Function block, hysteresis, rounding, PT100, PT1000, motor control, PID, wave packet, MAX31865

1) „Rnd_any_DecPt“:

Rounds a real floating number to any decimal point you want. For example 3.1415927 => 3.14 or 3.1416.

a. IN_Real: Given input number;
b. Out_Real_rounded: Output of rounded number;
c. Dec_Pt: amount of desired decimal points.

2) „Analog_IN_Realtemp_OUT“:

If you like to measure temperatures with an transmitter which converts the resistance of a probe (normally PT100, PT1000) into a voltage or currency signal using an analog input, you might find this function block useful. It converts the analog input into a temperature value.

a. Res_IN: resolution of the analog input. E.g. 1023 (0-1023) = 1024 = 10bit
b. PT_Range: Temperature range of the transmitter: E.g. -50 -> 150 °C = 200
c. MW_act: Actual value coming from the Input
d. MW_Ist0: Calculated temperature

3) “Two_Steps_Burner_PID”:

This is a bit weird function block. We needed this one to control a 200KW gas burner from a brewery. This burner is operated by 3 switches: General On ; Half power ; Full power. Before automatization it was operate manually. For saving time this function was designed. The two steps of the burner are controlled by a standard PID-controller. A 2-way hysteresis prevents the burner from switching between 2 steps again and again. How to set up a PID please refer to documentation to this regard you will find in the Web. Because this block is derived from a specialized application, you will have to play with it to find out how to make it fit to your requirements. The PID-block itself is documented here: “http://www.smarteh.si/data/web_support_files_file/226got17111001_lpc3_got.111_usrman[2].pdf “ (By the way, all other OpenPLC function blocks as well).

a. Start: Self explaining
b. T_Set: desired temperature
c. T_act: Measured temperature from the probe
d. L_Min and L_Max: Limitation of the PID output
e. Low_ST1, High_ST1, Low_ST2, High_ST2: Hysteresis Limits for burner level 1 and two
f. Burner_on, Level_1 and Level_2: Outputs to relays controlling the three burner functions. (On, Half power, Full power)

4) “Hysetrese2”:

This function block I converted from the “OSCAT-Library”. This library is always worth a look. Unfortunately, not all provided FB´s are usable in OpenPLC (or to be honest, due to my skills, not usable for me). Probably Mr. Alves might think different about this! The FB does this: It switches Out off if IN < Low; an Out on if IN > High.

a. IN: Input Value
b. Low: low border
c. High: High border

d. Out: Boolean output

5) „Well_pak_steu“:

This FB is used to control the power of an AC-resistive-load or the speed of an AC-motor. Typically used in controlling electric heaters or circulating pumps in heating systems. How does it work: By a given percentage between 0-100% the FB sends an equal amount of complete sinus waves to the load. E.g. 25% desired power of the load 1 AC-cycle is let through the following 3 are blocked. The algorithm is self-optimizing. It prevents low frequency outputs. The FB is set up for 50Hz Main power frequency and for 1% steps, which means a cycle time of 20ms. If you life in a 60Hz region you will have to work out if it works with 16ms cycle time and 200 as sampling rate (0.5% steps). I can´t test it, because I have 50Hz around. Anyway you have to insure that the cycle time is an integer multiple of the main power period time (duration of a whole AC sinus wave). If you want to test this FB you must use a fast switching solid state relay with zero crossing ability. With controlling motors low percent values will cause undesired rough motor noise or even a stop an go operating. Individual trials are recommended. The FB works by summing up the percentage values. Here are two examples (you will easily find out, that after a few complete turns the on an off relation exactly matches the desired percentage):

a. 30% desired load:
i. 1. Cycle: 30% => off
ii. 2. Cycle: 60% => off
iii. 3. Cycle 90% => off
iv. 4. Cycle 120% => on => -100
v. 6. Cycle 50% => off
vi. 7. Cycle 80% => off
vii. 8. Cycle 110% = on => -100
viii. And so on

b. 80% desired load_
i. 1. Cycle 80% => off
ii. 2. Cycle 160% => on => -100
iii. 3. Cycle 140% => on => -100
iv. 4. Cylce 120% => on => -100
v. 5. Cycle 100% => on => -100
vi. 6. Cycle 80% => off => -100
vii. And so on

c. “Summe”: internal use
d. “K_100”: sampling rate
e. “SSR”: Boolean out to zero crossing Solid state relay
f. “Start”: Self explaining
g. “K_T_02”: Cycle time
h. “Percent_Power”: Desired load in integral numers 0-100; Input

6) “PT_Calc“:

If you measure temperature with PT probes you might have recognized that there are several approaches. An easy way is described above: „Analog_IN_Realtemp_OUT“. There are two limitations: First, it’s not the best way in terms of accuracy. Second, you have to define the range of temperature to be measured in advance due to PT’s non-linearity over wider temperature ranges. Here you find a way which solves both limitations: You can use a PT probe in range of -200-850°C. And the resolution is much higher (16bit). I get my raw data from a MAX31865 chip, converted into a resistance at first. The resistance is converted into temperature by two

equitation. One for temperatures above and one below 0°C. The correct formula is selected automatically. The FB is usable for any PT (100, 1000, 500…). If you are not using a MAX-chip but you may have a reliable and precise resistance of the PT pro however, just get rid of the converting stuff (Rtd to resistance) and start your FB with the resistance directly. If you are looking for an Arduino sketch to get a MAX31865 and “Arduino like boards” running via SPI with OpenPLC, I posted one for MEGA and one for UNO in this forum as well.

a. “Rtd”: Input from MAX31865
b. “R_0_Grad”: Nominal resistance of the PT at 0°C
c. “R_Ref”: Reference Resistor on the MAX board (430 for PT100; 4300 for PT1000)
d. “R_Ist”: actual resistance of PT Probe
e. “a, b, c”: Constants of the PT probe. They may vary according to the quality of the probe. For standard versions you will find them in the web. If not provided, you can calculate them by yourself easily. (https://blog.beamex.com/pt100-temperature-sensor)
f. “Temp”: Output of calculated temperature
g. “K_a_kl0-K_e_kl0; K_242_02”: Constants used for the formula below 0°C. Instead of the origin formula for temperatures above 0°C which is solvable, the formula below 0°C isn´t.

To be honest, it is, but it might not fit your screen. For that an approximation formula is used. The described constants vary slightly. Go ahead and try out. If you find better values, please post.