



Asix4Internet
Manual

ASKOM[®] and **asix**[®] are registered trademarks of ASKOM Spółka z o.o., Gliwice. Other brand names, trademarks, and registered trademarks are the property of their respective holders.

All rights reserved including the right of reproduction in whole or in part in any form. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without prior written permission from the ASKOM.

ASKOM sp. z o. o. shall not be liable for any damages arising out of the use of information included in the publication content.

Copyright © 2005, ASKOM Sp. z o. o., Gliwice



ASKOM Sp. z o. o., ul. Józefa Sowińskiego 13, 44-121 Gliwice,
tel. +48 (0) 32 3018100, fax +48 (0) 32 3018101,
<http://www.askom.com.pl>, e-mail: office@askom.com.pl

1. Asix4Internet - visualisation and supervision via Internet

The **asix** 4 package contains tools that allow visualisation and supervision of execution of industrial processes with use of web browser. These are new modules related to the Internet:

AsPortal – Process Information Portal,

As2HTML – Library of scripts and CSS styles, which allows easy creation of the application for which visualisation is displayed in Internet Explorer 6.0,

As2WWW - Converter of **asix** system application into Web application.

To start the modules and use them on a local level, the **asix** 4 HASP key is necessary. To use the modules remotely, the **asix** HASP key extended by web user licence is necessary.

2. Requirements of Internet modules

2.1. Requirements of asix system

When web modules operate with **asix** system working on the same computer, all types of **asix** system licences are handled. The exception is access to alarms – then **asix** in *operator server* version (symbol *WAxS*) is required.

When **asix** system works on another computer, all versions of **asix** system licences are handled (including the former versions working under DOS operating system), and the required type of **asix** system licence is *operator server* (symbol *WAxS*).

2.2. Requirements of Windows system

For operation of web modules Windows 2000, Windows XP Professional or Windows 2003 operating system is required.

2.3. Configuration of operating system

In Windows XP system, it is necessary to disable the *Simple file disclosure* function. To do so, in the *Start* menu select *My computer* – this opens the ‘*My computer*’ window. After the window is opened, select *Folder options* in the *Tools* menu, and then the *View* tab. Find the *Use simple file disclosure* option in the *Advanced settings* list and disable this option.

In the operating system, it is necessary to install WWW server (by default, it is not installed during installation of the systems).

The other requirement is installation of the .NET 1.1 package. This package can be downloaded from www.microsoft.com or **asix** software installation CD.

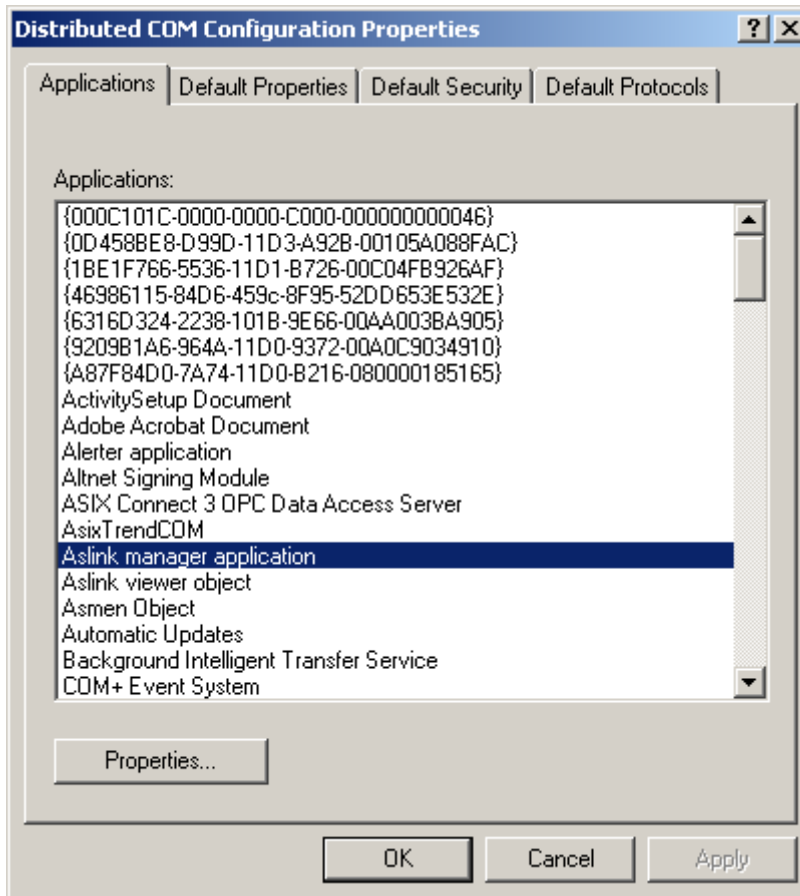
If during the installation of WWW server the .NET 1.1 package has already been installed in the system, the .NET package must be removed and installed once again.

2.4. Configuring system access rights to Aslink module

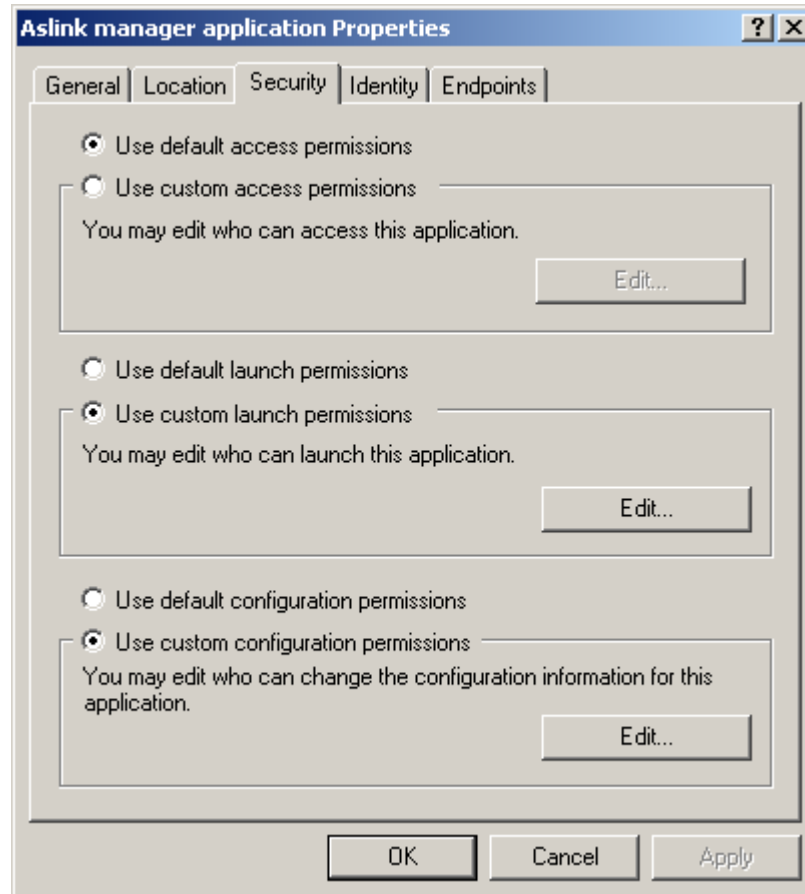
After installation of **asix** package has been completed, it is necessary to configure the system access rights to Aslink module. The configuration procedure for Windows 2000 and XP systems is described in the following steps. In case of Windows 2003 the procedure is identical – however, the rights are assigned to *NETWORK SERVICE* user (not to *ASPNET* user).

STEP 1

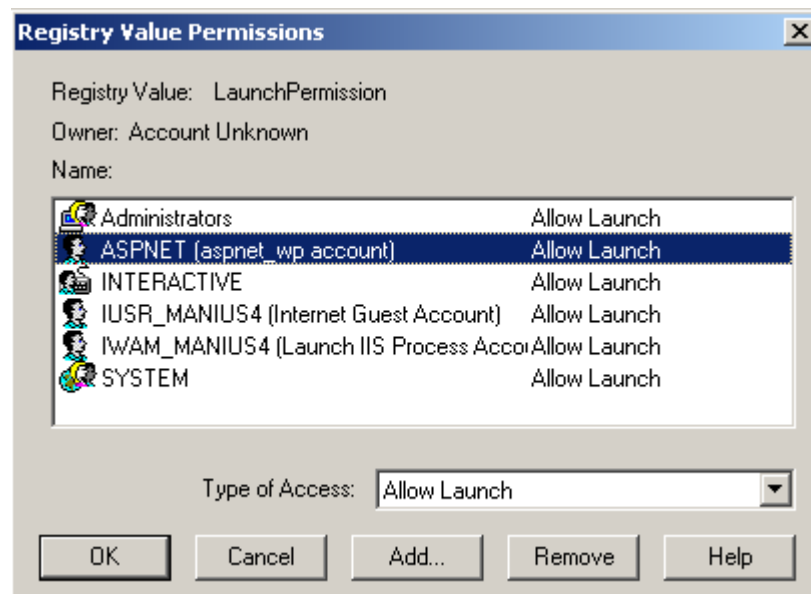
Run *dcomcnfg.exe*.

**STEP 2**

In the main window of the program, in the *Applications* tab, find the *Aslink manager application* item and select *Properties*.

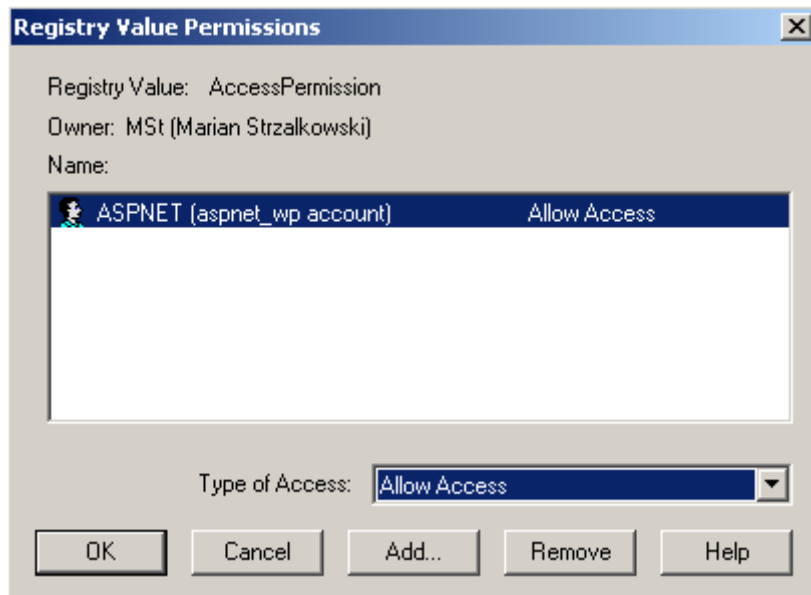
**STEP 3**

In the *Protections* tab, after the *Use custom launch permissions* option is selected and *Edit* button is pressed, the window will be opened where the ASPNET user should be assigned the *Allow launch* rights.

**STEP 4**

Similarly (the '*Properties: Aslink manager Application*'/*Protections* tab), the ASPNET user should be assigned access rights (after the *Use custom access*

permissions option is selected and *Edit* button is pressed, the 'Registry Value Permissions' window will be opened).

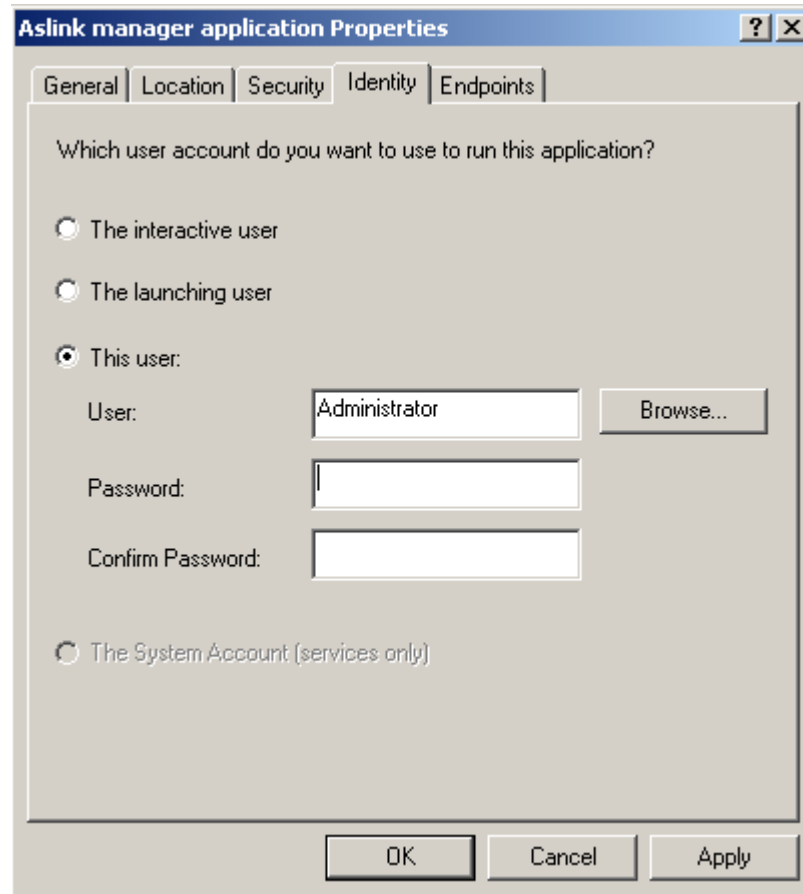


NOTE:

To save the modifications in authorisations, they should be accepted by clicking on *Apply* ('Aslink manager application Properties' window).

STEP 5

In the next step, identity of the user who will be authorised to start the Aslink module should be defined. It is done in the *Identity* tab in the 'Aslink manager application Properties' window. The username will be *Administrator*. The relevant password should be entered. If **asix** system is not operating on the given computer, the option *Starting user* may also be selected. The modifications are accepted by clicking on the button *Apply*.



2.5. Access to asix system directory

In Windows 2000 and XP operating system, *ASPNET* user should be assigned full access rights to directory **asix** system is installed in (by default, c:\asix). In Windows 2003, full access rights to this directory should be assigned to *NETWORK SERVICE* user.

2.6. Generating VariableBase

When generating VariableBase for the needs of AsixConnect 4 and AsPortal package (with use of VariableBase Manager), you should also carry out the operation of preparing the variablebase for .NET/AsPort.

2.7. Access to VariableBase

In Windows 2000 and XP operating system, *ASPNET* user should be assigned full access rights to directory the VariableBase is installed in. In Windows 2003, full access rights to this directory should be assigned to *NETWORK SERVICE* user.

2.8. Requirements for designing the application using As2HTML

Designer's tools:

- Notebook;
- Microsoft ASP.NET WebMatrix 0.6;
- FrontPage;
- Visual Studio 2003;
- any other HTML web page editor;
- in addition, the basic knowledge of HTML language is necessary.

3. Web Service server

The *Web Service* server of AsixConnect Server package provides access to full functionality of **asix** system application via *XML Web Services* protocol. The Web Service server is used as a data source by applications developed using As2HTML and As2WWW modules.

3.1. Installation

The *WebService* server is located in *c:\asix\WebService* directory. In order to make it available within the network, run the *Web information services* program. This program is available in *Start/Control panel/Administrating tools*. In the program window, highlight the *Default website* item. Select *Virtual directory* in the *Action/New* menu. Wizard is started. As *Alias* you should enter *WebService*, and as *Directory* – *c:\asix\WebService*. The other options are to be left unchanged. By default, on attempt of getting access to the server, Windows authentication is used. In order to enable the anonymous access, highlight the newly created virtual directory, select *Properties* in the *Action* menu, open the *Protections of directories* tab, click on *Edit* in the *Anonymous access* field, and enable the *Anonymous access* option.

Since then the Web Service server is available from:
http://computer_name/WebService/XConnectWebService.asmx.

See:
http://computer_name/WebService/XConnectWebService.asmx?WSDL for description of the server services in WSDL language.

3.2. Web.Config configuration file

For storage of default channel name the Web Service server uses the configuration file named *Web.Config*. This file located in *c:\asix\WebService* directory. Method of defining the channels is described in AsixConnect package documentation.

In order to define the default channel name, you should create the *appSettings* element in *Web.Config* file in the superior element. Then, one *add* element should be created in the *appSettings* element and two attributes should be defined in it. The first attribute should be named *key* and assigned the value *"DefaultChannelName"*. The latter attribute should be named *value* and assigned the channel name as the value. The channel name should be put in quotation marks.

EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="DefaultChannelName" value="AsEmis" />
  </appSettings>
```

For channel definition the *Control variable* and *Limits of variables* options should be activated.

4. AsPortal - Process Information Portal

The process data portal allows reviewing any process data, i.e. VariableBase of applications, current and archive values of measurements, archive of historical alarms as well as list of active alarms in tabular form and historical data in the form of charts. The portal is a universal application and in order to connect it to any application of **asix** system it is enough to configure the path to the VariableBase.

4.1. Configuration

4.1.1. Giving access to applications on WWW server

In order to make application available you need to:

- provide *c:\Asix\AsPort* as virtual directory of WWW server with default options (accept default name of *AsPort*),
- if rights to log into the Windows server have not been configured for the application users, then anonymous access should be activated in protections of both the virtual directories.

After the above-mentioned steps have been carried out, the application is available on the local level after the following address is typed in IE6 browser:

http://localhost/AsPort

The application is available within the local area network at

http://<name of computer in Windows system>/AsPort

For example, if computer name is *AsixWeb*, the application address is:

http://AsixWeb/AsPort

4.1.2. Web.config configuration file

The application makes it possible to define the list of displayed attributes separately for the current data, archive data as well as VariableBase and variable selection window. To this end, the lists of attributes should be provided in relevant items of the *Web.config* file:

<i>CTTableAttributesList</i>	–attributes in window of current data,
<i>HTTableAttributesList</i>	–attributes in window of archive data presented in tabulated form,
<i>HTChartsAttributesList</i>	–attributes in window of archive data presented as diagram,
<i>CurrentDataAttributesList</i>	–attributes displayed in the variable selection window and the variablebase

There are two special names of attributes. In case of the current data it is *ValueCV*, which means the current value of measurements. In windows of historical data, the particular name of attribute is *Aggregate*, which means the column of aggregates.

EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key = "CurrentDataAttributesList" value = "Name,Description,Unit,AKPiAName" />
    <add key = "CTTableAttributesList" value = "Name,AKPiAName,Description,ValueCV,Unit " />
    <add key = "HTTableAttributesList" value = "Name, Aggregate, Description"/>
    <add key = "HTChartsAttributesList" value = "Name, Aggregate, LoMeasurementRange " />
  </appSettings>
```

In the *Web.config* file, you can also set the number of lines displayed on the basis of the variablebase on one page. It is defined by the *PageSize* item. When 0 is entered, paging is disabled.

EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key = "PageSize" value = "8" />
  </appSettings>
```

4.2. Description of functions available to the user

4.2.1. Current data

The current data window is used to display attributes and values of current data of **asix** system. Current data are displayed in tabulated form. The column containing the current values of variables is formatted according to measurement quality.

For good quality:

- when warning limit is exceeded, the value is displayed in red font,
- when alarm limit is exceeded, the value is displayed in red font against yellow background,

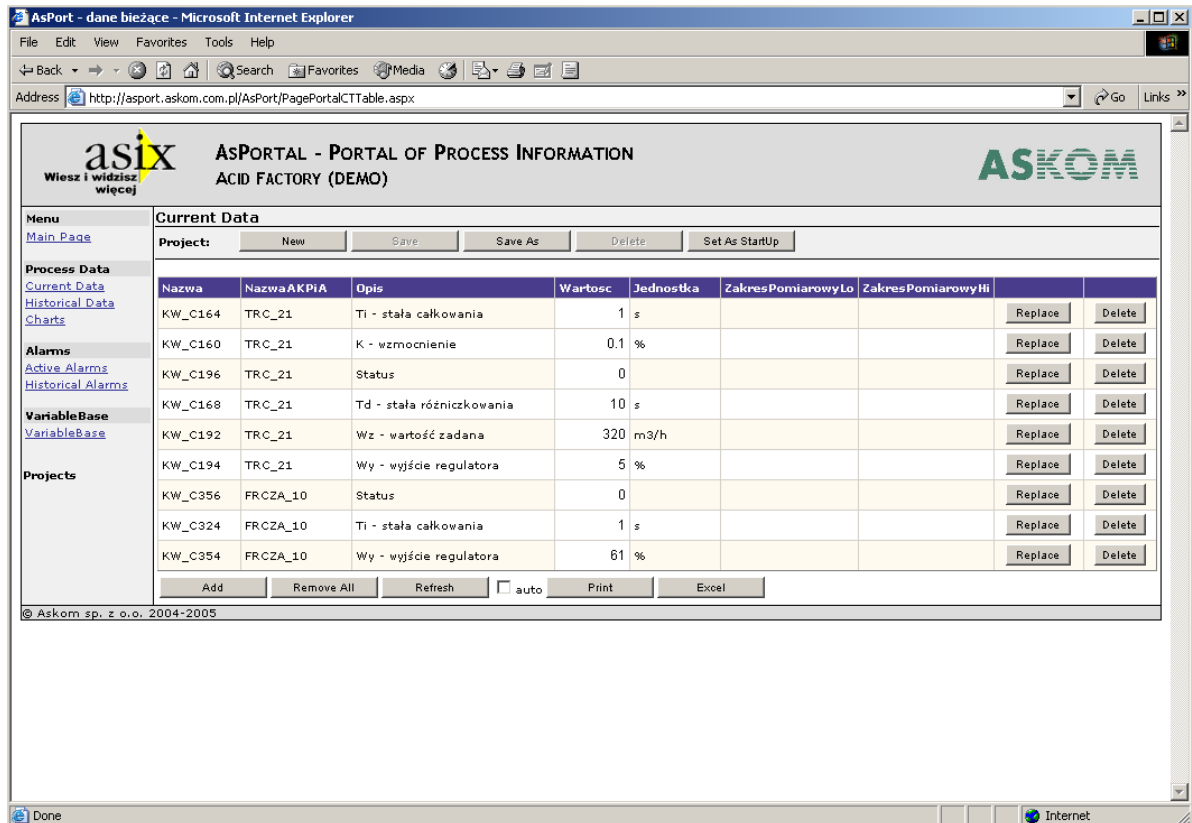
For bad quality:

- value is displayed in white font against red background,
- when sensor failure is detected, "??" is displayed in place of value,
- in case of communication error, the last known value with "?" character is displayed,
- in other cases the "?" character is displayed.

The available operations that are carried out in this window are:

- *Add Variables* – opens the window for selecting new variables (described in details in *4.2.5 Variable Selection Window*),
- *Refresh* – retrieves the current values of variables and displays them on the screen,
- *Remove* – removes a variable,

- *Replace* – opens the variable selection window; the newly selected variable is inserted in place of the former one.



4.2.2. Archive data

Windows of archive data allow presentation of historical values of measurements in tabulated form or as a chart.

In order to display archive data, the following fields should be set to define the period from which data are to be read:

Obszarowy	2004-12-07	11:05	2 minuty	30 sekund	<<	>>	>
Type of chart (does not regard table)	Beginning date of chart	Beginning time of chart	Length of period	Aggregation interval			

The menu buttons have the following functions:

- „<<” – resetting time by a length backwards,
- „>>” – resetting time by a length forwards,
- „>|” – setting the end of chart to the current moment.

The available operations to be carried out in the archive data window are:

- *Add* – opens the window for selecting new variables (described in details in 4.2.5 *Variable Selection Window*),
- *Remove all* – removes all variables,
- *Refresh* – application of changes and displaying the table/chart,
- *Export to table/chart* – exports the current collection of variables to the table/chart,

- *Remove* – removes a variable,
- *Replace* – opens the variable selection window; the newly selected variable is inserted in place of the former one.

Archive data displayed in tabulated form

The screenshot shows the AsPortal web interface in a Microsoft Internet Explorer browser window. The page title is "AsPortal - dane historyczne - Microsoft: Internet Explorer". The address bar shows "http://asport.askom.com.pl/AsPort/PagePortalHTTable.aspx".

The main content area is titled "ASPORTAL - PORTAL OF PROCESS INFORMATION ACID FACTORY (DEMO)". It features a "Historical Data - Table" section with a "Project:" dropdown set to "New". Below this, there are input fields for "Date:" (9.6.2005), "Time(hh:mm):" (13:34), "Length:" (15 minutes), and "Interval:" (1 minute).

A table lists variables with their aggregate, unit, and description:

Nazwa	Aggregate	Jednostka	Opis	Replace	Delete
KW_A000	Start	°C	Temperatura spalin przed odemglaczem	Replace	Delete
KW_A004	Start	°C	Temperatura kwasu siarkowego	Replace	Delete
KW_A008	Start	°C	Temperatura wody ciepłej	Replace	Delete
KW_A032	Start	°C	Temperatura H2S przed piecem	Replace	Delete

Below the table, there are buttons for "Add", "Remove All", "Show Data", navigation arrows, "Print", and "Excel".

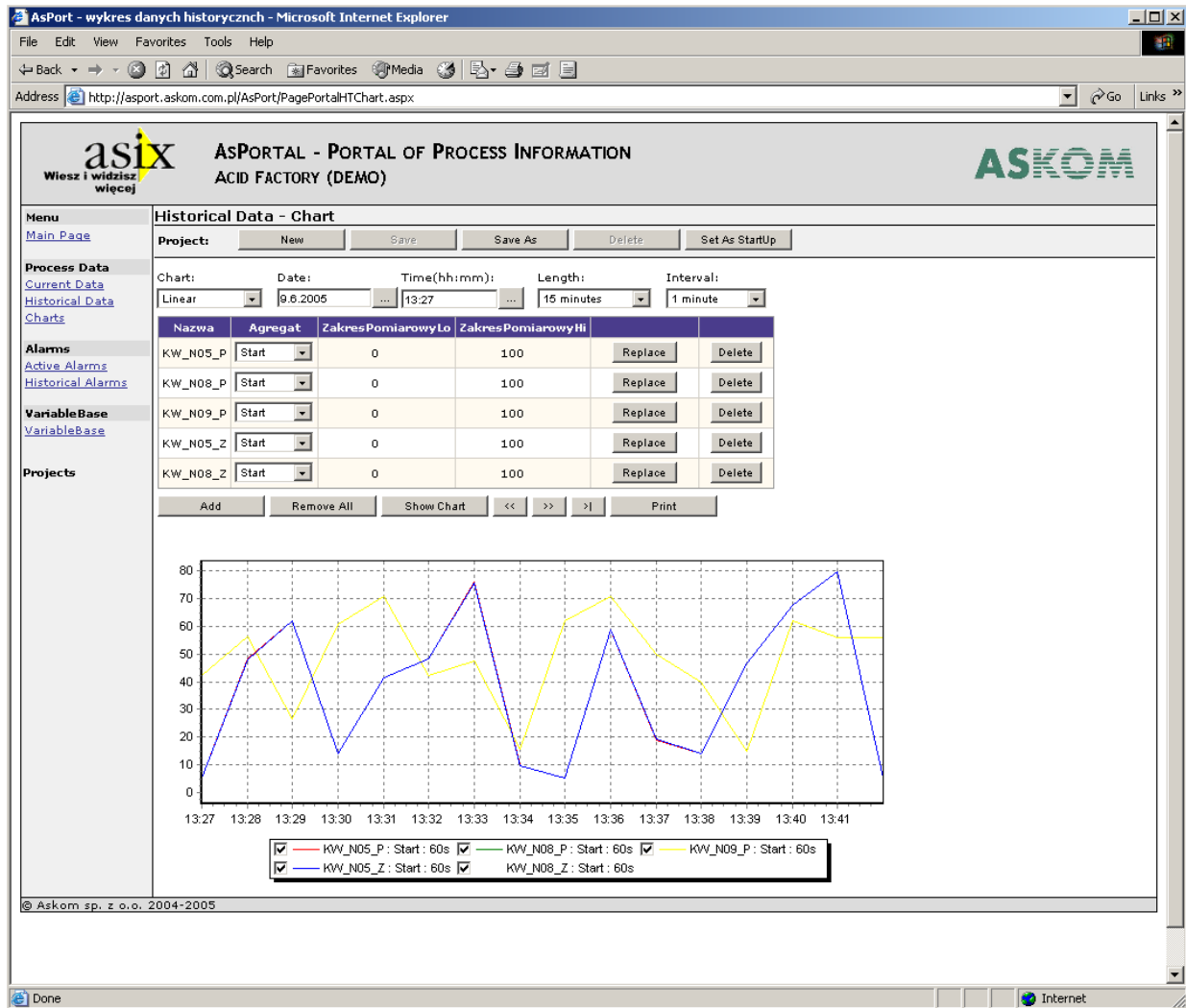
The main data table displays the following data points:

Time	KW_A000 Start °C	KW_A004 Start °C	KW_A008 Start °C	KW_A032 Start °C
9.6.2005 13:34:00	140	190	63,0	118
9.6.2005 13:35:00	180	170	73,0	58
9.6.2005 13:36:00	180	130	83,0	-2
9.6.2005 13:37:00	142	92	92,5	-41
9.6.2005 13:38:00	102	52	97,5	19
9.6.2005 13:39:00	60	10	87,0	82
9.6.2005 13:40:00	20	30	77,0	136
9.6.2005 13:41:00	20	70	67,0	76
9.6.2005 13:42:00	60	110	57,0	16
9.6.2005 13:43:00	98	148	47,5	-41
9.6.2005 13:44:00	138	188	37,5	1
9.6.2005 13:45:00	180	170	27,0	64
9.6.2005 13:46:00	180	130	17,0	124
9.6.2005 13:47:00	140	90	7,0	94
9.6.2005 13:48:00	100	50	3,0	34

At the bottom left, it says "© Askom sp. z o.o. 2004-2005".

Archive data displayed as a chart

Setting the aggregation interval to *auto* will result in automatic setting of its value to 1/360 of the length of display period. If archiving period of a specific variable is longer than the calculated value of interval, the interval value for this variable is increased up to the archiving period.



4.2.3. Alarms

The application makes it possible to present the current alarms appearing in the system and review historical alarms. The alarm contents are displayed in the colour that depends on the alarm type:

- System alarm
- Message
- Warning
- Alarm
- Important alarm

Displayed alarms may be filtered. After the *Show filter* link is pressed, the window with the available filtration options is dropped down. Alarms can be filtered by type, status, group or contents.

Clicking on the *Read* button refreshes the data.

In the window of historical alarms, the data display horizon (beginning time and length of period) should be set.

The following figures present windows of: active alarms with filter option enabled and historical alarms, respectively.

4.2.4. VariableBase

VariableBase enables the review of all variables of **asix** system. The base window stores its last item. Thus, after it is opened again, the last browsed page is loaded automatically.

4.2.5. Variable selection window

In the variable selection window, all the available variables are divided into groups. More than one variable may be selected from different groups.

In the bottom part of the screen, selected variables are displayed in tabulated form. Any variable may be removed from the list of selected variables. By clicking on **OK**, the variables are transferred to the home page.

When the variable is being replaced, selection of new variable makes you immediately return to the home page.

The variable selection window stores its last item separately for the current data, archive data and VariableBase, which makes it easier to add successive variables to designs.

The screenshot shows the AsPortal web application interface. The main content area is titled "Historical Data - Table" and displays a tree view of variables under the "KWAS" group. The selected variables are listed in a table below the tree view.

Nazwa	Opis	Jednostka	
KW_N05_P	Kłapa regulacyjna temperatury w aparacie kontaktowym (po 1 półce)-położenie suwaka		Select
KW_N08_P	Kłapa regulacyjna przepływu powietrza do pieca-położenie suwaka		Select
KW_N09_P	Kłapa regulacyjna przepływu H2S do pieca-położenie suwaka		Select
KW_N05_Z	Kłapa regulacyjna temperatury w aparacie kontaktowym (po 1 półce)-zadawanie		Select
KW_N08_Z	Kłapa regulacyjna przepływu powietrza do pieca-zadawanie		Select

Below the table, there is a "Selected Variables:" section with a list of variables and "Delete" buttons:

- KW_N05_P [Delete]
- KW_N09_P [Delete]
- KW_N05_Z [Delete]
- KW_N08_Z [Delete]

At the bottom of the window, there are "OK" and "Cancel" buttons.

4.2.6. Designs

The AsPortal application enables creation of designs for current data, archive data and alarms. Under a design, information on selected variables (alarms) is stored. Additionally, for archive data the aggregates, aggregation interval, period and type of chart, if any, are remembered.

Designs are handled with use of the menu in the upper part of the screen:



The menu commands have the following operation:

- New* –creating new (empty) design,
- Save* –saving changes in design,
- Save as* –saving design under the name of your choice,
- Remove* –removing design,
- Set as starting design* – setting a starting design to be loaded automatically wherever the page is opened.

5. As2HTML - visualisation in web browser

The As2HTML package enables creation of dynamic HTML pages containing process data from **asix** system application.

For correct operation, the As2HTML package requires the Internet Explorer 6 browser.

The following table contains the list of available modules of the package (*see: next page*).

Name of module	Description
<i>XConnect.htc</i>	<p>This module is operable in web browser on the client's side. It is designed to retrieve from asix system application the information on current data and active alarms from time to time. These data are entered into objects on HTML page then, according to designer's declarations.</p> <p>Data are retrieved from asix system application through the <i>Web Service</i> server.</p> <p>This module also handles NUMBER objects.</p>
<i>XConnectAL.htc</i> <i>XConnectBar.htc</i> <i>XConnectButton.htc</i> <i>XConnectChart.htc</i> <i>XConnectPictures.htc</i> <i>XConnectText.htc</i> <i>XConnectTexts.htc</i> <i>XConnectWatch.htc</i>	<p>These modules handle ALARM, BAR, BUTTON, PICTURES, TEXT, TEXTS and WATCH objects.</p>
<i>XConnectChart.htc</i>	<p>This module also handles CHART object.</p> <p>This module is operable in web browser on the client's side. It is designed to retrieve historical data and enter them into the <i>ActiveX</i> object, which displays a chart on the page in web browser. The price of AsixConnect Server package includes the <i>ActiveX</i> object named <i>TeeChart</i>.</p> <p>Historical data may be retrieved from asix system application and supplied to the <i>XConnectHT.htc</i> module in one of the following three ways:</p> <ul style="list-style-type: none"> • Read once on the server side and included into HTML page as XML Data Island. These data may be read again after the page is refreshed or the <i>postback</i> operation is carried out. • Read once on the server side through the <i>Web Service</i> server and then automatically supplemented with new data in the same way – "live" chart. • Read once on the client side through the <i>Web Service</i> server. These data may be read again after the page is refreshed or relevant functions of the <i>XConnectHT.htc</i> module are called.
<i>XConnect.css</i>	<p>This module contains auxiliary CSS styles of the AsDHTML package.</p>
<i>Number.css, Bar.css,</i> <i>Alarm.css, Button.css,</i> <i>Text.css, Texts.css</i>	<p>These modules contain CSS styles that define default appearance of NUMBER, BAR, ALARM, BUTTON, TEXT and TEXTS objects</p>
<i>XConnect.js</i>	<p>This module contains constants used for defining BAR, CHART and BUTTON objects.</p>
<i>XConnectNetCS.dll</i>	<p>This module makes it easier to prepare data for charts on the web server side.</p>
<i>webservice.htc</i>	<p>Module developed by Microsoft. This module enables access to utilities of <i>WebService</i> servers from the script level on HTML page in web browser.</p>

5.1. Preparation of design directory

In the design directory, you should create bin directory and copy the c:\Asix\XConnectNetCS.dll file into it.

In order to make the directory available within the network, run the *Web information services* program. This program is available in *Start/Control panel/Administrating tools*. In the program window, highlight the *Default website item*. Select *Virtual directory* in the *Action/New menu*. Wizard is started. As *Alias* you should enter the name under which the directory is to be visible in the Internet, and as *Directory* – full access path to the design directory. The other options are to be left unchanged.

By default, on attempt of getting access to the server, Windows authentication is used. In order to enable the anonymous access, highlight the newly created virtual directory, select *Properties* in the *Action* menu, open the *Protections of directories* tab, click on *Edit* in the *Anonymous access* field, and enable the *Anonymous access* option.

5.2. Design preparation

The CHART object may be placed on *aspx* pages only. The other objects may be placed on *aspx* or *html* pages.

Preparation of Visual Studio 2003 application, which is to use one of the objects, is adding the application design and application reference files to modules of the AsixConnect Server package.

Design and source files of application

This item only concerns the applications that use CHART object.

After the design is generated, you need to:

- Highlight the *References* directory in the design tree,
- Select *Add Reference* from the *Design* menu,
- Click on *Browse* and select the *XConnectNetCS.dll* file from the c:\asix\WebService subdirectory,
- Click on *OK* and once again click on *OK* to close the *Add Reference* window.

In every file with C# source code associated to the *aspx* page, the following line should be added in the *using* declaration area:

```
using XConnectNet;
```

HEAD section

Obligatory elements in HEAD section of HTML file:

```
<LINK href=../WebService/Visualization/XConnect.css" type=,text/css" rel=,stylesheet">  
<LINK href=../WebService/Visualization/Number.css" type=,text/css" rel=,stylesheet">  
<LINK href=../WebService/Visualization/Bar.css" type=,text/css" rel=,stylesheet">  
<LINK href=../WebService/Visualization/Button.css" type=,text/css" rel=,stylesheet">  
<script src=../WebService/Visualization/XConnect.js"></script>  
<script>
```

```
function init()
{
  // Region 1 – initialisation of NUMBER, BAR and other objects

  xConnectWS.start (2);

  // Region 2 - initialisation of CHART and object
}
</script>
```

The *init* script will be executed after the page has been loaded. In this script, page initialisation will be carried out.

The *start* function is designed to establish connection with the *Web Service* server and start the process of refreshing data on the page. Declaration of the *start* function is as follows:

```
[JScript]
function start (refreshPeriodS);
```

As *refreshPeriodS* parameter the period of refreshing the current data on page should be given. The unit of refreshing period is second.

BODY element

In the BODY element, you should place the *onload* attribute that refers to the *init* script:

```
<body onload=„init()“>
```

BODY section

Obligatory element in BODY section of HTML file:

```
<DIV id="xConnectWS">AsixConnect</DIV>
```

The *DIV* object named *xConnectWS* is designed to load the *XConnect.htc* module and provide access to its functions. The *XConnect.htc* module is loaded as a result of assigning *xConnectWS* value to the *id* attribute.

Data source

After the design itself and design files are prepared, you need to configure the *Web Service* data source, which is described in sections *Installation* and *Web Config configuration file*

5.3. NUMBER object

The NUMBER object is designed to display the variable of **asix** system application in the numerical form. In order to display the value of variable on *aspx/html* page, you need to:

- create on the page *DIV* element containing defined *id* attribute – for example:

```
<DIV id=„KW_A084“></DIV>
```

- in the *init* initiating script, relate the *DIV* object with variable using the *div2number* function:

```
xConnectWS.div2number (window.document.all.KW_A084, „KW_A084”);
```

```
[JScript]
function div2number (element, itemID, styles, values);
```

The *div2number* function is designed to transform the *DIV* object into *NUMBER* object.

As *element* parameter you should pass the reference of *DIV* object, which is to display the value of variable. For example, the expression *window.document.all.KW_A108* returns references to the object identified as *KW_A108*.

As *itemID* parameter you should pass the name of process variable the value of which is to be displayed by *DIV* element.

The *styles* and *values* are optional parameters, which are used to modify the appearance and operation of *NUMBER* object. Their description is given in section *Modifying the appearance of NUMBER and BAR objects* (See: *Asix4 documentation/OBJECTS*).

5.4. BAR object

The *BAR* object is designed to display the variable of **asix** system application in bar form. In order to display *BAR* object on *aspx/html* page, you need to:

- create on the page *DIV* element containing defined *id* attribute – for example:

```
<DIV id=„KW_A084”></DIV>
```

- in the *init* initiating script, relate the *DIV* object with variable using the *div2bar* function:

```
xConnectWS.div2bar (window.document.all.KW_A084, „KW_A084”, xConnectWS.BarDirN);
```

```
[JScript]
function div2bar (element, itemID, flags, styles, values);
```

The *div2bar* function is designed to transform the *DIV* object into *BAR* object.

As *element* parameter you should pass the reference of *DIV* object, which is to display the value of variable. For example, the expression *window.document.all.KW_A108* returns references to the object identified as *KW_A108*.

As *itemID* parameter you should pass the name of process variable the value of which is to be displayed by *DIV* element.

The *flags* parameter is used to define the appearance of the bar and it should be the logical sum of the following flags:

Name of flag	Description
<i>BarDirE</i>	Direction of bar growth. Respectively: E (to the right), W (to the left), S (downwards) and N (upwards). One of these flags should be used.
<i>BarDirW</i>	
<i>BarDirS</i>	
<i>BarDirN</i>	
<i>BarUseBase</i>	This flag should be used if the bar's base should be different from the minimum value of variable. In the variablebase, the attribute defining the value of bar base should be defined.
<i>BarHiLimit</i>	Use of this flag adds to the bar a crosswise line, which defines the value of upper warning limit.
<i>BarLoLimit</i>	Use of this flag adds to the bar a crosswise line, which defines the value of lower warning limit.
<i>BarHiHiLimit</i>	Use of this flag adds to the bar a crosswise line, which defines the value of upper alarm limit.
<i>BarLoLoLimit</i>	Use of this flag adds to the bar a crosswise line, which defines the value of lower alarm limit.
<i>BarAllLimits</i>	Use of this flag adds to the bar crosswise lines, which define the values of all limits.

The *styles* and *values* are optional parameters, which are used to modify the appearance and operation of BAR object. Their description is given in section *Modifying the appearance of NUMBER and BAR objects* (See: *Asix4 documentation/OBJECTS*).

5.5. BUTTON object

The BUTTON object is designed to display the control element, which enables the masks of web application to be closed and opened. It is mainly used by the AsWWW module when masks of the **asix** system application are converted into web application. However, it may also be used in own web applications.

In order to display BUTTON object on *aspx/html* page, you need to:

- create on the page *DIV* element containing defined *id* attribute – for example:

```
<DIV id=„BUTTON_10“></DIV>
```

- in the *init* initiating script, convert the *DIV* object into button using the *div2button* function:

```
xConnectWS.div2button (
  window.document.all.BUTTON_10,
  'Diagn.',
  ['img/F3C.png'],
  Button3D + ButtonLeftImage,
  function() {xConnectWS.closeAllNotBlockedMask ("");
    xConnectWS.openNewMask ('xkwdiag.aspx', 0, 87, 1024, 681, 0); },
  ['BUTTON_10_Text_Normal', 'BUTTON_10_Text_Inset', 'BUTTON_10_Text_Outset',
  'Button_Frame_Normal', 'Button_Frame_Inset', 'Button_Frame_Outset']);
```

```
[JScript]
function div2button (element, text, images, userFlags, userFunction, userStyles);
```

The *div2button* function is designed to transform the *DIV* object into BUTTON object.

As *element* parameter you should pass the reference of *DIV* object, which is to display the value of variable. For example, the expression `window.document.all.BUTTON_10` returns references to the object identified as *BUTTON_10*

As *text* parameter you should pass the text to be displayed on the button.

As *images* parameter you should pass the table of the names of graphic files to be displayed by the button. The first file is displayed when the button is in *unpressed* position, the second – in the *flat* position, and the third – in *pressed* position. The second and/or third file can be omitted and then the specified files are passed over.

The *flags* parameter is used to define the appearance of the bar and it should be the logical sum of the following flags:

Name of flag	Description
<i>Button3D</i>	This button uses graphic elements that give the impression of third dimension in appearance of the object.
<i>Button3States</i>	Without this flag, the button uses two pictures displayed in <i>unpressed</i> and <i>pressed</i> state. The use of this flag results in using the third picture displayed when the mouse cursor is not over the button.
<i>ButtonLeftImage</i>	Use of this flag results in displaying the picture to the left of the button, and text – to the right.
<i>ButtonTransparent</i>	Use of this flag makes the button be invisible, but react to the mouse click and perform the user function. This flag is usually used in a button applied to the <i>NUMBER</i> object, which aim is to display the station.

The *userFunction* parameter is used to pass the function to be executed in response to clicking on the button. As a parameter, the reference may be passed to the global function and directly to the anonymous function of *JavaScript* language, as in the above-mentioned example.

In the user function, the function to manage the masks provided by the *xConnectWS* object may be used.

Name of function	Description
<i>openNewMask</i>	Function calling syntax: function openNewMask (maskName, x, y, w, h, blockade) Parameters of this function are, respectively, the name of aspx/html page being opened, location and size of the page and logic parameter to define whether the mask has been blocked or not. The opening page overwrites the currently loaded one with use of HTML element named IFRAME.
<i>closeMask</i>	Function calling syntax: function closeMask (maskNameMask) Parameter is the name of aspx/html page to be closed. The name may use * and ? characters to close more than one page at once. The <i>closeMask</i> function also closes the blocked pages.
<i>closeAllNotBlockedMask</i>	Function calling syntax: function closeAllNotBlockedMask (exceptThisMask) This function closes every opened page that is not blocked. As parameter you may give the name of the page to be left opened.

The *userStyles* parameter is used to modify the object appearance. As its value a table consisting of six components, containing the names of styles used to define the appearance of the button and the button frame should be passed.

5.6. TEXT object

The TEXT object is designed to display values of attribute of variable from **asix** system application. In order to display the value of variable attribute on *aspx/html* page you need to:

- create on the page *DIV* element containing defined *id* attribute - for example:

```
<DIV id=„KW_A084_Description"></DIV>
```

- in the *init* initiating script, relate the *DIV* object with variable attribute using the *div2text* function:

```
xConnectWS.div2text (window.document.all.KW_A084_Opis, „KW_A084", „Description");
```

```
[JScript]
function div2text (element, monitoredVariable, attributeName, styles, values)
```

The *div2text* function is designed to transform the *DIV* object into TEXT object.

As *element* parameter you should pass the reference of *DIV* object, which is to display the value of variable. For example, the expression

`window.document.all.KW_A108_Description` returns references to the object identified as `KW_A108_Description`

As `monitoredVariable` parameter you should pass the name of process variable the value of which is to be displayed by `DIV` element.

As `attributeName` parameter the name of the process variable attribute should be passed.

The `styles` and `values` are optional parameters, which are used to modify the appearance and operation of TEXT object.

5.7. PICTURES object

The PICTURES object is designed to display one of many specified bitmaps. The selected picture to be displayed depends on the value of monitored variable. In order to display the value of variable on `aspx/html` page you need to:

- create on the page `DIV` element containing defined `id` attribute - for example:

```
<DIV id=„PICUTRES_104“></DIV>
```

- in the `init` initiating script, relate the `DIV` object with variable using the `div2number` function:

```
xConnectWS.div2pictures (window.document.all.PICTURES_104, "KW_N11", 'BINARY', 8, 4, ['img/KL_WY.png', 'img/KL_H_Y.png', 'img/KL_Z.png', 'img/KL_A.png']);
```

[JScript]

```
function div2pictures(element, itemID, coding, firstBit, statesCount, pictArray, bitMasks)
```

The `div2pictures` function is designed to transform the `DIV` object into PICTURES object.

As `element` parameter you should pass the reference of `DIV` object, which is to display the value of variable. For example, the expression `window.document.all.PICTURES_104` returns references to the object identified as `PICTURES_104`.

As `itemID` parameter you should pass the name of process variable the value of which is to be used to determine the bitmap to be displayed.

As `coding` parameter you should pass the text that defines the coding method. You may use one of the inscriptions: "BINARY", "NATURAL" or "UNDEFINED";

As `firstBit` parameter you should pass the number of the first (youngest) bit of monitored datum.

As `statesCount` parameter the number of object states should be passed.

As `pictArray` parameter you should pass the table of the names of bitmaps to be displayed in the PICTURES object. If the number of bitmaps is equal to the number of object states, then in case of communication errors the red cross-out is drawn against the background of the object. If the number of bitmaps is equal to the number of object states, then in case of communication errors the red cross-out is drawn against the background of the object.

The `bitMasks` parameter is only used for free coding (`coding` parameter equal to "UNDEFINED"). As its value the table of texts containing bitmaps of individual states

should be given. Details are given in **asix** system documentation in chapter on PICTURES object (See: *Asix4 documentation/OBJECTS*).

5.8. TEXTS object

The TEXTS object is designed to display one of many specified texts. The selected text to be displayed depends on the value of monitored variable. In order to display the text on *aspx/html* page, you need to:

- create on the page *DIV* element containing defined *id* attribute - for example:

```
<DIV id=„TEXTS_11“></DIV>
```

- in the *init* initiating script, relate the *DIV* object with variable using the *div2texts* function:

```
xConnectWS.div2texts (window.document.all.TEXTS_11, "KW_B000", 'NATURAL', 3, 2, ['F31', 'F31'], ['TEXTS_11_0', 'TEXTS_11_1']);
```

```
[JScript]
function div2texts (element, itemID, coding, firstBit, statesCount, textsArray,
    userStyles, bitMasks)
```

The *div2texts* function is designed to transform the *DIV* object into TEXTS object.

As *element* parameter you should pass the reference of *DIV* object, which is to display the value of variable. For example, the expression *window.document.all.TEXTS_11* returns references to the object identified as *TEXTS_11*.

As *itemID* parameter you should pass the name of process variable the value of which is to be used to determine the text to be displayed.

As *coding* parameter you should pass the text that defines the coding method. You may use one of the inscriptions: "BINARY", "NATURAL" or "UNDEFINED";

As *firstBit* parameter you should pass the number of the first (youngest) bit of monitored datum.

As *statesCount* parameter the number of object states should be passed.

As *textsArray* parameter you should pass the table of texts to be displayed in the TEXTS object.

As *userStyles* parameter you should pass the table of the names of styles used for formatting the object if value of the variable is available and communication error occurs. Additionally, for communication error the displayed text is always underlined.

The *bitMasks* parameter is only used for free coding (*coding* parameter equal to "UNDEFINED"). As its value the table of texts containing bitmaps of individual states should be given. Details are given in **asix** system documentation in chapter on PICTURES object (See: *Asix4 documentation/OBJECTS*).

5.9. ALARM object

In order to display the value of the last active alarm of **asix** system application on the *html* or *aspx* page, you need to:

- create on the HTML page two *DIV* elements containing the defined *id* attribute:

```
<DIV id=„LastAlarm"></DIV> <DIV id=„LastAlarmDateTime"></DIV>
```

- in the *init* initiating script, relate the *DIV* objects with the alarm system using the *divs2alarms* function:

```
xConnectWS. divs2alarms (window.document.all.LastAlarm,
                        window.document.all.LastAlarmDateTime,
                        true, false, true, true);
```

```
[JScript]
function divs2alarms (alarmTextBoundElement,
                    alarmTimeBoundElement,
                    showTime,
                    showTimeShort,
                    showDate,
                    showDateShort);
```

The *divs2alarms* function enables displaying information on the last, unconfirmed alarm active in two *DIV* objects on HTML page. The first *DIV* element is used for displaying the alarm contents, while the latter one for displaying the alarm time stamp. The other parameters of the *div2alarms* function are used to define how the alarm time stamp is to be displayed.

For formatting the appearance of *DIV* objects the styles defined in the *Alarm.css* file are used. The *Alarm_QualityGood* style is used when communication with alarm server works, while the *Alarm_QualityBad* style is used in case of communication error.

In order to change the appearance of ALARM object you should create two own styles named *Alarm_QualityGood* and *Alarm_QualityBad* in new CSS file and include this file into the *aspx/html* page.

```
<LINK href=„MyStyles.css" type=„text/css" rel=„stylesheet">
```

5.10. WATCH object

The WATCH object is designed to display the date and/or local time on *html/aspx* page. In order to place WATCH object on the page, you need to insert on the page the *DIV* element in which the watch will be placed. The *DIV* element must contain no internal elements. In order to insert *DIV* element, you may use the *FlowLayoutPanel* component available in HTML pallet; this component is rendered on the page as *DIV* element. For this element the following *class* attribute must be defined:

```
class=„ XConnectWatch"
```

EXAMPLE

```
<DIV class=„XConnectWatch"></DIV>
```

5.11. Obiekt WYKRES

5.11.1. Chart declaration

The CHART object is designed to display values of archive process variables of **asix** system application in the form of a chart. Only *aspx* pages are handled because the part of the chart development process, which is connected with including the chart design into web site, can take place on server only.

The CHART object may be handled in three variants:

1. Static chart generated on the server side.
2. Chart generated on the client side.

Static chart generated on the server side is used to display a chart of any period. Historical data are read from selected period and inserted into web site as XML Data Island. These data are then used by the script on the client side to create a chart. After the chart is displayed, it is not modified on an automatic basis any more. In majority of cases generation of this type of chart is fastest. In order to move the chart period, you can create buttons on the site, relate them to handling procedures on the server side and, using these procedures, provide software for retrieving new historical data.

Chart generated on the client side is used to display a chart of historical data from period that covers the current moment and some time ago. New data, which appear in the archive of **asix** system application, are added to the end of the chart on an automatic basis. The replaced data from the beginning of the chart are removed. Page is retrieved from server only once and data are refreshed by the script operating on the client side.

Chart generated on the client side is also used to display a chart of any period too. Historical data of specified variables are read on the client side when the page has already been loaded. After the chart is displayed, it is not modified on an automatic basis any more. In order to move the chart period, you can create buttons on the site, relate them to handling procedures on the client side and, using these procedures, provide software for moving the chart period.

Regardless of the type of your chart, you need to carry out the following steps:

1. Prepare design of the chart and save it into file; the file should be placed in the application's directory.
2. Insert on *aspx* page the *DIV* element in which the chart will be placed.

The *DIV* element must contain no internal elements. In order to insert *DIV* element, you may use the *FlowLayoutPanel* component available in HTML pallet; this component is rendered on the page as *DIV* element. For this element the following attributes must be defined:

```
- runat="server"  
- class="XConnectChart"  
- id=<identyfikator>
```

EXAMPLE

```
<DIV runat="server" id="Chart1" class="XConnectChart" style="Z-INDEX: 101; LEFT: 16px; WIDTH: 768px;
```

```
POSITION: absolute; TOP: 16px; HEIGHT:192px" ms_positioning="FlowLayout"></DIV>
```

3. In the code file, you should add declaration of the class field:

```
protected System.Web.UI.HtmlControls.HtmlGenericControl Chart1;
```

4. In the code file, in *Page_Load* function, the following line should be inserted

```
ChartXMLData.InsertChartHTML ("chart.tee", Chart1);
```

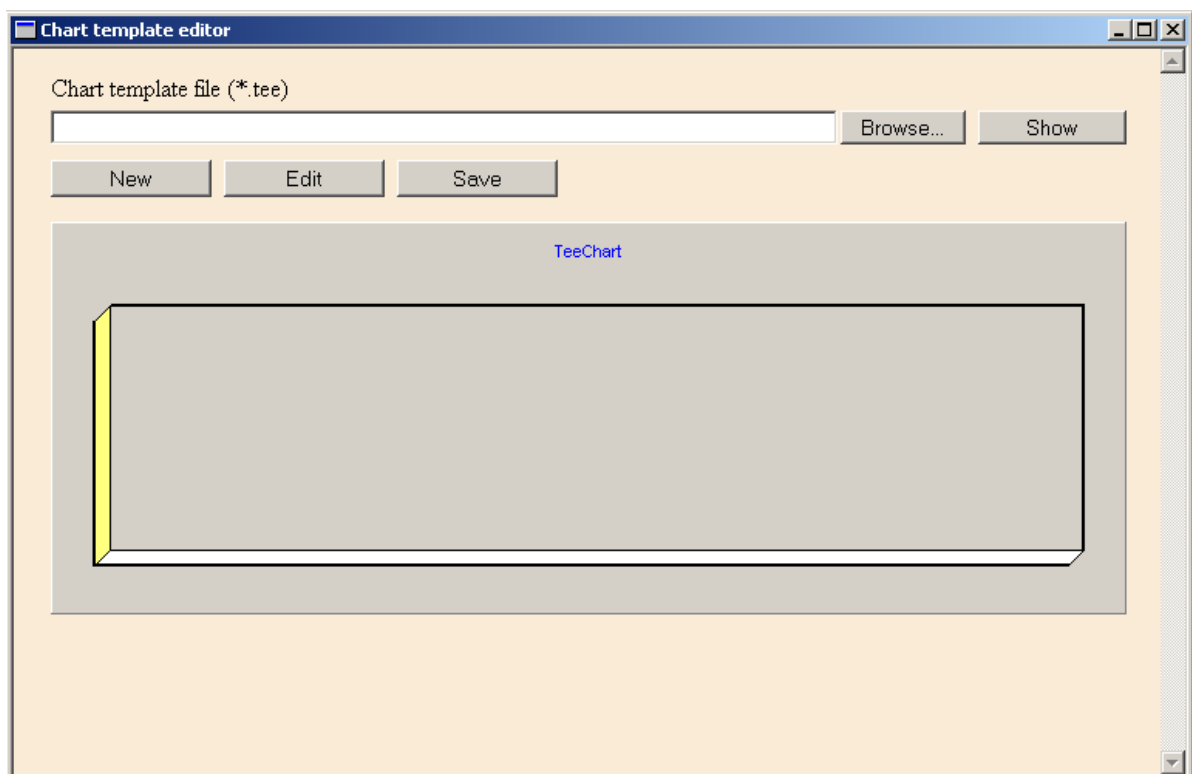
Rather than the name of *chart.tee* chart design file, the name of the file created in item 1 should be used.

Further steps are described in sections on individual types of charts.

5.11.2. Chart design

Charts are designed with use *ChartDesigner.hta* application. The basic element of design is series. In design, you should add one series for every variable of **asix** system application the chart of which is to be displayed. As type of series you should use *Line*, *Bar* or *Surface*. The other XY charts may also work, however they have not been tested. XYZ charts cannot be used. Apart from using series, it is recommended that chart axes and legend should be parameterised.

After *ChartDesigner.hta* application program is started, the main window appears:



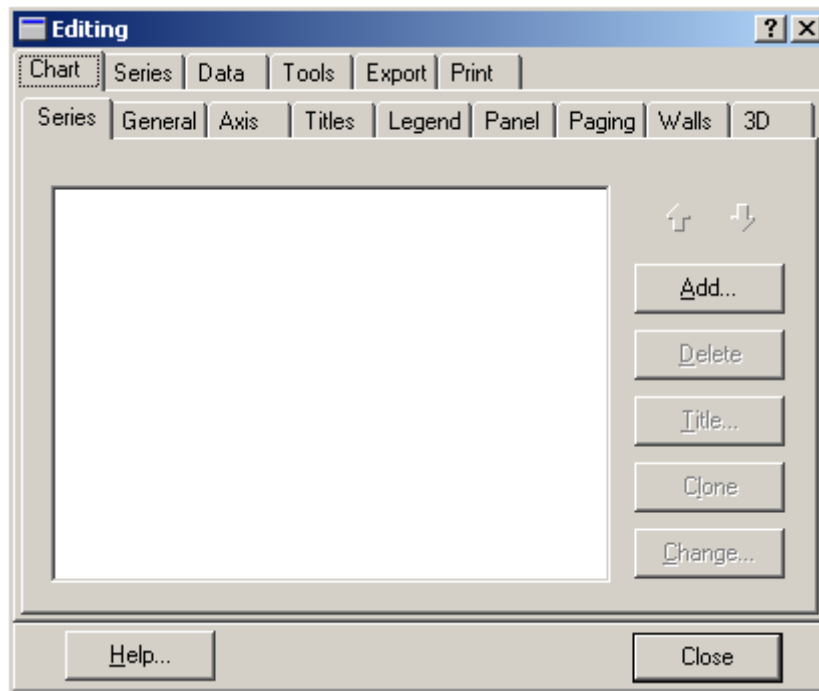
Description of elements displayed in the window:

Chart template file - in this field the name of the chart design file should be provided. The name can be entered or selected by clicking on *Review* button.

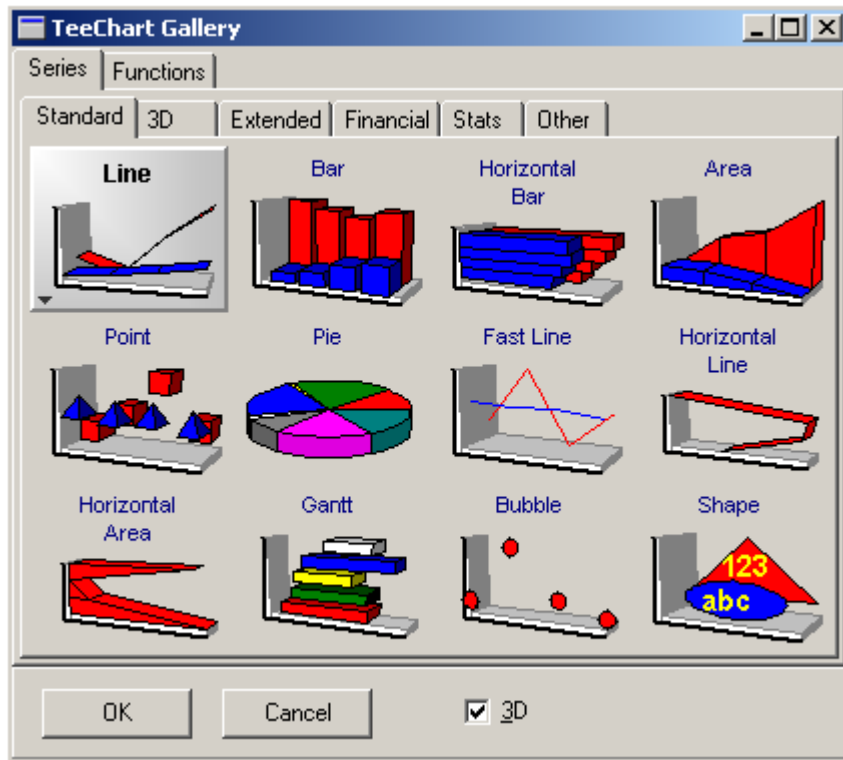
Review - opens the window, which allows the name of the chart design file to be selected.

- Show* - loads the chart design file with name as provided in the *Chart design file* field.
- New* - creates the new chart design file.
- Edit* - edits the current chart design file – the design editor is activated. After edition the chart should be saved into file – this file will be used together with *aspx* pages to display the chart.
- Save* - saves design into file.

Clicking *Edit* button activates the design editor.



In order to add a series to the chart, click on *Add* button and select *Line*, *Bar* or *Surface* chart.



After every series is created, you should open the *Series* tab, select the series created, open the *General* tab and in *Horizontal axis* group check the *Date/Time* field.



The other series options enable the chart appearance to be adjusted to designer needs.

5.11.3. Static chart generated on the server side

Static chart generated on the server side

Procedure for placing the chart on *aspx* page:

1. Prepare the page – see: 5.11.1 *Chart declaration*.
2. Add *WebControl* component named *Literal* to the *aspx* page and assign its identifier. In the example below, the *XMLIsland1* identifier will be used.
3. In the code file, insert the following lines in the *Page_Load* function:

```
ChartXMLData chartXMLData = new ChartXMLData();
chartXMLData.Add ("K11_Para", Aggregate.Start, "K11_Para",
    HTReadFlags.Default, "1M");
chartXMLData.SetChartPeriod "DAY-24H", "24H");
chartXMLData.ReadData (XMLIsland1);
Session.Add ("chartXMLData", chartXMLData);
```

4. On the *aspx* page, in the *init* function, in the place denoted as *Region2*, insert the following line:

```
window.document.all.Chart1.div2chartXML (null, "XMLIsland1");
```

The fragment of C# language program provided in item 5 executes the following operations:

- Creates the *ChartXMLData* class object designed to retrieve archive data and display them on the page.
- Using the *Add* function, adds one *K11_Para* variable to the object. The other parameters determine the aggregate parameter, name of variable in the legend, readout flags and sampling period.
- Using the *SetChartPeriod* function, sets the beginning and length of period from which data are to be retrieved.
- Using the *ReadData* function, reads archive data and places them in the *XMLIsland1* object.
- Using the *InsertChartHTML* function, inserts the chart definition into *Chart1* object. Rather than the names of *chart.tee* chart design file, the name of the file created in item 1 should be used.

Inserts the *chartXMLData* object into *Cache* memory of ASP.NET application so that this object can be used for other calls carried out from the user page when *postback* calls are handled. For example, the button to move the chart period forwards by one hour can be placed on the page. The code to carry out this operation is as follows:

```
ChartXMLData chartXMLData = (ChartXMLData)Session["chartXMLData"];
chartXMLData.MoveChartPeriod ("1H");
chartXMLData.ReadData (XMLIsland1);
```

div2chartXML function

```
[JScript]
function div2chartXML (teeFile, xmlDataIslandName)
```

The *div2chartXML* function is designed to create in *DIV* element the CHART object for which data have been prepared on the server side and passed as a part of *aspx* page. As *teeFile* parameter you should pass *undefined* value and as *xmlDataIslandName* parameter – the identifier of *Literal* type object into which data have been inserted on the server side.

5.11.4. Chart generated on the client side

Chart generated on the client side

Procedure for placing the chart on *aspx* page:

1. Prepare the page – see: *Chart declaration*.
2. On the *aspx* page, in the *init* function, below the place denoted as *Region2*, insert the following line:

```
window.document.all.Chart1.div2chart(
    undefined, ["KW_A000"], ["Start"], ["KW_A000"], 3600, 60);
```

Instead of *KW_A000* variable name, the appropriate name of variable should be passed. Instead of *Start* attribute name, the name of another attribute should be passed

3. If a live chart is expected to be displayed on the side, the following line including *goLive* function call should be inserted:

```
window.document.all.Chart1.goLive (5);
```

Otherwise, if the chart with archived data is expected to be displayed on the side, the following line including *moveChartPeriodTo* function call should be inserted:

```
window.document.all.Chart1.moveChartPeriodTo("DAY");
```

div2chart function

```
[JScript]
function div2chart (
    teeFile,
    itemNames,
    aggregateNames,
    itemLegends,
    periodLen,
    resampleInterval,
    readFlags)
```

The *div2Chart* function is designed to create CHART object in DIV element.

The *teeFile* parameter in the final version of application must be assigned the *undefined* value, and the chart must be loaded, according to description, in the *Page_Load* function. When testing the application, as value of the *teeFile* parameter you may pass the name of the chart design file and put off modification of the *Page_Load* function.

As *itemNames* parameter the table of the texts of variable names should be passed. You may use shortened notation to create tables of *Jscript* language:

```
["Nazwa1", "Nazwa2", "Nazwa2"]
```

As *aggregateNames* parameter the table of the names of variable aggregates should be passed.

As *itemLegends* parameter you should pass the table of texts to be displayed in the chart legend and represent the individual variables.

As *periodLen* parameter you should pass the length of chart period in seconds or using relative time.

As *resampleInterval* parameter you should pass the length of aggregate interval in seconds or using relative time.

The *readFlags* parameter is optional and more information on its using can be found in *AsixConnect4 manual*, chapter *Flagi ReadFlags*. There are two lines in init script, which may be used as value of this parameter:

```
ReadFlags_AddPointAtStartOfBadQualities
ReadFlags_AddPointAtEndOfBadQualities
```

The first constant variable should be used in systems operating on momentary data. The second one may be used in systems operating on averages (e.g. in gas emission control systems).

addSeries function

```
[JScript]
function addLiveSeries(
  seriesSerialNumbers,
  itemNames,
  aggregateNames,
  itemLegends,
  periodLen,
  resampleInterval,
  readFlags)
```

The *addSeries* function enables the series that differ in display period, length of aggregate interval or refreshing period to be placed on the same chart. If you want to use this option, first of all you should call the *div2chart* function with no parameters or passing in the *itemNames* parameter the names of lower number of variables than number of series defined in the chart design file. Then, by calling the *addSeries* function you should add the other variables to the chart.

As *seriesSerialNumber* parameter you should pass the table of numbers corresponding to the numbers of series in the design file of series to which variables from the second parameter of the *itemNames* function are to be assigned.

The remaining parameters have the same meaning as parameters of the *div2chart* function.

goLive function

```
[JScript]
function goLive (
  refreshIntervalS)
```

The *goLive* function enables switching the chart to the live mode – that displays recently registered data and refreshes these data in periodic mode.

As *refreshIntervalS* you should pass the length of chart period in seconds.

moveChartPeriodTo function

```
function moveChartPeriodTo (
  periodStart)
```

The *moveChartPeriodTo* sets new data origin on the chart. If the chart is in “live” mode, this mode is being switched off and historical data are displayed.

As *periodStart* parameter you should pass the beginning of the time period, from which the data will be received. The date may be passed as an absolute date or in text form in OPC format, as a relative date in *Date* object, or an absolute date in text form.

moveChartPeriodBy function

```
function moveChartPeriodBy (
    seconds)
```

The *moveChartPeriodBy* function allows the beginning of the chart to be moved forwards (the *second* parameter larger than zero) or backwards (the *second* parameter smaller than zero).

5.11.5. ChartXMLData class

The *ChartXMLData* class facilitates creation of web pages containing charts.

InsertChartHTML function

```
[C#]
static public void InsertChartHTML(
    String fileName,
    HtmlGenericControl div);
```

Static function used for all types of charts. It is designed to insert into *DIV* element the HTML code containing:

- Definition of HTML object displaying the chart.
- Description of chart read from the chart design file.

As *fileName* parameter you should pass the name of the file containing the chart definition; in the file name, you may use the relative path. As *div* parameter the object representing the *DIV* object appearing on the page should be given.

Add function

```
[C#]
public void Add(
    String itemID, Aggregate aggregate, String itemLegend, ReadFlags readFlags)
public void Add(
    String itemID, Aggregate aggregate, String itemLegend, ReadFlags readFlags,
    TimeSpan resampleInterval)
public void Add(
    String itemID, Aggregate aggregate, String itemLegend, ReadFlags readFlags,
    String resampleInterval)
```

The *Add* function is designed to add other variables to the chart being created. The *Add* function should be called as many times as many series have been created in the chart design file.

The *itemID* parameter defines the name of variable the samples of which are to be placed on the chart.

The *aggregate* parameter defines the aggregate to be calculated from archive data. Most frequently, the *Aggregate.Start*, which enables even distribution of points in the chart, is used.

As *itemLegend* parameter you should pass the text to define the variable in the chart legend. This parameter is often the same as *itemID*.

The *readFlags* parameter was described in *ReadFlags* flags.

For calculation of aggregate, the period is divided into intervals which length is defined in the *resampleInterval* parameter. The aggregate with identifier given in the *aggregate* parameter is calculated for every interval using the data archived during this interval. Every interval is assumed to be closed on the left and opened on the right. Value of the *resampleInterval* parameter may be given as *TimeSpan* type object or as a text containing the interval length in OPC format. If the *resampleInterval* parameter is omitted, then it is assumed to be equal to the period of archiving the variable.

SetChartPeriod function

```
[C#]
public void SetChartPeriod(
    DateTime periodStart, TimeSpan periodLen, TimeSpan resampleInterval)
public void SetChartPeriod(
    DateTime periodStart, TimeSpan periodLen)
public void SetChartPeriod(
    DateTime periodStart)

public void SetChartPeriod(
    String periodStart, String periodLen, String resampleInterval)
public void SetChartPeriod(
    String periodStart, String periodLen)
public void SetChartPeriod(
    String periodStart)
```

The *SetChartPeriod* function defines the period from which data are to be retrieved.

The *periodStart* parameter is obligatory and defines the beginning of the period. It may be given as *DateTime* class object (absolute value of time) or as a text containing the relative date in OPC format.

The *periodLen* parameter defines the period length. It may also be given as an absolute or relative value.

The *resampleInterval* parameter defines the length of aggregation interval. If used, the interval of all the variables displayed on the chart is modified.

When the *SetChartPeriod* function is called for the first time, the *dateTimeFrom* and *periodLen* parameters are obligatory. For successive calls, only the *dateTimeFrom* parameter may be given. Thus, we obtain the effect of changing the beginning of the chart without modifying the other parameters

MoveChartPeriod function

```
[C#]
public void MoveChartPeriod (int period)
public void MoveChartPeriod (TimeSpan period)
public void MoveChartPeriod (String period)
```

The *MoveChartPeriod* function allows the beginning of the chart to be moved forwards (the *period* parameter larger than zero) or backwards (the *period* parameter smaller than

zero). As value of the *period* parameter you may give the number of seconds, *TimeSpan* type object or a text containing the period length in OPC format.

ReadData function

```
[C#]
public void ReadData (System.Web.UI.WebControls.Literal XMLIsland)
```

The *ReadData* function reads historical data of specified variables from selected period and inserts the data into HTML page as XML Data Island. These data are then used to create a chart by the script on the client side.

5.11.6. ReadFlags

In addition, the read samples may be processed in order to make it easier to create charts based on them. The processing method is defined with use of the *ReadFlags* parameter. This parameter should be the logical sum of constants described below. If no special data processing is required, you should use the *ReadFlags.Default* flag.

Approximation of values of bad quality data

With lack of approximation, the value field of the point of bad quality is assigned zero value. If the *ReadFlags.LinearAproximationOfValueOfBadQuality* flag is used, this point is assigned the value of the nearest previous point of good quality (quality is still bad!). If there is no previous point of good quality, the value of any point of good quality from the current period or encountered during the previous readouts of the same variable is used.

Adding a point on boundary of the area of bad quality samples

Sample may be added in the beginning or in the end of the area of bad quality samples. You may use one of the following flags: *ReadFlags.AddPointAtStartOfBadQualities* or *ReadFlags.AddPointAtEndOfBadQualities*, or none of them.

Adding one sample on the right edge of area

Adding one sample on the right edge of the archive data area being read allows the whole specified period to be filled with a chart. To do so, use the *ReadFlags.AddPointAtRightBound* flag.

For example, if we are reading hour data for one day, we will receive 25 samples beginning from midnight of the current day until midnight of the next day.

Use of flags in Jscript scripts

In order to make it possible to use *ReadFlags* constants in JScript language, constants corresponding to *ReadFlags* constants of C# language have been defined in the *XConnect.js* file.

Name of flag in C# language	Name of flag in JScript language
<i>ReadFlags.Default</i>	<i>ReadFlags_Default</i>
<i>ReadFlags.AddPointAtStartOfBadQualities</i>	<i>ReadFlags_AddPointAtStartOfBadQualities</i>
<i>ReadFlags.AddPointAtEndOfBadQualities</i>	<i>ReadFlags_AddPointAtEndOfBadQualities</i>
<i>ReadFlags.AddPointAtRightBound</i>	<i>ReadFlags_AddPointAtRightBound</i>

5.12. Modification in appearance of NUMBER and BAR object

5.12.1. Variable states

For object modification purposes nine states of objects are distinguished. The states are given in the table below. The object state is defined on the basis of quality of variable related to it.

Name of object state	Description of object state
<i>QualityGood</i>	Quality of variable is good.
<i>QualityGood_HiHi</i>	Quality of variable is good, the upper alarm limit exceeded.
<i>QualityGood_Hi</i>	Quality of variable is good, the upper warning limit exceeded.
<i>QualityGood_Lo</i>	Quality of variable is good, the lower warning limit exceeded.
<i>QualityGood_LoLo</i>	Quality of variable is good, the lower alarm limit exceeded.
<i>QualityUncertain</i>	Quality of variable is uncertain.
<i>QualityBad</i>	Quality of variable is bad. This state includes all cases of bad quality, except for <i>Last Known</i> and <i>Sensor Failure</i> .
<i>QualityBad_LastKnown</i>	Quality of variable is bad, communication error occurred, the last known value of variable is available,
<i>QualityBad_SensorFailure</i>	Quality of variable is bad due to sensor failure. This state is signalled by control variable or device driver.

During object modification the appearance or behaviour of object is the same for different states of objects. In this case you may pass over the definitions of certain states and then, instead of the state that has been passed over, another state will be used according to the table below. Definitions of *QualityGood* and *QualityBad* are always required.

Name of omitted state	Name of replacing state
<i>QualityGood_HiHi</i>	<i>QualityGood_Hi</i>
<i>QualityGood_Hi</i>	<i>QualityGood</i>
<i>QualityGood</i>	-
<i>QualityGood_Lo</i>	<i>QualityGood</i>
<i>QualityGood_LoLo</i>	<i>QualityGood_Lo</i>
<i>QualityUncertain</i>	<i>QualityBad_LastKnown</i>
<i>QualityBad</i>	-
<i>QualityBad_LastKnown</i>	<i>QualityBad</i>
<i>QualityBad_SensorFailure</i>	<i>QualityBad</i>

5.12.2. Modification in appearance of NUMBER and BAR objects

For every object state the CSS style, which determines the appearance of *DIV* object, is defined. These styles are defined in *Number.css* and *Bar.css* files.

Name of object state	Name of default style of NUMBER object	Name of default style of BAR object
<i>QualityGood</i>	<i>Number_QualityGood</i>	<i>Bar_QualityGood</i>
<i>QualityGood_HiHi</i>	<i>Number_QualityGood_HiHi</i>	<i>Bar_QualityGood_HiHi</i>
<i>QualityGood_Hi</i>	<i>Number_QualityGood_Hi</i>	<i>Bar_QualityGood_Hi</i>
<i>QualityGood_Lo</i>	<i>Number_QualityGood_Lo</i>	<i>Bar_QualityGood_Lo</i>
<i>QualityGood_LoLo</i>	<i>Number_QualityGood_LoLo</i>	<i>Bar_QualityGood_LoLo</i>
<i>QualityUncertain</i>	<i>Number_QualityUncertain</i>	<i>Bar_QualityUncertain</i>
<i>QualityBad</i>	<i>Number_QualityBad</i>	<i>Bar_QualityBad</i>
<i>QualityBad_LastKnown</i>	<i>Number_QualityBad_LastKnown</i>	<i>Bar_QualityBad_LastKnown</i>
<i>QualityBad_SensorFailure</i>	<i>Number_QualityBad_SensorFailure</i>	<i>Bar_QualityBad_SensorFailure</i>

You can modify the object appearance by creating your own collection of styles in new CSS file and including it into *aspx/html* page following *css* files of the AsixConnect package or instead of them.

```
<LINK href=„/WebService/Visualization/XConnect.css“ type=„text/css“ rel=„stylesheet“>
<LINK href=„/WebService/Visualization/Number.css“ type=„text/css“ rel=„stylesheet“>
<LINK href=„/WebService/Visualization/Bar.css“ type=„text/css“ rel=„stylesheet“>
<LINK href=„/WebService/Visualization/Alarm.css“ type=„text/css“ rel=„stylesheet“>
<LINK href=„MyStyles.css“ type=„text/css“ rel=„stylesheet“>
```

The names of styles in new CSS file may be:

1. Unchanged – in this case you should replace declarations of using *css* files of AsixConnect package with declarations of using own *css* files; changes in appearance will concern all the objects appearing on the page,
2. New – in this case you should add declarations of using own *css* files of AsixConnect package. Changes in appearance will concern only the objects to which the names of new styles were passed on the initiation stage. New styles may be based on the styles defined in *css* files of AsixConnect package and modify only certain attributes and/or add new ones.

In order to pass the names of new styles to NUMBER or BAR object, in the *init* script you should create an object containing the names of new styles included in relevant properties and pass the object as *styles* parameter of the *div2number* or *div2bar* functions. The names of properties corresponding to individual states are identical with the names of states.

This is the example of passing a new collection of styles to NUMBER object:

```
var engineNumberStyle = new Object;
engineNumberStyle.QualityGood = "Engine_QualityGood";
engineNumberStyle.QualityGood_HiHi = "Engine_QualityGood_HiHi";
engineNumberStyle.QualityGood_Hi = "Engine_QualityGood_Hi";
engineNumberStyle.QualityGood_Lo = "Engine_QualityGood_Lo";
engineNumberStyle.QualityGood_LoLo = "Engine_QualityGood_LoLo";
engineNumberStyle.QualityUncertain = "Engine_QualityUncertain";
engineNumberStyle.QualityBad = "Engine_QualityBad";
engineNumberStyle.QualityBad_LastKnown = "Engine_QualityBad_LastKnown";
engineNumberStyle.QualityBad_SensorFailure = "Engine_QualityBad_SensorFailure";
```

xConnectWS.div2number (window.document.all.KW_A084, "KW_A084", engineNumberStyle);

The appearance of *BAR OBJECTS IS* formatted with use of two additional styles defined in the *Bar.css* file.

Name of style	Description of style	Name of object property when passing the styles
<i>Bar_Background</i>	This style is used for formatting the object background.	<i>Background</i>
<i>Bar_Marker</i>	This style is used for formatting the crosswise lines showing the values of warning and alarm limits.	<i>Marker</i>

5.12.3. Modification in operation of NUMBER object

For every state of the NUMBER object a function is defined, which, on the basis of information concerning the variable state, returns a value to be displayed in *DIV* object. If for one of the states the function always returned the same value, then this value could be directly used instead of the function.

After new value of variable is retrieved from **asix** system application, the *XConnect.htc* module defines the state of the variable. If constant value is defined for this state, then it is entered into *DIV* object. If a function is defined for this state, then this function is called and its result is entered into *DIV* object. As the function parameter the object containing the variable state is passed.

The object has the following fields:

Name of property	Description
<i>succeeded</i>	This field is assigned the value <i>true</i> if state of variable has been read out and the <i>timeStamp</i> , <i>quality</i> and <i>dataValues</i> fields are set. Otherwise, the field is assigned the value <i>false</i> .
<i>errorString</i>	If the <i>succeeded</i> field is assigned the value <i>false</i> , then the <i>errorString</i> field contains a text description of error.
<i>timeStamp</i>	Time stamp of variable value.
<i>quality</i>	Quality of variable.
<i>dataValues</i>	Value of variable in tabulated form. When variable is of scalar type, the table consists of one element.

Default values of functions or constants for individual states of variable for NUMBER object are as follows:

Name of state	Code of default function or constant
<i>QualityGood</i>	function (itemState) { return itemState.dataValues[0]; }
<i>QualityBad</i>	""
<i>QualityBad_LastKnown</i>	function (itemState) { return itemState.dataValues[0] + "??"; }
<i>QualityBad_SensorFailure</i>	""??

For other states the values are assigned in accordance with the algorithm described in 5.12.1 *Variable states*.

In order to modify the operation of NUMBER object, in the *init* script you should create an object containing constants or functions of variable and pass the object as the fourth parameter of the *div2number* function. As the third parameter of the *div2number* function you should pass the object with the names of styles or the *undefined* constant. The names of properties corresponding to individual states are identical with the names of states.

EXAMPLE

Example of modification in operation of NUMBER object:

```
var engineNumberText = new Object;

engineNumberText.QualityBad_LastKnown =
    function (itemState) { return "??" + itemState.dataValues[0] + "??"; };

engineNumberText.QualityGood = function (itemState) { return itemState.dataValues[0]; };
engineNumberText.QualityBad = "??";
engineNumberText.QualityBad_SensorFailure = "???"

xConnectWS.div2number (
    window.document.all.KW_A084, "KW_A084", undefined, engineNumberText);
```

5.13. Binding DHTML object with current value of variable

Processing of process variable value before assigning it to DHTML object property

In order to bind the property of DHTML object with current value of variable, you need to use the *bindProperty* function:

```
[JScript]
function bindProperty (
    element,
    propertyName,
    itemName,
    values)
```

As *element* parameter you should pass the reference to DHTML object the property of which is to be modified. For example, the expression *window.document.all.KW_A108* returns reference to object identified as *KW_A108*, and expression *window.document.all.KW_A108.style* returns reference to object containing the object styles identified as *KW_A108*.

As *propertyName* parameter you should pass the name of DHTML object property to be modified. This property will be referred to as the active property.

As *itemName* parameter you should pass the name of process variable the value of which is to affect the active value.

If value of the process variable is to be assigned directly to the active value all the time, then as *values* parameter the *null* value should be passed.

Processing of process variable value before assigning it to DHTML object property

In the operation of binding any value of DHTML object with value of process variable, nine states are distinguished. For their description, see: *5.12.1 Variable states*. For every state of the object a function should be defined, which, on the basis of information concerning the variable state, returns a value to be entered into the properties of DHTML object. If for one of the states the function always returned the same value, then this value could be directly used instead of the function.

After new value of variable is retrieved from **asix** system application, the *XConnect.htc* module defines the object state. If constant value is defined for this state, then this value is entered into the properties of DHTML object. If a function is defined for this state, then this function is called and its result is entered into the properties of DHTML object. As the function parameter the object containing the variable state is passed.

The object has the following fields:

Name of property	Description
<i>succeeded</i>	This field is assigned the value <i>true</i> if state of variable has been read out and the <i>timeStamp</i> , <i>quality</i> and <i>dataValues</i> fields are set. Otherwise, the field is assigned the value <i>false</i> .
<i>errorString</i>	If the <i>succeeded</i> field is assigned the value <i>false</i> then the <i>errorString</i> field contains a text description of error.
<i>timeStamp</i>	Time stamp of variable value.
<i>quality</i>	Quality of variable.
<i>dataValues</i>	Value of variable in tabulated form. When variable is of scalar type, the table consists of one element.

In the *init* script, you should create an object containing constants or functions of individual states and pass the object as *values* parameter of the *bindProperty* function. The names of properties corresponding to individual states are identical with the names of states.

EXAMPLE

Example of binding the *left* property of DHTML object to the value of process variable as a result of which the object moves upon changes in value of this variable:

```
var movingNumberPos = new Object;
movingNumberPos.QualityBad = 10; // default location
movingNumberPos.QualityGood =
  function (itemState) { itemState.dataValues[0]; };
xConnectWS.bindProperty (
  window.document.all.KW_A084.style, „left“, „KW_A084“, movingNumberPos);
```

6. As2WWW - converter of applications

The *As2WWW* module is designed for conversion of **asix** system application into web application. The application obtained as a result of this conversion is run on the web server and can be viewed by the user through the Internet Explorer 6 browser.

6.1. Conversion stages

Conversion of **asix** system application into web application consists of a few stages. These stages are:

1. Conversion of MSK mask files into XML format files,
2. Generating bitmaps containing all the static objects of masks (bitmaps will be used as backgrounds for web pages)
3. Separating bitmaps from DAT file of application and writing them in PNG format,
4. Generating the starting page,
5. Conversion of mask files in XML format into ASPX web server format,
6. Conversion of PUM menu definition files into ASPX web server format.

Every stage is carried out by separate program. Parameters for the program calls are specified below. Each of the programs handles the */o* parameter by means of which the program's output directory is determined.

Conversion of MSK mask files into XML format files is carried out with use of *msk2xml* program. Parameters for calling the program are as follows:

```
msk2xml <pattern_of_mask_file_names> [/s] [/o<output_directory>]
```

As *<pattern_of_mask_file_names>* you may pass full name of the mask, e.g. *KW.MSK_MAP* or a pattern including all the names appearing within directory, e.g. **.MSK*.

If the */s* parameter is used, statistics of the objects used in the masks will be displayed only.

Bitmaps containing the pictures of static objects of masks are generated with use of *msk2png* program. Parameters for calling the program are as follows:

```
msk2png <pattern_of_mask_file_names> [<ini_file>] {/d} [/i] [/o<output_directory>]
```

As *<pattern_of_mask_file_names>* you may pass full name of the mask, e.g. *KW.MSK_MAP* or a pattern including all the names appearing within directory, e.g. **.MSK*.

If the */d* parameter is used, the pictures of dynamic objects will be added as well.

If the */i* parameter is used, the pictures of interactive objects will be added as well.

Conversion of bitmap file of DAT application into PNG format files is carried out with use of *DatToBmp* program. Parameters for calling the program are as follows:

```
DatToBmp <dat_file> <name_of_bitmap> [<ini_file>] [/o<output_directory>]
```

As *<dat_file>* parameter you should pass the name of file containing all bitmaps of application.

As *<name_of_bitmap>* parameter you should pass the name of the bitmap to be marked out or the pattern of the names. If the * name is used, all bitmaps are marked out.

The starting page of application is generated with use of the *ini2frameset* program. Parameters for calling the program are as follows:

```
ini2frameset <ini file> [/o<output_directory>]
```

As *<ini file>* parameter you should pass the name of the *ini* file in application. As a result of the program operation, the *default.htm* file is created and it is the starting page of application.

Conversion of XML mask files into ASPX pages is carried out with use of *xml2aspx* program. Parameters for calling the program are as follows:

```
xml2aspx <pattern_of_xml_file_names> <ini file> [/o<output_directory>]
```

As *<pattern_of_xml_file_names>* usually the *.XML is given, which includes all the file names within directory.

Conversion of PUM menu definition files into ASPX web server format is carried out with use of *pum2aspx* program. Parameters for calling the program are as follows:

```
pum2aspx <pattern_of_pum_file_names> [/o<output_directory>]
```

As *<pattern_of_pum_file_names>* parameter usually the *.PUM is given, which includes all the file names within directory.

6.2. Structure of directories in web application

The first step of conversion is to create a directory in which the web application is to be placed. This may be a directory on the same file level as the **asix** system application directory and as its name may be used the name of **asix** system application directory with *_www* suffix. Then, the *bin*, *img*, *tee* and *xml* subdirectories should be created in the web application directory.

You should copy the *c:\asix\XConnectNetCS.dll* file into *bin* directory. In the *img* directory, bitmap files are placed, in the *tee* directory – chart templates, and in the *xml* directory – intermediate mask format, ie *xml* format.

6.3. Conversion automation

In order to automate the conversion process, you should create the *AsWWW.bat* batch file, which contains calling commands for programs carrying out the individual stages of

conversion. For example of the file to execute conversion of the Acid Manufacturing Plant demo application, see below.

The file is located in the main directory of application, i.e.

c:\Asix\Applications\Acid_Manufacturing_Plant and run from this directory.

```
mkdir c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\Xml
mkdir c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\Img
mkdir c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\Bin
mkdir c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\Tee
msk2xml MSK\*.MSK /oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW\XML
msk2xml MSK\stac\*.MSK
/oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW\XML
msk2png MSK\*.MSK ACID_MANUF_PLANT.ini
/oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW\IMG
msk2png MSK\stac\*.MSK ACID_MANUF_PLANT.ini
/oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW\IMG
dat2bmp MSK\KWAS.DAT * ACID_MANUF_PLANT.ini
/oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW\IMG
del c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\IMG\*.bmp
del c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\IMG\*.bak
Ini2Frameset ACID_MANUF_PLANT.ini
/oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW\
xml2aspx c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\*.xml
ACID_MANUF_PLANT.ini /oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW
Pum2Aspx MSK\*.pum /oc:\Asix\Applications\Acid_Manufacturing_Plant_WWW
copy c:\asix\XConnectNetCS.dll
c:\Asix\Applications\Acid_Manufacturing_Plant_WWW\bin\XConnectNetCS.dll
pause
```

6.4. Giving access to applications on WWW server

Aby udostępnić aplikację należy wykonać następujące kroki:

- Udostępnić jako wirtualny katalog serwera WWW katalog *c:\Asix\WebService* z domyślnymi opcjami (zaakceptować domyślną nazwę *WebService*),
- Udostępnić katalog, w którym znajduje się aplikacja internetowa jako wirtualny katalog serwera WWW (na przykład katalog *c:\Asix\Aplikacje\Wytownia_Kwasu_WWW* udostępnić jako *Wytownia_Kwasu_WWW*),
- Jeżeli użytkownicy aplikacji nie mają skonfigurowanego prawa logowania do serwera

In order to make application available you need to:

- Provide *c:\Asix\WebService* as virtual directory of WWW server with default options (accept default name of *WebService*),
- Provide the directory, in which the web application is located, as virtual directory of WWW server (for example, the

c:\Asix\ Applications\Acid_Manufacturing_Plant_WWW directory should be made available as *Acid_Manufacturing_Plant_WWW*).

- If rights to log into the Windows server have not been configured for the application users, then anonymous access should be activated in protections of both the virtual directories.

After the above-mentioned steps have been carried out, the application is available on the local level after the following address is typed in IE6 browser:

http://localhost/<name of virtual directory of application >

For application in the above-mentioned example the address would be *http://localhost/Acid_Manufacturing_Plant_WWW*.

The application is available within the local area network at *http://<name of computer in Windows system>/<name of virtual directory of application >*

For example, if computer name is *AsixWeb*, the application address is: *http://AsixWeb/Acid_Manufacturing_Plant_WWW*

6.4.1. Additional information on conversion of asix system application into web application

This section contains the list of **asix** system components that are not supported (or are supported provided that specific conditions are met) during the conversion into web application.

ACTIONS

Non-supported actions without warning:

- SCRIPT
- SET_BITS
- NOTHING

Non-supported actions with warning:

- HIDE_ALL
- VARIABLE_DESCRIPTION
- TIMER
- RUN
- PERFORM_INPUT
- REPORTS
- TABLE
- REPORT
- ASTREND
- START_UP

OBJECTS

1. General rules:

- text shading not supported
- text and picture twinkling not supported
- control not supported

2. Non-supported objects:

- CALCULATOR
- SWITCH
- SWITCH SET

3. Non-supported objects if dynamic:

- ELLIPSES
- LINES, LINES HV
- RECTANGLES

4. Restriction in supporting other objects:

BAR

- the object must be parameterised from the variablebase
- contour not supported

BUTTON

- key shortcuts not supported
- colours of lighted and shaded edge for pressed and normal position not supported
- rounding parameter not supported
- envelope parameter not supported
- frame thickness parameter not supported

CHART

- axis directions (EN only)
- chart description (Date-Time-Value only)

- axis X step, axis Y step, axis subinterval numbers not supported
- cursors not supported
- type of chart (line chart only)
- line parameters (colour only and full line with 1x1 thickness only)
- point markers not supported
- dynamic parameterisation
- legend formats not supported
- type of curve (current only)
- pre-calculation ('no' and 100% only)

DATE+TIME

- time may be displayed in two options only – with or without seconds
- date may be displayed in two options only – DD.MM.YY or DD.Month.YYYY

The latter form is used only if the 'month in words' option is selected and the 'year shortened' option is not selected

- capital letters parameter not supported
- shortened month parameter not supported
- leading zeros parameter not supported
- date connector parameter not supported

NUMBER

- the object must be parameterised from the variablebase
- vertical aligning not supported, middle space option is supported only

TEXT

- vertical inscriptions not supported

- spaces between lines not supported

TEXTS

- vertical inscriptions not supported
- spaces between lines not supported

Parameters of application conversion programs

1. xml2aspx – Program for conversion of mask files (as XML documents) into aspx pages

Application: xml2aspx <source> <ini file> [/o<output>] [/d] [/l<0..6>]

- obligatory parameters:

<source> - pattern of xml file names, e.g.: c:\asix\app\xml*.xml
<ini.file> - name of application's ini file, e.g.: c:\asix\app.ini

- optional parameters – may be given in any sequence

/o<dest> - object directory, e.g.: c:\asix\app\final
/d - debug option – objects of aspx pages will contain additional information – for testing purposes only!
/l<0..6> - log option – sets the level of keeping the user informed of the program execution, e.g.: /l6. Recommended use /l3.

Available options:

0 – no information
1 – errors only - to file
2 – errors and warnings - to file
3 - errors, warnings and information - to file
4 - errors only - to file and to screen
5 - errors and warnings - to file and to screen
6 – errors, warnings and information - to file and to screen

2. pum2aspx - Program for conversion of menu files (*.pum) into aspx pages

Application: pum2aspx <source> <ini file> [/o<output>] [/d] [/l<0..6>]

- obligatory parameters

<source> - pattern of pum file names, e.g.: c:\asix\app\xml*.pum
<ini.file> - name of application's ini file, e.g.: c:\asix\app.ini

- optional parameters – may be given in any sequence

/o<dest> - object directory, e.g.: c:\asix\app\final
/d - debug option – objects of aspx pages will contain additional information – for testing purposes only!
/l<0..6>- log option – sets the level of keeping the user informed of the program execution, e.g.: /l6. Recommended use /l3.

Available options:

0 - no information
1 - errors only - to file
2 - errors and warnings - to file
3 - errors, warnings and information - to file
4 - errors only - to file and to screen
5 - errors and warnings - to file and to screen
6 - errors, warnings and information - to file and to screen

3. ini2frameset – program to create the main file of web application – default.htm

Application: ini2frameset <ini file> [/o<output>]

- obligatory parameters
 - <ini.file> - name of application's ini file, e.g.: c:\asix\app.ini
- optional parameters – may be given in any sequence
 - /o<dest> - object directory, e.g.: c:\asix\app\final

Index

A			
Access to asix system directory	9		
Access to VariableBase	9		
Add function	39		
addSeries	38		
ALARM object	31		
Alarms	17		
archive data	15		
As2HTML - visualisation in web browser	21		
As2WWW - converter of applications	47		
Asix4Internet - visualisation and supervision via Internet	3		
ASPNET	5		
AsPortal	13		
B			
BAR object	25		
Binding DHTML object with current value of variable	45		
BUTTON object	26		
C			
Chart declaration	32		
Chart design	33		
Chart generated on the client side	37		
Configuration of operating system	5		
Configuring system access rights to Aslink module	5		
Conversion automation	48		
Conversion stages	47		
current data	14		
D			
Design preparation	23		
BODY Element	24		
BODY Section	24		
Data Source	24		
Design and source files of application	23		
HEAD Section	23		
Designs	20		
div2chartXML function	36		
F			
Funkcja addSeries	38		
Funkcja div2Chart	37		
G			
Generating VariableBase	9		
Giving access to applications on WWW server	13, 50		
goLive	38		
I			
InsertChartHTML function	39		
K			
Klasa ChartXMLData	39		
M			
Modification in appearance of NUMBER and BAR objects	42		
MoveChartPeriod function	40		
moveChartPeriodBy	39		
moveChartPeriodTo	38		
N			
NETWORK SERVICE	5, 9		
NUMBER object	24		
P			
PICTURES object	29		
Preparation of design directory	23		
R			
ReadData function	41		
Requirements of WINDOWS system	5		
S			
SetChartPeriod function	40		
Structure of directories in web application	48		
T			
TEXT object	28		
TEXTS object	30		
V			
Variable states	42		
VariableBase	19		
W			
WATCH object	31		
Web Service Server	11		
Web.Config	11		
Web.Config file	13		
WebService Server - installation	11		

1.	ASIX4INTERNET - VISUALISATION AND SUPERVISION VIA INTERNET	3
2.	REQUIREMENTS OF INTERNET MODULES.....	5
2.1.	REQUIREMENTS OF ASIX SYSTEM	5
2.2.	REQUIREMENTS OF WINDOWS SYSTEM	5
2.3.	CONFIGURATION OF OPERATING SYSTEM	5
2.4.	CONFIGURING SYSTEM ACCESS RIGHTS TO ASLINK MODULE	5
2.5.	ACCESS TO ASIX SYSTEM DIRECTORY	9
2.6.	GENERATING VARIABLEBASE	9
2.7.	ACCESS TO VARIABLEBASE	9
2.8.	REQUIREMENTS FOR DESIGNING THE APPLICATION USING As2HTML.....	10
3.	WEB SERVICE SERVER.....	11
3.1.	INSTALLATION	11
3.2.	WEB.CONFIG CONFIGURATION FILE	11
4.	ASPORTAL - PROCESS INFORMATION PORTAL.....	13
4.1.	CONFIGURATION	13
4.1.1.	<i>Giving access to applications on WWW server.....</i>	<i>13</i>
4.1.2.	<i>Web.config configuration file.....</i>	<i>13</i>
4.2.	DESCRIPTION OF FUNCTIONS AVAILABLE TO THE USER	14
4.2.1.	<i>Current data.....</i>	<i>14</i>
4.2.2.	<i>Archive data.....</i>	<i>15</i>
4.2.3.	<i>Alarms.....</i>	<i>17</i>
4.2.4.	<i>VariableBase.....</i>	<i>19</i>
4.2.5.	<i>Variable selection window.....</i>	<i>19</i>
4.2.6.	<i>Designs</i>	<i>20</i>
5.	AS2HTML - VISUALISATION IN WEB BROWSER.....	21
5.1.	PREPARATION OF DESIGN DIRECTORY	23
5.2.	DESIGN PREPARATION.....	23
5.3.	NUMBER OBJECT	24
5.4.	BAR OBJECT.....	25
5.5.	BUTTON OBJECT.....	26
5.6.	TEXT OBJECT	28
5.7.	PICTURES OBJECT	29
5.8.	TEXTS OBJECT.....	30
5.9.	ALARM OBJECT.....	31
5.10.	WATCH OBJECT	31
5.11.	OBIEKT WYKRES.....	32
5.11.1.	<i>Chart declaration.....</i>	<i>32</i>
5.11.2.	<i>Chart design.....</i>	<i>33</i>
5.11.3.	<i>Static chart generated on the server side.....</i>	<i>36</i>
5.11.4.	<i>Chart generated on the client side.....</i>	<i>37</i>
5.11.5.	<i>ChartXMLData class</i>	<i>39</i>
5.11.6.	<i>ReadFlags.....</i>	<i>41</i>
5.12.	MODIFICATION IN APPEARANCE OF NUMBER AND BAR OBJECT.....	42
5.12.1.	<i>Variable states</i>	<i>42</i>
5.12.2.	<i>Modification in appearance of NUMBER and BAR objects</i>	<i>43</i>
5.12.3.	<i>Modification in operation of NUMBER object</i>	<i>44</i>
5.13.	BINDING DHTML OBJECT WITH CURRENT VALUE OF VARIABLE	45
6.	AS2WWW - CONVERTER OF APPLICATIONS.....	47
6.1.	CONVERSION STAGES.....	47
6.2.	STRUCTURE OF DIRECTORIES IN WEB APPLICATION.....	48
6.3.	CONVERSION AUTOMATION	48
6.4.	GIVING ACCESS TO APPLICATIONS ON WWW SERVER.....	49
6.4.1.	<i>Additional information on conversion of asix system application into web application.....</i>	<i>50</i>