4.19 Curve Fit - Linearization

4.19.1 General

The CPP contains firmware that supports a curve fitting function. The method used in the curve fitting firmware is a piece wise linear approximation. This method requires that several values defining the input/output relationship (curve) be entered. Because the maximum number of values that may be entered will not completely define the curve, those values, which lie between the entered values, are approximated from a straight line between these values.

It therefore becomes the responsibility of the user to evaluate the values defining the curve, and select and enter those that best represent the curve. In those cases where there is a large change in output values for small changes in input values, the values chosen for this part of the curve should be closer together. Likewise, in parts of the curve where changes in the output values closely map those of the input values, the points may be spaced farther apart.

Curve approximation deserves some consideration. Assume that we have a portion of a large circle (an arc) to approximate. Envision two points on the arc connected with a straight line. The two points on the arc will be exact in their input/output representation, and the maximum error, Δ, will occur at the midpoint of the straight line - where the straight line is farthest from the arc. If the two input points are moved off of the arc by $\Delta/2$, one on one side of the arc and the other on the other side of the arc, then the midpoint of the straight line intersects the arc. This reported value would be totally correct, and the maximum error along any point on this segment approximation is $\Delta/2$. We have improved our representation of this curve by a factor of two. By properly selecting the spacing of the input points based on the shape of the curve very good representations of a curve can be realized in this fashion.

The full scale and zero values must be set up in the CPP prior to setting up the curve fit channels. If the full scale and zero values of the CPP are changed, then the curve fit numbers must be reentered. The full scale setting should also be a larger number than a number entered into the curve fit set up. If a curve fit input is –286.6, then the full scale setting for the channel should be greater than 286.6, 500.0 as an example. The only requirement for entered input values is that they start from the lowest value and proceed to larger values.

The curve fit feature can be used to solve a number of situations, including;

curve fit or Linearization,
offset signals (4-20mA or 1-5v),
negative going signals,
negative starting signals.

In the CPP, 4 – 20mA signals do not need offsetting. Selecting a 4 – 20mA, A/D input results in the A/D subsection compensating for the offset. If an offset signal is encountered, it is normally better corrected by scaling the full scale and zero settings of

the channel receiving the signal, rather than curve fitting. The curve fit can be used to produce the square root of a signal, but this is better accomplished in the computed channel feature of the CPP.

4.19.2 Set up a channel for curve fit

From the Main Menu select the curve fit option. The CPP responds with

Curve Fit Setup
Channel # = **01<cr>**
Enable Y/N = **Y<cr>**
Max # Points in curve = 02 **<cr>**
01In > 0.0 **20.0<cr>**
01Out > 0.0 **<cr>**
02In > 0.0 **100.0<cr>**
02Out > 0.0 **100.0<cr>**

Channel# = **<cr>** Ret to Main Menu

The above would remove the offset from a 4-20mA input signal, on channel #01 of the CPP. The decimal point of entered values must match those of the full scale setting of the CPP channel selected or the unit will not accept the value. A particular curve fit can be disabled by answering **N**, or no, when prompted. Entering an escape character results in the CPP reasking the previous question. Entering an E when the CPP is accepting input values, results in the CPP enabling this channel and asking for the next channel.

4.19.3 Some examples of various signal inputs are given below.

Negative going signal – Converted to positive going signal

Max # points in curve 02 **<cr>**
01In > 0.0 **–100.0<cr>**
01Out > 0.0 **100<cr>**
02In > 0.0 **0.0<cr>**
02Out > 0.0 **0.0<cr>**

Channel # = **<cr>**

As this signal increases in a negative direction, the signal stored in the CPP channel will increase positively. Note that the least positive input points must be entered first.

Negative A/D starting signal – Offset to zero start

Max # points in curve = 02 **<cr>**
01In > 0.0 **-100.0<cr>**
01Out > 0.0 **<cr>**
02In > 0.0 **<cr>**
02Out > **100.0<cr>**

Channel # = **<cr>**

A full scale negative input will result in a CPP value of zero, and a zero value input will be translated into a full scale value.

Thermistor input (44020)

The negative slope feature can be used to input values straight from a thermistor temperature measuring circuit. Thermistors exhibit a negative slope coefficient. For a standard -50°C to +50°C circuit, the relative voltages (normalized to one volt) are 0.873 for -50° and .313 for +50°. Set the CPP full scale value to 100.0, and the zero value to 0.0. The input to the A/D should be the negative slope output of the thermistor, or T3 as the input signal and ground as the signal ground.

Max # points in curve = 02 **<cr>**
01In > 0.0 **31.3<cr>**
01Out > 0.0 **50.0<cr>**
02In > 0.0 **87.3<cr>**
02Out > 0.0 **-50.0<cr>**
Channel # = **<cr>**

This setup will provide a temperature reading from -50°C to +50°C. The readings can be presented in Fahrenheit by setting the 01Out to 122.0°F, and 02Out to -58.0°F.

The full scale and zero values must be set up in the CPP prior to setting up the curve fit channels. If the full scale and zero values of the CPP are changed, then the curve fit numbers must be reentered.

Another temperature sensing device

Set the channel full scale to 500.0.

Max # points in curve = 02 **<cr>**
01In > 0.0 **<cr>**
01Out > 0.0 **142.6<cr>**
02In > 0.0 **500.0<cr>**
02Out > 0.0 **–286.6<cr>**
Channel # = **<cr>**