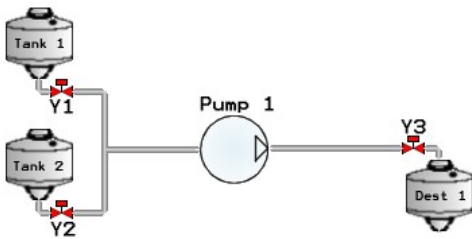




| | |
|------------------|---------------|
| Product: | RuleBase |
| Product version: | V1.1 |
| Document ID: | UM-RuleBase |
| Doc revision: | A |
| Written/Aprr.: | SL / RE |
| Date: | 16. Oct. 2008 |

Industrial Control Design AS



RuleBase V1.1

User Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

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

Contents

| | |
|--|----------|
| 1. INTRODUCTION..... | 3 |
| 1.1. About..... | 3 |
| 1.2. RuleBase composition..... | 4 |
| 1.3. Example 1..... | 5 |
| 1.4. Example 2..... | 6 |
| 1.5. The complete RuleBaseExample.xml..... | 7 |

1. Introduction

1.1. About

The purpose of the RuleBase component is to be a container for a collection of rules that describes the behaviour of a system. It will parse an arbitrary xml section and read the xml into the internal containers (std::vector).

Perhaps the most important feature of the RuleBase is that it provides the a way of specifying the rules in an xml syntax that may be chosen while configuring the system. Changing the xml is all that needs to be done to change the behavior.

The RuleBase does not provide any logics to interpret the rules. That task is left to inherited classes.

Rule based control may be useful for systems where it is inconvenient, difficult or impossible to set up mathematical or logical expressions describing the relationships between the inputs and the outputs. This will most often be true for systems with a lot of inputs and outputs.

1.2. RuleBase composition

The Rulebase consist of a set of rules. Each rule is defined by a set of parameters that is specified either as xml elements or xml attributes. The outline of the rulebase xml is as follows:

```
<Component Name="RuleBaseExample" Type="RuleBase">
  <Signals>
  </Signals>
  ...
  <RuleBase>
    <Parameters>
    </Parameters>
    <Rules>
    </Rules>
  </RuleBase>
</Component>
```

Before the RuleBase can parse the rules, it needs to know which parameters (ie. which rule syntax) to search for. This is done by declaring all the rulebase parameters in the <Parameters> section:

```
<Parameters>
  <Parameter Name="Destination"></Parameter> <!-- xml element -->
  <Parameter Name="From"></Parameter> <!-- xml element -->
  <Parameter Name="Y1"></Parameter> <!-- xml attribute -->
  <Parameter Name="Y2"></Parameter> <!-- xml attribute -->
  <Parameter Name="Y3"></Parameter> <!-- xml attribute -->
</Parameters>
```

This parameter declaration can parse a rulebase on the following format:

```
<Rules>
  <Destination Name="Dest 1" Y3="O">
    <From Name="Tank 1" Y1="O" Y2="C"></From>
    <From Name="Tank 2" Y1="C" Y2="O"></From>
  </Destination>
</Rules>
```

The values assigned to each parameter by the rule definitions are stored as std::strings in the rulebase. *The meaning of the values have no meaning to the rulebase, but must be interpreted by a class that inherits from the RuleBase.*

Requirements when setting up the rulebase:

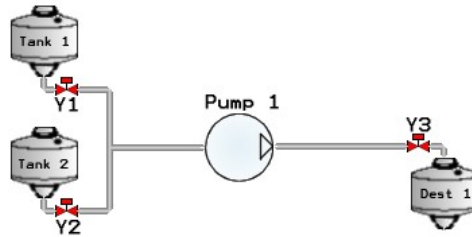
- The first parameter must define the name of the *outermost* xml element in the rulebase, the next parameter must be the name of the element *inside* the first etc. The parameters may be specified either as elements or attributes, the only requirement is that the elements are listed *before* all attributes in the <Parameters> section.
- To assign values to parameters specified as elements, the following syntax is used: <ParameterName Name="ParameterValue" ...>...</ParameterName>. That is, use the Name attribute to specify which value to assign to the parameter.
- Attributes may be specified anywhere in the rule section. For instance, the Y3 parameter was specified inside the Destination element above, but it could just as well be specified inside each of the From elements. When specified as above, it acts as a default value for Y3 for the two From rules inside the Destination element.
- When reaching the innermost element, all the rulebase parameters must have had values assigned for a rule to be created and added to the rulebase.

1.3. Example 1

Task: Set up an xml description that describes all the valid combinations of valve settings for the following system:

System description:

Pump from tank 1 or 2 via pump 1 to destination 1. Pumping is only allowed from one tank at a time.



The purpose of the rules is to define all the valid combinations of valve settings. This will later be used to check if a requested combination of valve settings are allowed.

This system will have only two possible combinations, and may be described as follows:

```
<Rules>
  <From Name="Tank 1" Destination="Dest 1" Y1="O" Y2="C" Y3="O"></From>
  <From Name="Tank 2" Destination="Dest 1" Y1="C" Y2="O" Y3="O"></From>
</Rules>
```

The first rule defines the first route as valve Y1=open, valve Y2=closed and valve Y3=open. This will create a route from Tank 1 via the pump to Dest 1 while cutting off Tank 2 by keeping valve Y2 closed.

As systems get more complex, the rules will get long and inconvenient. Instead of listing all parameters on every line, a preferred description would be:

```
<Rules>
  <Destination="Dest 1" Y3="O">
    <From Name="Tank 1" Y1="O" Y2="C"></From>
    <From Name="Tank 2" Y1="C" Y2="O"></From>
  </Destination>
</Rules>
```

This defines exactly the same rules, but valve Y3 is defines as open for all rules defined inside the <Destination="Dest 1"..> element.

This is a very simple example to show setup of a the rules only.

To enable the RuleBase to parse this xml, the parameters of the rulebase need to be defined in the <Parameters> section. For this system the parameters will be like this:

```
<Parameters>
  <Parameter Name="From" Type="Info"></Parameter>
  <Parameter Name="Destination" Type="Info"></Parameter>
  <Parameter Name="Y1" Type="Tank"></Parameter>
  <Parameter Name="Y2" Type="Tank"></Parameter>
  <Parameter Name="Y3" Type="Valve"></Parameter>
</Parameters>
```

Note again that the Type attribute is optional. The use of this parameter is left to the component implementing the interpretation of the rules.

1.4. Example 2

Construct the rules for a pump system where one of two pumps must be able to pump stuff from one of two tanks to one of two destinations. Only one pump is allowed to pump from a tank at a time, but the two pumps may deliver to the same destination simultaneously.

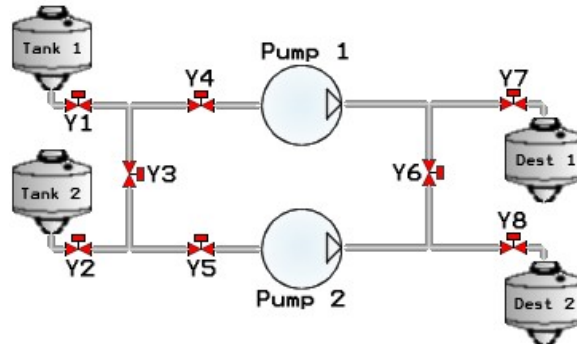


Figure 2

First, all parameters needed to define the rules must be listed inside the <Parameters> element. More parameters than actually needed may be listed to ease the understanding of the rules. Here, only the valves Y1-Y8 are required. The parameters From, Pump and Destination are included to ease the readability of the rules.

```
<Parameters>
  <Parameter Name="From" Type="Info"></Parameter>
  <Parameter Name="Pump" Type="Info"></Parameter>
  <Parameter Name="Destination" Type="Info"></Parameter>
  <Parameter Name="Y1" Type="Tank"></Parameter>
  <Parameter Name="Y2" Type="Tank"></Parameter>
  <Parameter Name="Y3" Type="Valve"></Parameter>
  <Parameter Name="Y4" Type="Pump"></Parameter>
  <Parameter Name="Y5" Type="Pump"></Parameter>
  <Parameter Name="Y6" Type="Valve"></Parameter>
  <Parameter Name="Y7" Type="Destination"></Parameter>
  <Parameter Name="Y8" Type="Destination"></Parameter>
</Parameters>
```

The Type attribute is optional. It will be stored by the RuleBase, but it's not used.

The order of which the parameters are listed in the <Parameters> section is important. It defines the syntax of how the rules must be entered. Like this:

```
<From Name=...>
  <Pump Name=...>
    <Destination Name=...></Destination>
    <Destination Name=...></Destination>
  </Pump>
</From>
```

Alternatively, like this:

The first parameters must be names of the xml elements that follows in the rule listing (see below). When no more elements are found, the RuleBase will try to find the rest of the parameters as attributes.

Study the following example rules:

```
<Rules>
  <!--
    OL = Open/Locked - Locked to open state.
    CL = Closed/Locked - Locked to closed state.
    OF = Open/Free, - Default open, but allowed to close
    CF = Closed/Free, - Default closed, but allowed to open.
  -->
  -->
  <From Name="Tank 1" Y1="OL" Y2="CF">
    <Pump Name="Pump 1" Y3="CF" Y4="OL" Y5="CF">
      <Destination Name="Dest 1" Y6="CF" Y7="OL" Y8="CL"></Destination>
      <Destination Name="Dest 2" Y6="OL" Y7="CL" Y8="OL"></Destination>
    </Pump>
    <Pump Name="Pump 2" Y3="OL" Y4="CF" Y5="OL">
      <Destination Name="Dest 1" Y6="OL" Y7="OL" Y8="CL"></Destination>
      <Destination Name="Dest 2" Y6="CF" Y7="CL" Y8="OL"></Destination>
    </Pump>
  </From>

  <From Name="Tank 2" Y1="CF" Y2="OL">
    <Pump Name="Pump 1" Y3="OL" Y4="OL" Y5="CF">
      <Destination Name="Dest 1" Y6="CF" Y7="OL" Y8="CL"></Destination>
      <Destination Name="Dest 2" Y6="OL" Y7="CL" Y8="OL"></Destination>
    </Pump>
    <Pump Name="Pump 2" Y3="CF" Y4="CF" Y5="OL">
```

```

    <Destination Name="Dest 1" Y6="OL" Y7="OL" Y8="CL"></Destination>
    <Destination Name="Dest 2" Y6="CF" Y7="CL" Y8="OL"></Destination>
  </Pump>
</From>
</Rules>

```

The first element must be an element called From

For each element, rule base parameters may be set as default values for the rules inside the element. For instance, in the example above, inside the first From element, there is an attribute `Y1="OL"`. This means that valve Y1 should have the value OL for all rules defined inside this From element. Comparing with the pipe diagram (figure 1), one can see that this makes sense.

The `<Rules>` table above defines 8 valid pump routes, that is 8 valid combinations of valve set-points. An implementation of a ValveControl component could use this rulebase to check if a requested combination of valve settings is valid (See ICD ValveControl.pdf).

Note again that the RuleBase component is useless by itself. It needs to be inherited and extended with methods that interpret the rules. For an example, see the ICD ValveControl component.

The rules could be specified in a different order. The ordering of the xml elements may be switched if the order of which the rulebase parameters are defined are changed accordingly. This may be useful because it often will lead to simpler rules.

1.5. The complete RuleBaseExample.xml

```

<?xml version="1.0" ?>
<Component Name="RuleBaseExample" Type="RuleBase">
  <InstanceHelp>RuleBase example file.</InstanceHelp>
  <Activate>3</Activate>
  <RuleBase>
    <!-- Lists all parameters that must be specified for a rule to be
         created. -->
    <!-- Elements must be specified first. -->
    <Parameters>
      <Parameter Name="From" Type="Info"></Parameter>
      <Parameter Name="Pump" Type="Info"></Parameter>
      <Parameter Name="Destination" Type="Info"></Parameter>
      <Parameter Name="Y1" Type="Tank"></Parameter>
      <Parameter Name="Y2" Type="Tank"></Parameter>
      <Parameter Name="Y3" Type="Valve"></Parameter>
      <Parameter Name="Y4" Type="Pump"></Parameter>
      <Parameter Name="Y5" Type="Pump"></Parameter>
      <Parameter Name="Y6" Type="Valve"></Parameter>
      <Parameter Name="Y7" Type="Destination"></Parameter>
      <Parameter Name="Y8" Type="Destination"></Parameter>
    </Parameters>
    <Rules>
      <!--
        OL = Open/Locked - Locked to open state.
        CL = Closed/Locked - Locked to closed state.
        OF = Open/Free, - Default open, but allowed to close
        CF = Closed/Free, - Default closed, but allowed to open.
      -->
      <From Name="Tank 1" Y1="OL" Y2="CF">
        <Pump Name="Pump 1" Y3="CF" Y4="OL" Y5="CF">
          <Destination Name="Dest 1" Y6="CF" Y7="OL" Y8="CL"></Destination>
          <Destination Name="Dest 2" Y6="OL" Y7="CL" Y8="OL"></Destination>
        </Pump>
        <Pump Name="Pump 2" Y3="OL" Y4="CF" Y5="OL">
          <Destination Name="Dest 1" Y6="OL" Y7="OL" Y8="CL"></Destination>
          <Destination Name="Dest 2" Y6="CF" Y7="CL" Y8="OL"></Destination>
        </Pump>
      </From>
      <From Name="Tank 2" Y1="CF" Y2="OL">
        <Pump Name="Pump 1" Y3="OL" Y4="OL" Y5="CF">
          <Destination Name="Dest 1" Y6="CF" Y7="OL" Y8="CL"></Destination>
          <Destination Name="Dest 2" Y6="OL" Y7="CL" Y8="OL"></Destination>
        </Pump>
        <Pump Name="Pump 2" Y3="CF" Y4="CF" Y5="OL">
          <Destination Name="Dest 1" Y6="OL" Y7="OL" Y8="CL"></Destination>
          <Destination Name="Dest 2" Y6="CF" Y7="CL" Y8="OL"></Destination>
        </Pump>
      </From>
    </Rules>
  </RuleBase>

```

```
<Subcomponents>  
</Subcomponents>  
  
<Signals>  
</Signals>  
  
<Alarms>  
</Alarms>  
  
</Component>
```