BURR - BROWN®
APPLICATION BULLETIN
Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706
Tel: (520) 746-1111 • Telex: 066-6491 • FAX (520) 889-1510 • Product Info: (800) 548-6132 • Internet: www.burr-brown.com/

Improved 60Hz Performance for ADS1211

By Russell Anderson

When multiplexing the input of the ADS1211, the data output will not have valid data using the new input until the third or fourth conversion. This is due to filter settling time. If the channel could change synchronously with the $\overline{\text{DRDY}}$ pulse, you would only have to discard two conversions. If the change occurs during the interval between $\overline{\text{DRDY}}$ pulses, three outputs will be discarded, since they contain samples from both the previous input and the new input. We will assume for this example that the inputs are changed asynchronously, and therefore, we have to discard three outputs each time we change the inputs.

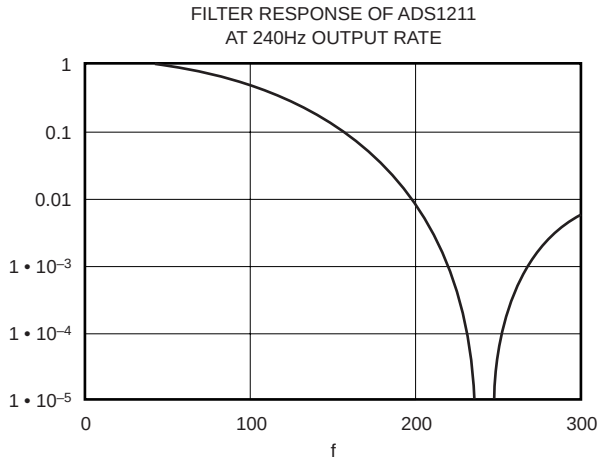
The desired output rate is 60Hz. If after each valid sample we changed the input, we would have to discard three out of every four samples, and we would get a throughput of 25%. If we increased our sample rate to 240Hz and averaged four adjacent samples, we would still have to discard three outputs when we changed the inputs. However, now the throughput would be 4/7 meaning that we would have valid data for 57% of the $\overline{\text{DRDY}}$ pulses. If we were synchronized or used the DSYNC bit, we could achieve 4/6 or 66% throughput. The valid data output rate is more than doubled and the result would be the average of four conversions. This 4-point average would decrease our noise by a factor of 2 ($\sqrt{4} = 2$), which would double the effective resolution.

The other benefit of having the ADS1211 output rate at 240Hz is a higher 3dB frequency of 24.6Hz. If we used an output rate of 60Hz, the 3dB frequency would be 15.7Hz. There are actually two filters that together give us an overall frequency response. First we have the frequency response of the ADS1211, and then we have the frequency response of the averaging process. The ADS1211 is a third-order sin(x)/x filter and the 4-point averaging is just a sin(x)/x filter. However, they don't have the sample notch frequencies. The notch for the ADS1211 frequency response is at 240Hz and the frequency notch for the averaging is at 60Hz. The 3dB values for the two filters by themselves are 62.8Hz (0.262 • 240) and 24.9Hz (0.416 • 60). The combination of the two filters gives a3dB frequency of 24.6Hz.

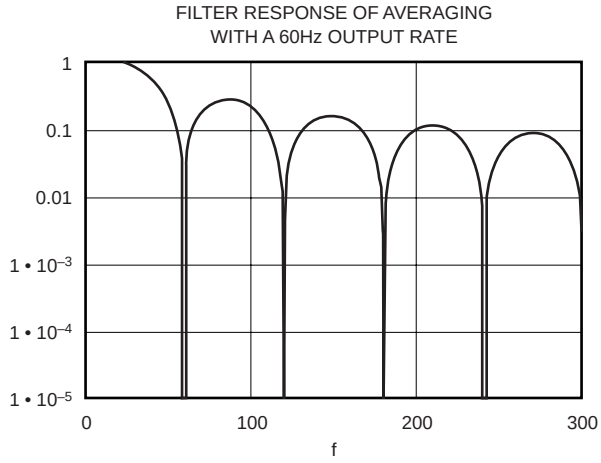
However, using 240Hz does have a drawback. Turbo mode 16 gives the highest number of effective bits. Using a 10MHz clock, at 60Hz the effective bits are 23 bits, and at 240Hz they are 21.5 bits, as shown on page 7 of the data sheet. Using the averaging does give us another bit, so we end up with 22.5 effective bits.

Printed in U.S.A. January, 2000

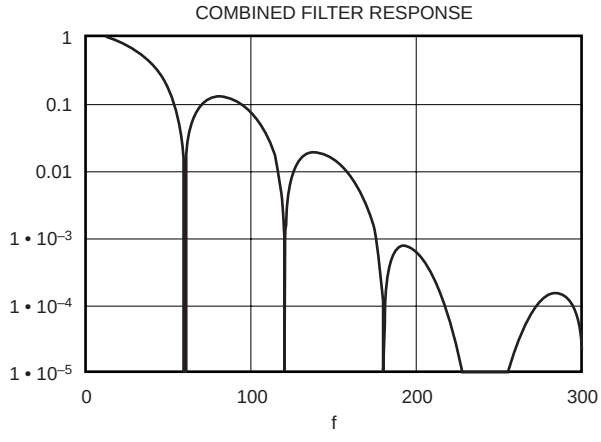
$$\left(\frac{\sin\left(\pi \cdot f \cdot \frac{64}{f_{mod}}\right)}{64 \cdot \sin\left(\pi \cdot \frac{f}{f_{mod}}\right)} \right)^3$$



$$\left| \frac{\sin\left(\pi \cdot f \cdot \frac{64 \cdot F_{div}}{f_{mod}}\right)}{64 \cdot \sin\left(\pi \cdot \frac{f \cdot F_{div}}{f_{mod}}\right)} \right|$$



$$\left(\frac{\sin\left(\pi \cdot f \cdot \frac{64}{f_{mod}}\right)}{64 \cdot \sin\left(\pi \cdot \frac{f}{f_{mod}}\right)} \right)^3 \cdot \left| \frac{\sin\left(\pi \cdot f \cdot \frac{64 \cdot F_{div}}{f_{mod}}\right)}{64 \cdot \sin\left(\pi \cdot \frac{f \cdot F_{div}}{f_{mod}}\right)} \right|$$



NOTE: f_{DATA} = 240; $f_{MOD} = f_{DATA} \cdot 64 =$; $F_{DIV} = f_{DATA}/60 = 4$.

FIGURE 1. ADS1210 using f_{DATA} of 240Hz and averaging four adjacent samples for 60Hz resulting data output.

This design will assume the use of a 10MHz clock for the ADS1211. The registers that need to be configured and the software operation to achieve our result are reviewed. We will assume a PGA setting of 1 and a turbo mode (TMR) of 16. Additionally, we will assume the default state for Byte 3 of the command register.

With these constraints, we have the following results:

$$f_{MOD} = \frac{f_{XIN} \bullet \text{Turbo Mode}}{512} = \frac{10\text{MHz} \bullet 16}{512} = 312.5\text{kHz}$$

$$\text{Decimation Ratio} = \frac{f_{XIN} \bullet \text{Turbo Mode}}{512 \bullet f_{DATA}} - 1 = \frac{10\text{MHz} \bullet 16}{512 \bullet 240} - 1 = 1,301$$

With four averages, this will give us a data out rate of 60.004Hz. This is within the valid range of 19 to 8,000 for the Decimation Ratio.

Plans should be made for Calibration. A decision could be made to do a periodic self-calibration, or the system could monitor the temperature and recalibrate when a significant temperature change has occurred.

For calibration, the following bits would be sent for a self-calibration:

INSR (Write, 1 byte, address = 0101) = 00000101

CMR (byte 2) (self cal, gain = don't care, channel = don't care) = 00100000

Setup

INSR (Write, 3 byte, address=0101) 01000101

CMR2 (normal, gain=1, channel=1) 00000000

Decimation Rate = 1301 = 515_H

CMR1 (TMR=16, 5_H) 10000101

CMR0 (1301 = 15_H) 00010101

Data Read

INSR (Read, 3 byte, address=0) 11000000

receive the next three bytes which will be the value of the Data Output Register.

Now the cycle continues. When $\overline{DRDY}$ goes low, generate the SCLK and data for the INSR followed by either writing to the command register to change the channel or reading the data. The data does not need to be read for the discarded data, just wait for the appropriate $\overline{DRDY}$ signal.

$\overline{DRDY}$ - *Setup*

$\overline{DRDY}$ - *Calibration*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Setup* then *Data Read*

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Setup* then *Data Read*

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Setup* then *Data Read*

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – Ignore

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

$\overline{DRDY}$ – *Data Read*

Note: If the software responded immediately to the $\overline{DRDY}$ pulse, this setup change would be considered synchronous and only two samples would have to be ignored.