ABB Fan Controller – Program Documentation

Classes

DigitalIoPin

- Provides an object interface to the board library for configuring pins.
- **read()** gets pin state
- **write()** sets pin state

Fan

- Communicates with the frequency controller board using modbus
- **setFrequency()** sets the frequency of the fan
- **getFrequency()** gets the frequency of the fan

Pressure

- Communicates with the pressure sensor using I2C
- **getPressureDiff()** gets the measured pressure difference

Button

- Wraps an IO pin to provide a button utility interface
- **get()** returns the pressed state
- **getRepeat()** when button is down returns true at specified tick interval
- **getDown()** returns true when button is down but was not down previously
- **getUp()** returns true when button is not down but was down previously

LiquidCrystal

- Communicates with a character LCD
- **print()** prints a null terminated char array on the display

Program

Startup

- Create DigitalIoPins and Button objects
- Create DigitalIoPins and LiquidCrystal object, running the required initialization
- Create Fan and Pressure objects, running the required initialization

Main loop

- **mode** variable stores the current mode, MANUAL or AUTO
- Button 3 (b0) is checked to switch mode between Manual or Auto

Manual Mode (mode == MANUAL)

- when button 1 (b1) is down, frequency is decremented
- when button 2 (b2) is down, frequency is incremented
- frequency is clamped between 0 and MAX_FREQ (20000)
- new frequency is sent to the fan
- current fan speed(frequency/MAX_FREQ) and pressure is fetched and printed on the lcd

Auto Mode (mode == AUTO)

- if button 1 is down, target pressure is decremented
- if button 2 is down, target pressure is incremented
- target pressure is clamped between 0 and MAX_PRESSURE (120)
- if button 1 or 2 is down, the target pressure is printed on the lcd
- current frequency and pressure is fetched
- a difference between current pressure and target pressure is calculated
- if difference is larger than EPSILON (1), frequency is incremented by a relatively small value multiplied by the difference, changing the fan speed with the aim to change the pressure towards the target
- frequency is clamped between 0 and MAX_FREQ (20000)
- a variable amount of ticks (scaled by the last difference) is waited before adjusting the fan speed again to give time for the pressure reading to update
- if button 1 or 2 (set target pressure) has not been pressed for 1 second, the current fan speed and pressure is printed on the lcd