PoelControl – Sensor Sampling

1.Status

- 2006-04-14: First draft – Added analog sensors
- 2006-05-19: Updated analog sensor section (spelling, wording)
- XXX – Needs info for digital sensors

2.Intro

PoelControl is a project aimed at the development of a flexible and powerful microcontroller based solutions for the control of an installation for vegetable fuel (poel) as installed here in my vehicle.

The objective of this document is to give detailed information about data sampling from the various input sources / sensors. Algorithms for sampling are described; methods used in the application are named.

3.Sampling

In the main application loop the methods for sensor sampling, poel logic and display updates are called contiously. Because the signals measured are not high frequency signal the sampling of sensor data is not performed on each loop.

Based on the 10 ms and 100 ms counters controlled by the interrupt based timer one sampling process of all sensors is performed every <n> * 100 ms. <n> defines the tick rate for sensor sampling i.e. the number of 100 ms intervalls between two samplings. The tick rate is defined by pcSensorTickRate in sensors.h and increased every 100 ms.

4.Analog Sensors

Temperature Sensors

Temperature sensors can be connected to the internal A/D 10-bit converter. The conversion is done on each *update sensors* cycle. A linear characteristics is assumed so that the temperature is calculated as follows:

$$T = Gradient * Voltage + Offset$$

Gradient and Offset are derived from the datasheet of the temperature sensor used. For the temperature sensor for the exhaust gases a moving average, see below, might be used.

Pressure Sensors

Pressure sensors can be connected to the internal A/D 10-bit converter. The conversion is done on each *update sensors* cycle. A linear characteristics is assumed so that the pressure is calculated as follows:

$$p = Gradient * Voltage + Offset$$

Gradient and Offset are derived from the datasheet of the pressure sensor used.

For the *boost pressure* and *fuel pressure* a moving average is calculated using the last <n> pressure values stored in a ring buffer. The size <n> of the ring buffer is defined by *pcMAElements* in tools.h. With this approach the fluctuation of the signal is reduced; incorrect detections of critical operating conditions are prevented.

Tank Level

For the tank level a sensor is used that changes resistance according to the tank leve. The sensor is connected to the internal A/D 10-bit converter. The conversion is done on each *update sensors* cycle. A linear characteristics is assumed so that the tank level is calculated as follows:

$$Level = Gradient * Voltage + Offset$$

Gradient and Offset are derived from the datasheet of the sensor used.

For the *tank level* a moving average is calculated using the last <n> values stored in a ring buffer. The size <n> of the ring buffer is defined by *pcMAElements* in tools.h. With this approach the fluctuation of the signal e.g. due to sloshing is reduced; incorrect *low fuel level* warnings are prevented.

Engine Speed

The engine speed signal is present on terminal *W*. For each revolution of the engine a fixed number <n> of impulses is emitted. With a impulse frequency of <f> the engine speed can be calculated as follows:

$$rev/min = \frac{f * 60}{n}$$

The number of impulses per revolution <n> can be found in the vehicle documentation.

For the *engine speed* a moving average is calculated using the last <n> values stored in a ring buffer. The size <n> of the ring buffer is defined by *pcMAElements* in tools.h. With this approach an average of the engine speed is calculated.

Vehicle Speed

The vehicle speed signal is present via the *GALA signal*. For each revolution of the wheels a fixed number <n> of impulses is emitted. With a impulse frequency of <f> and a wheel circumference <c> in m the vehicle speed <v> in km/h can be calculated as follows:

$$v = \frac{3.6 * f * c}{n}$$

The number of impulses per revolution <n> can be found in the vehicle documentation.

For the *engine speed* a moving average is calculated using the last <n> values stored in a ring buffer. The size <n> of the ring buffer is defined by *pcMAElements* in tools.h. With this approach an average of the vehicle speed is calculated.

5.Digital Sensors

To be written