Interfacing ADC with R-PI

e-Yantra Team

June 10, 2016

Contents

1 Objective

The objective is to Interface the ADC MCP3008 to Raspberry Pi with a Sharp IR Sensor as Raspberry Pi has no built in analogue inputs.Then the RPi is interfaced with 2 MC3008 ADC's where 2 Sharp IR sensors are connected.

2 Prerequisites

One should have:

- To know how to access the pins of an R-Pi.
- Some basic information regarding SPI protocol.
- Python programming skills.

3 Hardware Requirement

1. Raspberry Pi (I will be using Version 2 Model B)
2. Power adapter(5V)
3. MCP3008
4. Sharp IR Sensor
5. LM35
6. Connecting wires
7. Bread board

4 Software Requirement

1. MobaXterm (for windows user)
2. PyScripter (version 2.7 or above)

5 Theory and Description

5.1 MCP 3008

The MCP3008 is a successive approximation 10bit 8-channel Analogue-todigital converter (ADC). It is cheap, easy to connect and doesnt require any additional components.It uses the SPI bus protocol which is supported by the Pis GPIO header.

- 10 bit resolution.
- 8 input channels.
- Analog inputs programmable as single-ended or pseudo-di?erential pairs.
- SPI serial interface (modes 0,0 and 1,1).
- Single supply operation: 2.7V - 5.5V.

5.2 Pin Description:

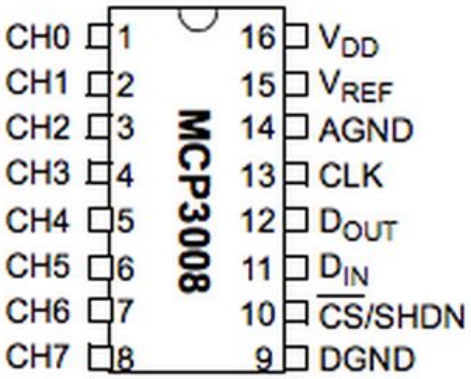

Figure 1: MCP3008

MCP3008 is a 16 pin IC.

1. Digital Ground (DGND):
   Digital ground connection to internal digital circuitry.
2. Analog Ground (AGND):
   Analog ground connection to internal analog circuitry.
3. Analog inputs (CH0 - CH7):
   Analog inputs for channels 0 - 7, respectively, for the multiplexed inputs. Each pair of channels can be programmed to be used as two independent channels in single-ended mode or as a single pseudo differential input where one channel is IN+ and one channel is IN.
4. Serial Clock (CLK):
   The SPI clock pin is used to initiate a conversion and clock out each bit of the conversion as it takes place.

5. Serial Data Input (DIN):
   The SPI port serial data input pin is used to load channel configuration data into the device.

6. Serial Data Output (DOUT):
   The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

7. Chip Select/Shutdown (CS/SHDN):
   The CS/SHDN pin is used to initiate communication with the device when pulled low. When pulled high, it will end a conversion and put the device in low-power standby. The CS/SHDN pin must be pulled high between conversions.

5.3 Serial Communication:

Communication with the 3008 devices is accomplished using a standard SPIcompatible serial interface.

- Initiating communication with either device is done by bringing the CS line low .If the device was powered up with the CS pin low, it must be brought high and back low to initiate communication.
- The first clock received with CS low and DIN high will constitute a start bit.
- The SGL/DIFF bit follows the start bit and will determine if the conversion will be done using single-ended or differential input mode.
- The next three bits (D0, D1 and D2) are used to select the input channel configuration. Table shows the bit configuration.
- The device will begin to sample the analog input on the fourth rising edge of the clock after the start bit has been received. The sample period will end on the falling edge of the fifth clock following the start bit.
- On the falling edge of the 5th clock, the device will output a low null bit. The next 10 clocks will output the result of the conversion with MSB first.

5.4 Using the MCP 3008 with the microcontroller SPI ports:

- With microcontroller SPI ports, it is required to send groups of eight bits.

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+

Figure 2: Table for bit configuration

- Communication with the 3008 devices may not need multiples of eight clocks, it will be necessary to provide more clocks than are required. This is usually done by sending leading zeros before the start bit.
- As is shown in Figure , the first byte transmitted to the A/D converter contains seven leading zeros before the start bit.
- Arranging the leading zeros this way induces the 10 data bits to fall in positions easily manipulated by the MCU.

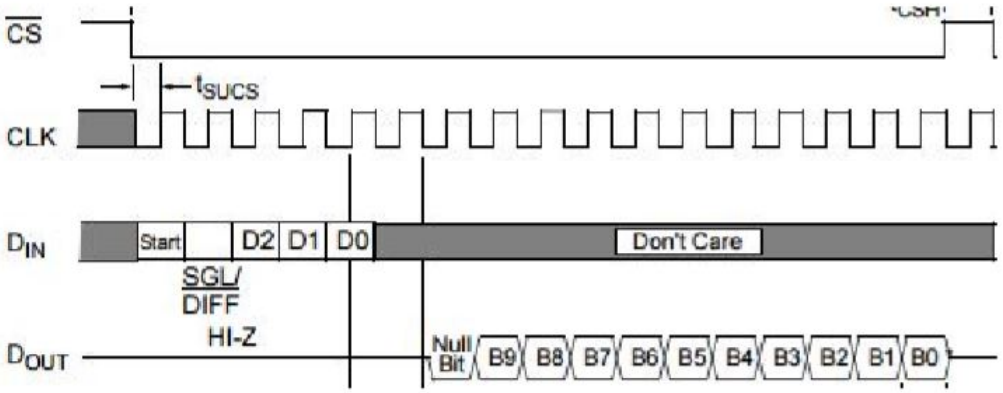

Figure 3: Communication with MCP 3008

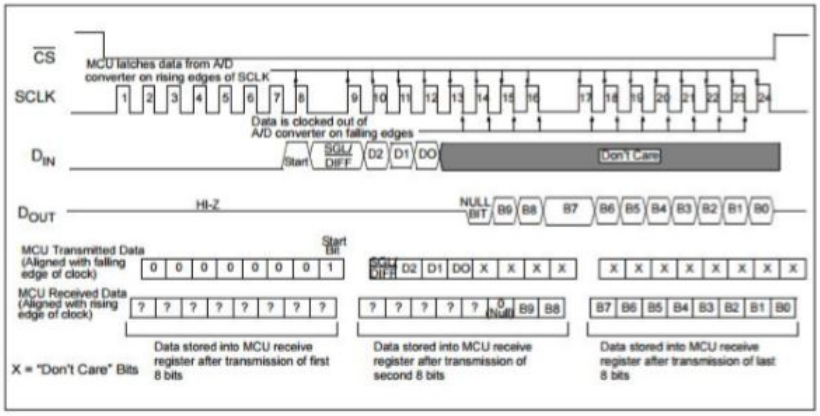

Figure 4: SPI Communication with MCP 3008 using 8-bit segments

6 Experiment

6.1 Interfacing an ADC with RPi using LM35(Temperature Sensor)

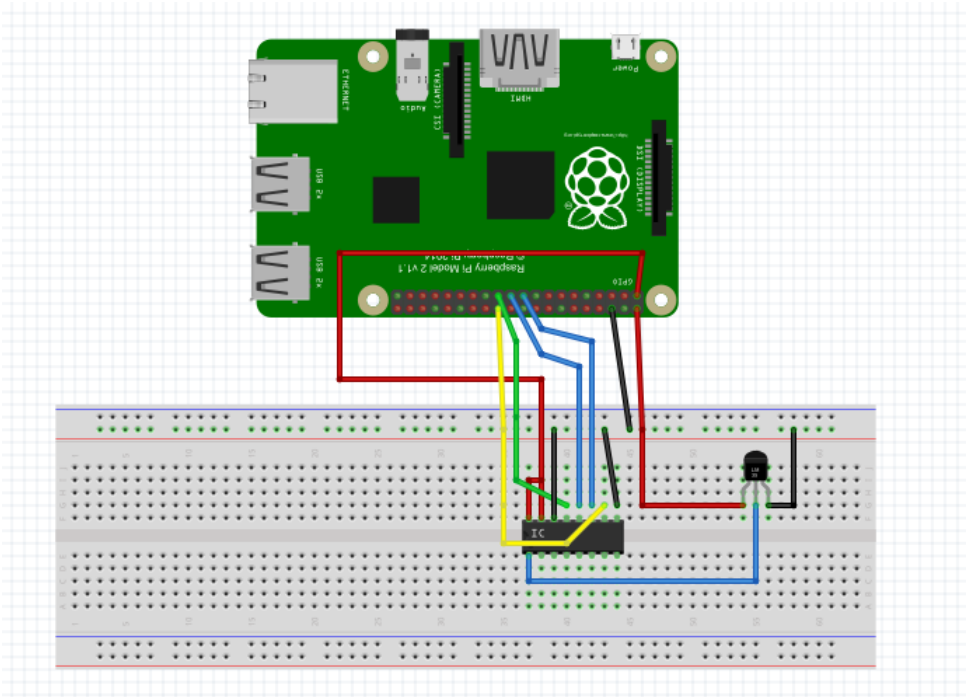

Figure 5: Circuit on Bread Board

- The pin connections are shown in the Circuit above.
- Here I have used a single ADC MCP3008 interfaced with RPi to which a Lm35 is connected.

Code

```
  import spidev
# module to control spi devices
import time
import math
import sys
# Open SPI bus
spi = spidev.SpiDev () # to create spi object
spi.open(0 ,0) #Clock polarity , Clock Phase
# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0  7
# Function name : ReadChannel
# Input : channel
# Output : data
# Example call : Readadc( channel )
def readadc(adcnum):
```

```
# read SPI data from MCP3008 chip , 8 possible adc    s (0 thru 7)
    if adcnum > 7 or adcnum < 0:
        return -1
    r = spi . xfer2 ([1  , 8 + adcnum << 4  , 0])
    adcout = (( r [1] & 3) << 8) + r [2]
    return adcout
# Function to calculate temperature from
# LM35 data , rounded to specified
# number of decimal places .
def ConvertTemp(data,places):
    temp = data*100
    temp = round(temp,places)
    return temp
try:
    while True:
        value = readadc(0)
        volts = ( value * 3.3) / 1024
        temperature = ConvertTemp( volts ,2)
        #Print out results
        print("Temp : {} volts : {} deg C: {}".format(
            value,volts,temperature ))
        time . sleep (0.5)
except KeyboardInterrupt:
    pass
spi.close()
sys.exit(0)
```

6.2 Interfacing an ADC with RPi using a Sharp IR Sensor

- The pin connections are shown in the Circuit above.
- The Connections are showed in the Circuit Schematic in the following figure.
- Here I have used a single ADC MCP3008 interfaced with RPi to which a Sharp IR Sensor is connected.

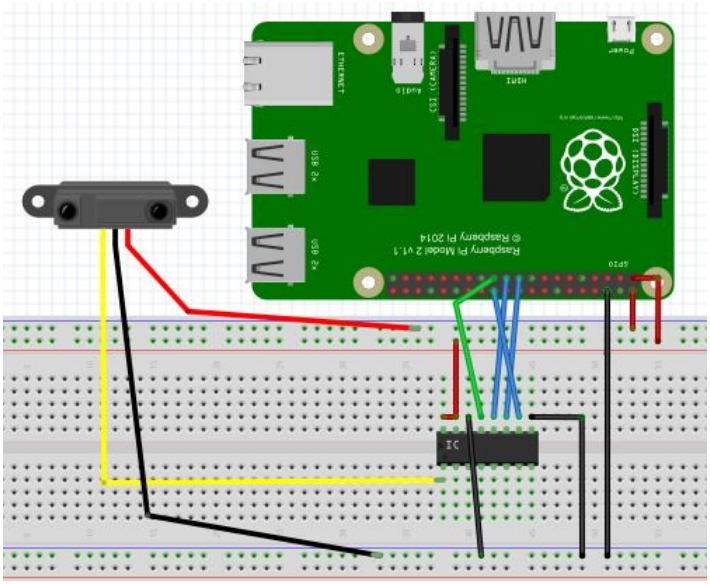

Figure 6: Circuit on Bread Board

6.3 Circuit Schematic

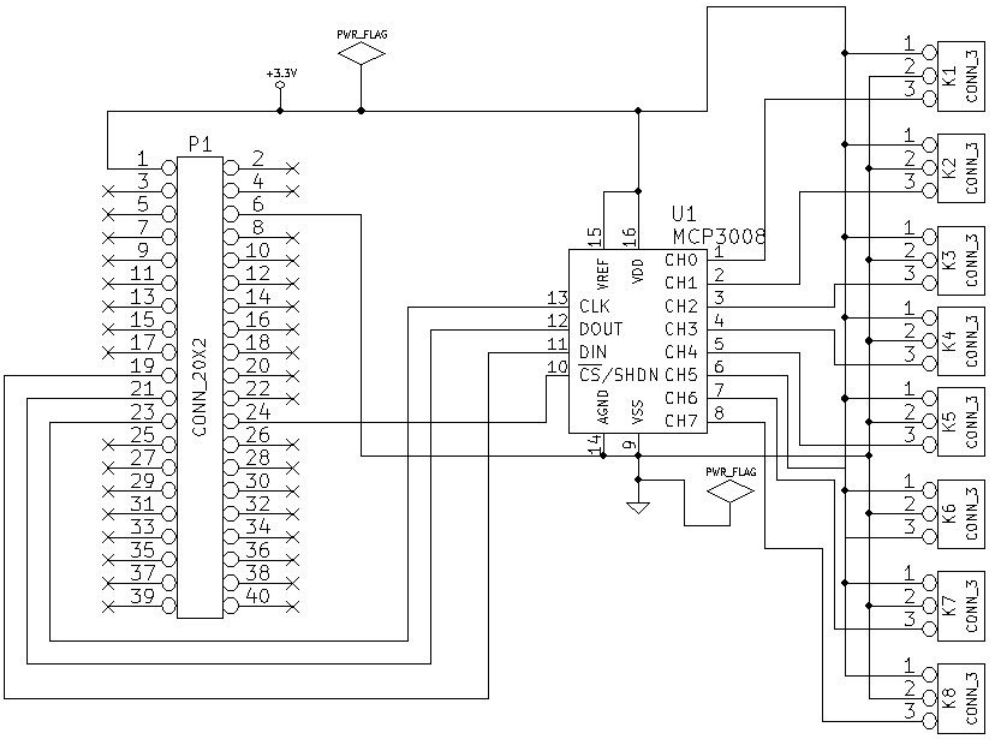

Figure 7: Circuit Schematic

Code

```
import os
import spidev # module to control spi devices
import time

# Open SPI bus
spi = spidev.SpiDev()# to create spi object
spi.open(0,0)#Clock polarity, Clock Phase

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
# Function name :ReadChannel
# Input : channel
# Output : data
# Example call: ReadChannel(channel)
def ReadChannel(channel):
        #Performs SPI Transaction and CS will be held active
        adc = spi.xfer2([1,(8+channel)<<4,0])
```

```
        data1 = ((adc[2]& 0xFC) >> 2)
        data = ((adc[1]&3) << 6) + data1

        return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
# Function name :ConvertVolts
# Input : data,places
# Output : volts
# Example call: ConvertVolts(data,places)
def ConvertVolts(data,places):
        volts = (data * 3.3) / float(1023)
        volts = round(volts,places)
        return volts

# This Function calculates the actual distance in millimeters(mm)
# from the input
# Function name :Sharp_GP2D12_estimation
# Input : adc_reading
# Output : distanceInt
# Example call: Sharp_GP2D12_estimation(adc_reading)
def Shar_sensor_estimation(adc_reading):
       distance =(10.00*(2799.6*(1.00/(pow(adc_reading,1.1546)))))
       distanceInt = distance
       if distanceInt>800:
               distanceInt=800
       return distanceInt

# Define sensor channels
sharp_channel =0

# Define delay between readings
delay =0.5

try:
        while True:
                # Read the sharp sensor data
                sharp_level = ReadChannel(sharp_channel)
                sharp_volts = ConvertVolts(sharp_level,3)
                value = Sharp_GP2D12_estimation(sharp_level)
                # Print out results
                print "--------------------------------------------------"
                print("sharp: Digital {} (Analog {}V)".format(sharp_level
```

```
                                                    sharp_volts))
                print("sharp: (Distance {} mm)".format(value))
                # Wait before repeating loop
                time.sleep(delay)
except KeyboardInterrupt:
        pass
```

7 Exercise

7.1 Interfacing 2 ADC's with RPi using Sharp IR Sensors

- The pin connections are shown in the Circuit Schematic.
- Here I have used 2 MCP3008 ADC's interfaced with RPi.
- In the this experiment I have used Sharp IR Sensor.
- 2 ADC's can be interfaced with RPi mainly because of the 2 chip select pins CE0 and CE1 in RPi.

7.2 Circuit Schematic

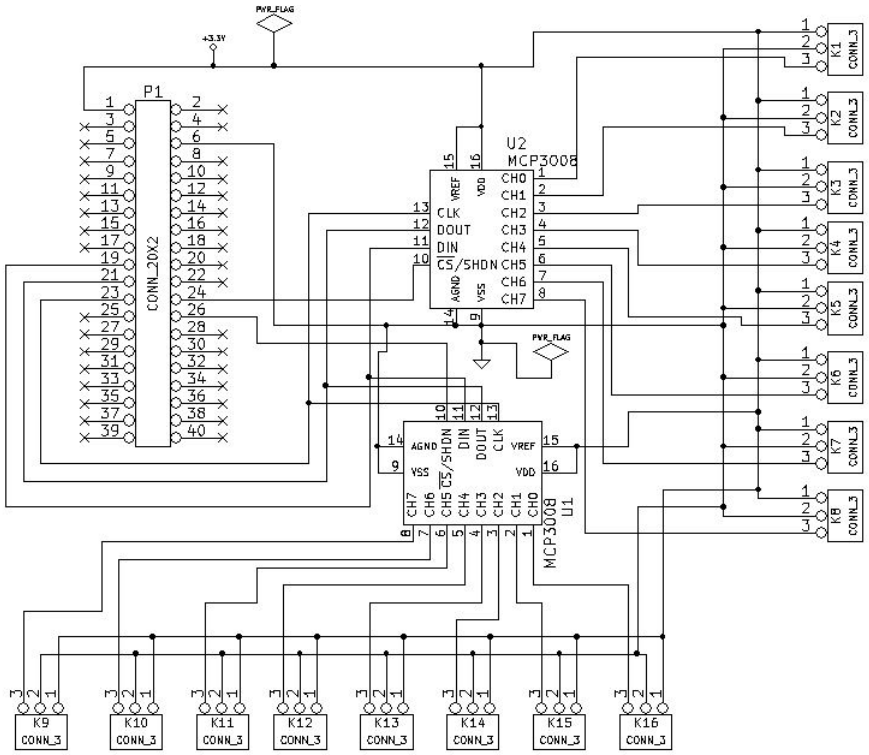

Figure 8: Circuit Schematic

Code

```
import os
import spidev # module to control spi devices
import time
import RPi .GPIO as GPIO # module to control Pi GPIO channels

# Open SPI bus
spi = spidev.SpiDev()# to create spi object
spi.open(0,0)#Clock polarity, Clock Phase

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
# Function name :ReadChannel
# Input : channel
# Output : data
# Example call: ReadChannel(channel)
def ReadChannel(channel):
        #Performs SPI Transaction and CS will be held active
        adc = spi.xfer2([1,(8+channel)<<4,0])
        data1 = ((adc[2]& 0xFC) >> 2)
        data = ((adc[1]&3) << 6) + data1
        return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
# Function name :ConvertVolts
# Input : data,places
# Output : volts
# Example call: ConvertVolts(data,places)
def ConvertVolts(data,places):
        volts = (data * 3.3) / float(1023)
        volts = round(volts,places)
        return volts

# This Function calculates the actual distance in millimeters(mm)
# from the input
# Function name :Sharp_GP2D12_estimation
# Input : adc_reading
# Output : distanceInt
# Example call: Sharp_GP2D12_estimation(adc_reading)
def Sharp_GP2D12_estimation(adc_reading):
       distance =(10.00*(2799.6*(1.00/(pow(adc_reading,1.1546)))))
       distanceInt = distance
```

```
        if distanceInt>800:
        distanceInt=800
        return distanceInt

# Define sensor channels
sharp_channel =0
white_channel =1

# Define delay between readings
delay =0.5

try:
while True:
        GPIO.setmode(GPIO.BOARD)# set up GPIO output channel
        GPIO.setup(24, GPIO.OUT)# set up CE0 Pin
        GPIO.setup(26, GPIO.OUT)# set up CE1 Pin

        GPIO.output(24,GPIO.LOW)# Enable CE0 Pin
        GPIO.output(26,GPIO.HIGH)# Disable CE1 Pin


        # Read the sharp sensor data
        sharp_level = ReadChannel(sharp_channel)
        sharp_volts = ConvertVolts(sharp_level,3)
        value = Sharp_GP2D12_estimation(sharp_level)

        GPIO.output(26,GPIO.LOW)# Enable CE1 Pin
        GPIO.output(24,GPIO.HIGH)# Disable CE0 Pin

        # Read the white line sensor data
        white_level=ReadChannel(white_channel)
        white_volts = ConvertVolts(white_level,3)

        # Print out results
        print "--------------------------------------------"
        print("sharp: D{} (A{}V)".format(sharp_level,sharp_volts))
        print("sharp: (Distance {} cm)".format(value))

        print "--------------------------------------------"
        print("white: D{} (A{}V)".format(white_level,white_volts))

        # Wait before repeating loop
        time.sleep(delay)
        except KeyboardInterrupt:
```

pass

8 References

1. www.raspberrypi-spy.co.ukanalogue-sensors-on-the-raspberry-pi-using-an-mcp3008/
2. www.en.wikipedia.org/wiki/SerialPeripheralInterface
3. www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/
4. developers.google.com/edu/python/
5. www.alldatasheet.com/Mcp3008+datasheet