Creating and Using Own Web Pages for S7-1200

SIMATIC STEP 7 V11

Application Description • March 2012

Applications & Tools

Answers for industry.

SIEMENS

Siemens Industry Online Support

This article is taken from the Siemens Industry Online Support. The following link takes you directly to the download page of this document:

http://support.automation.siemens.com/WW/view/en/58862931

Caution

The functions and solutions described in this article confine themselves to the realization of the automation task predominantly. Please take into account furthermore that corresponding protective measures have to be taken up in the context of Industrial Security when connecting your equipment to other parts of the plant, the enterprise network or the Internet. Further information can be found under the Item-ID 50203404.

http://support.automation.siemens.com/WW/view/en/50203404

You can also actively use our Technical Forum from the Service & Support Portal regarding this subject. Add your questions, suggestions and problems and discuss them together in our strong forum community:

http://www.siemens.com/forum-applications

SIEMENS

SIMATIC
S7-1200 Web server

SIMATIC STEP 7 V11

Warranty and liability

Note The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These application examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.
If there are any deviations between the recommendations provided in these application examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of Siemens Industry Sector.

Table of Contents

1 Automation Task

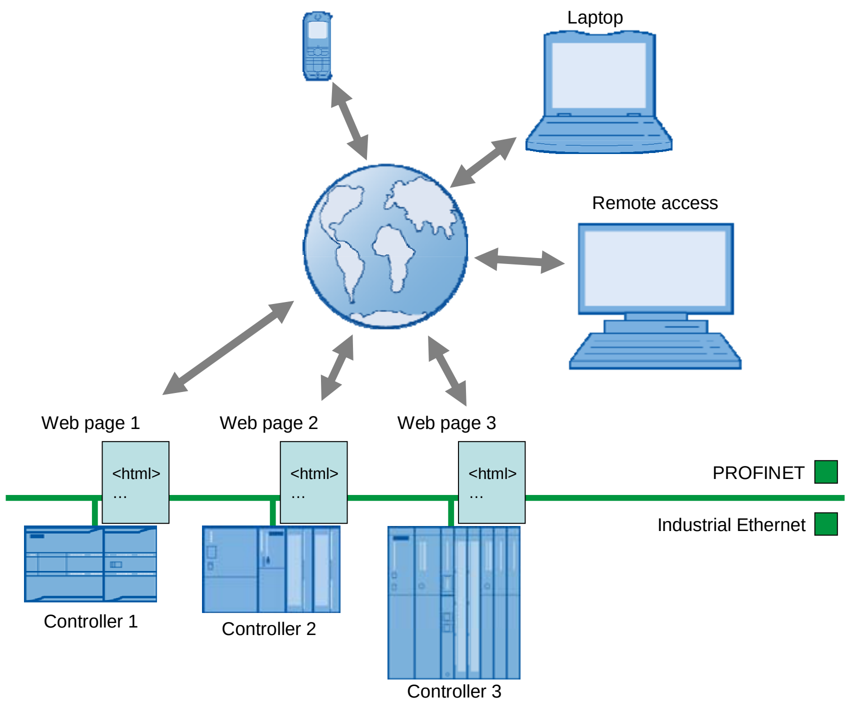
1.1 Overview

Overview of the automation task

Modern automation technology increasingly integrates internet technologies which – together with an integrated Ethernet-based communication – enable, for example, direct access to the system via the intranet. During the test and commissioning phase, the commissioning engineer wants to have flexible access to the CPU; individual data is to be visualized during operation for diagnostic purposes.

For access mechanisms via the internet or intranet it is reasonable to use already existing standards, such as, for example, http technology, standard web browsers and common "languages" such as HTML (Hypertext Markup Language) or JavaScript.

Figure 1-1



Description of the automation task

If you want to access a CPU via standard web mechanisms, the following requirements are to be met:

- Access the CPU with standard hardware and standard mechanisms via Industrial Ethernet. You do not require any additional hardware and software.
- Access the CPU individually related to the system and also visualized, if required. Each CPU has its individual web page "in the stomach".
- Operating personnel without any automation knowledge is also provided simple access to the CPU.

2 Automation solution

2.1 Overview of the general solution

Schematic layout

SIMATIC CPUs with PROFINET interface provide the opportunity to access CPU variables with the help of web pages provided by the system.

Access the web server of the CPU via a web browser. In addition to the standard mechanisms of the web page, such as, identification, diagnostic buffer, module status, communication, variable status and data logs, you can design and call individual web pages for your particular application.

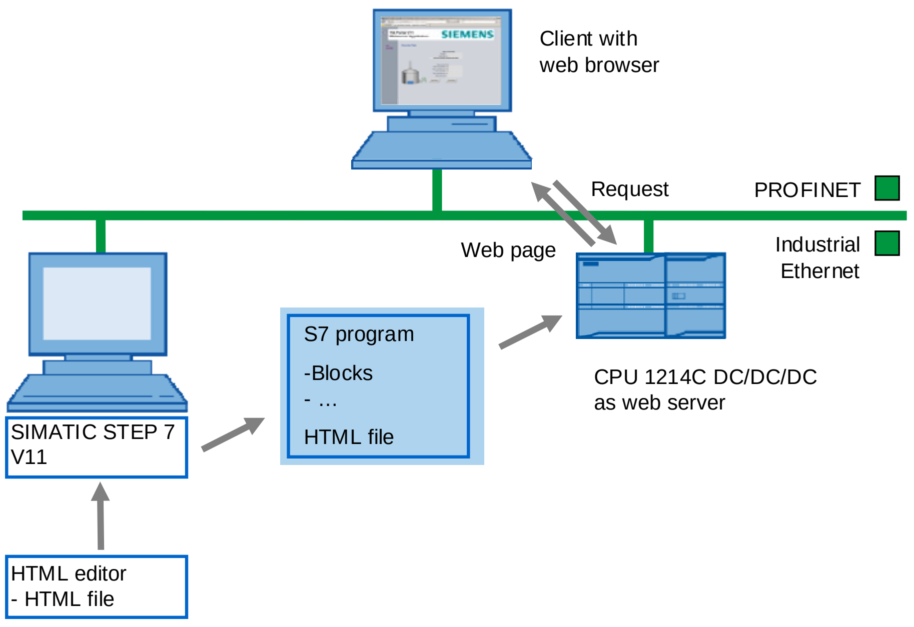
The web server with the web page is already integrated in the CPU.

To create your individual web page (user-defined web page), you can use any tools such as Microsoft Frontpage, Notepad++, etc.. For designing your web page, you can use all options provided to you by HTML, CSS (Cascading Style Sheets) and JavaScript.

In addition, there is a special command syntax (AWP command) for directed communication with the CPU.

The following figure gives an overview of the implemented solution.

Figure 2-1



Advantages and application options of web server applications

By having access options through various web browsers, control data can be displayed and to a limited degree controlled, by any computer or web-enabled devices without additional software installation.

Another advantage is the use of the entire network infrastructure of a plant without any additional hardware components. I.e. each place of the plant where a network access is provided, can access the respective controllers.

Evaluating, diagnosing and controlling the controllers can therefore also be performed over large distances or mobile communication devices such as, for example, PDAs.

However, due to the missing time deterministic of web applications, the use of the web server is not a full-fledged substitute for an operator control and monitoring system. The replacement of a HMI system is therefore not the aim of this application!

WARNING

No safety-relevant functions should be realized via the web server functionality due to the missing time deterministic of web applications!

CAUTION

In addition, please note that suitable security measures in compliance with the applicable Industrial Security standards must be taken, if your system is interconnected with other parts of the plant, the company's network or the Internet. For further information, please refer to the following entry:

http://support.automation.siemens.com/WW/view/en/50203404

Procedure for creating user-defined web pages at a glance

Figure 2-2

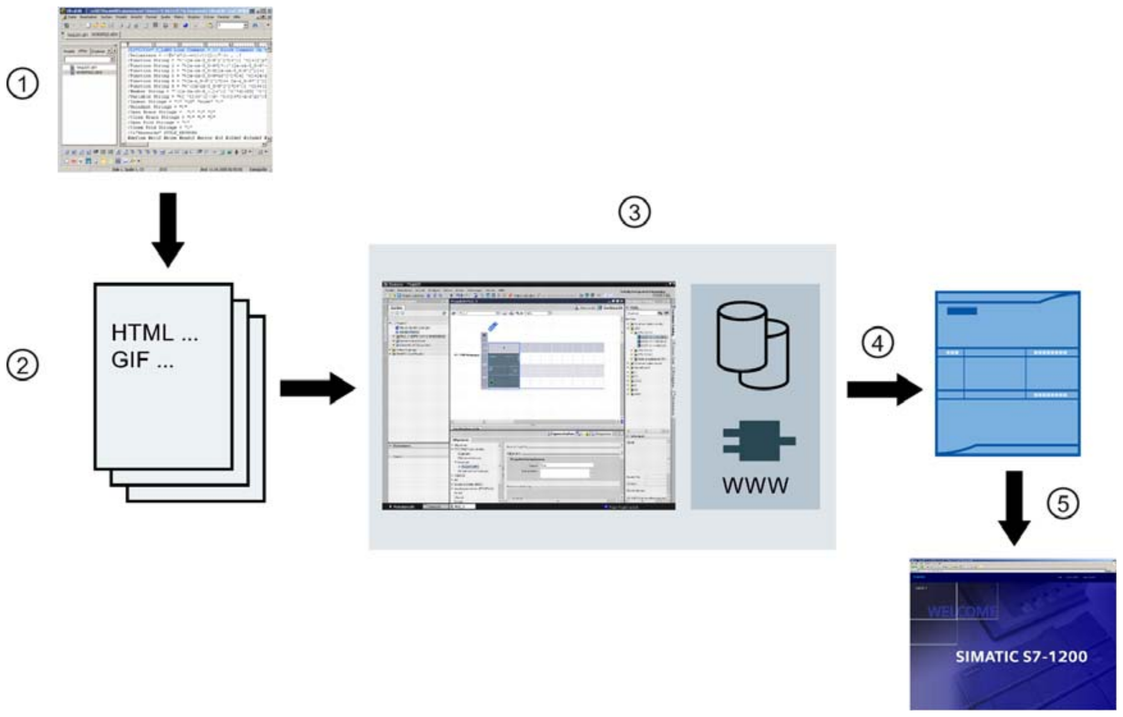


Table 2-1

No.	Instructions
1.	With an HTML editor, you create the HTML file for the user-defined web page.
2.	The web application consists of individual source files, for example, *.html, *.gif, *.js, …
3.	The HTML files with images etc. are stored in data blocks with SIMATIC STEP 7 V11 SP2. Call the WWW instruction in the S7 program
4.	Transfer all blocks to the CPU.
5.	Open the web page of the CPU via a web browser. Accessing the web server of the CPU can be irrespective of the configuration computer. Every output device with access to the PN interface of the CPU can display the web page.

Detailed explanations of the creation of a web page and programming in STEP 7 can be found from chapter 5 Configuration and Settings.

Structure of the application

This application was realized with a CPU 1214C DC/DC/DC. A PC is connected via the PROFINET interface. The PC serves for the creation of the S7 program and the HTML file, as well as for displaying the web page in a web browser.

Shown are all steps necessary to create a web page and to subsequently call it via the CPU.

Topics not covered by this application

This application is an introduction to user-defined web pages for beginners. Shown are simple methods for accessing the web page of a CPU with HTML and SIMATIC STEP 7 V11 SP2.

This application does not include a complete description of HTML. To gain deeper knowledge of HTML and JavaScript, please refer to the literature and internet pages specified in chapter 9 Literature.

Assumed knowledge

We assume that you are already familiar with SIMATIC S7 and STEP 7. Basic knowledge of HTML is not necessarily required.

2.2 Description of the content of this application

2.2.1 Content of the example application

The example application provides the following detailed contents:

- Configuration of the web server for a CPU with PN interface
- Creation of a user-defined web page for the CPU with the following functions: The figures in Table 2-2 relate to Figure 2-3 to Figure 2-5.

Table 2-2

No.	Description
1.	Displaying CPU variables
2.	Graphic display of CPU variables
3.	Setting of CPU variables
4.	Checking the input values with Javascript
5.	Displaying of texts which are linked with CPU variables
6.	Displaying of pictures which are linked with CPU variables
7.	Going to web pages with links in the navigation bar
8.	Cyclic refreshing of the web page
9.	Time-optimized refreshing of variables

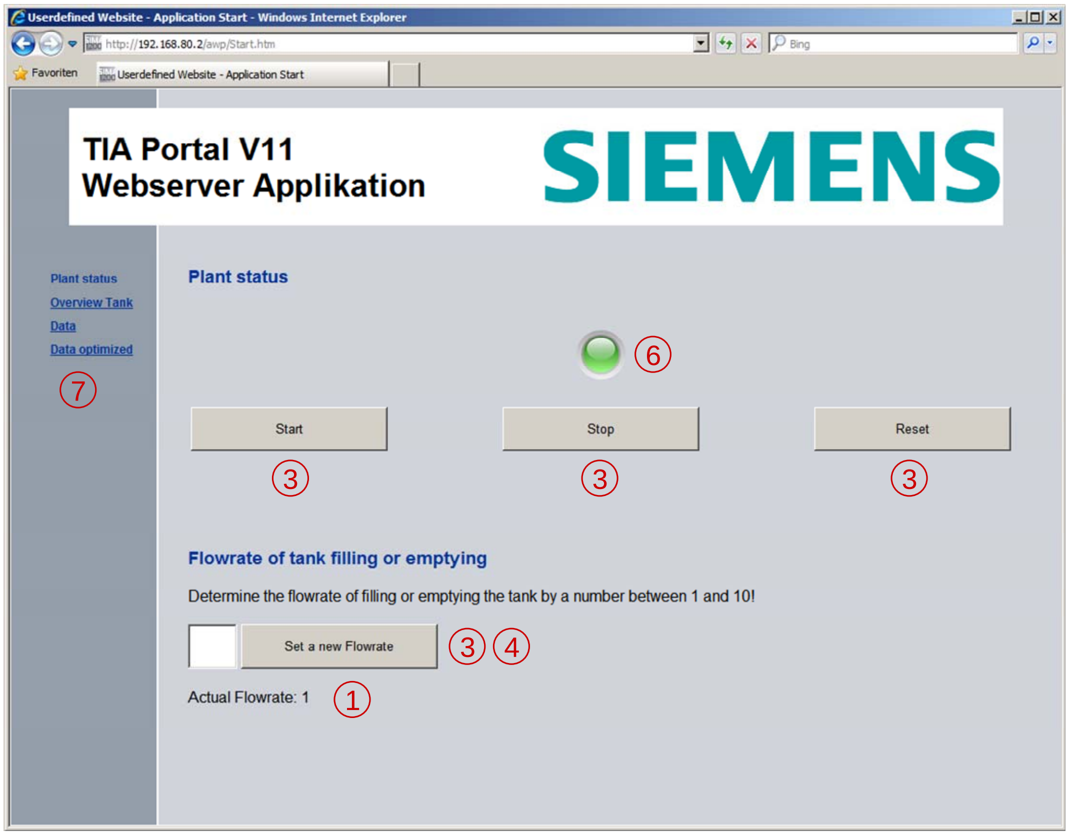
- Particularities in the S7 program creation
  - Providing variables for the web page
  - Further processing of variables from the web page in the S7 program

2.2.2 Overview and description of the user interfaces

The following figures show the web pages created in this application:

“Start” web page

Figure 2-3


Userdefined Website - Application Start - Windows Internet Explorer
http://192.168.80.2/awp/Start.htm
Bing
Favoriten
Userdefined Website - Application Start
TIA Portal V11
Webserver Applikation
SIEMENS
Plant status
Overview Tank
Data
Data optimized
⑦
Plant status
⑥
Start
Stop
Reset
③
③
③
Flowrate of tank filling or emptying
Determine the flowrate of filling or emptying the tank by a number between 1 and 10!
Set a new Flowrate
③ ④
Actual Flowrate: 1
①

The web page shows the start page of the application.

You can start the application via the "***Start***" button and end it with the ***"Stop"*** button. The ***"Reset"*** button puts the application back into its original state. The operating status of the application is displayed via the LED.

The flow rate with which the tank is filled or emptied is defined via the "***Set a new Flowrate***" button. By default, a medium flow rate of 5 is set. The higher the value for ***"Flowrate"***, the faster the tank is emptied or filled.

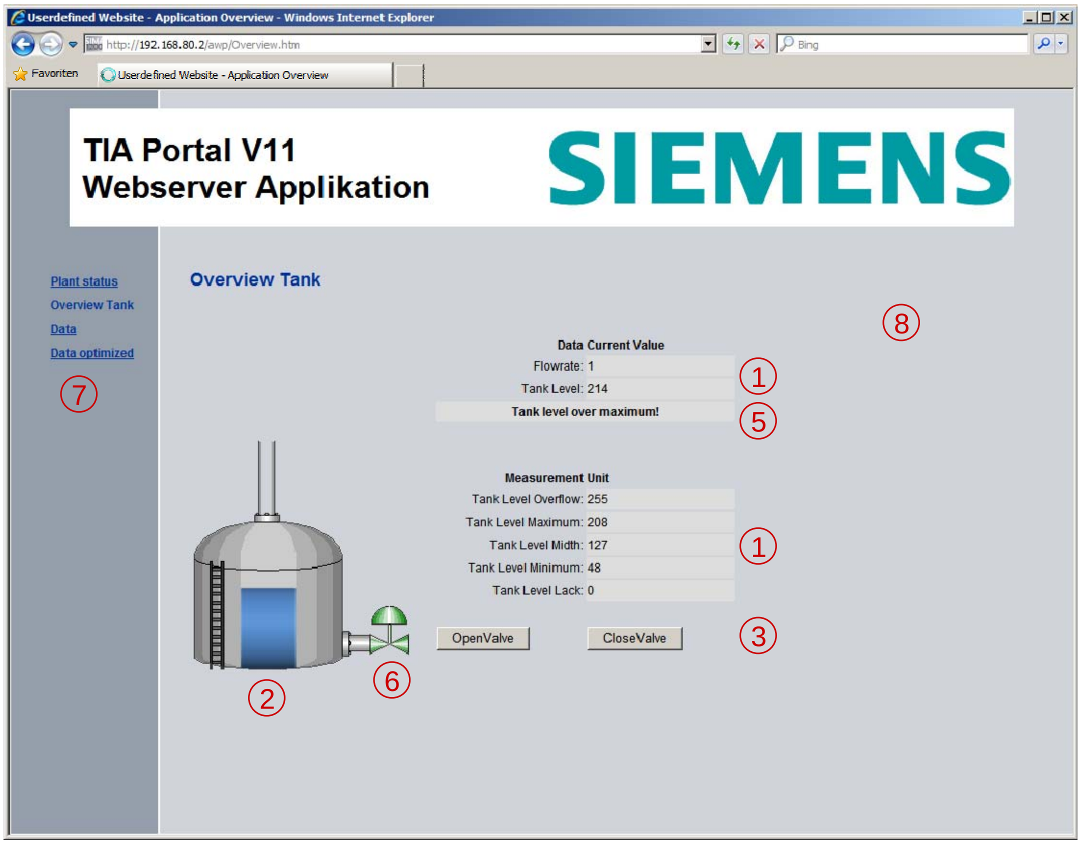
Entering the flow rate is monitored by a script. If there is an incorrect entry, a message is output.

The current flow rate is displayed under “Actual Flowrate”.

The navigation bar provides links to other pages of the application.

“Overview Tank“ web page

Figure 2-4


Userdefined Website - Application Overview - Windows Internet Explorer
http://192.168.80.2/awp/Overview.htm
Favoriten
Userdefined Website - Application Overview
TIA Portal V11
Webserver Applikation
SIEMENS
Plant status
Overview Tank
Data
Data optimized
Overview Tank
OpenValve
CloseValve
1
2
3
5
6
7
8

Data Current Value

Flowrate:	1
Tank Level:	214

Tank level over maximum!

Measurement Unit

Tank Level Overflow:	255
Tank Level Maximum:	208
Tank Level Midth:	127
Tank Level Minimum:	48
Tank Level Lack:	0

The web page shows a tank with the *"**Tank Level**"*. The limit values of the tank filling level can be found under ***"Measurement"***.

Via the ***"OpenValve"*** button, the tank valve can be opened so that the liquid is emptied. With the ***"CloseValve"*** button, the tank valve is closed so that the tank is filled. Dependent on the pressed button, the valve position is indicated via the color.

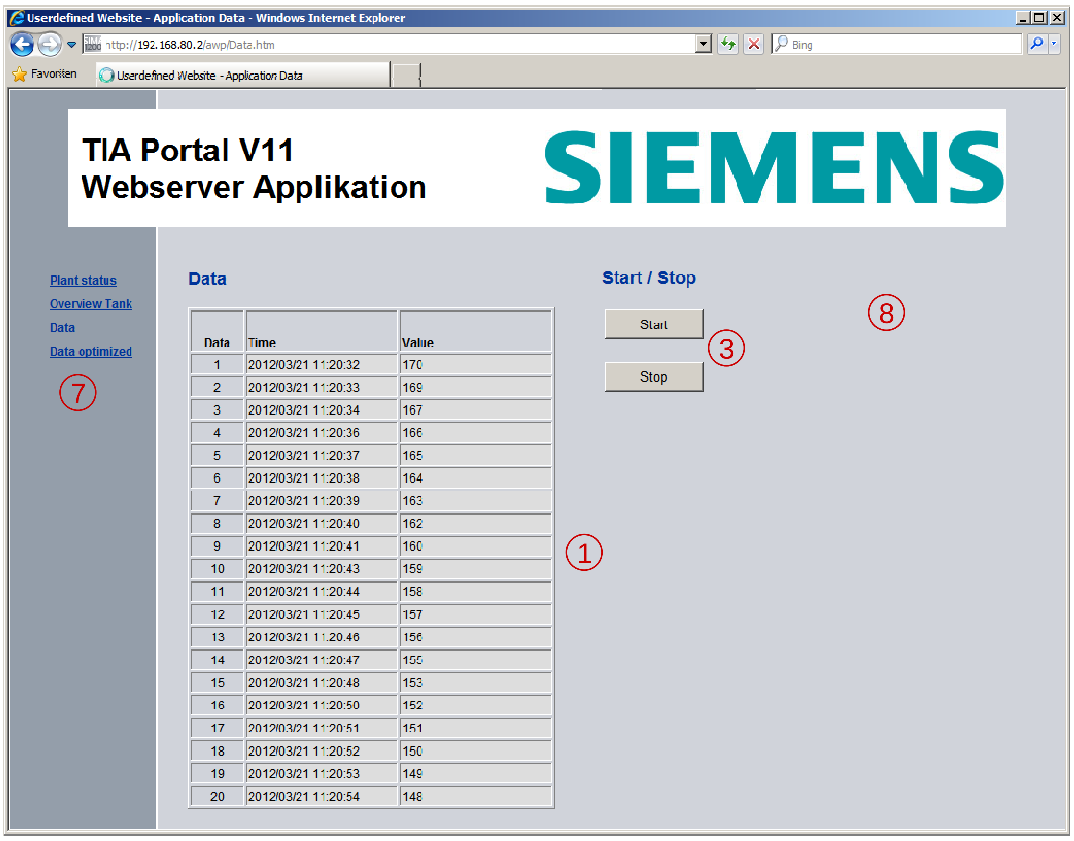
Via the message, the status of the tank filling level is indicated in clear text.

The page is automatically updated.

The navigation bar provides links to other pages of the application.

“Data“ web page

Figure 2-5


Userdefined Website - Application Data - Windows Internet Explorer
http://192.168.80.2/awp/Data.htm
Bing
Favoriten
Userdefined Website - Application Data
TIA Portal V11
Webserver Applikation
SIEMENS
Plant status
Overview Tank
Data
Data optimized
7
Data
Start / Stop
Start
Stop
3
8
1

Data	Time	Value
1	2012/03/21 11:20:32	170
2	2012/03/21 11:20:33	169
3	2012/03/21 11:20:34	167
4	2012/03/21 11:20:36	166
5	2012/03/21 11:20:37	165
6	2012/03/21 11:20:38	164
7	2012/03/21 11:20:39	163
8	2012/03/21 11:20:40	162
9	2012/03/21 11:20:41	160
10	2012/03/21 11:20:43	159
11	2012/03/21 11:20:44	158
12	2012/03/21 11:20:45	157
13	2012/03/21 11:20:46	156
14	2012/03/21 11:20:47	155
15	2012/03/21 11:20:48	153
16	2012/03/21 11:20:50	152
17	2012/03/21 11:20:51	151
18	2012/03/21 11:20:52	150
19	2012/03/21 11:20:53	149
20	2012/03/21 11:20:54	148

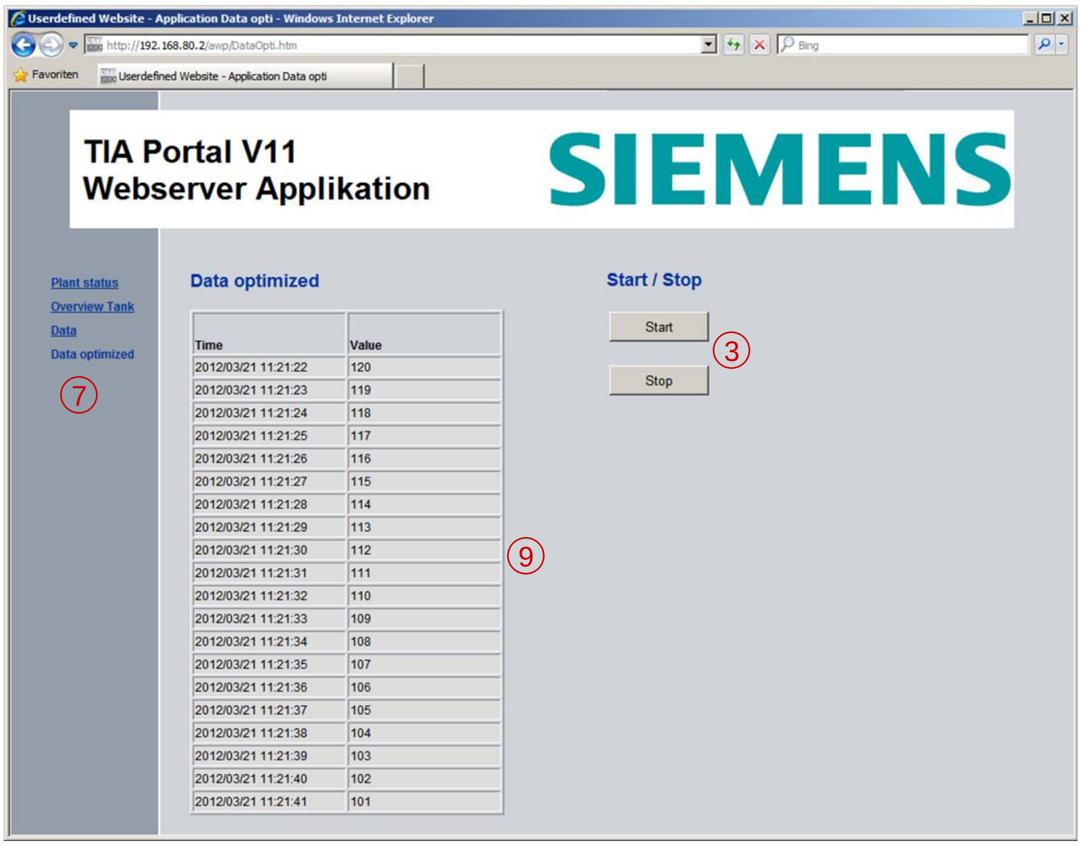
The web page shows the ring buffer of the last 20 tank fill levels.

The page is automatically refreshed.

The navigation bar provides links to other pages of the application.

"Data optimized" web page

Figure 2-6


Userdefined Website - Application Data opti - Windows Internet Explorer
http://192.168.80.2/awp/DataOpti.htm
Bing
Favoriten
Userdefined Website - Application Data opti
TIA Portal V11
Webserver Applikation
SIEMENS
Plant status
Overview Tank
Data
Data optimized
7
Data optimized
Start / Stop
Start
Stop
3
9

Time	Value
2012/03/21 11:21:22	120
2012/03/21 11:21:23	119
2012/03/21 11:21:24	118
2012/03/21 11:21:25	117
2012/03/21 11:21:26	116
2012/03/21 11:21:27	115
2012/03/21 11:21:28	114
2012/03/21 11:21:29	113
2012/03/21 11:21:30	112
2012/03/21 11:21:31	111
2012/03/21 11:21:32	110
2012/03/21 11:21:33	109
2012/03/21 11:21:34	108
2012/03/21 11:21:35	107
2012/03/21 11:21:36	106
2012/03/21 11:21:37	105
2012/03/21 11:21:38	104
2012/03/21 11:21:39	103
2012/03/21 11:21:40	102
2012/03/21 11:21:41	101

The web page is divided in a HTML file (main page) without variables and additionally an HTML file each for reading and an HTML file for writing control variables. The additional HTML files are embedded as frames (*iframe*) in the main page.

The web page shows the last 20 tank fill levels. The variables to be read are not transmitted individually to the web server but in 3 output strings. A JavaScript splits the output strings into individual variables and provides the main page with the outputs. Refreshing the frames is also taken on by JavaScript.

You can start the application via the ***"Start"*** button and end it with the ***"Stop"*** button. Since the buttons are configured in an independent frame, not all of the page will be updated when writing.

The navigation bar provides links to other pages of the application.

2.3 Hardware and software components used

This application was created with the following components.

Hardware components

Note For this application, you require the current firmware version of the CPU. Depending on the CPU type, the following entries contain links to the corresponding downloads:

- S7-1200:
  http://support.automation.siemens.com/WW/view/en/41886031/133100

Table 2-3

Component	Qty.	Order number	Note
CPU 1214C DC/DC/DC Firmware V2.2	1	6ES7214-1AE30-0XB0	Alternatively every S7-1200 with firmware V2.2 can be used.
PG/PC with Ethernet interface	1	-	-
IE FC TP STANDARD CABLE	1	6XV1840-2AH10	Connecting cable IE; minimum order quantity 20m
RJ45 plug-in connector	2	6GK1901-1BB10-2AA0	Can be tailored

Standard software components

Table 2-4

Component	Qty.	Order number	Note
SIMATIC STEP 7 V11 SP2	1	6ES7822-1AA01-0YC5	
Software tool for creating HTML files, e.g. Frontpage, Notepad++, …	1	-	
Web browser, e. g. Internet Explorer, Mozilla Firefox[1)]	1	-	Application optimized for Internet Explorer 8.0.

[1)] The web server of the CPU S7-1200 supports the web browser:

- Internet Explorer 8.0 or higher
- Mozilla Firefox 3.0 or higher
- Opera 11.0 or higher

Note For the use of other browsers, adjustments may have to be made.

Note At the time of development of this application, it wasn't possible to write tags with Internet Explorer 9.0 to the CPU. This is neither a problem of the S7-1200's web server nor of the programmed application.

We recommend use of Internet Explorer 8.0 to amend the situation."

Sample files and projects

The following list includes all files and projects used in this example.

Table 2-5

Component	Note
Example_S7-1200_Webserver.zip	The zip file contains the STEP 7 project with the related HTML file. The HTML file with the associated files, are located in the \html directory.
58862931_S7-1200_Webserver_DOKU_V10_en.pdf	This document.

3 Basics for Creating Web Pages

General definitions

In the context of web design, the term web page is used for a document in the World Wide Web, which can be called from a web server with a web browser by specifying a Uniform Resource Locator (URL). In this context, it is also referred to an HTML page or an HTML document.

A user-defined web page is understood as a web page with an additional command syntax (AWP commands) which can be used to access a S7 CPU with PN interface.

3.1 General principles of web pages

If you already have basic knowledge of HTML, you may skip this chapter and continue reading at chapter 3.2 Principles of standard web pages.

3.1.1 Principles of HTML

HTML stands for "Hypertext Markup Language" and is a text-based markup language for structuring headers, texts, lists, tables or images. Among other things, HTML does not use loops and variables and is therefore not a programming language.

Structure

An HTML document consists of three areas:

- Document type declaration (doctype) at the beginning of the file, stating the document type definition (DTD) used, e.g. HTML 4.01 Transitional.
- HTML head for information which is not to be displayed in the display area of the web browser.
- HTML body for information which is displayed in the web browser.

HTML elements (tags)

Elements are used to identify and structure different parts of a web page.

The HTML files contain "**HTML elements"** that are marked by tags (tag pairs). Almost all HTML elements are marked by an introductory and a concluding tag. The content in between is the "scope of application" of the corresponding element.

Example: Text paragraphs are marked by the <p> tag. The end of a tag is represented by an introductory "</".

<p>This is a text.</p>

Tags are cascadable and can be nested.

Typical tags

The following table gives an overview of the most important tags for structuring information, which are also used in this example application:

Table 3-1

Representation	Function	Example
`<!-- … -->`	Comment	`<!-- This is a comment! -->`
`<a> … </a>`	Link	
`<b> … </b>`	Boldface	`<b>This text is bold</b>.`
`<body> …</body>`	Content is displayed in the web browser	
`<div> … </div>`	Grouping of other elements	
`<form> … </form>`	Defines a form	
`<h1> … </h1>`	Text heading	
`<head> … </head>`	Head area of an HTML file	
`<html> … </html>`	Fundamental web page tag	
`<iframe> … </iframe>`	Defines an embedded window	
`<img>`	Integration of an image	
`<input>`	Creates a form element	
`<link>`	Defines a logic relationship to other files	
`<meta>`	Defines meta data	
`<p> … </p>`	Text paragraph	
`<script> … </script>`	Defines an area for scripts (e.g. JavaScript)	
`<style> … </style>`	Definition domain for stylesheet formatting	
`<table> … </table>`	Table Creates a table in combination with `<tr>` and `<td>`	
`<td> … </td>`	Table column	
`<th> … </th>`	Table head	
`<tr> … </tr>`	Table row	

3.1.2 Using forms

Forms are used for being able to perform interactions with the user in HTML.

For example, the user can fill in input fields in a form and then send the form by clicking a button. The content of the form is thus sent to the web server.

With the "POST" method, the content of the form is transferred from the web browser to the web server with a special POST request.

3.1.3 Basics on Cascading Style Sheets (CSS)

CCS is a formatting language for HTML elements. With the help of style sheets, for example, font, font size, colors, border, height, width etc. are specified for HTML elements.

You can define central formats for all, e.g. first order headings, table cells, etc.

CSS formats have the following structure:

Selector {Property: value}

A selector may contain several declarations (property: value).

Typical CSS properties

The following table gives an overview of the most important properties for formatting HTML elements which are also used in this example application:

Table 3-2

CSS property	Function	Examples for values
`position`	Position type	`static, relative, absolute, fixed`

`top`
`left`
`bottom`
`right`
	Start position from top
	Start position from left
	Start position from bottom
	Start position from right
		`10px, 2%`

`width` `height`	Width Height	`100px, 20%`
`direction`	Direction	`ltr, rtl`
`z-index`	Layer position for overlapping	`1, 2`
`font-family`	Font	`Arial, Helvetica`
`font-style`	Font style	`italic, oblique, normal`
`font-size`	Font size	`20px,100%, small, medium, large`
`font-weight`	Font weight	`bold, normal, bolder, lighter, 100 to 900`
`text-decoration`	Text decoration	`underline, blink, none`
`text-transform`	Text transformation	`uppercase, lowercase`

CSS property	Function	Examples for values
`color`	Text color	`rgb(51,102,170), #FFFFFF`
`vertical-align:`	Vertical alignment	`top, middle, bottom`
`text-align`	Horizontal alignment	`left, center, right, justify`

`margin`
`margin-top`
`margin-right`
`margin-bottom`
`margin-left`
	Margin/Distance general
	Margin/distance top
	Margin/distance right
	Margin/distance bottom
	Margin/distance left
		`10px, 5%`

`padding`
`padding-top`
`padding-right`
`padding-bottom`
`padding-left`
	Padding general
	Padding top
	Padding right
	Padding bottom
	Padding left
		`10px, 5%`

`border[-top, -right, -bottom, -left]`
`border[-top, -right, -bottom, -left]`
`border[-top, -right, -bottom, -left]`
`border[-top, -right, -bottom, -left]`
`border-collapse`
	Border general
	Border thickness
	Border color
	Border type
	Border model
		`2px solid white`
		`2px, 1%, thin, medium, thick`
		`#FFFF00, white`
		`none, hidden, dotted, solid, dashed, double`
		`separate, collapse`

`background`
`background-color`
`background-image`
`background-repeat`
`background-attachment`
`background-position`
	Background color and image
	Background color
	Background image
	Repetition effect
	Water mark effect
	Background position
		`Image.png no-repeat`
		`rgb(51,102,170), #FFFFFF`
		`Image.png`
		`repeat, no-repeat, repeat-x, repeat-y`
		`scroll, fixed`
		`10px 10px, top, bottom, center, left, right`

`list-style-type`	List style type	`none, square, circle, disc`
`empty-cells`	Display of empty cells	`show, hide`

Integration of Cascading Style Sheets (CSS) in HTML

There are several ways to integrate style sheets into an HTML file:

- within an HTML element
- between the `<script>` and `</script>` tags
- in an external CSS file

Style sheets are defined in a separate CSS file if you want to use uniform formats in several HTML files. This CSS file is simply integrated in the HTML file. The syntax is as follows:

```
<link rel="stylesheet" type="text/css" href="<Formats>.css">
```

The defined style sheets are addressed with the id and class attributes of the HTML tags. CSS provides extensive formatting options and the overview in HTML file is maintained.

3.1.4 Principles of JavaScript

JavaScript is an own programming language and was created for the purpose of optimizing HTML pages. JavaScript is a client-side programming language, which means that the JavaScript programs are executed in the web browser and interpreted by the web browser during runtime.

JavaScript is supposed to supplement HTML, not to replace it.

JavaScript is fundamentally different from the programming language Java. The similarity of the name is due to the intention to make a connection to the then very popular programming language for marketing reasons.

With JavaScript, you can, among other things, expand the HTML page by the following functions:

- Processing of keyboard entries
- Dynamic modification of the web page

Integration of JavaScript in HTML

There are several ways to integrate JavaScript commands in an HTML file:

- between the `<script>` and `</script>` tags
- for references
- as parameter of an HTML tag
- in an external JS file

It is very useful to note down JavaScript code in separate files, if you want to use the same JavaScript functions in several HTML files. As a result, you only need to enter the code once and you can reference it in several HTML files.

The syntax is as follows:

```
<script src="<Script>.js" type="text/javascript"> </script>
```

3.1.5 Automatic refreshing of the web page

Duration of loading speed of page

The refresh time of a web page depends on the contents of the page. The statistic parts and the dynamic parts (variables) have to be updated.

Time of variable transmission

The internal transmission time between CPU and the build-in web server depends on the number of the variables to be transferred. The size of the variables is virtually irrelevant. The transmission rate can be increased by a higher communication load at the expense of the program cycle time.

You can find an overview of the transmission time in the table below, depending on the number of variables and the configured communication load:

Table 3-3

Number of variables	Communication load [%]	Refresh time [s]
10	20	2.4
10	40	2.1
20	20	3.3
20	40	2.8
40	20	5.9
40	40	4.8

Note Delete variables from your HTML pages that are not used in order to increase the transmission rate. Commenting out variables is not sufficient.

Options

The setting for automatic refreshing, is only valid for standard web pages and not for user-defined web pages.

In principle, HTML is static and does not respond to modifications of the content. Therefore, if values change in the S7 program, it is useful to have the changed values displayed in the web browser.

There are several ways to refresh the display of the web page:

- Manual refreshing with "F5" button
- Automatic refreshing with a meta date in the head of the HTML file
- Automatic refreshing with JavaScript in the body of the HTML file

For the writing of variables in the CPU, a separate HTML page without automatic refresh function should be created. This prevents that entries that are not yet completed, are overwritten when the page is automatically refreshed.

Manual refreshing

With the "F5" function key (Internet Explorer: "View > Refresh"), the display in the web browser is refreshed manually.

Refreshing with HTML

With the following code line in the head of the HTML file, the display in the web browser is refreshed cyclically:

```
<meta http-equiv="refresh" content="10; URL=Example.htm">
```

The refresh cycle is entered in seconds. With "content="10;", the refresh cycle is 10 seconds. The actual refresh cycle depends on the amount of data of the page.

Enter the web page to be refreshed via "URL= ". In the application example, they are the "Overview.htm" and "Data.htm" files.

Refreshing with JavaScript

In the body of the HTML file, the following JavaScript refreshes the display in the web browser every 10 seconds:

```
<script type="text/javascript">
setInterval("document.location.reload()",1000);
</script>
```

3.2 Principles of standard web pages

Requirements

The web server has to be enabled in the properties of the PLC.

If you require safe access to the standard web pages, enable the "Permit access only with HTTPS" checkbox.

Automatic refreshing of the standard web pages is enabled. The refresh interval is preset to 10 s and cannot be changed.

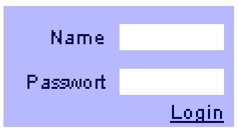
Access via HTTP or HTTPS

With the URL "http://ww.xx.yy.zz" or "https://ww.xx.yy.zz" you get access to the standard web pages. "ww.xx.yy.zz" corresponds to the IP address of the S7-1200 CPU.

HTTPS is used for the encryption and authentication of the communication between browser and web server. When the "Permit access only with HTTPS" checkbox is enabled, calling the web pages of S7 1200 CPU is only possible via HTTPS.

Login

You do not have to login to be able to access the web pages. To execute certain actions, such as, changing the operating state of the CPU or for write access, the user has to be logged on as "admin". The input fields for login can be found in the top left corner of each standard web page.



If you login as "admin" user, you have to enter the user name and the password there.

Name: admin.

Password: no or configured CPU password (for password-protected CPU).

The configuration of the password is described in chapter 5.6. Setting CPU password.

Standard web pages of the SIMATIC S7-1200

The web server of the S7-1200 already offers plenty of information regarding the respective CPU via integrated standard web pages. These standard web pages are listed individually in the table:

Table 3-4

Designation	Content
intro	Introductory page for the standard web pages
Start Page	The start page provides an overview of general information of the CPU, the CPU name, the CPU type and basic information on the current operating state.
Identification	Display of the static identification information, such as serial, order and version number
Diagnostic Buffer	Display of the diagnostic buffer contents with the latest entries first.
Module Information	Display whether the components of a station are in order, whether, for example, maintenance requests are pending or components are not available.
Communication	Display of the communication connections for open communication (OUC), display of resources as well as address parameters.
Variable Status	Display of the status of operands of the user program for watching and changing the values.
Data Logs	Data archive in CSV format for transfer to the hard disk of the PGs. The data archives are created with data log instructions in the user program and filled with data.
User Pages	The user web pages provide a list of web pages with customer-specific web applications.

3.3 Principles of user-defined web pages

The following chapters provide basic knowledge of user-defined web pages in relation to the application.

Context-related information can be found in the online help of SIMATIC STEP 7 V11 SP2 and to the “WWW” (SFC 99) instruction.

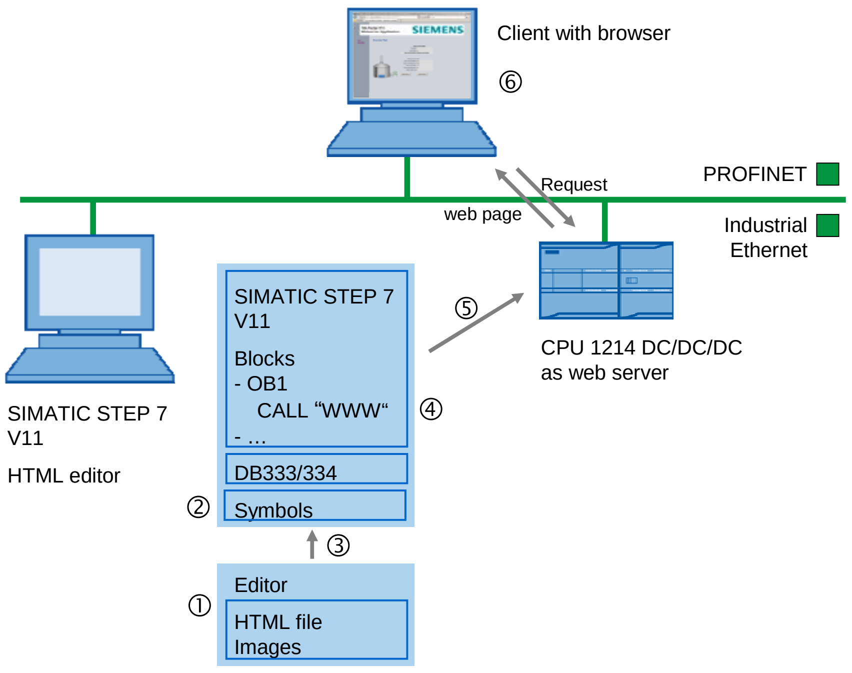
Advantages

The creation of a user-defined web page is advantageous if no permanent HMI system is required, but diagnostic information and visualizations are occasionally needed. Since standard web technologies are used, no additional visualization hardware and software is required.

A solution with AWP is reasonable for **simple applications** and the web page can be designed individually according to your requirements.

3.3.1 Creating user-defined web pages

Figure 3-1



Procedure

Table 3-5

No.	Instructions
1.	Create the HTML file for the CPU with an HTML editor. The entire web application consists of individual source files, for example, *.html, *.png, *.js, *.css, etc.. To be able to access CPU variables, a corresponding command syntax (AWP commands) is provided.
2.	Assign a symbolic name in STEP 7 to variables which you want to use on the web page.
3.	Generate data blocks (Web Control DB and fragment DBs) with STEP 7 from the source files. The DB numbers can be freely configured (default: DB 333 and from DB334). The DBs are stored under "Program blocks > System blocks > Web server" in the Project tree. These data blocks consist of a control data block that controls the display of the web pages and one or several data block fragments with the compiled web pages.
4.	With STEP 7, you create an S7 program. For the synchronization between user program and web server but also for the initialization you have to call the WWW (SFC 99) instruction in the user program.
5.	Transfer all blocks to the CPU with STEP 7.
6.	Open a web browser and enter the URL "http://ww.xx.yy.zz" or "https://ww.xx.yy.zz". "ww.xx.yy.zz" corresponds to the IP address of the S7-1200 CPU. The web browser requests the web page of the CPU via the http protocol; the CPU provides the web page as web server.

Access to the web server of the CPU is possible independently of the configuration computer; every output device with an integrated web browser and access to the PN interface of the CPU can display the web page.

To be able to get write access to the web page, you have to be logged on.

3.3.2 Blocks required for user-defined web pages

WWW (SFC99)

With the help of the “WWW” (SFC99) instruction, the CPU interprets the data blocks and can use them as user-defined web pages.

Web control DB and fragment DBs

The basis of the web page designed by you is an HTML file (or several connected HTML files with images):

To enable the CPU to interpret the HTML file, it is stored in data blocks together with other required files. Use STEP 7 for this purpose:

The Web Control DB (default: DB333) contains the following:

- Status and control variables of the web page
- Communication status (e.g. whether a request from the web browser to the web server is pending)
- Error information

Additionally to the Web Control DB there are also fragment DBs starting by default with DB334. These DBs contain the coded web pages and media data (e.g. images).

All Web Control DBs are located in the “Program blocks > System blocks > Web server” folder.

The size of the user-defined web pages therefore also determines the size of the user program. The size of the user program, the data and the configuration is limited by the available load memory and the main memory of the CPU.

Note If you need to reduce the space for your user-defined web pages, remove some of the inserted images, where applicable.

Typical use of variables

In the table below you can find an overview for the use of variables:

Table 3-6

Representation	Function	Example	Information
`:="<Name>"`	Display CPU variable	`:="TankLevelMinimum":`	Chap. 3.4
`<!-- AWP_In_Variable Name ='"<Name>"' -->`	Configuration to be able to write a variable on the CPU with a separate "POST" method	`<!-- AWP_In_Variable Name='"OpenValve"' -->`	Chap. 3.5
`<!-- AWP_Enum_Ref Name='"<Name>"' Enum="<Variable>" -->`	Assignment of enumerations (texts) to the value of a variable	`<!-- AWP_Enum_Ref Name='"Alarm"' Enum="AlarmValue" -->`	Chap. 3.6

3.4 Displaying variables from the CPU on the web page

3.4.1 Interaction between web browser and CPU

Figure 3-2

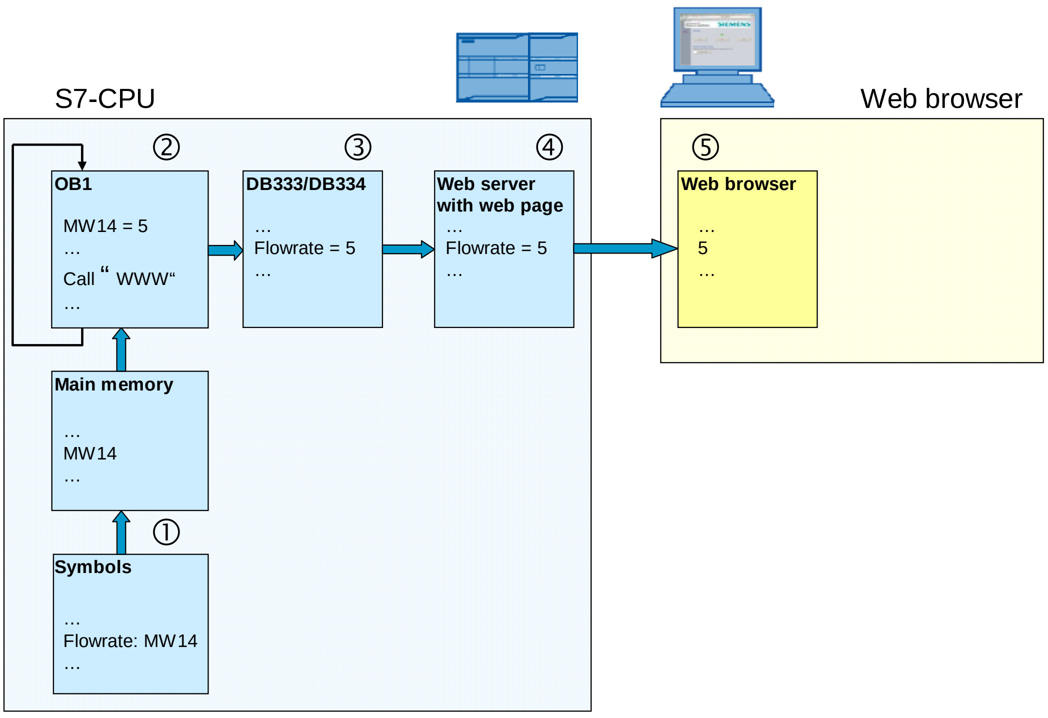


Table 3-7

No.	Description
1.	Variables which are displayed or written on the web page must have a symbolic name. A variable in a DB, for example, is accessed with `"DB_name".Variablen_name`.
2.	In the S7 program, the “WWW” (SFC99) instruction is called.
3.	By calling the “WWW” (SFC99) instruction, the Web Control DB (default: DB333) is initialized.
4.	The web server of the CPU converts the data with the help of the information in the Web Control DB (default: DB333) to a format (= web page) which can be interpreted by a web browser. The web page of the CPU is called in a web browser via the IP address of the CPU.
5.	With each request from the web browser, the web page is refreshed (manually or automatically). Information on the refreshing of a web page can be found in chapter 3.1.5 Automatic refreshing of the web page. A request to the web server can also be created with the "Post" method when writing a variable to the CPU. After having "sent" the web page, the entire web page is refreshed.

3.4.2 Requirements

To be able to display variables of the CPU on the web page, the following *prerequisites* apply:

Table 3-8

S7 program	HTML file
• Each variable must be assigned a symbolic name. The variable can only be displayed on the web page via symbolic names. • The “WWW” (SFC99) instruction must be called (if variables are pre-processed in the S7 program, cyclic call) • For variables the standard data types (“DTL” is not displayed), self-created PLC data types and structures are permitted.	• It is **not** necessary to declare variables via an AWP command in the HTML file.

3.4.3 Procedure

S7 program:

In the S-7 program no programming is required.

HTML file:

A variable can be displayed at any position on the HTML page. The syntax is as follows:

```
:="<Variable>":
```

Example of the "TankLevelMaximum" variable:

```
<p>:="TankLevelMaximum":</p>
```

The display of the variable is performed independent of the data type. Refreshing the variable is described in chapter 3.1.5 Automatic refreshing of the web page.

3.5 Writing variables on the CPU with the help of the web page

3.5.1 Interaction between web browser and CPU

Figure 3-3

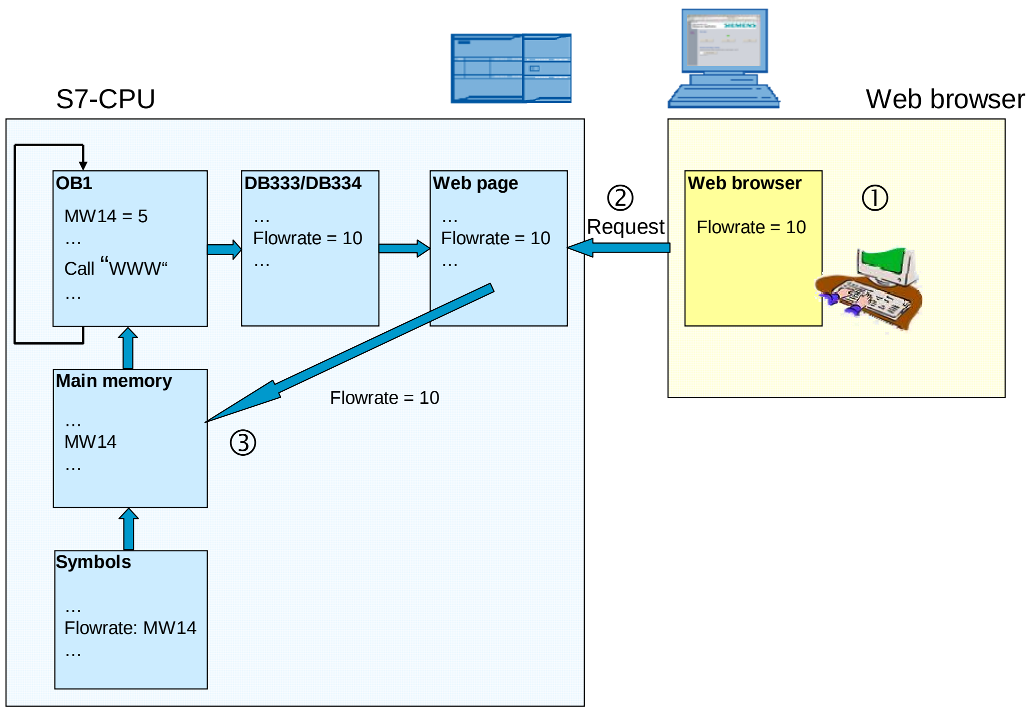


Table 3-9

No.	Description
1	Via the web page, the user changes the "Flowrate" variable to the value "10".
2	The web browser reports a request ("POST" method).
3	The S7 program accepts the changed "Flowrate" variable, the display in the web browser is refreshed, and the new values are displayed.

3.5.2 Requirements

To be able to write variables on the CPU via the web page, the following prerequisites apply:

Table 3-10

S7 program	HTML file
• Each variable must be assigned a symbolic name. A variable can only be addressed via symbolic names. • The “WWW” (SFC99) instruction has to be called cyclically. • For variables the standard data types (apart from “DTL”), self-created PLC data types and structures are permitted.	• Variables must be declared via the AWP command `<!-- AWP_In_Variable … -->` in the HTML file. • The variables must be transferred to the CPU (e.g. POST method in the HTML file).

3.5.3 Procedure

S7 program:

The “WWW” (SFC99) instruction has to be called cyclically.

HTML file:

The AWP command via which variables can be written in the CPU is as follows:

```
<!-- AWP_In_Variable Name='"Variable"' -->
```

Example of how to write the "Flowrate" variable:

```
<!-- AWP_In_Variable Name='"Flowrate"' -->
```

The AWP command typically stands at the beginning of the HTML file.

Transferring the variables from the web browser

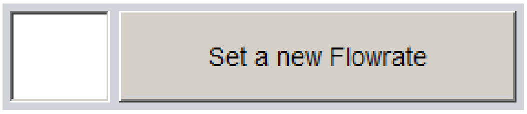
When calling the form, the POST method is selected for transferring the data from the web browser to the web server. The form consists of two units:

- A field for entering the value:
  The field is named via a variable and designates the variable from the AWP command, e.g.
  `<!-- AWP_In_Variable Name='"Flowrate"' -->`.
- A button with which the entry of the value is confirmed.

Via "submit", the form data is transferred.

Example:

Appearance on the web page:



Code:

```
<form method="post" action="" onsubmit=”return check();”>
  <input type="text" name='"Flowrate"' size="2">
  <input type="submit" value="Set a new Flowrate">
</form>
```

3.6 Linking variables with texts in the HTML file

Occasionally, it makes sense on a web page to output messages directly as a text and not as a variable. For this purpose, STEP 7 provides enumerations. With an enumeration, you can link values with concrete texts. These texts can be created in one or several languages. Our application contains single-language text messages.

The following graphic illustrates the interaction between web browser and CPU:

Figure 3-4

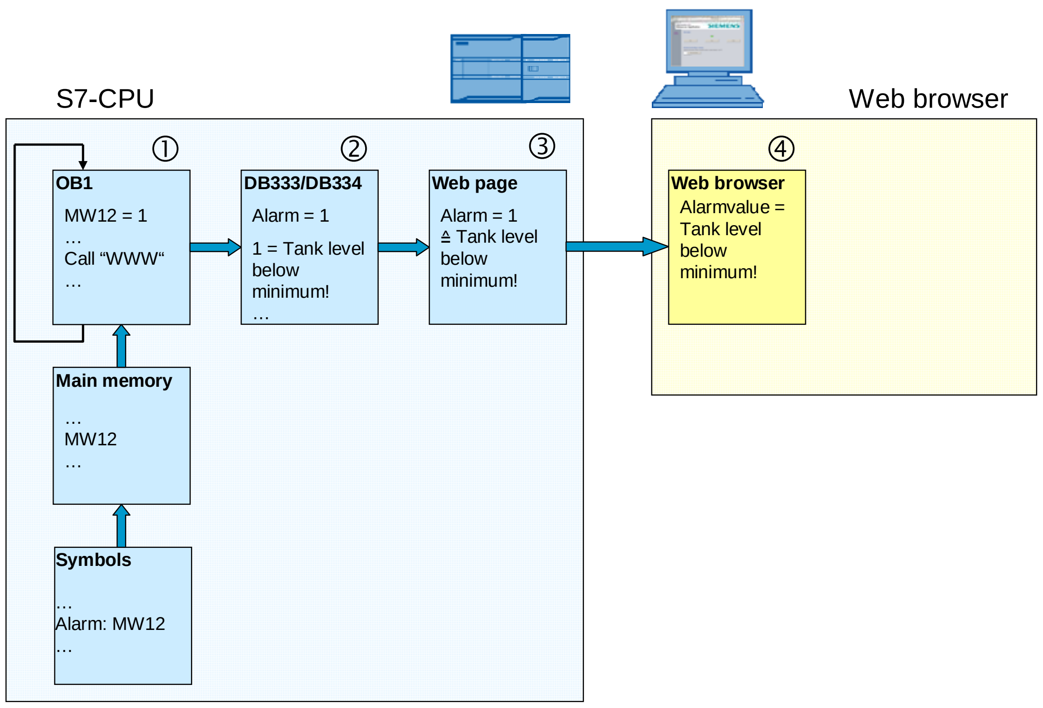


Table 3-11

No.	Description
1.	The S7 program calls the "WWW" (SFC99) instruction and sets the value of MW12 ("Alarm") to "1".
2.	Due to the cyclic calling of the "WWW" (SFC99) instruction, the "Alarm" variable in DB333/334 is also refreshed.
3.	The web server links the "Alarm" value with the related text.
4.	In the web browser, the related text is output instead of the "Alarm" value.

3.6.1 Requirements

To output indications as text, the following prerequisites apply:

Table 3-12

S7 program	HTML file
• Each variable must be assigned a symbolic name. A variable can only be addressed via symbolic names. • The “WWW” (SFC99) instruction must be called (if variables are pre-processed in the S7 program, cyclic call) • For variables, all numerical data types are approved.	• It is not necessary to declare variables via an AWP command in the HTML file, because they are only read but not written. • All language-dependent files incl. the HTML file must be stored in the same directory.

3.6.2 Procedure

Creating ENUM TYPE

The AWP command, via which ENUM types are defined, is:

```
<!-- AWP_Enum_Def Name= ="<Name Enum type>"
Values='0:"<Text_1>", 1:"<Text_2>", ... , x:"<Text_x>"' -->
```

Example for the "AlarmValue" ENUM type:

```
<!-- AWP_Enum_Def Name="AlarmValue" Values='0:"Tank empty!",
1:"Tank level below minimum!", 2:"Tank level between minimum
and midth!", 3:"Tank level between midth and maximum!",
4:"Tank level over maximum!", 5:"Tank level overflow!"' -->
```

Typically the AWP command is at the beginning of the HTML file or in a separate HTML file.

Assigning ENUM TYPE

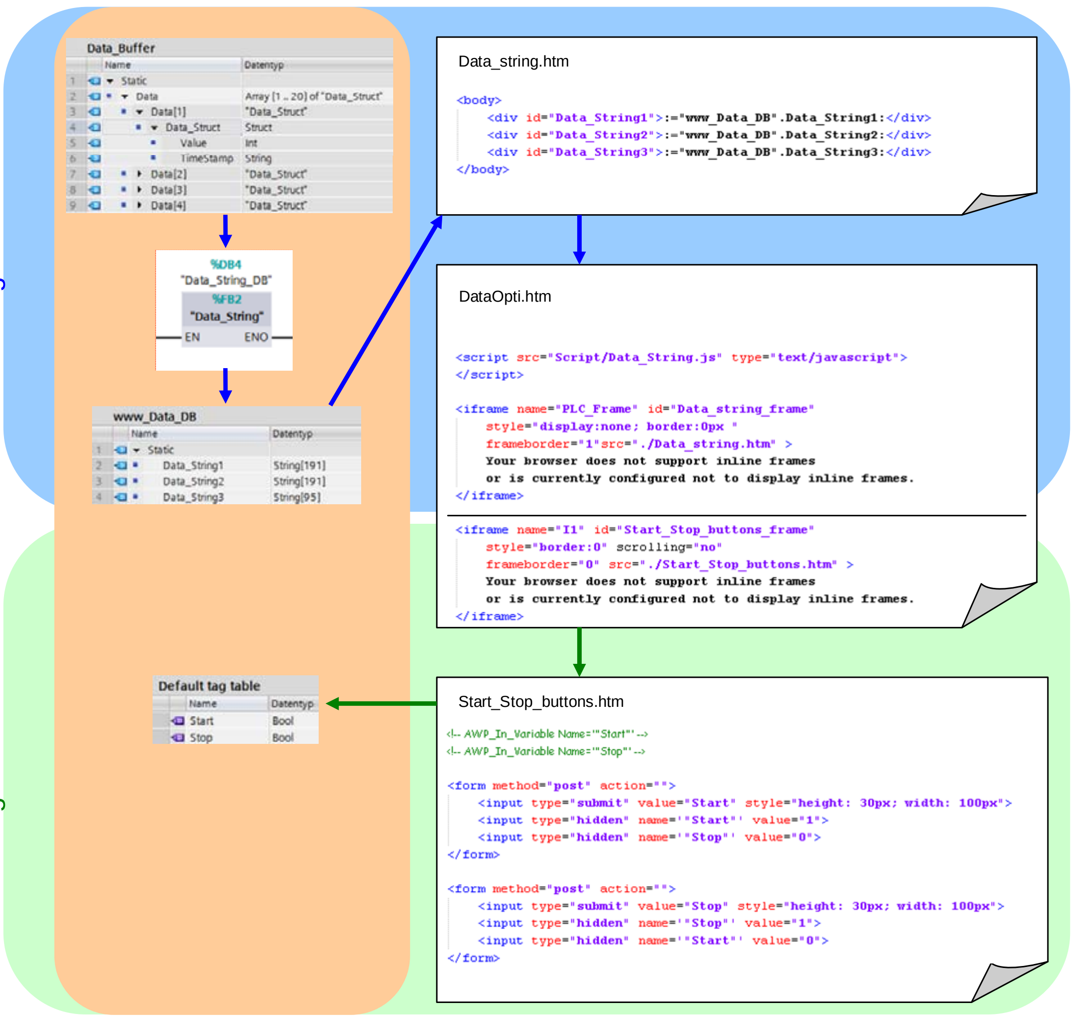
The syntax for the displaying of texts instead of the value is as follows, e.g. for the "Alarm" variable:

```
<!-- AWP_Enum_Ref Name='"Alarm"' Enum="AlarmValue" -->
:="Alarm":
```

3.7 Creating time-optimized HTML pages (optional)

Creating time-optimized HTML pages is not described in the manuals of the S7-1200. The concept is displayed in the figure below.

Figure 3-5 Reading/writing variable



Data_Buffer

	Name	Datentyp
1	Static	
2	Data	Array [1 .. 20] of "Data_Struct"
3	Data[1]	"Data_Struct"
4	Data_Struct	Struct
5	Value	Int
6	TimeStamp	String
7	Data[2]	"Data_Struct"
8	Data[3]	"Data_Struct"
9	Data[4]	"Data_Struct"

www_Data_DB

	Name	Datentyp
1	Static	
2	Data_String1	String[191]
3	Data_String2	String[191]
4	Data_String3	String[95]

Default tag table

Name	Datentyp
Start	Bool
Stop	Bool

Data_string.htm

```
<body>
    <div id="Data_String1">:="www_Data_DB".Data_String1:</div>
    <div id="Data_String2">:="www_Data_DB".Data_String2:</div>
    <div id="Data_String3">:="www_Data_DB".Data_String3:</div>
</body>
```

DataOpti.htm

```
<script src="Script/Data_String.js" type="text/javascript">
</script>

<iframe name="PLC_Frame" id="Data_string_frame"
    style="display:none; border:0px "
    frameborder="1"src="./Data_string.htm" >
    Your browser does not support inline frames
    or is currently configured not to display inline frames.
</iframe>
```

```
<iframe name="I1" id="Start_Stop_buttons_frame"
    style="border:0" scrolling="no"
    frameborder="0" src="./Start_Stop_buttons.htm" >
    Your browser does not support inline frames
    or is currently configured not to display inline frames.
</iframe>
```

Start_Stop_buttons.htm

```
<!-- AWP_In_Variable Name='"Start"' -->
<!-- AWP_In_Variable Name='"Stop"' -->

<form method="post" action="">
    <input type="submit" value="Start" style="height: 30px; width: 100px">
    <input type="hidden" name='"Start"' value="1">
    <input type="hidden" name='"Stop"' value="0">
</form>

<form method="post" action="">
    <input type="submit" value="Stop" style="height: 30px; width: 100px">
    <input type="hidden" name='"Stop"' value="1">
    <input type="hidden" name='"Start"' value="0">
</form>
```

Table 3-13

No.	Description
1.	There is a user-defined HTML page (below called main page) to be displayed. In this application this is the “DataOpti.htm” HTML page. It is not refreshed. Therefore your code does not contain the respective instructions within the meta data.
2.	The main page does <u>not</u> contain variables and therefore <u>no</u> direct access to the control variables. All control variables to be read or to be written are packed in additional HTML files (here “Data_string.htm” and “Start_Stop_buttons.htm“), which on their part are embedded as frames (iframe) in the main page.

No.	Description
	Therefore there is no need from the side of the user to refresh the main page periodically and there is also no system-related page refreshing. Only the individual frames are refreshed.
3.	Another important step is minimizing the number of control variables to be read. In the user program of the controller (here FB "Data_String") the individual control variables are separated by commas, written into an output string that is then transferred as a single variable to the web server which as a result achieves an enormous time saving. The string is inserted as output variable (see chapter 3.4 Displaying variables from the CPU on the web page) in a separate HTML file (here "Data_string.htm"), which appears as an embedded frame (iframe, see table point 2) in the main page. Here, this string is only provided for further processing by a JavaScript (here "Data_String.js"). The display:none CSS property, noted for the frame, prevents its display on the main page.
4.	For input variables (see chapter 3.5 Writing variables on the CPU with the help of the web page) forms (`form`) are inserted in separate HTML files (here "Start_Stop_buttons.htm"). The HTML files are embedded as frames (iframe, see table point 2) in the main page. Several forms can be written in a HTML file.
5.	The main page provides a reference to a JavaScript file (here "Data_String.js"), that performs the following tasks cyclically (in the following example every 500ms): – Splitting the output strings (see table point 3) into individual variables with the `split (",")` method which are subsequently stored in an `ss_values` output array. – Providing the output elements (here table cells) of the main page with the control variables, which are provided as elements of the `ss_values` output array. Accessing the output elements is performed with the `getElementById` method. – Refreshing the frames (see table point 3) that contain the output string with the `reload` () method.

Figure 3-6 Extract from "Data_String.js" JavaScript

```
setTimeout("init_plc()",1500);

function init_plc()
{
    setInterval("plc_update()",500);
}

function plc_update()
{
    var iframe = document.getElementById('Data_string_frame');
    if (iframe.readyState == 'complete')
    {
        var Data_string1 = window.frames['Data_string_frame'].document.getElementById('Data_String1').innerHTML;
        var ss_values = Data_string1.split(",");

        document.getElementById('Table1').rows[1].cells[0].innerHTML =  ss_values[0];
        document.getElementById('Table1').rows[1].cells[1].innerHTML =  ss_values[1];
        ...

        parent.frames['Data_string_frame'].window.location.reload();
    }
 }
```

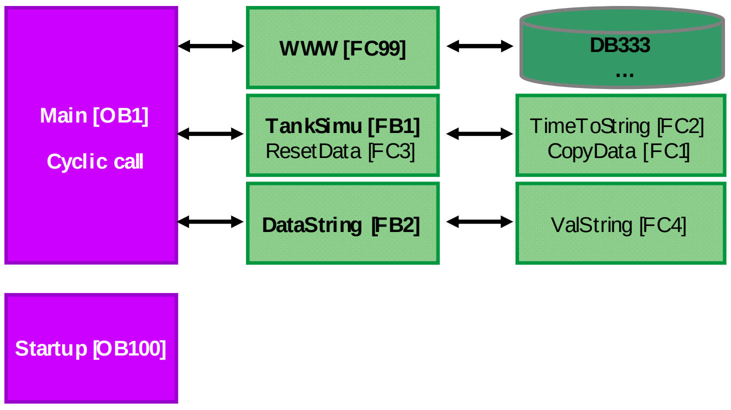
4 Functional Mechanisms of this Application

4.1 Functional principle of the S7 program

The S7 program of this application only serves for representing individual functional principles of STEP 7 by way of example.

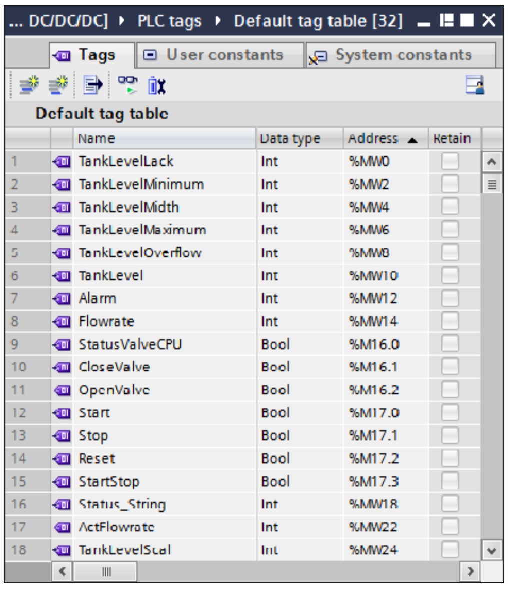
The call structure in the S7 program looks as follows:

Figure 4-1



The following symbols and variables are used in the ***"Data_Buffer"*** data block:

Figure 4-2

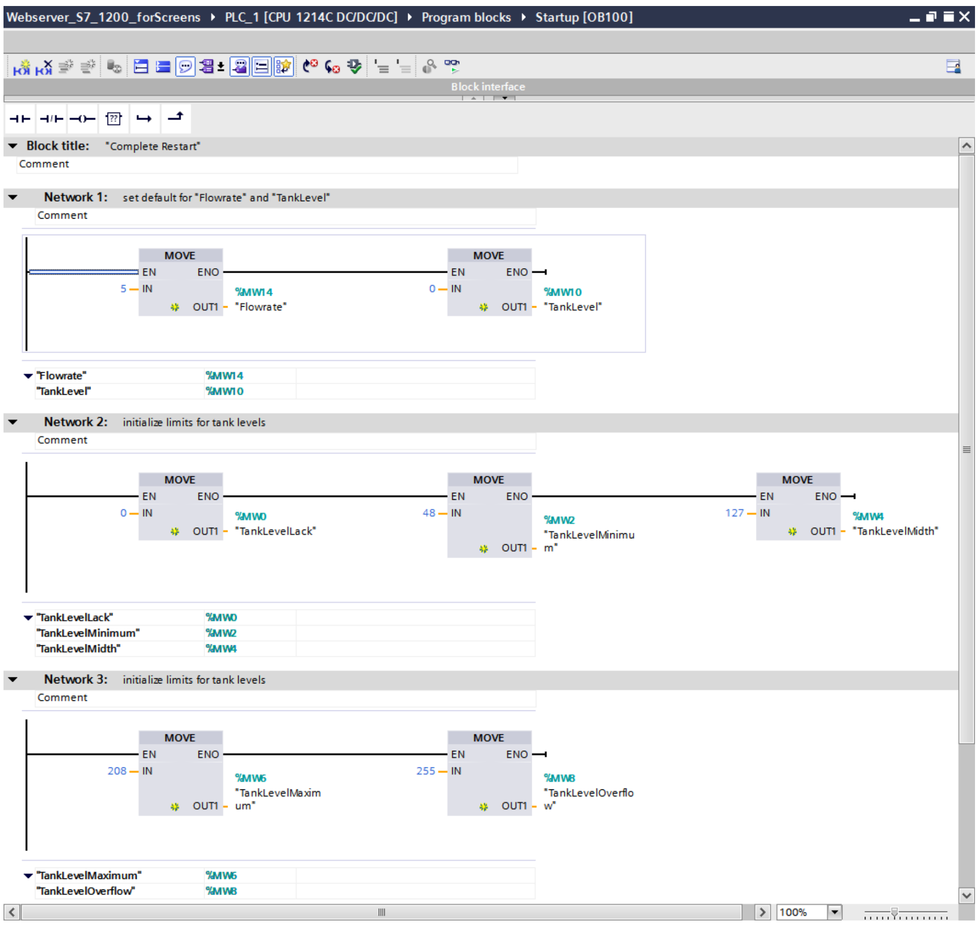


	Name	Data type	Address	Retain
1	TankLevelLack	Int	%MW0	☐
2	TankLevelMinimum	Int	%MW2	☐
3	TankLevelMidth	Int	%MW4	☐
4	TankLevelMaximum	Int	%MW6	☐
5	TankLevelOverflow	Int	%MW8	☐
6	TankLevel	Int	%MW10	☐
7	Alarm	Int	%MW12	☐
8	Flowrate	Int	%MW14	☐
9	StatusValveCPU	Bool	%M16.0	☐
10	CloseValve	Bool	%M16.1	☐
11	OpenValve	Bool	%M16.2	☐
12	Start	Bool	%M17.0	☐
13	Stop	Bool	%M17.1	☐
14	Reset	Bool	%M17.2	☐
15	StartStop	Bool	%M17.3	☐
16	Status_String	Int	%MW18	☐
17	ActFlowrate	Int	%MW22	☐
18	TankLevelScal	Int	%MW24	☐

4.1.1 Startup (OB100)

In the ***“Startup”*** (OB100) OB a start value for the flow rate ***"Flowrate"*** and the limit values for the variables are stored.

Figure 4-3



Webserver_S7_1200_forScreens ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Startup [OB100]

Block interface

Block title: "Complete Restart"
Comment

Network 1: set default for "Flowrate" and "TankLevel"
Comment

```
MOVE  EN ENO   5 — IN   OUT1 — %MW14 "Flowrate"
MOVE  EN ENO   0 — IN   OUT1 — %MW10 "TankLevel"
```

"Flowrate"	%MW14
"TankLevel"	%MW10

Network 2: initialize limits for tank levels
Comment

```
MOVE  EN ENO   0 — IN   OUT1 — %MW0 "TankLevelLack"
MOVE  EN ENO  48 — IN   OUT1 — %MW2 "TankLevelMinimum"
MOVE  EN ENO 127 — IN   OUT1 — %MW4 "TankLevelMidth"
```

"TankLevelLack"	%MW0
"TankLevelMinimum"	%MW2
"TankLevelMidth"	%MW4

Network 3: initialize limits for tank levels
Comment

```
MOVE  EN ENO 208 — IN   OUT1 — %MW6 "TankLevelMaximum"
MOVE  EN ENO 255 — IN   OUT1 — %MW8 "TankLevelOverflow"
```

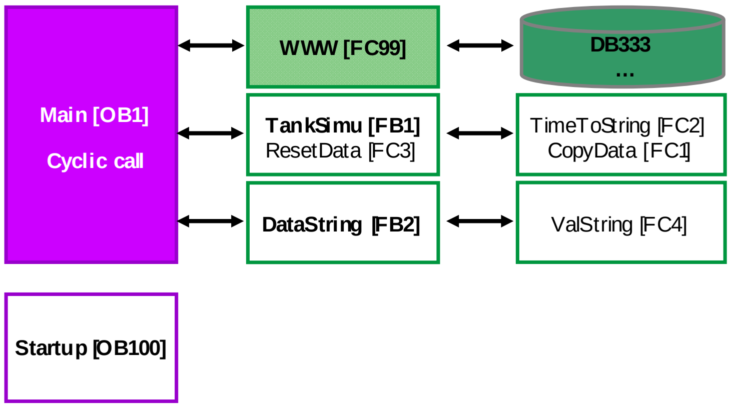
"TankLevelMaximum"	%MW6
"TankLevelOverflow"	%MW8

100%

4.1.2 Main (OB1)

In OB ***"Main"*** (OB1) the status of DB333 is polled cyclically to be able to recognize a request from the web browser. The cause for a request is that a variable changed by the user is to be transferred from the web browser to the web server.

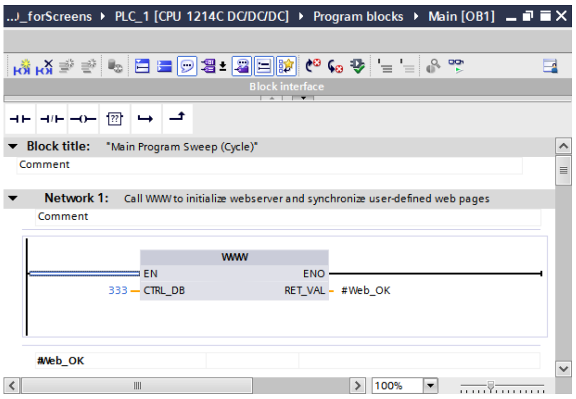
Figure 4-4



Synchronizing user-defined web pages

The "WWW" (SFC99) instruction initializes the web server of the CPU. The error information is output via "RET_VAL".

Figure 4-5


...)_forScreens ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Main [OB1]

Block interface

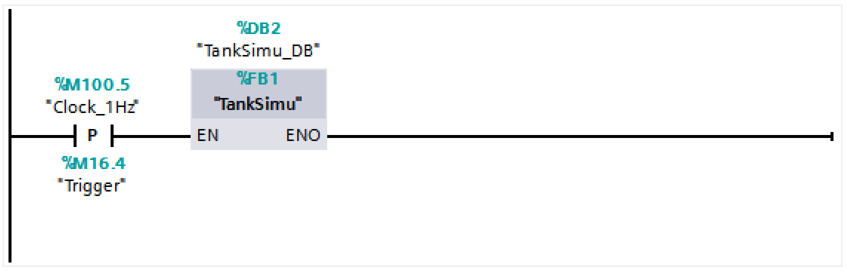
Block title: "Main Program Sweep (Cycle)"
Comment

Network 1: Call WWW to initialize webserver and synchronize user-defined web pages
Comment

Calling the tank simulation

To ensure that filling or emptying of the tank does not happen too quickly, the ***“TankSimu”*** function block is called in OB1 only once per second.

Figure 4-6



Polling the "Start" or "Stop" and "Reset" buttons

The status of the ***"Start"*** and ***"Stop"*** buttons are polled by the web page. If one of the buttons has been clicked, the status is stored in the ***"StartStop"*** PLC variable.

In addition the status of the ***"Reset"*** button is polled. By clicking ***"Reset"*** a defined initial position is created with the ***“ResetData”*** function.

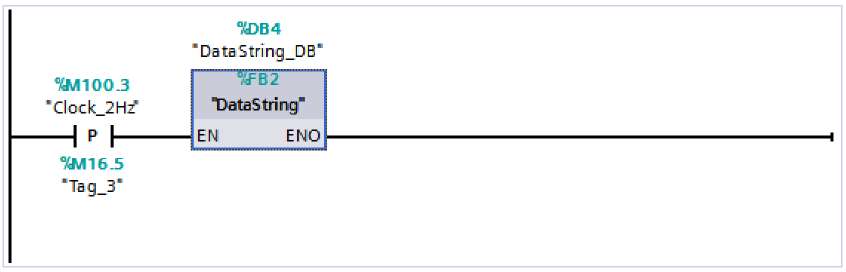
Polling of the "OpenValve" or "CloseValve" buttons

Additionally, the status of the ***"OpenValve"*** and ***"CloseValve"*** buttons is polled by the web page. If one of the buttons has been clicked, the status is stored in the ***"StatusValveCPU"*** PLC variable.

Calling the *“DataString“* function block

Refreshing the data string is called twice every second. This corresponds to the refresh time in Javascript.

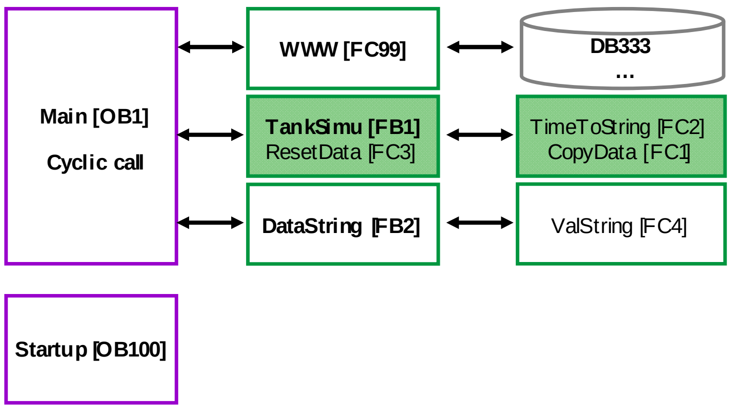
Figure 4-7



4.1.3 TankSimu (FB1)

Functionality of the FB1

Figure 4-8



In FB1, the filling or emptying of a tank is simulated, dependent on the flow rate and the valve position.

The block is only run once every second.

The user can define the flow rate via the ***"Flowrate"*** variable on the web page. The tank filling level is increased or reduced with the flow rate when calling FB1. The current filling level is stored in the ***"TankLevel"*** PLC variable.

Via the two PLC variables ***"OpenValve"*** and ***"CloseValve"***, the valve position is read in and stored in the CPU in the ***"StatusValveCPU"*** PLC variables.

Dependent on the tank filling level, the following heights are displayed:

- Tank has been fully drained (TankLevelLack)
- Tank filling level is at minimum (TankLevelMinimum)
- Tank filling level is 50 % (TankLevelMidth)
- Tank filling level is at maximum (TankLevelMaximum)
- Tank is overflowing (TankLevelOverflow)

Via the ***"Alarm"*** variable, the tank filling level is output in clear text (also as enumeration)

StartStop status

Only when the ***"StartStop"*** is set, the tank filling level changes and values are entered in the ring buffer.

Valve status

Via the ***"StatusVentilCPU"*** bit, the button pressed last (OpenValve or CloseValve) is memorized.

Dependent on this bit, the tank is either emptied or filled.

Filling the tank

The filling of the tank starts with a query whether the tank is already full.

If the tank is not full, the tank filling level is increased with the flow rate. The tank filling level is limited to the ***"TankLevelOverflow"*** value.

Emptying the tank

The emptying of the tank is similar to the filling of the tank. The tank filling level is reduced with the flow rate and is limited to 0.

Alarm status

Subsequently, the tank filling level is compared with the specifications for the limit values of the tank filling level.

Depending on the filling level reached, the values "0" to "5" are stored in the ***"Alarm"*** variable. With the value of the ***"Alarm"*** variable, HTML texts (enumerations) are stored, which display the filling level of the tank in clear text.

Storing data in the ring buffer

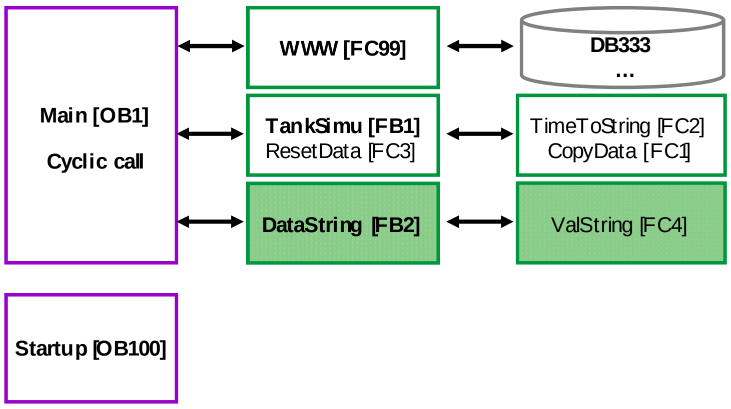
The current filling level is stored together with the time stamp (string) in a ring buffer of 20 value pairs and displayed.

With the ***“TimeToString”*** function a time stamp is created as string from the local time. With the ***“CopyData”*** function the value pairs are copied in the ring buffer.

4.1.4 DataString (FB2)

Functionality of the FB2

Figure 4-9



In FB2 the data pairs (time stamp, value) is written in the ring buffer, separated by commas and written in data strings. Since a string can only be 254 characters long, 3 data strings are required that are then transferred to the web server as individual variables.

The time stamp is already saved as string in the ring buffer. The filling level (value) is converted in a string with the ***"ValString"*** function.

The block is only run twice every second.

4.2 Functional principle of the HTML file

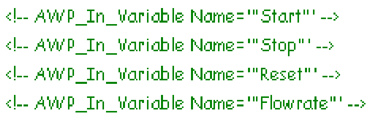
The following chapter provides a detailed explanation of the individual sections of the HTML file. For the creation of the HTML pages only fixed values are used for the position and size of the elements. This prevents the elements from moving and overlapping when the browser window is made smaller.

4.2.1 AWP commands

Basics

AWP commands are inserted as HTML comments in HTML files. AWP commands can be located at any position in the HTML file. However, for reasons of clarity it is appropriate to list the central AWP commands at the beginning of the HTML file.

Figure 4-10



```
<!-- AWP_In_Variable Name='"Start"' -->
<!-- AWP_In_Variable Name='"Stop"' -->
<!-- AWP_In_Variable Name='"Reset"' -->
<!-- AWP_In_Variable Name='"Flowrate"' -->
```

Explanations

Table 4-1

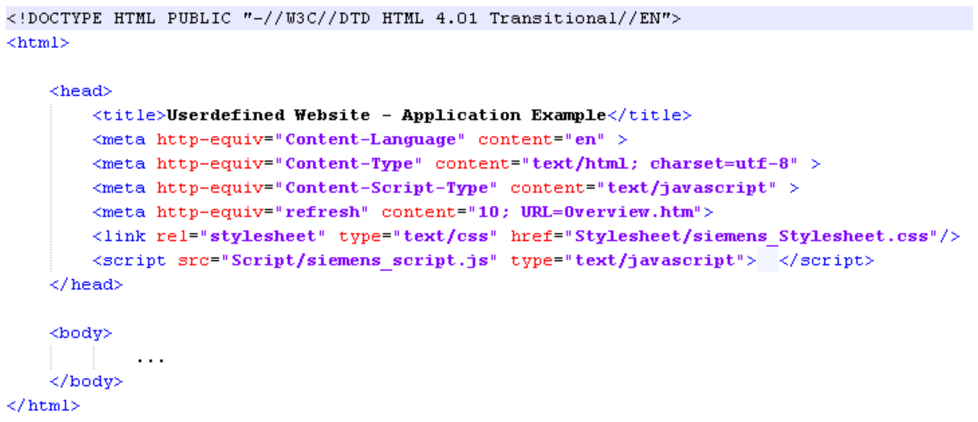
Code	Explanation
`<!-- AWP_In_Variable Name='"Start"' -->`	All variables transferred to the CPU must be identified as AWP_In_Variable. Note: Keep in mind that the quotation marks are nested. The variable is written between quotation marks and framed by an inverted comma (' " … " ').
`<!-- AWP_Enum_Def Name="AlarmValue" Values='0:"Tank empty!", 1:"Tank level below minimum!", 2:"Tank level between minimum and midth!", 3:"Tank level between midth and maximum!", 4:"Tank level over maximum!", 5:"Tank level overflow!"' -->`	ENUM types are defined with AWP_Enum_Def.
`<!-- AWP_Enum_Ref Name='"Alarm"' Enum="AlarmValue" -->:="Alarm":`	The ENUM types are assigned to variables with AWP_Enum_Ref.

4.2.2 Information on doctype and head of the HTML file

Basics

The following information must be contained in every HTML file so that it is HTML compliant. The only exception is the "`<meta http-equiv="refresh" …>`" tag: If you refrain from automatically refreshing the page and work with "F5" instead, you can omit this tag.

Figure 4-11



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>

    <head>
        <title>Userdefined Website - Application Example</title>
        <meta http-equiv="Content-Language" content="en" >
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" >
        <meta http-equiv="Content-Script-Type" content="text/javascript" >
        <meta http-equiv="refresh" content="10; URL=Overview.htm">
        <link rel="stylesheet" type="text/css" href="Stylesheet/siemens_Stylesheet.css"/>
        <script src="Script/siemens_script.js" type="text/javascript">  </script>
    </head>

    <body>
            ...
    </body>
</html>
```

Explanations

Table 4-2

Code	Explanation
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`	Specifying the HTML document type: the document type is HTML in the language version V4.01 in the "transitional" variant. The "EN" language code refers to the language of the tags, i.e. English. The document type always stands before the "`<html>`" tag.
`<html> … </html>`	Contains the HTML content
`<title>Userdefined Website – Application Example</title>`	Title of the web page which will later be displayed in the head of the web browser.
`<meta http-equiv="Content-Language" content="en" >`	Language of the file content
`<meta http-equiv="Content-Type" content="text/html; charset=utf-8" >`	With "`content="text/html`", the MIME type is specified, followed by the used UTF-8 character set.
`<meta http-equiv="refresh" content="10; URL=Overview.htm">`	Optional meta date: with this command, the web page is refreshed every 10 seconds. Especially for process monitoring it is appropriate to have the web page

Code	Explanation
	refreshed cyclically. For pages with input fields, cyclic refreshing may cause problems. Further information on the refreshing of the web page can be found in chapter 3.1.5 Automatic refreshing of the web page.
`<link rel="stylesheet" type="text/css" href="Stylesheet/siemens_Stylesheet.css">`	Via `<link...>`, a CSS file is referenced which contains all information on the optical design of the web page, e.g. white background color, etc..
`<script src="Script/siemens_script.js" type="text/javascript"> </script>`	The area for scripts (e.g. JavaScript) is defined between `<script...>` and `</script>`. Note down the instructions within the area in the script language or integrate a separate file with your script with `src`.
`<body> …</body>`	Contains the text body

4.2.3 Displaying of areas

Basics

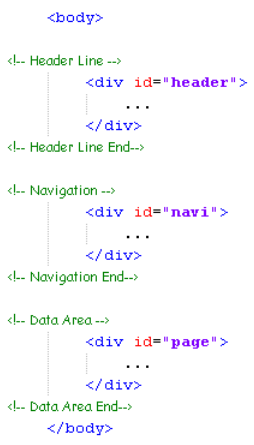
Three areas are used in the HTML file:

- Header area (header)
- Navigation bar (navi)
- Data area (page)

Explanations

The figure below shows the areas in the HTML file:

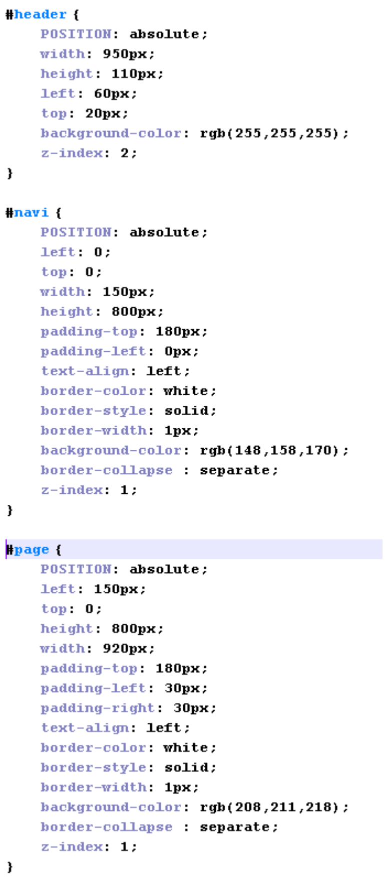
Figure 4-12



```
    <body>

<!-- Header Line -->
        <div id="header">
            ...
        </div>
<!-- Header Line End-->

<!-- Navigation -->
        <div id="navi">
            ...
        </div>
<!-- Navigation End-->

<!-- Data Area -->
        <div id="page">
            ...
        </div>
<!-- Data Area End-->
    </body>
```

The formatting of the areas is centrally defined in a separate CSS file:

Figure 4-13



```
#header {
    POSITION: absolute;
    width: 950px;
    height: 110px;
    left: 60px;
    top: 20px;
    background-color: rgb(255,255,255);
    z-index: 2;
}

#navi {
    POSITION: absolute;
    left: 0;
    top: 0;
    width: 150px;
    height: 800px;
    padding-top: 180px;
    padding-left: 0px;
    text-align: left;
    border-color: white;
    border-style: solid;
    border-width: 1px;
    background-color: rgb(148,158,170);
    border-collapse : separate;
    z-index: 1;
}

#page {
    POSITION: absolute;
    left: 150px;
    top: 0;
    height: 800px;
    width: 920px;
    padding-top: 180px;
    padding-left: 30px;
    padding-right: 30px;
    text-align: left;
    border-color: white;
    border-style: solid;
    border-width: 1px;
    background-color: rgb(208,211,218);
    border-collapse : separate;
    z-index: 1;
}
```

Table 4-3

Code	Explanation

`#page {`
`   POSITION: absolute;`
`   left: 150px;`
`   top: 0;`
`   height: 800px;`
`   width: 920px;`
`   padding-top: 180px;`
`   padding-left: 30px;`
`   padding-right: 30px;`
`   text-align: left;`
`   border-color: white;`
`   border-style: solid;`
`   border-width: 1px;`
`   background-color:`
`rgb(208,211,218);`
`   border-collapse : separate;`
`   z-index: 1;`
`}`
	CSS formats have the following structure:
	Selector {Property: value }
	
	In our example, “page” is the selector with several declarations (property:value):
	
	More information on formatting of HTML elements can be found in chapter 3.1.3 Basics on Cascading Style Sheets (CSS).

4.2.4 Displaying of images

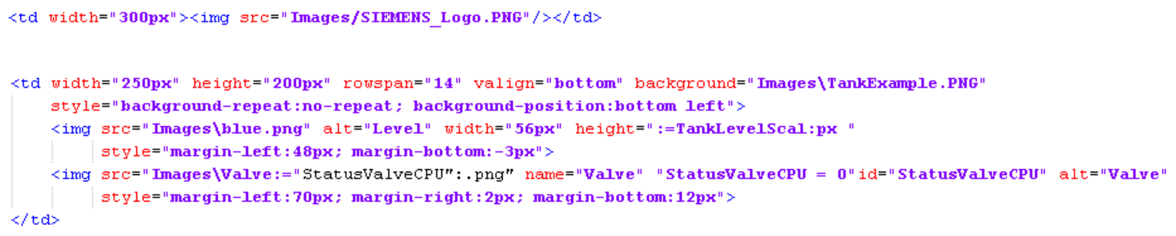
Basics

There are several images used in the HTML file:

- Static images
- Background image
- Image with variable height
- Dynamic image which is changed dependent on a status bit in the CPU.

Explanations

Figure 4-14



```
<td width="300px"><img src="Images/SIEMENS_Logo.PNG"/></td>


<td width="250px" height="200px" rowspan="14" valign="bottom" background="Images\TankExample.PNG"
    style="background-repeat:no-repeat; background-position:bottom left">
    <img src="Images\blue.png" alt="Level" width="56px" height=":=TankLevelScal:px "
         style="margin-left:48px; margin-bottom:-3px">
    <img src="Images\Valve:="StatusValveCPU":.png" name="Valve" "StatusValveCPU = 0"id="StatusValveCPU" alt="Valve"
         style="margin-left:70px; margin-right:2px; margin-bottom:12px">
</td>
```

Table 4-4

Code	Explanation
`<img src="Images/SIEMENS_Logo.PNG">`	Images are integrated via the "`img`" tag.
`background="Images\TankExample.PNG" style="background-repeat:no-repeat; background-position:bottom left"`	"`background`" specifies the background image with its properties.
`<img src="Images\blue.png" alt="Level" width="56px" height=":=TankLevelScal:px " style="margin-left:48px; margin-bottom:-3px">`	For images with variable height, such as, for example, level indicator, a "`TankLevelScal`" variable is specified with ":" and a unit of measure e.g. "`px`". instead of a value for "`height`". In this example, "`margin`" positions the level indicator in the background image.
`<img src="Images\Valve:="StatusValveCPU":.png" name="Valve" "StatusValveCPU = 0" id="StatusValveCPU" alt="Valve" style="margin-left:70px; margin-right:2px; margin-bottom:12px">`	This image depends on the "StatusValveCPU" variable. This variable can adopt the states "0" and "1". The stored images have the designation Valve0.png (valve closed) and Valve1.png (valve open). When the valve is closed, "StatusValveCPU" has the value 0: the call of the image is made up of: "Valve" + "0" + ".png" = Valve0.png With "alt", you specify a text which will be displayed if the image cannot be called by the web page. With "`margin`" the valve is positioned to the background image.

4.2.5 Creating a table with texts

Basics

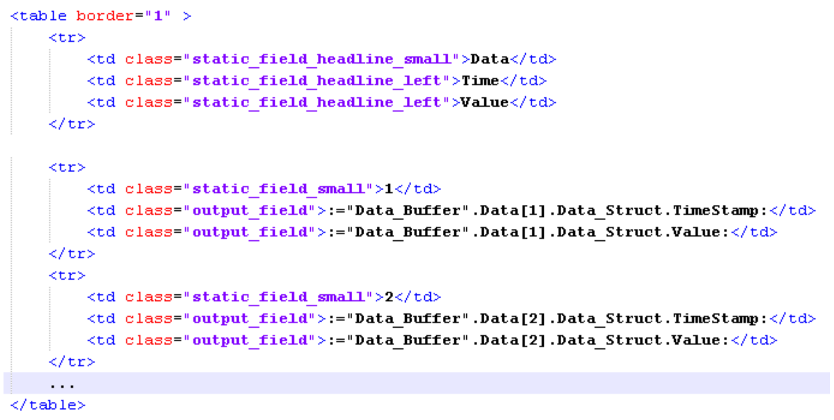
The use of a table is recommended to avoid that the contents of the web page are moved, depending on the size of the window.

Of course, you can also define a table centrally for your web page via CSS (Cascading Style Sheet).

Explanations

In the following figure, only the header and the first two lines of the table are shown for reasons of clarity.

Figure 4-15



```
<table border="1" >
    <tr>
        <td class="static_field_headline_small">Data</td>
        <td class="static_field_headline_left">Time</td>
        <td class="static_field_headline_left">Value</td>
    </tr>

    <tr>
        <td class="static_field_small">1</td>
        <td class="output_field">:="Data_Buffer".Data[1].Data_Struct.TimeStamp:</td>
        <td class="output_field">:="Data_Buffer".Data[1].Data_Struct.Value:</td>
    </tr>
    <tr>
        <td class="static_field_small">2</td>
        <td class="output_field">:="Data_Buffer".Data[2].Data_Struct.TimeStamp:</td>
        <td class="output_field">:="Data_Buffer".Data[2].Data_Struct.Value:</td>
    </tr>
    ...
</table>
```

Table 4-5

Code	Explanation

`<table border="1">`
…
`</table)`
	The stroke width (`border`) of the table is "1".
	Create a table without a frame (invisible table) with `border="0"`.

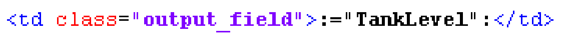
`<tr>`
`<td class="static_field_headline_small"> Data</td>`
`<td class="static_field_headline_left"> Time</td>`
`<td class="static_field_headline_left"> Value</td>`
`</tr>`
	`<tr>` stands for table row.
	The content of a cell stands between `<td>` (table data) and `</td>`.
	The formats e.g. `"static_field_headline_small"` of the table data are defined in the CSS file.
	`"class="<name>""` assigns the formats from the CSS file to the elements in the HTML file. This achieves a uniform appearance for all, e.g. tables.

4.2.6 Outputting CPU variables

Explanations

Variables of the CPU are always displayed via the symbol name:

Figure 4-16



```
<td class="output_field">:="TankLevel":</td>
```

Instead of ***"TankLevel"***, always the current value from the CPU is output on the web page.

Note

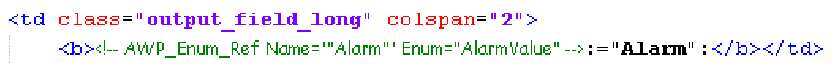
Since the variable is located in a table, the `"<td> · </td>"` tag is displayed additionally here. The central "`output_field`" format is defined in the CSS file.

4.2.7 Outputting texts via enumerations

Explanations

Via enumerations, texts can be allocated to the individual values of a CPU variable.

Figure 4-17



```
<td class="output_field_long" colspan="2">
    <b><!-- AWP_Enum_Ref Name='"Alarm"' Enum="AlarmValue" -->:="Alarm":</b></td>
```

Instead of the individual values of "Alarm", the previously assigned texts in HTML are output. These texts were stored as "AlarmValue" enum type. These texts are transferred to the web page via DB333.

Note

Since the enumeration is located in a table, the `"<td> · </td>"` tag is displayed additionally here. The central "`output_field_long`" format is defined in the CSS file. The result is output in bold, which is indicated by the `"<b> · </b>"` tag.

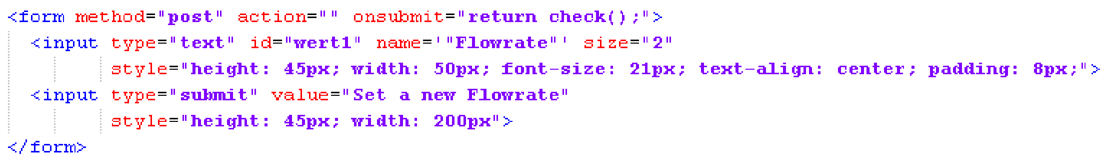
4.2.8 Setting variables in the CPU with value and button

Basics

To be able to transfer variables to the CPU via the web page, you have to work with forms and, for example, the "POST" method.

Explanations

Figure 4-18



```
<form method="post" action="" onsubmit="return check();">
  <input type="text" id="wert1" name='"Flowrate"' size="2"
          style="height: 45px; width: 50px; font-size: 21px; text-align: center; padding: 8px;">
  <input type="submit" value="Set a new Flowrate"
          style="height: 45px; width: 200px">
</form>
```

Table 4-6

Code	Explanation
`<form method="post" action="" onsubmit="return check();">` `<input type="text" id="wert1" name='"Flowrate"' size="2" style="height: 45px; width: 50px; font-size: 21px; text-align: center; padding: 8px;">` `<input type="submit" value="Set a new Flowrate" style="height: 45px; width: 200px">` `</form>`	Calling the form with the "`post`" method. Under "`action`", no details are required because with "`action`" the current page is called by default. With the called "`onsubmit`" event handler, the "`check()`" function is executed that is defines in the JS file. With a click on "`submit`", the function checks whether the input is in the range of 1 to 10. If this condition is met, the "`check()`" function reports back TRUE otherwise the return value is FALSE and an additional message is output. With `input type="text"`, an input field is linked, whose content is sent to the web server of the CPU with "`submit`" (only if check() = TRUE). "`submit`" is controlled via a button called "Set a new Flowrate".

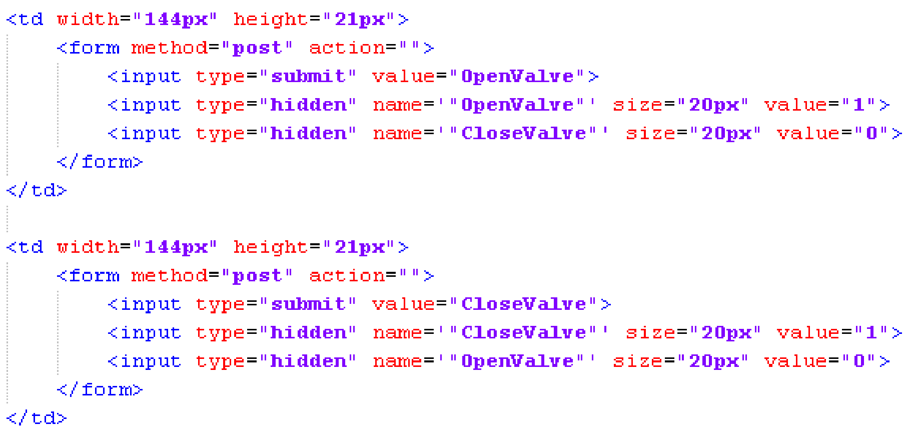
4.2.9 Setting variables in the CPU via button only

Basics

To assign variables in the CPU a predefined value, you have to work with a form, the "POST" method and a hidden value.

Explanations

Figure 4-19



```
<td width="144px" height="21px">
    <form method="post" action="">
        <input type="submit" value="OpenValve">
        <input type="hidden" name='"OpenValve"' size="20px" value="1">
        <input type="hidden" name='"CloseValve"' size="20px" value="0">
    </form>
</td>

<td width="144px" height="21px">
    <form method="post" action="">
        <input type="submit" value="CloseValve">
        <input type="hidden" name='"CloseValve"' size="20px" value="1">
        <input type="hidden" name='"OpenValve"' size="20px" value="0">
    </form>
</td>
```

Note

Since the buttons are located in a table, you can additionally see the "`<td> · </td>`" tags here.

Table 4-7

Code	Explanation

`<form method="post" action="">`
`<input type="submit" value="OpenValve">`
`<input type="hidden" name='"OpenValve"' size="20px" value="1">`
`<input type="hidden" name='"CloseValve"' size="20px" value="0">`
`</form>`
	Calling the form with the "`post`" method. Under "`action`", no details are required because with "`action`" the current page is called by default.
	With input `type="hidden`", the "OpenValve" variable is assigned the value 1, the "CloseValve" variable the value 0.
	With "`submit`", the values of the variables are sent to the web server of the CPU.

`<form method="post" action="">`
`<input type="submit" value="CloseValve">`
`<input type="hidden" name='"CloseValve"' size="20px" value="1">`
`<input type="hidden" name='"OpenValve"' size="20px" value="0">`
`</form>`
	Reverse action to the row above: Calling the form to assign the value 1 to "CloseValve" and the value 0 to "OpenValve".

5 Configuration and Settings

This chapter contains all information on how you can create and operate a web page for a CPU with PN interface for yourself. The CPU 1214C DC/DC/DC is used as an example in this chapter. All steps are presented by means of the completed example application.

If you only want to take the completed example application into operation, please continue reading in chapter 6 Installation.

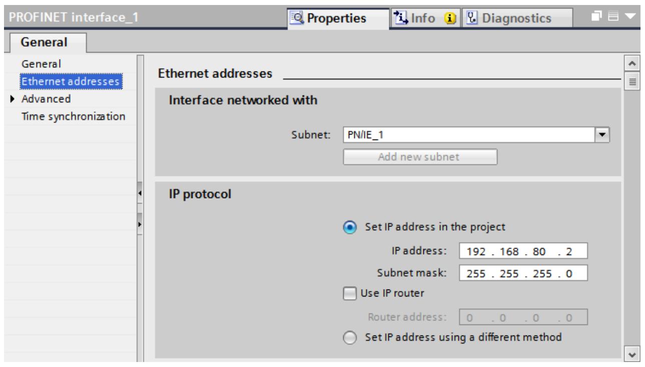
5.1 Procedure for creating a web page

The configuration and settings in STEP 7 and the writing of the HTML file are closely linked. The following procedure is recommendable for that:

1. Configuration of the S7-1200 (1214C DC/DC/DC)
2. Creating the variables in the variable table or DB
3. Creating the HTML files
4. Web server settings and generating of the data blocks
5. Setting CPU password
6. Creating, compiling and loading of the S7 program
7. Calling the web page with a web browser

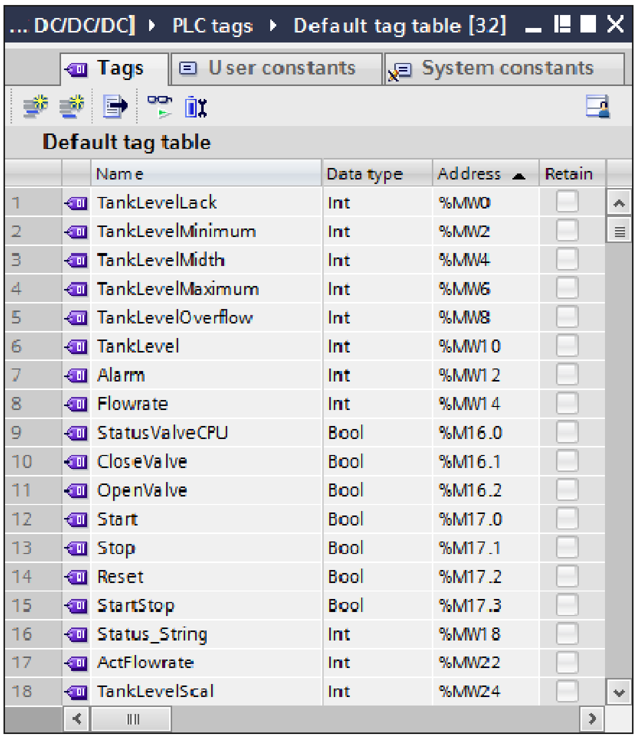
5.2 Configuration of the S7-1200 (1214C DC/DC/DC)

Table 5-1

No.	Action	Comment
1.	Start STEP 7 V11 and create a new project with the "Webserver_S7_1200" name via "Project > New…".	
2.	Insert a S7-1200 station via "Add new device“ >> "PLC > SIMATIC S7-1200 > CPU 1214C DC/DC/DC > 6ES7 214-1AE30-0XB0 V2.2". The device view of the PLC opens.	
3.	Click the "Add new subnet" button in the properties of the Ethernet interface.	 PROFINET interface_1 — Ethernet addresses — Interface networked with — Subnet: PN/IE_1 — Add new subnet — IP protocol — Set IP address in the project — IP address: 192 . 168 . 80 . 2 — Subnet mask: 255 . 255 . 255 . 0 — Use IP router — Router address: 0 . 0 . 0 . 0 — Set IP address using a different method
4.	Assign the IP address of CPU to the Ethernet interface. Via this IP address, you will later access the web page of the CPU with your web browser.	

5.3 Creating the variables in the variable table or DB

Table 5-2

No.	Action	Comment
1.	Double click the "Default tag table" in the "PLC_1 > PLC tags" folder.	
2.	Edit the variables.	

... DC/DC/DC] ▸ PLC tags ▸ Default tag table [32]

Tags | User constants | System constants

Default tag table

	Name	Data type	Address ▲	Retain
1	TankLevelLack	Int	%MW0	☐
2	TankLevelMinimum	Int	%MW2	☐
3	TankLevelMidth	Int	%MW4	☐
4	TankLevelMaximum	Int	%MW6	☐
5	TankLevelOverflow	Int	%MW8	☐
6	TankLevel	Int	%MW10	☐
7	Alarm	Int	%MW12	☐
8	Flowrate	Int	%MW14	☐
9	StatusValveCPU	Bool	%M16.0	☐
10	CloseValve	Bool	%M16.1	☐
11	OpenValve	Bool	%M16.2	☐
12	Start	Bool	%M17.0	☐
13	Stop	Bool	%M17.1	☐
14	Reset	Bool	%M17.2	☐
15	StartStop	Bool	%M17.3	☐
16	Status_String	Int	%MW18	☐
17	ActFlowrate	Int	%MW22	☐
18	TankLevelScal	Int	%MW24	☐

5.4 Creating the HTML files

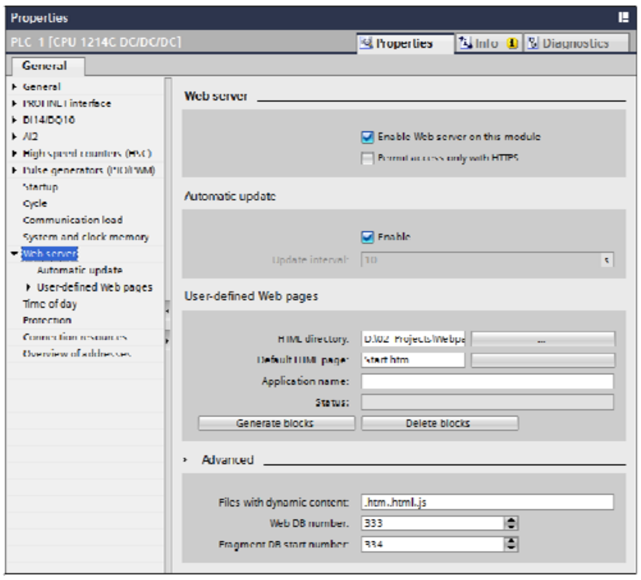
To create the HTML file, you need the list of variables from chapter 5.3 Creating the variables in the variable table or DB and a corresponding editor. Convenient editors such as Microsoft Frontpage are recommendable, which automatically create tags or mark correct inputs in color already during creation, or simple editors such as Notepad++.

Table 5-3

No.	Action	Comment
1.	Create the HTML files with an editor. Save the HTML files with the required images, stylesheets and scripts in the "\html" directory.	Detailed information on the creation of the HTML file can be found in chapter 3.3 Principles of user-defined web pages and 4.2 Functional principle of the HTML file.

5.5 Web server settings and generating of the data blocks

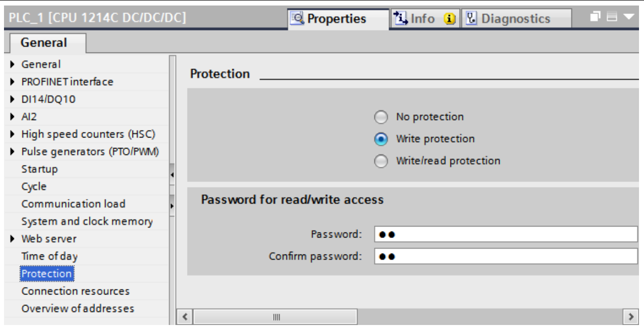
Table 5-4

No.	Action	Comment
1.	Click "Web server" in the properties of the PLC. Activate "Enable Web server on this module" and “Automatic update”. If you require safe access to the standard web pages, enable the "Permit access only with HTTPS" checkbox.	 Properties – PLC_1 [CPU 1214C DC/DC/DC] Web server [✓] Enable Web server on this module [ ] Permit access only with HTTPS Automatic update [✓] Enable Update interval: 10 s User-defined Web pages HTML directory: D:\02_Projects\Webpa Default HTML page: start.htm Application name: Status: Advanced Files with dynamic content: .htm;.html;.js Web DB number: 333 Fragment DB start number: 334
2.	Enter the HTML directory of your HTML files and select a start HTML page from the directory. Give the application a name.	
3.	Generate the Web Control DB (default: DB333) and the Fragment DBs (default: from DB334), by clicking "Generate blocks". STEP 7 V11 verifies the project with regard to the variables, loads the complementary files, such as, for example, the enumerations or images, reads in the variables of the HTML file, verifies the fragments, and writes all data in DB333 and from DB334. The status of the generation is displayed in an independent window or in the inspector window under Info.	

5.6 Setting CPU password

By default the password “s7” is specified for the web server. In order to specify a password for the web server, proceed as follows:

Table 5-5

No.	Action	Comment
1. .	Click "Protection" in the properties of the PLC. Enable "Write protection" of the CPU.	 PLC_1 [CPU 1214C DC/DC/DC] Properties Info Diagnostics General General PROFINET interface DI14/DQ10 AI2 High speed counters (HSC) Pulse generators (PTO/PWM) Startup Cycle Communication load System and clock memory Web server Time of day Protection Connection resources Overview of addresses Protection No protection [✓ Write protection] Write/read protection Password for read/write access Password: ●● Confirm password: ●●
2.	Enter a password and confirm this password.	

Note It makes no difference to the web server whether “Write protection” or “Write/read protection” is enabled. The web server does not support read protection.

5.7 Creating, compiling and loading of the S7 program

An exemplary S7 program can be found in the appendix to this entry. The following aspects must be considered when creating the S7 program:

- Call the "WWW" (SFC99) instruction. The "WWW" instruction initializes the web server of the CPU. With the cyclic calling of the "WWW" instruction, you ensure that changed variables of the CPU can be displayed on the web page. The cyclic calling of the "WWW" instruction is done in OB1.
- Enter the number of the Web Control DB (e.g. 333) at the CTRL_DB input parameter of the "WWW" instruction.

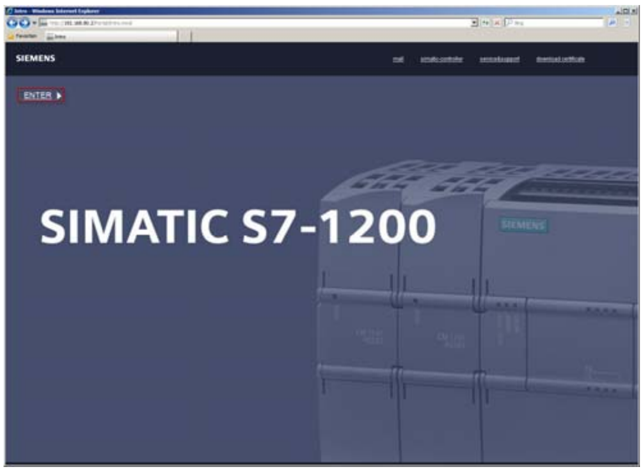
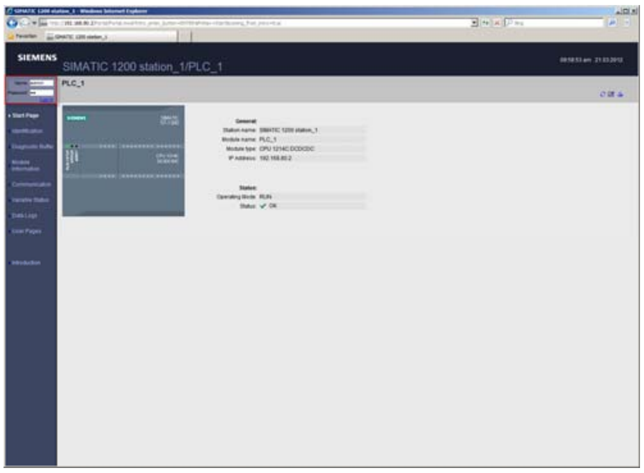
Table 5-6

No.	Action	Comment
1.	Compiling Right click the S7 1214C DC/DC/DC and select "Compile > All".	
2.	Loading project Right click the S7 1214C DC/DC/DC and select "Download to device > All". Set your PG/PC interface in the dialog window as follows: • PN/IE • <network adapter> • (local) PN/IE Select the S7 1214C DC/DC/DC and subsequently click "Load".	The dialog window for setting the PG/PC interface is only displayed during the first loading.

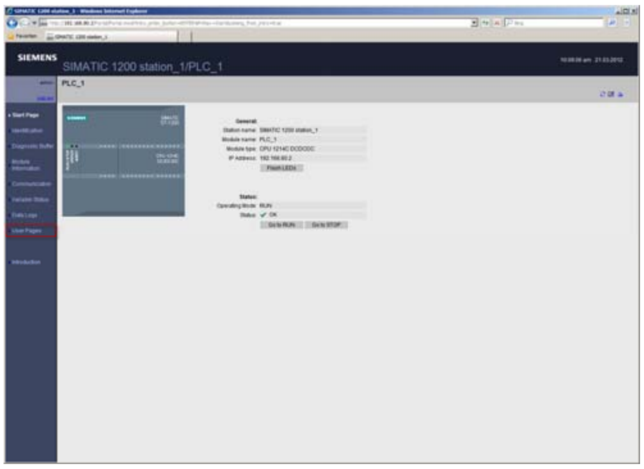
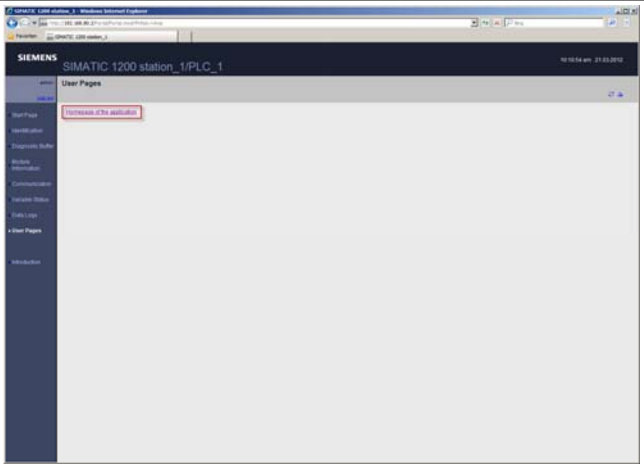
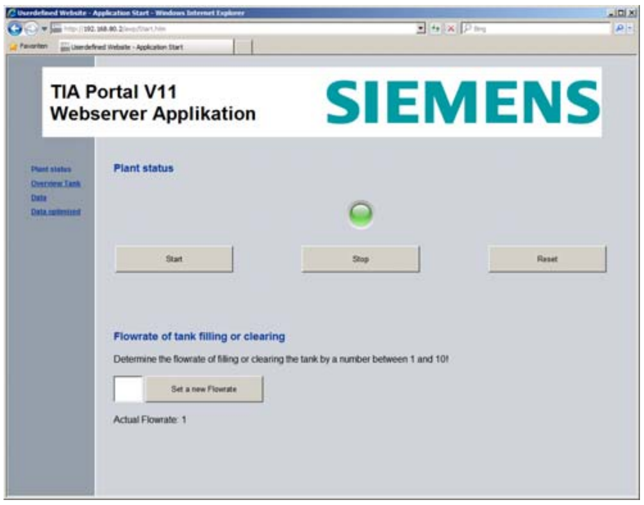
Note If you want to use a different S7-1200, you have to exchange the CPU under “Devices & networks”. After the exchange, the web server has to be re-enabled (see 5.5 Web server settings and generating of the data blocks.

5.8 Calling the web page with a web browser

Table 5-7

No.	Action	Comment
1.	• Start a web browser, e.g. the Internet Explorer. Enter the IP address of the CPU as the address, e.g. http://192.168.80.20. Note: your PC and the CPU must be located in the same subnet. The start web page of the CPU is opened. • Click "ENTER".	
2.	Enter the name "admin" and the password “s7“. Then, click "Log in". The complete web page of the CPU is opened.	
	Note: If “There is a problem with this website's security certificate” appears instead of the desired page, go to the "Introduction" introductory page and download the Siemens security certificate for, e.g. IE as follows: • Click the "download certificate" link on the introductory page. The "File Download – Security Warning" dialog is displayed. • Click "Open" in the "File Download – Security Warning" dialog to open the file. The "Certificate" dialog is displayed. • Click the "Install Certificate…" button in the "Certificate" dialog, to call the assistant for importing the certificate. • Follow the dialogs in the "Certificate Import Wizard" in order to import the certificate. Select the "Trusted Root Certification Authorities" certification storage.	

5.8 Calling the web page with a web browser

No.	Action	Comment
3.	Click "User Pages" to go to the user-defined web page.	 SIEMENS SIMATIC 1200 station_1/PLC_1 PLC_1 General: Station name: SIMATIC 1200 station_1 Module name: PLC_1 Module type: CPU 1214C DC/DC/DC IP Address: 192.168.80.2 Status: Operating Mode: RUN Status: ✓ OK
4.	To start the example application, click the "Homepage of the application". The "Start" web page is opened.	 SIEMENS SIMATIC 1200 station_1/PLC_1 User Pages Homepage of the application
5.	A detailed explanation of the operation of the example web page can be found in chapter 7 Operating the Application.	 Userdefined Website - Application Start - Windows Internet Explorer TIA Portal V11 Webserver Applikation SIEMENS Plant status Start Stop Reset Flowrate of tank filling or clearing Determine the flowrate of filling or clearing the tank by a number between 1 and 10! Set a new Flowrate Actual Flowrate: 1

6 Installation

6.1 Hardware and software installation

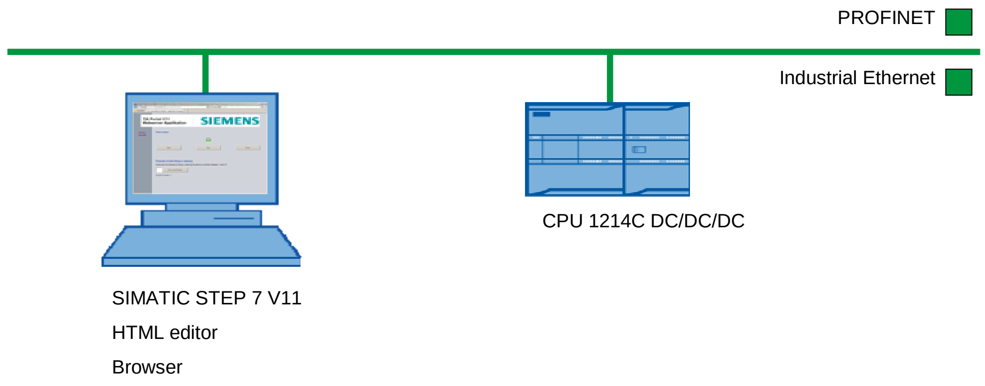
Hardware installation

The following figure shows the hardware structure of the example application.

The PC with the web browser must be connected to the CPU via Industrial Ethernet, e.g.

- directly at the PN interface of the CPU
- via a switch

Figure 6-1



Note Please observe the installation and connection guidelines from the corresponding manuals.

Software installation

Table 6-1

No.	Action	Comment
1.	Install STEP 7 V11 SP2.	
2.	Install a tool for creating the web page, e.g. MS Frontpage or Notepad++ on the PC with which you want to create the web page.	
3.	Install a web browser on the PC, e.g. Internet Explorer or Firefox, with which you want to access the web page of the CPU.	

6.2 Installation of the application example

Table 6-2

No.	Action	Comment
1.	Unzip the “Example_S7-1200_Webserver.zip” file in your project directory.	
2.	Start the SIMATIC STEP 7 V11 SP2.	
3.	Open the project in SIMATIC STEP 7 V11 SP2.	
4.	Go to the device view.	
5.	If you are using a different CPU, change the device.	
6.	In the CPU properties, assign the IP address of your CPU to the Ethernet interface.	Information in chapter 5.2 Configuration of the S7-1200 (1214C DC/DC/DC)
7.	Select the S7-1200 and load the entire project in the CPU.	
8.	Start a web browser and call the web page of your CPU via the IP address.	Information in chapter 5.8 Calling the web page with a web browser

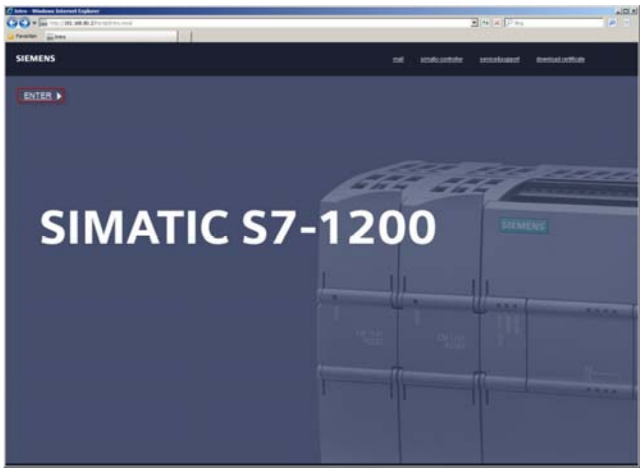
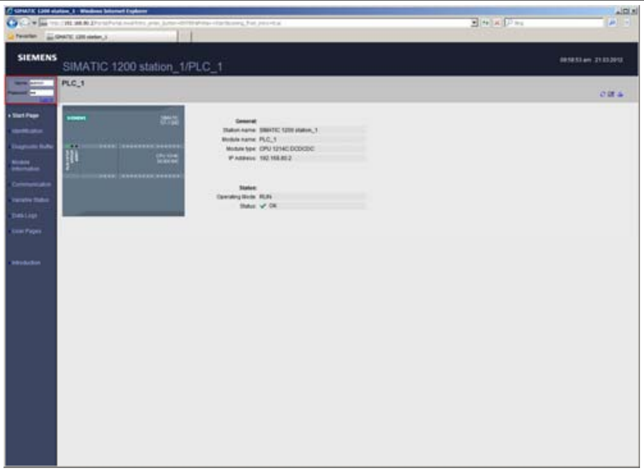
7 Operating the Application

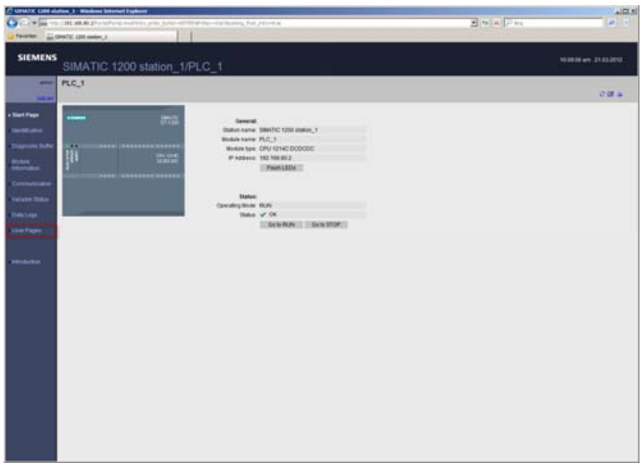
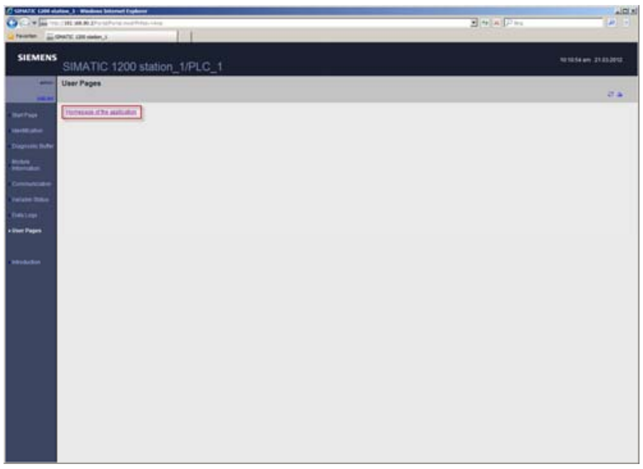
In this chapter

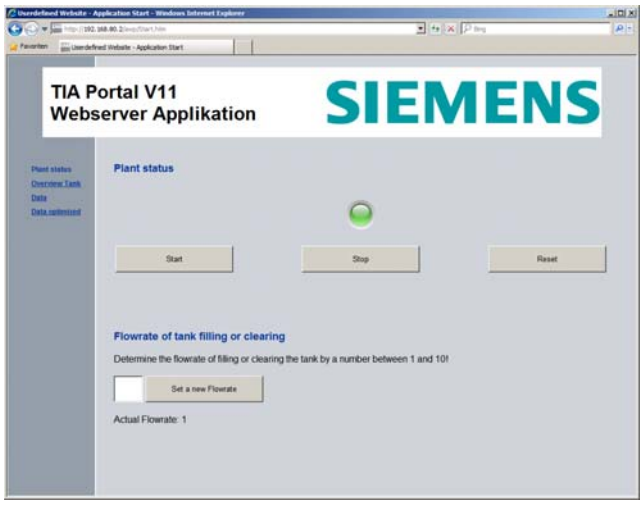
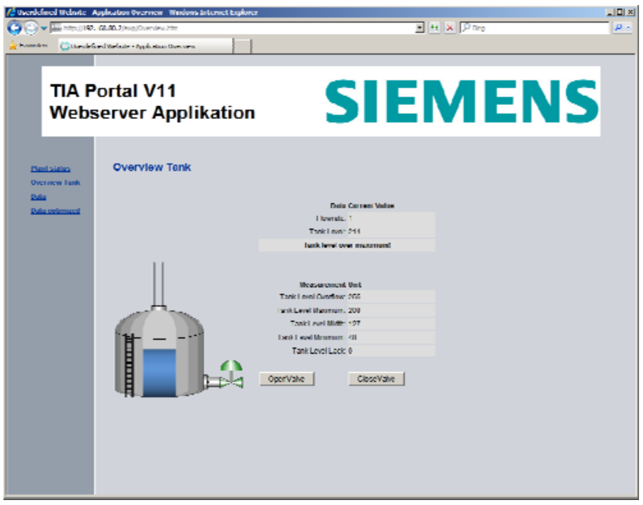
This chapter provides information on how to operate the example application.

Operation

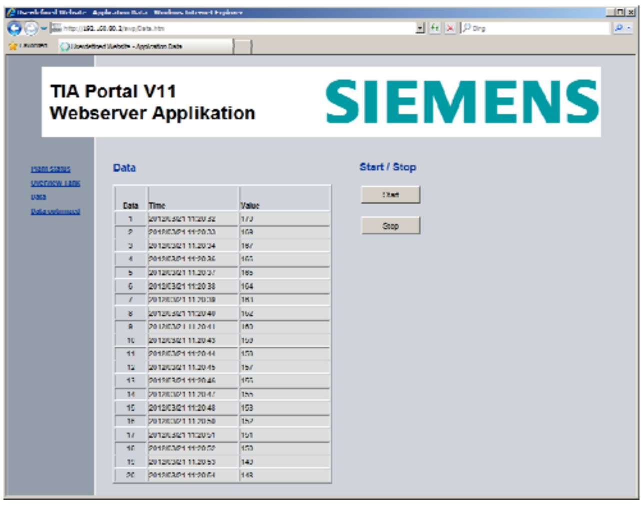
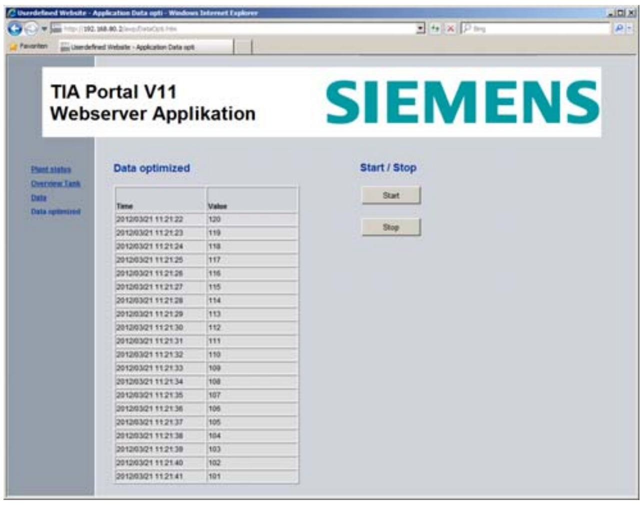
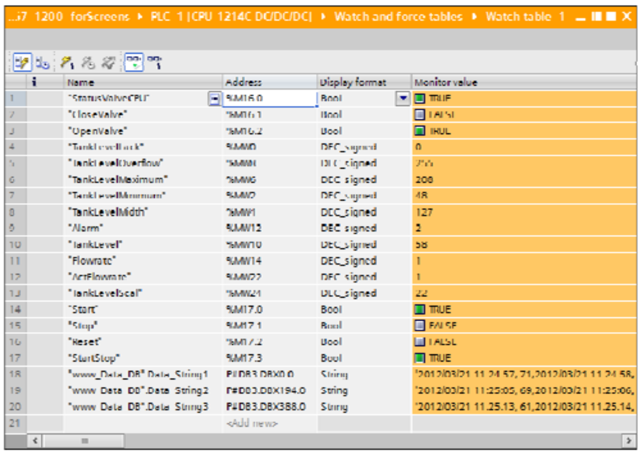
Table 7-1

No.	Action	Comment
1.	• Start a web browser, e.g. the Internet Explorer. Enter the IP address of the CPU as the address, e.g. http://192.168.80.20. The start web page of the CPU is opened. • Click "ENTER".	
2.	Enter the name "admin" and the password “s7“. Then, click "Log in". The complete web page of the CPU is opened.	
	Note: If “There is a problem with this website's security certificate” appears instead of the desired page, go to the "Introduction" introductory page and download the Siemens security certificate for, e.g. IE as follows: • Click the "download certificate" link on the introductory page. The "File Download – Security Warning" dialog is displayed. • Click "Open" in the "File Download – Security Warning" dialog to open the file. The "Certificate" dialog is displayed. • Click the "Install Certificate…" button in the "Certificate" dialog, to call the assistant for importing the certificate. • Follow the dialogs in the "Certificate Import Wizard" in order to import the certificate. Select the "Trusted Root Certification Authorities" certification storage.	

No.	Action	Comment
3.	Click "User Pages" to go to the user-defined web page.	 SIEMENS SIMATIC 1200 station_1/PLC_1 PLC_1 Start Page Identification Diagnostic Buffer Module Information Communication Variable Status Data Logs User Pages Introduction General: Station name: SIMATIC 1200 station_1 Module name: PLC_1 Module type: CPU 1214C DC/DC/DC IP Address: 192.168.80.2 Flash LEDs Status: Operating Mode: RUN Status: ✓ OK Go to RUN Go to STOP
4.	To start the example application, click the "Homepage of the application". The "Start" web page is opened.	 SIEMENS SIMATIC 1200 station_1/PLC_1 User Pages Homepage of the application Start Page Identification Diagnostic Buffer Module Information Communication Variable Status Data Logs User Pages Introduction

No.	Action	Comment
5.	Via the web pages, you have direct access to the CPU: ***Plant Status*** web page: • Start the application by clicking the "Start" button. • By clicking "Stop" the application is stopped. • The ***"Reset"*** button puts the application back into its original state. • The operating status of the application is displayed via the LED (red: off; green: on) • The flow rate can be entered manually. In the S7 program, a medium flow rate of 5 is preset. The higher the value for "Flowrate" is set, the faster the flow rate. Entering the flow rate is monitored by a script. If there is an incorrect entry, a message is output. Via the links on the navigation bar, you can switch between the web pages.	 TIA Portal V11 Webserver Applikation Plant status Flowrate of tank filling or clearing Determine the flowrate of filling or clearing the tank by a number between 1 and 10! Actual Flowrate: 1
6.	***Tank overview*** web page: • Via "TankLevel" you can see the current filling level of the tank. Additionally, the filling level is commented via clear text. • The key figures of the filling level are displayed below. • When clicking the "OpenValve" button, the tank is emptied. • The valve is closed via "CloseValve" – the tank is filled again. • Valve green: open Valve red: closed • The web page is automatically refreshed and the values are adjusted. Via the links on the navigation bar, you can switch between the web pages.	 TIA Portal V11 Webserver Applikation Overview Tank Data Current Value Flowrate: 1 Tank Level: 211 Tank level over maximum! Measurement Unit Tank Level Overflow: 255 Tank Level Maximum: 200 Tank Level Middle: 127 Tank Level Minimum: 40 Tank Level Lock: 0

Userdefined Website - Application Start - Windows Internet Explorer
Plant status
Overview Tank
Data
Data optimized
Start
Stop
Reset
Set a new Flowrate
SIEMENS
Userdefined Website - Application Overview - Windows Internet Explorer
OpenValve
CloseValve

No.	Action	Comment
7.	***Data*** web page: • The web page shows the ring buffer of the last 20 tank fill levels. • The page is automatically refreshed. Via the links on the navigation bar, you can switch between the web pages.	
8.	***Data optimized*** web page: • The web page shows the last 20 tank fill levels. • The data is automatically refreshed via JavaScript. • Start the application by clicking the "Start" button. • By clicking "Stop", the application is stopped. Via the links on the navigation bar, you can switch between the web pages.	
9.	In parallel, you can monitor the change of the variables in the watch table in STEP 7 V11.	

TIA Portal V11
Webserver Applikation

Data

Data	Time	Value
1	2012/03/21 11:20:32	17?
2	2012/03/21 11:20:33	169
3	2012/03/21 11:20:34	167
4	2012/03/21 11:20:36	166
5	2012/03/21 11:20:37	165
6	2012/03/21 11:20:38	164
7	2012/03/21 11:20:39	161
8	2012/03/21 11:20:40	162
9	2012/03/21 11:20:41	160
10	2012/03/21 11:20:43	159
11	2012/03/21 11:20:44	158
12	2012/03/21 11:20:45	157
13	2012/03/21 11:20:46	155
14	2012/03/21 11:20:47	155
15	2012/03/21 11:20:48	153
16	2012/03/21 11:20:50	152
17	2012/03/21 11:20:51	151
18	2012/03/21 11:20:52	150
19	2012/03/21 11:20:53	149
20	2012/03/21 11:20:54	148

TIA Portal V11
Webserver Applikation

Data optimized

Time	Value
2012/03/21 11:21:22	120
2012/03/21 11:21:23	119
2012/03/21 11:21:24	118
2012/03/21 11:21:25	117
2012/03/21 11:21:26	116
2012/03/21 11:21:27	115
2012/03/21 11:21:28	114
2012/03/21 11:21:29	113
2012/03/21 11:21:30	112
2012/03/21 11:21:31	111
2012/03/21 11:21:32	110
2012/03/21 11:21:33	109
2012/03/21 11:21:34	108
2012/03/21 11:21:35	107
2012/03/21 11:21:36	106
2012/03/21 11:21:37	105
2012/03/21 11:21:38	104
2012/03/21 11:21:39	103
2012/03/21 11:21:40	102
2012/03/21 11:21:41	101

...i7 1200 forScreens ▸ PLC 1 [CPU 1214C DC/DC/DC] ▸ Watch and force tables ▸ Watch table 1

	i	Name	Address	Display format	Monitor value
1		"StatusValveCPU"	%M16.0	Bool	🟩 TRUE
2		"CloseValve"	%M16.1	Bool	▢ FALSE
3		"OpenValve"	%M16.2	Bool	🟩 TRUE
4		"TankLevelLack"	%MW0	DEC_signed	0
5		"TankLevelOverflow"	%MW8	DEC_signed	255
6		"TankLevelMaximum"	%MW6	DEC_signed	208
7		"TankLevelMinimum"	%MW2	DEC_signed	48
8		"TankLevelMidth"	%MW4	DEC_signed	127
9		"Alarm"	%MW12	DEC_signed	2
10		"TankLevel"	%MW10	DEC_signed	58
11		"Flowrate"	%MW14	DEC_signed	1
12		"ActFlowrate"	%MW22	DEC_signed	1
13		"TankLevelScal"	%MW24	DEC_signed	22
14		"Start"	%M17.0	Bool	🟩 TRUE
15		"Stop"	%M17.1	Bool	▢ FALSE
16		"Reset"	%M17.2	Bool	▢ FALSE
17		"StartStop"	%M17.3	Bool	🟩 TRUE
18		"www_Data_DB".Data_String1	P#DB3.DBX0.0	String	'2012/03/21 11:24:57, 71,2012/03/21 11:24:58,
19		"www_Data_DB".Data_String2	P#DB3.DBX194.0	String	'2012/03/21 11:25:05, 69,2012/03/21 11:25:06,
20		"www_Data_DB".Data_String3	P#DB3.DBX388.0	String	'2012/03/21 11:25:13, 61,2012/03/21 11:25:14,
21			<Add new>		

8 Glossary

AWP

Automation Web Programming

AWP command

An AWP command is understood as the special command syntax with which data is exchanged between the CPU and the HTML file.

CSS

CSS (Cascading Style Sheets) defines how a section or content marked in HTML is displayed.

HTML file

HTML files are the basis of the World Wide Web and are displayed by a web browser.

In this document, we refer to the HTML file when you are editing the web page, e.g. with Frontpage. When you are working with the web page in a web browser, we refer to it as the web page.

HTTP

The Hypertext Transfer Protocol (HTTP) is a protocol for transferring data over a network.

HTTPS

The Hypertext Transfer Protocol Secure is a communication protocol that is used in the World Wide Web for the exchange of sensitive data.

MIME type

With the help of the Multipurpose Internet Mail Extensions (MIME) standard, the web browser is informed – e.g. during an HTTP transfer – which data the web server sends, for example whether it is clear text, an HTML document or a PNG image.

UTF-8

UTF-8 (abbreviation for 8-bit UCS Transformation Format) is the most widely used coding for unicode characters.

Each unicode character is assigned a specially coded byte chain of a variable length. UTF-8 supports up to four bytes on which all unicode characters can be displayed.

Web browser

Web browsers are visualization programs for web pages and can communicate with web servers.

Typical web browsers are:

- Microsoft Internet Explorer
- Mozilla Firefox

Web page

See HTML file.

Web server

A web server stores web pages and makes them available. A web server is a software program which transfers documents with the help of standardized transfer protocols (http, HTTPS) to a web browser.

A web server that you can expand with user-defined web pages, is integrated in a CPU with PROFINET interface.

9 Literature

9.1 Bibliographic references

The following list is by no means complete and only provides a selection of appropriate sources.

Table 9-1

	Topic	Title
/1/	STEP7 SIMATIC S7 - 1200	Automating in STEP 7 with SIMATIC S7-1200 Author: Hans Berger Publicis Publishing ISBN: 978-3895783562
/2/	HTML	HTML und CSS, Praxisrezepte für Einsteiger Robert R. Agular mitp ISBN 978-3-8266-1779-9
/3/	HTML	HTML Handbuch Stefan Münz/Wolfgang Nefzger Franzis Verlag ISBN 3-7723-6654-6
/4/	Javascript	JavaScript und Ajax, Das umfassende Handbuch Christian Wenz Galileo Press ISBN 978-3-8362-1128-4

9.2 Internet Links

The following list is by no means complete and only provides a selection of appropriate sources.

Table 9-2

	Topic	Title
/1/	Reference to this document	http://support.automation.siemens.com/WW/view/en/58862931
/2/	Siemens Industry Online Support	http://support.automation.siemens.com
/3/	HTML, JavaScript	http://www.w3.org/ http://www.w3.org/TR/1999/PR-html40-19990824/ http://en.wikipedia.org/wiki/JavaScript
/4/	S7-1200 System Manual	http://support.automation.siemens.com/WW/view/en/36932465

10 History

Table 10-1

Version	Date	Modification
V1.0	03/2012	First issue