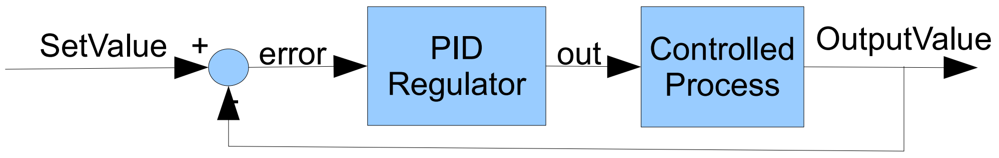
PID Regulator

The PID control

A PID algorithms is the most often used feedback control in industrial control systems. A regulation systems is a feedback circuit where the PID controller gets an **e**(rror):

$$e = \text{SetValue} - \text{OutputValue} \qquad (1)$$

an works out an **out** signal controlling a Process.



Sometimes PID regulator is called three-term control consisting of:

- the proportional part
- the integral part
- the derivative part.

Not all terms are used in all control systems. Their use and its influence on the the regulation loop is defined by three parameters corresponding to the regulator terms:

- Kp: Proportional gain
- Ki: Integral gain
- Kd: Derivative gain

The formula describing PID algorithm is a function of error changing in time:

$$out = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{d}{dt} e(t) \qquad (2)$$

The most popular regulator is the PI controller where Kd = 0.

In addition to the standard calculation algorithm there are many other variants known and applied.

An exhaustive discussion of PID feedback control can be found in innumerable papers, books and last but not least in Internet. The effective use of a PID regulator requires some basic knowledge about process control.

Floating- or Fixed-point implementation

Today PID regulators are basically used in a digital form: just a function realizing calculation according to formula (2). Considering the calculations to be performed floating-point arithmetic is the obvious and first choice. In cases of microcomputers with limited resources the floating-point might be unavailable or unacceptable due to time constrains. In such cases fixed-point arithmetic may used. Although the limitation of a fixed-point arithmetic limits the flexibility of the regulator this solution is in many situations acceptable.

The PID regulators in our analog boards Ai4Ao1 and Ai4Ao2 are realized using fixed-point arithmetic.

Implementation in Ai4Ao2 board

The Ai4Ao2 board contains by default two PID regulators each linked with an output. The input (ProcessOutput) used in a regulator is freely configurable; all inputs Ai0..Ai3 can be used as an input to any of the two regulators. The SetValue is a parameter. The other parameters are: the Kp, Ki and Kd gains as well as a input offset *in_zero*. The parameter ranges:

- SetValue: 4-20mA (0x1999 … 0x7fff)
- Kp, Ki, Kd: positive integer
- in_zero: 4-20mA (0x1999 … 0x7fff)
- sampling interval: 80 msec (12.5 Hz)

A standard mode of regulator working is when the *in_zero* equals 0. The zero can be shifted by setting *in_zero* to a value larger then 4, typically 12mA (0x4ccc). The shifted mode is used when the controlled object (sensor) has an +/- output, for instance: -50°C to 50°C, which means that:

- 4mA represents -50°
- 2mA represents 0°
- 20mA represents +50°

Software handling

While programming the ioboard.dll is used. The dll provides the following functions for handling a PID Regulator:

```
public bool SetControllerState(byte boardId, byte inNum, byte outNum, byte state)
```

where:

- boardId: number 0..lastId
- inNum: regulator input number: 0..3
- outNum: regulator output number: 0 (for Ai4Ao1), 0..1 (for Ai4Ao2)
- state: 0 - Disable, 1 - Enable, 2 - Reset

The function returns true if successfull setting, otherwise false.

```
public bool GetControllerState(byte boardId, byte outNum, ref byte state, ref byte inNum)
```

where:

- boardId: number 0..lastId
- outNum: regulator output number: 0 (for Ai4Ao1), 0..1 (for Ai4Ao2)
- state: regulator state; expected 0 (disabled) or 1 (enabled)
- inNum: regulator input number

The function returns true if successfull setting, otherwise false.

```
public bool SetControllerParameters(byte boardId, byte outNum, double reference,
                                     ushort kp, ushort ki, ushort kd, double in_zero)
```

where:

- boardId: number 0..lastId

- reference: set value: 4..20 (mA)
- kp, ki, kd: regulator parameters (integer > 0)
- outNum: regulator output number: 0..3
- in_zero: regulator input zero: 4..20 (mA)

The function returns true if successfull setting, otherwise false.

```
public bool GetControllerParameters(byte boardId, byte outNum, ref double reference,
                                    ref ushort kp, ref ushort ki, ref ushort kd, ref double in_zero)
```

where:

- boardId: number 0..lastId
- outNum: regulator output number: 0..3
- reference: set value
- kp, ki, kd: regulator parameters
- in_zero: regulator input zero

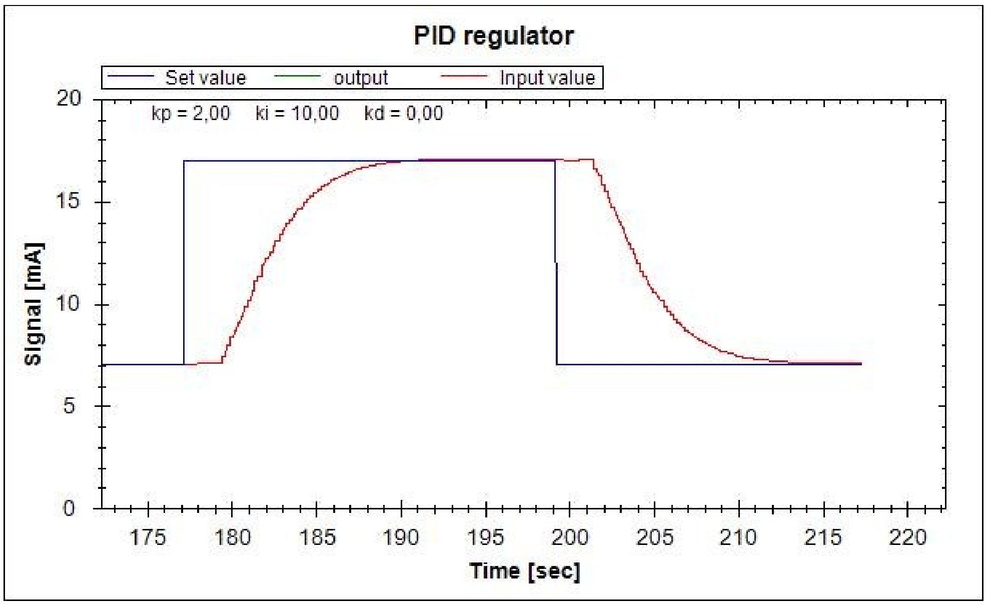
The function returns true if successfull setting, otherwise false.

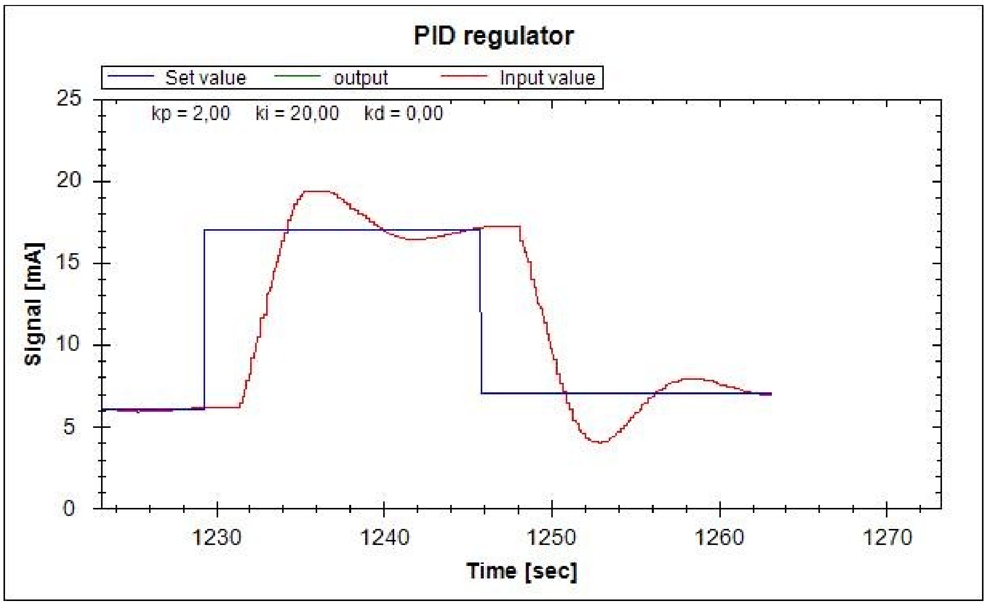
Test

The regulator has been tested with an object with 2s delay. The result are shown on diagrams below.

The following diagrams show typical changes of the regulated *input value* caused by a jump of the *setup value*. There are diagrams for different values of parameter gains Kp and Ki (Kd=0 in all tests).

Standard mode

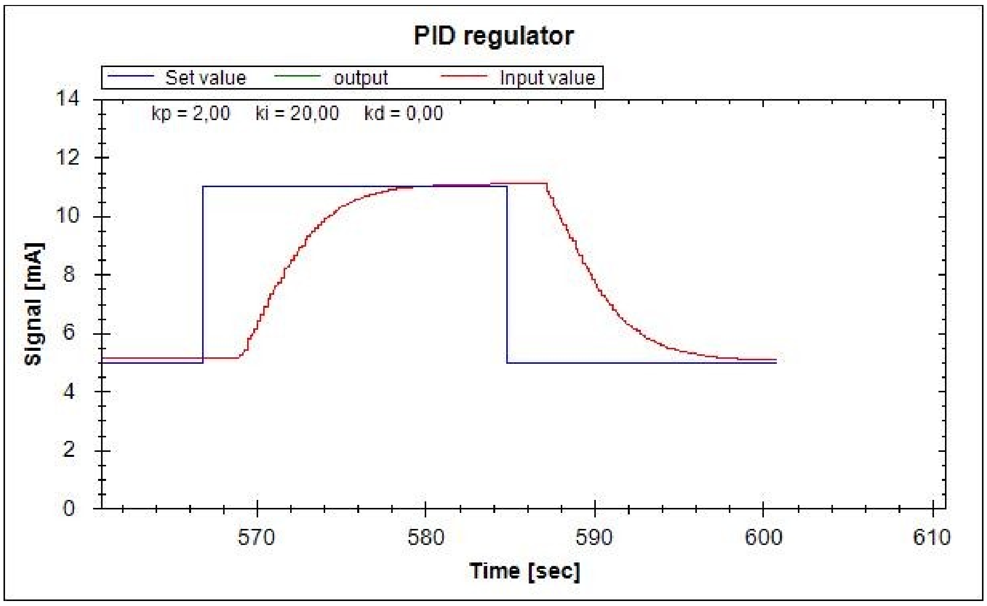


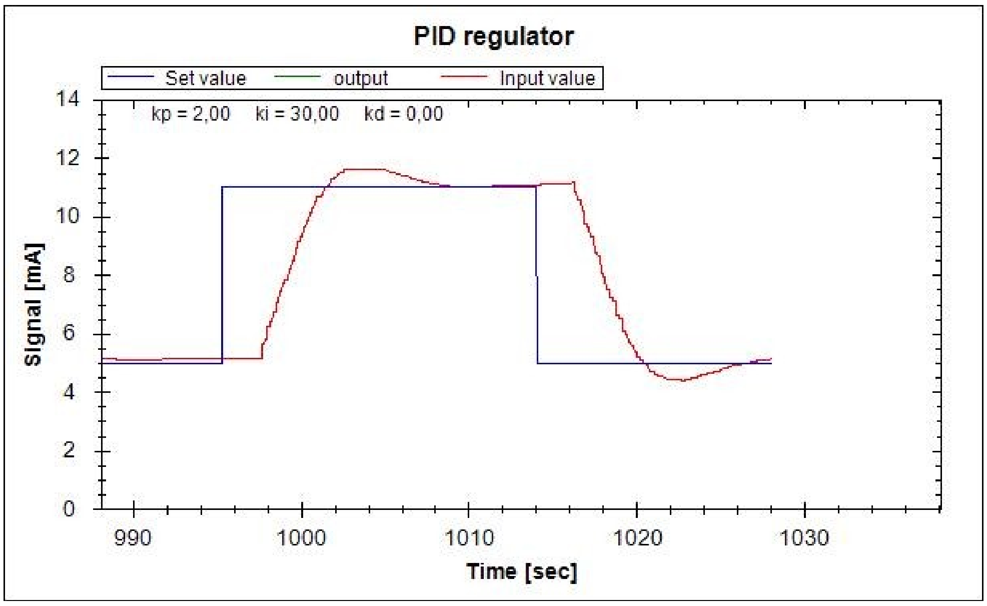


With zero shifted to 12mA

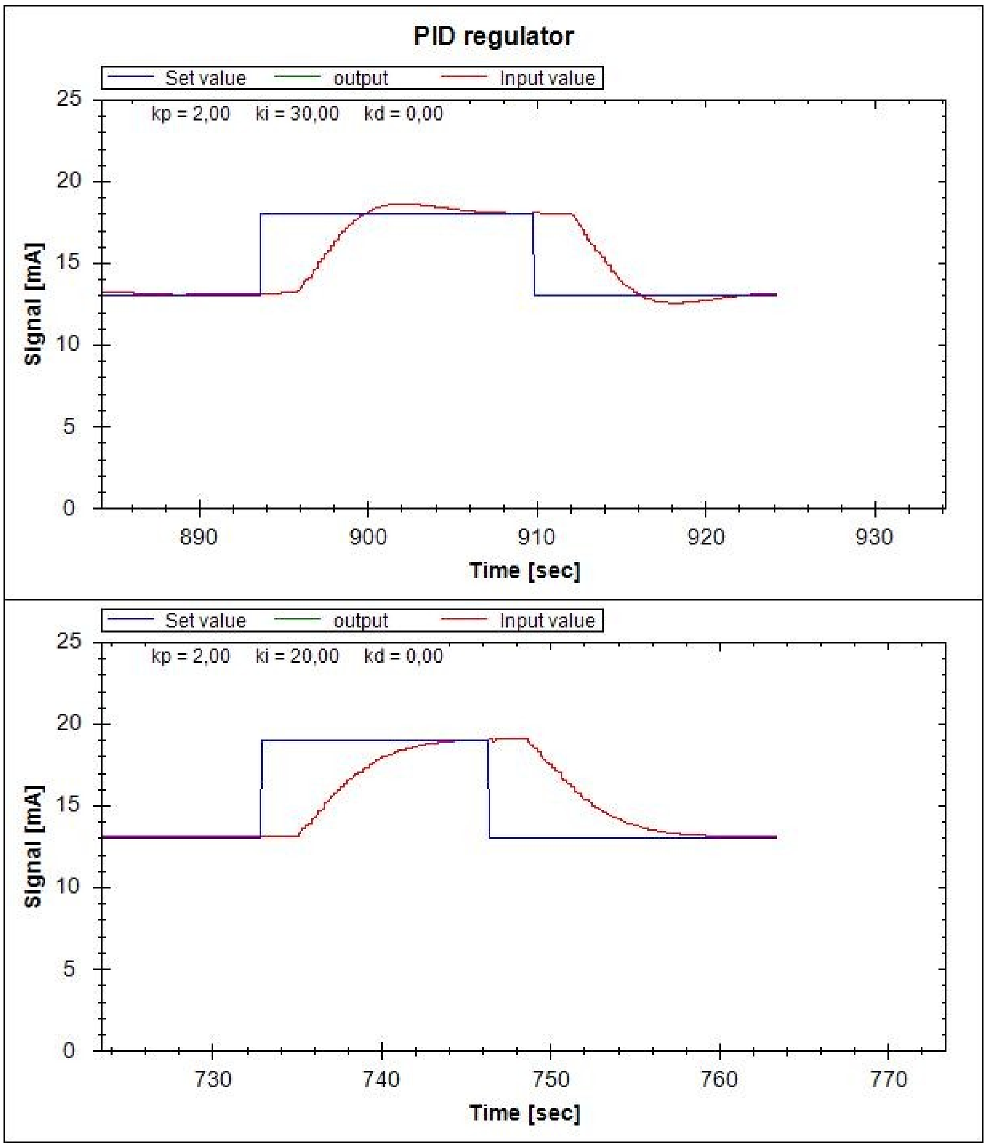
The same experiments have been done for zero shifted to 12mA. In such a case there are two different regulation regions.

Regulation region 4..12mA





Regulation region 12..20mA


PID regulator
Set value
output
Input value
kp = 2,00 ki = 30,00 kd = 0,00
Signal [mA]
Time [sec]
Set value
output
Input value
kp = 2,00 ki = 20,00 kd = 0,00
Signal [mA]
Time [sec]