

PMOD for DE1-SoC

By Fred Aulich REV1.0 Nov 2018

This document contains all the files and information for making PMODS modules work with the DE1-SoC

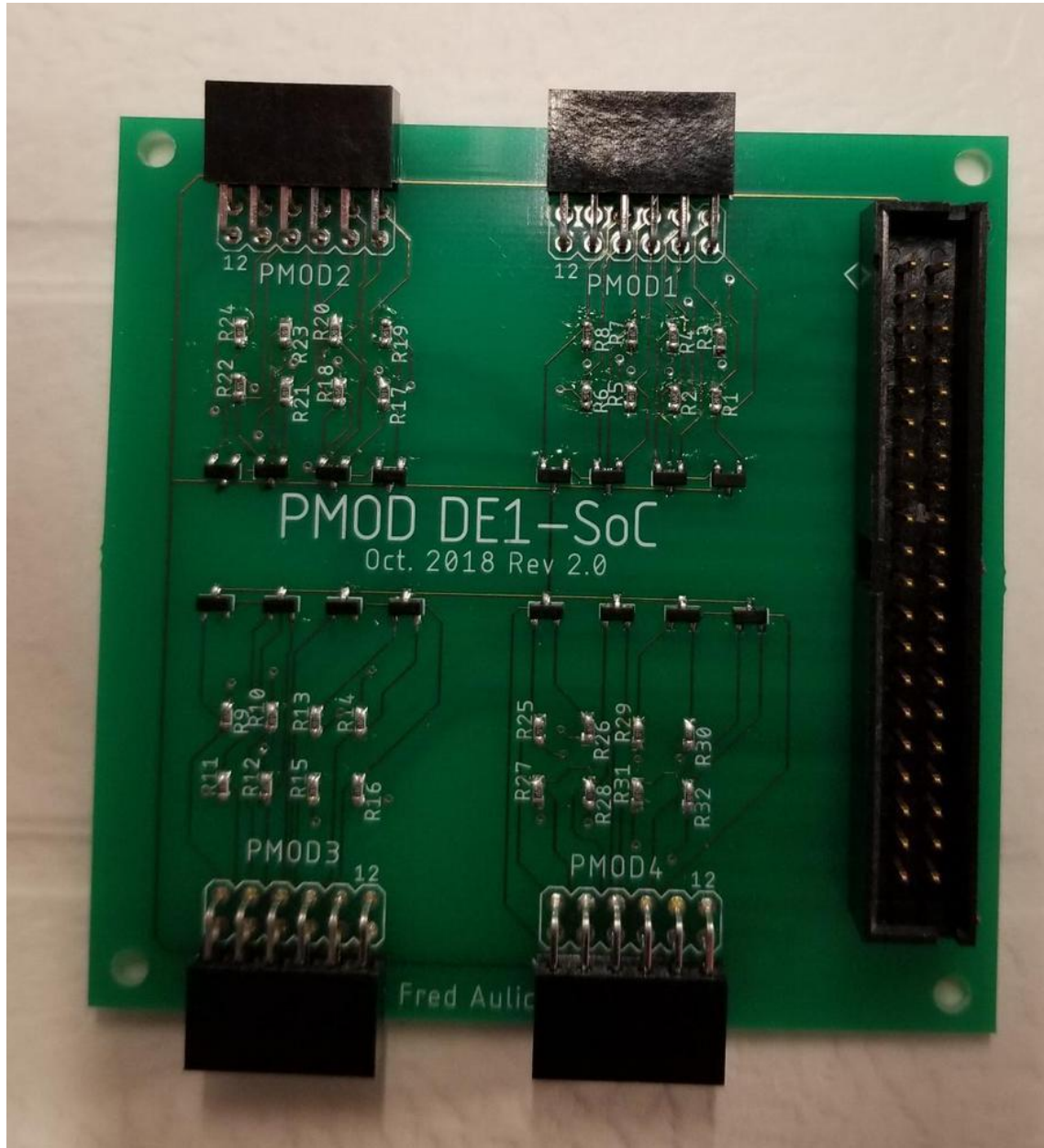


Figure 1 - PMOD DE1-SoC board

Figure 1 shows the PC board with 4 PMOD ports and a GPIO connector. All parts on the board were bought at Digikey

<https://www.digikey.ca>

Below is a list of the parts the were bought to populate the board. All parts are pretty standard.





	Part Number: ED10529-ND Manufacturer Part Number: 302-S401 Manufacturer: On Shore Technology Inc. Customer Reference: FPGA HEADER Description: CONN HEADER VERT 40POS GOLD
	S5520-ND PPTC062LJBN-RC CONN FEMALE 12POS DL .1" R/A TIN
	CR0603-JW-201GLFCT-ND CR0603-JW-201GLF RES SMD 200 OHM 5% 1/10W 0603
	AZ23C3V3-E3-08GICT-ND AZ23C3V3-E3-08 DIODE ZENER 3.3V 300MW SOT23

Figure 2 Descriptive list of parts bought at Digikey

Quantity	Part number	Description
32	CR0603-JW-201GLF	200 ohm surface mount resistor
16	AZ23C3V3-E3-08	Zener Diode
4	PPTC062LJBN-RC	12 pin right angle female connector
1	ED10529-ND	40 pin header

Table 1 reference part list and quantity per board

The files for manufacturing the **PC board** can be found on my work PC in the following directory

aulich>Documents>EAGLE>projects>PMOD_REV_final.

All these files need to be zipped up and sent to the manufacturer. To view the PC design, you will need to download an education copy of the CAD PC board design tool called **EAGLE**. You can get a free version which is what I used to design the PCB in **Figure 1**.

Pinout for each PMOD 12 pin header

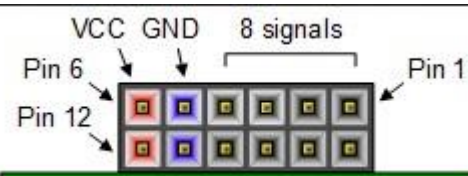


Figure 3- Pin assignment for each PMOD header

Each PMOD header consists of 12 pins. 8 pins are for signals. Two pins are reserved for **ground** and 2 pins are reserved for **VCC**.

Depending on which PMOD protocol is used, will determine how the signal pins are assigned. There are four different protocol used for the most common PMODS

SPI- Serial Peripheral Interface Protocol

Pin	Signal	Direction
1	SS	Out
2	MOSI	Out
3	MISO	In
4	SCK	Out
5	GND	
6	VCC	

Table 2-Serial Peripheral Interface

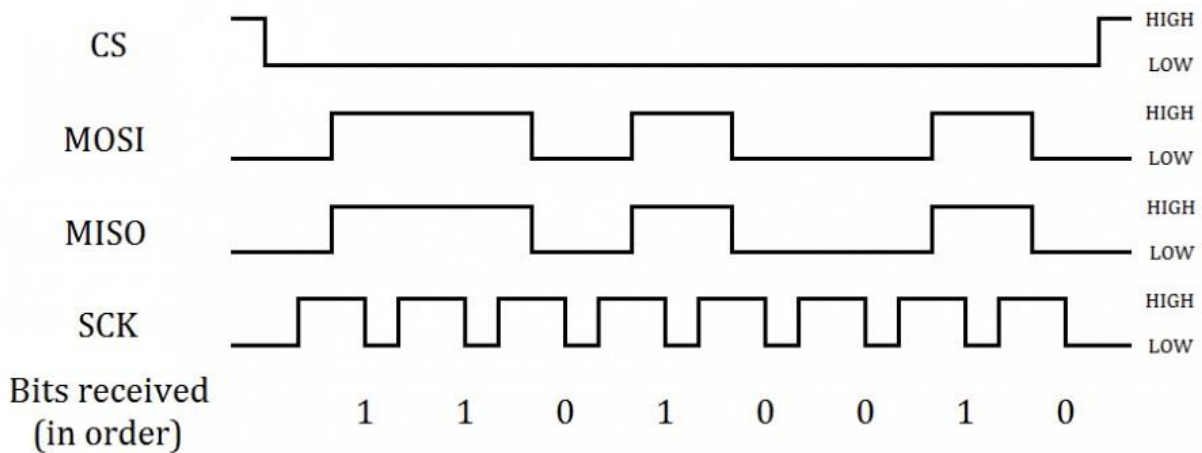


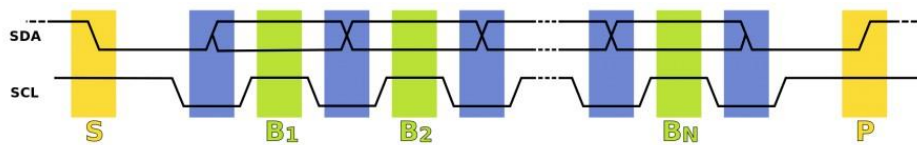
Figure 4- Timing diagram for SPI

Please **note** that **CS** and **SS** [Pin 1] are the same thing.

I2C - Inter-Integrated Circuit protocol

Pin Number	Signal	Pin Number	Signal
1	SCL	2	SCL
3	SDA	4	SDA
5	GND	6	GND
7	VCC	8	VCC

Table 3- Inter-integrated Circuit



- Here, you can see that a communication session begins with a start bit, where the SDA line is driven low while the SCL line remains high through the pull-up resistors.
- Once SCL is driven low by the master device, the SDA line is either driven low or left floating high to set the data bit level as shown by each of the blue bars.
- When the Serial Clock line reaches a high voltage level (after any clock stretching has been completed), the SDA line is then read by the receiving device to collect the data bit, as shown by each of the green bars.
- This process of SCL transitions and data bits being placed and read continues until a stop bit, indicated by the SDA being pulled high by the pull-up resistors while the SCL line is high, occurs.

UART- Universal Asynchronous Receive Transmit

Pin	Signal	Direction
1	CTS	Out
2	RTS	In
3	RXD	In
4	TXD	Out
5	GND	
6	VCC	

- CTS - Permission for peripheral to send data to host
 RTS - Request from peripheral to send data to host
 RXD - Data from peripheral to host
 TXD - Data from host to peripheral

Table 4- Universal Asynchronous Receiver Transmit

Actually Communicating via UART

In principle, UART only requires two signal lines to successfully communicate: a TXD (transmit data) and RXD (receive data) line as well as a common ground line (used as a reference point). When communicating with another UART device, the TXD line will be attached to a corresponding RXD line, and vice-versa. No clock line is used with the UART protocol. Rather, users instead specify a particular baud rate for the two devices to operate at.

A baud rate indicates how many bits, including data, start, stop, and parity bits, are transferred over the data lines in a one second time frame. A common UART communication configuration uses a start bit, 8 data bits, no parity, and a single stop bit.

GPIO – General Purpose Input Output

Pin	Signal	Direction
1	IO1	In/Out
2	IO2	In/Out
3	IO3	In/Out
4	IO4	In/Out
5	GND	
6	VCC	

Table 5- General Purpose Input Output

GPIO

Pmods also communicate with the host board through general purpose input/output (GPIO) pins. This does not follow a strict set of rules regarding how the Pmod and the host should communicate. Rather, the host is able to send high, low, pwm, analog, audio, or a variety of other signal types to the Pmod at any time and have the Pmod (more or less) immediately respond as designed which is great if you want to get a response whenever you press a button or flip a switch. The host could also receive the same variety of signals from the Pmod at any time, but it depends on how the code is designed to have the host respond to or ignore the signals. Also, because we are using input/output pins for communication, we can become rapidly limited in our capability to operate a large amount of peripheral modules.

More detailed information about each protocol can be found at the following location

[https://www.digilentinc.com/Pmods/Digilent-Pmod %20Interface Specification.pdf](https://www.digilentinc.com/Pmods/Digilent-Pmod_%20Interface_Specification.pdf)

Pinout for 40 pin header on GPIO on the DE1-SoC board

JP1		JP2	
Pin 1	D ₀	Pin 1	D ₀
	D ₁		D ₁
D ₂	D ₃	D ₂	D ₃
D ₄	D ₅	D ₄	D ₅
D ₆	D ₇	D ₆	D ₇
Unused		Unused	
D ₈	D ₉	D ₈	D ₉
D ₁₀	D ₁₁	D ₁₀	D ₁₁
D ₁₂	D ₁₃	D ₁₂	D ₁₃
	D ₁₄		D ₁₄
	D ₁₅		D ₁₅
D ₁₆	D ₁₇	D ₁₆	D ₁₇
D ₁₈	D ₁₉	D ₁₈	D ₁₉
D ₂₀	D ₂₁	D ₂₀	D ₂₁
Unused		Unused	
D ₂₂	D ₂₃	D ₂₂	D ₂₃
D ₂₄	D ₂₅	D ₂₄	D ₂₅
D ₂₆	D ₂₇	D ₂₆	D ₂₇
D ₂₈	D ₂₉	D ₂₈	D ₂₉
D ₃₀	D ₃₁	D ₃₀	D ₃₁
	Pin 40		Pin 40

The pinout that was chosen for the 40 pin header was that of the **NIOS** pinout. This way the PMODs can be used for both the **DE1-SoC** board or when the **NIOS** processor is downloaded to the **DE1-SoC**. Pin assignments for this header can be found on the DESL web page

<http://www-ug.eecg.toronto.edu/msl/handouts/40pinNIOS.pdf>

Table 6 gives a breakdown of how each of the pins **D0-D31** are connected. Each PMOD uses 8 signal pins. So this means that 4 PMOD ports can be assigned to each GPIO header on the DE1-SoC development board.

PMOD pin values and locations

PMOD1 to 40 pin header

PIN	Description	Header location	Direction	Pin	Description	Header location	Direction
1	CS	D0 – Y17	output	7	CS	D4 – AK19	output
2	MOSI	D1- Y18	output	8	MOSI	D5 – AJ19	output
3	MISO	D2 – AK16	input	9	MISO	D6 – AJ17	input
4	SCLK	D3 – AK18	output	10	SCLK	D7 – AJ16	output
5	GND			11	GND		
6	VCC			12	VCC		

PMOD2 to 40 pin header

PIN	Description	Header location	Direction	PIN	Description	Header location	Direction
1	CS	D8 – AH18	output	7	CS	D12 – AF16	output
2	MOSI	D9 – AH17	output	8	MOSI	D13 – AG17	output
3	MISO	D10 – AG16	input	9	MISO	D14 – AA19	input
4	SCLK	D11 – AE16	output	10	SCLK	D15 – AC20	output
5	GND			11	GND		
6	VCC			12	VCC		

PMOD3 to 40 pin header

PIN	Description	Header location	Direction	PIN	Description	Header location	Direction
1	CS	D16 – AH19	output	7	CS	D20 – AD19	output
2	MOSI	D17 – AJ20	output	8	MOSI	D21 – AD20	output
3	MISO	D18 – AH20	input	9	MISO	D22 – AE18	input
4	SCLK	D19 – AK21	output	10	SCLK	D23 – AE19	output
5	GND			11	GND		
6	VCC			12	VCC		

PMOD4 to 40 pin header

PIN	Description	Header location	Direction	PIN	Description	Header location	Direction
1	CS	D24 – AF20	output	7	CS	D28 – AF18	output
2	MOSI	D25 – AF21	output	8	MOSI	D29 – AG20	output
3	MISO	D26 – AF19	input	9	MISO	D30 – AG18	input
4	SCLK	D27 – AG21	output	10	SCLK	D31 – AJ21	output
5	GND			11	GND		
6	VCC			12	VCC		

Table 6- PMOD assignments for 40 pin GPIO header on DE1-SoC board

Using different PMOD modules with the DE1-SoC board

PMOD Joy stick and Push button [JSTK]



This PMOD uses the **SPI** protocol to get and retrieve information. The PMOD device is considered the **Slave** device and the DE1-SoC is considered the **Master** device. For detailed information about this PMOD and how it works go to the following link on the **Diligent** website.

<https://reference.digilentinc.com/reference/pmod/pmodjstk/reference-manual>

Information about the PMOD Joystick and button

A total of 5 bytes or 40 bits are sent by the PMOD Joystick. The first 4 bytes contain information about the **X** axis and **Y** axis. The 5th byte contains information about whether **button1** or **button 2** was pressed. Only **10** bits are used for the **X** and **Y** axis. All 8 1st byte and 2 bits 2nd byte for **X** axis and all 8 bits 3rd byte and 2 bits 4th byte and 3 bits for 5th byte. See **figure 5** for further explanation.

The overall order of bytes that will be sent by the Pmod JSTK to the system board will be as follows:

1	2	3	4	5
X (low)	X (high)	Y (low)	Y (high)	Buttons
all 8 used	2 least bits used	all 8 used	2 least bits used	3 least bits used

total of 40 bytes sent 5x8

Figure 5 - PMOD Joystick byte transfer description

The high bytes for both the **X** and **Y** axis determine the direction the Joystick is being moved. Below **table 7** gives all the possible directions.

Movement of Joystick	X Axis - Byte 2		Y axis - Byte 4	
	Bit 1	Bit 0	Bit 1	Bit 0
<i>up</i>	0	1	1	1
<i>down</i>	0	1	0	0
<i>left</i>	0	0	0	1
<i>right</i>	1	1	0	1
<i>Centre</i>	0	1	0	1

Table 7- Joystick movements

Table 8 gives the values of byte 5 bit 2 and bit 1 will determine which button on the PMOD Joystick has been pressed. If any of the buttons are pressed, then a 1 is sent to the DE1-SoC otherwise it stays at zero.

Button Pushed	Bit 2-Byte 5	Bit 1 –Byte 5
Button 1	0	1
Button 2	1	0
Neither	0	0

Table 8 Push button values byte 5

With this information we can now develop **Verilog** code that will move the Lego Jeep around. We will use the following criteria for getting the desired results.

- If Button 1 is pushed the Jeep will move forward.
- If the joystick is moved left the Jeep moves left
- If the joystick is move right the Jeep moves right
- If the joystick is moved down the Jeep will go backwards.
- If button 1 is pushed again the Jeep will stop
- GPIO-0 will be used to connect to the PMOD daughter board
- GPIO-1 will be used to connect to the Lego controller

Below figure 6 and 7 show pictures of how the Lego controller and PMOD daughter board are connected to the DE1-SoC board.

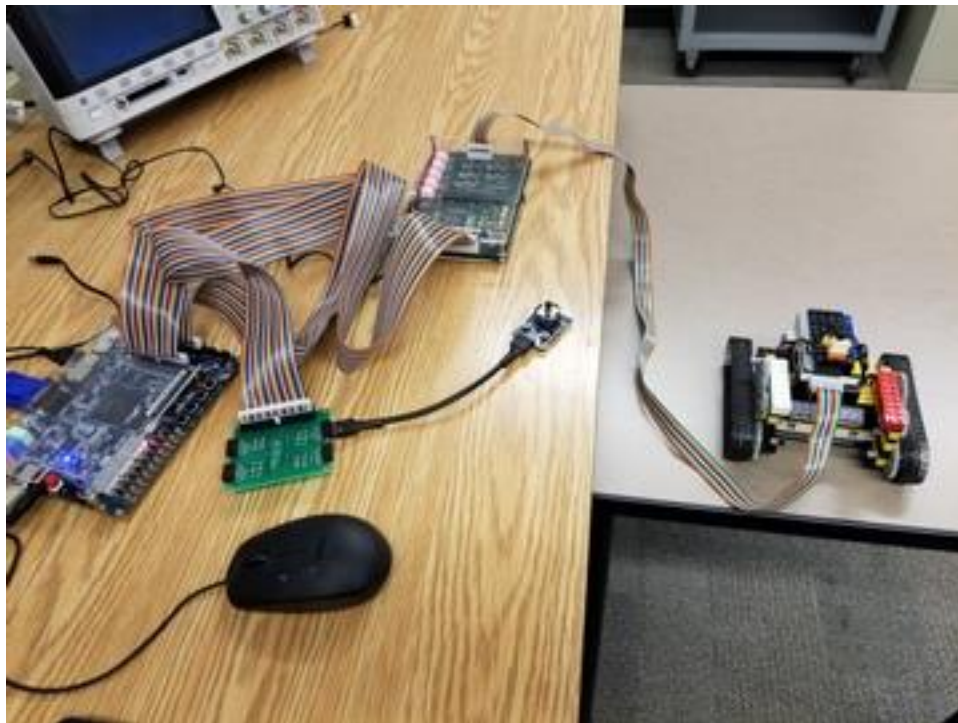


Figure 6- Lego controller and Joystick PMOD connected to GPIO 0 and 1



Figure 7 -closer view PMOD board is connected to **GPIO 0** and Lego controller connected to **GPIO 1**

The Verilog code to perform the task explained in the above bullets can be found on the **DESL** website at the following location;

http://www-ug.eecg.utoronto.ca/desl/peripherals_test_code.html

The file is called **PMOD_jstk.zip**. Download the files and run the program

The Verilog code has been written with a lot of comments to help understand the flow of the code.

Note that the switches in the Verilog code are used to select one of the **8** different PMOD ports. **Table 9** explains how the setting work. In **table 6** above we describe what each port pin represents.

Port Selection	Switch 3 enable	Switch 2	Switch 1	Switch 0
PMOD 1 top 6	1	0	0	0
PMOD 1 bottom 6	1	0	0	1
PMOD 2 top 6	1	0	1	0
PMOD 2 bottom 6	1	0	1	1
PMOD 3 top 6	1	1	0	0
PMOD 3 bottom 6	1	1	0	1
PMOD 4 top 6	1	1	1	0
PMOD 4 bottom 6	1	1	1	1

Table 9- switch setting to enable one of 8 PMOD ports

Below are results showing the timing information for the SPI protocol when each of the directions explained in the bullets (see **page 9**) are executed. These results were generated using the Agilent 3000 series MSO.

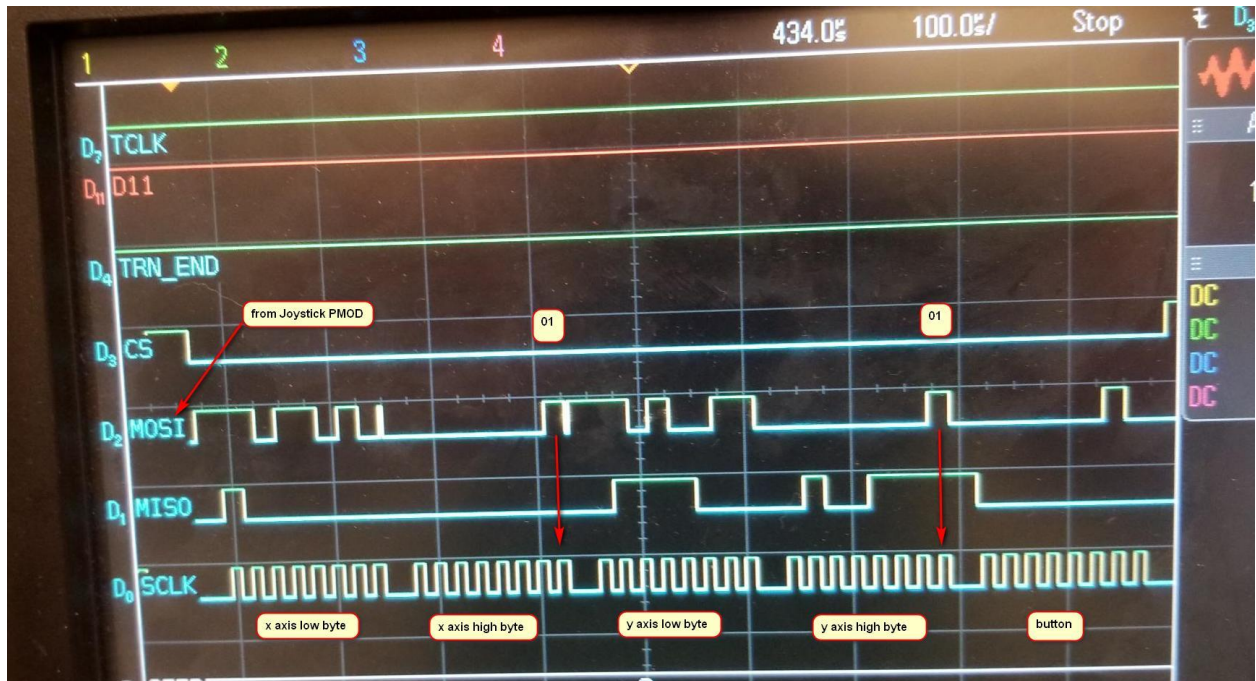


Figure 8- this is when the Jeep is moving forward

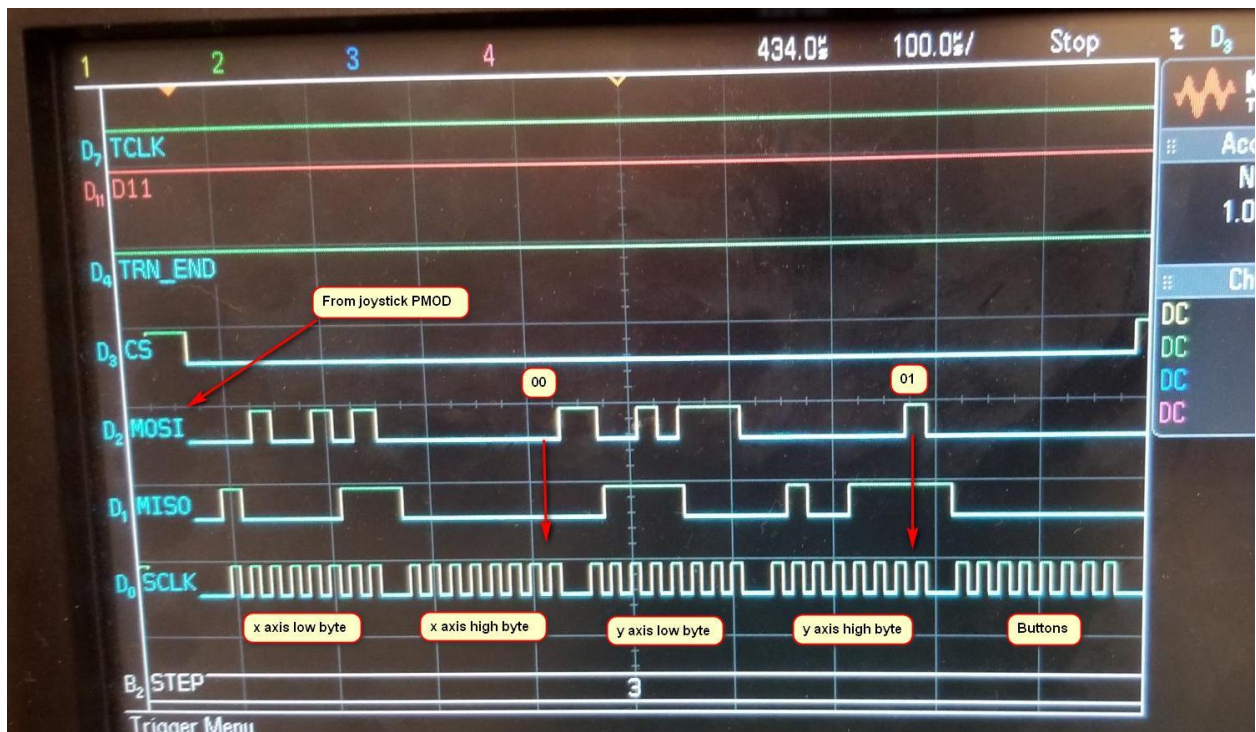


Figure 9- this is when the jeep is moving left

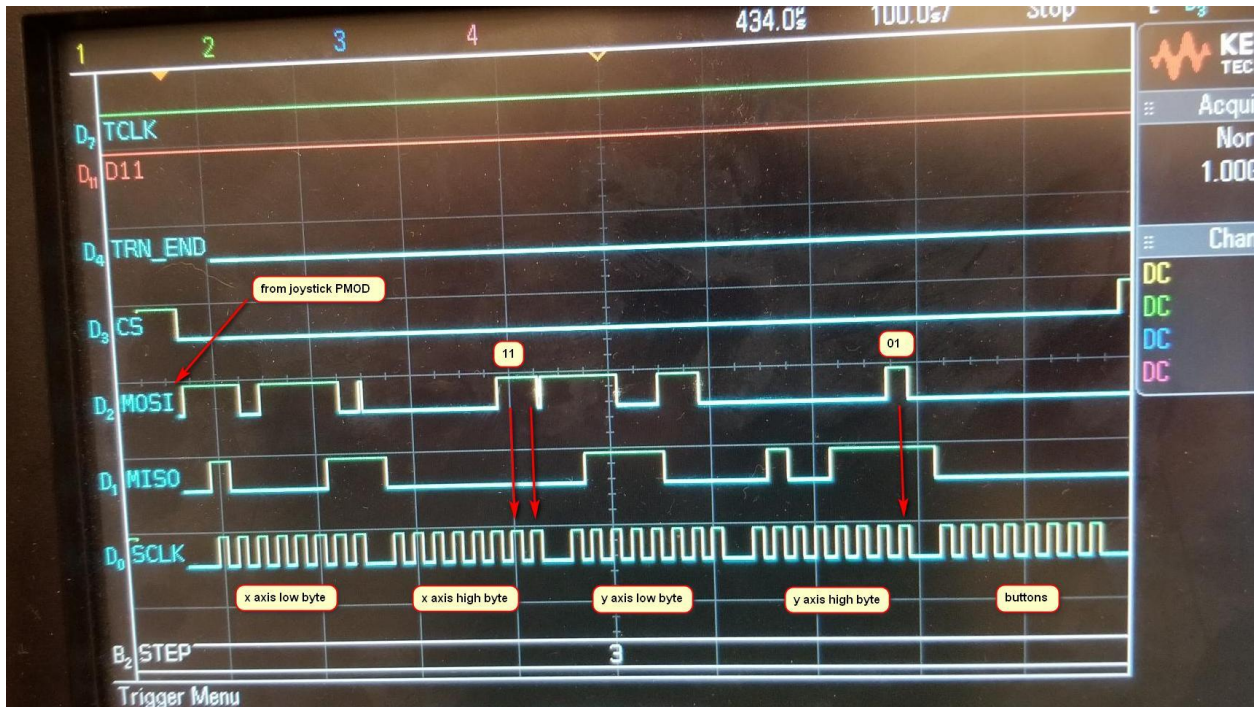


Figure 10- This is when the jeep is moving right



Figure 11- this is when the jeep is going backwards

This concludes the tutorial on the PMOD Joystick JSTK