eNET-AIO16-16F, Calibration concepts

Explanation of the USB-AIO calibration technique (for reference)

The USB-AIO16-16F ADC includes a real-time ADC data calibration feature, enabled on some models.

In the current design this is implemented as a 128k×16 SRAM between the CPLD and FX2 chip, on the GPIF address/data bus (with additional control signals).

During ADC data acquisition:

Each 16-bit A/D conversion's count value is applied to the address lines of the SRAM by the CPLD, causing the 16-bit data value at that SRAM address to be asserted on the SRAM's data lines, which in turn are read by the GPIF logic of the FX2 as if the SRAM data was the ADC conversion data. This means the SRAM performs as a simple lookup table where each ADC count value maps to a "calibrated" ADC count value.

In addition to the 16 ADC data bits used as addresses into the 128k×16 SRAM an additional address line is needed: this 17th address line selects between the two 64k×16 "halves" of the Calibration SRAM, and is controlled by the ADC reading's Input Range Polarity (such that all unipolar gains use one half of the Calibration SRAM and all bipolar gains use the other half of the SRAM; thus, TWO calibration lookup tables are implemented, one for unipolar and one for bipolar ranges. This range-polarity-specific dual-lookup scheme was implemented in Revision E of the USB-AIO Family design after discovering the need over the years prior.)

Because the SRAM is volatile it is necessary to load these two lookup tables at each reset.

During Calibration Lookup Table Loading:

Because the lookup is performed using a simple (single-ported) SRAM this means the data lines from the CPLD (that provide the ADC conversion data to the SRAM address lines) must be tristated and the SRAM address lines controlled from the FX2, while the data lines typically used to send (calibrated) ADC data to the FX2 are used to write into the SRAM lookup tables.

At reset the FX2 firmware loads both unipolar and bipolar calibration lookup tables with one-to-one (uncalibrated) tables.

That is, address the first SRAM address ("0b0 0000 0000 0000 0000") "UNI[0]" in the SRAM contains "0x0000", followed sequentially by 0x0001, 0x0002 etc, until UNI[0xFFFF] which holds 0xFFFF, then the next address, ("0b1 0000 0000 0000 0000") "BIP[0]", holds 0x0000 and continues through 0xFFFF in the last word of the SRAM. Note: The nomenclature used here, "UNI[n]" and "BIP[n]" is intended to reflect the Unipolar/Bipolar address line effectively splitting the SRAM in two halves.

At some point after reset application software loads actual calibrated lookup tables into the SRAM through the FX2 across the USB cable. During this process the ADC operation must be stopped.

Note:

> In order to support a 64k (×16) lookup table the ADC front-end must be designed to acquire data guaranteed to sample a wider range than the spec. E.g., in a 0-10V range the ADC must convert 0V into a count value above 0x0000, and must convert 10V into a count value below 0xFFFF, across the operational temperature range and lifespan.
>
> Thus, all calibration tables will have fewer than 65536 possible output values, with a repeating sequence of 0x0000 at the beginning and 0xFFFF at the end — and this causes "missing codes" in the calibrated data, and thus non-monotonic data.

Calibration Table Generation and Supporting Hardware:

In order to calculate a calibration table it is necessary to apply known voltages to the input and compare the voltage to the ADC output, at the two extremes of the ADC range. To avoid the customer having to source these known voltages our ADC circuit includes a calibration-feature-specific-mux (4-to-1) which allows the output of the ADC's channel-select muxes to be swapped for an on-card calibration reference without affecting the analog input signal pins.

This calibration mux can select among AGND and 0.9V, in addition to the actual ADC signals from the channel mux circuit. (0.9V was selected as this calibration mux output is fed to the gain amplifier, and thus needs to support ×10 gain.)

Our firmware / software reads these references many times and averages the data to produce the acquired value, and calculates the lookup table using a simple Y=mX+B interpolation, for each of the unipolar/bipolar calibrations, at the gain amplifier setting chosen by the customer.

In addition the 0.9V is physically measured during production test and the value stored in on-card EEPROM to further improve the calibration.

This "acquire known voltages, compare vs expected "perfect" readings, calculate lookup table" process takes just a millisecond or two, but the calibration table upload across USB consumes many more milliseconds.

eNET-AIO16-16F Calibration feature design notes:

It is not expected the eNET-AIO16-16F will implement real-time calibration using the above technique; no discrete SRAM is needed.

Instead, the spirit of the feature described shall be retained, as follows:

The eNET-AIO16-16F shall provide user application software the ability to read multiple calibration reference voltages without impacting the connected analog input pins (no need for customer to disconnect/connect anything), so the actual/expected reference readings can be used for calibration.

The design shall include separate calibration for unipolar/bipolar ranges (or as otherwise required by the analog front-end circuit).

Calibration can be performed at any time; indeed, in rough environmental conditions that experience wide/fast temperature swings calibration might be performed every hour or even more often — because calibration takes a small fraction of a second and no (physical) user interaction is required, calibration could occur at any time the ADC is not otherwise occupied.

I propose the eNET-AIO16-16F Family design eschew 16-bit digital ADC output data and instead use IEEE single-precision floating point "Voltage" as its preferred output data format. Instead of implementing calibration as a 128k×32 memory table the FPGA can simply perform the Y=mX+B calculation "live" in real-time. (Which the FX2 is *completely* incapable of, even at 100kHz let alone 1MHz, thus the SRAM-based design in the existing USB-AIO16-16F Family.)

Using floating point "Voltage" output (vs UInt16 “Counts") will eliminate the "missing-codes" and “non-monotonic data” problem caused by the SRAM lookup.

Using the FPGA to perform the linear interpolation (Y=mX+B) eliminates the SRAM/tables. In theory this allows more calibrations; per-range perhaps, instead of merely per range’s-polarity, accomplished by simply having more than two sets of calibration constants.