SIEMENS

Application Description • 02/2014

Basics on Creating HTMLs for SIMATIC CPUs

SIMATIC STEP 7 V12

http://support.automation.siemens.com/WW/view/de/68011496

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 and without prior notice. If there are any deviations between the recommendations provided in this application example 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 will be excluded. Such an exclusion will 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.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com.

Table of Contents

1 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 an S7 CPU with PROFINET interface.

1.1 General principles of web pages

If you already have basic knowledge of HTML, you may skip this chapter and continue reading with chapter 1.2 Principles of standard web pages.

1.1.1 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.

```
<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 1-1

Representation	Function	Example
`<!-- … -->`	Comment	`<!-- This is a comment! -->`
`<a href="…"> … </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 src="…">`	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 style sheet 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	

1.1.2 Using forms

In order to interact with the application, please use forms in HTML.

For example, you can fill in input fields in a form and then send the form by clicking on a button. This sends the content of the form 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.

Note Do not use the forms on pages which are automatically refreshed. Your entries will be overwritten when refreshed.

1.1.3 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 1-2

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`	Write 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 / font stype	`bold, normal, bolder, lighter, 100 to 900`

CSS property	Function	Examples for values
text-decoration	Text decoration	underline, blink, none
text-transform	Text transformation	uppercase, lowercase
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 (general) Margin top Margin right Margin bottom Margin 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 Repeat 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 `<style>` and `</style>` tags
- in an external CSS file

Define the style sheets 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="<Formate>.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.

1.1.4 Principles of JavaScript

JavaScript enables evaluating user interactions and changing, reloading, or crating contents; hence, it expands the possibilities of HTML/CSS.

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

Define the JavaScript code in a separate file if you want to use the same 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>
```

1.1.5 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 for CPU 1516-3 PN/DP

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 1-3

Number of variables	Communication load [%]	Refresh time [s]
10	20	1.2
10	40	1.1
20	20	1.4
20	40	1.3
40	20	1.6
40	40	1.5

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” in the properties of the PLC, 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>
- Automatic refreshing with a meta date in the head of the HTML file
- Automatic refreshing with JavaScript

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

Press <F5> (Internet Explorer: "View > Refresh") in order to refresh the display in the web browser.

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.html">
```

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, this is "Overview.html".

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()",10000);
</script>
```

1.2 Principles of standard web pages

Requirements

In STEP 7, the following settings are required in the properties of the PLC:

- The web server must be activated.
- 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 can lie in the range of between 1 to 999.

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 S7-1500 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 S7-1500 CPU is only possible via HTTPS.

Log in

The user with the name "Anybody" is the default setting in the user list. This user has minimum access rights (read access to the intro and start page). The access rights of user “Everybody” can be extended. User “Everybody” has been defined without password.

In order to use the full functionality of the web pages, you need to be logged in. Log on with a user name and password defined in the web configuration of STEP 7. Then you can access the web pages enabled for this user with the appropriate access rights.

The input fields for login can be found in the top left corner of each standard web page.

Figure 1-1 Login window

Standard web pages of SIMATIC S7-1500

The web server of the S7-1500 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 1-4

Name	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	Symbolic display of the status of a module with comments
Messages	Display of the message buffer content
Communication	Display of the information on the PROFINET interfaces of the CPU; display of the resource demand of the connection
Topology	Display of topological setup and status of the PROFINET devices of the PROFINET IO system in a graph and table
User pages	In the “User pages” of the web server you can load your created HTML pages for reading data of the target system.
File browser	The file browser lists all data files and directories which exist on the SIMATIC memory card. The files can be downloaded, deleted, renamed or uploaded. The directories can be recreated, deleted or renamed.

A detailed description of the setup of the standard web pages is available in the S7-1500 Web server function manual; it is not subject of this application document.

1.3 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 V12 and, amongst others, to the “WWW” (SFC 99) instruction.

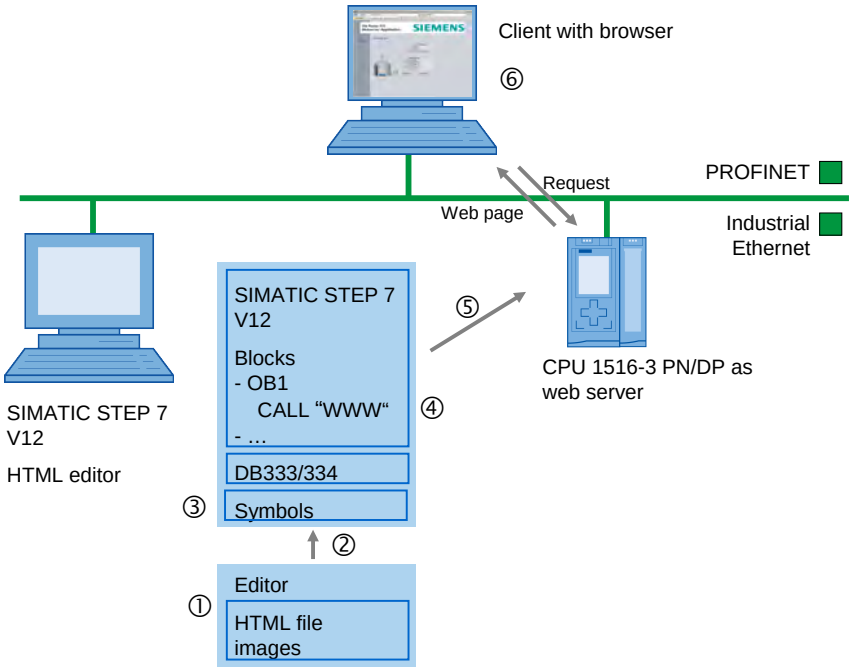
Advantages

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.

1.3.1 Creating user-defined web pages

Figure 1-2 Overview of creating user-defined web pages



Procedure

Table 1-5

No.	Instruction
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.	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 navigation. 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.
3.	Assign a symbolic name in STEP 7 to variables which you want to use on the web page.
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 S7-1500 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.

1.3.2 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” file of the program navigation of STEP 7.

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.

1.4 Displaying variables from the CPU on the web page

Typical use of variables

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

Table 1-6

Representation	Function	Example	Informa-tion
`:="<Name>"`	Display CPU variable	`:="TankLevelMinimum":`	Chapt. 1.4
`<!-- 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"' -->`	Chapt. 1.5
`<!-- AWP_Enum_Ref Name='"<Name>"' Enum="<Variable>" -->`	Assignment of enumerations (texts) to the value of a variable	`<!-- AWP_Enum_Ref Name='"Alarm"' Enum="AlarmValue" -->`	Chapt. 1.6

1.4.1 Requirements

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

Table 1-7

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. • A cyclic call of the “WWW” (SFC99) instruction is necessary if variables are pre-processed in the S7 program. • For variables, the standard data types, user-generated PLC data types, and structures are permitted.	• It is not necessary to declare variables via an AWP command in the HTML file.

1.4.2 Interaction between web browser and CPU

Figure 1-3 Interaction between web browser and CPU when reading variables

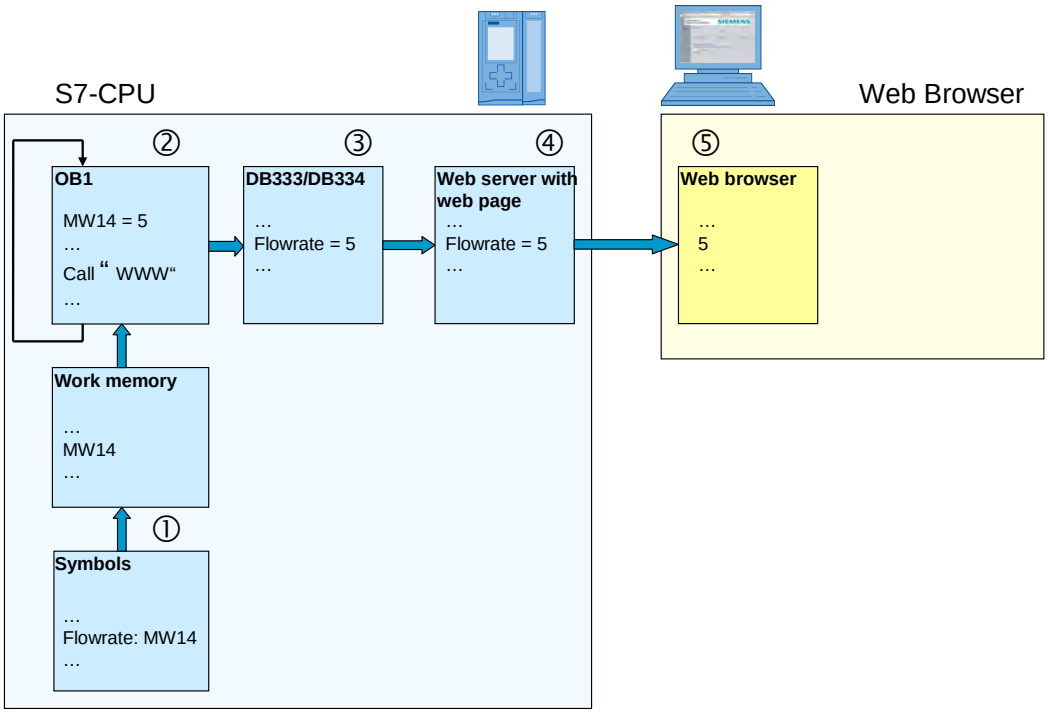


Table 1-8

No.	Description
7.	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`.
8.	In the S7 program, the “WWW” (SFC99) instruction is called.
9.	By calling the “WWW” (SFC99) instruction, the web control DB (default: DB333) is initialized.
10.	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.
11.	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 1.1.5 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.

1.4.3 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 1.1.5 Automatic refreshing of the web page.

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

1.5.1 Requirements

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

Table 1-9

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, user-generated 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).

1.5.2 Interaction between web browser and CPU

Figure 1-4 Interaction between web browser and CPU when writing variables

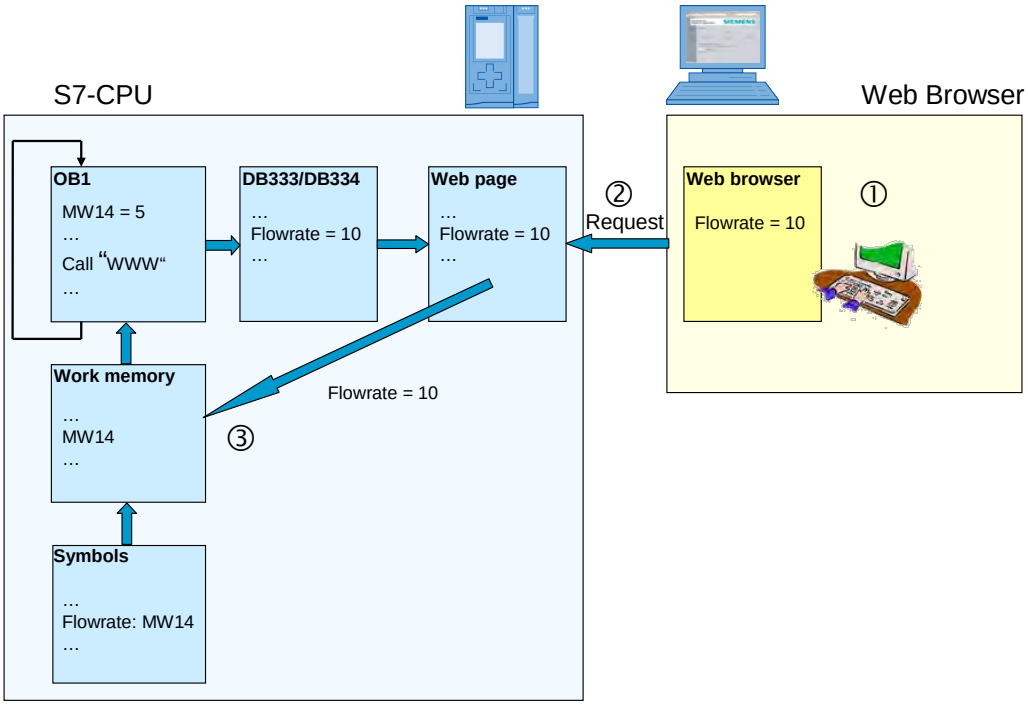


Table 1-10

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.

1.5.3 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"' -->
```

Typically, the AWP command is at the first instruction in the HTML file in which the variable is used.

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.

The form data is transferred via "submit".

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>
```

1.6 Linking variables with texts in the HTML file

In some cases, 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.

1.6.1 Requirements

To output indications as text, the following prerequisites apply:

Table 1-11

S7 program	HTML file
• Each variable must be assigned a symbolic name. A variable can only be addressed via symbolic names. • A cyclic call of the “WWW” (SFC99) instruction is necessary if variables are pre-processed in the S7 program. • 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.

1.6.2 Interaction between web browser and CPU

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

Figure 1-5 Interaction between web browser and CPU when converting variables to text

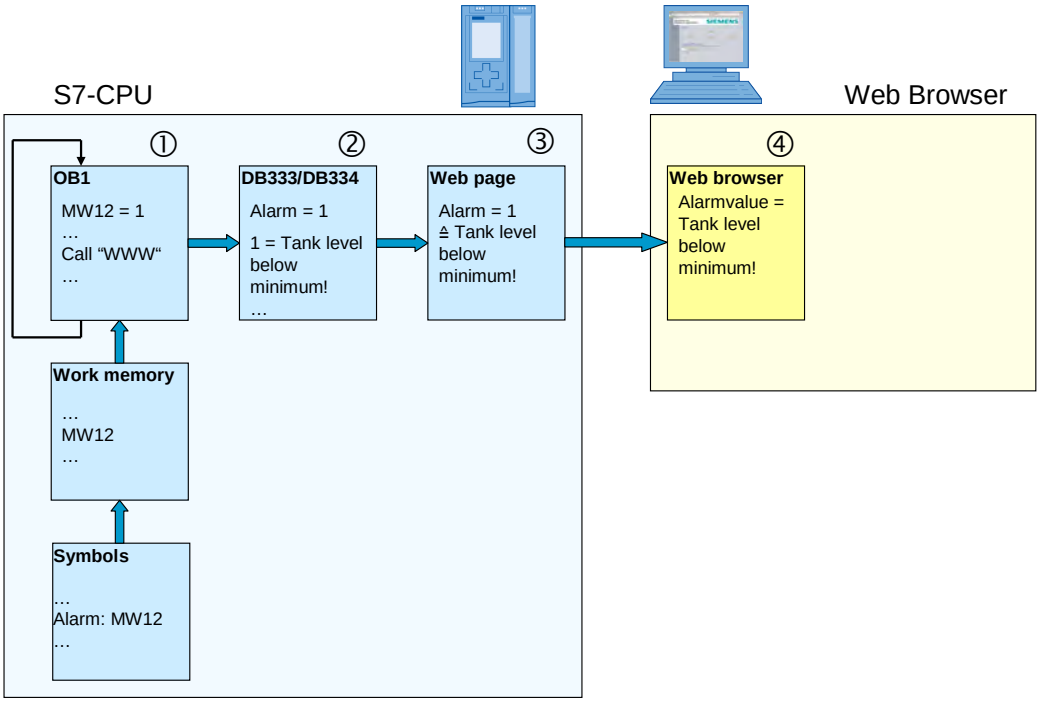


Table 1-12

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.

1.6.3 Procedure

Creating ENUM TYPE

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

```
<!-- AWP_Enum_Def Name= ="<Name des Enum Typs>"
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 first instruction in the HTML file in which the variable is used.

Assigning ENUM TYPE

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

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

1.7 Creating time-optimized HTML pages (optional)

Creating time-optimized HTML pages is not described in the manuals of the S7-1500. The idea behind refreshing via JavaScript is that of a sub-web page being integrated into the user-defined web page. This is done with a so-called “inline frame” (iFrame). In order to fetch new values from the controller, not the complete web page is reloaded but only the sub-web page in the inline frame. This has the advantage that only few data needs to be called by the controller. If, for example, pictures are used on the main page, they need not be requested from the controller for each refreshing, which they do in both of the other introduced mechanisms. Refreshing via JavaScript therefore causes a low data transmission which may also have a positive effect on the network load as well as the cycle time of the controller. The concept is displayed in the figure below.

Figure 1-6 Read/write variable

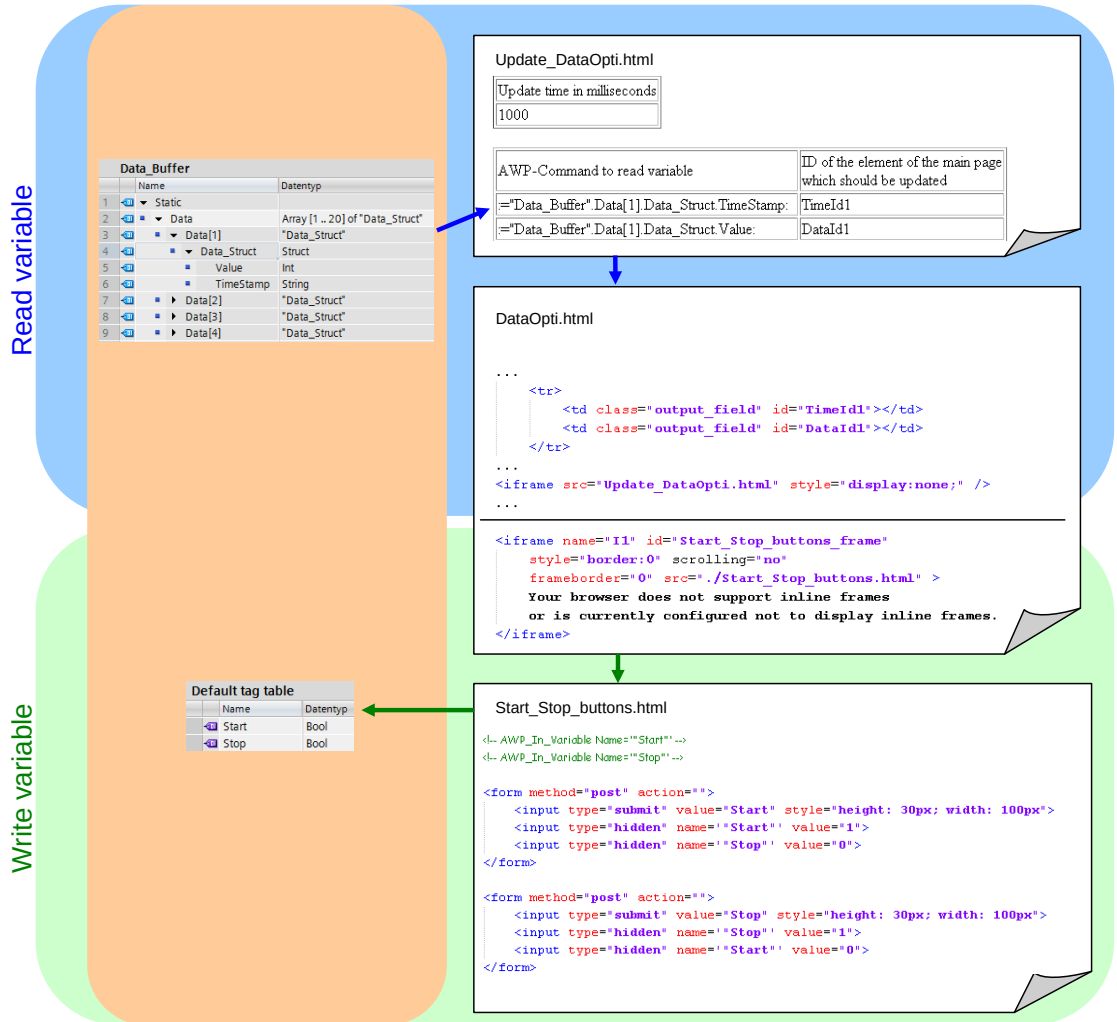


Table 1-13

No.	Description
1.	There is a user-defined HTML page (below called main page) to be displayed. In this application this is the “DataOpti.html” HTML page. It is not refreshed.
2.	This main page does not include variables. All control variables to be read or to be written are packed in additional HTML files (here “Update_DataOpti.html” and “Start_Stop_buttons.html“), which on their part are embedded as frames (iframe) in the main page. Only the individual frames are refreshed.
3.	The “Update_DataOpti.html” file contains two tables. In the second line of the first table, the update time is given in milliseconds (e.g. 1000). From the second line of the second table on, the left column contains AWP commands of the variable to be read (e.g. `:="Data_Buffer".Data[1].Data_Struct.TimeStamp:`). The right column contains the respective IDs (e.g. `TimeId1`). The IPs must be unique. The main page “DataOpti.html” contains the entered IDs instead of the AWP commands of the variable to be read. I.e., in order to use an element for the update, the `"ID"` attributes must be entered and pre-assigned with the respective ID name (e.g. `id="TimeId1"`). The separate HTML file “Update_DataOpti.html” is embedded as frame (`iframe`, see point 2) in the main page “DataOpti.html”. The respective syntax is as follows: `<iframe src="Update_DataOpti.html" style="display:none;" />` The “`display:none`” CSS property, noted for the frame, prevents its display on the main page.
4.	The JavaScript functions (see Figure 1-7 JavaScript functions) in the “Update_DataOpti.html” file automatically determine the current size of the second table. The HTML elements with the ID are determined in the main page. The JavaScript program code replaces the entire content of this HTML element with the content of the left table column of the second table. Finally, a wait time is set from the time defined in the first table and the update is then renewed.
5.	For input variables (see chapter 1.5 Writing variables on the CPU with the help of the web page) forms are inserted in separate HTML files. The HTML files are embedded as frames in the main page. Several forms can be written in a HTML file.

Figure 1-7 JavaScript functions

```
/* String.trim() is not supported by every browser - thus add this functionality is necessary */
if (!String.trim)
{
    String.prototype.trim = function () { return this.replace(/^(\s|\u00A0)+|(\s|\u00A0)+$/g, ''); };
}

/* Main function for calling the subroutines */
function mainJavaScript()
{
    updateSingleVariablesTable();
    updateFrame();
}

/* Function for updating the variables */
function updateSingleVariablesTable()
{
    var table = document.getElementById("singleVariablesTable");
    for(i = 1; i < table.rows.length; i++)
    {
        var tagValue = table.rows[i].getElementsByTagName("td")[0].innerHTML;
        var tagId = table.rows[i].getElementsByTagName("td")[1].innerHTML.trim();
        parent.document.getElementById(tagId).innerHTML = tagValue;
    }
}

/* Function for updating the web page */
function updateFrame()
{
    window.setTimeout(function(){location.reload()}, document.getElementById("updateInterval").innerHTML.trim());
}
```


Note

The “Update_DataOpti.html” file can be adapted to your application with little effort. You only need to enter your variable with the respective ID into the second table. On your main page you replace the variable with the respective ID. This does not require any changes at JavaScript.

Further information on this topic is available at the following FAQ:
How can user-defined web pages and standard web pages be updated automatically in STEP 7 (TIA Portal)?

1.8 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.

1.8.1 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 1-8 AWP commands

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


Explanations

Table 1-14

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.

1.8.2 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. The <link…> and <script…> tags are also optional.

Figure 1-9 Information on doctype and head of the HTML file

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

    <head>
        <title>Userdefined Website - Application Overview</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="1; URL=Overview.html">
        <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 1-15

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 Overview</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. MIME stands for Multipurpose Internet Mail Extensions. MIME type describes the type of the transferred data.

Code	Explanation
`<meta http-equiv="refresh" content="1; URL=Overview.html">`	Optional meta date: with this command, the web page is refreshed every second. Especially for process monitoring it is appropriate to have the web page 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 Section 1.1.5 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.

1.8.3 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 1-10 Areas of the HTML file

```
<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 1-11 Formatting the areas in the CSS file

```
#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 1-16

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 1.1.3 Basics on Cascading Style Sheets (CSS).

1.8.4 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 1-12 Representation of pictures in the HTML file

```
<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">
    <div style="position:relative; bottom:-49px; left:48px;">
        <img src="Images\blue.png" alt="Level" width="56px" height=":=TankLevelScal:px ">
    </div>
    <div style="position:relative; bottom:16px; left:175px;">
        <img src="Images\Valve:="StatusValveCPU":.png" name="Valve" "StatusValveCPU = 0"
            id="StatusValveCPU" alt="Valve" >
    </div>
</td>
```

Table 1-17

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 ">`	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`".
`<img src="Images\Valve:="StatusValveCPU":.png" name="Valve" "StatusValveCPU = 0" id="StatusValveCPU" alt="Valve" >`	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 image call consists 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.

1.8.5 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 and last line of the table are shown for reasons of clarity.

Figure 1-13 Representation of table in the HTML file

```
<table border=1 id='Table1'>
    <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" id="TimeId1"></td>
        <td class="output_field" id="DataId1"></td>
    </tr>
    ...
    <tr>
        <td class="static_field_small">20</td>
        <td class="output_field" id="TimeId20"></td>
        <td class="output_field" id="DataId20"></td>
    </tr>
</table>
```

Table 1-18

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.

1.8.6 Outputting CPU variables

Explanations

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

Figure 1-14 Representation of tags in the HTML file

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

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

1.8.7 Outputting texts via enumerations

Explanations

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

Figure 1-15 Representation of enumerations in the HTML file

```
<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 stored as enum-type "AlarmValue" and are transferred to the web page via DB333.

1.8.8 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 1-16 Representation of entries in the HTML file

```
<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 1-19

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 since with `action` the current page is called by default.
	With the calling `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".

1.8.9 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 1-17 Representation of buttons in the HTML file

```
<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>
```

Table 1-20

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 since 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".

1.8.10 Logging on directly on user-definable web pages

Basics

In order to write variable to the CPU, you must generally be logged in at the web server of the CPU. Login is not necessary if you wish to grant write rights to the user-defined pages for user “Everybody”. However, this cannot be recommended from the point of view of plant security. The login window is available in the top left corner of the standard web pages. The login-window was integrated on the user-defined web pages so you need not change to the standard web pages for the login.

Explanations

Figure 1-18 Login on user-defined web pages in the HTML file

```
<body onload="loginCheck()">   <!-- check login with each refresh of the website -->
    ...
    <!-- BEGIN Login Area -->
    <!-- use 'src="../../Portal/Portal.mwsl"' if you 'Generate blocks' with 'Application name' and
         use 'src="../Portal/Portal.mwsl"' if you 'Generate blocks' without 'Application name'-->
    <iframe id="WebserverIFrame" name="WebserverIFrameName"
            src="../../Portal/Portal.mwsl" style="display:none"></iframe>

    <!-- area for login -->
    <div id="logForm" class="Login_Area" colspan="2"></div>

    <!-- function for check login -->
    <script type="text/javascript">

    function loginCheck()
    {
            var iFrameElement = document.getElementById('WebserverIFrame');
            var loginForm = iFrameElement.contentWindow.document.getElementById('Login_Area_Form');
            if(loginForm)
            {
                    //alert("Not logged in");
            document.getElementById('logForm').innerHTML = loginForm.parentNode.innerHTML;
            }
            var logoutForm = iFrameElement.contentWindow.document.getElementById('Logout_Area_Form');
            if(logoutForm)
            {
                    //alert("admin is logged in");
            document.getElementById('logForm').innerHTML = logoutForm.parentNode.innerHTML;
            // delete '/Applicationname' in next line, if you 'Generate blocks' without 'Application name'
            document.getElementsByName("Redirection")[0]["value"] = "../awp/Applicationname/Start.html";
            }
    }
    </script>
    <!-- END Login Area -->
```

Table 1-21

Code	Explanation
`<body onload="loginCheck()">`	For each refreshing of the web page, the JavaScript function `loginCheck()` is automatically executed.
`<iframe id="WebserverIFrame" name="WebserverIFrameName" src="../../Portal/Portal.mwsl" style="display:none"></iframe>`	In the user-defined page, an `iframe` is defined which contains the standard web page. The `iframe` is made invisible with the `style="display:none"` attribute.
`<div id="logForm" class="Login_Area" colspan="2"></div>`	`class="Login_Area"` assigns the formats from the CSS file to the login window with `id="logForm"`. The ID is defined in the `loginCheck()` function.
`function loginCheck()`	The JavaScript function `loginCheck()` polls in the iFrame on the standard web page whether the form for logging in or logging out exists. The respective form for logging in or logging out is added respectively on the user-defined web page.

Note Further information on this topic is available at the following FAQ:

How can you access user-defined web pages of the S7-1200 directly?

2 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.

Firewall

The firewall is used for restricting the network access based on sender or target address and used services. For the data traffic handled via the firewall, it decides based on fixed rules whether certain network packages are transported or not. In this way, the firewall attempts to prevent unauthorized network access.

The function of a firewall is not to detect attacks. It exclusively implements the rules for the network communication.

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. In the web browser we refer to this page as 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 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 (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.

3 Related Literature

3.1 Bibliography

This table offers you a variety of pertinent literature.

Table 3-1

	Topic	Title
/1/	HTML	HTML und CSS, Praxisrezepte für Einsteiger [HTML and CSS, practical recipes for beginners] Robert R. Agular mitp ISBN 978-3-8266-1779-9
/2/	HTML	HTML Handbuch [HTML Manual] Stefan Münz/Wolfgang Nefzger Franzis Verlag ISBN 3-7723-6654-6
/3/	Javascript	JavaScript und Ajax, Das umfassende Handbuch [JavaScript and Ajax, The comprehensive manual] Christian Wenz Galileo Press ISBN 978-3-8362-1128-4

3.2 Internet link specifications

This table contains a selection of links on further information.

Table 3-2

	Topic	Title
/1/	Link to this document	http://support.automation.siemens.com/WW/view/de/68011496
/2/	Siemens Industry Online Support	http://support.automation.siemens.com
/3/	HTML, JavaScript	http://www.selfhtml.de/ http://de.selfhtml.org/ http://www.little-boxes.de/
/4/	S7-1500 Web server function manual	http://support.automation.siemens.com/WW/view/en/59193560
/5/	S7-1500 System Manual	http://support.automation.siemens.com/WW/view/en/59191792
/6/	Programming guideline for S7-1200/1500	http://support.automation.siemens.com/WW/view/en/81318674

4 History

Table 4-1

Version	Date	Modifications
V1.0	02/2014	First version