**Experiment 1: Does the ADS7825 sample on the rising or falling edge of /BUSY?**

**Method:**

1. Vary the input voltage such that it has different values at the falling and rising edges of /BUSY
2. Initiate conversion
3. Compute the voltage equivalent of the ADC output code
4. Compare voltage equivalent of code to the input voltage values at the rising and falling edges of /BUSY

**Test conditions:**

[A0, A1] = [0, 0]

SPI conditions:

/CS = “0” (shorted to GND)

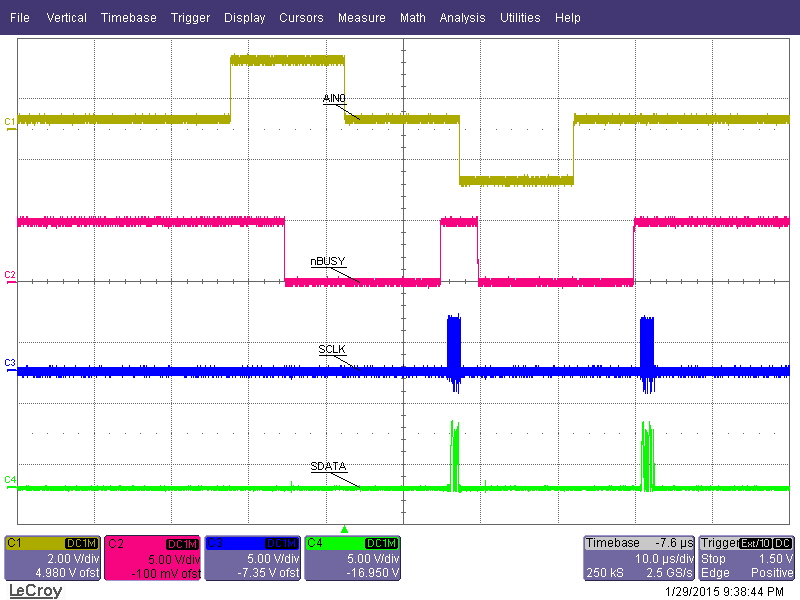
R/nC width = ~2us

f\_SCLK = 5MHz

CPOL = 0, CPHA = 0 🡺 capture (MISO) on rising edge of SCLK

Delay first SCLK rising edge by 21us from falling edge of R/nC

Generate 18 SCLKs and capture MISO starting on 3rd rising edge of SCLK



**Conversion #2**

**Conversion #1**

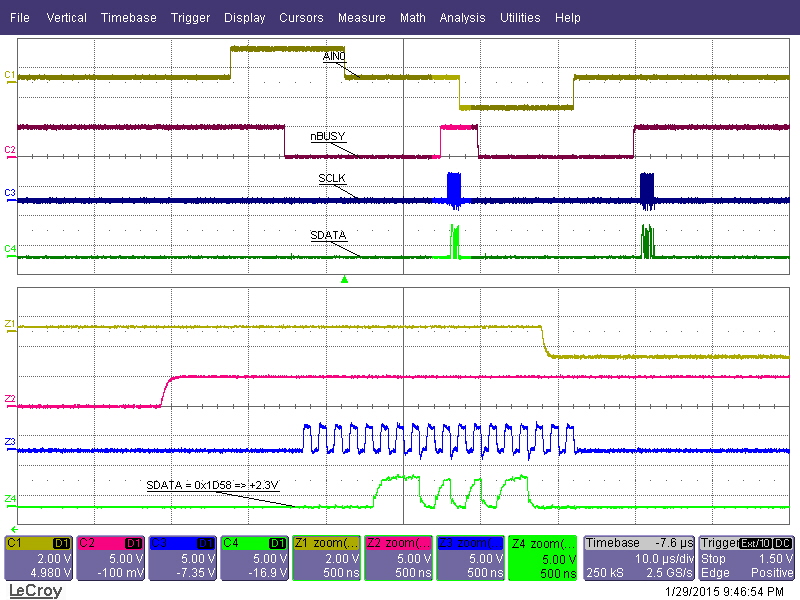
**-1.7V**

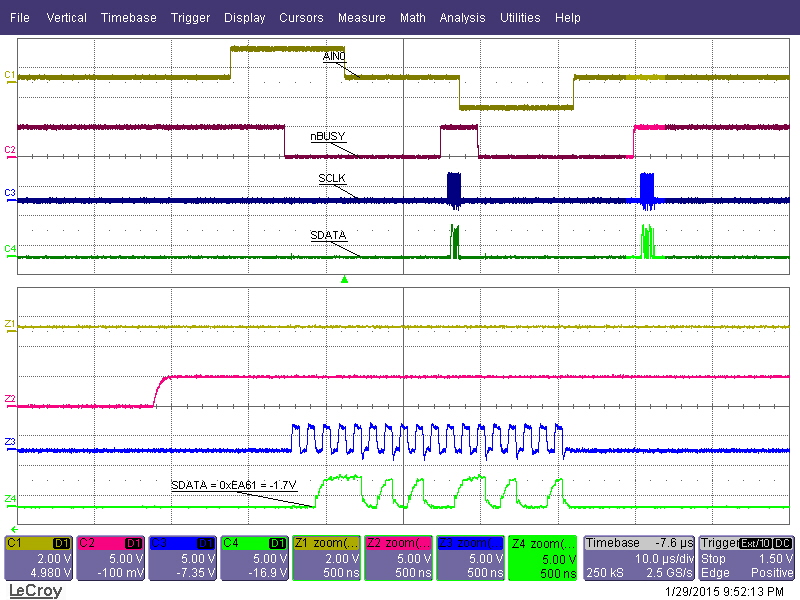
**+2.3V**

**300mV**

**300mV**

**300mV**

Conversion #1 close-up:

Conversion #2 close-up:

**Conclusion:**

The results of Conversion #1 and Conversion #2 correspond to input voltage values that occurred at the FALLING EDGES of /BUSY. Therefore, the ADS7825 samples the input voltage at the falling edge of /BUSY.

**Experiment #2: To which input channel does the output data belong?**

**Method:**

1. Apply different voltages to the input channels
2. Set input channel address [A0, A1] to desired value
3. Initiate conversion and compare voltage equivalent of ADC output code to input voltages
4. Repeat steps (1) to (3) multiple times

**Analog Input conditions:**

AIN0 = 5V (shorted to power supply)

AIN1 = Hi-Z (1.79V when AIN1 selected, ~0V otherwise)

AIN2 = 2.5V (shorted to CAP pin)

**SPI conditions:**

/CS = “0” (shorted to GND)

R/nC width = ~2us

f\_SCLK = 5MHz

CPOL = 0, CPHA = 0 🡺 capture (MISO) on rising edge of SCLK

Delay first SCLK rising edge by 21us from falling edge of R/nC

Generate 18 SCLKs and capture MISO starting on 3rd rising edge of SCLK

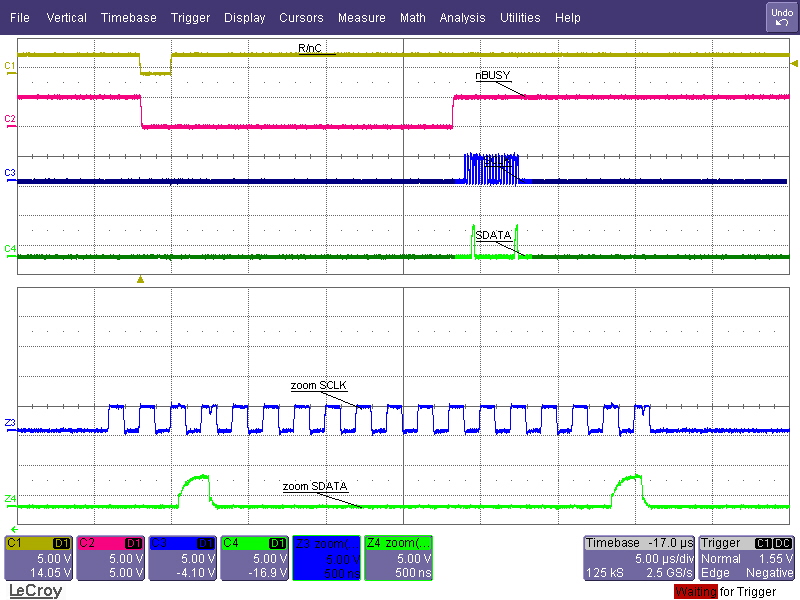
**Expected code out per channel:**

Code(AIN0) = 16384 = 0x4000

Code(AIN1) = 5865 = 0x16E9

Code(AIN2) = 8192 = 0x2000

Timing diagram below shows typical conversion cycle; ADC powered up with [A0, A1] = [0, 0], AIN0 = 5V; code out = 0x4001



**Latch conversion output data; Start acquiring (or “tracking”) next CH**

**Take Input sample; Latch next CH**

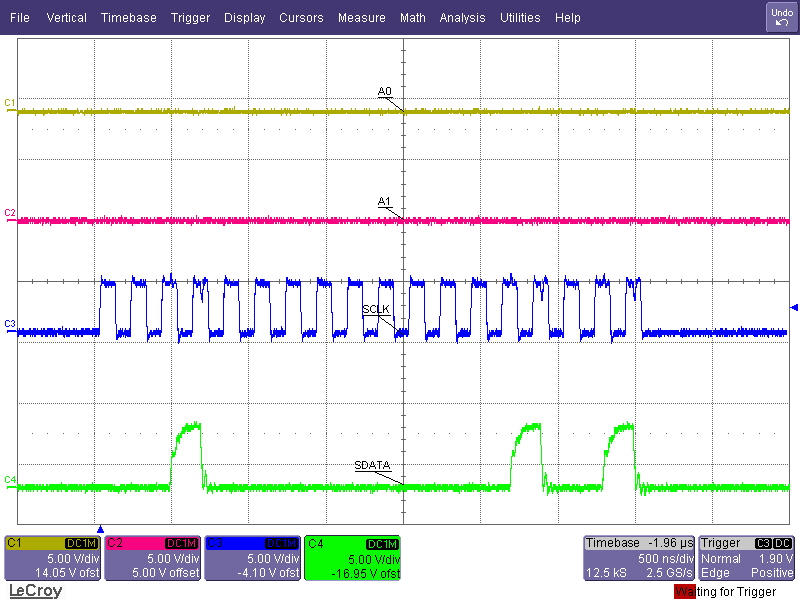
**Convert Input Sample**

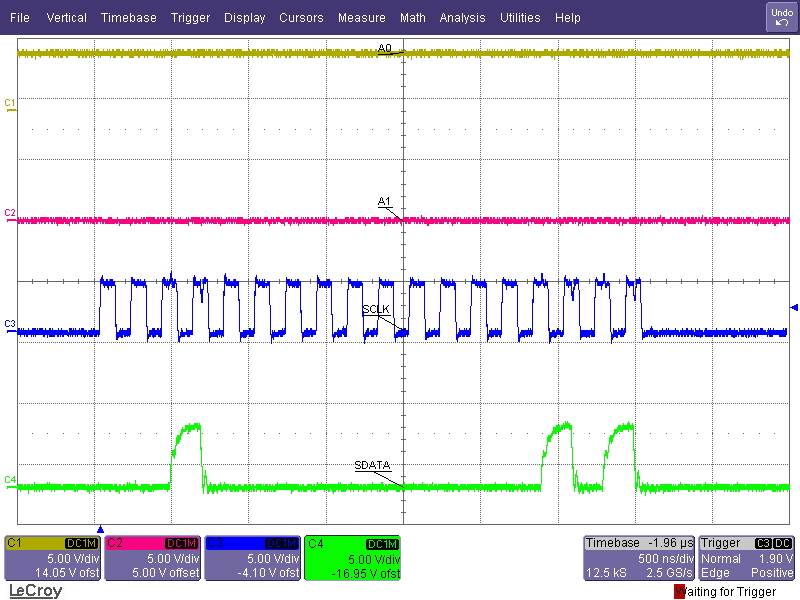
**Delay to next convert signal >> 25us (very long sampling time)**

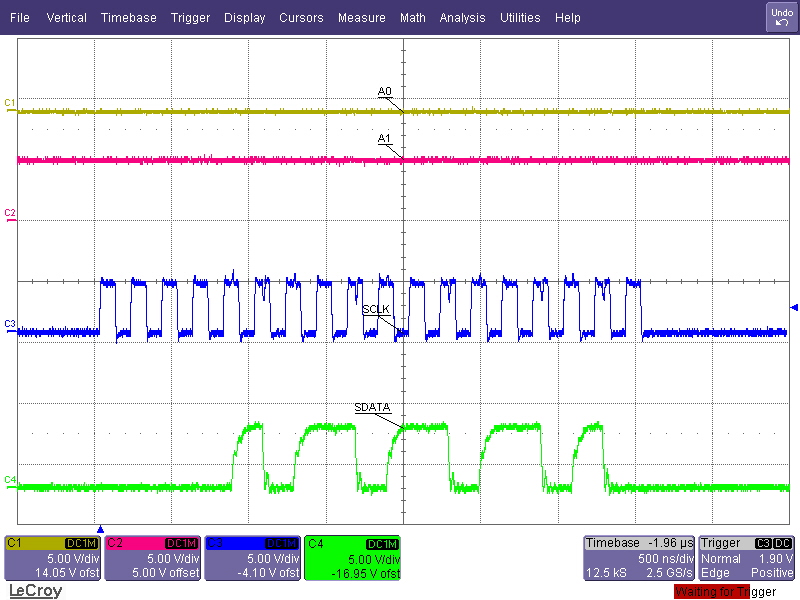
**MSB**

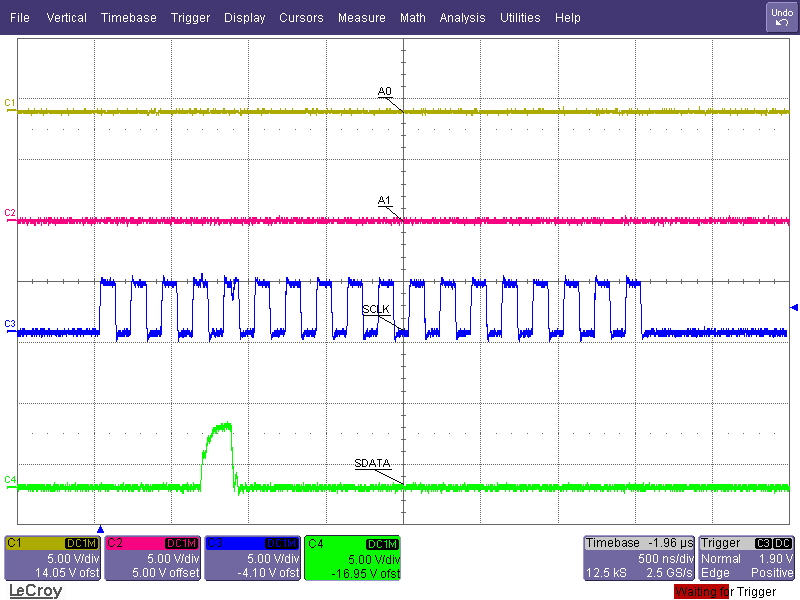
**LSB**

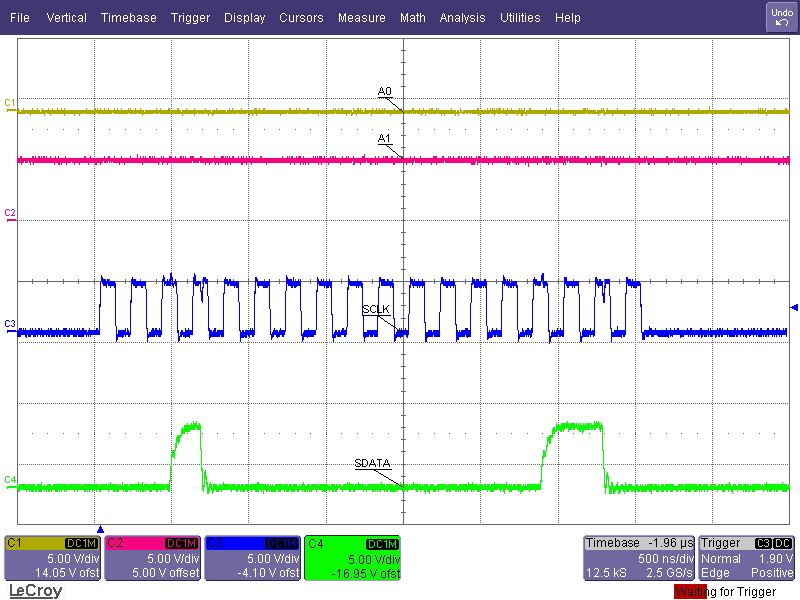
Convert with no change to [A0, A1] 🡺 expected code = 0x4000, actual code = 0x4009; Next channel to convert = AIN0 (expect 0x4000 next)

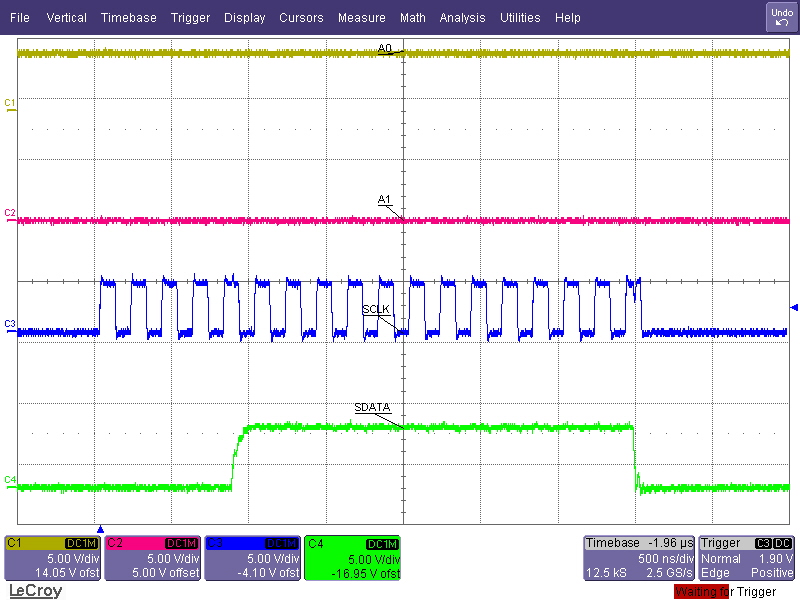


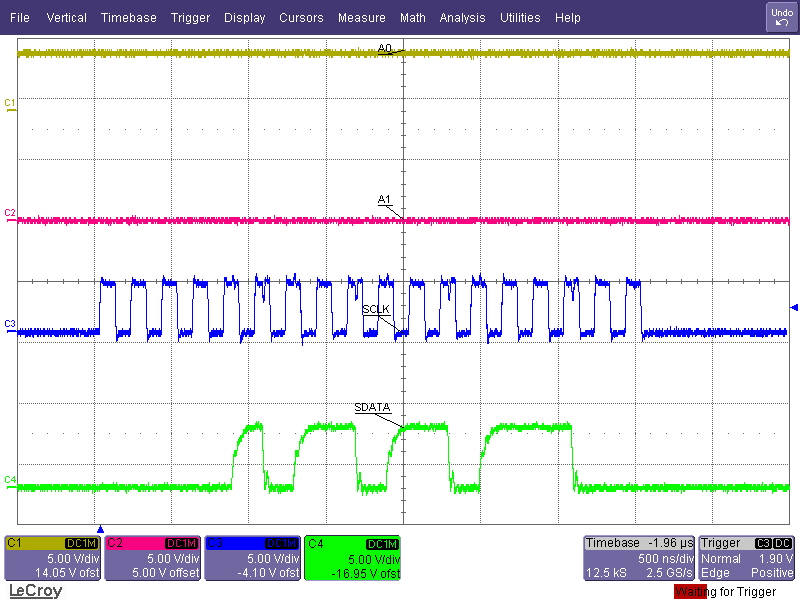
Change [A0, A1] to [1, 0] and convert 🡺 expected code = 0x4000, actual code = 0x4005; Next channel to convert = AIN1 (expect 0x16E9 next)

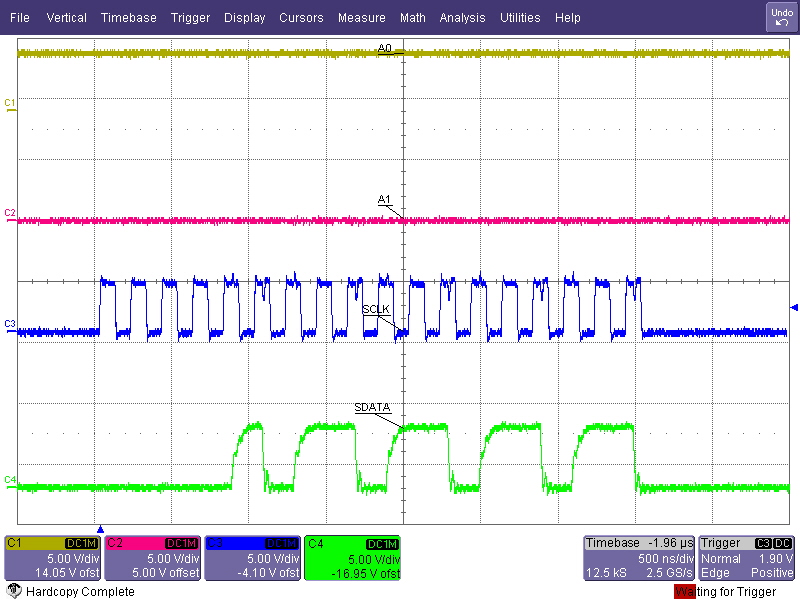
Change [A0, A1] to [0, 1] and convert 🡺 expected code = 0x16E9, actual code = 0x16DA; Next channel to convert = AIN2 (expect 0x2000 next)

Change [A0, A1] to [0, 0] and convert 🡺 expected code = 0x2000, actual code = 0x2000; Next channel to convert = AIN0 (expect 0x4000 next)

Change [A0, A1] to [0, 1] and convert 🡺 expected code = 0x4000, actual code = 0x4006; Next channel to convert = AIN2 (expect 0x2000 next)

Change [A0, A1] to [1, 0] and convert 🡺 expected code = 0x2000, actual code = 0x1FFF; Next channel to convert = AIN1 (expect 0x16E9 next)

No change to [A0, A1] and convert 🡺 expected code = 0x16E9, actual code = 0x16DC; Next channel to convert = AIN1 (expect 0x16E9 next)

No change to [A0, A1] and convert 🡺 expected code = 0x16E9, actual code = 0x16DB; Next channel to convert = AIN1 (expect 0x16E9 next)

**Conclusion:**

The ADS7825 datasheet states that the output data captured in a given cycle corresponds to the input channel whose address was latched on the previous falling edge of /BUSY. We have verified that this statement is correct.