Electronics
MCP 3008 Communication

Terry Sturtevant

Wilfrid Laurier University

January 30, 2019
The Python examples of code to communicate with the MCP3008 can be confusing. Looking at the code and the appropriate timing diagram helps.
MCP Communication

- The Python examples of code to communicate with the MCP3008 can be confusing.
The Python examples of code to communicate with the MCP3008 can be confusing.

Looking at the code and the appropriate timing diagram helps.
adc = spi.xfer2([1, (8 + adc_channel) << 4, 0])

This transfers 3 bytes; the first is simply '1', and the last is simply '0'.
\texttt{adc}=\texttt{spi.xfer2}(\lfloor1, (8+\texttt{adc\_channel})\ll4,0\rfloor)\\
This transfers 3 bytes;
adc=spi.xfer2([1,(8+adc_channel)<<4,0])

This transfers 3 bytes; the first is simply '1', and the last is simply '0'
\((8 + \text{adc\_channel}) \ll 4\)
(8 + adc\_channel)\ll4

The channel is a 3 bit value, so adding 8 gives a single hex digit;
\[(8 + \text{adc\_channel}) \ll 4\]

The channel is a 3 bit value, so adding 8 gives a single hex digit; this value then is shifted 4 bits to the left.
adc = spi.xfer2([1, (8 + adc_channel) << 4, 0])

First byte (first non-zero bit is start)
adc = spi.xfer2 ([1, (8 + adc_channel) <<< 4, 0])

Second byte (1st bit indicates single or double; next 3 indicate channel)
adc = spi.xfer2 ([1, (8 + adc_channel) << 4, 0])

Third byte (don’t care)
\[ \text{data} = ((\text{adc}[1]\&3)<<8) + \text{adc}[2] \]
data = ((adc[1] & 3) << 8) + adc[2]

The response is also a 3 byte transfer;
\[ \text{data} = ((\text{adc}[1] \& 3) << 8) + \text{adc}[2] \]

The response is also a 3 byte transfer; the first byte will be '0', and the last two contain data.
\((\text{adc}[1] \& 3) \ll 8\)
\( (\text{adc}[1] \& 3) << 8 \)

Anding with 3 gets the bottom two bits of the second byte;
(adc[1] & 3) << 8

Anding with 3 gets the bottom two bits of the second byte; it is then shifted left 8 bits so it will be bits 8 and 9
\[ \text{data} = ((\text{adc}[1]\&3) << 8) + \text{adc}[2] \]

First byte (adc[0]) (discard)
\[ \text{data} = ((\text{adc}[1] \& 3) \ll 8) + \text{adc}[2] \]

Second byte (adc[1]) (last two bits matter)
\[
data = ((\text{adc}[1] \& 3) << 8) + \text{adc}[2]
\]

Third byte (adc[2]) (bottom 8 bits of 10 bit value)