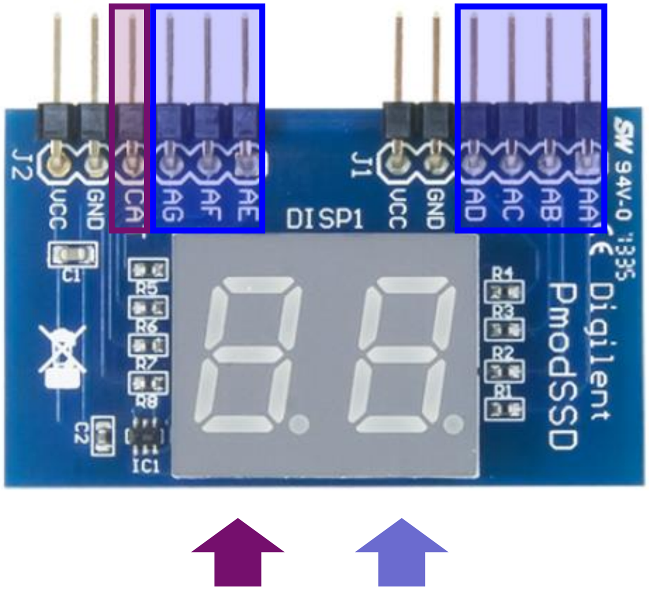
Two-digit Seven Segment Display

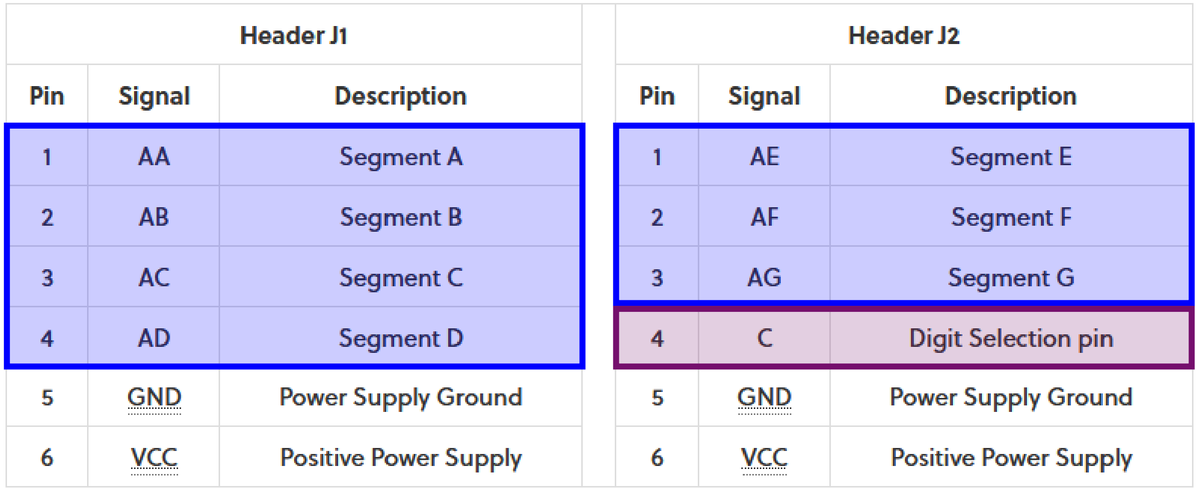
Digit	Segments	Value (ssd)
0	A B C D E F	"1111110"
1	B C	"0110000"
2	A B D E G	"1101101"
3	A B C D G	"1111001"
4	B C F G	"0110011"
5	A C D F G	"1011011"
6	A C D E F G	"1011111"
7	A B C	"1110000"
8	A B C D E F G	"1111111"
9	A B C F G	"1110011"
A		
b		
C		
d		
E		
F		

Pmod SSD: Time Multiplexing (1/2)

- Seven pins (i.e., **`ssd`**) are "shared" by two displays to control the seven segments of each digit.
- One pin (i.e., **`sel`**) is to select which display to drive.

Left Digit: **`sel=1`** *Right Digit*: **`sel=0`**



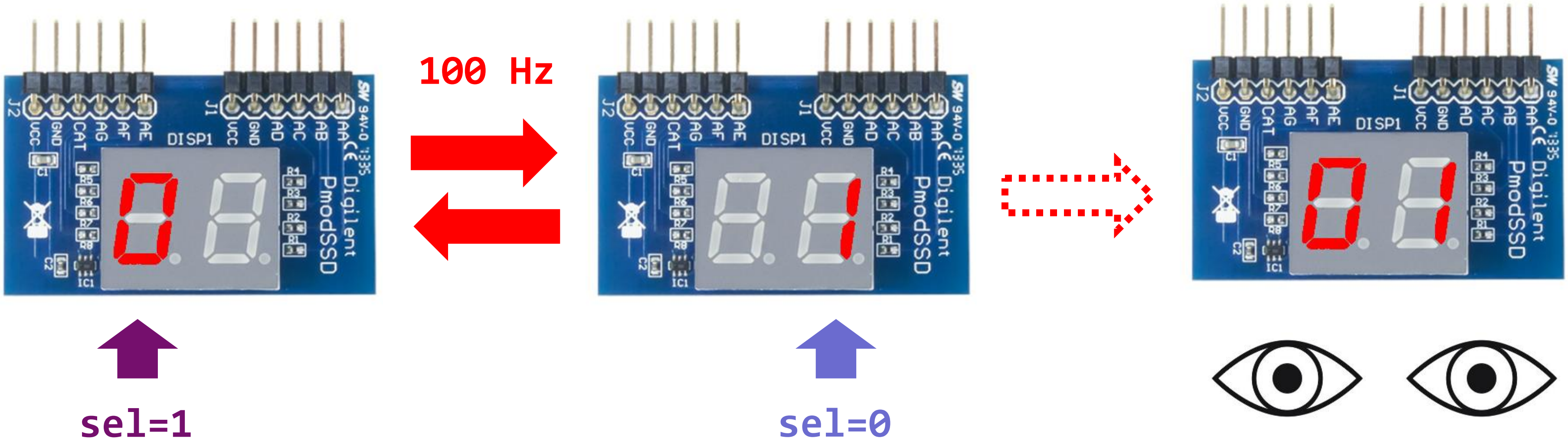


Header J1		
Pin	Signal	Description
1	AA	Segment A
2	AB	Segment B
3	AC	Segment C
4	AD	Segment D
5	GND	Power Supply Ground
6	VCC	Positive Power Supply

Header J2		
Pin	Signal	Description
1	AE	Segment E
2	AF	Segment F
3	AG	Segment G
4	C	Digit Selection pin
5	GND	Power Supply Ground
6	VCC	Positive Power Supply

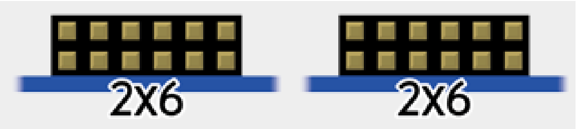
Pmod SSD: Time Multiplexing (2/2)

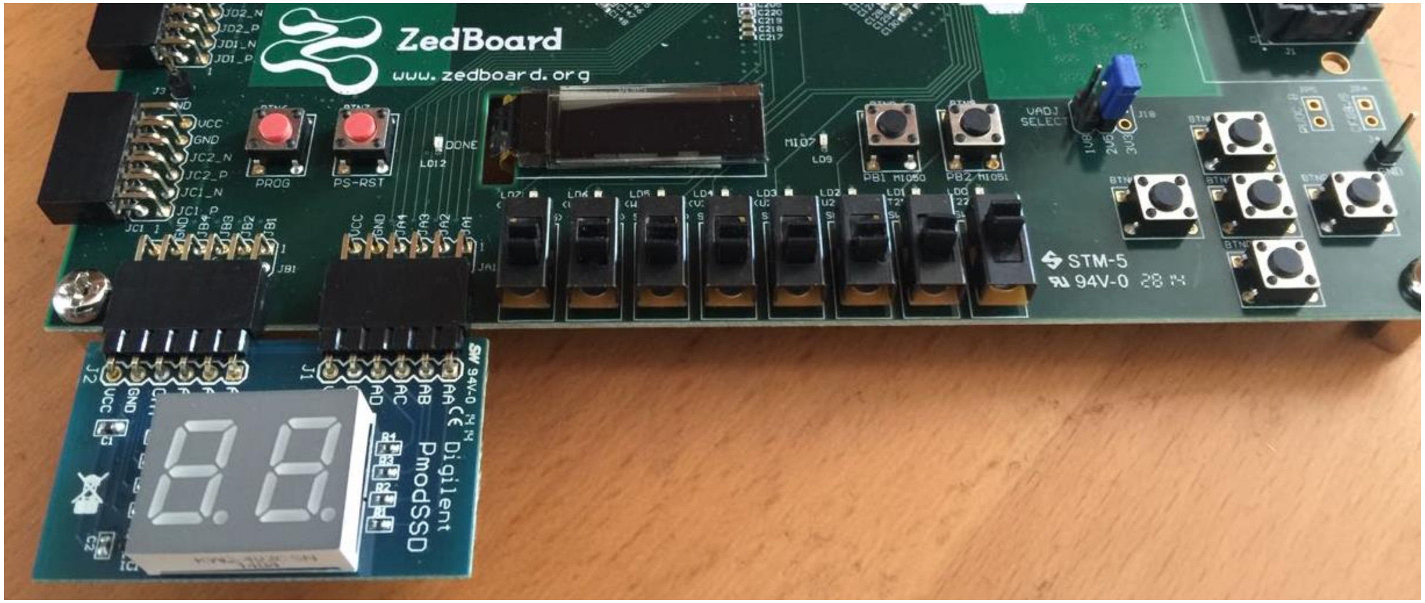
- To display both digits, we need to alternate between the two digits faster enough than eyes can perceive.
  - For example, we can alternately activate the 7-segment on the right (and then on the left) at a rate of (at least) 50 Hz.
  - It will then look like both digits are displayed simultaneously.
    - Why? Persistence-of-vision!

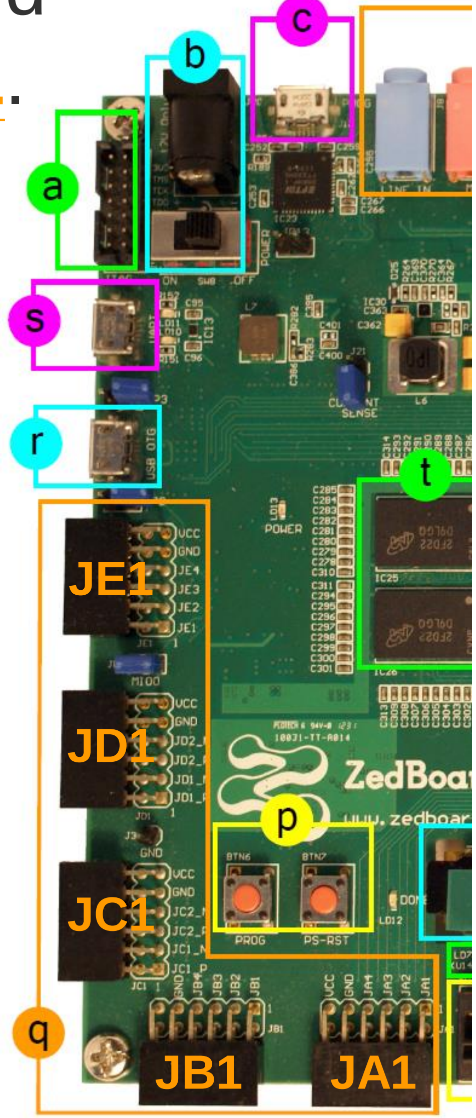


Pmod SSD: Connect to ZedBoard

- You can connect Pmod SSD to ZedBoard through either **JA1 & JB1** or **JC1 & JD1**.
  - Either the top row of pins or the bottom rows of pins is fine, but the used pins must be specified in the XDC file.







Pmod SSD: XDC Constraint File

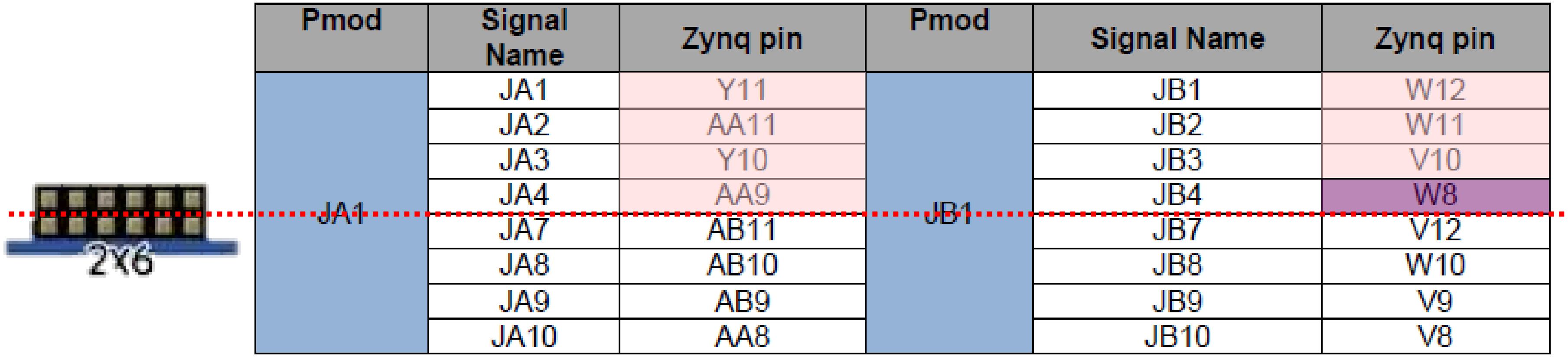
- To drive the Pmod SSD, you also need the following:

Seven Segments (ssd)

```
set_property IOSTANDARD LVCMOS33 [get_ports ssd]
set_property PACKAGE_PIN Y11  [get_ports {ssd[6]}]
set_property PACKAGE_PIN AA11 [get_ports {ssd[5]}]
set_property PACKAGE_PIN Y10  [get_ports {ssd[4]}]
set_property PACKAGE_PIN AA9  [get_ports {ssd[3]}]
set_property PACKAGE_PIN W12  [get_ports {ssd[2]}]
set_property PACKAGE_PIN W11  [get_ports {ssd[1]}]
set_property PACKAGE_PIN V10  [get_ports {ssd[0]}]
```

Digit Select (sel)

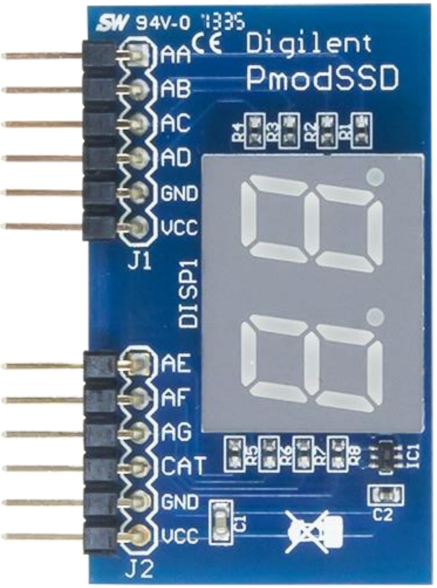
```
set_property IOSTANDARD LVCMOS33 [get_ports sel]
set_property PACKAGE_PIN W8 [get_ports sel]
```

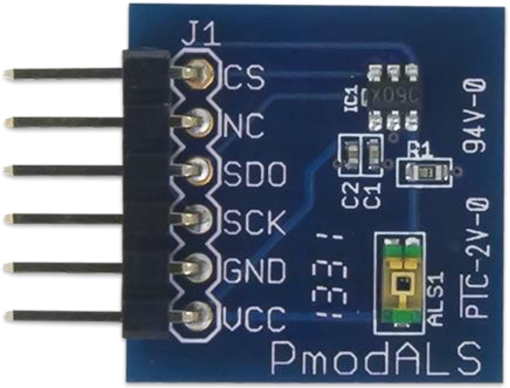


Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JA1	JA1	Y11	JB1	JB1	W12
	JA2	AA11		JB2	W11
	JA3	Y10		JB3	V10
	JA4	AA9		JB4	W8
	JA7	AB11		JB7	V12
	JA8	AB10		JB8	W10
	JA9	AB9		JB9	V9
	JA10	AA8		JB10	V8

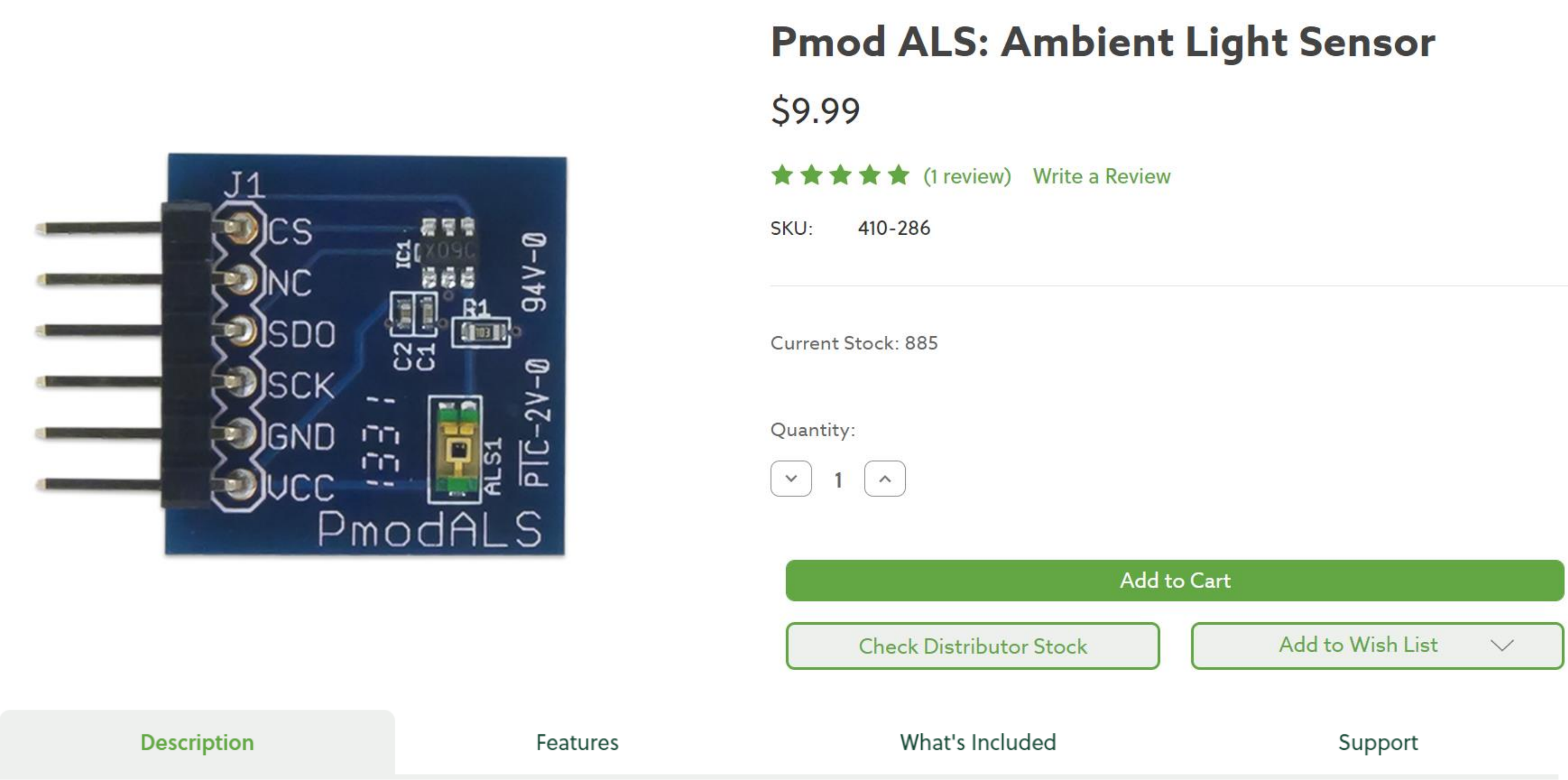
Outline

- Digilent Pmod™ Peripheral Modules
  - Communication Protocols
  - Pmod Ports on ZedBoard
- Case Study 1: Pmod SSD (GPIO)
- Case Study 2: Pmod ALS (SPI Protocol)





Case Study 2: Pmod ALS (1/2)



Pmod ALS: Ambient Light Sensor

$9.99

★★★★★ (1 review) Write a Review

SKU: 410-286

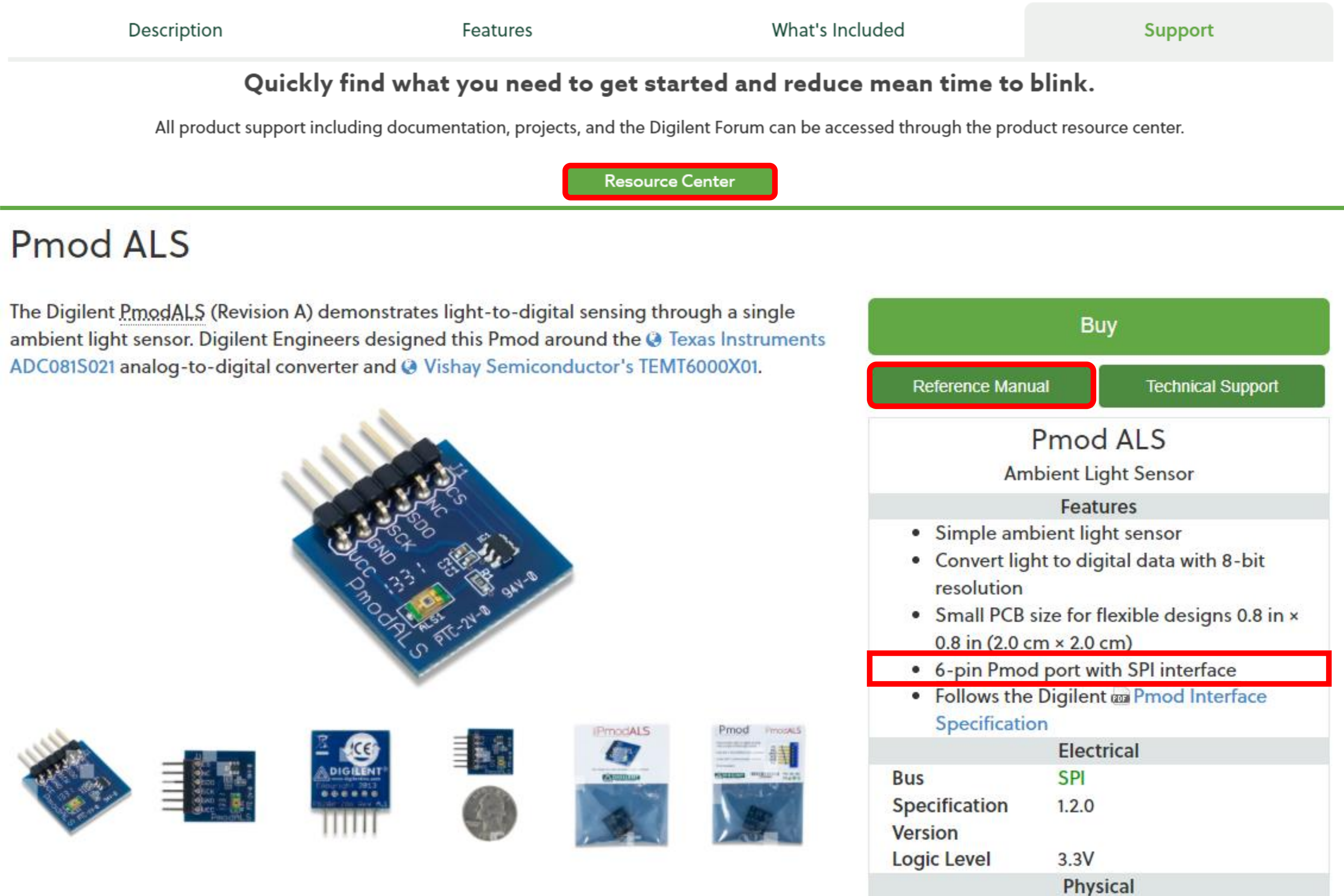
Current Stock: 885

Quantity: 1

- The Pmod ALS demonstrates light-to-digital sensing through a single ambient light sensor.

https://digilent.com/shop/pmod-ssd-seven-segment-display/

Case Study 2: Pmod ALS (2/2)



Description | Features | What's Included | Support

Quickly find what you need to get started and reduce mean time to blink.

All product support including documentation, projects, and the Digilent Forum can be accessed through the product resource center.

Resource Center

Pmod ALS

The Digilent PmodALS (Revision A) demonstrates light-to-digital sensing through a single ambient light sensor. Digilent Engineers designed this Pmod around the Texas Instruments ADC081S021 analog-to-digital converter and Vishay Semiconductor's TEMT6000X01.

Buy

Reference Manual | Technical Support

Pmod ALS

Ambient Light Sensor

Features

- Simple ambient light sensor
- Convert light to digital data with 8-bit resolution
- Small PCB size for flexible designs 0.8 in × 0.8 in (2.0 cm × 2.0 cm)
- 6-pin Pmod port with SPI interface
- Follows the Digilent Pmod Interface Specification

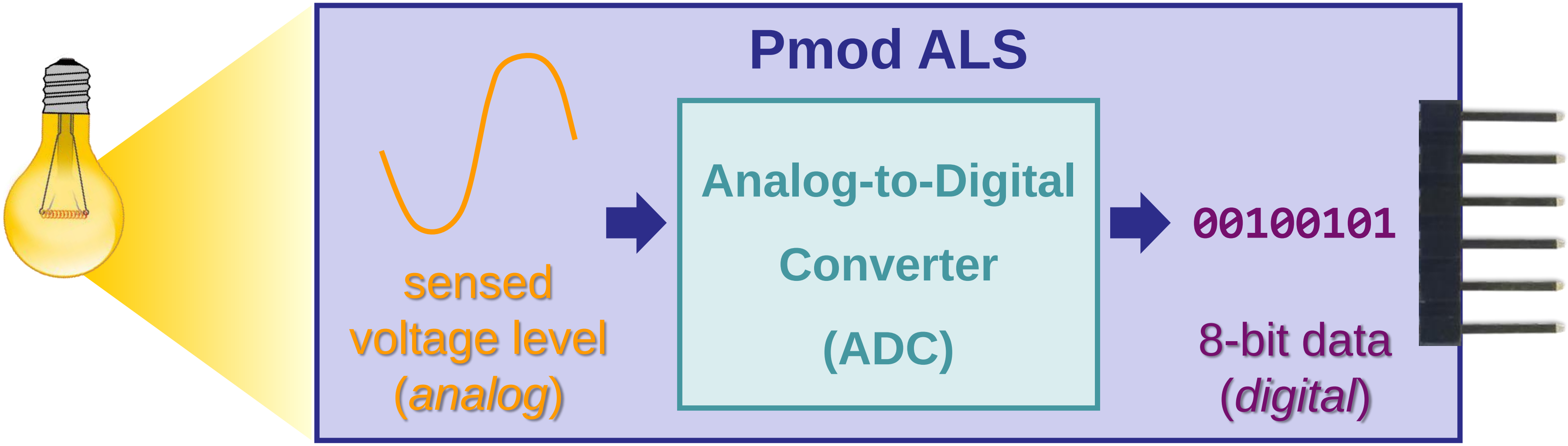
Electrical

Bus	SPI
Specification Version	1.2.0
Logic Level	3.3V

Physical

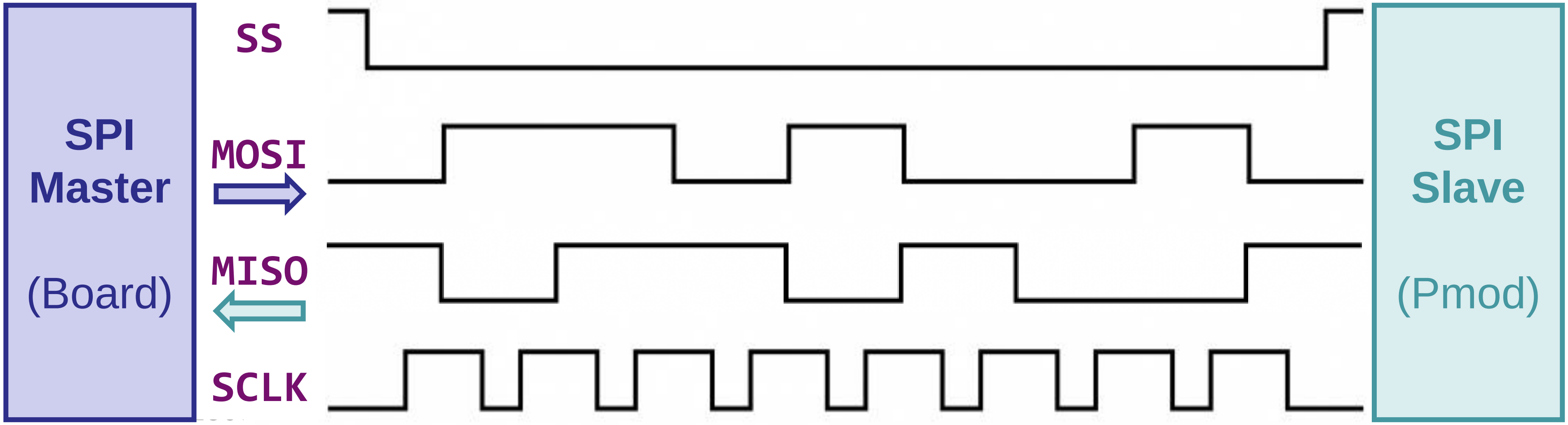
Pmod ALS: Function Description

- The PmodALS utilizes a single ambient light sensor (ALS) for user input.
  - The amount of light (where the ALS is exposed to) determines the voltage level;
  - The voltage level is then passed into an on-board analog-to-digital converter (ADC), which converts it to 8 bits of data.
    - A value of `0` indicates the lowest light level;
    - A value of `255` indicates the highest light level.


Pmod ALS
sensed voltage level (analog)
Analog-to-Digital Converter (ADC)
00100101
8-bit data (digital)

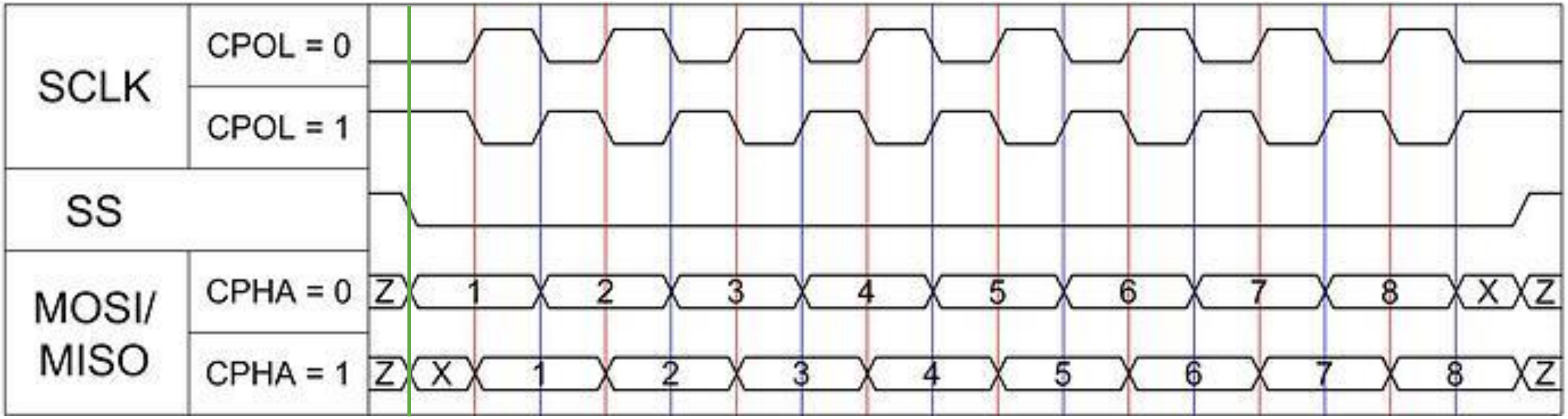
Pmod ALS: Interfacing via SPI (1/4)

- The host board must communicate with the PmodALS via the Serial Peripheral Interface (SPI) protocol.
  - SPI communication is a full-duplex data link using four wires.
    - The master (i.e., the board) initiates or terminates the communication by pulling down or pulling up the Slave Select (**SS**) respectively.
    - The master drives a Serial Clock (**SCLK**) line to provide a sync. clock.
    - The master transmits data via the Master-Out-Slave-In (**MOSI**) wire and receives data via the Master-In-Slave-Out (**MISO**) wire, at the rate of one bit of data per clock cycle.



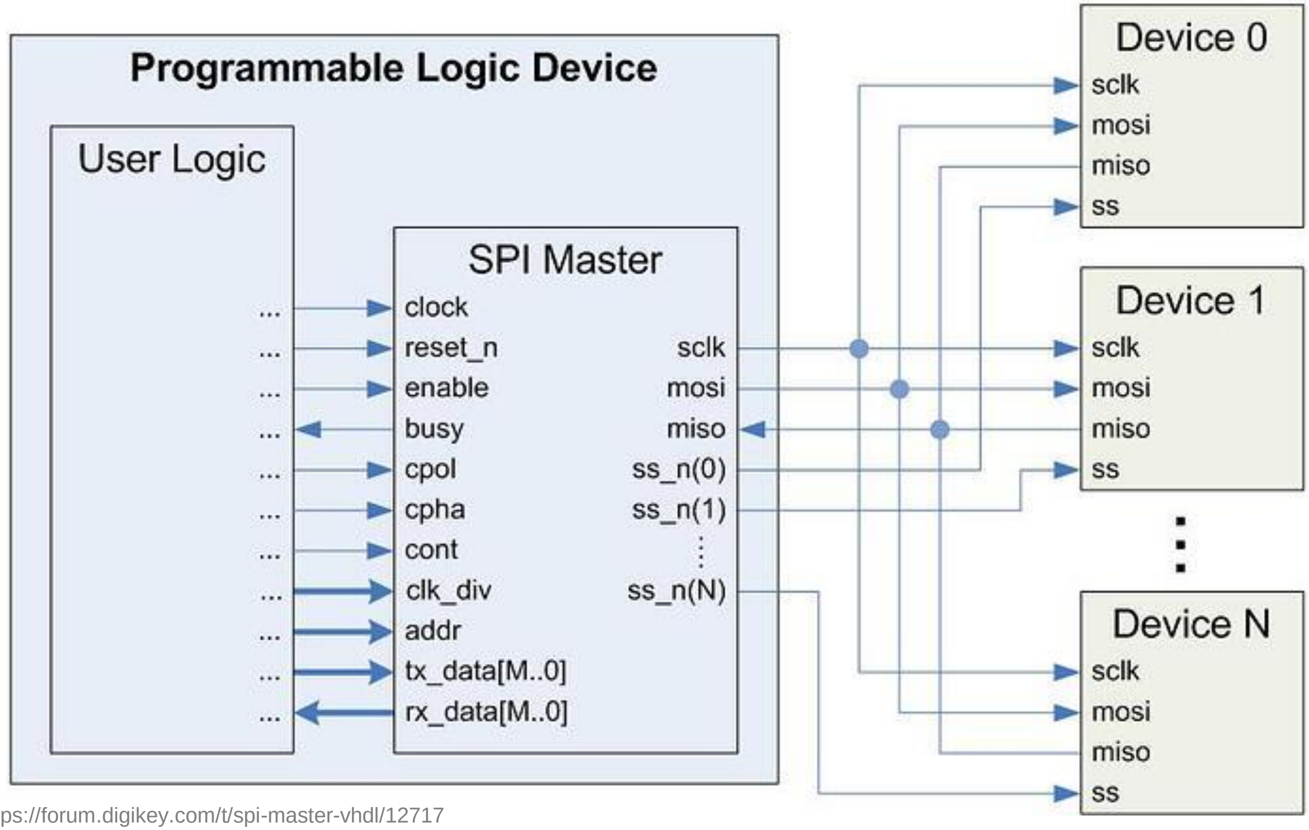
Pmod ALS: Interfacing via SPI (2/4)

- SPI has four operation modes based on two parameters: clock polarity (**CPOL**) and clock phase (**CPHA**).
  - If **CPOL=0**, the first clock edge is a rising edge; otherwise (if **CPOL=1**), the first clock edge is a falling edge.
  - If **CPHA=0**, the first bit is written on the SS falling edge and read on the first SCLK edge; otherwise, the first bit is written on the first SCLK edge and read on the second SCLK edge.
  - The mode is chosen based on how the device is designed.



Pmod ALS: Interfacing via SPI (3/4)

- We may use the third-party SPI master to ease the communication with a SPI-based Pmod.
  - All you have to do is to “properly configure” the SPI master in “**`generic map`**” and “**`port map`**”.

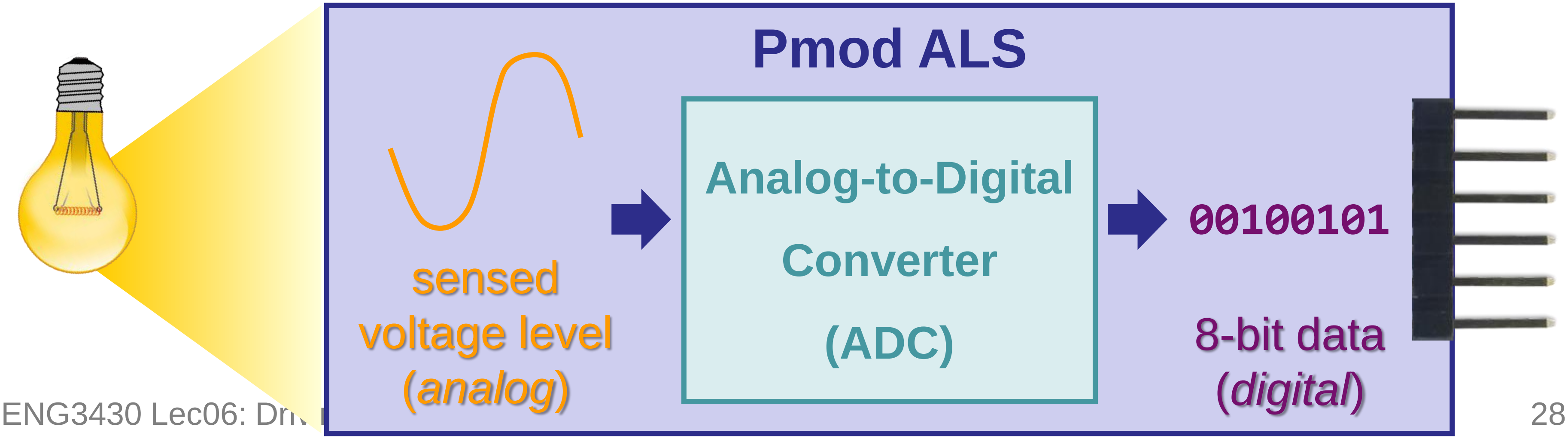


https://forum.digikey.com/t/spi-master-vhdl/12717

```
spi_master
GENERIC MAP(
      slaves  => ?,
      d_width => ?)
PORT MAP(
      clock   => ?,
      reset_n => ?,
      enable  => ?,
      cpol    => ?,
      cpha    => ?,
      cont    => ?,
      clk_div => ?,
      addr    => ?,
      tx_data => ?,
      miso    => ?,
      sclk    => ?,
      ss_n    => ?,
      mosi    => ?,
      busy    => ?,
      rx_data => ?);
```

Pmod ALS: Interfacing via SPI (4/4)

- The on-board ADC of Pmod ALS is read-only.
  - The only wires required are the Slave Select (**`SS`**), Master-In-Slave-Out (**`MISO`**), and Serial Clock (**`SCLK`**).
- The on-board ADC generates 8-bit data regarding the light level but the actual data to be sent are of 15 bits.
  - The first 3 bits are leading zeros; bits 4~11 indicate the light level with the MSB first; and bits 12~15 are trailing zeros.
- The serial clock should run between 1 MHz ~ 4 MHz.


Pmod ALS
sensed voltage level (*analog*)
Analog-to-Digital Converter (ADC)
00100101
8-bit data (*digital*)

Class Exercise 6.2

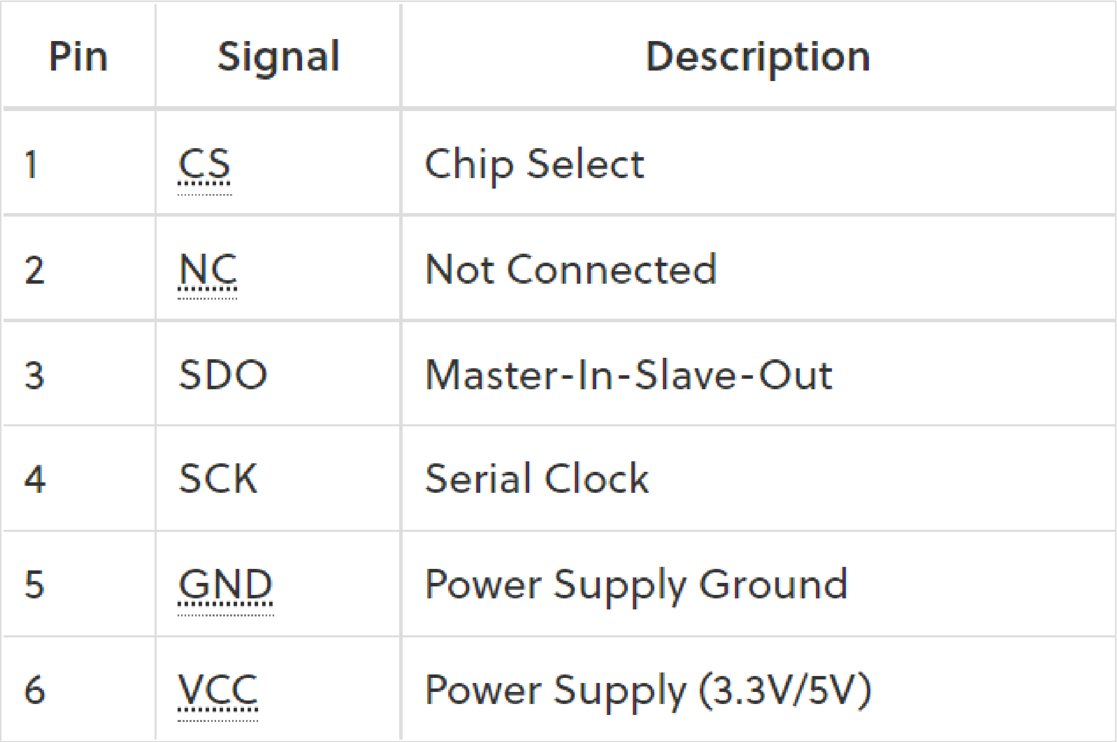
- The on-board ADC generates 8-bit data regarding the light level but the actual data to be sent are of 15 bits.
  - The first 3 bits are leading zeros; bits 4~11 indicate the light level with the MSB first; and bits 12~15 are trailing zeros.

- Given the 15-bit data **rx_data** received from Pmod ALS, extract the information about the light level (i.e., bits 4~11) and assign it to an 8-bit signal **s**:

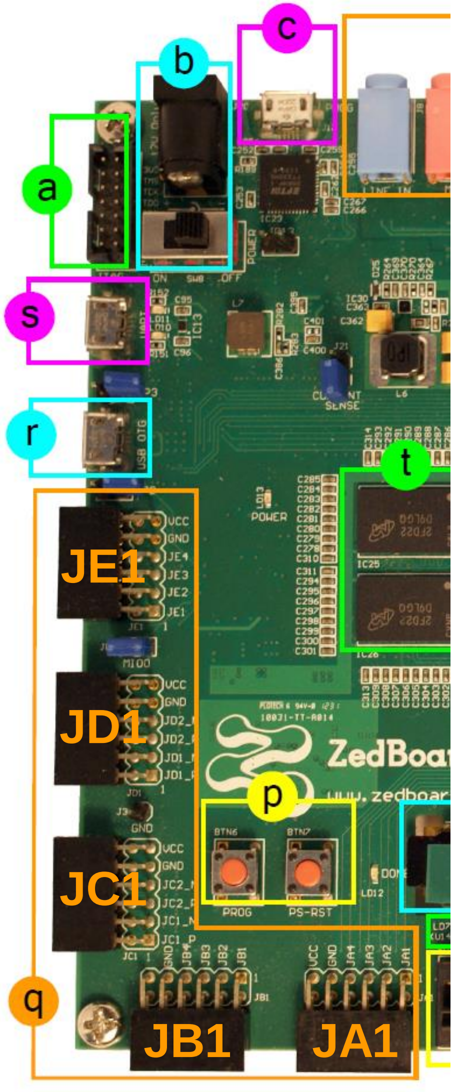
```
signal s : std_logic_vector(7 downto 0);
```

Pmod ALS: Connect and Pinout Table

- You can connect Pmod ALS to ZedBoard through either of JA1, JB1, JC1, or JD1.
  - Either the top row of pins or the bottom rows of pins is fine, but the used pins must be specified in the XDC file.



Pin	Signal	Description
1	CS	Chip Select
2	NC	Not Connected
3	SDO	Master-In-Slave-Out
4	SCK	Serial Clock
5	GND	Power Supply Ground
6	VCC	Power Supply (3.3V/5V)



Pmod ALS: XDC Constraint File

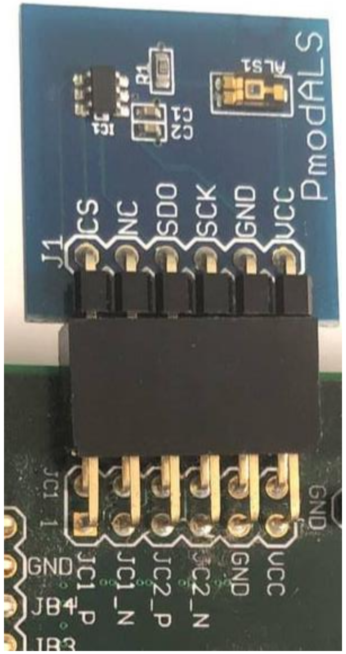
- To drive the Pmod ALS, you also need the following:

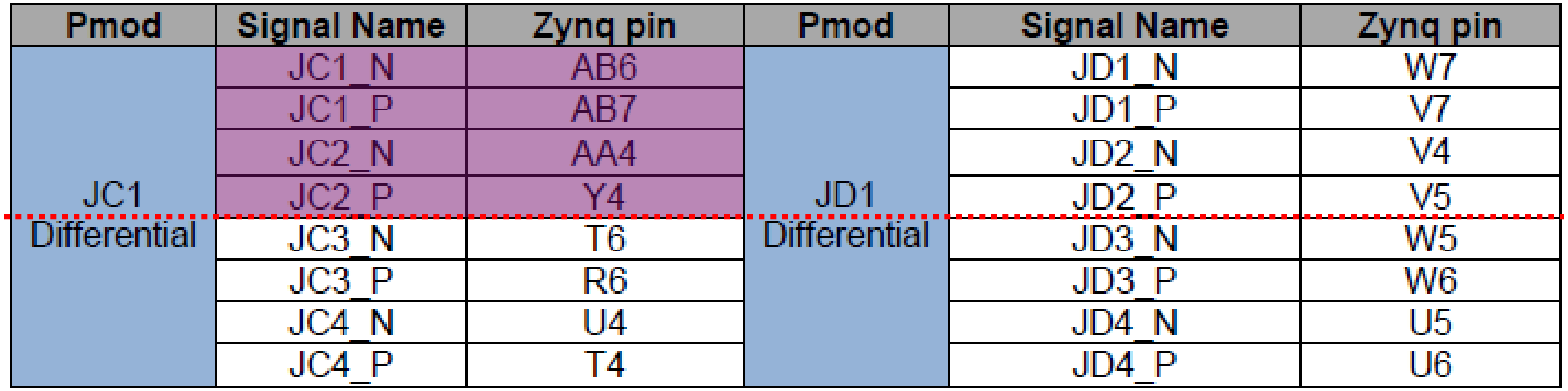
```
set_property PACKAGE_PIN AB7 [get_ports {cs}];
set_property IOSTANDARD LVCMOS33 [get_ports cs];

~~set_property PACKAGE_PIN AB6 [get_ports {mosi}];~~
~~set_property IOSTANDARD LVCMOS33 [get_ports mosi];~~

set_property PACKAGE_PIN Y4 [get_ports {miso}];
set_property IOSTANDARD LVCMOS33 [get_ports miso];

set_property PACKAGE_PIN AA4 [get_ports {sclk}];
set_property IOSTANDARD LVCMOS33 [get_ports sclk];
```

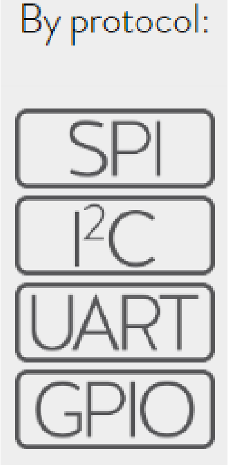


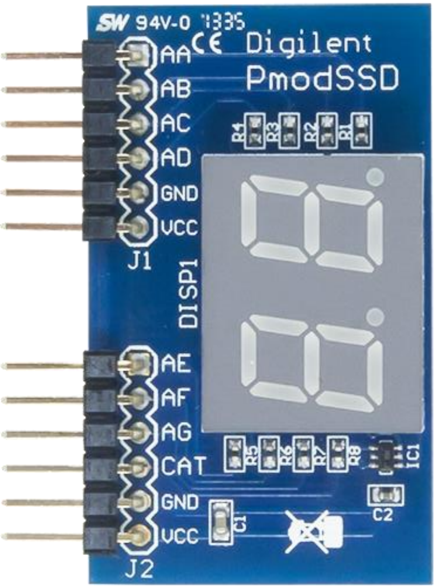


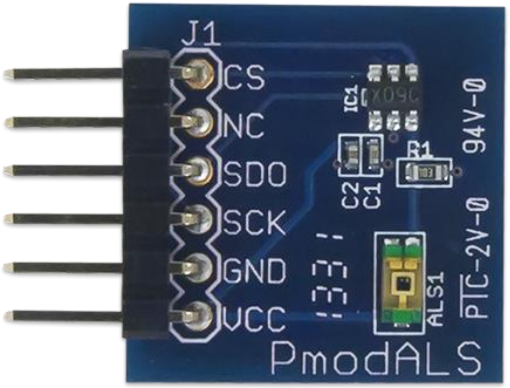
Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JC1 Differential	JC1_N	AB6	JD1 Differential	JD1_N	W7
	JC1_P	AB7		JD1_P	V7
	JC2_N	AA4		JD2_N	V4
	JC2_P	Y4		JD2_P	V5
	JC3_N	T6		JD3_N	W5
	JC3_P	R6		JD3_P	W6
	JC4_N	U4		JD4_N	U5
	JC4_P	T4		JD4_P	U6

Summary

- Digilent Pmod™ Peripheral Modules
  - Communication Protocols
  - Pmod Ports on ZedBoard
- Case Study 1: Pmod SSD (GPIO)
- Case Study 2: Pmod ALS (SPI Protocol)


By protocol:





Interfacing via I^{2}C or UART?

- We may also use the third-party I^{2}C master or the third-party UART transmitter/receiver to ease the communication with an I^{2}C or UART-based Pmod.
  - In a similar way that we use the third-party SPI master!

```
i2c_master
GENERIC MAP(
    input_clk => ?,
    bus_clk   => ?)
PORT MAP(
    clk       => ?,
    reset_n   => ?,
    ena       => ?,
    addr      => ?,
    rw        => ?,
    data_wr   => ?,
    busy      => ?,
    data_rd   => ?,
    ack_error => ?,
    sda       => ?,
    scl       => ?);
```

```
uart_tx -- Transmitter
GENERIC MAP(
    g_CLKS_PER_BIT => ?)
PORT MAP(
    i_clk       => ?,
    i_tx_dv     => ?,
    i_tx_byte   => ?,
    o_tx_active => ?,
    o_tx_serial => ?,
    o_tx_done   => ?);
```

```
uart_rx -- Receiver
GENERIC MAP(
    g_CLKS_PER_BIT => ?)
PORT MAP(
    i_clk       => ?,
    i_rx_serial => ?,
    o_rx_dv     => ?,
    o_rx_byte   => ?);
```

All you have to do is to “properly configure” them in “generic map” and “port map”!!!

https://forum.digikey.com/t/i2c-master-vhdl/12797
https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html