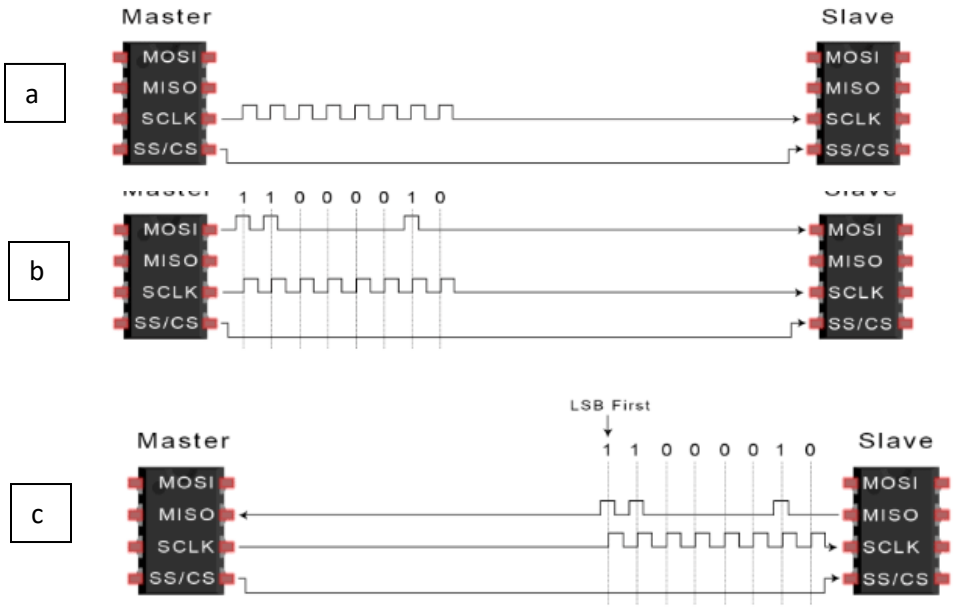
Practical 4: RPI-3B SPI and Interrupt

Student Name & Number: James Hansen – HNSJAM004

Ciaran McKey – MCKDAV12

Question 1:

a) SPI communication protocol works in a synchronous system where both devices (slave and master) share a common clock signal. The clock signal synchronizes the master output data bits to the sampling bits of the slave. One bit of data is transferred per clock cycle therefore the speed of data transfer is determined by the frequency of the clock. SPI communication is initiated by the master as the master configures and generates the clock signal. The clock signal can be modified using “clock polarity” and “clock phase”. These properties determine when the data bits are output and sampled. “Clock polarity” sets whether the bits are output/sampled on rising/falling edges of the clock cycle. “Clock phase” sets whether the bits are output/sampled on the first/second edge of the clock cycle; regardless of rising/falling state. A timing diagram is shown below:
   a. The master outputs a clock signal to the slave via the SCLK channel and sets SS LOW, to enable the slave.
   b. The master outputs data bits, one at a time on the rising edge of the clock cycle from the MOSI channel and is sampled by the slave simultaneously.
   c. If there is a response required, the slave replies with data bits, one at a time on the rising edge of the clock cycle. This can occur while data is being transmitted from the master to slave.



b) Interrupts, in terms of embedded systems, is a signal to the processor emitted by hardware or software that indicates an event that needs immediate attention. When an interrupt occurs, the controller completes the current instruction ad invokes the execution of an ISR or threaded callback (also known as Interrupt Handler). The threaded callback handler is a function which that tells the processor what to do when the interrupt occurs.

c) Function that converts a 10-bit ADC reading from the potentiometer to a 3V3 limited voltage output.

```
def GetData(channel): # channel must be an integer 0-7
    adc = spi.xfer2([1,(8+channel)<<4,0]) # sending 3 bytes
    data = ((adc[1]&3) << 8) + adc[2]
    return data
```

```
# function to convert data to voltage level,
# places: number of decimal places needed
def ConvertVolts(data,places):
    volts = (data * 3.3) / float(1023)
    volts = round(volts,places)
    return volts
```

d) Function that converts a 10-bit ADC reading from the temperature sensor to a reading in degree Celsius

```
def Temperature (voltage):
    temp = voltage
    temp = int ((temp - 0.5)/0.01 )
    return temp
```

e) Function that converts a 10-bit ADC reading from the LDR to a percentage representing the amount of light received by the LDR

```
def Percent (voltage):
    percentage = (int (voltage/3.1*100))
    return percentage
```

f) Flowchart of System