

Manual Web-Server xx7

Controls Division

0	Content	
0.1	Document History	0-4
0.2	Source list	0-4
0.3	Trademarks	0-4
1	Introduction	
1.1	Minimum requirements for the Web-Server	1-1
1.2	Features	1-1
1.3	Web-Server without hardware add-on components	1-1
1.4	HTML pages integrated in the Firmware	1-2
2	Structure and function	
2.1	HTML-Server	2-1
2.2	Data-Server	2-1
2.3	HTML pages, images, graphics etc.	2-1
2.4	Local directory	2-2
2.5	Web-Builder	2-3
2.6	Web-Connect	2-3
2.6.1	Connection possibilities	2-3
2.7	Web-Browser	2-4
3	Installation and Settings	
3.1	Web-Builder	3-1
3.2	Web-Connect	3-2
3.3	Web-Browser and Network settings	3-3
3.3.1	The PC's proxy server	3-3
3.3.2	Modem connection	3-3
3.3.3	The PC's cache memory	3-4
3.4	Settings and Test of TCP/IP	3-6
3.4.1	TCP/IP	3-6
3.4.2	IP address for PC	3-7
3.4.3	PC name	3-8
3.4.4	Test of the IP address and the TCP/IP protocol	3-8
3.4.5	Finding IP address	3-9
4	Configuration and Administration	
4.1	Web-Builder	4-1
4.1.1	Web-Builder structure	4-1
4.1.2	Using Web-Builder	4-3
4.1.3	Include HTML pages in user program	4-4
4.2	Web-Connect	4-6
4.3	Variable pages	4-7
4.3.1	Before use the Variable list	4-7
4.4	Description of the PCD media	4-8
4.4.1	Modifying entries in a variable page	4-9
4.5	Access to PCD data	4-9
4.6	Configuration of the Web-Server with the Saia xx7 I/O-Builder	4-11
4.6.1	Definition of serial port	4-12

4.6.2	Modem	4-12
4.6.3	Definition of the Web-Server	4-13
4.7	Configuration and definition of the Web-Server in the user program.....	4-14
4.7.1	Definition of serial port	4-14
4.7.2	Modems	4-14
4.7.3	Definition of Web-Servers	4-16
5	Example Web-Project	
5.1	Structure of pages.....	5-1
5.2	PCD data point into the HTML Pages	5-3
5.2.1	Page: input.htm.....	5-3
5.2.2	Page: output.htm.....	5-4
5.3	Link pages.....	5-10
5.4	Generation of DBs	5-12
5.5	Insertion of HTML pages in the user program	5-13
6	Troubleshooting	
6.1	Variable List (varlist).....	6-1
A	Appendix A	
A.1	How to create html pages without editor.....	A-1
A.2	CGI (Common Gateway Interface) of the Web-Server	A-1
A.2.1	Readval.exe	A-1
A.2.2	Writeval.exe	A-2
A.2.3	Ordervalues.exe.....	A-2
A.2.4	Readfile.exe	A-3
A.3	HTML coding.....	A-4
A.3.1	Writing values in the PCD	A-4
A.3.2	Enter the Password.....	A-4
A.4	Dynamic data update in the html page	A-5
A.4.1	The concept	A-5
A.4.2	Starting page.....	A-6
A.4.3	Update page.....	A-6
A.4.4	Force page	A-7
A.4.5	Display page	A-8
A.5	General principles of Web-Server technology	A-10
A.5.1	General notes concerning Web-Servers	A-10
A.5.2	TCP/IP protocol.....	A-10
A.5.3	IP addressing	A-11
A.5.4	Client-Server-Technologie.....	A-12
A.5.5	DNS - Domain Name Service	A-12
A.5.6	Routing and gateways.....	A-13
A.6	General principles of HTML	A-14
A.6.1	Mark up elements.....	A-14
A.6.2	Cross-linking with hyperlinks.....	A-14
A.6.3	Software-independent plain text.....	A-15
A.6.4	Universal in application	A-15
A.6.5	Formatting for HTML elements	A-15



A.6.6	A programming language for WWW pages.....	A-16
A.6.7	JavaScript and HTML.....	A-16
A.6.8	JavaScript, JScript, ECMA-262, language versions.....	A-17
A.6.9	HTML editors.....	A-17
B	Appendix B	
B.1	Icons	B-1
B.2	Adresse.....	B-1

0.1 Document History

Document-No.	Edition	Modification	Publication	Remarks
26/775	E3	31.12.2003	29.02.2004	General revision
	E4	2008-06-26	2008-06-26	Chapter 3.2 and 4.2 : hint to 26/800 Chapter 5.6 and 6.1 deletet
	EN05	2014-03-26		Change of Logo

0.2 Source list

Parts of this manual (in particular the Appendix) have been copied from TeamOne's SELFHTML file.

SELFHTML can be downloaded from Internet under the WWW address:

<http://www.teamone.de/selfhtml/>.

0.3 Trademarks

- Saia PCD® is a registered trademark of Saia-Burgess Controls AG
- STEP®, SIMATIC®, S7-300®, S7-400® and Siemens® are registered trademarks of Siemens Ltd.
- Windows 95/98, Windows NT, Word, Excel, PowerPoint, FrontPage and Microsoft Internet Explorer are registered trademarks of The Microsoft Corporation.
- Netscape Navigator is a registered trademark of The Netscape Communications

Technical modifications and changes depending on state of the art.

Saia-Burgess Controls AG, 2002. © All rights reserved.

1 Introduction

1.1 Minimum requirements for the Web-Server

1

The following minimum requirements are necessary for the Web-Server functionality:

PCD System	PCD2.M157, M177, M257, M487
PCD FW	V2.200
PCD serial port	Any one
PC	<ul style="list-style-type: none"> • IBM PC or compatible • minimum 133 MHz Pentium with 32MB RAM • Microsoft Windows 95/98/2000/XP or Windows NT 4.0. • Winsock 1.1 API compliant networking package • Telephony Application Programming Interface (TAPI) • Microsoft Internet Explorer 5.5 or Netscape Navigator 6

1.2 Features

Within the Saia PCD, the integrated Web-Server allows the user to create an own control and monitoring function quick and efficient with the help of HTML pages.

Standard Microsoft programs (such as Word, Excel, PowerPoint or FrontPage) are used to create these control and monitoring functions.

Traditional Web-Servers require a large amount of resources, e.g. in memory or computing capacity. PCs do not have unlimited resources in their processors and they may be fully utilised for controlling machines or processes. Therefore, control technology uses only Web-Servers in the form of additional PCs or expensive add-on components (co-processors).

1.3 Web-Server without hardware add-on components

Is it possible to have a Web-Server without hardware add-on components? Yes it is. To define the precise function of add-on components it is necessary to examine the structure of a Web-Server more closely:

The job of a Web-Server is to prepare and administer files (HTML pages, image files, Java Applets, etc.) and despatch those files on request. Whereas file administration takes a simple form, this is not the case for file despatch – since all communication must take place with standard browsers via the TCP/IP protocol. However, TCP/IP places heavy demands on memory resources and CPU power.

Saia-Burgess Web-Server functions on the PCD xx7 series, are splitted: The file administration part is located in the PCD's CPU and the part treating the request from TCP/IP communications is installed on the PC that is used to call control and monitoring functions. It's then possible to communicate between PCs and PCDs via a simple, efficient protocol that uses significant less resources than TCP/IP. Reduced to the essential transfer function, the protocol also guarantees deterministic behaviour and does not impact the real-time behaviour of the PCD.

Standard browsers, like Microsoft Internet Explorer or Netscape Communicator, request files from Web-Servers under the TCP/IP protocol. To enable communication with the Web-Server in the PCD, the Web-Connect software must be installed on that PC, which works as the communication part of the Web-Server. Web-Connect receives requests from the browser in the TCP/IP protocol and forwards them with an efficient protocol to the PCD, this by the communication configured, it could be serial direct or with a modem. PCD answers are converted back into TCP/IP and conveyed to the browser.

Splitting the Web-Server dispenses with the need for additional TCP/IP co-processors or TCP/IP add-on components for the PCD's. It's then possible to connect and use the PCD Web-Server by a point-to-point serial line or also by a modem. PCs connected in a local network (LAN) can also access the Web-Server PCD via Web-Connect software.

Advantages of the SBC Web-Server:

- Simple, low-cost creation of control and monitoring interfaces with familiar Microsofttools, such as Word, Excel, PowerPoint, FrontPage, etc.
- No Run time license necessary. In fact no license are necessary to operate or connect to the Web-Server.
- Operation of the control and monitoring interface with standard browsers, like Microsoft Internet Explorer or Netscape Communicator. Browsers have developped a standard user interface (with the familiar "Forward", "Back", "Home" and "Refresh" buttons) so that untrained staff can immediately operate a control and monitoring interface via a browser. Navigation by simply clicking on "Left" is intuitive and presents no problems.
- No expensive infrastructure required on PCD or PC, such as Ethernet networks, Internet providers, Interface modules, etc.
- With an integrated modem, ideal for Tele servicing of machines and processes over great distances.
- With the 4 built-in variable pages, the status of each process data point can be watched and, if necessary modified.
- The access to the control and monitoring interfaces is protected with 4, individual definable passwords.

1.4 HTML pages integrated in the Firmware

The Web-Server comes from the factory with 4 stored HTML pages that can be displayed with any standard browser. To acces these pages no special actions are necessary. The 4 pages stored in FW are as follows:

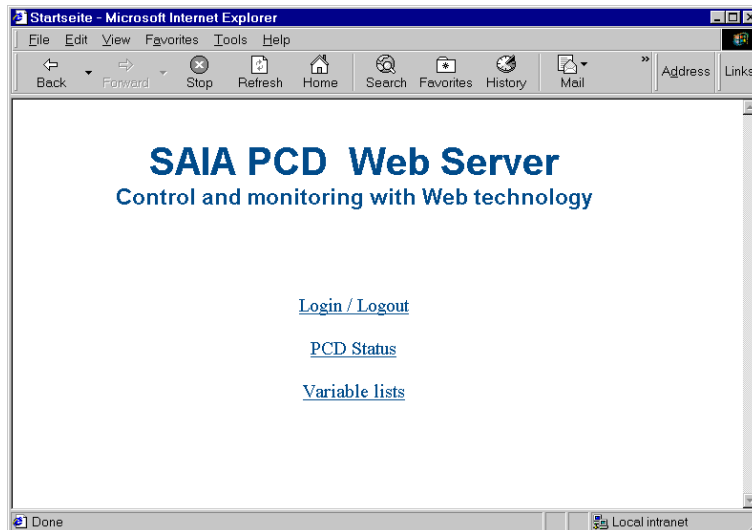
start.htm	Standard start page: If no page is referenced on the URL when connecting to a station, the start.htm page is always the first file to be sent to the Web-Browser. A page with this name always have to be included in the web project. Starting from this page, links are then made to other HTML pages.
pwdform.htm	Standard password entry page: The password can be entered on this page. The user's own pwdform.htm page can be defined.
status.htm	Page where you can see the type of PCD, its actual status and the operating system version (firmware).
varlist.htm	Standard variable list: Under this menu item 4 variable pages are stored that can be edited as desired by the user. All PCD data points can be accessed from these pages (if permitted by password level). The user's own varlist.htm page can be defined.

Rules governing the 4 HTML pages already contained in the Web-Server from factory:

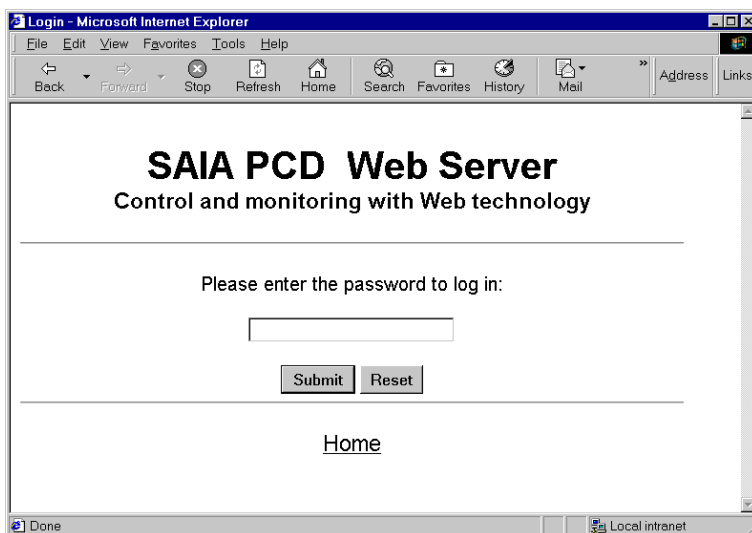
- If HTML pages are stored in the user program and they have the same name as the ones mentioned (start.htm, pwdform.htm, etc...), when the Web-Browser invokes these pages it will be loaded with the ones stored in the user program.
- If the HTML pages listed above are invoked in the Web-Browser but do not exist in the user program, the default standard pages will be loaded into the Web-Browser.

1

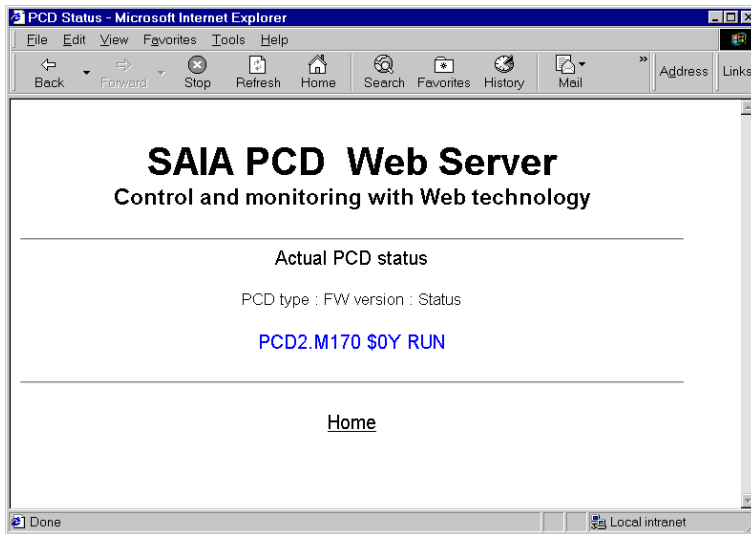
Page: **start.htm**



Page: **pwdform.htm**

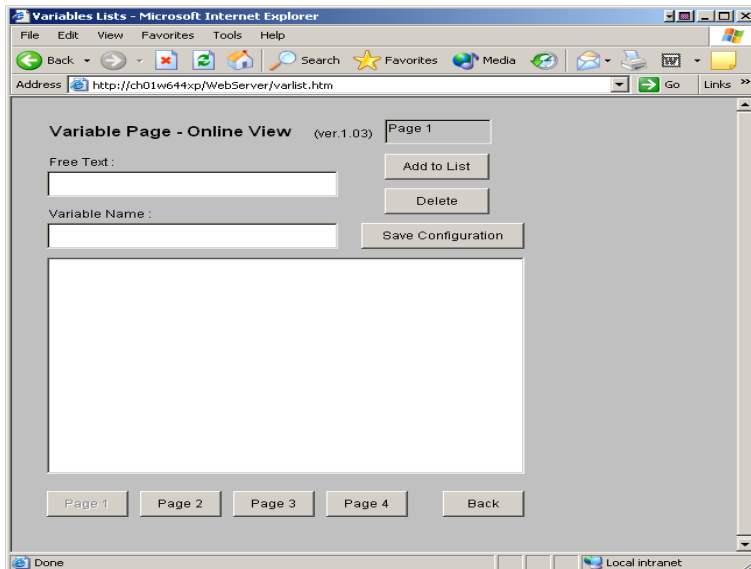


Page: **status.htm**



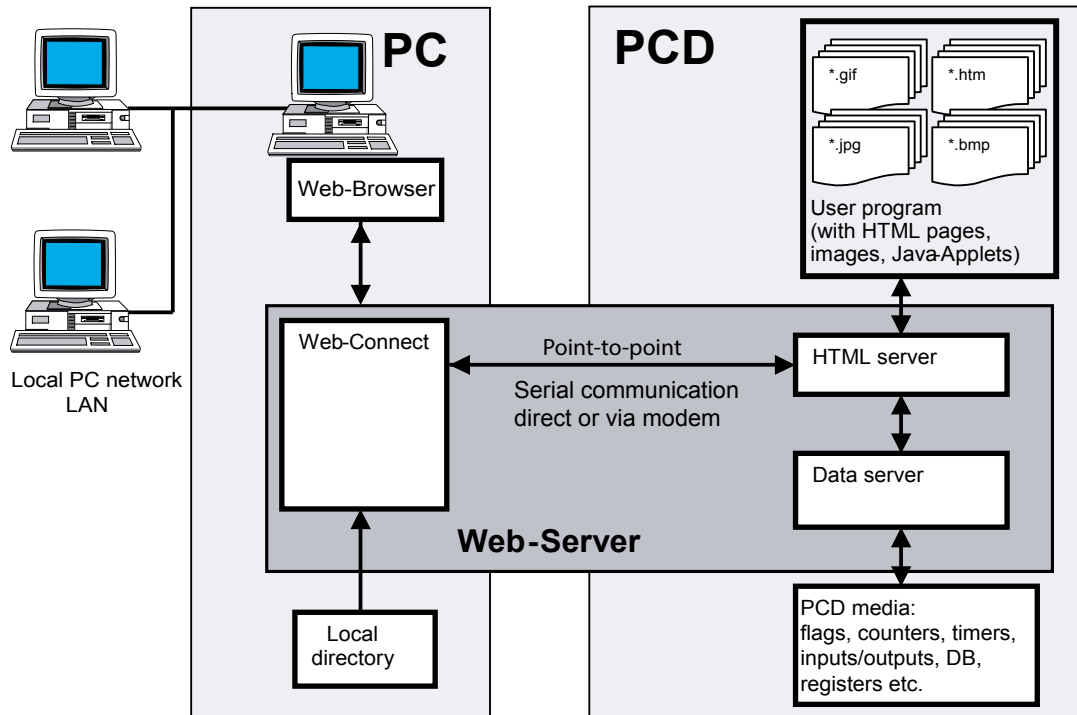
1

Page: **varlist.htm**



2 Structure and function

The PCD's integral Web-Server comprises the following individual elements:



2.1 HTML-Server

The HTML-Server is the heart of the entire Web-Server. It sends HTML pages requested by the Web-Browser (and any necessary images/files) across the S-BUS Driver to the PC.

HTML pages, images or files are stored in the user program memory of the PCD, it belongs to the user program and so it's downloaded from the PG5.

In the HTML server, HTML pages are also checked for their process data point identification key (PDP key). As soon as the HTML-Server finds a PDP key in a HTML page, it sends that PDP key to the Data-Server.



The HTML-Server checks only files with suffix *.HTML or *.HTM for a possible PDP key to be transferred to the data server.

This means that it is compulsory for all files containing a PDP key to have the suffix *.HTML or *.HTM.

2.2 Data-Server

The Data-Server processes PDP keys received from the HTML-Server and transfers them in the requested PCD data directly from the PCD's memory to the HTML-Server.

The Data-Server can access DB, flags, register, inputs, outputs, timers, counter, texts and PCD status.

2.3 HTML pages, images, graphics etc

One of the Web-Server's essential advantages is the creation of a control and monitoring interface using Microsoft standard tools. Any software tool that generates

HTML can be used to create HTML pages. For example, individual pages of a control and monitoring project can conveniently be edited as a text file using Word.

Any tables, graphics, diagrams, photographs, etc. that can be displayed by the Web-Browser may be inserted. The full functionality of the relevant Microsoft application can be used for this. No prior knowledge is required to link any number of pages into complex menu structures.

To integrate current PCD data into the control and monitoring function, simple text commands in the form of a PDP key are inserted in the text file. In this way PCD media can be displayed during run-time in various formats.

All HTML pages, images or files belonging to a control and monitoring application are stored in the PCD as data blocks (DB's) of the user program. The user can store any file in any preferred format and with whatever suffix (*.htm, *.html, *.bmp, *.giv, *.txt, *.zip or any other) in the PLC, as long as the file does not exceed **32kByte** in size. Each HTML page, image or file belonging to a control and monitoring application is stored in an individual DB. The control and monitoring application can contain up to **1022** separate HTML pages, images or files and can take up the entire user program memory.



- It should be noted that information for the control and monitoring application is stored in the DBs of user program memory, with the same DBs being occupied by the user program and the control and monitoring application. Make sure that the DBs of the control and monitoring application are not accessed in the user program.
- The HTML server only checks files with suffix *.HTML or *.HTM for the PDP key.



If the control and monitoring application needs to display large HTML pages, images or files that doesn't fit in the PCD memory then there is the possibility to store this HTML pages, images or files on the hard disk of the PC where the Web-Connect software is running. The HTML pages images or files have to be stored in the Web-Connect directory.

2.4 Local directory

The local directory, is a directory on the PC, where the Web-Connect software is running. By default its name and location is "C:\WebPages", this location and name can be changed in the Web-Connect option and during the Software installation.

This directory can contain files, which from now on will be called "the local files".

It is possible to save files in this directory which we don't want to save in the PCD. This option is very useful when the files we want to save are quite huge. If we would save these files in the PCD it could take a long time to load them, not so if they are saved in the local directory. The mechanism will be the following, the file is requested to the PCD and if the file doesn't exist there, the Web-Connect software will check then the local directory to find and return it. This operation is totally transparent for the user.

Here you also save the local web-site, like the default.html page and all files going with it, this will be treated in the chapter "Web-Connect".

2.5 Web-Builder

The PCD xx7 controllers are programmed with Siemens® STEP®7 programming software. Control and monitoring files are stored in the PLC's user program memory. STEP®7-compatible PCD2 controllers (STEP®7 Siemens®) are equipped with 512 kByte of main memory. This also allows extensive control and monitoring functions to be stored in the PLC.

All files belonging to the control and monitoring function (HTML pages, images, etc.), are stored in data blocks. First the web files must be converted into STEP®7 source files. Conversion takes place with the easy-to-use Web-Builder software tool. It can be used to select HTML pages and images to be converted, and define the range of data blocks where HTML pages are to be stored. Then it is only necessary to import the STEP®7 source file into the Siemens® STEP®7 programming software. This takes place in the usual way with the "Insert, External Source" function. Then the "Process, Translate" function is used to translate the source file and add the data blocks to the STEP®7 project. Afterwards the latter can be loaded into the PLC.

There exists two versions of the Web-Builder:

Demo version:

- Don't needs any password for the installation
- Can convert maximal two HTML pages, images, etc in to DB's
- Not possible to select the option VarList Sites.

Full version:

- Needs a password for the installation
- No limitation on the number of HTML pages, images, etc that can be convert.

2.6 Web-Connect

To display in the Web-Browser the control and monitoring functions stored in the controller, it is necessary to start up the Web-Connect, PC software.

This software receives the request coming from the clients (Browser) and forwards them to the right PCD station using the adequate way and vice-versa.

This software is doing the link between the PC-Browser and the communication to the PCDs.

The Web-Connect program enables the user to :

- Configure the station that will be accessed from this PC.
- Configure the channels that will be used to access the Station.
- Forward request from the PC to the PCDs
- Forward request from other PC on the same LAN to the PCDs.

As you can see there are few different ways to access to a PCD, we can make two categories of connection from that, direct access to the PCD and indirect access to the PCD.

2.6.1 Connection possibilities

Direct mode:

The Web-Browser which is making the request and the Web-Connect software are running both on the same PC. But then depending which is the PCD concerned by the request, it can take different connection types, here they are:

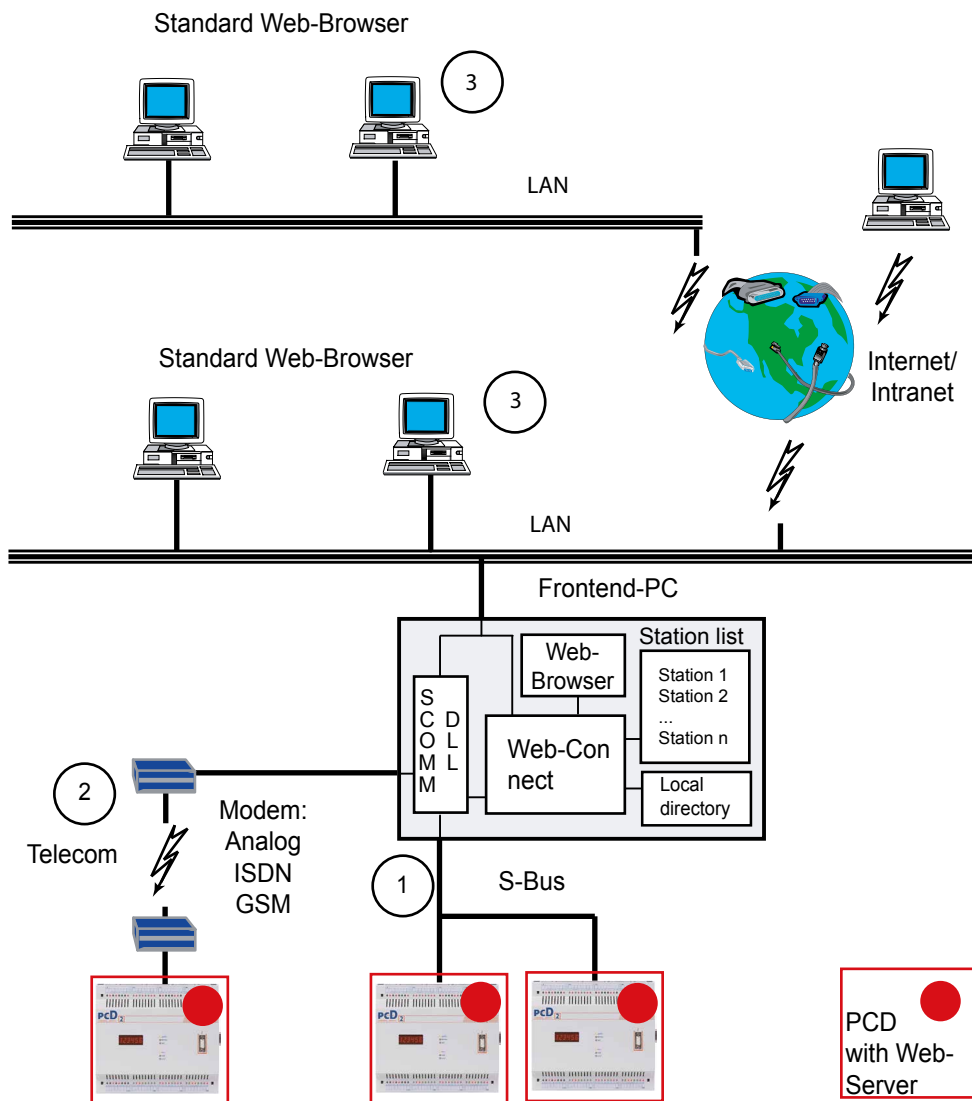
- ① Direct connection to the serial port from the PC.
The PCD is directly connected to the PC with a serial line or by a gateway.
- ② Direct world-wide connection via modem

The PCD is equipped with a modem and communicates with the PC via that modem.

Indirect mode:

- ③ Indirect, via LAN network
Other PCs, where no Web-Connect software is running, can also make requests to the PCD with the Web-Server. To do so they have to do an indirect access. In fact the request will go first to the PC where Web-Connect software is running and then, this PC will send the request to the PCD. The answer will be sent back to the request originator on the same way.

This both ways of working are possible, because the Web-Connect software, doesn't make a difference if the request is coming from the local Web-Browser or if it is coming from a Web-Browser running on an other PC of the LAN.



2.7 Web-Browser

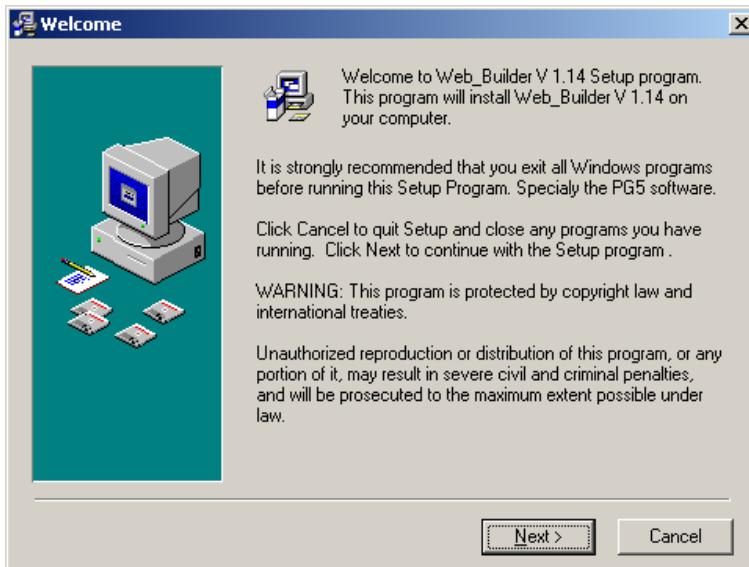
We recommend to use Netscape Navigator or Microsoft Explorer as Web-Browser to display HTML pages, images or files stored in the PCD, which belong to the control and monitoring application.

3 Installation and Settings

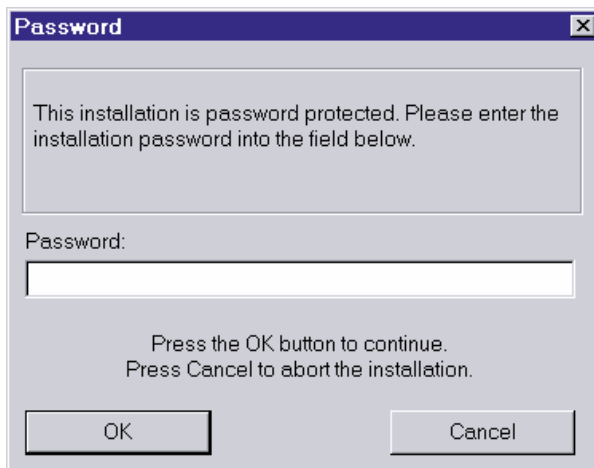
3.1 Web-Builder

Web-Builder is installed with a setup program:

Make sure, that you have a license for the Web-Builder. In Windows, select the **“Start“**, **“Execute“** menu and select the path used to save the Web-Builder. The **“Setup.exe“** file must be selected and started up in this path. Web-Builder will then be installed on the PC.

3

A customized password must be entered in the relevant menu. (Only for the Full version)



When the correct password has been entered, Web-Builder SW is installed by default in path C:\Programs\SBC\xx7\Web-Builder. In the Windows program menu the following program call has been installed for the Web-Builder under Programs, SBC, xx7:



3.2 **Web-Connect**

For installation and settings please refer to the manual 26-800 Web-Connect. This document can be found on Saia support site www.sbc-support.com under Software.

3.3 Web-Browser and Network settings

If no Web-Browser has yet been installed on the PC, this must be done in accordance with the installation information for each individual Web-Browser.

Microsoft Virtual Machine must be installed.

Installation of Microsoft Virtual Machine takes place during the installation of the Web-Browser.

MS Internet Explorer or Netscape Navigator must be installed (version see **1.1 Minimum requirements**).

Web pages from which the Web-Browsers can be downloaded.

MS Internet Explorer in German:

<http://www.eu.microsoft.com/germany/>

MS Internet Explorer in English:

<http://www.microsoft.com/>

Netscape Navigator in German:

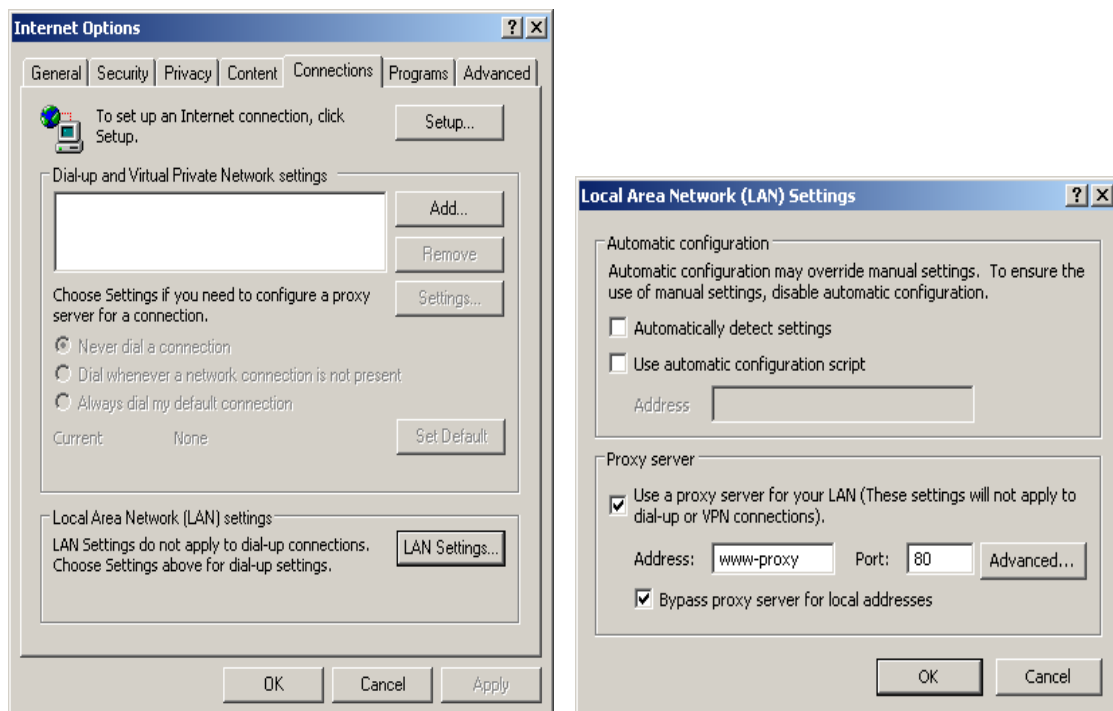
<http://home.de.netscape.com/de/>

Netscape Navigator in English:

<http://home.netscape.com/>

3.3.1 The PC's proxy server

If the PC is connected to Internet via a proxy server, the proxy server must be bypassed for local addresses. Taking the Internet Explorer as an example, this can be done with the following option:



“Tools”→”Internet Options...”, select option “Connections” and select button “LAN Settings” by selecting option “Bypass proxy server for local addresses”.

3.3.2 Modem connection

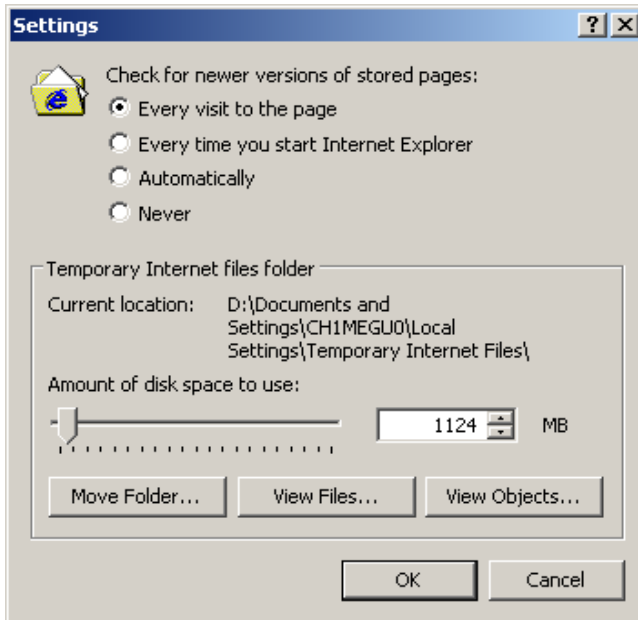
If the PC communicates with Internet via a modem connection, the “Never dial a connection” option must be selected. This option appears in Internet Explorer under “Tools”→”Internet Options”..., tab “Connections”.

It is necessary to select this mode, to avoid that every time the PC tries to connect to internet just to get access to the local software Web-Connect.

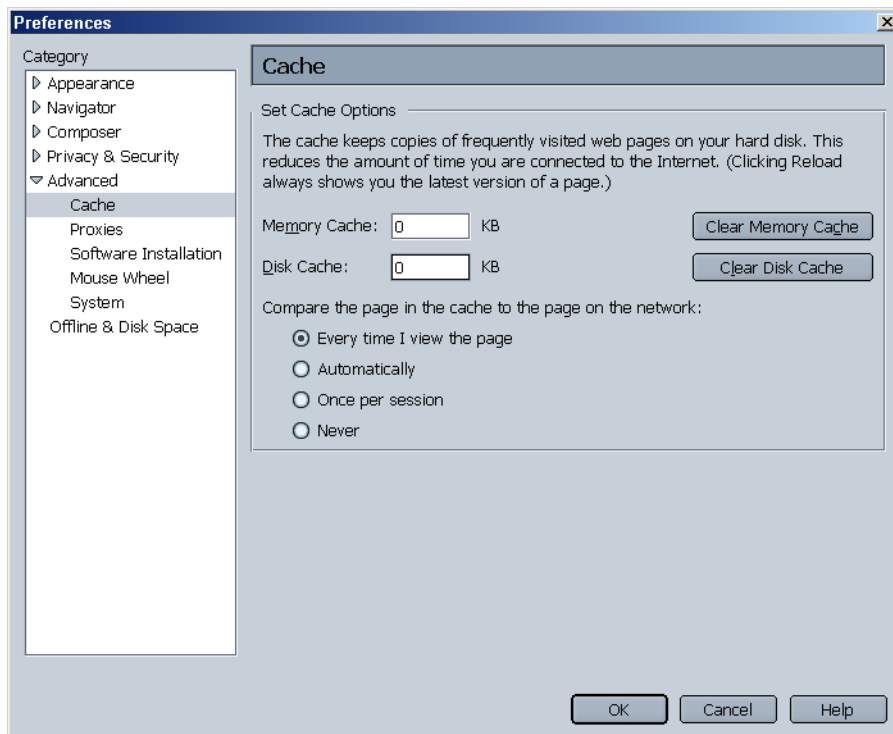
3.3.3 The PC's cache memory

In order to refresh HTML pages periodically, the Web-Browser's cache memory must be switched off.

This is possible with Internet Explorer under "Tools"→"Internet Options...", "General", "Settings...", selecting the option "Check for newer versions of stored pages" "Every visit to the page".

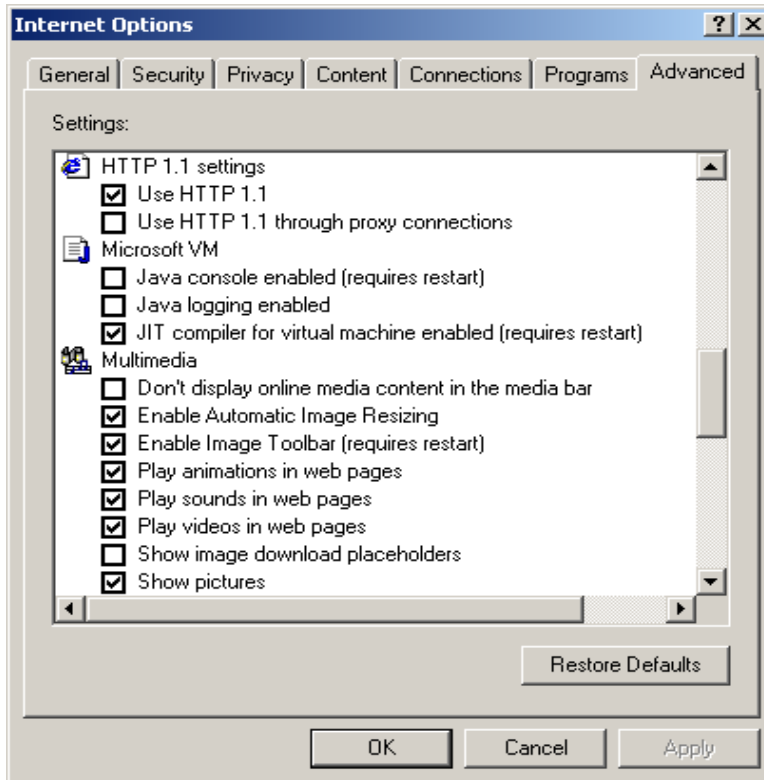


With Netscape Communicator – menu: "Edit"→"Preference":





To optimise the execution of Java Applets under Internet Explorer, the Java Console option must be disabled.

**3**

To check Java Console settings, go into Internet Explorer and select “**Tools**”, “**Internet Options**”, “**Advanced**”, “**Microsoft VM**” (Virtual Machine).

This involves ensuring that the “**Java Console Enabled**” option is **not** selected.

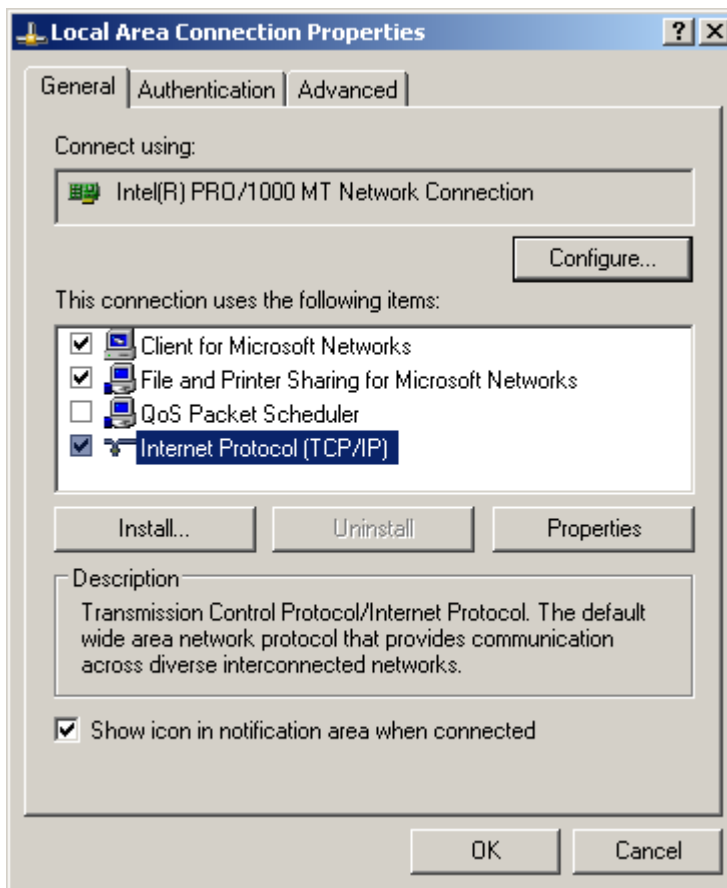
3.4 Settings and Test of TCP/IP

3.4.1 TCP/IP

The PC, on which the Web-Browser is supposed to display the PCD's control and monitoring functions must have TCP/IP.

It is possible to test if the TCP/IP protocol is installed on the PC by checking the 'Protocol' option under **"Start", "Settings", "Control Panel", "Network Connections", e.g. "Local Area Connection"**.

The TCP/IP protocol is installed on the PC if a TCP/IP protocol is defined under the tab **"General"**.

3

3.4.2 IP address for PC

You will see later, that access to the PCD Web-Server, is made with a Web-Browser. In this Web-Browser a valid assignment to the PC, where the Web-Connect SW is running has to be defined. This assignment is done through the URL which has to include the IP address or the PC name.

Now, if you don't want to use the PC name and prefer to use the IP address, this has to be a fixed one. You first need to have, at your disposition, a unique and valid IP address.

Having its own, unique address means the option **"Obtain an IP address automatically"** under the PC's Network TCP/IP settings must **not** be selected. This option appears in WIN 2000 under **"Start"**, **"Settings"**, **"Control Panel"**, **"Network Connections"** and e.g. **"Local Area Connection"** and then **"General"**.

Under the same item it's possible to define under **"Properties"** the IP-address of the PC.

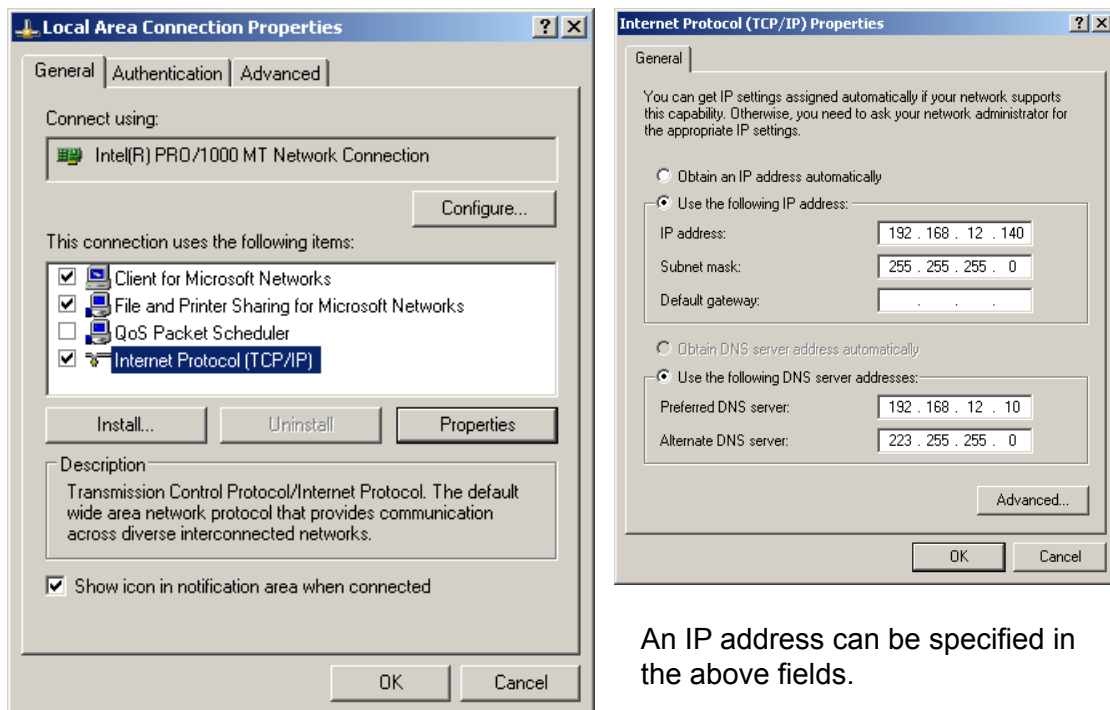


Every PC has the same default IP address as a factory setting.

This is: 127.0.0.1

Therefore, if no IP address has been specified, this address can be used in the Web-Browser to access to the PLC's HTML pages.

For WIN 2000

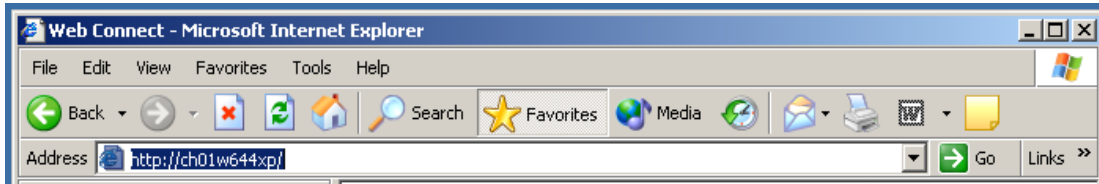


An IP address can be specified in the above fields.

3.4.3 PC name

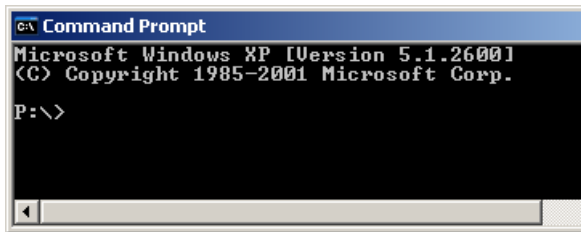
As said before, it is also possible to work with the PC name instead of the IP address. So in the URL you can enter **localhost** and you will find your PC name.

In our case the name is ch01w644xp.

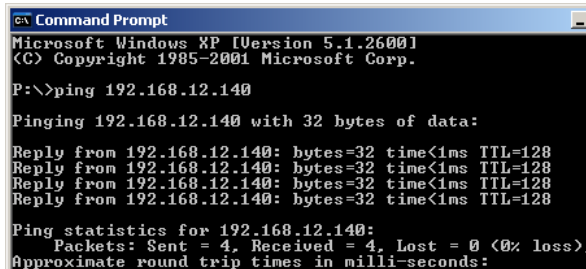


3.4.4 Test of the IP address and the TCP/IP protocol

To check if a TCP/IP protocol has been installed on the PC and if the IP address is correct, the following test can be carried out:

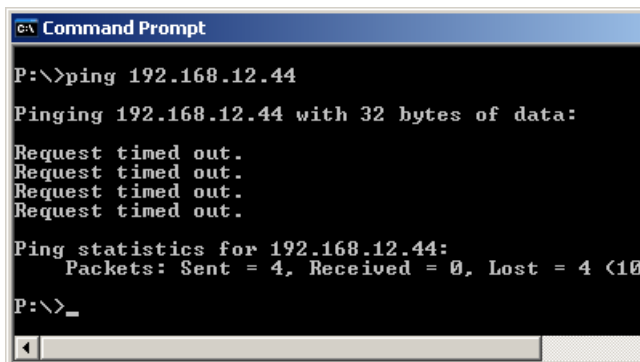


Start the DOS console with “**Start**”, “**Programs**”, “**Accessories**”, “**Command Prompt**”.



Enter the command ping “**IP Number of PC**“. e.g. ping 192.168.12.140

If the IP address is correct and TCP/IP is installed, the following reply is shown on the screen.



If the IP address doesn't exist, the following reply is shown on the screen.

```
C:\ Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

P:\>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:

Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

If the PC's IP address is not known, an attempt can be made by using the address 127.0.0.1.

3

It is also possible to check the PC's name you are using with the ping function, look for **localhost**:

```
C:\ Command Prompt
P:\>ping localhost

Pinging CH01W644XP [127.0.0.1] with 32 bytes of data:

Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

P:\>
```

“ping localhost”

In this example the PC name is: ch01w644XP.

3.4.5 Finding IP address

The function “**ipconfig**” is another possibility to find out what is configured on your PC.

```
C:\ Command Prompt
P:\>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . . : 
    IP Address. . . . . : 192.168.12.140
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.12.221

P:\>
```

4 Configuration and Administration

4.1 Web-Builder

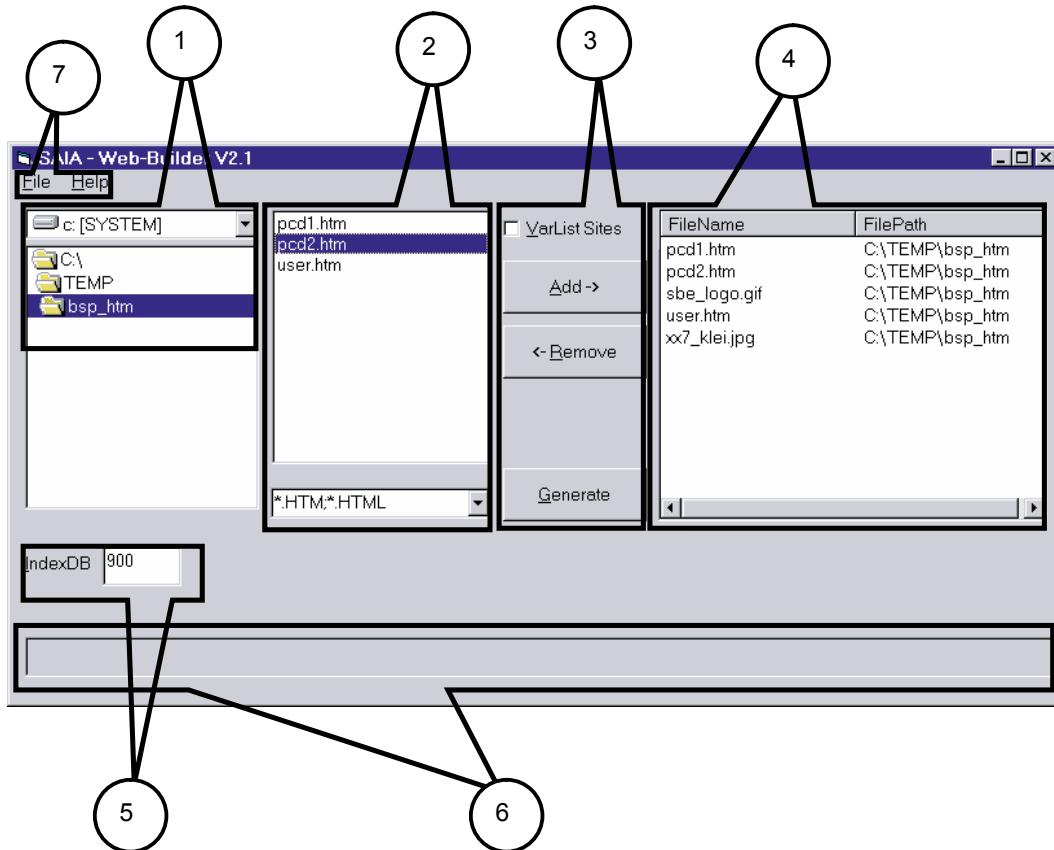
4.1.1 Web-Builder structure

Web-Builder can be called from the Task List by activating the field



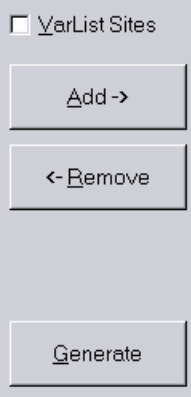

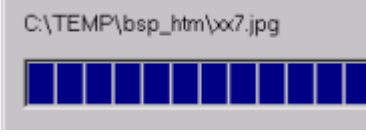

The following screen is then displayed:

4



Individual screen components are:

1	<p>Navigator, which can be used to select the desired path for saving control and monitoring files and images that are to be converted.</p>	
2	<p>Select window: Only the files contained in the folder "html" of your CPU, are displayed here. The following display masks can be selected: *.HTM, *.HTML: to show HTML files *.gif, *.jpg, *.bmp: to show graphics and image files *.* to show all files</p>	

<p>3</p>	<p>Function buttons, which can be used to add files or images from the select window to the translation window, or to remove them.</p> <ul style="list-style-type: none"> • Add→ Files marked in the select window are copied to the translation window. • ←Remove Files marked in the translation window are deleted from the translation window. • Generate All files and images entered in the translation window are converted into DB's. The DB's are stored in a *.IL file. When this button is activated, a window is displayed where the *.IL file in which to store DB's can be specified. • VarListSites If this option is selected, 4 files are generated in the translation window to allow storage of the Web-Server's variable pages. This enables variable page entries, which have been edited on-line, to be stored on the PCD. If this option is not selected variable pages can still be edited, but cannot be saved to the PCD. 											
<p>4</p>	<p>Translation window All files and images to be translated must be listed in this window. Any file listed in this window will be converted into a DB.</p>	<table border="1"> <thead> <tr> <th>FileName</th> <th>FilePath</th> </tr> </thead> <tbody> <tr> <td>Input.htm</td> <td>C:\TEMP\bsj</td> </tr> <tr> <td>Output.htm</td> <td>C:\TEMP\bsj</td> </tr> <tr> <td>user.htm</td> <td>C:\TEMP\bsj</td> </tr> <tr> <td>xx7.jpg</td> <td>C:\TEMP\bsj</td> </tr> </tbody> </table>	FileName	FilePath	Input.htm	C:\TEMP\bsj	Output.htm	C:\TEMP\bsj	user.htm	C:\TEMP\bsj	xx7.jpg	C:\TEMP\bsj
FileName	FilePath											
Input.htm	C:\TEMP\bsj											
Output.htm	C:\TEMP\bsj											
user.htm	C:\TEMP\bsj											
xx7.jpg	C:\TEMP\bsj											
<p>5</p>	<p>IndexDB Definition of the first DB in which files and images will be stored. The IndexDB number must also be defined in the user program's CDB. A UDT is also generated with the index number. The number of DBs generated is equal to the sum of files entered in the translation window + 1.</p>											
<p>6</p>	<p>Progress Bar While DBs are being generated, the name of the DB currently in translation and the progress of that translation are represented graphically.</p>											
<p>7</p>	<p>When DB generation is complete, this window indicates which DBs have been generated. The number of DBs generated is equal to the sum of files entered in the translation window + 1.</p>											



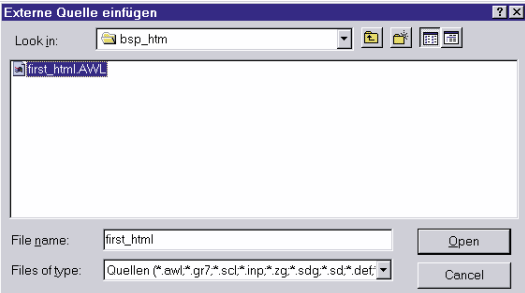
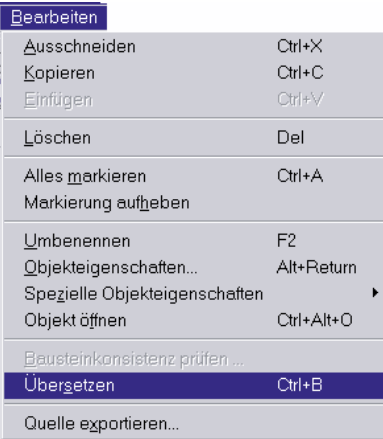
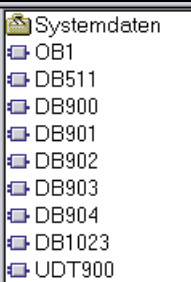
<p>8 Menu Structure File</p> <ul style="list-style-type: none"> • New Opens a new window • Open Opens an existing project, which reproduces the files and images entered in the translation window, together with their appropriate paths. • Close Closes the window • Save/Save as The files and images entered in the translation window, together with their appropriate paths, can be saved as a project with this item. • Print Prints information belonging to current project • Exit Shuts down the Web-Builder 	
---	--

4.1.2 Using Web-Builder

In general, all files and images stored in the PCD for control and monitoring functions must be stored as Data Blocks (DBs). Files and images are converted and saved using the Web-Builder. The following procedure can be used for this purpose:

<p>1 Start Web-Builder</p>									
<p>2 Select path in which files and images are stored</p>									
<p>3 Select files and images that must be converted into DBs by adding them to the translation window</p>	<table border="1"> <thead> <tr> <th>FileName</th> <th>FilePath</th> </tr> </thead> <tbody> <tr> <td>Input.htm</td> <td>C:\TEMP\bsp_htm</td> </tr> <tr> <td>Output.htm</td> <td>C:\TEMP\bsp_htm</td> </tr> <tr> <td>user.htm</td> <td>C:\TEMP\bsp_htm</td> </tr> </tbody> </table>	FileName	FilePath	Input.htm	C:\TEMP\bsp_htm	Output.htm	C:\TEMP\bsp_htm	user.htm	C:\TEMP\bsp_htm
FileName	FilePath								
Input.htm	C:\TEMP\bsp_htm								
Output.htm	C:\TEMP\bsp_htm								
user.htm	C:\TEMP\bsp_htm								
<p>4 Enter the IndexDB</p>									
<p>5 Conversion of files and images</p>									
<p>6 If desired, information from the translation window can be saved in a project file. This project file can be re-loaded later</p>									

4.1.3 Include HTML pages in user program

1	Start Siemens® Simatic® Manager of Step®7	
2	Include external sources with menu item Insert, External source	
3	Select the *.IL file generated with the Web-Builder	
4	Translate the newly included source file with the menu item: Edit, Compile	
5	After translation, DBs stored in the source file have been set out. In the example on the left, these are: DBs 900 to 904 and UDT900	

4

6	<p>Definition of web server in CDB. In the example, this is CD 1023. The IndexDB must at least be defined in it. If no port is defined, the default definition is port 1 with 19,200 Baud, 8 databits, odd parity and 1 stop bit.</p>																														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Adresse</th> <th style="text-align: left;">Name</th> <th style="text-align: left;">Typ</th> <th style="text-align: left;">Anfangswert</th> <th style="text-align: left;">Kommentar</th> </tr> </thead> <tbody> <tr> <td>0.0</td> <td></td> <td>STRUCT</td> <td></td> <td></td> </tr> <tr> <td>+0.0</td> <td>Identificator</td> <td>STRING[12]</td> <td>'SAIA xx7 CDB'</td> <td>CDB Identification</td> </tr> <tr> <td>+14.0</td> <td>WebPara</td> <td>STRING[30]</td> <td>'COM1:PTP_MPI,RS232,38400,8,o,1'</td> <td>Parameter of serial Port</td> </tr> <tr> <td>+46.0</td> <td>IndexDB</td> <td>STRING[15]</td> <td>'WEB:INDEXDB=900'</td> <td>Index DB -> identical with Web-Bu:</td> </tr> <tr> <td>=64.0</td> <td></td> <td>END_STRUCT</td> <td></td> <td></td> </tr> </tbody> </table>		Adresse	Name	Typ	Anfangswert	Kommentar	0.0		STRUCT			+0.0	Identificator	STRING[12]	'SAIA xx7 CDB'	CDB Identification	+14.0	WebPara	STRING[30]	'COM1:PTP_MPI,RS232,38400,8,o,1'	Parameter of serial Port	+46.0	IndexDB	STRING[15]	'WEB:INDEXDB=900'	Index DB -> identical with Web-Bu:	=64.0		END_STRUCT		
Adresse	Name	Typ	Anfangswert	Kommentar																											
0.0		STRUCT																													
+0.0	Identificator	STRING[12]	'SAIA xx7 CDB'	CDB Identification																											
+14.0	WebPara	STRING[30]	'COM1:PTP_MPI,RS232,38400,8,o,1'	Parameter of serial Port																											
+46.0	IndexDB	STRING[15]	'WEB:INDEXDB=900'	Index DB -> identical with Web-Bu:																											
=64.0		END_STRUCT																													
7	<div style="display: flex;"> <div style="flex: 1;"> <p>Load user program.</p> <p>If it is the first time the CDB has been loaded, or if entries in the CDB have been changed, the controller power supply must be switched off and on to activate CDB entries</p> </div> <div style="flex: 1; border: 1px solid gray; padding: 5px;"> <p>Zielsystem</p> <p>Zugangsberechtigung ▶</p> <p>Laden Ctrl+L</p> <p>Laden in PG</p> </div> </div>																														

4.2 Web-Connect

For details please refer to the manual 26-800 Web-Connect. This document can be found on Saia support site www.sbc-support.com under Software.

4.3 Variable pages

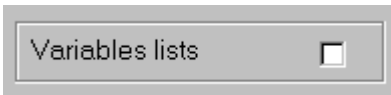
The web server includes 4 variable pages that can be used to access and write to any PLC data point during run-time. Variable pages are suitable for service and debug purposes and correspond to VATs (variable tables) under Step®7 from Siemens®.

Each variable page can have up to 16 fully definable process data points. For each process data point a free text can also be defined. Passwords protect the functionalities of reading and writing process data points and of saving variable pages to the user program.

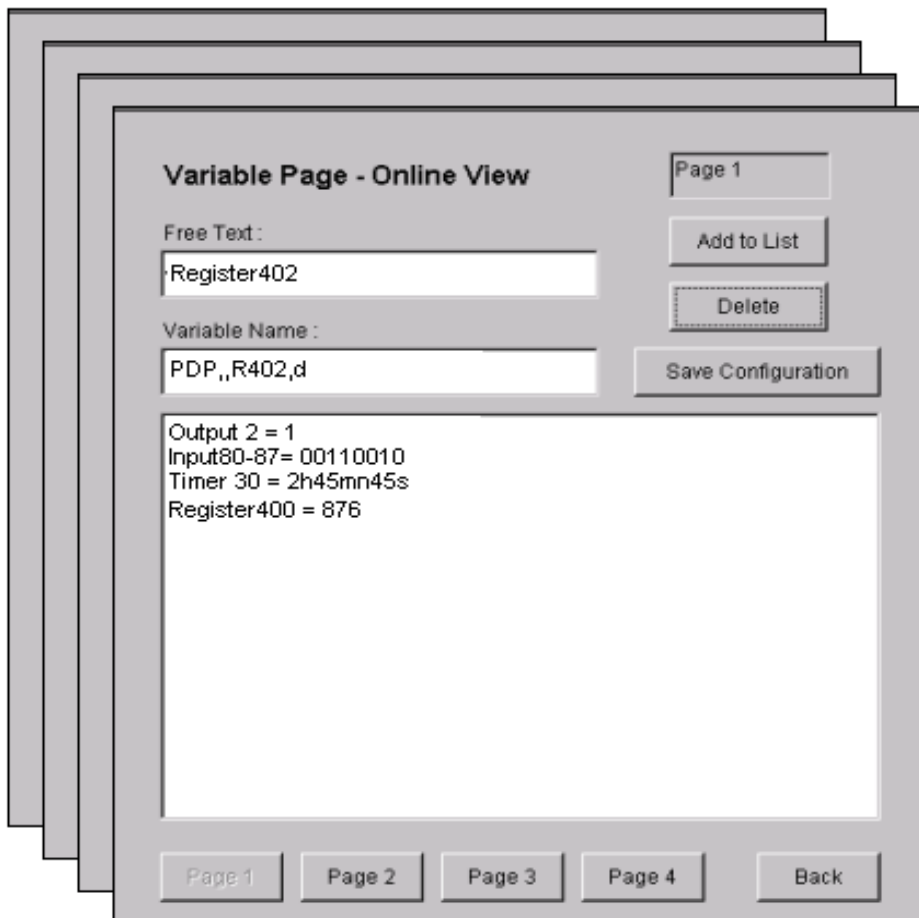
4.3.1 Before use the Variable list

Data Blocks

When Variable list pages are stored, they are stored in Data Blocks of the user program. So in order to be able to use this functionality, you first need to create those Data Blocks, this is done in the Web-Builder Settings, by checking the Box "Variables Lists"



If variable pages are to be stored in the PCD's user memory, a memory range must first be reserved. This takes place in Web-Builder with the VarList Sites option.



Free Text field	Free text about the process data point
Variable Name field	Process data point with PDP key (without % character)
Add to List button	Free text and PDP key are added to the list
Delete button	Selected list entry is deleted
Save Configuration button	Current variable page entries are saved to the PLC (only possible for password level 4 and if the appropriate space has been reserved in PLC memory).
Page1...Page 4	Switch to the other variable pages
Back	Back to start page.

4.4 Description of the PCD media

The entries in the Variable Name field have the following syntax:

PDP,,<Adr>,<Format>

Adr Address of PLC data point. Description below.

Format Format of the PLC data point addressed. Description below.

Address:

Range	Data block num.	Size	Offset	Example
I (Input)		B (Byte)	Ofs.BitNr	I10.3
		W (Word) D (Word)	Ofs	IB20
Q (Output)		B (Byte)	Ofs.BitNr	Q30.4
		W (Word) D (Word)	Ofs	QW15
M (Flag)		B (Byte)	Ofs.BitNr	M40.5
		W (Word) D (Word)	Ofs	MD33
DB (Data block)	Num.	DBX	Ofs.BitNr	DB20.DBX11.6
		DBB (Byte) DBW (Word) DBD (Word)	Ofs	DB21.DBB4
T (Timer)			TimerNum	T45
C (Counter)			CounterNum	C60

Format:

u	Decimal, unsigned	
d	Decimal, signed	
b	Binary	bit, byte, word and dword
x,X	Hexadecimal (lower, upper case)	
o	Octal	
f	Floating point	for DWORD only
s	Character string	Zero terminated string (C convention)
n	S7 character string	STEP7 string. The first address byte contains the maximum length of the string. The second byte contains the actual length. The string data itself is stored from the 3rd byte.
t	Timers	Only in association with timers.
D	Date	Date from a STEP7 DATE_AND_TIME (DT) variable. This format is supported beginning from the version FW 2.300
T	Time	Time from a STEP7 DATE_AND_TIME (DT) variable. This format is supported beginning from the version FW 2.300

Example:

PDP,,M100.0,u

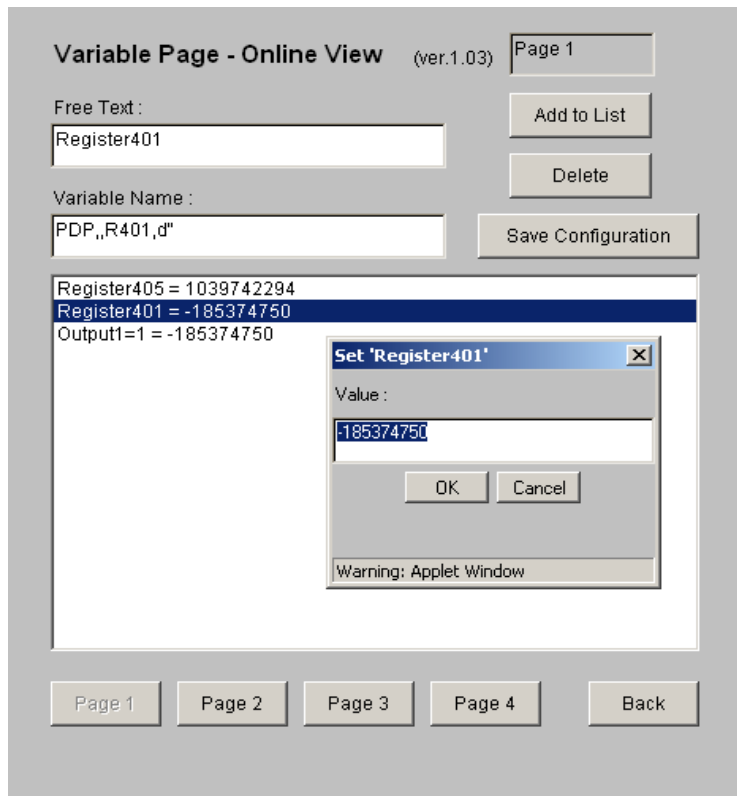
PDP,,DB11.DBB40,n

PDP,,MD33,f

PDP,,QW3,X

4.4.1 Modifying entries in a variable page

Process data points in variable pages can be modified by double-clicking on the specific process data point.



4

4.5 Access to PCD data

If you wish to access PCD process data points from your HTML page, these process data points must be defined with a special process data point key (PDP key) in the HTML page. PDP keys can be defined in HTML pages or in control functions.

A PDP key is as described in the above chapter, but the tag is surrounded now by the character "%". Two characters "%%" at the beginning of the TAG and one "%%" at the end.



A PDP key has the following format:

%%PDP,<NetAdr>,<Adr>,<Format>%

- NetAdr** Reserved, nor currently supported.
- Adr** Address of PLC data point. Description below.
- Format** Format of the PLC data point addressed. Description below.



- In HTML pages, PDP keys must be delimited by “%”.
In control functions, “%” delimiters are not needed.
- If the PDP key has been incorrectly defined, a read operation will result in the display of an error message rather than the data.
For a write operation, the value will not be written to the PLC.
- The PDP key is only evaluated for pages with the suffix *.html or *.htm.
If a PDP key is stored in any other type of file, they are not considered and so no value will be displayed.
- The PDP key is not case sensitive for the media information, but the PDP has to be in capital letters.

Example:

```
%%PDP,,M100.0,u%
%%PDP,,DB11.DBB40,n%
%%PDP,,MD33,f%
%%PDP,,QW3,X%
```

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	Ident	STRING[12]	'SAIA xx7 CDB'	CDB identification
+14.0	ComParaSeriel	STRING[40]	'COM1:PTP_MPI,RS232,19200,8,n,1'	Serial Interface or ...
+56.0	ComParaModem	STRING[40]	'// COM1:PTP_MPI,MODEM'	... Modem connection (now just as comment!)
+98.0	WebIndexDB	STRING[40]	'WEB:INDEXDB=900'	IndexDB -> Identical with WebBuilder!
+140.0	WebRamDisk	STRING[40]	'WEB:RAMDISK=16'	16k RamDisk

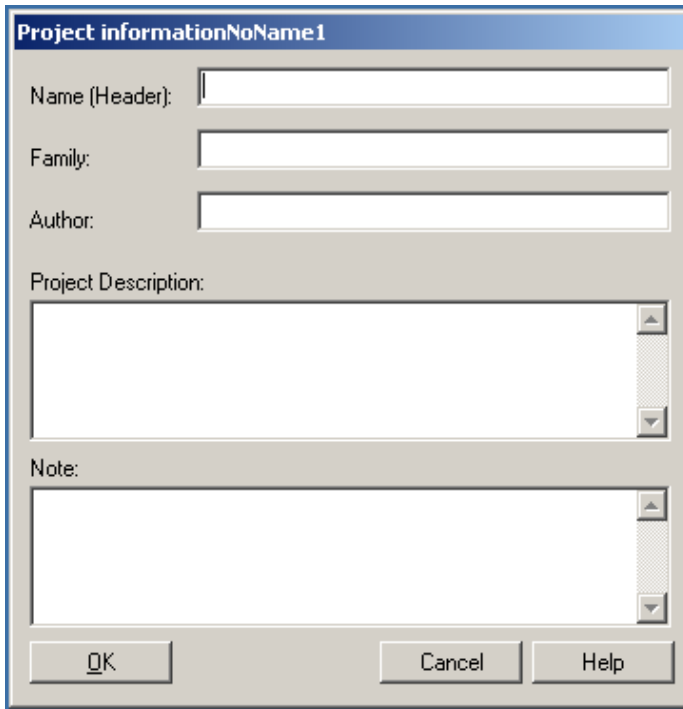
Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	Identificator	STRING[12]	'SAIA xx7 CDB'	CDB Identification
+14.0	Memory	STRING[8]	'MEM7:512'	Memoryrange of PCD
+24.0	Timeout	STRING[24]	'COM1:PTP_MPI,TIMEOUT=2'	Timeout of serial Port
+50.0	WebPara	STRING[30]	'COM1:PTP_MPI,RS232,38400,8,o,1'	Parameter of serial Port
+82.0	WebPassword	STRING[60]	'WEB:PASSWORD=,,,'	Password for Web-Server-> void= no password
+144.0	IndexDB	STRING[15]	'WEB:INDEXDB=900'	Index DB -> identical with Web-Builder
=162.0		END_STRUCT		

4.6 Configuration of the Web-Server with the Saia xx7 I/O-Builder

The most settings described in chapter “4.7 Configuration and definition of the Web-Server in the user program” can be done in a more comfortable way with the Saia xx7 I/O-Builder. Select this tool with “**Programs**”, “**Saia-Burgess**”, “**xx7**”, “**I/O-Builder**”:

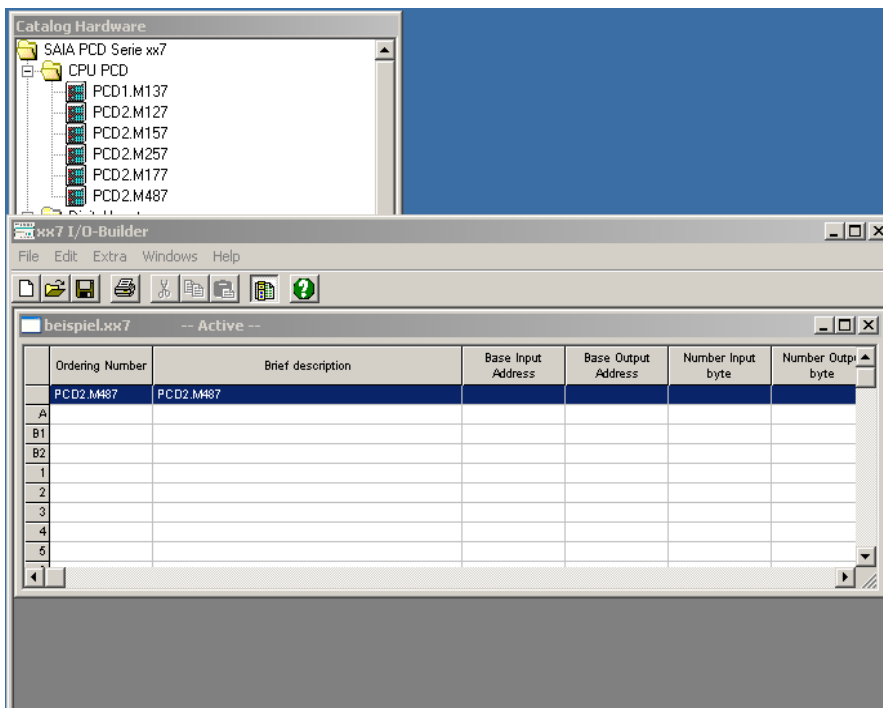


Set up a new project



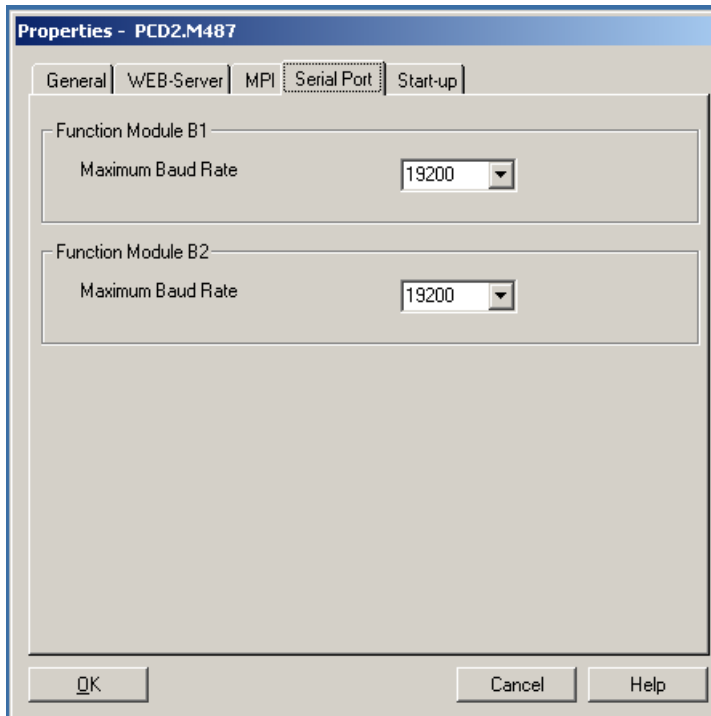
4

Select a PCD out of the Hardware catalogue.



4.6.1 Definition of serial port

Double click on the chosen PCD, that opens the following window. Select the tab "Serial port".

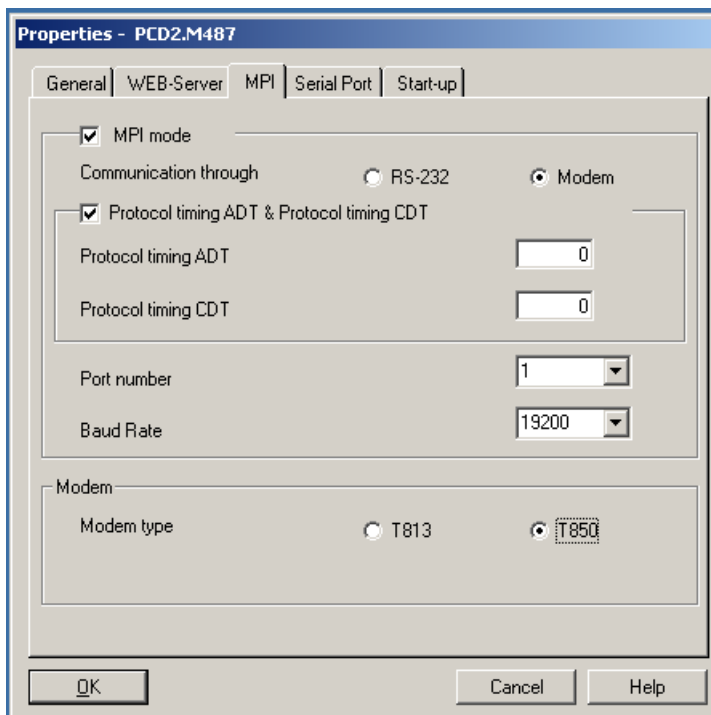


4

If there are further settings necessary, it's possible to add them later in the CDB of the user program.

4.6.2 Modem

Select the tab "MPI" and enable MPI Modus.



Select a modem:

For Analogue: T813
For ISDN: T850

Timeout for hang up:

Default value is 30 minutes

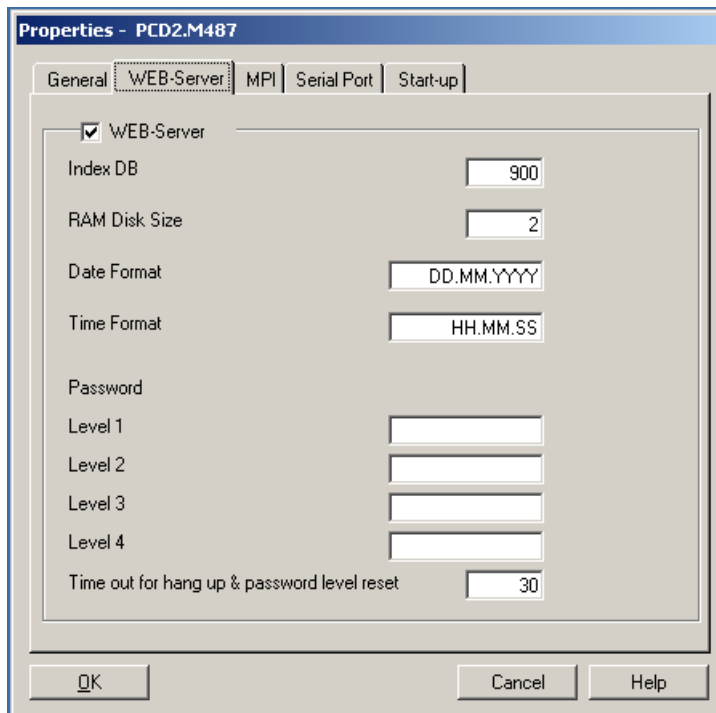
Protocol times:

ADT 2000 ms by direct connection
10000 ms with modem connection

CDT 220 ms by direct connection
1000 ms with modem connection

4.6.3 Definition of the Web-Server

Select the tab WEB-Server and enable WEB-Server.



4

All the settings necessary for the definition of the Web-Servers can be done here:

- Set up the Index DB
- Size of the RAM Disk in kB
- Format of the date
- Format of the time
- Passwords and levels
- Timeout for hang up and password level reset in minutes (default are 30 minutes)

Possible formats, rights etc. are described in chapter 4.7.3.

4.7 Configuration and definition of the Web-Server in the user program

4.7.1 Definition of serial port

Configuration

The serial port is configured by writing the appropriate parameters to a CDB (Configuration Data Block).

Communication parameters

If there is no CDB, or if the relevant entries are not present in it, by default port 1 is initialized as the MPI port with 19'200 Baud, 8 data bits, odd parity and 1 stop bit. This port can then be used either for a web server application or for communication with Step®7 programming software.

The following parameters can be used to change the port or its properties:

COM<n>:PTP_MPI,RS232,<baud>,<data>,<parity>,<stop>

n Port 1 or 3

baud 110...38'400 Baud rates. n.b.: Port 2 does not support 38'400 Baud

data Data bits (7 or 8)

parity Possible values are:

- E Even
- O Odd
- N None
- L Force Low
- H Force High

stop Stop bits (1 or 2)



To activate settings, the supply voltage must be switched off and on again!

Example:

COM1: PTP_MPI,RS232,9600,8,N,1

COM2: PTP_MPI,RS232,19200,8,E,2

COM1: PTP_MPI,RS232,38400,8,O,1

4.7.2 Modems

A SBC modem can be connected to the PCD's port 1.

To allow FW to initialize the modem correctly, the following command must be defined in the CDB:

Analogue Modem T813

COM1: PTP_MPI,T813

→ From FW 2.0.206

COM1: PTP_MPI,MODEM

→ only for backwards compatibility with FW V 2.0.200

ISDN Modem T850

COM1: PTP_MPI,T850[,<prot>[,<msn>]]

→ From FW 2.0.206

prot Protocol. Valid strings are:

V.110	X.75-NL
HDLC_ASYNC	V.120
HDLC_TRANSPARENT	X.31B
BYTE_TRANSPARENT	X.31D



- If there is no or an invalid string for the protocol, **X.75-NL** is the default protocol.
- msn Multiple subscriber number
- Up to 22 digits (0...9; *; #)
 - If no or invalid msn is specified, msn is set to * → answers always

Example:

COM1: PTP_MPI,T850,V.110,21

COM1: PTP_MPI,T850,V.110



- To activate settings, the supply voltage must be switched off and on again!
- To not interrupt modem communications.
Check the status of the DCD signal before breaking off a modem connection.
- The modem must be configured to reset at DTR=0. (AT&D3)

4

Switching a configured port between FW and user program

A port configured with the command “PTP_MPI...” (with or without modem) can if required also be used by the user program, for example to transmit text from the user program to the port. This is done by calling SFC 200.

FW has control of port Call SFC 200, with VKE = 0
Wait until VKE is at 1 after the call.

User program has control of port Call SFC 200, with VKE = 1
Wait until VKE is at 1 after the call.

Timeout

The exchange of data on the port can be coupled with a timeout. This involves re-starting the defined timeout for every telegram received or transmitted.
If no exchange of data takes place during the timeout, the password and any modem connection are reset after the timeout has elapsed.
The default setting for the timeout is 30 minutes.

COM<n>:PTP_MPI,TIMEOUT=<timeout>

n Port 1 or 3
timeout Timeout in minutes.

N.B.: Setting a value of 0 switches off the timeout function.



To activate settings, the supply voltage must be switched off and on again, or the CPU must be switched from Stop to Run!

Example:

COM1: PTP_MPI,TIMEOUT=5

COM2: PTP_MPI,TIMEOUT=0

Protocol times

If required, you may define your own character delay time (CDT; default = 220ms for direct connection, 1000 ms for modem connection) or answer delay time (ADT; default = 2000ms for direct connection, 10000 ms for modem connection) for the point to point communication between the PCD and the PC.

By adjusting these two times, connections can be realized across great distances or via modem. It is important to set the same values for both sides (PC and PCD).

COM<n>: PTP_MPI,CDT=<cdt>

COM<n>: PTP_MPI,ADT=<adt>

n: Port 1 or 3
cdt, adt Timeouts in ms



To activate settings, the supply voltage must be switched off and on again!

Example:

COM1: PTP_MPI,CDT=300

COM2: PTP_MPI,ADT=2500

4.7.3 Definition of Web-Servers

Configuration

Like the communications parameters for the serial port, the web server is configured in the CDB.

4

IndexDB

To allow access to your own HTML pages, you must define the IndexDB in the CDB. The IndexDB is the first DB generated by the “Web-Builder” conversion tool.

WEB: INDEXDB = <num>

Num Number of index data block



- To activate settings, the supply voltage must be switched off and on again, or the CPU must be switched from Stop to Run!
- If a file is loaded from the web browser, the following order of search applies:
 1. DBs (files stored in DBs)
 2. Firmware (files stored in FW)
 3. PC (files stored on the PC's hard disk)
- At offset 2 (DBW2) of the IndexDB the current password level of the web server can be read.
- At offset 4 (DBW4) of the IndexDB a web server life-sign can be read. This word is incremented every time the web server is queried.

Example:

WEB: INDEXDB=900

WEB: INDEXDB=256

Password

Up to 4 pass words can be defined.

Each password corresponds to a password level. This means you can log onto the web server at four different password levels and execute the relevant functions.

In any specific password level you will have all rights other than those in higher levels.

Level 1	Read only HTML pages
Level 2	Read PLC data points
Level 3	Write PLC data points
Level 4	Modify / save variable list

By default, no password is defined and password level 4 is active.

Passwords are defined as follows.

WEB: PASSWORD=<pw1>,<pw2>,<pw3>,<pw4>

pw1... Password for every level. It is not necessary to define all 4 passwords
pw4



- To activate settings, the supply voltage must be switched off and on again, or the CPU must be switched from Stop to Run!
- Upper case/lower case not supported for passwords. SAIA and Saia are not equivalent.
- All 4 passwords do not have to be defined.

For example, if you only wish to define the password for password level 4 you can define it as follows:

WEB: PASSWORD=,,, <pw4>

Example:

WEB: PASSWORD=saia1,saia2,saia3,saia4

WEB: PASSWORD=, , ,saia4

WEB: PASSWORD=, ,saia3,saia4

4

RAM Disk

The web server has an internal RAM Disk of 2kB in FW which can be stored for the temporary storage of data.

If necessary, the RAM disk can be increased in size as follows:

WEB: RAMDISK=<size>

Size size of RAM Disk in kB



- To activate settings, the supply voltage must be switched off and on again!
- The RAM disk's memory range is deducted from user memory. Make sure that sufficient space still remains free in user memory, if you wish to increase the size of the RAM disk.

Example:

WEB:RAMDISK=4

WEB:RAMDISK=16

Date Format from DATA_AND_TIME variable

The web server displays the data from Step[®]7 DATE_AND_TIME variable by default in the following format: DD.MM.JJJJ (e.g: 10.09.2001).

It's possible to change this format via CDB:

WEB: DATEFORMAT=<F1><T1><F2><T2><F3>

F1,F2,F3	D	display day without leading zero
	DD	display day with leading zero
	M	display month without leading zero
	MM	display month with leading zero
	YY	display year with two digits
	YYYY	display year with four digits

T1,T2 T1,T2 Valid separator.

A valid separator has to be inside of one of the following ranges:

Decimal values of ASCII characters between 33 and 47 or 58 and 64.



- A change takes affect with the next stop to run
- If there is an error, the default format is used
- This instruction needs at least the FW 2.300

Example:

WEB: DATEFORMAT=D/M/YY

WEB: DATEFORMAT=YYYY.MM.DD

Time Format from DATA_AND_TIME variable

The Webserver displays the time from STEP7 DATE_AND_TIME variable by default in the following format: HH:MM:SS (e.g: 11:55:00).

It's possible to change this format via CDB:

WEB: TIMEFORMAT=<F1><T1><F2><T2><F3>

F1,F2,F3	H	display hour without leading zero
	HH	display hour with leading zero
	M	display minutes without leading zero
	MM	display minutes with leading zero
	S	display seconds without leading zero
	SS	display seconds with leading zero
T1,T2	T1,T2	Valid separator.

A valid separator has to be inside of one of the following ranges:

Decimal values of ASCII characters between 33 and 47 or 58 and 64.

4

- A change takes effect with the next stop to run
- If there is an error, the default format is used
- This instruction needs at least the FW 2.300

Example:

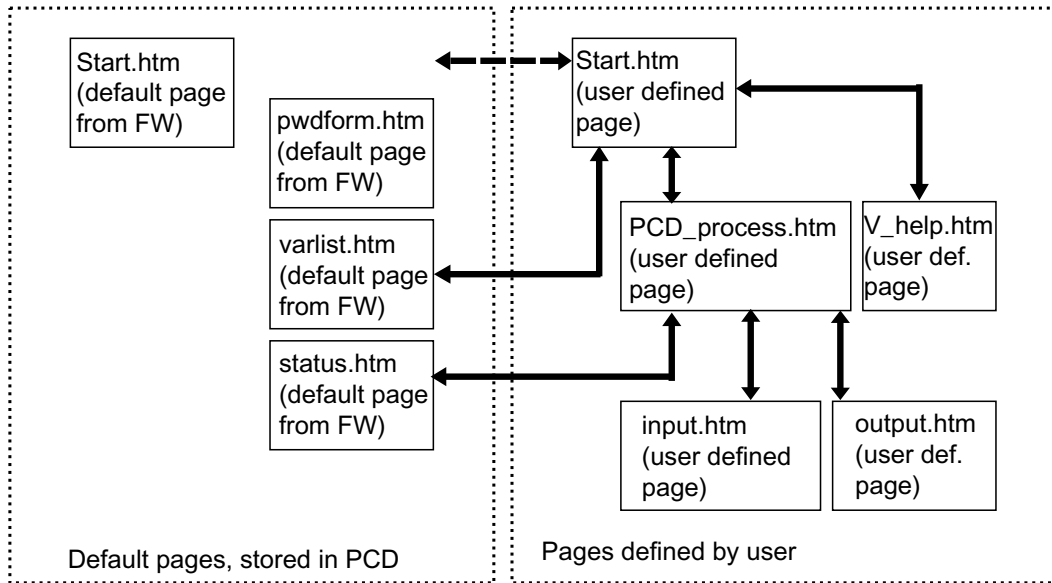
WEB: TIMEFORMAT=H/M/S

WEB: TIMEFORMAT=HH.MM.SS

5 Example Web-Project

5.1 Structure of pages

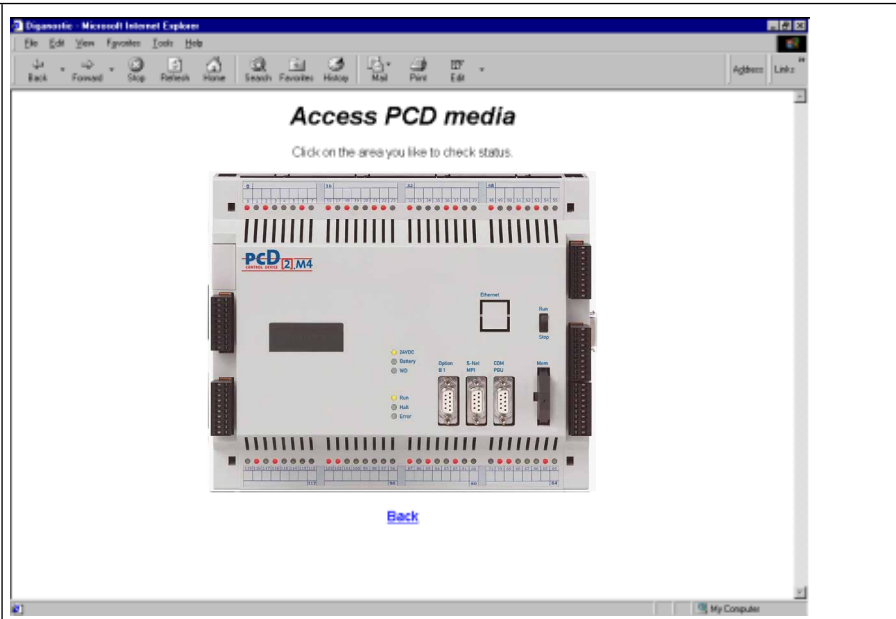
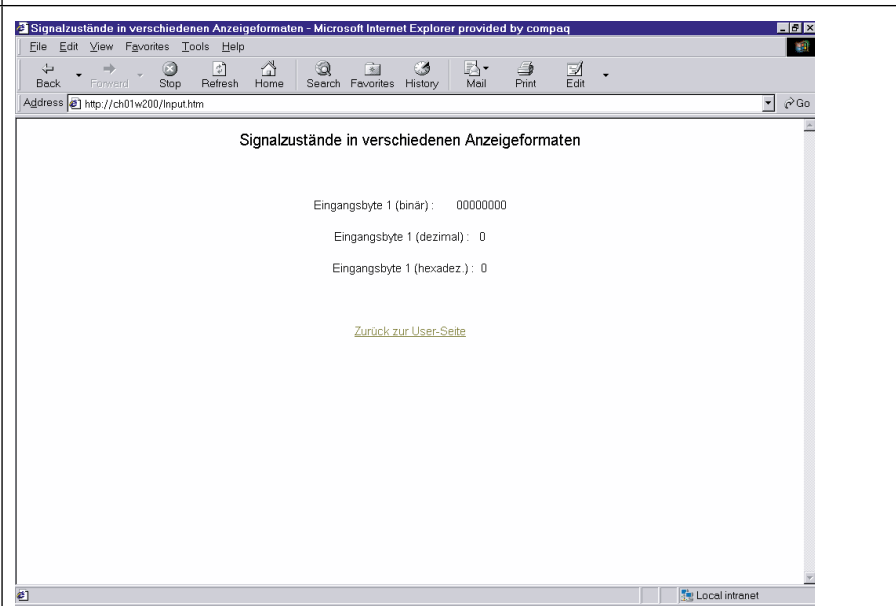
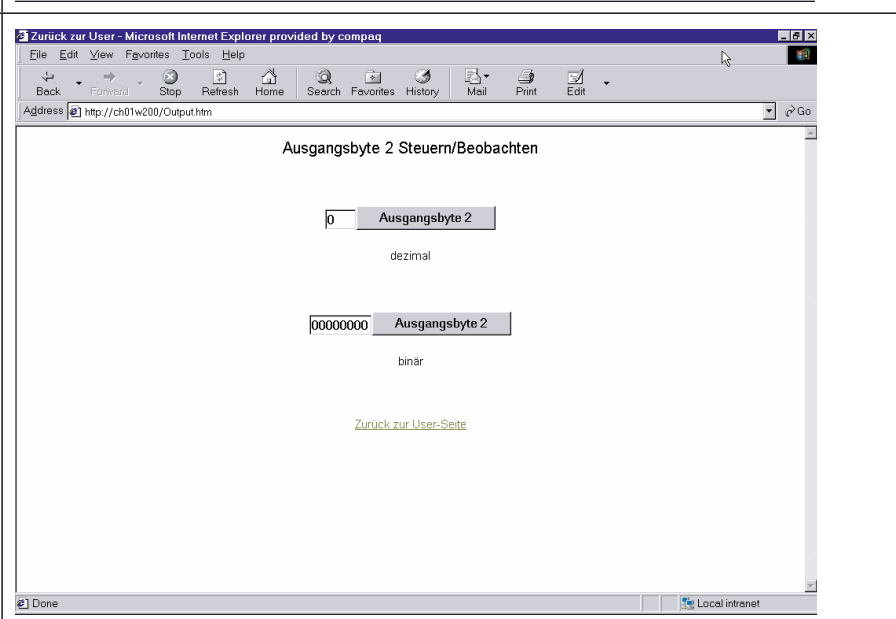
In the following example four HTML pages are to be defined and downloaded to the PLC. The structure is as follows:



We will not show here, how to create all the HTML pages, but only the interesting ones. We mean by interesting, the ones which have PCD media in it. So we will describe in detail how to do the input.html and the output.html pages. All the pages of this project can be found in the PG5 project “WEB_doc_project”.

The four pages defined in the user program appear as following in the Web-Browser:

<p>Page: start.htm</p> <p>Start page, with a picture and links to the pages: varlist.htm, pcd_process.htm and V_help.htm</p>	
--	--

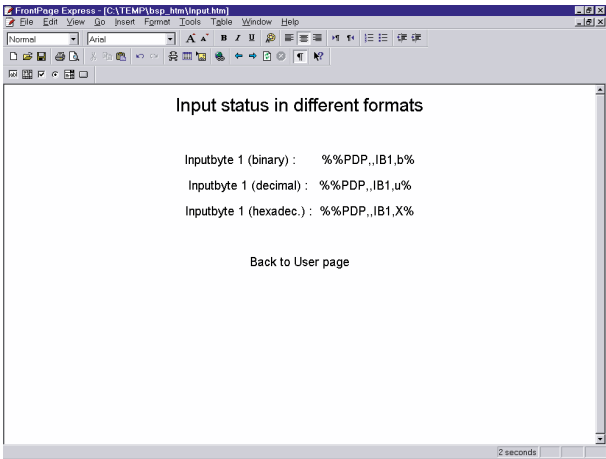
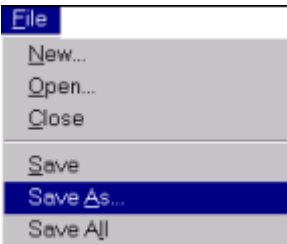
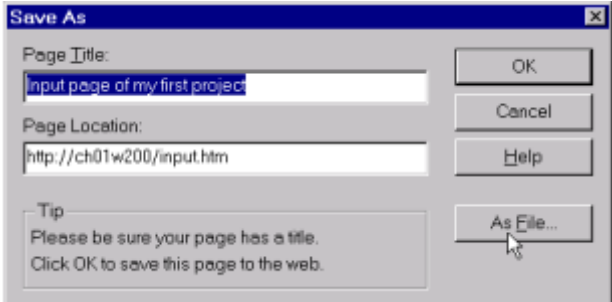
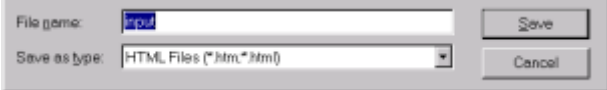
<p>Page: pcd_process.htm is a visual selection page, which allows access to the pages: input.htm output.htm status.htm</p>	
<p>Page: input.html Displays the actual status of the inputs 0 to 7</p>	
<p>Page: output.htm Display and modification of outputs 0 to 7</p>	

5.2 PCD data point into the HTML Pages

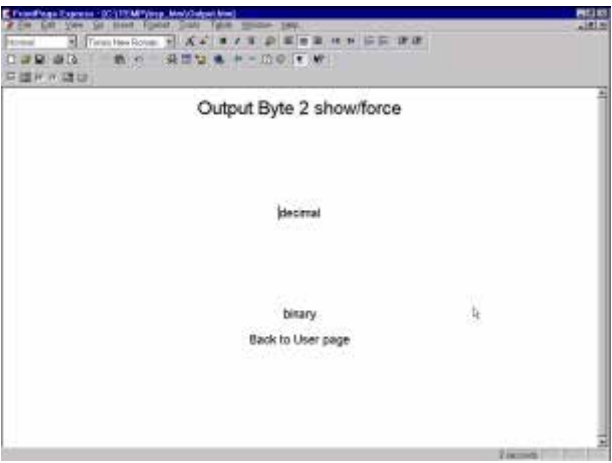
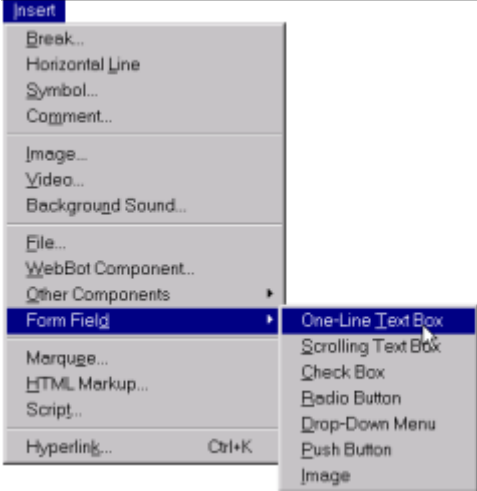
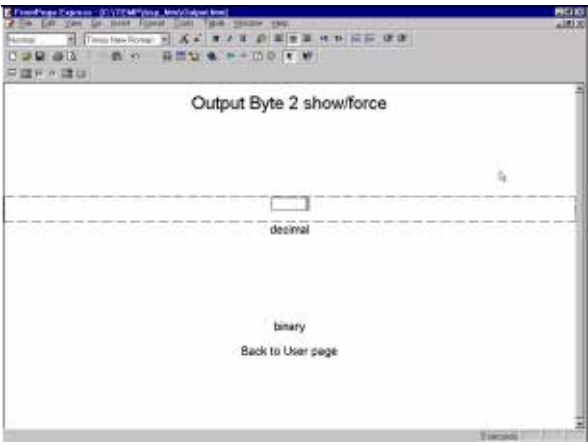
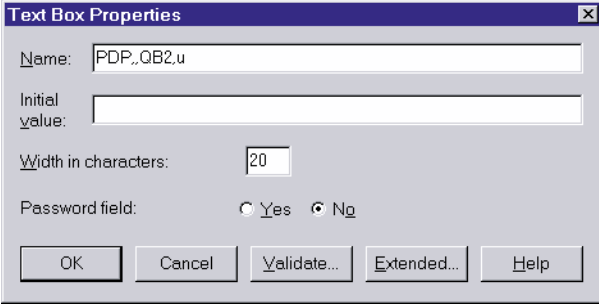
We will show here how to introduce in your html page a dynamic area, where value of the PCD media can be displayed. We will show it with the FrontPage express editor, but of course you can use any HTML editor.

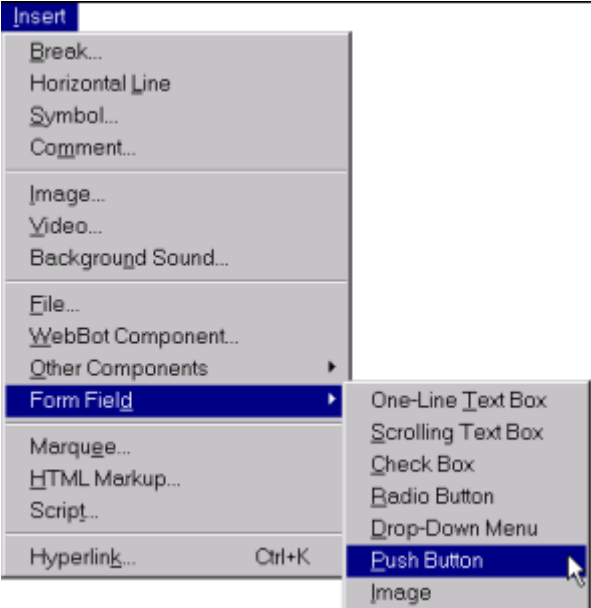
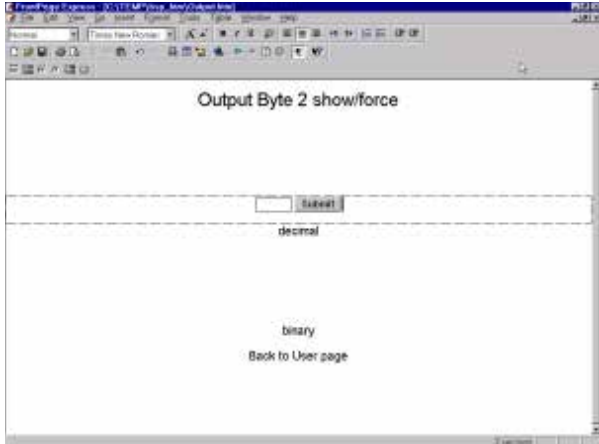
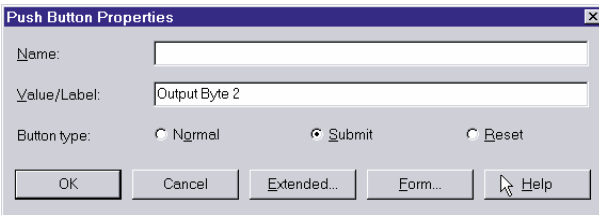
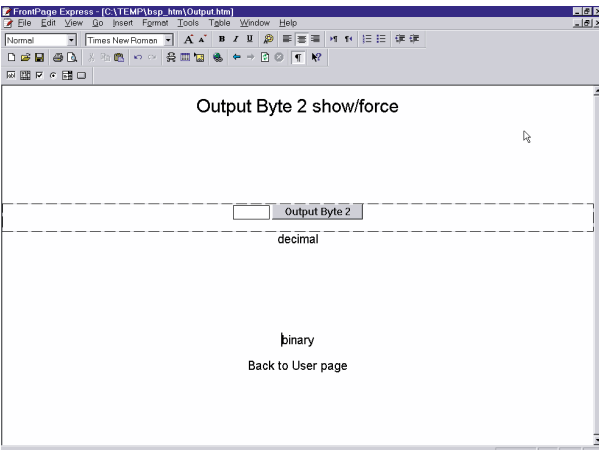
5.2.1 Page: input.htm

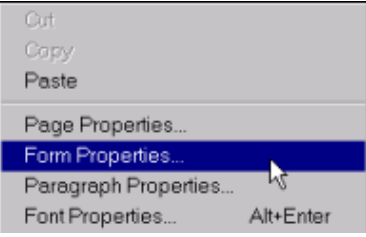
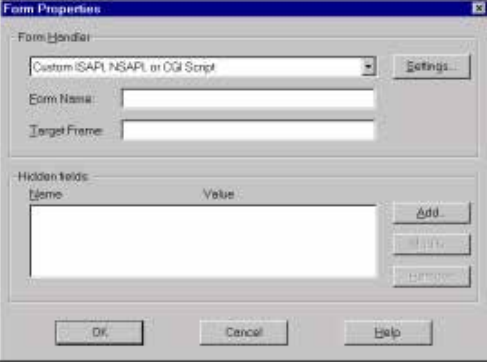
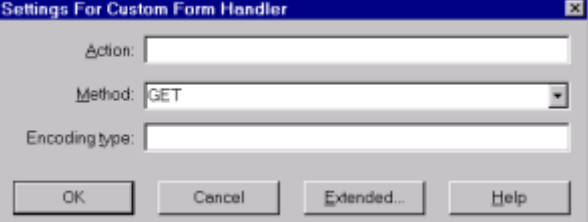
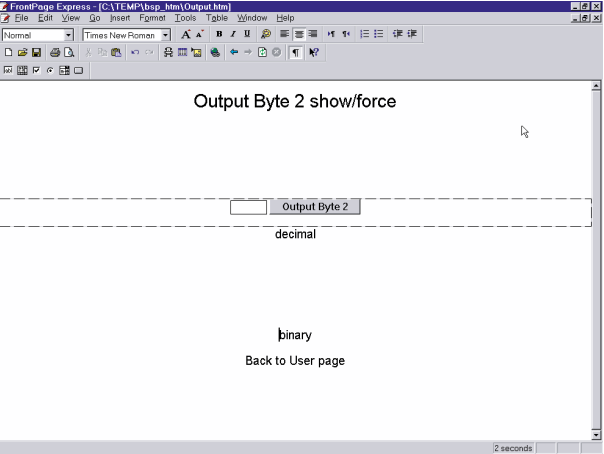
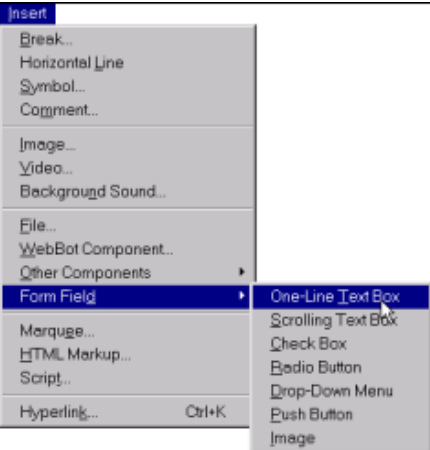
In this page we demonstrate how easy it is to put some fields in a page to show the value of a media, at the time the HTML page is loaded from the PCD. The example shows inputs, but any area can be displayed like this, see the chapter "Variable pages", to check the right syntax.

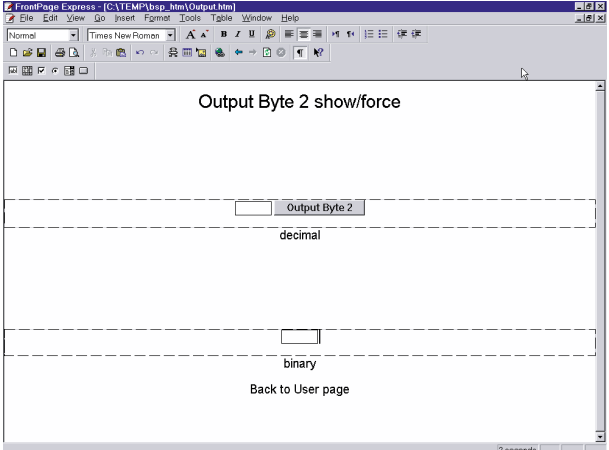
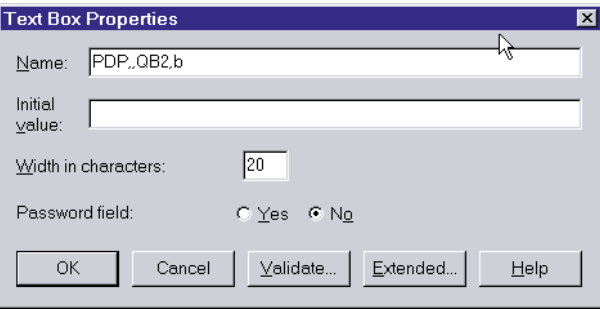
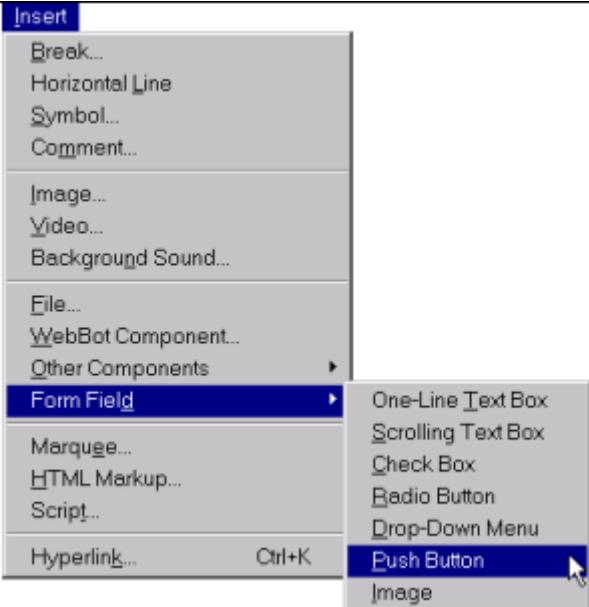
<p>1 Open a new, empty page. Insert text and position text.</p> <p>Note syntax for PDP key: %%PDP,,IB1,b% %%PDP,,IB1,u% %%PDP,,IB1,X%</p>	
<p>2 Save file with the Save as... option.</p>	
<p>3 Enter page title correctly.</p> <p>Select the As File option</p>	
<p>4 Enter file name: input. File should not be closed.</p>	

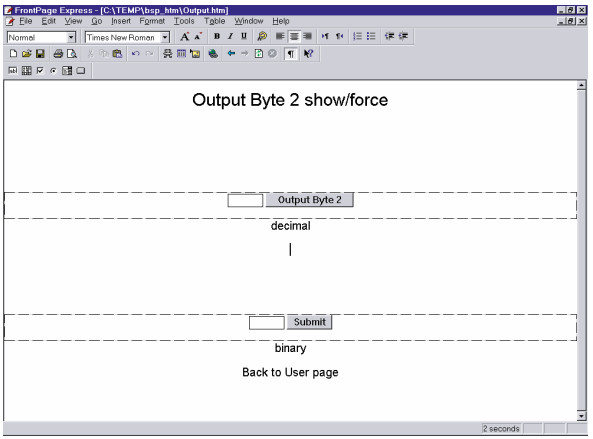
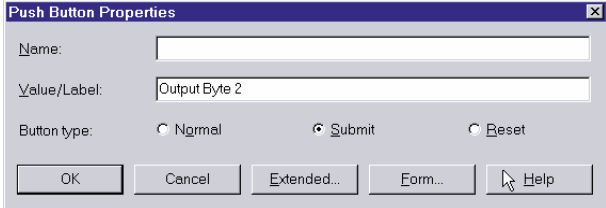
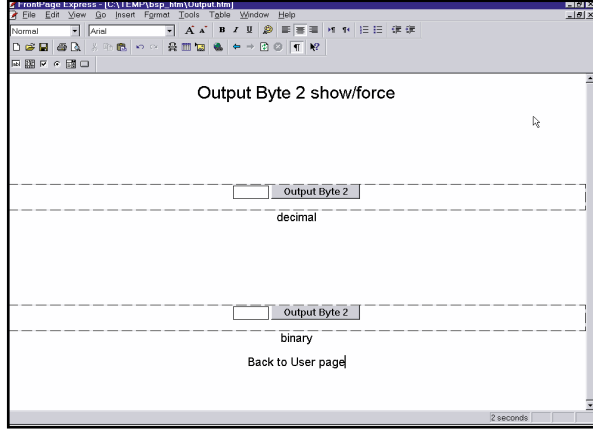
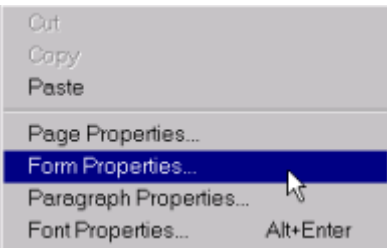
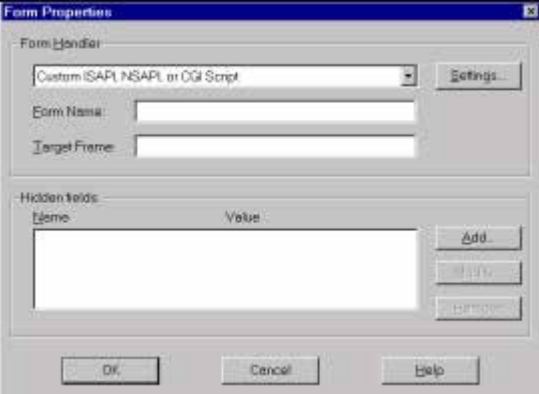
5.2.2 Page: output.htm

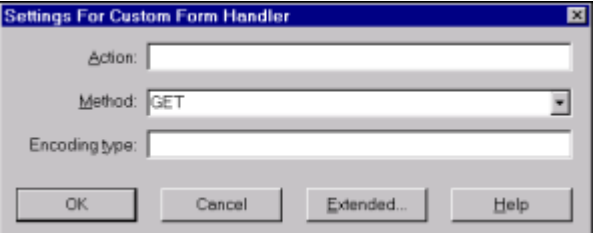
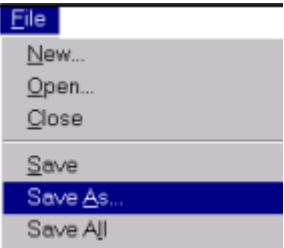
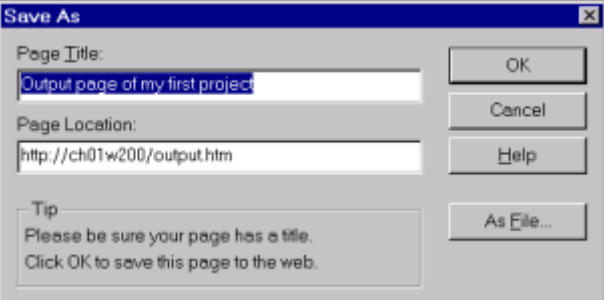

<p>1 Open a new, empty page. Insert text and position text.</p>	
<p>2 Insert input/output fields: Position cursor over the text decimal. Insert field with: Insert, FormField, One-Line Text Box</p>	
<p>3 Position the field</p>	
<p>4 Double-click on the field. Enter the PDP key Note the correct syntax: PDP,,QB2,u</p>	

<p>5</p>	<p>Insert the Submit Button. Position cursor near entry field in the same frame (Form Field). Insert, Form Field, Push Button</p>	
<p>6</p>	<p>Double-click on the button</p>	
<p>7</p>	<p>Insert text which will appear on the button</p>	
<p>8</p>	<p>The page now appears as follows</p>	

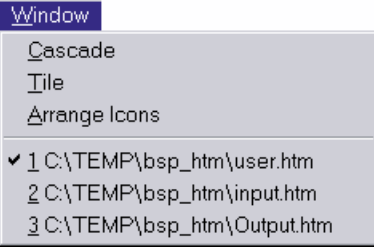
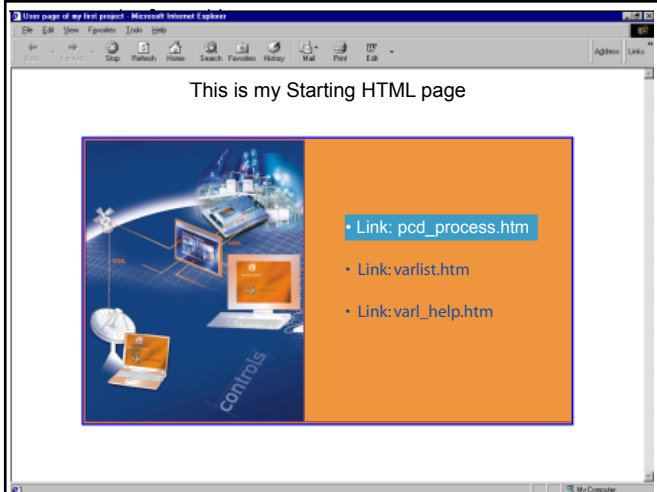
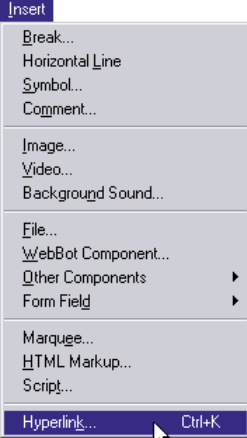
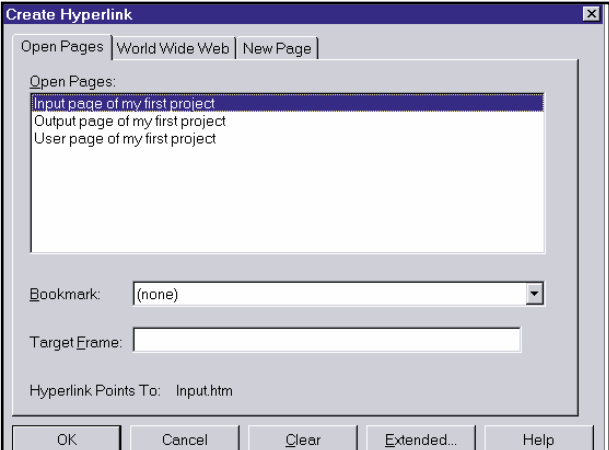
<p>9</p>	<p>Position cursor to right of the button and click the right-hand mouse button. Select the Form Properties option</p>	
<p>10</p>	<p>Select the Settings option</p>	
<p>11</p>	<p>Method: GET Select Close window</p>	
<p>12</p>	<p>The page appears as follows: The same procedure as set out in steps 2 to 11 should now follow for the binary value entry field.</p>	
<p>13</p>	<p>Insert input/output fields: Position cursor over the text decimal. Insert field with: Insert, FormField, One-Line Text Box</p>	

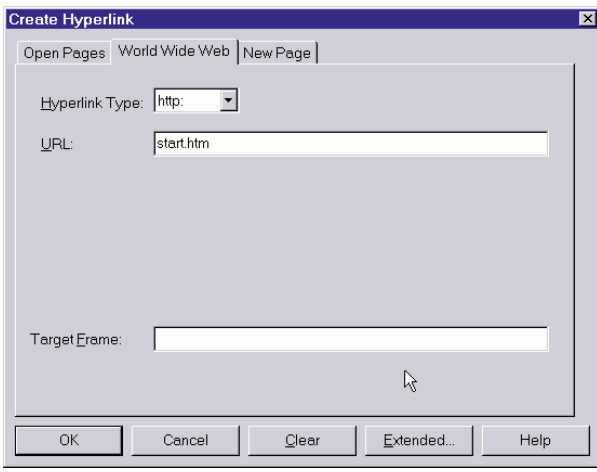
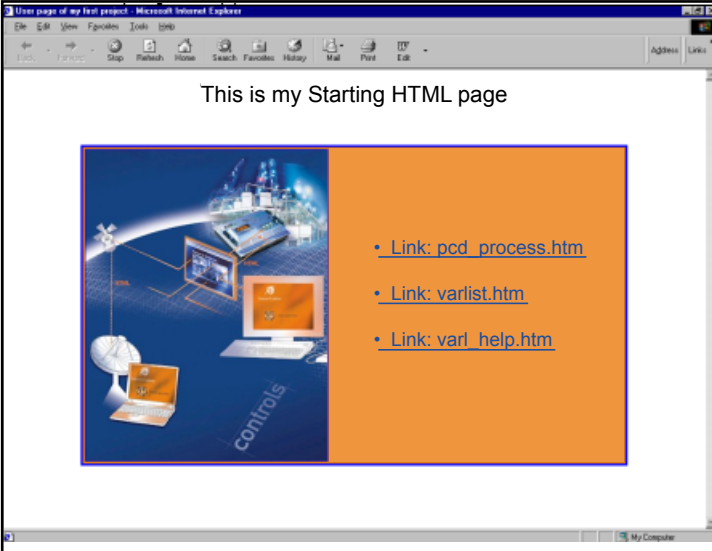
<p>14</p>	<p>Position field</p>	
<p>15</p>	<p>Double-click on the field. Enter the PDP key. Note correct syntax: PDP,,QB2,b</p>	
<p>16</p>	<p>Insert Submit Button. Position cursor near the entry field in the same frame (Form Field). Insert, Form Field, Push Button</p>	

<p>17</p>	<p>Double-click on button</p>	
<p>18</p>	<p>Insert text to appear on button</p>	
<p>19</p>	<p>The page now appears as follows</p>	
<p>20</p>	<p>Position cursor to right of the button and click right-hand mouse button. Select the Form Properties option.</p>	
<p>21</p>	<p>Select the Settings option.</p>	


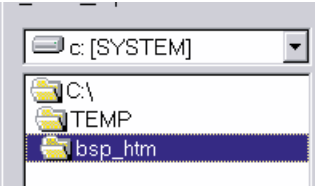


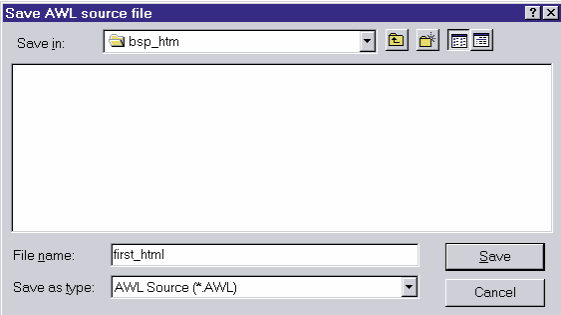
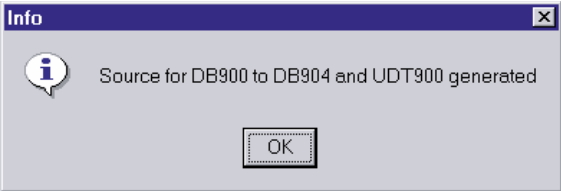
22	Select Method: GET Close window	
23	Save file with the Save As...option.	
24	Enter page title correctly. Select the As File option	
25	Enter file name: output . Then save.	

5.3 Link pages



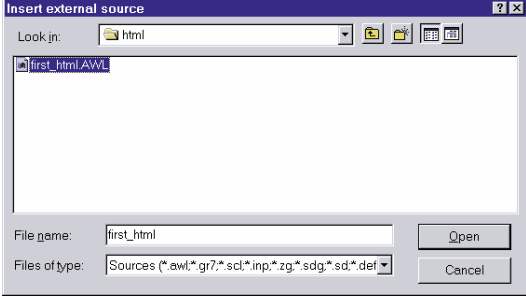
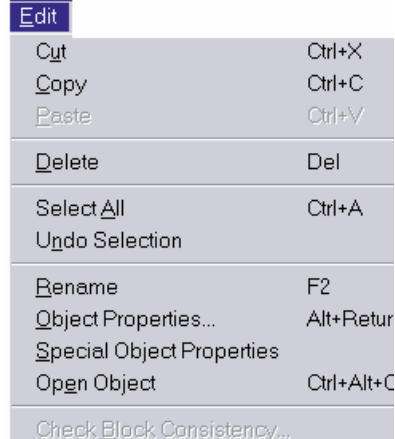
<p>1</p>	<p>All HTML pages to be inked must be open. Load the user.htm page into Frontpage Express using the “Window” option.</p>	
<p>2</p>	<p>Mark text to be linked</p>	
<p>3</p>	<p>Create hyperlink with: Insert, Hyperlink</p>	
<p>4</p>	<p>Select page to be linked from folder Open Pages The links to the other pages are similar to the input page</p>	

<p>5</p> <p>The link to the start page is made in the Word Wide Web folder. The text: start.htm must be entered in the URL field</p>	
<p>6</p> <p>The finished page looks like this</p>	

5.4 Generation of DBs

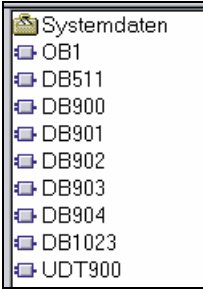

1	Start the Web-Builder									
2	Select path in which files or images previously developed with FrontPage Express have been stored									
3	Select files user.htm, input.htm, output.htm etc.	<table border="1" data-bbox="703 568 1193 696"> <thead> <tr> <th>FileName</th> <th>FilePath</th> </tr> </thead> <tbody> <tr> <td>Input.htm</td> <td>C:\TEMP\bsp_htm</td> </tr> <tr> <td>Output.htm</td> <td>C:\TEMP\bsp_htm</td> </tr> <tr> <td>user.htm</td> <td>C:\TEMP\bsp_htm</td> </tr> </tbody> </table>	FileName	FilePath	Input.htm	C:\TEMP\bsp_htm	Output.htm	C:\TEMP\bsp_htm	user.htm	C:\TEMP\bsp_htm
FileName	FilePath									
Input.htm	C:\TEMP\bsp_htm									
Output.htm	C:\TEMP\bsp_htm									
user.htm	C:\TEMP\bsp_htm									
4	Enter the IndexDB									
5	Convert files and images									
6	Save the DBs in a file called first_html									
7	Confirm generation									

5.5 Insertion of HTML pages in the user program

1	Start Siemens® Simatic® Manager from STEP®7.	
2	Insert the external sources under menu item Insert, External Source	
3	Select the *.il file with which the Web-Builder was generated	
4	Translate the newly inserted source file with menu items: Edit, Compile	

5

Insertion of HTML pages in the user program

5	<p>When translation is complete, DB's stored in the source file have been set out.</p> <p>In the example, these are: DB's 900 to 904 and UDT900</p>																															
6	<p>Definition of web server in CDB. In the example on the left, this is CD 1023. It must at least contain a definition of the IndexDB. If no other port is defined, port 1 is defined as the default with 19,200 Baud, 8 data bits, odd parity and 1 stop bit</p>																															
<table border="1" style="width: 100%; border-collapse: collapse; text-align: left;"> <thead> <tr> <th style="width: 15%;">Address</th> <th style="width: 20%;">Name</th> <th style="width: 15%;">Type</th> <th style="width: 25%;">Initial value</th> <th style="width: 25%;">Comment</th> </tr> </thead> <tbody> <tr> <td>0.0</td> <td></td> <td>STRUCT</td> <td></td> <td></td> </tr> <tr> <td>+0.0</td> <td>Identificator</td> <td>STRING[12]</td> <td>'SAIA xx7 CDB'</td> <td>CDB Identification</td> </tr> <tr> <td>+14.0</td> <td>WebPara</td> <td>STRING[30]</td> <td>'COM1:PTP_MPI,RS232,38400,8,o,1'</td> <td>Parameter of serial Port</td> </tr> <tr> <td>+46.0</td> <td>IndexDB</td> <td>STRING[15]</td> <td>'WEB:INDEXDB=900'</td> <td>Index DB -> identical with Web-1</td> </tr> <tr> <td>=64.0</td> <td></td> <td>END_STRUCT</td> <td></td> <td></td> </tr> </tbody> </table>			Address	Name	Type	Initial value	Comment	0.0		STRUCT			+0.0	Identificator	STRING[12]	'SAIA xx7 CDB'	CDB Identification	+14.0	WebPara	STRING[30]	'COM1:PTP_MPI,RS232,38400,8,o,1'	Parameter of serial Port	+46.0	IndexDB	STRING[15]	'WEB:INDEXDB=900'	Index DB -> identical with Web-1	=64.0		END_STRUCT		
Address	Name	Type	Initial value	Comment																												
0.0		STRUCT																														
+0.0	Identificator	STRING[12]	'SAIA xx7 CDB'	CDB Identification																												
+14.0	WebPara	STRING[30]	'COM1:PTP_MPI,RS232,38400,8,o,1'	Parameter of serial Port																												
+46.0	IndexDB	STRING[15]	'WEB:INDEXDB=900'	Index DB -> identical with Web-1																												
=64.0		END_STRUCT																														
7	<p>Load the user program.</p> <p>Note.</p> <p>Do of the fact that on this example I/O's are used, an I/O configuration has to be defined before.</p> <p>If the CDB is being loaded for the first time, or if entries in the CDB have been changed, the controller's supply voltage must be switched off and on again to activate CDB entries.</p>																															

6 Troubleshooting

6.1 Variable List (varlist)

Q: How to access the varlist function ?

A: The URL to access the varlist function is the following : http://pc_name/station_name/varlist.htm

Q: The varlist is not running, nothing happens !

A: Check that the Java settings of the browser are correct and that the browser version is supported for our product. See chapter: **3.3.3 The PC's cache memory**.

A Appendix A

A.1 How to create html pages without editor

This description is intended for people who wish to write html pages without editor, have more dynamic html pages with automatic update of values or manipulating the media PCD with Java applets.

Of course a good knowledge of HTML, JavaScript and Java programming is expected to understand the following chapters.

All examples have been made and tested with the Browser IE 6.0

A.2 CGI (Common Gateway Interface) of the Web-Server

In the Web-Server (inside the PCD), there are already 4 CGIs (executable) implemented:

- readval.exe
- writeval.exe
- ordervalues.exe
- readfile.exe

All these CGIs are intended to manipulate values of the PCD through the tags.

1) All these CGIs, have the restraints of the Web-Server access level, in fact if the Web-Server is in access level 2, it is only possible to read the values and not to write them, etc...

2) All this functions are based on the GET method of the HTTP protocol.



A.2.1 Readval.exe

Read value is used to read a single PCD media.

The syntax of it, is the following:

`http://pc_name/station_name/cgi-bin/readVal.exe?TAG`

pc_name: the PC where the Web-Connect is running.

Station_name: The name assigned to a PCD, corresponding to his communication settings.

/cgi-bin/readVal.exe: the CGI

? : separation between the CGI and the parameters.

TAG : media tag, which has to be read.

The return answer will contain the value of the specified media.

Example

This is the request to read the MW 100, the value should be in decimal format:

`http://pc_name/station_name/cgi-bin/readVal.exe?PDP,,MW100,u`

The answer to that in the HTTP protocol will be:

```
HTTP/1.0 200 OK[0x0A]Content-Type:text/plain[0x0A]Content-
Length:4[0x0A][0x0A]3400
```

The value is: 3400

Access level of PCD

It is also possible to read in which level actually the PCD is, this in the purpose to limit some functionality when you are writing your own applet.

`http://pc_name/station_name/cgi-bin/readVal.exe?PasswordLevel`

This will return the actual access level of the Web-Server (between 0 to 4).

A.2.2 Writeval.exe

Write value is used to write a single PCD media.

The syntax of it, is the following:

```
http://pc_name/station_name/cgi-bin/writeval.exe?TAG+xxx
```

pc_name: the PC where the Web-Connect is running.
Station_name: The name assigned to a PCD, corresponding to his communication settings.
/cgi-bin/writeval.exe: the CGI
?: separation between the CGI and the parameters.
TAG : media tag that has to be written.
+: separation between parameters
xxx: the value to be written, the format of the value was specified in the tag itself !

The answer to the request is the value you had sent.

```
HTTP/1.0 200 OK[0x0A]Content-Type: text/plain[0x0A]Content-Length: 3[0x0A][0x0A]xxx
```

xxx: is the value sent

[0x0A]: it is a hexadecimal value that can't be represented as an ASCII character.

Example

This is the request to write the MW 100 with the value 2300 and the value is in decimal format.

```
http://pc_name/station_name/cgi-bin/writeVal.exe?PDP,,MW100,u+2300
```

The answer to that in the HTTP protocol will be:

```
HTTP/1.0 200 OK[0x0A]Content-Type:text/plain[0x0A]Content-Length:4[0x0A][0x0A]2300
```

So the value written is: 2300

A.2.3 Ordervalues.exe

Order values is used when access to more than one PCD media is needed in the same time. In fact the functionality allows to define a list of media to be read, but this will just define the list of media and not read it. To read it, you have to use the "read-file.exe", see next chapter.

The syntax is as follows:

```
http://pc_name/station_name/cgi-bin/OrderValues.exe?listname+dummy+ssss+TAG1+TAG2+TAG3+TAG4+TAG5+TAG6+TAG7
```

pc_name: the PC where the Web-Connect is running.
Station_name: The name assigned to a PCD, corresponding to his communication settings.
/cgi-bin/ordervalues.exe: the CGI
?: separation between the CGI and the parameters.
Listname: list (or file) name of the media, that will be ordered (listed, referenced). This name is then used to read the media or to rewrite the list. It is possible to have in the same time more than one list in the Web-Server, this will depend on the amount of memory allocated to the Web-Server RAM disk.
+: separation between parameters

CGI (Common Gateway Interface) of the Web-Server

- dummy:** this is a dummy parameter, not used today, but it must be in the request anyway, so the word “dummy” should be written in the request.
- ssss:** this has to be a number, which corresponds to the size of memory necessary to reserve in the RAM disk of the Web-Server, to save the actual sent list of media, and also to prepare the answer to this media list. Answer is the media list plus the media value. See readfile.exe. So the size here specified has to be large enough for the whole the list plus the answer. The number has to be specified in byte, so writing here 1000, means 1000 bytes reserved for the list specified. The default size of memory allocated for the RAM disk is 2000bytes, but it is possible to configure a bigger size.
- TAGx:** the media to be accessed.

The answer to this request is normally:

„HTTP/1.0 200 OK[0x0A]Content-Type: text/plain [0x0A][0x0A]done[0x0A]”, it just means, that the http request has been received correctly, but it doesn’t mean that the syntax of tags are correct.

[0x0A]: is it a hexadecimal value that can’t be represented as an ASCII character.

Example

We create a list of tags, this list of tags will be called room2, and will contain 3 tags which refer to the media MW 100, MW 102, and M 104.0.

```
http://pc_name/station_name /cgi-bin/OrderValues.exe?room2+dummy+1000+PDP,,MW100,u+PDP,,MW102,u+PDP,,M104.0,b
```

the answer is:

```
HTTP/1.0 200 OK[0x0A]Content-Type:text/plain[0x0A][0x0A]Done.[0x0A]
```

A.2.4 Readfile.exe

Read file is used to read the media values that have been predefined in the list with the “ordervalue” function. The function will return the list of the tags with their value.

The syntax is as follows :

```
http://pc_name/station_name/cgi-bin/ReadFile.exe?listname
```

- pc_name:** the PC where the Web-Connect is running.
- Station_name:** The name assigned to a PCD, corresponding to his communication settings.
- /cgi-bin/readfile.exe:** the CGI
- ?:** separation between the CGI and the parameters.
- Listname:** list (or file) name of the media list that have been ordered before with „ordervalue”.

```
HTTP/1.0 200 OK.Content-Typecontent/unknown Content-Length:
343[0x0A][0x0A]TAG1=20689[0x0A]TAG2=61377[0x0A]TAG3=543[0x0A]
TAG4=543[0x0A] TAG5=543[0x0A]
```

It will return the tags defined in the list and their values.

[0x0A]: it is a hexadecimal value that can’t be represented as an ASCII character.

Example



In this example we read the list created in the OrderValues chapter example. So the list name is "room2".

The request:

```
http://pc_name/station_name/cgi-bin/ReadFile.exe?room2
```

The answer:

```
HTTP/1.0 200 OK[0x0A]Content-Type: content/
unknown[0x0A]Content-Length: 47[0x0A][0x0A]PDP,,MW100,u=3450[0x
0A]PDP,,MW102,u=320[0x0A] PDP,,M104.0,b=0[0x0A]
```

A.3 HTML coding

In this chapter, we do not explain the HTML coding, but just how to work with the Web-Server directly from the HTML code.

A.3.1 Writing values in the PCD

In the Web-Server documentation we have already described how to write a value in the PCD, the example is made with the FrontPage Express editor. Here we show how to do it directly in the HTML code.

- 1) first of all, create a form and use the method „GET”. In this form you will need a text input field to enter the value to write and a button to submit the form.
- 2) The text input field is defined by the keyword „input” and the type „text”. Very important is the name of it, which is the tag referring to the PCD media that has to be written. In this example we refer to the MW200 and the format is binary. Note that the tag is specified without the % at the beginning and at the end. Because we don't want that the tag will be replaced with the value.
- 3) An input filed type „submit” is a button with the function to submit the form to the server.

```
<form method="GET">
  <p align="left"><input type="text" size="32"
name="PDP,,MW200,b">
  <input type="submit" value="MW 200"></p>
</form>
```

This will send one value to the Server to write (force) a media, it is of course, also possible to write more than one media in the same time, for it, just add as much as desired input filed as described in step 2.

A.3.2 Enter the Password

As you know, the access to the PCD Web-Server can be protected with passwords. To enter the password, a special page is available in the PCD, this page is named „pwdform.htm”. But for some reasons you like to create you own html page, with a different presentation. Here is the code necessary for it.

- 1) A form needs to be created, but make sure to use this time the method „POST”.
- 2) Then an input filed, the TYPE in our case is „PASSWORD”, like this the entered data will be hidden”, but you can also use a TYPE „text” if you don't want to hide the data entered. **Important is that the name is „pwd”.**
- 3) Then a button is needed, so input field of the type „submit” is necessary.
- 4) The Reset function is optional, it is just to reset the form if a mistake has been made while typing the password.



```
<FORM ACTION="login.cgi" METHOD="POST">
<P><INPUT TYPE="PASSWORD" NAME="pwd" SIZE="25"></P>
<INPUT TYPE="SUBMIT" VALUE="Submit">
<INPUT TYPE="RESET" VALUE="Reset">
</FORM>
```

A.4

As described in the documentation, you saw that it is possible to display the value of a PCD media into a html page, this by writing a tag in it. Then this tag is replaced by the media value when the Web-Server PCD is sending the file to the PC (and then into the Browser).

So the value in the PCD is changing, the only way to have an updated value is to upload the page again. This can work fine in some applications, but in some cases where you have pictures in the pages or large pages html. Loading the html page every time to have the value updated can have undesired side effect, which are: The page will disappear on the browser till the new page is there, uploading the whole page and eventually with pictures can take a long time.

So, it is possible to avoid those problems with a little trick. It's described below how to do it.

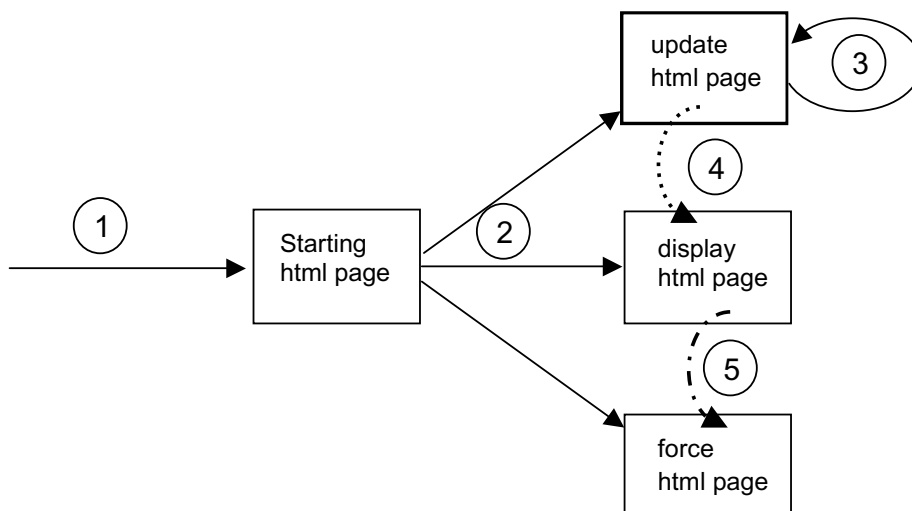
Also the same problem can happen if you have a form in your html page to force a value inside the PCD, when you are submitting the form, it will reload the html page.

This trick to update automatically the values and to show them, is not an official feature of the PCD Web-Server, is it just presented here as an example of the html and internet possibilities. This example is working and based on the Internet Explorer Version 6. For future version of the Internet Explorer or other browsers, the example may need some adaptation. Saia-Burgess Controls is not responsible for any problem that this example can cause or generate on your application.



A.4.1 The concept

The concept is to have more than one page to do all we want. In fact, one page will be displayed to the user, but in the background (invisible) we will have another page that will update values and also write values to PCD when necessary. This is possible thanks to the java script and the communication between different html pages. One page will do the work and communicate it to the other pages. Or one page is working and transfers this work to another page.



To make this system working with all possibilities (update values and force values),

we need 4 html pages. But finally only one page will be displayed.

The page **Starting** is called **(1)** by the user. This page will not do more than call **(2)** the other three pages and this by creating frames. One frame for each page, and the only visible frame will be the frame which contains the page **display** .

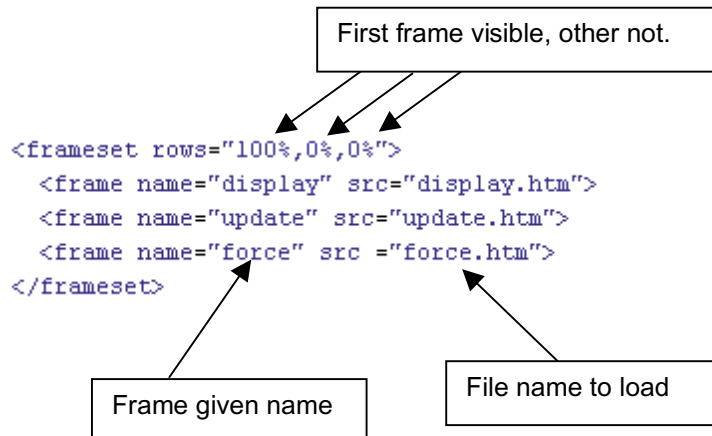
Now, the page **update** is the page that contains all the tags and which is regularly reloaded **(3)** from the PCD. This is done by an automated function in this page and a timeout. When the page is loaded, it is calling a function **(4)** in the display page to signal that new values are arrived, on the display page the function will then rewrite the value in each field that needs to be updated.

The page **force** is doing nothing till now than waiting. When the user like to force a value of the PCD, he will write a value in a field on the display page and /or click a button. This will call a function , that will take the value to be written, and transfer it **(5)**to the force page. Values are then introduced in a form like in the force page and submitted. This has the advantage, that only the force page will be reloaded after a submit of the force, the display page is not affected.

Now we saw the concept, let's see an example. This example is also delivered with the documentation. Project name is "**Dynamic_update**".

A.4.2 Starting page

As described before it is just making the frame and loading the 3 others pages. Now, for easier work with the different documents (from one to the other one), we give a name to each frame.



A.4.3 Update page

This page is the page collecting the PCD media and updating them automatically.

The PCD media are taken from the PCD always through a simple tag and directly placed in **(1)** in Java script Object (variable). It will be then easy to communicate this value to different pages or functions.

The value and object described just before, are of course available only when this page has been totally loaded. So we use the onload event **(2)**. Then it means that when the page is totally loaded function st () is executed.

```

<html>
<head>
<script>
  var MD200=%%PDP,,MD200,u%
  var MD204=%%PDP,,MD204,u%
  function st()
  {
  if (parent.display.document.readyState == "complete") parent.display.updated();
  setTimeout("window.location.reload()",3000)
  }
</script>
</head>
<BODY onload="st()">
</body>
</html>

```

The code is annotated with five numbered circles:

- 1**: Points to the variable declarations: `var MD200=%%PDP,,MD200,u%` and `var MD204=%%PDP,,MD204,u%`.
- 2**: Points to the `</html>` tag.
- 3**: Points to the condition in the `if` statement: `(parent.display.document.readyState == "complete")`.
- 4**: Points to the function call: `parent.display.updated();`.
- 5**: Points to the `setTimeout` function call: `setTimeout("window.location.reload()",3000)`.

This function will check if the display page exists **(3)** and if this condition is true, then it will call the function `updated()` **(4)** on the display page. We will see this function in the display page chapter. When the function has returned, it will start a timer **(5)**, for the reload of the page itself. When the timer has finished, the page upload is reloaded from the PCD and all is restarting as described from **(1)**.

Define variable

The variable or PCD media that have to be accessed inside the PCD, has to be defined in this page on part **(1)**.

A

Refresh time

If the refresh time has to be redefined, then this one has to be done in this update page only. This is done on the `setTimeout` function call, in part **(5)**. In this example it is set to 3000 milliseconds = 3 seconds.



Do not put this value too small, problems can occur in the synchronization of the html pages.

A.4.4 Force page

This page is just used to send values down to the PCD. This just to avoid that the Display page is reloaded after a submit of a form.

So mainly this page contains a form **(1)**, which we gave the name „F1” to work easier with it. In this form we have an input field, the name of the input field **(2)** corresponds to the tag (media of PCD) that we want to write the value too. As this input field will not be displayed (because the force page is not visible), we can declare this input field as „hidden” type.

The display page is calling the `send()` function and one parameter is the value to be sent to the media. So the function `send()` does first affect **(3)** the receive value to the input field belonging to this page. Then, the form in this page has to be submitted **(4)** and send to the PCD.

Finally to have an update of new values displayed on the screen, we push the reload **(5)** of the update page to be made (this command is totally optional).

Then the function returns and the page is reloaded automatically from the Web-Server and then he is ready for the next force of value.

```

<html>
<head>
<script>
function send(t)
{
  document.all.T1.value = t;
  document.F1.submit();
  parent.update.location.reload();
}
</script>
</head>
<BODY>
<form method="GET" name="F1">
<input type="hidden" name="PDP,,MD200,u" id="T1">
</form>
</body>
</html>

```

7.4.5 Display page

The display page is the most important one, as it is the only one which will be displayed. In this page we can find some java script functions on the top of the page and the html body, where finally the content of the display will be made.



In our example the browser will show this:

- (1) An input field and 2 buttons, that are used to increase or decrease the value inside the input field. But this field will be increased only when the value changed inside the PCD. So with that it is possible to be sure that the value has been increased or decreased.
- (2) A simple text field that just displays a value and which is also steadily updated.

Now, here are 2 functions in javascript:

- (3) Change() function is called when a button is pressed and the value has to be changed inside the PCD. The parameter is the field that has the value and the other parameter is the value to add or subtract to it. Then the function is just making the calculation and calling the function in the force page to force the value inside the PCD.

Example of dynamic data update and force value

Here; display the value of the Marker DWord 204, update every 3 secondes

!!! Don't forget to have the PCD in RUN to see the value moving !!!

- (4) This function is the one called from the update page, when the new value arrived from the PCD. Here we just make a test with the „SOE”, to see if the user is actually editing the input field, then in this case we don't want to overwrite the value that is being entered. After that we take the value from the page “update” and we put it inside the input field called here „T1”. And the second value in „T2”.

```
<script language="JavaScript"><!--
```

```
var SOE = 0
```

```
function change(t,i)
{
  i = Number(t.value) + i;
  parent.force.send(i);
}
```

3

```
function updated()
{
  if (SOE==0) document.all.T1.value = parent.update.MD200;
  document.all.T2.value = parent.update.MD204;
}
// --></script>
```

4

A

Now in the body of the html page, we need to create the text field and the button.

This is done as shown in the example below.

The input field is created (5) with the keyword “input” and type “text”, then the id is given by id=“T1”. In this example we also detect when the user put the cursor in the field and when the cursor is leaving the field, this is optional and this detection is done by onFocus and onBlur. The buttons are 2, one to increment (6) and the other one to decrement (7). The keyword is also input, but type is “button”, respectively they have ID “B1” and “B2”. Onclick is the event generated when the button is clicked. In this case we call the function described before in this document and we give the text field from which the value has to be modified and also give the value to add or subtract from it.

```
<td rowspan="2" width="50%"><input type="text" size="20" id="T1" onBlur="SOE=0" onFocus="SOE=1" >
```

5

```
<td width="50%"><input type="button" name="B1" value=" + " onclick="change(T1,+1)">
```

6

```
<td width="50%"><input type="button" name="B2" value=" - " onclick="change(T1,-1)"></td>
```

7

A.5 General principles of Web-Server technology

A.5.1 General notes concerning Web-Servers

The basis of Web-Server technology is the representation of web information (web = totality of different, associated HTML pages and their files, so that a menu structure emerges) using standard SW tools (Web-Browsers).

This includes the possibility of saving web information displayed directly to the PC on which the Web-Browser is located.

However, if web information is saved on a remote PC, this PC is called the Web-Server, only serving to store web information and transfer it on request to the Web-Browser.

The transfer of web information between the Web-Server PC and the Web-Browser PC can take place across any PC port (serial port, modem, LAN, Internet....) and normally occurs under the TCP/IP protocol.

Due to the limits of transmission capacity (e.g. on Internet) and the desire for fast loading times, the data structure of this web information has been very efficiently organized and needs little space in the Web-Server's memory.

The size of a normal HTML page is in the area of a few kBytes; image files are compressed with powerful algorithms, executable programs (e.g. Java-Applets) are processed interpretative in the browser (on the PC).

Since memory space is also limited in the controllers of industrial applications, Web-Server technology is ideally suited to this field too, for example to save images and files from control and monitoring interfaces directly to the industrial controller.

These small volumes of data place only a modest burden on the PCD's CPU and therefore allow the majority of the CPU's capacity to be used for control purposes.

As a result, control and monitoring information is always stored decentrally, directly on the controller. Any user now accessing this control and monitoring information, regardless of the device used to access the data, always receives the same, up-to-date information.

This dispenses with the need for synchronization of control and monitoring information whenever the application is updated (previously required by control and monitoring solutions that save information on the PC).

An additional advantage is the fact that web information can be edited with well known Microsoft tools, such as Word or FrontPage.

Web information is manipulated with standard browsers, designed for intuitive use and familiar to everyone.

A.5.2 TCP/IP protocol

TCP/IP is the lowest common denominator for overall data communication on Internet. From a historical point of view, it was only with this protocol that a limited network became a network of networks. Regardless of whether you call WWW pages, send e-mail, download files with FTP or work on a remote computer with Telnet: data is always addressed and transported in the same way. TCP stands for **Transmission Control Protocol**; IP stands for **Internet Protocol**.

When you send an e-mail or call a HTML file on WWW, data is broken up into little packages during transmission through the network. Each package contains information about the address to which it should be sent and the place of that package in the sequence of transmission.

The IP looks after addressing. An addressing plan exists for this purpose: the so-called IP addresses.

To ensure that packages of data actually reach their destination, and in the correct order, is the job of the TCP. The TCP uses sequential numbers for individual packages in a transmission. The transmission of data is not considered complete until all packages in a transmission have been received in full at their destination.

Every computer participating in the Internet is logged onto the network with an IP address. Computers that are connected to Internet are called **Hosts**. Therefore, when you use your PC to surf the WWW or collect new e-mails, you are logged onto Internet with an IP address. Your service provider, through whose host computer you dial up, can arrange fixed IP addresses for you. Major service providers, e.g. online services like CompuServe or AOL, also assign impersonal, dynamic IP addresses for every Internet dial-up. For a computer to participate on Internet, it must have software that supports the TCP/IP protocol. For example, under MS Windows this is the **winsock.dll** file in the Windows directory.

A.5.3 IP addressing

A typical IP address written in decimal looks like this: 149.174.211.5 – i.e. four numbers separated by stops. The stops have the job of addressing higher and lower ranking networks. Just as a telephone number in the international network has a country code, area code, subscriber number and sometimes also a direct dialling-in number, Internet also has an area code – the **network number**, and a direct dialling-in number - the **host number**.

The first part of an IP address is the network number and the second part is the host number. The location of the boundary between network number and host number is defined by a classification plan for network types. The following table explains this plan. In the 'IP addressing' and 'Typical IP address' columns, the network number (area code) is printed in bold characters. The rest of the IP address is the host number of a computer within that network.

Network type	IP addressing	Typical IP address
Class A network	xxx .xxx.xxx.xxx	103 .234.123.87
Class B network	xxx.xxx .xxx.xxx	151.170 .102.15
Class C network	xxx.xxx.xxx .xxx	196.23.155 .113

Class A networks form the highest level of the hierarchy. Only the first number in an IP address is the network number. All other numbers are host numbers within the network. Network numbers in this kind of network can have figures between 1 and 126, i.e. there can only be 126 class A networks in the whole world. An IP address that belongs to a Class A network is therefore identifiable by having its first number between 1 and 126. The American military network is an example of such a class A network. Within a class A network the relevant network operator can allocate at will the second, third and fourth figures of individual IP addresses to participants in the network. Since all three figures can have values between 0 and 255, a class A network operator can assign up to 16.7 million IP addresses to host computers in that network.

Class B networks are the second-highest level in the hierarchy. Such networks have network numbers that extend across the first two figures of the IP address. For the first figure, class B networks can have values between 128 and 192. An IP address that belongs to a class B network is therefore identifiable by having its first number between 128 and 192. Values between 0 and 255 are allowed for the second number. This means that around 16,000 such networks are possible. Since the third and

fourth numbers in such networks may also have values between 0 and 255, any class B network can have up to 65,000 host computers connected. Class B networks are mainly assigned to large companies, universities and online services.

Class C networks occupy the lowest level of the hierarchy. The first number of a class C network's IP address lies between 192 and 223. The second and third numbers also form part of the network number. Over two million such networks can be addressed in this way. These addresses are mainly assigned to small and medium-sized companies with direct Internet connections, but also to the smaller Internet providers. Since only one number is left with a value between 0 and 255, the maximum number of host computers that can be connected in a C network is 255.

Some people doubt whether this addressing plan can meet future needs. Ideas already exist for restructuring network and host computer addressing.

A.5.4 Client-Server-Technologie

Any host computer intending to offer to other computers such Internet services as World Wide Web, Gopher, e-mail, FTP, etc. must have the appropriate **server** software running on it. A host computer can only offer an Internet service if the appropriate server software is active on the computer and if the computer is online.

Servers are programs that permanently wait to receive enquiries regarding their service. A WWW server, therefore, waits to receive enquiries that will call off WWW pages from the server computer.

In contrast, **clients** are software programs that typically request data from servers. Their WWW browser, for example, is a client. If you click on a reference that leads to an HTTP address, the browser (i.e. the WWW client) starts a request to the relevant server on the remote host computer. The server evaluates the request and transmits the data requested. Protocols exist to regulate communication between clients and servers. Client-server communication in the WWW is basically regulated by the HTTP protocol. This type of protocol runs above the TCP/IP protocol.

Normally, a client requests data and a server transmits data. However, exceptions also exist whereby a client not only can request data, but also send data to a server (e.g. when you use FTP to load a file on the server computer; when you send an e-mail or fill in and return a form in the WWW). These cases are also referred to as **Client-Push** (the client pushes data onto the server).

Another exception is when the server becomes active first the sends the client something without being asked. This is called **Server-Push** (the server pushes data onto the client). New technologies want to elevate this exception to a rule. They are called **push technologies**. These technologies are intended to enable a client to receive data regularly, without personally asking for it. They make it possible to realize broadcasting services like current news, etc. Netscape and Microsoft Internet Explorer (both from version 4.0) have the relevant interfaces to make use of such services.

A.5.5 DNS - Domain Name Service

Generally, computers work better with numbers and people work better with names. For this reason a system has been invented that translates the numeric IP addresses into address names that are clear for the final user.

The system created has a hierarchical structure like the IP addresses. An address name in this system belongs to a **top level domain** and within that to a **sub-level domain**. Each sub-level domain can in turn contain subordinate domains, but does not have to. The individual parts of these address names are separated from each other

by stops, as with IP addresses. An example of such an address name is: **teamone.de**.

Top level domains occupy the final position of a domain name. As abbreviations, they more or less speak for themselves. The abbreviations for top level domains are either country codes or type codes. Examples are:

de = Germany (Deutschland)
au = Austria
ch = Switzerland
it = Italy
my = Malaysia
com = commercial proprietor
org = organization
net = general network
edu = American educational establishments
gov = American government offices
mil = American military installations

Each of these top level domains represents an administrative area for which an “administrative authority” also exists with responsibility for assigning the names of sub-level domains within its administrative area. For example, if you wanted to apply for a domain name like **MyCompany.de**, the application would have to be made to the DE-NIC (German Network Information Center). Commercial providers will do this for you if you use them for any corresponding service. However, you will only receive your chosen name if the address name has not been assigned elsewhere. Unscrupulous types therefore had the bright idea of reserving for themselves the names of large companies that had not yet applied for their own domain, to sell them on later at a high price, when even these large companies had recognized a sign of the times. Since then, measures have been taken to prevent such machinations. However, legal action is frequently resorted to in the cause of attractive domain names. If two companies happen to have identical names (but no other connection) and want to reserve the same domain name, only one application can be accepted. To minimize this type of conflict, new top-level suffixes have since been introduced.

The owners of two-part domain names can in turn assign sub-level domains. This has resulted, for example, in a domain name **seite.net**. The operators of this domain have in turn assigned sub-domains, producing domain addresses like **java.seite.net** or **javascript.seite.net**.

A.5.6 Routing and gateways

Internet, as the network of networks, cannot at present allow the direct transmission of data from one IP address to another within their own sub-network. In all other cases, when data is to be sent to another network number, computers that regulate traffic between networks come into the plan. These computers are called **gateways**. They route data from host computers in their own sub-network to gateways in other sub-networks and route incoming data from the gateways of other sub-networks to the host computer addressed in their own sub-network. Without gateways there would be no Internet.

Routing is the specific term for this activity and possible routes from a gateway computer’s own network to other networks are defined on the gateway computer in **routing tables**.

A gateway also has the job of finding an alternative route if the normal route does not work, perhaps due to a fault on the line or a data jam. Gateways constantly send



themselves test packages, to check the connection is working and to find routes for data transfer where the traffic is light.

This means, therefore, that when data transfer takes place on Internet it is not at all clear from the outset which route data will take. Even the separate packages that make up a single transmission can take completely different routes. For example, if from Germany you call a WWW page that is located on a computer in the USA it may be that half the page comes over the Atlantic and the other half over the Pacific before your WWW browser can display it. Neither you nor your browser would realize it.

A.6 General principles of HTML

A.6.1 Mark up elements

HTML stands for **H**yper **T**ext **M**arkup **L**anguage. The language is defined with the help of SGML (Standard Generalized Markup Language). SGML was laid down under ISO standard 8879.

As a markup language, HTML has the job of describing the logical components of a document. It therefore contains instructions for marking typical elements of a document, such as headers, text spaces, lists, tables or graphical references.

The HTML model assumes a hierarchical division. HTML describes documents. Documents have global properties, such a title or a background colour. The actual content consists of elements, e.g. a first-order heading. Some of these elements in turn have sub-elements. A text space, for example, might contain a text position that is marked as bold, a list comprises separate list items and a table is divided into individual cells.

A fixed extent can be defined for most of these elements. In this way, a header extends from the first to the last character, a list from the first to the last list item, or a table from the first to the last cell. Marks indicate the start and end of elements. The following model applies for marking up a heading:

[Heading] <i>Text of heading</i> [End of heading]

For an element that in turn has sub-elements, a list perhaps, the same model applies:

[List]
[List item] <i>Text of list item</i> [End of list item]
[List item] <i>Text of list item</i> [End of list item]
[End of list]

WWW browsers for HTML files work out the markup instructions and then display the elements on screen in a form that is easily recognizable to the eye. However, screen representation is not the only conceivable form of output. For example, HTML can just as readily output computer-generated voices onto audio systems.

A.6.2 Cross-linking with hyperlinks

One of HTML's most important features is the ability to define hyperlinks. Hyperlinks can lead to other places in the same project, but also to any other addresses on the world-wide web and even Internet addresses that are not part of the WWW.

With this simple, basic feature, HTML opens up completely new worlds. Moving between computers that are physically far apart is reduced to a mouse-click with modern, graphical WWW browsers. In your own HTML files you can include hyperlinks that allow context-specific links between your content and that of other bidders. This fundamental idea is the basis of the entire world-wide web, to which it owes its name.



A.6.3 Software-independent plain text

HTML has a plain-text format. You can process HTML files with any text editor that can store data as straight text files. There is no specific software required for the creation of HTML files. Powerful programs that specialize in the editing of HTML may have existed for a long time, but this does not alter its crucial characteristic: HTML is not tied to any specific, commercial software product. This might even be HTML's single most important feature, any you should never lose sight of it when anyone tries to tell you that web publishing is only possible with specific software products.

The plain-text instructions of HTML have been designed for machines **and** humans. Anyone who is not prejudiced against visible instructions on the screen will find in HTML an astonishingly simple shell language. It is based on English words, but the number of instructions is so limited that even those without extensive knowledge of the English language can assimilate HTML.

Since HTML has a plain-text format, it is excellent for generating with the help of programs. CGI programs, for example, make use of this capability. Whenever you use a search service on the WWW and, after your search request, are presented with the results, what you see on the screen is HTML code that has been generated by a program.

A.6.4 Universal in application

HTML was actually invented as a markup language for creating WWW pages. However, HTML files do not just work on the WWW. It is no problem to open an HTML file locally on any computer with a WWW browser. This makes HTML files ideally suited for local documentation, CD-ROM interfaces, readme files, etc. With HTML and the languages that complement it directly (CSS and JavaScript, which also function locally) you can create sophisticated projects not intended for use on the WWW. Regardless of whether you want to get your diary in shape for the next millennium, add HTML-based online help to the next version of your software, or produce an informative CD - HTML is by far the most widely used file format in the world. Your HTML files will run on any computer with a WWW browser installed - and any computer that does not have a WWW browser can these days be called obsolete in every respect.

A

A.6.5 Formatting for HTML elements

HTML is a "logical" language. With HTML, you define the basic structure of your WWW pages, the elements, structures, hyperlinks and referenced elements (such as graphics, multimedia, etc.). However, HTML was not really designed to specify exactly how an element should look. Therefore, you can define a heading in HTML and specify that it should be a first order heading. However, you cannot specify inside HTML how large the heading should be displayed, nor its font, etc. This is handled by the WWW browser during display. To do this it uses a mixture of basic settings, that the user can make, and fixed, programmed representations of individual HTML elements.

In the early days of HTML's great success, the language was "misused" for all kinds of physical formatting. This meant that an HTML command for displaying flashing text would suddenly pop up, and later a command became popular that could be used to define the font, type size (in 7 relative stages) and type colour. However, in the end all these implementations in HTML are inconsequential.

At this point Cascading Style Sheets (CSS) come in. This is a language that directly

complements and was specifically developed for HTML. It slots seamlessly into HTML and allows HTML elements to be formatted as desired. With the help of CSS style-sheets, for example, you can specify that all first order headings should be displayed as 24 points in height, written in red Helvetica, with a 16-point following space and a green double-line on the frame above. However, you can also just as easily specify for any text that it alone should be 3 centimetres high and should have the background colour yellow. In addition, the CSS language also contains instructions for the precise positioning of elements on the screen and for other output media, such as print or audio systems. CSS style-sheets therefore provide HTML with a powerful forward thrust.

The CSS language, just like HTML, has an official standard. Like HTML it is maintained and developed by the W3 consortium. In the meantime, more recent browser versions from Netscape and Microsoft also interpret the style-sheets.

A.6.6 A programming language for WWW pages

Among other things, forms can also be defined in HTML. Such forms can contain entry fields, selection lists, buttons, etc. The user can fill in a form and send it back via the WWW. This invention is a blessing for many purposes. However, HTML does not allow you as the supplier of a form, for example, to check the user's entries for completion or plausibility before the form is sent back.

Or another case: although you can integrate a VRML file into HTML, when that HTML file is on the WWW you cannot know whether the user has a browser that can display VRML. It would be useful if the instruction to integrate the VRML file could be made dependent on the user's browser being able to display VRML.

For these and countless other useful but less used purposes, Netscape invented a programming language called JavaScript. You can include JavaScript instructions directly inside HTML files, or integrate them as a separate file.

JavaScript is increasingly gaining in importance for modern WWW pages. Recent developments in JavaScript are above all responsible for what has been called "Dynamic HTML". Development has started and nothing can stop it now: JavaScript allows retrospective access to all elements of an HTML file during display. Fascinating new effects are therefore made possible, which normal HTML cannot do. A text can be replaced with another at the click of a mouse, a user event or time control can make elements disappear automatically, fade in, or change position on the screen.

If you want to create sophisticated WWW pages, you will find it hard to do without at least simple JavaScripts any more. For this reason SELFHTML contains extensive documentation on JavaScript. In any case, you should devote some attention to JavaScript as well as HTML

A.6.7 JavaScript and HTML

JavaScript is not a direct component of HTML, but a separate programming language. This language has, however, been specially created for the purpose of giving HTML authors a tool for optimizing WWW pages.

JavaScript programs can either be written directly in the HTML file, or in separate files. Unlike Java programs, they are not compiled, but interpreted as source text during run-time, i.e. like batch files or shell scripts. Modern WWW browsers such as Netscape or Microsoft Internet Explorer have the appropriate interpreter software for this.

A programming language such as JavaScript contains many confusing elements for



the beginner: special characters, variables, if-then statements, loops, functions, methods, parameters, objects, properties and much more. To get properly to grips with these elements, you must learn to imagine what is happening in the computer when program code is executed. This is a prolonged learning process that users only master with much practice. However, JavaScript is excellently suited to this, because it is a comparatively simple language without many of the work areas found in a “major” programming language, e.g. things like main memory management or file operations. In addition, JavaScript comes down to a specific environment, i.e. a WWW page that either has been displayed or will be.

A.6.8 JavaScript, JScript, ECMA-262, language versions

JavaScript is a script language introduced and licensed by Netscape. New versions of the Netscape browser therefore always offer rather more JavaScript than the competition, who can only wait to see what innovations Netscape has implemented.

Netscape 2.0 interprets JavaScript language standard 1.0, which was launched at the time. MS Internet Explorer mostly understands this language standard from version 3.0.

From version 3.0, Netscape interprets JavaScript standard 1.1. MS Internet Explorer mostly interprets this JavaScript language range from product version 4.0.

Netscape interprets JavaScript standard 1.2 from version 4.0. Some of the instructions in this language version are also interpreted by MS Internet Explorer 4.0.

Netscape and MS Internet Explorer also differ in the interpretation of individual object features. For example with Netscape, Cookies (a well known JavaScript feature) also function locally, whereas MS Internet Explorer, at least in version 3.x, requires a HTTP connection to support Cookies. Even with internal processes, such as loops that have to run through several times, there are differences between individual browsers and browser versions.

MS Internet Explorer 4.x does interpret JavaScript, but it also has Microsoft's own language variant **JScript**, which provides special instructions for the operating system extensions of MS Internet Explorer.

Both Netscape and Microsoft stress that their language interpreters comply with the standard for Internet script languages: **ECMA-262**. The ECMA committee, which comprises various software manufacturers and includes Microsoft and Netscape, seeks to define a generally valid standard for the field of script languages, as is the case with HTML or CSS style sheets.

A.6.9 HTML editors

HTML pages can be developed with any preferred HTML editor. One can divide HTML editors fundamentally into either text-based editors or WYSIWYG editors (WYSIWYG = **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et). With text-based editors you work directly with the HTML instructions. As a rule such editors have a toolbar with buttons and various menu commands to insert HTML tags in the text. Tags are displayed visibly in the text. WYSIWYG editors also offer a toolbar with buttons and menu commands to support the setting of HTML tags in the dialogue. However, the difference is that tags are not displayed (or only if requested). Instead, the text displayed on the screen is already as it will appear later, on the Web-Browsers.






At most of the following addresses you cannot download the relevant HTML editors. These relate mostly to shareware products, i.e. you are allowed to test the program, but must buy and register it before you can definitely use it.

This selection makes no claims to be comprehensive.

1-4-All	http://www.mmsoftware.com/14All/	HTML editor
4W Publisher	http://www.4w.com/4wpublisher/	Database tool that can automatically create reports in HTML form
AOL Press	http://www.aolpress.com/press/index.html	Wysiwyg HTML editor
Arachnophilia	http://www.arachnoid.com/arachnophilia/	HTML editor
Coffee Cup	http://www.coffeecup.com/editor/	HTML editor with ready-made JavaScripts, graphics, etc.
Claris Homepage	http://www.clarishomepage.com/	Wysiwyg editor
Frontpage	http://www.microsoft.com/frontpage/	Wysiwyg HTML editor by Microsoft
Horizon Web Text	http://www.gcnet.com/bw/horizon/	32-bit HTML editor
HTMLed (Pro)	http://www.ist.ca/	HTML editor
HTML Assistant	http://www.exit0.com/ez1/products/pro2000.html	HTML editor
SchemaText	http://www.schema.de/sitehtml/site-d/schemat5.htm	Professional authoring system for major web project
NetObjects	http://www.netobjects.com/	Powerful authoring tool
SuperHTML	http://www.superhtml.de/	HTML editor
Tarantula	http://www.indian-sites.com/nostrumindia/	Wysiwyg editor
WebMedia Publisher	http://www.wbmedia.com/publisher/	WYSIWYG HTML publisher

B Appendix B

B.1 Icons

	In manuals, this symbol refers the reader to further information in this manual or other manuals or technical information documents. As a rule there is no direct link to such documents.
	This symbol warns the reader of the risk to components from electrostatic discharges caused by touch. This could happen, if cassettes have to be opened for changing jumpers like described for such cassettes in chapter 2.8 and 2.9. Recommendation: at least touch the Minus of the system (cabinet of PGU connector) before coming in contact with the electronic parts. Better is to use a grounding wrist strap with its cable attached to the Minus of the system.
	This sign accompanies instructions that must always be followed.
	Explanations beside this sign are valid only for the Saia PCD Classic serie.
	Explanations beside this sign are valid only for the Saia PCD xx7 serie.

B

B.2 Adresse

Saia-Burgess Controls AG
Bahnhofstrasse 18
CH-3280 Murten | Switzerland

Telephone +41 26 672 72 72

Telefax +41 26 672 74 99

E-mail: support@saia-pcd.com

Homepage: www.saia-pcd.com

Technical support: www.sbc-support.com