ProfiView

1.7

Configuration manual

Document ver. 1.07

History track:

Ver.	Date	Description
1.03	15.2.2009	Fixing distribution package with all basic features.
1.04	23.3.2009	Added new attributes to MOBJECT (multiplicator, decimalplaces, wholenumberdigits), added possibility to define value attribute of ALARM as decimal value, added possibility to define value of graphical and textdynamic objects as decimal value.
1.05	24.4.2010	Corrected typo, added polyline object.
1.06	25.6.2010	Font correction
1.07	28.6.2010	Post review corrections

Obsah

1. Purpose of this document

The purpose of this document is to understand the basic logic of ProfiView functionality and to describe and learn how to configure a project in ProfiView.This document doesn't describe installation of ProfiView. Installation is described in separated document called “Installation guide”.

2. What is ProfiView

ProfiView is software which can:

Visualize graphically and textually production process, measurement of quantities etc...
Monitors production status, process, running systems etc...
Alarming of unexpected situations, values etc...
Reporting of monitored process or measurement.

ProfiView is a cheap and effective tool with modular architecture, which is possible to use in any area, where you need to visualize or monitor some process or status, where you need regularly report the status or process and alarm unexpected or limit situations/values.

ProfiView, thanks to it's modularity, is able to communicate with any device, which offers digital interface with implemented any communication protocol. You just need to have the right module for ProfiView. ProfiView still extends the device portfolio. Ask for support of your device.

Typical application of ProfiView is connection to industry PLC, e.g. product of OMRON, read memory of PLC, and then visualize them in pictures or text or numbers on the screen.

Example of application: Monitoring and reporting of sewage plant process, monitoring of machine run in production hall and calculation of productivity, monitoring and visualization free parking places.

ProfiView is free to download on **http://profiview.maxpro.cz** (under shareware licence). You can purchase support, installation, configuration, development and more.

2.1. Technical parameters of ProfiView ver 1.07

- Multi OS software (Windows, Linux), tested for Windows XP, Windows Vista
- Easy configuration and application change
- Detailed documentation for installation and project configuration
- Supports PLC with RS232 cable interface
- Supports remote connection via AT modem, AT GSM modem.
- Supports e.g. OMRON PLC with CMODE protocol – for other go to **http://profiview.maxpro.cz**
- Detailed logs of ProfiView activity = easy trouble shooting and analysis
- Possibility to integrate your own communication modules (for developers)
- One installation can handle more implementations/configuration (projects)
- Supports more languages: English, German, Czech
- Free download, under Shareware licence

3. ProfiView Configuration files

There are 2 configuration files needed to start ProfiView:

1) log4j.properties – this file is described in ProfiView installation guide. ProfiView cannot run without correct log4j.properties file. It is recommended not to touch this except setting up the log level. See ProfiView installation guide document.
2) Profiview.properties – this file is also described in ProfiView installation guide. Without correct profiview.properties file the ProfiView cannot start.

4. ProfiView project configuration

Before we start to talk about project configuration, we have to explain what is the project. Project is a configuration file, saying what ProfiView should display on the screen, where to connect to external device, how to display/represent values received from external device on the monitor/screen. Project is usually saved in extra directoryas subdirectory of the general project directory which is part of ProfiView installation (home) directory.

ProfiView can have configured more then one project but can run only on one project at the moment. The information which project will be used by ProfiView is defined in `profiview.properties` file by parameter `PROFIVIEW_XML_CONFIGURATION_FILE`. This variable can have a value of one XML file only defining the project details. You can prepare more `profiview.bat` files where each file will contain specific XML file of specific project. But only one can be used to start ProfiView.

4.1. Project location

It is recommended but not mandatory, that all projects are saved in directory `./projects` where projects directory contains subdirectory for each project. Such directory is part of default ProfiView directory structure. Default installation package is distributed with one example project which should demonstrate all possible options that you can use on your project. The project directory path is `%PROFIVIEW_HOME%/projects/default`, where `%PROFIVIEW_HOME%` is the directory where ProfiView is installed. Inside default directory must be the `project.XML` file where the file name can be any name. It is the file pointed from `profiview.properties` file variable `PROFIVIEW_XML_CONFIGURATION`. Part of project are also picture files.

4.2. Project picture files

Project picture files can be JPG, PNG, GIF or BMP. Using other formats can work, but were not tested. Recommended are GIFs or PNGs. The picture files can be saved anywhere in computer but recommended is to keep them in project directory. It may be another subdirectory inside project directory.

5. Project XML file

Before we start we have to mention, that during startup of ProfiView is start process consistency check where part of this process is check of XML project configuration file. So if you change anything and suddenly ProfiView cannot start, look into the logfile. You most likely created inconsistency or mistake in project configuration file.

The `project .XML` file is defining the project. It is pure XML file according the XML definition. So you can use any XML features according to the standard. But generally it is recommended to make just one XML file without includes even the includes may work. The XML file is made of following sections:

1) `<PLCHANDLER>` – definition of communication with PLC module
2) `<MEMORYMANAGER>` – mapping of memory bytes from PLC into graphical objects in visualization screen
3) `<ALARMMANAGER>` – specifies alarms to be monitored and displayed on screen.
4) `<VISUALLOG>` – specifies parameters of graphical design of visual log window.
5) `<VISUALOBJECTS>` – definition of the graphical objects on the screen and it's behavior.

5.1. PROFIVIEW

Entire XML file is closed in tag `<PROFIVIEW>`. This tag can have optional attributes:

Name	Type	Presence	Description
leftupx	int	Optional*	position of let up corner of ProfiView main window (after application starts) – axes x
leftupy	int	Optional*	position of let up corner of ProfiView main window (after application starts) – axes y
width	int	Optional **	width of the main ProfiView window
height	int	Optional **	height of the main ProfiView window

*) If any of the attribute is missing, then the other attribute is ignored and default value is used.
**) If any of the attribute is missing, then the other attribute is ignored and default value is used.

If any of this attribute is missing, then is used the default position of the window (leftupx=0, leftupy=0, width = width of the screen, height=height of the screen). In other words the window is started by default as maximized window.
Example:

```
<PROFIVIEW width=1024 height=768 leftupx=0 leftupy=0>
       .
       .
       .
</PROFIVIEW>
```

5.2. Plc handler

This section is configuring communication with external device like PLC. If you want to communicate with PLC, you have to create section `<PLCHANDLER>` and inside you create sub section with any name you want. Mandatory is attribute of external module configuration called `<PLCHANDLER>` section where attribute `name` must exist and it's value is name of the module for the specific external device sometimes called “PLC handler”. The name is case sensitive. The same is valid if you don’t communication with external device directly but over a modem. Then the module (PLC handler) is able to talk to modem and over modem to PLC. The subsection of `<PLCHANDLER>` must be called with the same name like is the name of the software module for external device. But the name should be in capital letters. In our default project we use module which is talking to OMRON PLC over standard modem where the PLC uses CMODE serial protocol. Our configuration looks like following:

```
  <PLCHANDLER name="CModePlcModem">
       <CMODEPLCMODEM>
          <PORT>COM16</PORT>
          <BITRATE>9600</BITRATE>
          <DATABITS>8</DATABITS>
          <PARITY>0</PARITY>
          <STOPBITS>1</STOPBITS>
          <FLOWCONTROL>1</FLOWCONTROL>
          <CONNECT>
             <MODEM_COMMAND>
                    <COMMAND eol="yes">AT</COMMAND>
                    <ANSWER type="substring" timeout="30000">OK</ANSWER>
             </MODEM_COMMAND>
             <MODEM_COMMAND eol="yes">
                    <COMMAND>ATS6=1</COMMAND>
                    <ANSWER>OK</ANSWER>
             </MODEM_COMMAND>
             <MODEM_COMMAND>
                    <COMMAND eol="yes">ATDT494511256</COMMAND>
                    <ANSWER timeout="20000" type="substring">CONNECT</ANSWER>
             </MODEM_COMMAND>
          </CONNECT>
          <DISCONNECT>
             <MODEM_COMMAND>
                    <COMMAND eol="no">+++</COMMAND>
                    <ANSWER type="substring" timeout="15000">OK</ANSWER>
             </MODEM_COMMAND>
             <MODEM_COMMAND>
                    <COMMAND eol="yes">AT</COMMAND>
                    <ANSWER type="substring" timeout="10000">OK</ANSWER>
             </MODEM_COMMAND>
             <MODEM_COMMAND>
                    <COMMAND eol="yes">ATH</COMMAND>
                    <ANSWER type="substring" timeout="10000">OK</ANSWER>
             </MODEM_COMMAND>
          </DISCONNECT>
       </CMODEPLCMODEM>
</PLCHANDLER>
```

Which tags and attributes will be used for configuration of PLC handler depends on each PLC handler (module) implementation. The example used above is configuring plc handler CMODEPLCMODEM which is distributed as default PLC handler module with ProfiView. If you have your own customized PLC handler, you have to ask the developer or producer of such PLC handler to give you configuration details of such handler. In the configuration details must be described all parameters you have to write into project.XML file section <PLCHANDLER>.

In the example above (default distributed PLC handler module) we see that required parameters are:

PORT	defines serial port name
BITRATE	defines serial port bit rate in bits per second
DATABITS	number of data bits
PARITY	used parity
STOPBITS	used number of stop bits
FLOWCONTROL	used flow control mechanism
CONNECT	set of commands needed to connect including expected answers (if needed)
DISCONNECT	set of commands needed to disconnect including expected answers (if needed)

For more information see Appendix A – `CModePlcModem` configuration

5.3. Memory manager

Memory manager is a software component of ProfiView defining the memory bytes or areas monitored inside external device (PLC). Further more, it is also mapping this memory areas into graphical (visual) objects configured later in Visual objects section. Memory manager configuration is placed in section `<MEMORYMANAGER>`. Inside this section can exist only `<MOBJECT>` which can have attributes but can have empty body. Later we will describe visual objects, but now we can already say that more visual objects can have reference to one memory object. The attributes are:

Name	Type	Presence	Description
type	string	Mandatory	In this version of ProfiView can have only one value: “scheduled”
name	string	Mandatory	Name of the `MOBJECT`. Must be unique within entire `MEMORYMANAGER` configuration. The same number must be later used by visual object to connect memory byte or area with visual object representing the content of the memory cell. Byte or area
unitnumber	int	Mandatory	Unit number corresponds with unit number of the PLC. It used for building of PLC command. If only one PLC module is used, then the number is “0”.
command	string	Mandatory	Command used for reading of the expected memory byte or area content from external device (PLC).
address	Int	Mandatory	Memory address of the expected value in PLC memory.
refresh	int	Mandatory	Number of seconds defining the period of value refresh (period of reading from the external device memory).
format	string	Mandatory	Format of the number saved in PLC memory. It must have one of the following value: “hex”, “bcd”, “bit”, “int”, “uint”. If is used value “bit”, then must be used also optional attribute “bitid” in MOBJECT configuration.
bitid	int	Optional	Bit used as the value. Number must be “0” or higher. Value “0” = LSB.g
report	string	Optional	Can have value "yes" or "no", if the value is "yes" then this object is monitored by reporter log.
multiplicator	double	Mandatory	This value can be decimal value with decimal point “.” Which is used to multiply the value read from ext, device.
decimalplaces	int	Mandatory	*) Number of decimal digits behind decimal point. It also does rounding of the number. The value cannot be less than zero.
wholenumber digits	int	Mandatory	*) Number of all digits of the final value (after multiplication) excluding decimal point.

**) Wholenumber must be equal or higher then `decimalplaces`. If the number of digits in the number is less then `wholenumberdigits`, then the number is suffixed by blank spaces. If the number of digits in the number is more then `wholenumberdigits`, then number is not modified and is fully displayed.*

5.3.1. Example of MEMORYMANAGER configuration:

In this example we define MOBJECTs when using CMODE PLC from OMRON as part of the default ProfiView distribution. We will read bytes from “DM” memory (see OMRON manual for devices and CMODE protocol). PLC uses two bytes in DM memory for monitoring of water level in a tank and for run of water pump. The memory byte at address 0114 of DM memory contains value of the water level in BCD format and address 0107 contains information about water pump. But the value 0000 is divided to bits, where only bit 4 (starting from 0) is the one monitoring the water pump. Configuration of MEMORYMANAGER would then look like following:

```
<MEMORYMANAGER>
      <MOBJECT type="scheduled" name="N1" unitnumber="0" command="RD"
               address="0114" refresh="120" format="bcd" report=”yes”
               multiplicator=”0.01” decimalplaces=”2” whonumberdigits=”5”/>
      <MOBJECT type="scheduled" name="DHEL2A" unitnumber="0" command="RR"
               address="0107" refresh="120" format="bit" bitid="4"
               report=”no” multiplicator=”1” decimalplaces=”0”
               whonumberdigits=”3”/>
</MEMORYMANAGER>
```

5.4. Alarm Manager

Alarm manager works similar to Memory manager. It monitors specific values in external device and compares the values with predefined threshold. Definition of which value is monitored in external device, is done by definition of MOBJECT which must exist in configuration section for Memory manager.. It possible to define in which moment the Alarm manager should react and generate alarm. Alarms then exist for specific period of time, from last occurrence of the alarm. If alarm appears again before period is over, then the period calculation is started again.

Alarm manager, has it's own configuration section called `<ALARMMANAGER>` section which is part of `<PROFIVIEW>` section. Alarm manager has following mandatory attributes:

Name	Type	Presence	Description
visible	string	Optional	Defines if the window alarm manager will be visible after ProfiView start. The values can be “yes” or “no”. If the attribute is missing, then it means automatically “yes”. If the values is not understood, then it is automatically “no”.
refresh	int	Mandatory	Number of seconds defining period how often the ProfiView will check existence of an alarm. Alarms are connected with MOBJECTs so this refresh value should not be lower then refresh period of related MOBJECT.
leftupx	int	Mandatory	Position of window where alarms are managed – axes x.
leftupy	int	Mandatory	Position of window where alarms are managed – axes y.
width	int	Mandatory	Width of window where alarms are managed
height	Int	Mandatory	Height of window where alarms are managed

Inside ALARMMANAGER sections are defined ALARMs. Each ALARM has it’s own configuration. Alarm must have following attributes:

Name	Type	Presence	Description
name	string	Mandatory	Name of alarm used to display the alarm on the screen
mobjectname	string	Mandatory	Name of mobject monitored and whose value is alarmed
type	string	Mandatory	Type of monitoring of alarm. Can heave following value: **`thresholdup`** – alarm is generated when MOBJECT value is higher then “value” attribute. **`thresholddown`** – alarm is generated when MOBJECT value is lower then “value” attribute. **`equal`** – alarm is generated when MOBJECT value is equal to the “value” attribute.
value	double	Mandatory	This is the value used for decision whether the alarm exists or not.. Decimal value can be used (with decimal point “.”)
severity	Int	Mandatory	Defines urgency or importance of the alarm. Severity is also used for coloring the alarm in alarm monitor window. Low severity (no. 3) has yellow color, mid severity (no. 2) has orange color and high severity (no. 1) has red color.
maxage	int	Mandatory	Defines number of minutes as a maximal age of the alarm. During the time is the alarm displayed in alarm monitor window. If the defined number of minutes elapse from last observation of the alarm, then the alarm will disappear from the window.

audiofilename	string	Optional	This attribute specifies location of audio file, which will be played when Alarm is triggered. It will be played only once, it doesn't play in loop. Only WAV file format is supported in this version. .

Alarm body can contain description of the alarm plus some explanation what to do. This text is then displayed in alarm details window.

When alarm is triggered, then is displayed on the screen a window, where are visible details of alarm. If audio file is configured, then also audio sample will be played when dialog is shown. Only WAV file format is supported at the moment.

Example of ALARMMANAGER configuration:

```
<ALARMMANAGER visible="yes" refresh="10" leftupx="200" leftupy="200" width="400"
height="200">
        <ALARM name="LOW LEVEL" mobjectname="tank1" type="thresholddown"
value="3.2" severity="3" maxage="0"> Text to be displayed in the alarm detail. Can
contain advices what to do when the alarm is active.</ALARM>
        <ALARM name="Big tank 2" mobjectname="tank2" type="thresholddown"
value="3" severity="2" maxage="2"> Text to be displayed in the alarm detail. Can
contain advices what to do when the alarm is active.</ALARM>
        <ALARM name="Pump status" mobjectname="pump" type="thresholddown"
value="3" severity="1" maxage="1" audiofilename="/resources/alarm.wav"> Text to be
displayed in the alarm detail. Can contain advices what to do when the alarm is
active.</ALARM>
  </ALARMMANAGER>
```

5.4.1. Alarm reporting

Default configuration of ProfiView logs defines extra log files, where only alarms are reported. This is a text files, saved in %PROFIVIEW_INSTALL%/logs and the default name of the file is `alarms.log`. This file contains all occurances of all known alarms. It means, if some alarms occurs peridically and on the display is shown just once, with updated last occurance timestemp, then here in this file you can see all occurances. Example of one record (one line) from such file can look like follows:

```
2009.03.22;22:12:58:468 ;name=Status of pump mobjectname=pump type=thresholddown
value=3.0 severity=1 description= Text to be displayed in the alarm detail. Can
contain advices what to do when the alarm is active.
FirstObservationTimestamp=22.03.2009 22:12:40+0100
lastObservationTimestamp=22.03.2009 22:12:40+0100 last known value of mObject=0.0
```

In default log configuration are the `alamrs.log` files created automatically. Each file can have 1MB size max and ProfiView creates max 10 files. Both values can be modified (as well as name and location of the alarm log file) in `log4j.properties` configuration file in section `AlarmReporter`.

5.5. Visual log

ProfiView primarily logs to a file, where you can define level of detail what will be be logged. The maximal detail you can get with level `DEBUG`. Minimal information you can get with level `FATAL`. Other levels in between are `ERROR`, `WARN`, `INFO`. Visual log is actually just window, where you can display the same things that are logged to standard log file.

Visual log is configuration used for definition of parameters of Visual log window where are online visible activities of ProfiView. Configuration of the log entry in the Visual log window is done by configuration `log4j.properties` file. Don't try to change the definition of the log entry without deep knowledge of log4j configuration. Configuration in project XML configuration file is located in section `<VISUALLOG>`. This tag has following attributes:

Name	Type	Presence	Description
visible	string	Optional	Defines if the window visual log will be visible after ProfiView start. The values can be "yes" or "no". If the attribute is missing, then it means automatically "yes". If the values is not understood, then it is automatically "no".
leftupx	int	Mandatory	Position of window where log entries are displayed – axes x.
leftupy	int	Mandatory	Position of window where log entries are displayed – axes y.

width	int	Mandatory	Width of window where log entries are displayed
height	Int	Mandatory	Height of window where log entries are displayed
fontsize	int	Mandatory	Size of font used in visual log window.

Visual log doesn't have any body. Visual log is optional section. If the section is missing, then default values are used: `fontsize=10`, position of the window is on the right side of the screen.

Example configuration:

```
<VISUALLOG visible="yes" leftupx="600" leftupy="0" width="424" height="700"
fontsize="10"/>
```

5.6. Visual Manager

Visual manager is responsible for displaying of the valus on the screen. He contains objects, that represent the values in some format. The format can be picture (or animation-like graphics), text or number. All pictures and texts and numbers can be either statis (not changing) or dynamic (changing according to monitored value changes).Visual manager can also display background as specific color or as specific picture. On this background are then painted all visual objects (sometimes called “components”). Another static only type of object supported by this version is polyline which is painted as line made of line segments with defined positions..

Components for Visual manager are configured project XML configuration file, in section `<VISUALOBJECTS>`. Visual objects (components) must be placed in section `<VISUALOBJECTS>` and they are defined as `<OBJECT>`. Objects must have some attributes and may have some values. The values are then inside object as `<VALUE>` (as body of OBJECT). Visual objects section can also contain other general configuration values valid for visual manager. In the following text we will describe the general visual manager configuration parameters, visual objects and their values. Each visual object can refer to any memory object. More visual objects can refer to one memory object.

5.6.1. Visual manager configuration parameters

`<VISUALOBJECTS>` – this is section containing all definition for displaying of visual objects. This tag can have also attributes.

Name	Type	Presence	Description
width	String	Optional *	width of area used for painting and displaying of graphical objects
height	String	Optional *	height of area used for painting and displaying of graphical objects
backgroundcolor	String color	Optional	color of the background, where visual objects are located. Default color is white. Format of the color is R,G,B. Allowed value range is 0-255.

**) If any of the parameter is missing, then the other is ignored and default value is used for both.*

Area defined by this attributes is the area which is scroll able and defines default size of ProfiveView main window. If at least one attribute is missing, then these attributes are ignored and the painting area is defined by size of screen.

`<GLOBALOFFSET>` – this parameter is defining the offset of placement of each visual object on the screen. It is used for moving of the entire graphical presentation (all visual objects/components) across the screen without need of reconfiguration of placement of each visual object. This parameter must have two attributes:

x – offset in pixels in X axis
y – offset in pixels in Y axis

Example:

```
<VISUALOBJECTS width="1024" height="768" backgroundcolor="255,125,0">
        <GLOBALOFFSET x="0" y="0"/>
                .
                .
                .
</VISUALOBJECTS>
```

Each visual object (visual component) must be connected with one MOBJECT if is required to present actual values of MOBJECT. One MOBJECT can be referenced from more visual objects. If MOBJECT used in configuration of visual object is not found in configuration of MOBJECTs, then it is not an error. Just warning is generated and application can start in this case.

If visual object is connected to a MOBJECT, then it should be somehow defined what to display when then MOBJECT value is unknown (before/after external device is connected) or during reading error. Such value is usually called default value. The visual object type can be "`graphical`", "`intdynamic`", "`textdynamic`", "`polyline`". For each of the visual object type exist some mandatory and optional attributes. They are described in the following sections for each visual object type separately. Visual objects are defined by `<OBJECT>` section with specific attributes depending on the visual object type.

5.6.2. GRAPHICAL OBJECT - attributes

Graphical object is probably the most popular visual object. It represents monitored values as picture on the screen. There can be defined for each value (or value range) specific picture. Also default values is represented by picture.

OBJECT attributes – graphical object

Name	Type	Presence	Description
type	String	Mandatory	This parameter defines, what type is this object. Mandatory value is "**`graphical`**" – object made of picture(s) changing with MOBJECT value
name	String	Mandatory	Must be unique within VISUALOBJECTS configuration. If visual object must represent an MOBJECT, then name of visual object must be exactly the same like MOBJECT name. It is case sensitive. If you decide to use "mobject" parameter for definition of Mobject, then this name can be different from MOBJECT name.
mobject	string	Optional	Name of memory object related to this object. Should exist in list of MOBJECTS in configuration section for Memory manager. If this mobject attribute is not present in graphical object configuration, then value of "name" attribute is used instead.
width	Int	Mandatory	Number of pixels defining the picture width. If it is not the same like the picture width, then the ProfiView makes a scaling of the picture size.
height	Int	Mandatory	Number of pixels defining the picture height. If it is not the same like the picture height, then the ProfiView makes a scaling of the picture size.
leftupx	Int	Mandatory	Position of the left up corner of the picture on the screen. X value. This position can be influenced by GLOBALOFFSET parameter. The real position is then leftupx+GLOBALOFFSET (x).
leftupy	Int	Mandatory	Position of the left up corner of the picture on the screen. Y value. This position can be influenced by GLOBALOFFSET parameter. The real position is then leftupy+GLOBALOFFSET (y).
defimg	String	Mandatory	Path to the file which is used as default image for the object. Default image is used after the ProfiView starts (before it connects to the external device) and when ProfiView cannot read the real value or for any reason the value reading resulted with error.

5.6.3. GRAPHICAL Object - values

Each graphical object can have included objects called `<VALUE>`. The values are defining what picture will be used if the related MOBJECT will have specific value or value in specific range. If the MOBJECT value is out of the range then default image will be used (see graphical object attribute "defimg"). Each value has following attributes.

Name	Type	Presence	Description
from	Double	Mandatory	Low border of the value range. Decimal value can be used (with decimal point ".")
to	Double	Mandatory	High border of the value range. This value must be equal or

			higher then attribute “from”. Decimal value can be used (with decimal point “.”)
img	String	Mandatory	Path to the file which is used if related MOBJECT will be in the defined range. It is better to use absolute path, but relative path is also possible. Relative path will be then relative to directory where ProfiView is installed.

5.6.4. GRAPHICAL Object - example

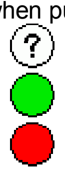
In this example is used configuration of the graphical object using a pictures that are representing the MOBJECT, where MOBJECT contains percentage value in range from 0 to 100 (inclusive). But the pictures will show only 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%. There is configured that:
0% = range from 0 to 3
10% = range from 4 to 14
20% = range from 15 to 24
Etc… It is more logical if e.g. 10% is in range from 4 to 14 instead of 1 to 10, because then value 10 of MOBJECT would mean 10% but value 11 would be already displayed as 20% which is not really correct.

```
     <OBJECT type="graphical" name="N1" width="65" height="110" leftupx="77"
leftupy="194" defimg="/projects/default/tankanalog_unknown.gif">
             <VALUE from="0" to="3" img="/projects/default/tankanalog_000.gif"/>
             <VALUE from="4" to="14" img="/projects/default/tankanalog_010.gif"/>
             <VALUE from="15" to="24" img="/projects/default/tankanalog_020.gif"/>
             <VALUE from="25" to="34" img="/projects/default/tankanalog_030.gif"/>
             <VALUE from="35" to="44" img="/projects/default/tankanalog_040.gif"/>
             <VALUE from="45" to="54" img="/projects/default/tankanalog_050.gif"/>
             <VALUE from="55" to="64" img="/projects/default/tankanalog_060.gif"/>
             <VALUE from="65" to="74" img="/projects/default/tankanalog_070.gif"/>
             <VALUE from="75" to="84" img="/projects/default/tankanalog_080.gif"/>
             <VALUE from="85" to="94" img="/projects/default/tankanalog_090.gif"/>
             <VALUE from="95" to="100" img="/projects/default/tankanalog_100.gif"/
>
          </OBJECT>
```

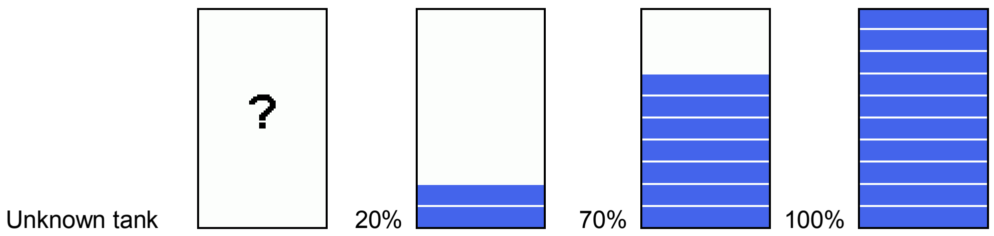
5.6.5. GRAPHICAL Object – How to use pictures

ProfiView accepts a few formats of picture: BMP, GIF, TIFF, JPG and NPG. More formats can be accepted by ProfiView, but the above mentioned are recommend. It is better to work with formats allowing to configure transparent color like GIF. **We strongly recommend to use GIFs**. Transparent background color allows the user to create library of pictures usable with any background. Example of picture from default projects are:

Pump – is made of three pictures. One is default with question mark displayed after start or when the pump status is unknown. Second picture is used when pump is ON and the third one is used when the pump is OFF.



Tank – is made of 12 pictures. One is for unknown status of tank and the rest is representing tank from 0% to 100%.



5.6.6. INTDYNAMIC Object - attributes

This object represents the real value of MOBJECT as a number on the screen on specific position with specific font color and size and with specific default value.

Name	Type	Presence	Description
type	string	Mandatory	This parameter defines, what type is this object. Mandatory value is "**intdynamic**" – and specifies that this object INTDYNAMIC object type - object displaying a number using some font and color, and is representing the concrete value of MOBJECT.
name	string	Mandatory	Must be unique within VISUALOBJECTS configuration. If this visual object must represent an MOBJECT, then name of visual object must be exactly the same like MOBJECT name. It is case sensitive. If you decide to use "mobject" parameter for definition of Mobject, then this name can be different from MOBJECT name.
mobject	string	Optional	Name of memory object related to this object. Should exist in list of MOBJECTS in configuration section for Memory manager. If this mobject attribute is not present in intdynamic object configuration, then value of "name" attribute is used instead.
leftupx	int	Mandatory	Position of the left down corner of the number on the screen. X value. This position can be influenced by GLOBALOFFSET parameter. The real position is then leftupx+GLOBALOFFSET (x).
leftupy	int	Mandatory	Position of the left down corner of the number on the screen. Y value. This position can be influenced by GLOBALOFFSET parameter. The real position is then leftupy+GLOBALOFFSET (y).
fontcolor	int,int,int	Mandatory	These three numbers represent RGB value of the color. All three numbers must be in range 0 – 255 (inclusive). The first integer is intensity of RED, second is GREEN, third is BLUE.
fontsize	int	Mandatory	Size of the font used on the screen.
deftext	string	Mandatory	This is the default text displayed instead of the real MOBJECT value. Default text is used after the ProfiView starts (before it connects to the external device) and when ProfiView cannot read the real value or for any reason the value reading resulted with error.
prefix	string	Optional	This string is all the time displayed before the integer value. E.g. is prefix is "[%]" then the final displayed value can look like "[%]100". Prefix is not displayed with default text.
suffix	string	Optional	This string is all the time displayed after the integer value. E.g. is suffix is "[%]" then the final displayed value can look like "100[%]". Suffix is not displayed with default text.

In object intdynamic (in this version of ProfiView) cannot be specified used font type. Only parameters of default font type can be specified like size and color.

Object intdynamic doesn't contain any VALUE object. If you need to display numbers but divided into ranges, then is better to use textdynamic OBJECT.

5.6.7. INTDYNAMIC Object - example

In this example we demonstrate logical value of `MOBJECT` called `tank1`, displayed in black color, with no prefix. Suffix is displayed after the int value, so the final result is then showing value in percent. Default text after start of ProfiView or during communication error is in this example "???". Suffix is not displayed with default text.

```
<OBJECT type="intdynamic" name="tank1value" mobject="tank1" leftupx="354"
leftupy="300" fontcolor="0,0,0" fontsize="12" deftext="???" prefix=""
suffix="[%]"/>
```

5.6.8. TEXTDYNAMIC Object - attributes

This type of object represents a text on the screen where the text must be predefined and must be mapped to specific range of MOBJECT value. There is possible to define position of the text, default text, assign to each MOBJECT value range a specific text, configure text size and text color. Typical example of using this component is translation of values 0 or 1 to text like "ON" or "OFF".

Name	Type	Presence	Description
type	string	Mandatory	This parameter defines, what type is this object. Mandatory value is "**textdynamic**" – and specifies that this object TEXTDYNAMIC object type - object displaying a values as text using some font and color, and the texts are representing the concrete value of MOBJECT.
name	string	Mandatory	Must be unique within VISUALOBJECTS configuration. If this visual object must represent an MOBJECT, then name of visual object must be exactly the same like MOBJECT name. It is case sensitive. If you decide to use "mobject" parameter for definition of Mobject, then this name can be different from MOBJECT name.
mobject	string	Optional	Name of memory object related to this object. Should exist in list of MOBJECTS in configuration section for Memory manager. If this mobject attribute is not present in intdynamic object configuration, then value of "name" attribute is used instead.
leftupx	int	Mandatory	Position of the left up corner of the text on the screen. X value. This position can be influenced by GLOBALOFFSET parameter. The real position is then leftupx+GLOBALOFFSET (x).
leftupy	int	Mandatory	Position of the left up corner of the text on the screen. Y value. This position can be influenced by GLOBALOFFSET parameter. The real position is then leftupy+GLOBALOFFSET (y).
fontcolor	int,int,int	Mandatory	These three numbers represent RGB value of the color. All three numbers must be in range 0 – 255 (inclusive). The first integer is intensity of RED, second is GREEN, third is BLUE.
fontsize	int	Mandatory	Size of the font used on the screen.
deftext	string	Mandatory	This is the default text displayed. Default text is used after the ProfiView starts (before it connects to the PLC) and when ProfiView cannot read the real value or for any reason the value reading resulted with error.

If there is not any `<VALUE>` object configured for textdynamic or if attribute "name" doesn't correspond with any MOBJECT, then this object behaves like a static text.

5.6.9. TEXTDYNAMIC Object - values

Each object defines a range in which will be displayed specific text with specific color.

Name	Type	Presence	Description
from	double	Mandatory	Low border of the value range. Decimal value can be used (with decimal point ".")
to	double	Mandatory	High border of the value range. This value must be equal or higher then attribute "from". Decimal value can be used (with decimal point ".")
text	string	Mandatory	Text to be displayed if the MOBJECT value is in range defined by "from" and "to".
fontcolor	int,int,int	Mandatory	The three number represent RGB value of the color. All three int numbers must be in range 0 – 255 (inclusive). The first integer is intensity of RED, second is GREEN, third is BLUE.

It is not possible to change size and font type of the text depending on the MOBJECT value.

5.6.10. TEXTDYNAMIC Object - example

In this example we demonstrate logical value of MOBJECT (format = bit) where bit = 1 means ON and bit = 0 means OFF. After starting of ProfiView, during disconnect from external device or during some error reading will be displayed “??” in blue color. If the bit is one, then is displayed ON in green otherwise OFF in red.

```
    <OBJECT type="textdynamic" name="EGGISstatus" leftupx="360" leftupy="446"
fontcolor="0,0,255" fontsize="12" deftext="??">
             <VALUE from="0" to="0" text="OFF" fontcolor="255,0,0"/>
             <VALUE from="1.0" to="1.99" text="ON"  fontcolor="0,255,0"/>
          </OBJECT>
```

5.6.11. POLYLINE Object - attributes

This type of object represents a line (or more precisely) polyline. It is standalone graphical object, which is not connected to any memory object. The purpose of this object is to quickly paint some lines with defined color and width on the screen. Polyline can be made of max 100 segments. It means 1 start coordinate plus 99 coordinates in VALUE section (see bellow).

Name	Type	Presence	Description
type	string	Mandatory	This parameter defines, what type is this object. Mandatory value is “**polyline**” – and specifies that this is object type POLYLINE.
name	string	Mandatory	Must be unique within VISUALOBJECTS configuration. This object is not connected to any memory object.
startupx	int	Mandatory	Position where the line starts. X value. This position can be influenced by GLOBALOFFSET parameter. The real position is then startupx+GLOBALOFFSET (x).
startupy	int	Mandatory	Position where the line starts. Y value. This position can be influenced by GLOBALOFFSET parameter. The real position is then startupy+GLOBALOFFSET (y).
linecolor	int,int,int	Mandatory	These three numbers represent RGB value of the color. All three int numbers must be in range 0 – 255 (inclusive). The first integer is intensity of RED, second is GREEN, third is BLUE.
linewidth	int	Mandatory	Width of the line.

If there is not any VALUE object configured for polyline, then the line is not painted.

5.6.12. POLYLINE Object - values

Each `<VALUE>` object (section) defines segment of the line. There can exist 100 number of segments, which means 1 start coordinate and 99 <VALUE> objects.

Name	Type	Presence	Description
x	int	Mandatory	Position (X value) to which will be painted a line segment.
y	int	Mandatory	Position (Y value) to which will be painted a line segment.

5.6.13. POLYLINE Object - example

In this example we demonstrate painting of square via black polyline.

```
<OBJECT type="polyline" name="line" startx="600" starty="600" linecolor="0,0,0"
linewidth="1" >
            <VALUE x="600" y="700"/>
            <VALUE x="700" y="700"/>
            <VALUE x="700" y="600"/>
            <VALUE x="600" y="600"/>
          </OBJECT>
```

6. National characters in configuration

In XML project configuration you can use text with national specific characters. ProfiView reads the file and the needs to display the national characters correctly on the screen. However each country defines some character set, which may cover other countries, but this is not a rule. Therefore you have to be careful in which coding you include the special characters into XML configuration file.

In XML configuration file exist the first line saying that this file is in XML format and says also specific encoding.
Example from default project is: `<?xml version="1.0" encoding="ISO-8859-1"?>`

This means that encoding of the XML file is in ISO-8859-1. There is a long list of all known character sets, therefore we do not place the list into this documentation. You can find the list on internet when you try to search for “XML encoding” using your favorite internet search engine like Google, Bing etc.

7. Setting of ProfiView language

ProfiView supports different languages. Definition of the language is in configuration file `profiview.properties`. In this file should exist parameter called LANGUAGE which can have following values in this version:

ENGLISH	– for English language
GERMAN	– for German language
CZECH	– for Czech language

If this parameter is not found by ProfiView during the application start, then English is used as default. The language cannot be changed during application run. After change of this configuration you have to restart ProfiView.

Example such configuration line, which should exist in `profiview.properties` file:

```
LANGUAGE=ENGLISH
```

If there is not your required language, please contact supplier of ProfiView at http://profiview.maxpro.cz.

8. Default project – example

As a part of ProfiView distribution package is distributed an example project. The project is very simple and there are demonstrated all existing visual objects and it’s configuration. There is also used DummyPlc which simulates connection over serial line directly to PLC. But in default project are visible configuration sections for all distributed communication modules e.g. CModePlcModem. (CModePlcModem is used to connect to PLC over a specific phone number via modem.) In the default project are configured dynamic texts to display some static texts. It means they are not connected to any of the three configured PLC memory words. To display water level in tanks as text is used visual object intdynamic. For displaying the water level in tanks as picture are used 12 pictures demonstrating the current water level. The range is from 0 to 100% with grid of 10%. For each level exists an extra picture showing the current water level. One picture is showing unknown water level. This picture is used after start of application before reading data or when some error occurred and the data cannot be read from external device. The default project source code you can find in Annex B.

9. Annex A – CModePlcModem configuration

This module is communicating with PLC from OMRON using serial line protocol CMODE and talking to PLC over standard serial modem. It means, that first the handler has to mange modem to connect remotely to PLC and once the connectivity is set up, it will start to communicate with PLC using CMODE protocol. To be able to use this modul, you have to configure PLCHANDLER attributes like following:

```
<PLCHANDLER name=”CModePlcModem”>
        <CMODEPLCMODEM>
                .
                .
                .
```

```
       </CMODEPLCMODEM>
</PLCHANDLER>
```

Inside the section CMODEPLCMODEM must be used following parameters with following possible values:

Serial line (between PC and modem) parameters:

PORT	string	port name like known in operating system
BITRATE	int	number of bits per second
DATABITS	int	number of data bits
PARITY	int	0 - no parity, 1 - even (default), 2 - odd, 3 - mark, 4 - space
STOPBITS	int	1 - 1 stop bit (default), 2 - 2 stop bits, 3 - 1.5 stop bits
FLOWCONTROL	int	1 - none (default), 2 - RTSCTS, 3 - RTSCTS - IN, 4 - RTSCTS - OUT, 5 - XONXOFF, 6 - XONXOFF - IN, 7 - XONXOFF – OUT

Other section is defining the set of commands needed to remotely connect to plc and later disconnect. It is organized into commands and expected answers. The order of the commands in the XML file defines the order of commands how they are sent to modem. Each command and answer have some attributes where each attribute has some allowed range of values:

Tag	attribute	type	values
COMMAND	eol	string	“**yes**” = after the command is sent to modem will be sent also end of line characters. “**no**” = after the command is sent to modem will not be sent end of line characters
ANSWER	type	string	“**substring**” = the value of given answer will be searched in the modem responses as a substring only (not as a full string) “**wholestring**” or missing attribute = value of given answer will be searched in the modem responses as a full string.
	timeout	int	Any number higher then zero. It is number of milliseconds when the module is waiting for the response from modem. In other words, it is the timeout of expected modem response.

Example of CModePlcModem handler configuration is following:

```
<PLCHANDLER name=”CModePlcModem”>
       <CMODEPLCMODEM>
          <PORT>COM16</PORT>
          <BITRATE>9600</BITRATE>
          <DATABITS>8</DATABITS>
          <PARITY>0</PARITY>
          <STOPBITS>1</STOPBITS>
          <FLOWCONTROL>1</FLOWCONTROL>
          <CONNECT>
             <MODEM_COMMAND>
                    <COMMAND eol="yes">AT</COMMAND>
                    <ANSWER type="substring" timeout="30000">OK</ANSWER>
             </MODEM_COMMAND>
             <MODEM_COMMAND eol="yes">
                    <COMMAND>ATS6=1</COMMAND>
                    <ANSWER>OK</ANSWER>
             </MODEM_COMMAND>
             <MODEM_COMMAND>
                    <COMMAND eol="yes">ATDT494511256</COMMAND>
                    <ANSWER timeout="20000" type="substring">CONNECT</ANSWER>
             </MODEM_COMMAND>
          </CONNECT>
          <DISCONNECT>
             <MODEM_COMMAND>
                    <COMMAND eol="no">+++</COMMAND>
                    <ANSWER type="substring" timeout="15000">OK</ANSWER>
             </MODEM_COMMAND>
             <MODEM_COMMAND>
                    <COMMAND eol="yes">AT</COMMAND>
                    <ANSWER type="substring" timeout="10000">OK</ANSWER>
             </MODEM_COMMAND>
```

```
              <MODEM_COMMAND>
                     <COMMAND eol="yes">ATH</COMMAND>
                     <ANSWER type="substring" timeout="10000">OK</ANSWER>
               </MODEM_COMMAND>
           </DISCONNECT>
        </CMODEPLCMODEM>
</PLCHANDLER>
```

10. Annex B – default project configuration

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PROFIVIEW>
  <PLCHANDLER name="CModePlcModem">
       <CMODEPLCMODEM>
          <PORT>COM16</PORT>
          <BITRATE>9600</BITRATE>
          <DATABITS>8</DATABITS>
          <PARITY>0</PARITY> <!-- 0 - no parity, 1 - even (default), 2 - odd, 3 - mark, 4 -
space-->
          <STOPBITS>1</STOPBITS> <!-- 1 - 1 stop bit (default), 2 - 2 stop bits, 3 - 1.5 stop
bits-->
          <FLOWCONTROL>1</FLOWCONTROL><!-- 1 - none (default), 2 - RTSCTS, 3 - RTSCTS - IN, 4
- RTSCTS - OUT, 5 - XONXOFF, 6 - XONXOFF - IN, 7 - XONXOFF - OUT-->
          <CONNECT>
             <MODEM_COMMAND><COMMAND eol="yes">AT</COMMAND><ANSWER type="substring"
timeout="30000">OK</ANSWER></MODEM_COMMAND> <!-- eol="eys or "no" yes, meanse that after
sending a command to modem will be automatically sent end of line like 0x0A 0x0D. -->
             <MODEM_COMMAND
eol="yes"><COMMAND>ATS6=1</COMMAND><ANSWER>OK</ANSWER></MODEM_COMMAND>
             <MODEM_COMMAND><COMMAND eol="yes">ATDT541233445</COMMAND><ANSWER timeout="20000"
type="substring">CONNECT</ANSWER></MODEM_COMMAND> <!-- Not case sensitive, type =
"wholestring" or "substring", timeout in [ms] default value 15000 -->
          </CONNECT>
          <DISCONNECT>
             <MODEM_COMMAND><COMMAND eol="no">+++</COMMAND><ANSWER type="substring"
timeout="15000">OK</ANSWER></MODEM_COMMAND>
             <MODEM_COMMAND><COMMAND eol="yes">AT</COMMAND><ANSWER type="substring"
timeout="10000">OK</ANSWER></MODEM_COMMAND>
             <MODEM_COMMAND><COMMAND eol="yes">ATH</COMMAND><ANSWER type="substring"
timeout="10000">OK</ANSWER></MODEM_COMMAND>
          </DISCONNECT>
       </CMODEPLCMODEM>
  </PLCHANDLER>
  <ALARMMANAGER visible="yes" refresh="10" leftupx="200" leftupy="200" width="400"
height="200">
        <ALARM name="High watter" mobjectname="tank1" type="thresholdup" value="90"
severity="3" maxage="1" audiofilename="/resources/tada.wav" > Tank 1 is almost full. Close
valve 2 to avoid overfull.</ALARM>
        <ALARM name="Low watter" mobjectname="tank2" type="thresholddown" value="20"
severity="2" maxage="1" audiofilename=""> Tank 2 is almost empty. Open valve 1 to speedup
filling.</ALARM>
        <ALARM name="Pump 1 status" mobjectname="pump1" type="equal" value="5" severity="1"
maxage="1" audiofilename=""> Pump one has status 5.</ALARM>
  </ALARMMANAGER>
  <VISUALLOG visible="Yes" leftupx="600" leftupy="0" width="424" height="700" fontsize="10"/>
  <MEMORYMANAGER>
      <MOBJECT type="scheduled" name="tank1" unitnumber="0" command="RD" address="0114"
refresh="10" format="bcd" report="yes" multiplicator="10" decimalplaces="0"
wholenumberdigits="3"/>
      <MOBJECT type="scheduled" name="tank2" unitnumber="0" command="RD" address="0115"
refresh="10" format="bcd" report="yes" multiplicator="10" decimalplaces="0"
wholenumberdigits="3"/>
      <MOBJECT type="scheduled" name="pump1" unitnumber="0" command="RD" address="0116"
refresh="10" format="bcd" report="yes" multiplicator="1" decimalplaces="0"
wholenumberdigits="3"/>
      <MOBJECT type="scheduled" name="pump2" unitnumber="0" command="RD" address="0117"
refresh="10" format="bcd" report="yes" multiplicator="1" decimalplaces="0"
wholenumberdigits="3"/>
  </MEMORYMANAGER>
  <VISUALOBJECTS backgroundcolor="255,255,255">
       <GLOBALOFFSET x="0" y="0"/>
          <OBJECT type="textdynamic" name="headline" leftupx="378" leftupy="63"
fontcolor="0,0,0" fontsize="24" deftext="DEFAULT PROJECT" />
```

```
          <OBJECT type="textdynamic" name="secondheadline" leftupx="378" leftupy="107"
fontcolor="0,0,0" fontsize="16" deftext="Optimized for screen 1024x768" />

          <OBJECT type="graphical" name="tank1" mobject="tank1" width="65" height="110"
leftupx="280" leftupy="198" defimg="/projects/default/tankanalog_unknown.gif">
             <VALUE from="0" to="4" img="/projects/default/tankanalog_000.gif"/>
             <VALUE from="5" to="14" img="/projects/default/tankanalog_010.gif"/>
             <VALUE from="15" to="24" img="/projects/default/tankanalog_020.gif"/>
             <VALUE from="25" to="34" img="/projects/default/tankanalog_030.gif"/>
             <VALUE from="35" to="44" img="/projects/default/tankanalog_040.gif"/>
             <VALUE from="45" to="54" img="/projects/default/tankanalog_050.gif"/>
             <VALUE from="55" to="64" img="/projects/default/tankanalog_060.gif"/>
             <VALUE from="65" to="74" img="/projects/default/tankanalog_070.gif"/>
             <VALUE from="75" to="84" img="/projects/default/tankanalog_080.gif"/>
             <VALUE from="85" to="94" img="/projects/default/tankanalog_090.gif"/>
             <VALUE from="95" to="100" img="/projects/default/tankanalog_100.gif"/>
          </OBJECT>
          <OBJECT type="graphical" name="tank2" mobject="tank2" width="65" height="110"
leftupx="642" leftupy="198" defimg="/projects/default/tankanalog_unknown.gif">
             <VALUE from="0" to="4" img="/projects/default/tankanalog_000.gif"/>
             <VALUE from="5" to="14" img="/projects/default/tankanalog_010.gif"/>
             <VALUE from="15" to="24" img="/projects/default/tankanalog_020.gif"/>
             <VALUE from="25" to="34" img="/projects/default/tankanalog_030.gif"/>
             <VALUE from="35" to="44" img="/projects/default/tankanalog_040.gif"/>
             <VALUE from="45" to="54" img="/projects/default/tankanalog_050.gif"/>
             <VALUE from="55" to="64" img="/projects/default/tankanalog_060.gif"/>
             <VALUE from="65" to="74" img="/projects/default/tankanalog_070.gif"/>
             <VALUE from="75" to="84" img="/projects/default/tankanalog_080.gif"/>
             <VALUE from="85" to="94" img="/projects/default/tankanalog_090.gif"/>
             <VALUE from="95" to="100" img="/projects/default/tankanalog_100.gif"/>
          </OBJECT>
          <OBJECT type="graphical" name="tank1_bottom" width="65" height="33" leftupx="280"
leftupy="308" defimg="/projects/default/tank_bottom.gif" />
          <OBJECT type="graphical" name="tank2_bottom" width="65" height="33" leftupx="642"
leftupy="308" defimg="/projects/default/tank_bottom.gif" />

          <OBJECT type="graphical" name="mainpump" mobject="pump1" width="22" height="22"
leftupx="354" leftupy="147" defimg="/projects/default/pump_small_unknown_3.gif">
             <VALUE from="0" to="5" img="/projects/default/pump_small_OFF.gif"/>
             <VALUE from="6" to="10" img="/projects/default/pump_small_ON.gif"/>
          </OBJECT>

          <OBJECT type="graphical" name="pump2" mobject="pump2" width="22" height="22"
leftupx="612" leftupy="147" defimg="/projects/default/pump_small_unknown_3.gif">
             <VALUE from="0" to="5" img="/projects/default/pump_small_OFF.gif"/>
             <VALUE from="6" to="10" img="/projects/default/pump_small_ON.gif"/>
          </OBJECT>

          <OBJECT type="graphical" name="valve1" width="22" height="22" leftupx="354"
leftupy="370" defimg="/projects/default/valve.gif" />
          <OBJECT type="graphical" name="valve2" width="22" height="22" leftupx="612"
leftupy="370" defimg="/projects/default/valve.gif" />


          <OBJECT type="intdynamic" name="tank1value" mobject="tank1" leftupx="354"
leftupy="300" fontcolor="0,0,0" fontsize="12" deftext="???" prefix="" suffix="[%]"/>
          <OBJECT type="intdynamic" name="tank2value" mobject="tank2" leftupx="716"
leftupy="300" fontcolor="0,0,0" fontsize="12" deftext="???"/>
          <OBJECT type="textdynamic" name="tank2valuepercent" leftupx="739" leftupy="300"
fontcolor="0,0,0" fontsize="14" deftext="%"  />

          <OBJECT type="polyline" name="pipe1" startx="312" starty="198" linecolor="0,0,0"
linewidth="1" >
            <VALUE x="312" y="158"/>
            <VALUE x="419" y="158"/>
            <VALUE x="568" y="381"/>
            <VALUE x="674" y="381"/>
            <VALUE x="674" y="339"/>
          </OBJECT>
          <OBJECT type="polyline" name="pipe2" startx="312" starty="339" linecolor="0,0,0"
linewidth="1" >
            <VALUE x="312" y="381"/>
            <VALUE x="419" y="381"/>
            <VALUE x="568" y="158"/>
            <VALUE x="674" y="158"/>
            <VALUE x="674" y="198"/>
          </OBJECT>
          <OBJECT type="textdynamic" name="mobjectstitle" leftupx="348" leftupy="450"
fontcolor="0,0,0" fontsize="14" deftext="MObjects" />
          <OBJECT type="textdynamic" name="tank1text" leftupx="440" leftupy="450"
fontcolor="0,0,0" fontsize="14" deftext="Tank 1 :" />
          <OBJECT type="textdynamic" name="tank2text" leftupx="440" leftupy="472"
fontcolor="0,0,0" fontsize="14" deftext="Tank 2 :" />
          <OBJECT type="textdynamic" name="pump1text" leftupx="440" leftupy="494"
fontcolor="0,0,0" fontsize="14" deftext="Pump 1 :" />
          <OBJECT type="textdynamic" name="pump2text" leftupx="440" leftupy="516"
fontcolor="0,0,0" fontsize="14" deftext="Pump 2 :" />

          <OBJECT type="intdynamic" name="tank1valuetext" mobject="tank1" leftupx="530"
leftupy="450" fontcolor="0,0,0" fontsize="14" deftext="unknown" />
          <OBJECT type="intdynamic" name="tank2valuetext" mobject="tank2" leftupx="530"
leftupy="472" fontcolor="0,0,0" fontsize="14" deftext="unknown" />
          <OBJECT type="intdynamic" name="pump1valuetext" mobject="pump1" leftupx="530"
leftupy="494" fontcolor="0,0,0" fontsize="14" deftext="unknown" />
          <OBJECT type="intdynamic" name="pump2valuetext" mobject="pump2" leftupx="530"
leftupy="516" fontcolor="0,0,0" fontsize="14" deftext="unknown" />

          <OBJECT type="textdynamic" name="footer" leftupx="310" leftupy="565"
fontcolor="0,0,0" fontsize="14" deftext="Simulation - random values only - using DummyPlc
handler." />

  </VISUALOBJECTS>
</PROFIVIEW>
```