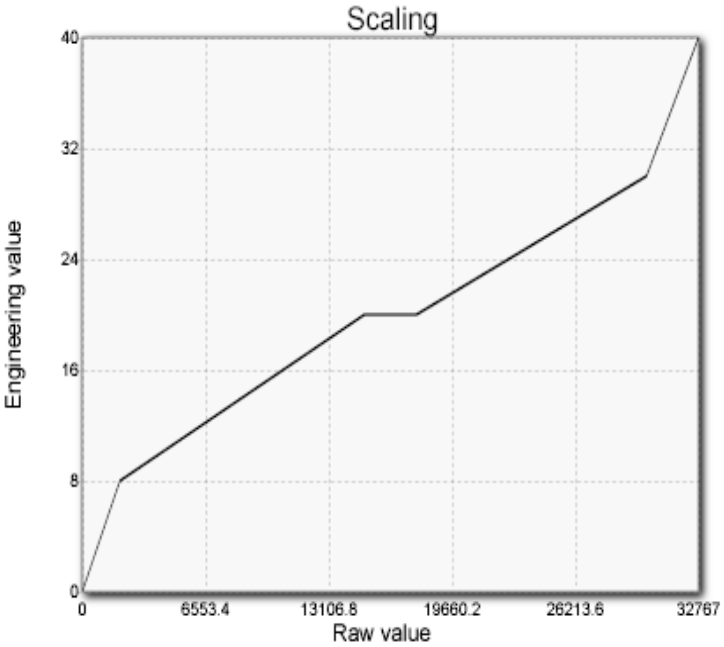
Product:	AnalogChannel with multipoint scaling
Document ID:	UM - AnalogChannel with multipoint scaling
Document type:	User Manual
Doc revision:	B
Date:	07.10.2008
Pages:	9

Industrial Control Design AS



AnalogChannel with multipoint scaling
User Manual

Industrial Control Design AS, www.icd.no, support@icd.no, forum.icd.no

Contents

1. Introduction

This document describes how the AnalogChannel object works , and how to set-up and use AnalogChannel in CDP.

The purpose of AnalogChannel is to provide scaling for input and output signals, including corrections for non-linearities.

AnalogChannel objects support Unscaled, TwoPoint scaled and MultiPoint scaled channels.

The scaling is set up by editing the component xml file, or by changing the scaling parameters on the web page for a channel.

The AnalogChannel C++ class inherits from and extends the CDP Signal class. AnalogChannel object should therefore be thought of as normal signals with some extra features for working with raw values typically acquired from hardware.

2. Functionality

2.1. Range limiting

Range limiting is enabled by specifying one of the limits Min, Max, HardwareMin, HardwareMax.

2.2. Unscaled

When unscaled the channels scaled value equals the raw value.

2.3. Two point scaling

The scaling is denoted by specifying two points of (raw value, engineering value): (raw0, eng0) and (raw1, eng1).
A scaling factor and offset is calculated from the two points and used to scale from raw value to enginnering value and vice versa:

```
ScalingFactor = (eng1-eng0) / (raw1-raw0);
Offset = eng0 - raw0 * ScalingFactor;
```

Scaling from rawvalue to engineeringvalue (when reading analog intputs):

```
scaledValue =  rawValue * ScalingFactor + Offset;
```

Scaling from engineering value to raw value (when writing analog outputs):

```
rawValue =  (scaledValue-Offset) / ScalingFactor;
```

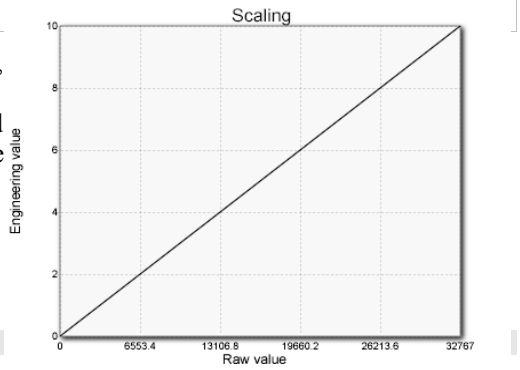


Figure 1Two point scaling

2.4. Multi-point scaling

Instead of specifying only two points as in two point scaling, an optional number of points may be specified to create an arbitrary “scaling curve”.

Nr	Raw value	Engineering value	Delete
1	0	0 m/s	Delete
2	2000	8 m/s	Delete
3	15000	20 m/s	Delete
4	17768	20 m/s	Delete
5	30000	30 m/s	Delete
6	32767	40 m/s	Delete

Table 1 - Scaling points

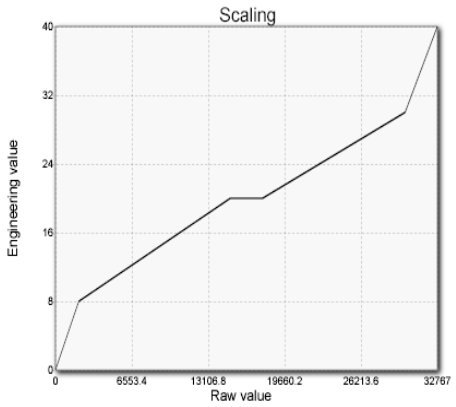


Figure 2 - Scaling curve when using multi point scaling (using points as in Figure 1)

If for instance the scaling points in Table 1 are used, a "scaling curve" like the one showed in Figure 2 is defined.
When scaling, a linear scaling will be calculated from the two points closest on each side of the value that is to be scaled.

3. AnalogChannel Configuration

3.1. Overview

The scaling of each channel is determined by the scaling points specified in the component .xml configuration file for the IOServer.

The .xml configuration may be edited in a text editor, or the scaling points may be modified from the web page for each channel (see Chapter Web page for the AnalogChannel).

3.2. Configure using Web page

3.2.1. Opening the configuration page

In the web-browser, type the ip-address of the controller as web page destination. When the controller welcome page appears, browse to the IOServer component containing your signal using the links on the left. Click on the signal to open the properties page.

3.2.2. Information on the web page

Figure 3 shows an example of a web page for an AnalogChannel. In addition to the standard signal attributes, attributes specific to this AnalogChannel instance is also listed:

RawValue
The unscaled and unlimited value as read from the hardware.

Hardware datatype
Data type used when interpreting the data read from/written to the hardware.

Register number
Some IOServers calculate and use a register number when addressingthe hardware. This is this number is for this channel.

Module/channel number:
Some IOServers use module number and channel number when addressing the hardware. This is the module number used for this channel.

Scaling type:
Show what kind of scaling is used when converting from raw value to engineering value. Currently, the scaling type is determined by how many scaling points that are specified.

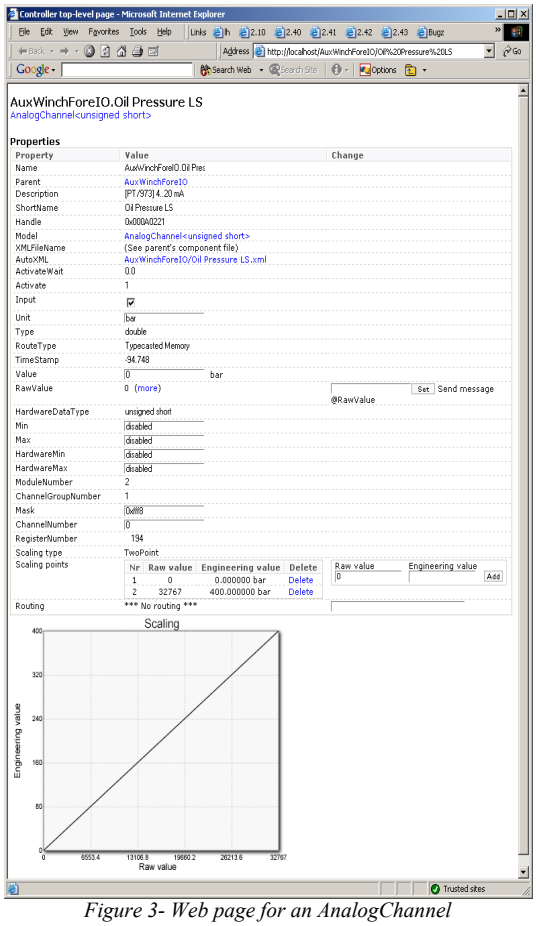


Figure 3- Web page for an AnalogChannel

Type: The type of the Signal, i.e, the datatype that will be used inside CDP when working with the Signal.

HardwareDataType: The datatype that the raw value is interpreted as when read from/written to hardware.

Scaling points:
Lists the scaling points that are specified for this channel. Includes fields for modifying the scaling points.

3.2.3. Editing scaling points

Scaling points may be added or deleted from the web page for the analog channel. To add a scaling point, enter values for raw value and engineering value in the fields to the right of the scaling point table, then click the "Add scaling point" button. The scaling point table and the scaling graph should be updated with the new point.

To delete a point, click the "Delete" link to the right of a scaling point inside the scaling point table.

3.3. Configure in .xml

3.3.1. Two point scaling

```
<Channel Nr="0" Name="Joystick"  Unit="m/s"  ...>
  <Scaling>
    <Point HardwareValue="0"     EngineeringValue="0.0"></Point>
    <Point HardwareValue="32767" EngineeringValue="100.0"></Point>
  </Scaling>
</Channel>
```

3.3.2. Multi point scaling

```
<Channel Nr="0" Name="Joystick"  Unit="m/s"  ...>
  <Scaling>
    <Point HardwareValue="0"     EngineeringValue="0.0"></Point>
    <Point HardwareValue="2000"  EngineeringValue="8.0"></Point>
    <Point HardwareValue="30000" EngineeringValue="30.0"></Point>
    <Point HardwareValue="32767" EngineeringValue="40.0"></Point>
  </Scaling>
</Channel>
```

If 0 or 1 points are specified, the channel will be unscaled.

If two points are specified, two point linear scaling will be used.

If more than two points are specified, multi-point scaling will be used. Multipoint scaling finds the two scaling points closest to (on each side of) the value that is to be scaled, calculates a linear scaling based on these two points and scales the raw value based on this linear scaling.

3.3.3. Range limiting

The minimum or maximum allowed signal values may be specified in either engineering units or in raw value units. This is called “range limiting”. Zero to four of these limits may be specified. If range limiting is specified in both engineering and hardware units, the limit that limits the most will be used. If no limits are specified, the signal range is not limited.

Use the *Min* and *Max* attributes to specify range limiting in *engineering* units:

```
<Channel Nr="0" Name="Joystick" Unit="m/s" Min="0" Max="5">
  <Scaling>
    <Point ...></Point>
  </Scaling>
</Channel>
```

Use the *HardwareMin* and *HardwareMax* attributes to specify range limiting in *raw value* units:

```
<Channel Nr="0" Name="Joystick" Unit="m/s" HardwareMin="0" HardwareMax="32767" ...>
  <Scaling>
    <Point ...></Point>
  </Scaling>
</Channel>
```

3.3.4. Masking

Use the *Mask* attribute to specify a bitmask to be ANDed with the raw value before conversion to engineering value (for inputs, and after conversion to hardware value for outputs). For instance, to mask away the least significant bits:

```
<Channel Nr="0" Name="Joystick" Unit="m/s" Mask="0xfffe" ...>
  <Scaling>
    <Point ...></Point>
  </Scaling>
</Channel>
```

4. Using AnalogChannel in C++

4.1. About

AnalogChannel instances are used in IOServers to provide scalable signal inputs and outputs.

AnalogChannel is an extension of the Signal<T> class.

4.2. Methods

```
unsigned long GetRawValue();
```
Returns the unscaled and unmasked value.

```
T ReadUnscaled();
```
Returns the unscaled, but limited signal value.

```
void WriteUnscaled(HardwareDataType rawValue);
```
Write the unscaled and unlimited signal value.

```
double Scale(double rawValue);
```
Scales from raw value to engineering value.

```
double Unscale(double scaledValue);
```
Scales from raw value to engineering value.

```
void SetBuffer(void* pBuffer, double* pTimeStamp);
```
Specify the buffer from/to which the signal should read/write it's (hardware) value and timestamp.

4.3. Typical IOServer code

The following code snippet is a typical example of code used during Configure() of an IOServer to dynamically create AnalogChannel objects:

```
CDPXMLConfiguration xmlSignal;

while (xmlPacket.GetChildElement("Signal", xmlSignal))
{
  std::string signalName = xmlSignal.GetAttribute("Name");
  std::string signalType = xmlSignal.GetAttribute("Type");
  std::string objectType = std::string("AnalogChannel<") + signalType + ">";
  SignalBase* pSignal = dynamic_cast<SignalBase*>( CDPBuilder::CreateCDPObject( objectType,
signalName, m_pParent) );
  if (pSignal!=NULL)
  {
    pSignal->SetInput( m_bInput );
    pSignal->Configure(xmlSignal.StartPosition(), xmlSignal.StopPosition());
    pSignal->SetBuffer( pBuffer, pTimeStamp );
  }
}
```