SIEMENS

SIMATIC NET

S7Beans / Applets für IT–CPs

Programmierhilfe

Ergänzung zur Dokumentation der IT–CPs
CP 243–1 IT
CP 343–1 IT
CP 443–1 IT

Ausgabe 06/2003
C79000–G8900–C180–01

Klassifizierung der Sicherheitshinweise

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise sind durch ein Warndreieck hervorgehoben und je nach Gefährdungsgrad folgendermaßen dargestellt:

Gefahr

bedeutet, daß Tod, schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Warnung

bedeutet, daß Tod, schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht

mit Warndreieck bedeutet, daß eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht

ohne Warndreieck bedeutet, daß ein Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Achtung

bedeutet, das ein unerwünschtes Ergebnis oder Zustand eintreten kann, wenn der entsprechende Hinweis nicht beachtet wird.

Hinweis

ist eine wichtige Information über das Produkt, die Handhabung des Produktes oder den jeweiligen Teil der Dokumentation, auf den besonders aufmerksam gemacht werden soll und deren Beachtung wegen eines möglichen Nutzens empfohlen wird.

Marken

SIMATIC®, SIMATIC HMI®und SIMATIC NET®sind eingetragene Marken der SIEMENS AG.

Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

Sicherheitstechnische Hinweise zu Ihrem Produkt:

Bevor Sie das hier beschriebene Produkt einsetzen, beachten Sie bitte unbedingt die nachfolgenden sicherheitstechnischen Hinweise.

Qualifiziertes Personal

Inbetriebsetzung und Betrieb eines Gerätes dürfen nur von **qualifiziertem Personal** vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieses Handbuchs sind Personen, die die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Bestimmungsgemäßer Gebrauch von Hardware–Produkten

Beachten Sie folgendes:

Warnung

Das Gerät darf nur für die im Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Fremdgeräten und -komponenten verwendet werden.

Der einwandfreie und sichere Betrieb des Produktes setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Bevor Sie mitgelieferte Beispielprogramme oder selbst erstellte Programme anwenden, stellen Sie sicher, dass in laufenden Anlagen keine Schäden an Personen oder Maschinen entstehen können.

EG–Hinweis: Die Inbetriebnahme ist so lange untersagt, bis festgestellt wurde, dass die Maschine, in die diese Komponente eingebaut werden soll, den Bestimmungen der Richtlinie 89/392/EWG entspricht.

Bestimmungsgemäßer Gebrauch von Software–Produkten

Beachten Sie folgendes:

Warnung

Die Software darf nur für die im Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Software–Produkten, Fremdgeräten und -komponenten verwendet werden.

Bevor Sie mitgelieferte Beispielprogramme oder selbst erstellte Programme anwenden, stellen Sie sicher, dass in laufenden Anlagen keine Schäden an Personen oder Maschinen entstehen können.

Vor der Inbetriebnahme

Beachten Sie vor der Inbetriebnahme folgendes:

Vorsicht

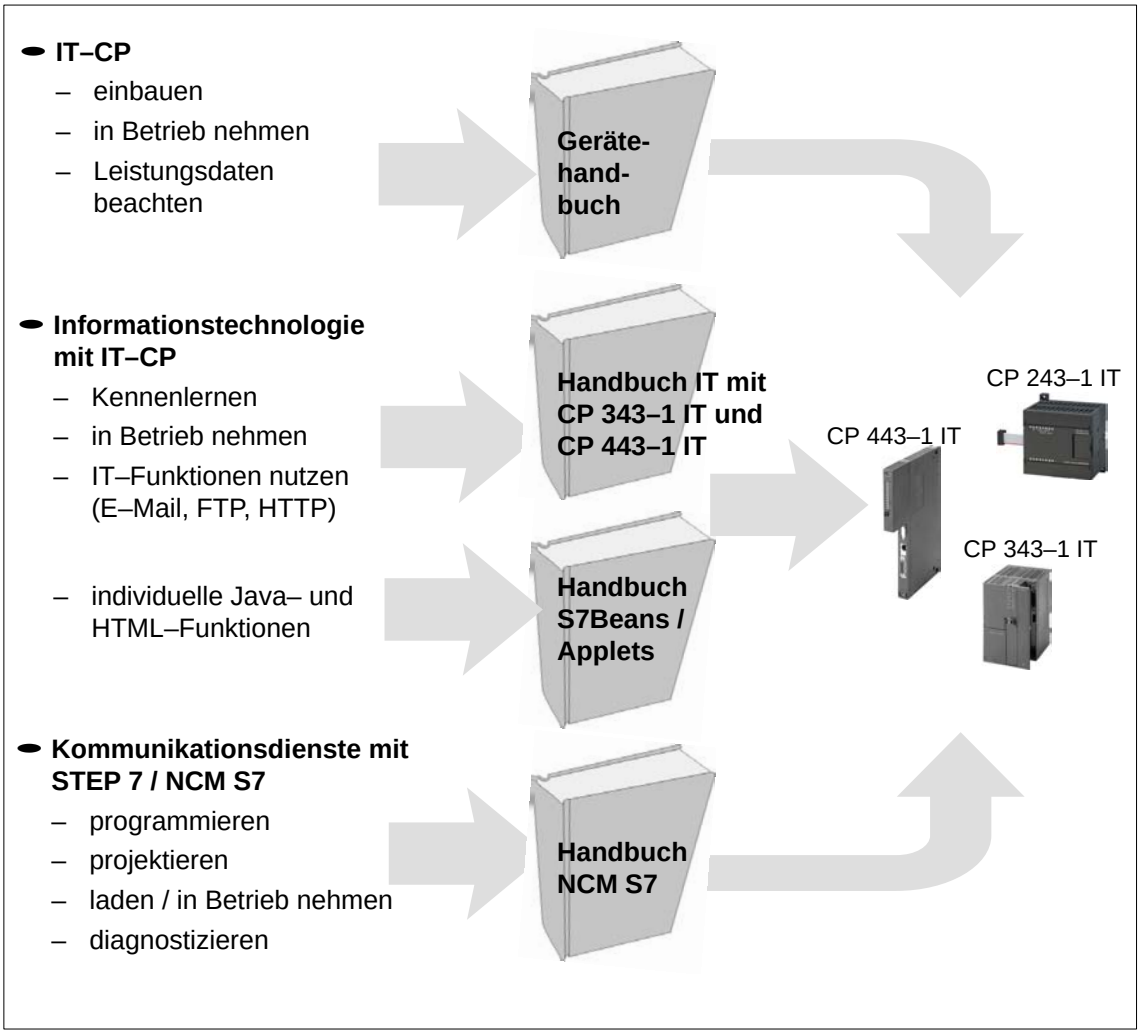
Vor der Inbetriebnahme sind die Hinweise in der entsprechenden aktuellen Dokumentation zu beachten. Die Bestelldaten hierfür entnehmen Sie bitte den Katalogen, oder wenden Sie sich an Ihre örtliche Siemens–Geschäftsstelle.

Siemens AG
Automation and Drives
Industrial Communication
Postfach 4848, D-90327 Nürnberg

Vorwort / Literaturhinweis

Handbücher zum Thema IT bei SIMATIC

Informationstechnologie mit IT–CPs bei SIMATIC wird in den folgenden Handbüchern beschrieben:



Auf der Manual Collection CD finden Sie die genannten Dokumente. Mit diesem Symbol werden Sie an einigen Stellen im Text darauf hingewiesen, dass Sie auf der Manual Collection CD weitere Zusätze und Beispiele finden.

Gültigkeitsbereich dieser Programmierhilfe

Dieses Dokument ist gültig

- für den CP 443–1 IT für die SIMATIC S7–400

  und

  für den CP 343–1 IT für die SIMATIC S7–300

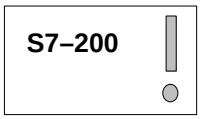
  – mit der zugehörenden Projektiersoftware STEP 7 ab Version 5.0 SP3 mit der Option NCM S7 für Industrial Ethernet;
- für den CP 243–1 IT für die SIMATIC S7–200

  – mit der zugehörenden Projektiersoftware STEP 7 MicroWin ab Version 3.2 SP1.
- S7 BeansAPI ab Version 2.5
- JBuilder ab Version 3.0

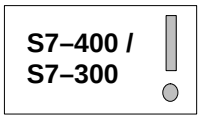
Lesehinweise

Achtung

Berücksichtigen Sie bitte in der folgenden Beschreibung die bei S7–200 teilweise von S7–300/S7–400 abweichenden technischen Merkmale, auf die wir Sie mit den folgenden Symbolen am Seitenrand aufmerksam machen.



So gekennzeichnete Stellen beschreiben Besonderheiten, die nur für S7–200 gelten.



So gekennzeichnete Stellen beschreiben Besonderheiten, die nur für S7–300 / S7–400 gelten.

Literaturhinweise /.../

Hinweise auf weitere Dokumentationen sind mit Hilfe von Literaturnummern in Schrägstrichen /.../ angegeben. Anhand dieser Nummern können Sie dem Literaturverzeichnis am Ende des Handbuchs den Titel der Dokumentation entnehmen.

❑

Inhalt

Auf der Manual Collection CD finden Sie die komplette Anleitung und die Programmierhilfe. Mit diesem Symbol werden Sie an einigen Stellen im Text darauf hingewiesen, dass Sie auf der Manual Collection CD weitere Zusätze und Beispiele finden.

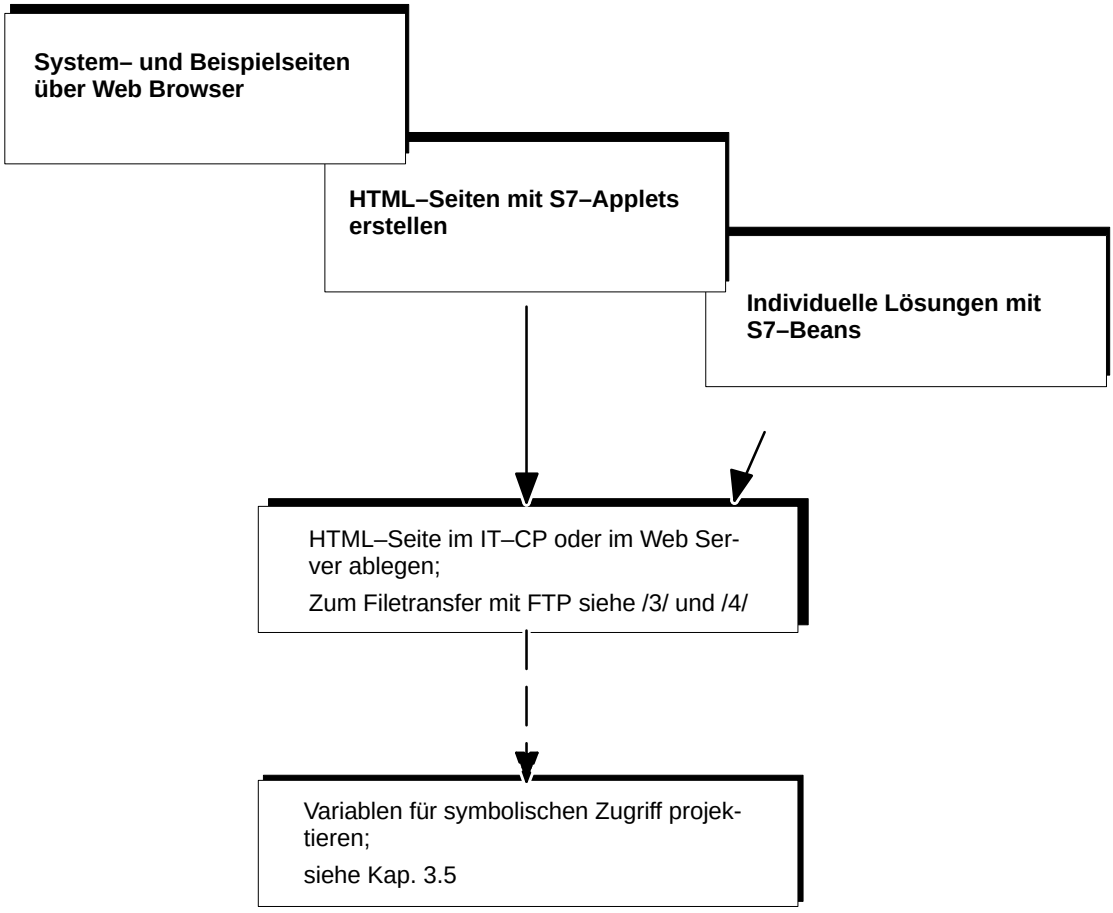
1 HTML–Seiten gestalten – Übersicht

Abgestuftes Konzept

Der IT–CP bietet mehrere Stufen, um eine Geräte– und Prozessdatenüberwachung mittels HTML–Seiten zu realisieren:

So gehen Sie vor...

...um Ihre individuellen HTML–Seiten zu erstellen oder anzupassen:



- System– und Beispielseiten über Web Browser

  Sie möchten ohne größeren Programmieraufwand die für den IT–CP vordefinierten Möglichkeiten der HTML–Prozesskontrolle nutzen.

  Die Möglichkeiten hierzu werden in /3/ vorgestellt.

- HTML–Seiten mit S7–Applets erstellen

  Der IT–CP gibt Ihnen vorgefertigte S7–Applets an die Hand, mit denen Sie HTML–Seiten erstellen und an Ihre Aufgabe anpassen können.

Die Aufrufe mit den zugehörenden Aufrufparametern werden im vorliegenden Handbuch zu den S7–Applets / Beans beschrieben.

- Individuelle Lösungen mit S7–Beans

  Sie möchten auf Ihre Anwendung zugeschnittene graphische Möglichkeiten nutzen und hierzu komplexere Applets bereitstellen.

  Sie möchten Ihre Prozessdaten nicht nur in Anlagenbildern darstellen sondern darüber hinausgehend programmtechnisch nutzen; beispielsweise zu einer Auswertung in einer Datenbank.

  Dies erreichen Sie, indem Sie folgende Möglichkeiten nutzen:

  – Anwendungsspezifische Java–Applets und Java–Applikationen erstellen und dabei sowohl vorgefertigte S7–Beans, Java–Beans anderer Hersteller und eigenen Java–Quelltext verwenden.

  – Java–Quellcode erstellen; dabei anwendungsspezifische Applets, Java-Beans und vorgefertigte S7–Beans verwenden.

2 Web Browser

2.1 Den IT–CP über Web–Browser erreichen

Anforderungsprofil

Für den Zugriff auf die HTML–Seiten im IT–CP oder im Web Server benötigen Sie einen Web Browser wie z.B. Netscape Navigator oder Internet Explorer. Der Web Browser muss folgende Voraussetzungen erfüllen:

- JDK (Java Development Kit) 1.1.X wird unterstützt; weitere Informationen siehe in /8/.

Der Netscape Navigator sowie Internet Explorer erfüllen diese Anforderungen. Web Browser mit entsprechendem Leistungsumfang können verwendet werden.

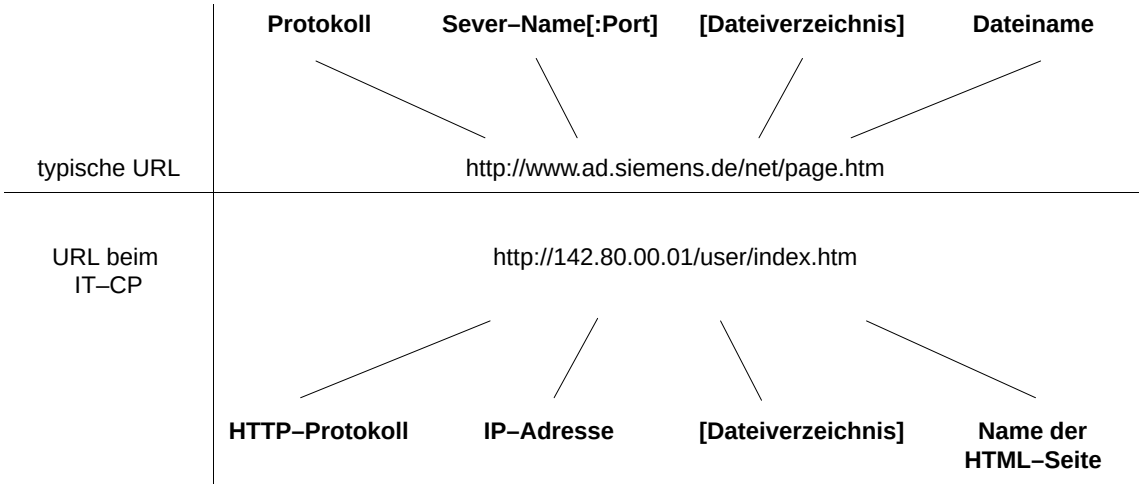
Entsprechende andere Web Browser erfüllen die genannten Anforderungen nur bedingt. Sie benötigen eine Plug–In Komponente um der Java Referenzimplementierung einer SUN Java Virtual Machine zu entsprechen.

Bitte beachten Sie weitere Angaben, auch zu den Versionen der hier genannten Produkte, in den Gerätehandbüchern /1/ auf der Manual Collection CD.

Hinweise und ggf. auch erforderliche Programmzusätze finden Sie darüberhinaus im Internet (siehe Anhang B).

URL: Uniform Resource Locator

Im World Wide Web hat sich die Adressierung über URL durchgesetzt. Auch den IT–CP erreichen Sie von Ihrem Web Browser aus über die URL. Diese URL kann nahezu beliebig komplex sein, besteht jedoch im Prinzip aus vier wesentlichen Teilen. Das folgende Schema verdeutlicht den Aufbau (typische URL) und gibt konkret die Inhalte für den Aufruf von IT–CPs an.



Beim Zugriff auf den IT–CP mittels Web Browser verwenden Sie das HTTP–Protokoll, um den Web Server auf dem IT–CP anzusprechen:

Die IP–Adresse teilen Sie dem CP über die Projektierung mit STEP 7 zu. Sofern ein Anschluss Ihres Industrial Ethernet zu Ihrem Intranet oder zum Internet hergestellt ist, ist der CP über die IP–Adresse im Intranet bzw. Internet zu erreichen.

Auf die detaillierte Struktur der IP–Adresse und auf die Möglichkeiten der Subnetzbildung über Subnetzmasken soll hier nicht weiter eingegangen werden. Detaillierte Informationen finden Sie in der Online–Hilfe von STEP 7 sowie in der weiterführenden Literatur z.B. in /7/.

2.2 Einstellungen im Web Browser

Allgemein

Bevor Sie auf den IT–CP über Ihren Web Browser zugreifen, müssen Sie einige Einstellungen vornehmen bzw. überprüfen. Am Beispiel des Netscape Navigator werden nachfolgend die Einstellungen erläutert.

Die hier gezeigten Einstellungen sind so gewählt, dass die Ausführung der im IT–CP verwendeten S7–Applets und S7–Beans (JavaBeans) ermöglicht wird.

Einstellungen im Netscape Navigator/Communicator

Beim Netscape Navigator sind die meisten Einstellungen in den Präferenzen vorzunehmen. Wählen Sie den Menübefehl **Bearbeiten ▸ Einstellungen... (Edit ▸ Preferences...)**

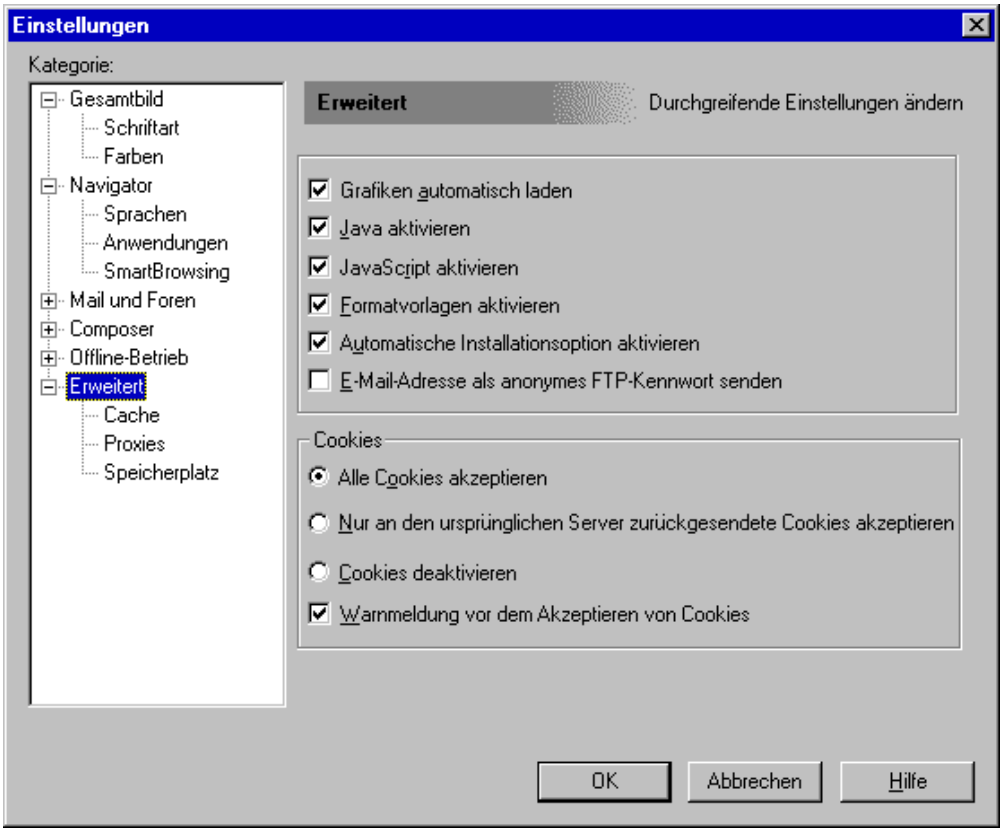


Bild 2-1

Empfehlenswert sind folgende Einstellungen:

- Java–Interpreter starten

  Indem Sie in “Kategorie – erweitert” (”Category – Advanced”) die Auswahl “Java aktivieren” (”Enable Java”) treffen, veranlassen Sie den Start des Java

Interpreters beim ersten Auftreten eines Applets in einer HTML Seite und lassen die Ausführung der Applets zu.

Hinweis

Wenn Sie die Einstellung bei bereits geladener HTML–Seite vornehmen, werden die in dieser HTML–Seite enthaltenen Java–Applets auch nach der Einstellung nicht ausgeführt. Erst wenn Sie die HTML–Seite neu laden, wird der Java Interpreter gestartet und die Java–Applets werden ausgeführt.

- Proxy–Server einstellen

  Fragen Sie hierzu gegebenenfalls Ihren Systemadministrator!

Einstellungen im Internet Explorer

- Java–Interpreter starten

  Sie finden die Funktionen für die Java–Anwendungen unter dem Menübefehl “Extras – Internetoptionen” im Register “Erweitert”; dort unter dem Eintrag “VM”.

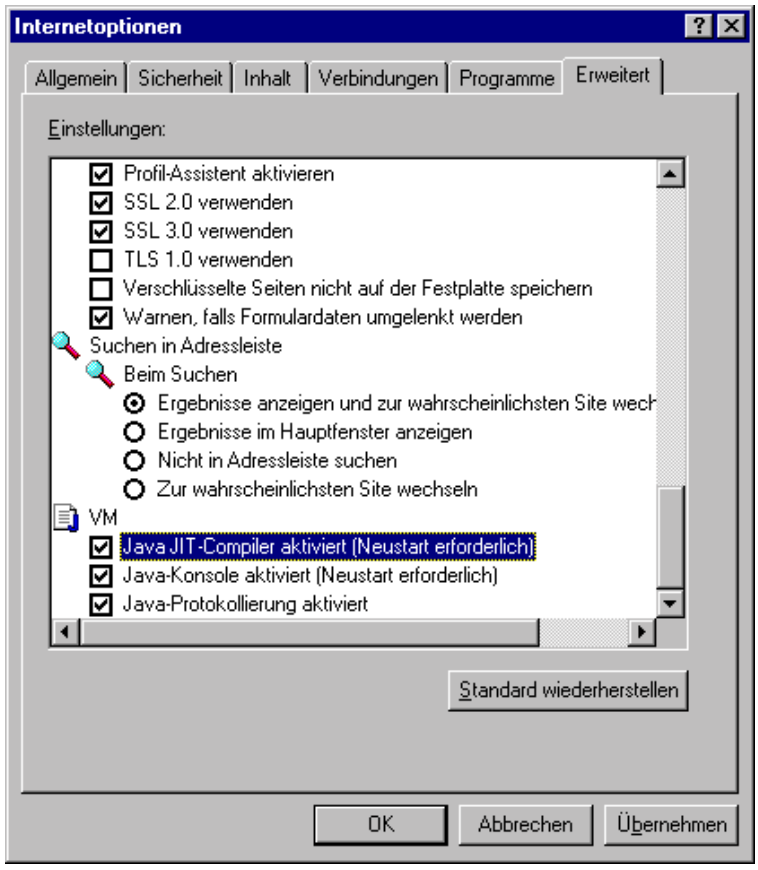


Bild 2-2

- Proxy–Server einstellen.

Fragen Sie hierzu gegebenenfalls Ihren Systemadministrator!

Java Konsole starten

Um den Ablauf der Java–Applet Bearbeitung zu verfolgen oder um Aufschlüsse über Störungen zu erhalten, können Sie die Java Konsole starten. Wählen Sie

- beim Netscape Navigator den Menübefehl **Communicator ▸ Extras ▸ Java–Konsole (Communicator ▸ Java)**;
- beim Internet Explorer verwenden Sie bitte die Option im Dialogfeld Internetoptionen Register “Erweitert” wie oben dargestellt. Um die Java Konsole aufzurufen, verwenden Sie bitte den Menübefehl **Ansicht ▸ Java Befehlszeile**

Anmerkung:

Beim IE muss die Java–Konsole vor einer möglichen Verwendung zuerst aktiviert werden. Hierzu im Dialog ”Extras – Internetoptionen” im Register ”Erweitert” unter ”Microsoft VM” die Option ”Java–Konsole aktiviert” einschalten.

❑

3 HTML–Seiten im IT–CP

Dieses Kapitel beantwortet die folgenden Fragen:

- Wie werden HTML–Seiten erstellt, die auf Informationen in der S7–Station zugreifen?
- Was sind S7–Applets und wie werden diese in HTML–Seiten genutzt? Worauf ist zu achten?
- Wo können selbst erstellte HTML–Seiten abgelegt werden?
- Wie ist eine graphische Darstellung von Prozessinformationen zu erhalten?
- Wie werden HTML–Seiten geprüft und getestet?

3.1 HTML–Seiten für den IT–CP erstellen

Nutzen

Erschließen Sie mit individuellen HTML–Seiten die Möglichkeit,

- an Ihre Anlage angepasste Prozessdarstellungen im Web Browser zu erhalten;
- Prozessdaten numerisch oder graphisch im Web Browser darzustellen;
- die Ergebnisse von Statusabfragen in die Darstellung einzubeziehen;
- die Abfrage und die Darstellung von Prozessdaten auch innerhalb einer HTML–Seite auf mehrere S7–Stationen und auf verteilte Anlagen auszudehnen.

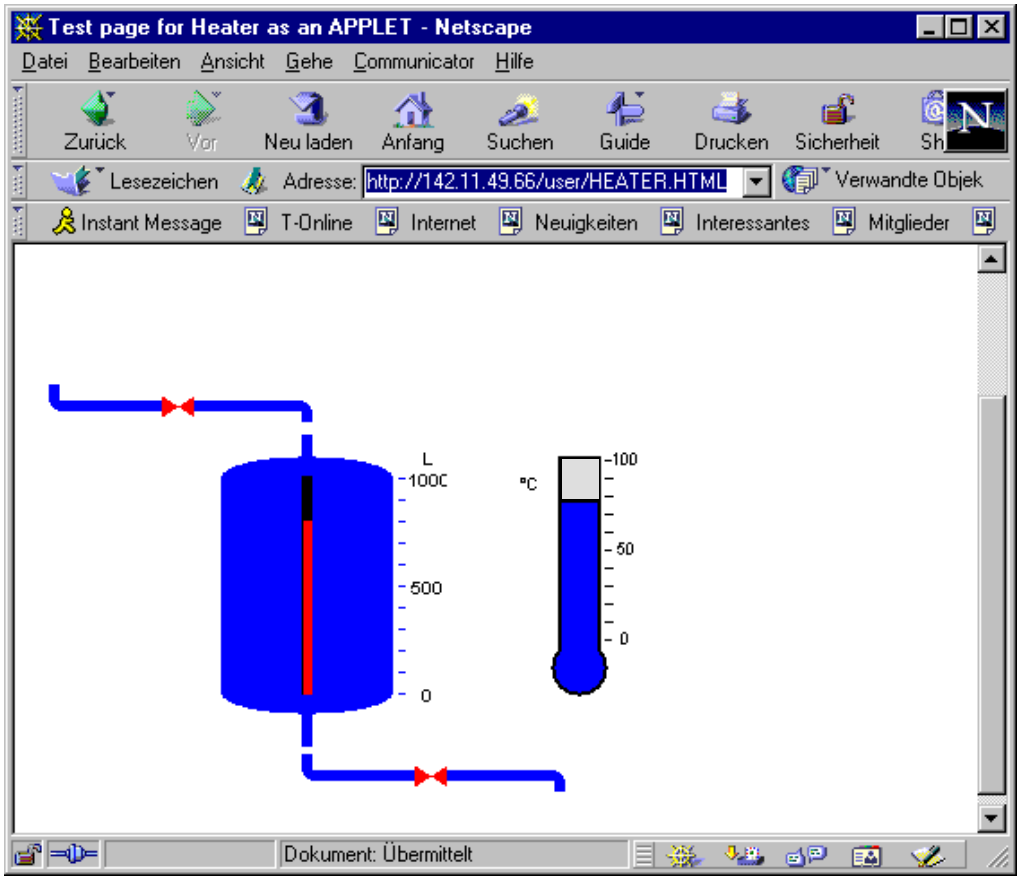


Bild 3-1

HTML–Editor

Für die Erstellung eigener HTML–Seiten benötigen Sie einen Editor. Die syntaktischen Konventionen für HTML–Seiten sind leichter zu beherrschen, wenn geeignete HTML–Editoren eingesetzt werden. Diese gestatten in der Regel die Eingabe von formatiertem Text und das Einbeziehen von Graphiken. Im Hintergrund erfolgt automatisch die Umsetzung in die HTML–Syntax. Ein Umschalten der Ansicht und eine direkte Eingabe in der HTML–Form ist meist ebenfalls möglich.

Beispiele:

- AOLPress

  Ein leistungsfähiger, als Shareware beziehbarer HTML–Editor, der die oben genannten Merkmale lückenlos aufweist.

- Netscape Composer

  Im Funktionsumfang mit AOLPress vergleichbarer HTML–Editor. Composer ist im Funktionsumfang des Netscape Communicator enthalten.

- FrontPage von Microsoft (nur mit Einschränkung verwendbar)

  Ein leistungsfähiger HTML–Editor. Bestimmte Optionen können jedoch zu spezifischen HTML–Codierungen führen, die nur mit dem Internet–Explorer von Microsoft verarbeitet werden können.

  Um mit FrontPage HTML–Codierungen zu erhalten, die unter möglichst vielen Browsern verarbeitet werden können, sollten Sie in FrontPage unter ”Extras – Seitenoptionen” im Register ”Kompatibilität” folgende Einstellungen tätigen:

  – Option ”Mit Microsoft FrontPage–Servererweiterungen” darf nicht aktiviert sein;

  – Option ”Active Server Pages (ASP)” darf nicht aktiviert sein;

  – Die Optionen ”ActiveX–Steuerelemente”, ”VBScript”, ”JavaScript” und ”Dynamic HTML” sowie ”Frames” und die verschiedenen Versionen der ”Cascaded Style Sheets (CSS)” sind browserspezifisch und damit nur eingeschränkt nutzbar;

  – Die Option ”Java–Applets” ist für den Einsatz der S7BeansApi und den daraus erstellten Applets unbedingt notwendig und muß aktiviert sein.

  Nach Fertigstellung der Seiten können diese mit Hilfe eines FTP–Clients oder mittels des FrontPage–Befehls ”Datei – Web veröffentlichen” zum IT–CP übertragen werden. Folgende Verzeichnisse und Unterverzeichnisse werden generell von FrontPage erstellt, jedoch auf dem IT–CP nicht benötigt:

  – ”_private”

  – ”_vti_cnf”

  – ”_vti_pvt”

  – ”_vti_bin”

  – ”_vti_log”

  – ”_vti_txt”

Diese Verzeichnisse können in Frontpage unter ”Ansicht – Berichte – Veröffentlichungsstatus” von der Übertragung durch FrontPage ausgeschlossen werden.

S7–Applets sind Applets für SIMATIC S7

Der IT–CP stellt einige Applets zur Verfügung, mit denen Sie Zugriffe auf die Steuerung vom Web Browser aus auf Ihrem PC ausführen können. Sie benötigen keine Java–Kenntnisse, um diese S7–Applets zu nutzen. Wenn Sie den weiteren Anweisungen folgen, werden Sie problemlos die Aufrufe in Ihre HTML–Seite integrieren.

Erweiterte Zugriffs– und Darstellungsmöglichkeiten – das JavaBeans–Konzept

Das JavaBeans–Konzept ermöglicht es, Objekte (Java Komponenten) zu erstellen und auf einfache Weise zu ausführbaren Programmen zu verbinden.

Für den IT–CP steht eine S7–Beans–Klassenbibliothek (S7BeansAPI) zur Verfügung. Die darin enthaltenen Objektklassen können Sie für einen objektorientierten Zugang zu unterschiedlichen Informationen der SIMATIC S7 und für eine graphische Darstellung von Prozessvariablen nutzen.

Mit der S7–Beans–Klassenbibliothek steht eine offene Schnittstelle zur Verfügung, die Ihnen Erweiterungen der Prozessdatenauswertung beispielsweise in Richtung Datenbanken, Tabellenkalkulation oder Management–Informationssysteme ermöglicht.

Dateien organisieren – Ressourcen des IT–CP

Der IT–CP stellt für die Ablage Ihrer HTML–Seiten Speicherplatz zur Verfügung. Hierzu finden Sie die Angaben im Gerätehandbuch zum IT–CP/1/.

Beachten Sie die Hinweise in der auf dem IT–CP befindlichen liesmich.htm–Datei.

S7–400 / S7–300

Am einfachsten gelangen Sie über den Link “Information” in der Homepage des IT–CP zur liesmich.htm–Datei.

Sie finden dort Informationen über die Bedeutung der standardmäßig mitgelieferten Dateien. Sie können entscheiden, welche Dateien für Ihre Anwendung sinnvoll sind. Mittels FTP–Funktionen (siehe Kap. 3) können Sie die Dateien im IT–CP nach Ihren Anforderungen organisieren.

3.2 HTML–Seiten gestalten – einige Grundlagen

Eine Anmerkung vorweg

Der Erstellung von HTML–Seiten widmet sich eine ganze Reihe empfehlenswerter Literatur. Beachten Sie auch die Literaturempfehlungen zum Thema Web, HTML usw. im Anhang zu dieser Anleitung. Die vorliegende Dokumentation beschränkt sich bewusst darauf, zu vermitteln, wie die mit dem IT–CP mitgelieferten Funktionen in Ihre HTML–Anwendung einbezogen werden können.

Sie sollen mit diesen Informationen in die Lage versetzt werden, ohne ausführliches Studium der HTML–Technik einfache HTML–Seiten zu erstellen und zu betreiben und dabei die S7–Applets zu nutzen.

Der erfahrene Ersteller von HTML–Seiten wird die Informationen zur Parametrierung der S7–Applets in den Folgekapiteln direkt anwenden können. Dem weniger erfahrenen Anwender sollen noch einige Hinweise zum Themenbereich gegeben werden. Eine Vertiefung ermöglicht die eingangs erwähnte weiterführende Literatur.

Die Struktur planen

Zu Beginn sollten Sie sich über Ihre Dokumenten– und Seitenstruktur klar werden. HTML ist die Technologie, die es ermöglicht, in den im Web Browser dargestellten HTML–Seiten von Thema zu Thema – im Falle des IT–CP von Steuerung zu Steuerung oder von Anlage zu Anlage – zu springen. Dementsprechend leben die HTML–Seiten von den Verweisen (Links), die Sie bei der Seitenerstellung festlegen müssen. Diese wiederum sind abhängig von der Ablage der HTML–Dokumente.

Dokumente (HTML–Seiten) verknüpfen

HTML–Seiten werden über Links verknüpft, die wie folgt aufgebaut sind:

- URLs mit absoluter Adressangabe

  Beispiel:
  <A HREF=”http://www.ad.siemens.de/net/index.htm”>Kennzeichnungs - text</A>

- URLs für vereinfachte Verknüpfungsinformation mit relativer Adressangabe:

  – Verwenden von relativen URLs: Diese enthalten in der Regel nur den Namen des Ordners und der Datei oder sogar nur den Dateinamen, wenn sich die aufgerufene HTML–Seite im selben Verzeichnis wie die aufrufende HTML–Seite befindet.

    Beispiel:
    <A HREF=”prozessbild.htm”>Kennzeichnungstext</A>

  – Verwenden von serverbezogenen URLs: diese Adressangaben beginnen mit einem ”/” und zeigen an, dass die gesuchte HTML–Seite auf dem selben Server wie die aufrufende HTML–Seite zu finden ist.

    Beispiel:
    <A HREF=”/bilder/prozessbild.htm”>Kennzeichnungstext</A>

HTML Seiten gestalten

Um Seiten benutzergerecht und ansprechend zu gestalten, werden Ihnen verschiedene Beschreibungselemente in HTML an die Hand gegeben. In der gängigen Literatur finden Sie entsprechende Informationen meist unter den folgenden Themenbereichen:

- Tabellen darstellen

  Die Tabelle ist ein wichtiges Mittel zur Informationsstrukturierung. Speziell für die Erstellung von HTML–Seiten ist es geeignet, einige Schwächen der HTML–Formatierung auszugleichen. Die spaltenweise Darstellung von HTML–Text ist nämlich nur über die Verwendung von Tabellen zu erreichen.

- Einfügen von Bildern

  Gerade der Verwendung von Bildern in HTML–Seiten widmet die Fachliteratur breiten Raum. Die in Frage kommenden Bilddateiformate sind GIF und JPG.

- HTML–Formulare (beim IT–CP nur in Verbindung mit JavaScript anwendbar)

  Formulare werden da benötigt, wo der Benutzer in einer standardisierten Form Informationen im System ablegen soll. HTML bietet insbesondere verschiedene Steuerelemente für die Interaktion mit dem Benutzer an. Auch für die Eingabe von Prozessdaten können die Formularfunktionen von Bedeutung sein.

- Formatvorlagen verwenden

  Formatvorlagen gestatten, allgemeingültige Formate festzulegen und in den HTML–Dokumenten verwenden. Eine solche Vorgehensweise ist Ihnen von Publishing–Systemen wie beispielsweise MS–Word oder Word Perfect bekannt.

  Erwähnt sei, dass es für die Verknüpfung von Formatvorlagen mit HTML–Doku-

menten verschiedene Techniken gibt.

- Frames erstellen

  Mittels Frames können Sie HTML–Seiten in mehrere Bereiche unterteilen. Dies kann die Übersichtlichkeit deutlich verbessern. Sie können beispielsweise dafür sorgen, dass das Navigationsmenü sichtbar bleibt, während neue Bereiche geladen werden.

  Beachten Sie auch die Hinweise zum Thema ”Anzahl der Applet–Instanzen in einer HTML–Seite ist begrenzt” im Kapitel 3.4.

- Applets einbetten

  Das ist das Thema des vorliegenden Kapitels, in dem es um die Verwendung der speziellen S7–Applets geht.

- JavaScript verwenden

  Mit JavaScript kann bei entsprechender Erfahrung des Erstellers die Funktion von HTML–Seiten und darin die Interaktion mit dem Benutzer erweitert werden.

Hinweis

Das Internet ist eine Fundgrube für die HTML–Seitengestaltung. Sie können jederzeit HTML–Seiten in Ihren HTML–Editor laden , speichern und als Vorlage für die eigene Seitengestaltung verwenden.

Beispielsweise können Sie HTML–Seiten, die Ihnen als Mustervorlagen geeignet erscheinen, auch im HTML–Editor aufrufen und anschließend speichern. Dabei werden sämtliche Daten der Seiten einschließlich der Graphiken gespeichert.

Beachten Sie jedoch, dass bei einigen HTML–Seiten das Layout über Copyright geschützt ist. Diese Seiten können natürlich nicht für die eigene Seitengestaltung verwendet werden.

HTML–Editor – Beispiele / Empfehlungen

Bereits im Kap.1.2 haben wir Ihnen die Voraussetzungen genannt, die ein HTML–Editor erfüllen sollte. Sie finden dort auch Empfehlungen für geeignete HTML–Editoren.

S7–Applets

Mit den S7–Applets beziehen Sie die Prozessdatendarstellung und die Eingabe von Prozessdaten in Ihre HTML–Seite ein. Den S7–Applets ist das Folgekapitel gewidmet.

3.3 Die eigene ”Homepage” gestalten und ablegen

Das Dateisystem des IT–CP flexibel nutzen

Die vorhandene Startseite bietet Grundfunktionen, die für viele Anforderungen genügt.

Tatsächlich bietet das Dateisystem des IT–CP jedoch ein flexibles Instrument für eine an Ihre Anlage angepasste Präsentation von Funktionen und Daten. Indem Sie eine eigene Startseite gestalten, haben Sie das Instrument, die Sicht auf Ihre gesamte Anlage oder darüberhinaus auszudehnen.

Sie können die vorhandene Startseite verändern oder durch Ihre eigene ”Homepage” ersetzen.

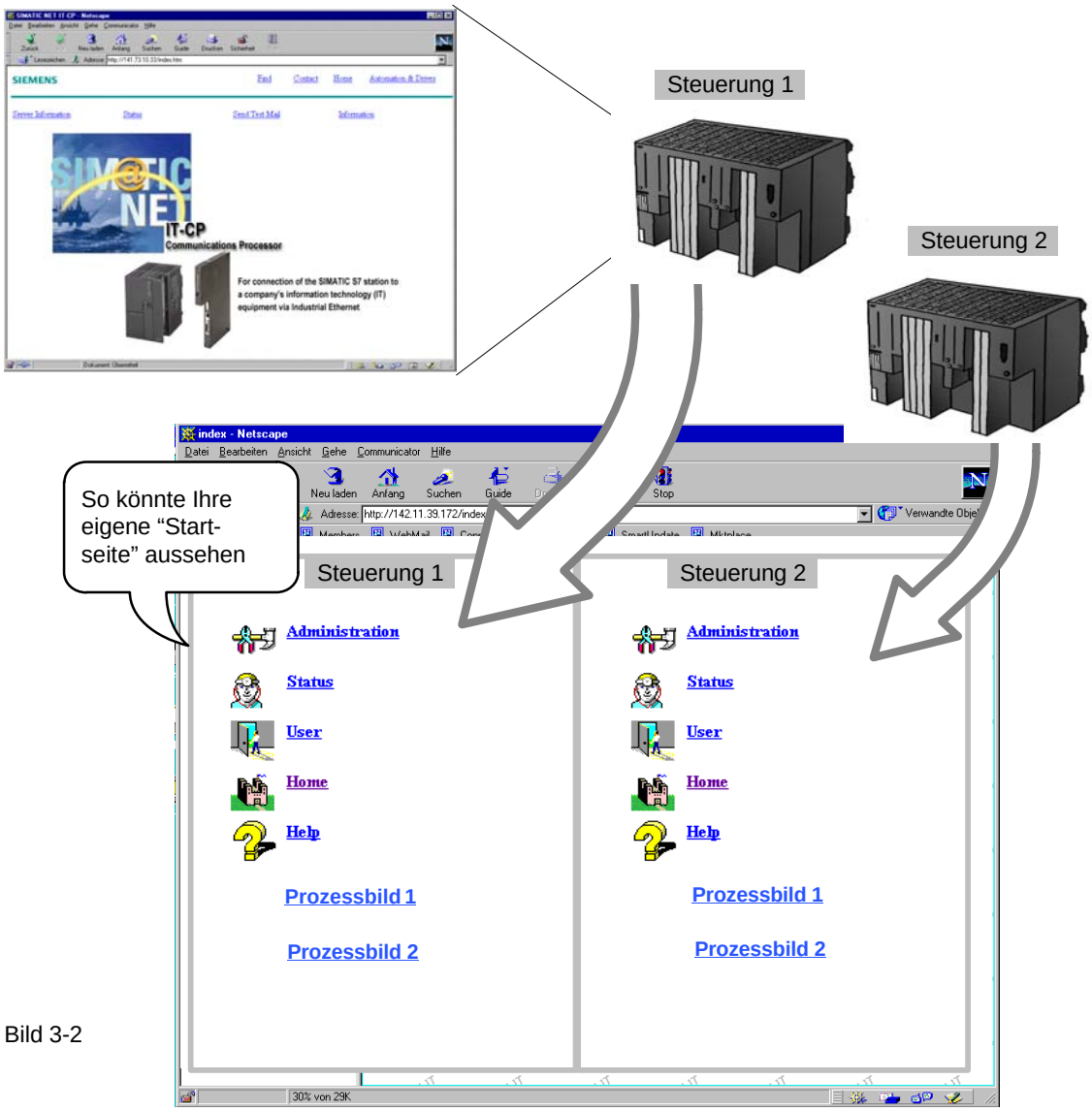


Bild 3-2

So können Sie vorgehen

Wenn Sie von der bestehenden Startseite ausgehen wollen, laden Sie diese in Ihren HTML–Editor und fügen dort Ihre zusätzlichen Anweisungen ein.

- Der online–Weg

  Sie laden die HTML–Startseite aus dem IT–CP in Ihren HTML–Editor und speichern diese zur weiteren Bearbeitung zunächst lokal in Ihrem PC.

- Der offline–Weg

Sie finden die HTML–Startseite auch auf der Manual Collection CD. Sie können so Ihre Startseite zunächst unabhängig von einem Zugang zum IT–CP gestalten und zu einem späteren Zeitpunkt in den IT–CP laden.

Zu beachten ist

Berücksichtigen Sie zu den folgenden Punkten die Angaben im Gerätehandbuch zum IT–CP /1/.

- Die Anzahl ablegbarer Dateien ist durch die Größe des Dateisystems begrenzt;
- Die Zeichenzahl in den anzugebenden URLs ist begrenzt;
- Die Länge der Dateinamen ist begrenzt.

S7–Applets einbeziehen

Der flexible Zugriff auf verteilte HTML–Systemseiten ist **ein** Aspekt der Homepage–Gestaltung.

Zusätzliche Möglichkeiten der Informationsabfrage gewinnen Sie, wenn Sie in Ihre HTML–Seiten die S7–Applets und S7–Beans einbeziehen, denen die weiteren Kapitel gewidmet sind.

Beispiele

Beispiele für spezifisch gestaltete HTML–Seiten finden Sie sowohl auf der Manual Collection CD.

S7–400 / S7–300

Beispiele finden Sie auch im CP–Filesystem, dort im Verzeichnis /examples/.

HTML–Seiten laden

Verwenden Sie einen FTP–Client und die FTP–Dateiverwaltungsfunktionen wie in /3/ und /4/ beschrieben, um die vorhandenen HTML–Seiten durch weitere zu ergänzen oder zu ersetzen.

3.4 S7–Applets

Bedeutung

S7–Applets sind spezielle Applets, die über den IT–CP lesende und schreibende Zugriffe auf eine S7–Station ermöglichen.

Für die Bearbeitung der Applets ist immer der Web Browser zuständig, in dem das Applet gestartet wurde. Dieser aktiviert das Applet und weist ihm entsprechend der Parametrierung einen Rahmen innerhalb der aktuellen HTML–Seite zu.

S7–400 / S7–300

Das folgende Beispiel zeigt den Fall, dass alle mitgelieferten S7–Standard–Applets innerhalb einer HTML–Seite verwendet werden. Es ist zu erkennen, dass die S7–Applets hier in einer HTML–Tabelle eingebettet sind:

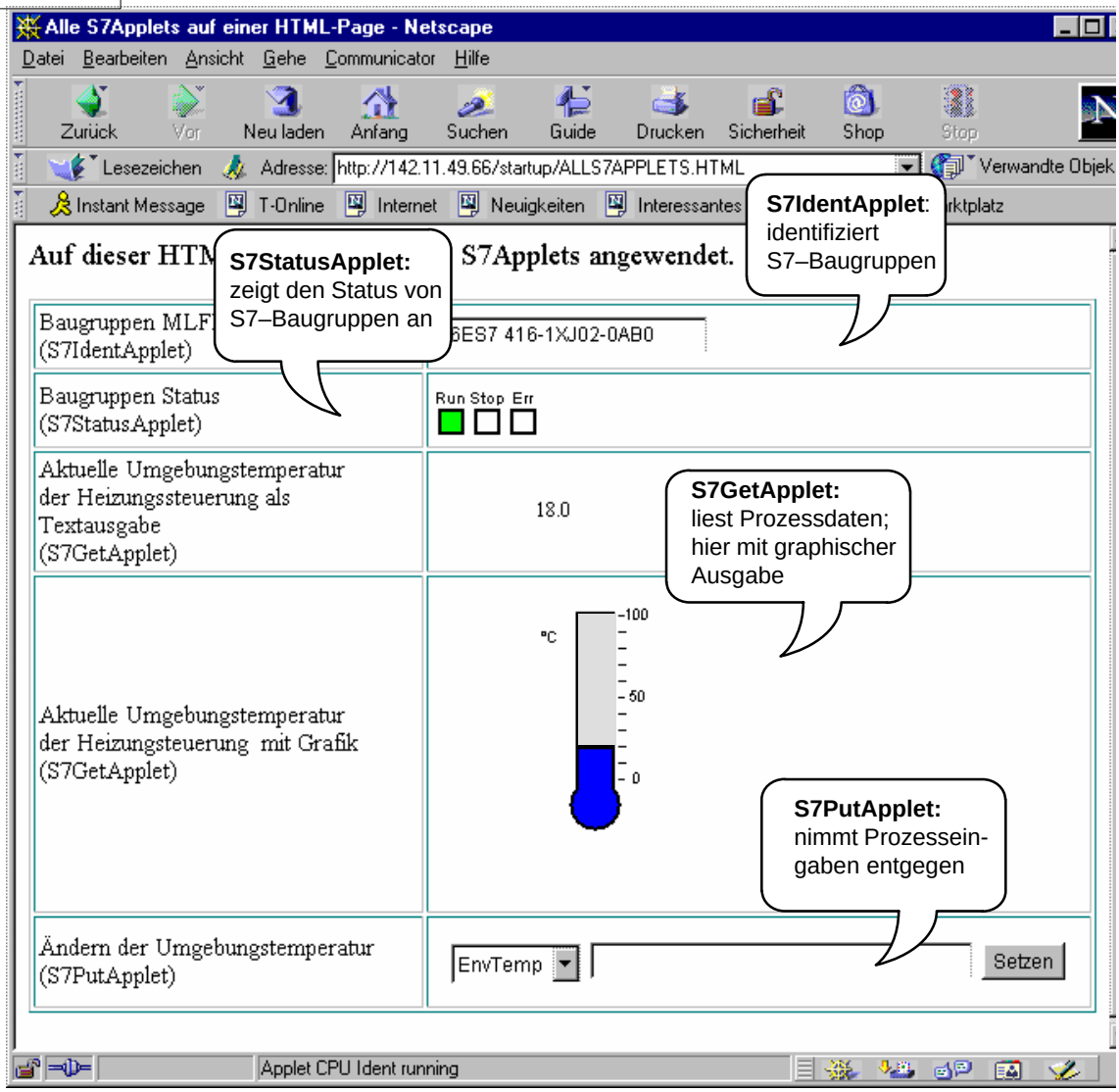


Bild 3-3

Weitere Informationen gibt die folgende Tabelle. Die anschließenden Kapitel be-

schreiben im Detail, wie Sie die S7–Applets verwenden und parametrieren.

Tabelle 3-1

S7–Applet	Bedeutung
S7IdentApplet (nur S7–300/400)	S7–Baugruppen anhand der Bestellnummer und dem Ausgabestand identifizieren;
S7StatusApplet (nur S7–300/400)	Den Status von S7–Baugruppen anzeigen; Beispiel: Run/Stop
S7GetApplet	Prozessdaten zyklisch lesen; z.B. Merkerwort oder Daten in Datenbaustein. Die Prozessdaten • werden symbolisch oder absolut adressiert; • können grafisch dargestellt werden. Für die graphische Ausgabe werden JavaBeans verwendet.
S7PutApplet	Prozessdaten in den HTML–Seiten eingeben und in die Steuerung übertragen/schreiben; z.B. Merkerwort oder Daten in Datenbaustein. Die Prozessdaten werden symbolisch oder absolut adressiert.

So spielen die Komponenten zusammen

Die S7–Applets übermitteln vom Web Browser aus spezielle Lese– oder Schreibaufträge an den IT–CP. Der IT–CP leitet diese Aufträge je nach Anforderung an die jeweilige Baugruppe oder CPU weiter.

Die S7–Applets können Meldungen über Aktionen und Fehlerzustände in der Java Konsole ausgeben (siehe Kap. 3.6). So erhalten Sie Informationen über den aktuellen Status der Bearbeitung.

Die Anzahl der Applet–Instanzen in einer HTML–Seite ist begrenzt

Beachten Sie, dass die Anzahl der in einer HTML–Seite aufrufbaren Applets begrenzt ist! Die mögliche Anzahl hängt vom verwendeten Web Browser und der Systemumgebung (z.B. dem Betriebssystem) ab. Informieren Sie sich bitte anhand der Dokumentation des Web Browsers.

Umgehungsmöglichkeit durch S7–Beans:

Für komplexere Darstellungen sollten Sie die Möglichkeiten nutzen, die Ihnen die S7–Beans–Klassenbibliothek bietet. In individuell erstellten Applets können Sie mittels der S7–Beans auch auf eine größere Anzahl von Prozessgrössen zugreifen. Eine Mengenbegrenzung der Applets können Sie dadurch umgehen.

3.4.1 Applet–Aufruf und Parametrierung

S7–Applets in der HTML–Seite aufrufen

S7–Applets besitzen, wie alle Java–Programme, die Dateinamenserweiterung ”class”. Der Applet–Aufruf wird innerhalb der HTML–Seite mit einem entsprechenden HTML–Tag eingebettet (siehe Tabelle unten). Der Aufruf zur Identifikation einer S7–Baugruppe in einer S7–Station wird beispielsweise wie folgt im Applet–Tag angegeben:

<Applet CODE=”de.siemens.simaticnet.itcp.applets.S7IdentApplet.class” ...>

Die dargestellte Zuweisung legt den Namen oder die Adresse der Datei mit dem Applet fest. Die hier verwendete Schreibweise kennzeichnet eine relative Adresse. Sämtliche Applets sind zu einem jar–File zusammengefasst. Wird im Aufruf zusätzlich das Attribut CODEBASE verwendet, befindet sich das Applet in dem Verzeichnis, das unter CODEBASE angegeben wird.

Beispiel:

Mit dem folgenden Aufruf wird eine S7–Baugruppe identifiziert, die sich im Rack 0 am Steckplatz 3 einer S7–Station befindet. Die gelesenen Informationen werden in der HTML–Seite in schwarzer Schrift auf grünem Hintergrund dargestellt.

```
<APPLET CODE=”de.siemens.simaticnet.itcp.applets.S7IdentApplet.class”
CODEBASE=”/applets/”

ARCHIVE=”s7applets.jar, s7api.jar” NAME=”s7_MLFB” WIDTH=150
HEIGHT=35>

<PARAM name=”RACK” value=0>
<PARAM name=”SLOT” value=3>
<PARAM name=”BACKGROUNDCOLOR” value=”0x00FF00”>
 </APPLET>
```

Weitere Verwendungsbeispiele finden Sie anschließend bei der Beschreibung der einzelnen S7–Applets.

Allgemeiner Teil der Parametrierung

Neben dem Namen des S7–Applets müssen Sie einige allgemeine Attribute und Parameter angeben. Zusätzlich zu den allgemeinen Attributen und Parametern, die für jedes S7–Applet zu verwenden sind, gibt es funktionsabhängige Attribute und Parameter. Diese werden bei den jeweiligen S7–Applets beschrieben.

Die folgenden Tabellen geben die HTML–Tags, die Attribute und Parameter an, die bei den meisten S7–Applets zu belegen sind:

Tabelle 3-2 Allgemeine HTML–Tags bei der Verwendung von Applets

HTML–Tag und Attribut	Bedeutung
<APPLET...>...</APPLET>	Dieses Tag bettet ein Applet in einem HTML–Dokument ein. Das Applet wird durch Attribute und Parameter spezifiziert. Beispiel siehe oben.
<PARAM name="..." value="...">	Dieses Tag kennzeichnet Applet–Parameter. Jeder Parameter wird über einen Namen identifiziert und jedem Parameter wird ein Wert zugewiesen. Beispiel: <PARAM name="RACK" value=0>

Tabelle 3-3 Allgemeine Attribute der S7–Applets

Attribut Name	Typ	Beschreibung
CODE	string	Das Attribut bezeichnet das aufzurufende Applet. Anzugeben sind jeweils • Package–Pfad = konstant • Applet–Typ = variabel Beispiel: CODE="de.siemens.simaticnet.itcp.applets.S7IdentApplet.class"
CODEBASE	string	Das Attribut bezeichnet den Pfad, unter dem die Applet–Datei oder das Applet–Archiv (siehe ARCHIVE) abgelegt ist. Beispiel: CODEBASE="/applets/"
ARCHIVE	string	Das Attribut benennt das Applet–Archiv, in dem das Applet enthalten ist. Beispiel: ARCHIVE="s7applets.jar, s7api.jar"
NAME	string	Eindeutiger Applet Name; Über diesen Namen können beispielsweise Ausgaben in der Java Konsole dem Applet zugeordnet werden. Beispiel: NAME="s7_MLFB"
WIDTH	int	Breite der Darstellung in der HTML–Seite; Legt die Darstellungsbreite in der HTML–Seite fest. Der Wert muss ausreichend groß für die Ausgabe gewählt werden. Beispiel: WIDTH=150
HEIGHT	int	Höhe der Darstellung in der HTML–Seite; Legt die Darstellungshöhe in der HTML–Seite fest. Der Wert muss ausreichend groß für die Ausgabe gewählt werden. Beispiel: HEIGHT=35

Tabelle 3-4 Allgemeine Parameter der S7–Applets

Parameter Name	Typ	Beschreibung
BACKGROUNDCOLOR	string	Hintergrundfarbe (24 bit RGB in hexadezimal) Der Parameter steuert die Farbintensität für die Farbanteile RGB. Zur Hilfestellung hier folgende Eckwerte: weiß: 0xFFFFFF schwarz: 0x000000 rot: 0xFF0000 grün: 0x00FF00 blau: 0x0000FF Hinweis: Sie können die Auswirkung der Einstellung über die Eingabehilfe direkt testen (siehe Kap.3.4.2).

3.4.2 Parametrierhilfen

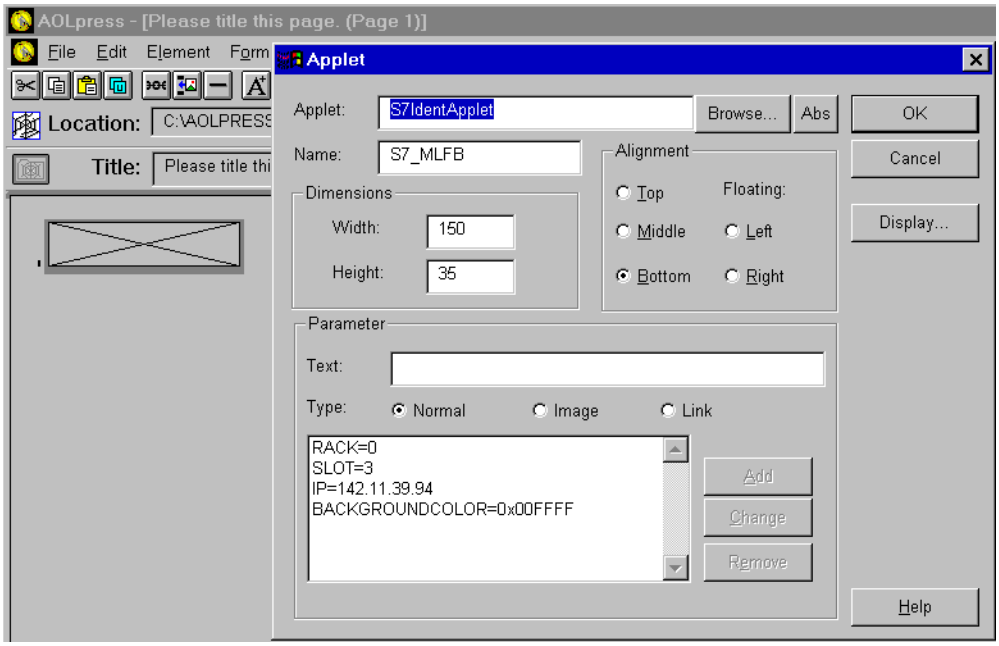
Um Sie bei der Parametrierung der S7–Applets zu unterstützen, können Sie auf verschiedene Parametrierhilfen zurückgreifen. Diese gewährleisten eine sichere, syntaktisch korrekte Parametrierung der S7–Applets. Nachfolgend werden beschrieben:

- Eingabehilfe beim HTML–Editor
- Online–Parametrierung für Testzwecke

Die auf der Manual Collection CD und auf dem IT–CP im Verzeichnis Examples verfügbaren HTML–Seiten stellen darüberhinaus einen einfachen, effizienten Weg dar, per “Copy und Paste” vorhandene Aufrufparametrierungen wiederzuverwenden!

Eingabehilfe beim HTML–Editor

Einige HTML–Editoren wie beispielsweise AOLPress, bieten Eingabehilfen zum syntaktisch korrekten Aufrufen und Parametrieren von Java–Applets an. Nachfolgend eine beispielhafte Darstellung eines Dialogfeldes zur Applet–Parametrierung bei AOLPress.



Anmerkung: Der Nachteil dieser Eingabehilfe besteht darin, dass Sie die Parameterbezeichnungen nach wie vor korrekt **eingeben** müssen.

Online–Parametrierung für Testzwecke

Die S7–Applets bieten die Möglichkeit einer Online–Parametrierung. Durch Doppelklick auf das Ausgabefeld in der HTML–Seite ist es dort möglich, die aktuellen Parameter des bereits aktivierten S7–Applets in einem Dialogfeld zu verändern.

Die Online–Parametrierung kann über den Applet–Parameter EDIT ein– oder ausgeschaltet werden. Wird der Parameter im Applet–Aufruf nicht verwendet, ist die Online–Parametrierung **standardmäßig ausgeschaltet**!

Beispiel: In einem Anlagenbild wird per Doppelklick auf den Füllstand eines Kessels der Dialog für die Online–Parametrierung des zugehörenden S7–Applets aufgerufen.

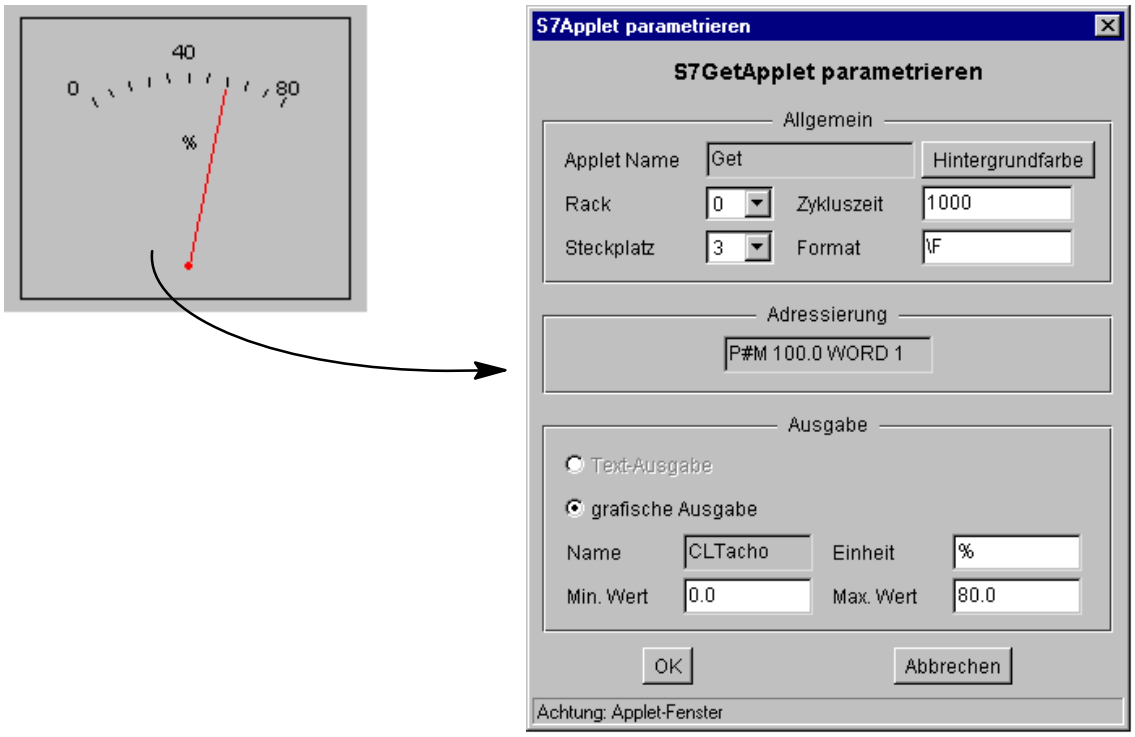


Bild 3-4

Diese Funktion dient in erster Linie zu Erprobungszwecken. Sollen die geänderten Parameter über den Aufruf der HTML–Seite hinaus erhalten bleiben, müssen diese in der HTML–Seite mittels eines HTML–Editors übernommen werden.

Die Zugriffsrechte haben auch bei einer Online–Parametrierung ihre Gültigkeit!

Hinweis

Einstellungen bleiben nur so lange gültig, wie kein HTML–Seitenwechsel erfolgt. (Bei bestimmten Browsern kann bereits das Ändern der Fenstergröße zu einem Applet–Neustart und Neuinitialisieren der Parameter führen.)

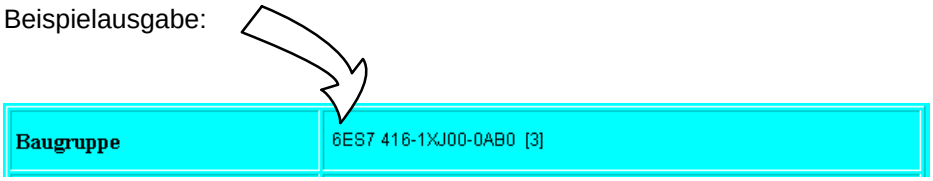
3.4.3 S7IdentApplet – Beschreibung

S7–400 / S7–300

Bedeutung

Dient zur Identifizierung einer S7–Baugruppe in einer S7–Station. Das Applet liest die Bestellnummer und den Ausgabestand der angegebenen S7–Baugruppe.

Beispielausgabe:



Aufruf–Tags

CODE=”de.siemens.simaticnet.itcp.applets.S7IdentApplet.class”
CODEBASE=”/applets/”
ARCHIVE=”s7applets.jar, s7api.jar”

Parametrierung

Zusätzlich zu den allgemeinen Parametern (siehe Kap. 3.4.1) sind die folgenden funktionsspezifischen Parameter zu belegen:

Tabelle 3-5 Applet spezifische Parameter

Parameter Name	Typ	Beschreibung
SLOT	byte	Slot Nummer (Steckplatz) der angesprochenen Baugruppe (1..18)
RACK	byte	Rack Nummer der angesprochenen Baugruppe (0..7)

Tabelle 3-6 optionale Applet spezifische Parameter

EDIT	bool	Die Online–Parametrierung kann ein– oder ausgeschaltet werden. Parametriermöglichkeit ein = true aus = false Wird der Parameter im Applet–Aufruf nicht verwendet, ist die Online–Parametrierung **standardmäßig ausgeschaltet!**
LANGUAGE	string	Die für die Beschriftungen, Meldungen und Diagnoseanzeigen verwendete Sprache kann fest eingestellt werden. Unterstützt werden zur Zeit die Sprachen Deutsch und Englisch. Als Parameter werden die zweistelligen ISO–639 Codes benutzt. Parametriermöglichkeit: Deutsch = “de” Englisch = “en” Wird der Parameter im Applet–Aufruf nicht verwendet, wird die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt (kann bei Windows–Betriebsystemen unter Systemsteuerung – Ländereinstellungen geändert werden). Wird eine nicht unterstützte Sprache angegeben, so wird versucht die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt.
DEBUGLEVEL	int	Stellt den Detail–Level der Debug–Ausgabe in der Java–Konsole ein. Parametriermöglichkeit: 0 = Keine Ausgabe 1 = Alle Meldungen ausgeben 2 = Nur Warn– und Fehlermeldungen ausgeben 3 = Nur Fehlermeldungen ausgeben 4 = Nur Meldungen fataler Fehler ausgeben Wird der Parameter im Applet–Aufruf nicht verwendet, so wird der Level 3 verwendet, d.h. es werden nur Fehlermeldungen ausgegeben.

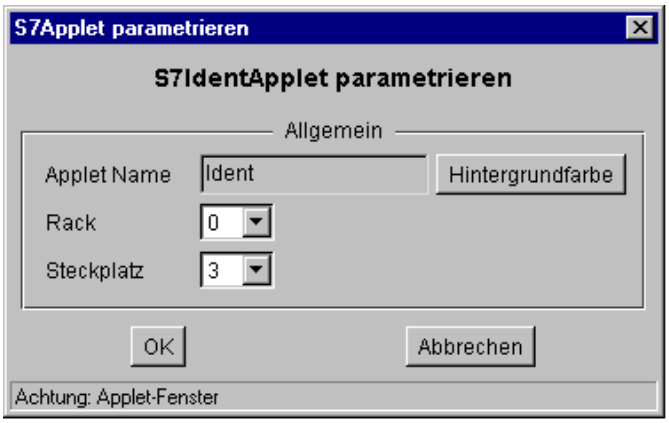
Zugriffsrechte

Unter dem beim Zugriff verwendeten Benutzernamen muss folgendes Zugriffsrecht eingeräumt sein (siehe Dialog “Bearbeiten Benutzereintrag in Kap. 1.4):

- “die Bestellnummer von Baugruppen abzufragen”;

Parametrierhilfen (Bedeutung und Anwendung siehe Kap. 3.4.2)

Die **Online–Parametrierung** für Testzwecke wird unterstützt. Doppelklicken Sie hierzu auf das Ausgabefeld um den Parametrierdialog zu öffnen.



3.4.4 S7IdentApplet – Beispiel

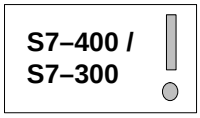
Mit dem hier gezeigten Beispiel ermitteln Sie den Identifikations–Code einer im Rack 0 auf dem Steckplatz 3 vorhandenen S7–Baugruppe und geben diesen numerisch aus.

```
<APPLET CODE=”de.siemens.simaticnet.itcp.applets.S7IdentApplet.class”
CODEBASE=”/applets/”
ARCHIVE=”s7applets.jar, s7api.jar” NAME=”s7_MLFB” WIDTH=150 HEIGHT=35>
<PARAM name=”RACK” value=0> <PARAM name=”SLOT” value=3>
<PARAM name=”BACKGROUNDCOLOR” value=”0x00FFFF”>
<PARAM name=”EDIT” value=”true”>
 </APPLET>
```

Ergebnis:

Baugruppe	6ES7 416-1XJ00-0AB0 [3]

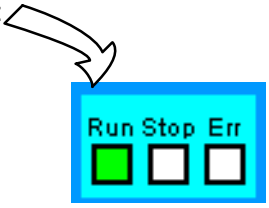
3.4.5 S7StatusApplet – Beschreibung



Bedeutung

Ermittelt Zustandsinformationen zur angegebenen Baugruppe.

Beispielausgabe:



Datenausgabe / Anzeige

Die Ausgabe erfolgt graphisch und mit Zusatztext. Die folgende Tabelle gibt Aufschluss.

Tabelle 3-7

Anzeige / Farbe	Zusatztext	Bedeutung
grün	Run	Das Anwenderprogramm läuft.
gelb	Stop	Das Anwenderprogramm ist angehalten.
grau	unbekannt	Die Verbindung zum IT–CP ist aufgebaut; warten auf Antwort.
rot	Fehler	Es liegt eine Fehlermeldung der Baugruppe vor.
blau	Fehler	Es besteht keine Verbindung zur angesprochenen Baugruppe.

Zugriffsrechte

Unter dem beim Zugriff verwendeten Benutzernamen muss folgendes Zugriffsrecht eingeräumt sein (siehe Dialog “Bearbeiten Benutzereintrag in Kap. 1.4):

- “den Status von Baugruppe abzufragen”;

Aufruf–Tags

CODE=”de.siemens.simaticnet.itcp.applets.S7StatusApplet.class”
CODEBASE=”/applets/”
ARCHIVE=”s7applets.jar, s7api.jar”

Parametrierung

Zusätzlich zu den allgemeinen Parametern (siehe Kap. 3.4.1) sind die folgenden funktionsspezifischen Parameter zu belegen:

Tabelle 3-8 Applet spezifische Parameter

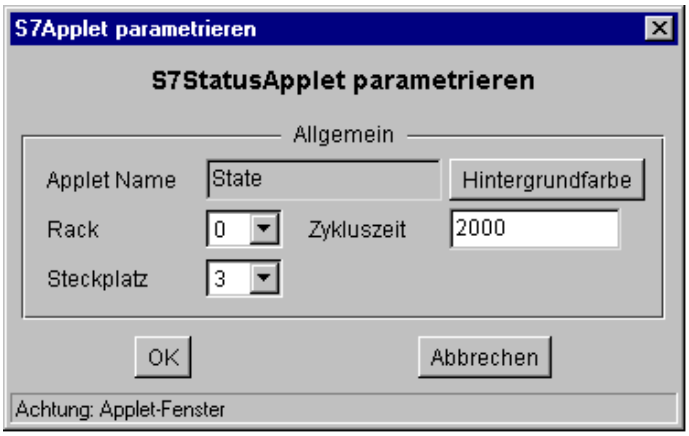
Parameter Name	Typ	Beschreibung
SLOT	byte	Slot Nummer (Steckplatz) der angesprochenen Baugruppe; (1–18)
RACK	byte	Rack Nummer der angesprochenen Baugruppe; (0–7)
CYCLETIME (Zykluszeit)	int	Zykluszeit für den Leseauftrag; Angabe in Millisekunden. >5000 (empfohlener Wert)

Tabelle 3-9 optionale Applet spezifische Parameter

EDIT	bool	Die Online–Parametrierung kann ein– oder ausgeschaltet werden. Parametriermöglichkeit ein = true aus = false Wird der Parameter im Applet–Aufruf nicht verwendet, ist die Online–Parametrierung **standardmäßig ausgeschaltet!**
LANGUAGE	string	Die für die Beschriftungen, Meldungen und Diagnoseanzeigen verwendete Sprache kann fest eingestellt werden. Unterstützt werden zur Zeit die Sprachen Deutsch und Englisch. Als Parameter werden die zweistelligen ISO–639 Codes benutzt. Parametriermöglichkeit: Deutsch = “de” Englisch = “en” Wird der Parameter im Applet–Aufruf nicht verwendet, wird die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt (kann bei Windows–Betriebsystemen unter Systemsteuerung – Ländereinstellungen geändert werden). Wird eine nicht unterstützte Sprache angegeben, so wird versucht die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt.
DEBUGLEVEL	int	Stellt den Detail–Level der Debug–Ausgabe in der Java–Konsole ein. Parametriermöglichkeit: 0 = Keine Ausgabe 1 = Alle Meldungen ausgeben 2 = Nur Warn– und Fehlermeldungen ausgeben 3 = Nur Fehlermeldungen ausgeben 4 = Nur Meldungen fataler Fehler ausgeben Wird der Parameter im Applet–Aufruf nicht verwendet, so wird der Level 3 verwendet, d.h. es werden nur Fehlermeldungen ausgegeben.

Parametrierhilfen (Bedeutung und Anwendung siehe Kap. 3.4.2)

Die **Online–Parametrierung** für Testzwecke wird unterstützt. Doppelklicken Sie hierzu auf das Ausgabefeld um den Parametrierdialog zu öffnen.



3.4.6 S7StatusApplet – Beispiel

Mit dem hier gezeigten Beispiel, geben Sie den Status einer im Rack 0 auf dem Steckplatz 3 vorhandenen S7–Baugruppe graphisch aus.

```
<APPLET CODE=”de.siemens.simaticnet.itcp.applets.S7StatusApplet.class”
ARCHIVE=”s7applets.jar, s7api.jar” NAME=”s7_status_3” WIDTH=80
HEIGHT=20>
<PARAM name=”RACK” value=0>
 <PARAM name=”SLOT” value=3>
<PARAM name=”CYCLETIME” value=5000>
<PARAM name=”BACKGROUNDCOLOR” value=”0xFFFFFF”>
<PARAM name=”EDIT” value=”true”>
 </APPLET>
```

Ergebnis:

3.4.7 S7GetApplet – Beschreibung

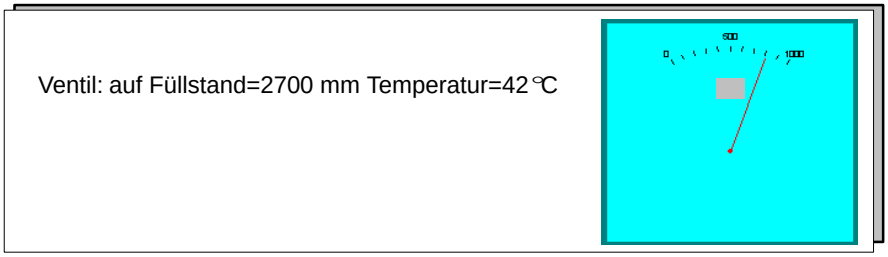
Bedeutung

Das Applet liest zyklisch Daten bzw. Datenbereiche entsprechend der Parametrierung aus der S7–CPU. Die Benennung der Variablen kann symbolisch (nicht bei S7–200) oder durch Adressangabe erfolgen.

Mit Hilfe eines Formatstrings, der später ausführlich erläutert wird, legen Sie die Ausgabeform für die Daten fest.

Die Prozesswerte können alternativ numerisch oder graphisch angezeigt werden.

Beispielausgabe (numerisch und graphisch):



Wie Sie die graphische Darstellung nutzen und die JavaBeans verwenden, darüber informiert Kap. 4.

Voraussetzungen

Die Benennung der Variablen kann symbolisch oder durch Adressangabe erfolgen.

Der symbolische Zugriff (nicht bei S7–200) setzt eine entsprechende Symbol–Projektierung im IT–CP voraus. Bei symbolischem Zugriff werden die Zugriffsrechte entsprechend der Variablenprojektierung geprüft (siehe Kap. 3.5).

Zugriffsrechte

S7–400 / S7–300

Unter dem beim Zugriff verwendeten Benutzernamen muss folgendes Zugriffsrecht eingeräumt sein (siehe Dialog “Bearbeiten Benutzereintrag” in Kap. 1.4):

- “auf die projektierten Symbole zuzugreifen“ (bei symbolischem Zugriff);
- ”Variablen über absolute Adressen zu lesen” (bei absolutem Zugriff);

Aufruf–Tags

CODE=”de.siemens.simaticnet.itcp.applets.S7GetApplet.class”
CODEBASE=”/applets/”
ARCHIVE=”s7applets.jar, s7api.jar”

Parametrierung

Zusätzlich zu den allgemeinen Parametern (siehe Kap. 3.4.1) sind die folgenden funktionsspezifischen Parameter zu belegen:

Tabelle 3-10 Applet spezifische Parameter

Parameter Name	Typ	Beschreibung
SLOT	byte	Slot Nummer (Steckplatz) der angesprochenen Baugruppe (1..18) (bei S7–200 immer R/S=0/0)
RACK	byte	Rack Nummer der angesprochenen Baugruppe (0..7) (bei S7–200 immer R/S=0/0)
CYCLETIME	int	Zykluszeit für den Leseauftrag; Angabe in Millisekunden. Wertebereich: >5000 (Empfehlung)
FORMAT	string	Die Zeichenfolge im Parameter Format legt fest, wie die gelesenen Variablenwerte darzustellen sind (Beispiel und Erläuterung siehe unten).

Tabelle 3-11 optionale Applet spezifische Parameter – Verwendung von JavaBeans

Parameter Name	Typ	Beschreibung
DISPLAY	string	Kennzeichnet ein JavaBean, das zur graphischen Datendarstellung verwendet werden kann. derzeit mögliche Werte sind (siehe auch Tabelle 4-2, Seite 77): • CLTacho • CLLevel • CLThermo
MINVAL	int, float	MINVAL und MAXVAL sind Bereichsangaben zur Formatierung der graphisch dargestellten Variablen.
MAXVAL	int, float	
DIMENSION	string	Hier kann eine physikalische Einheit angegeben werden.
EDIT	bool	Die Online–Parametrierung kann ein– oder ausgeschaltet werden. Parametriermöglichkeit ein = true aus = false Wird der Parameter im Applet–Aufruf nicht verwendet, ist die Online–Parametrierung **standardmäßig ausgeschaltet!**

Tabelle 3-11 optionale Applet spezifische Parameter, (Suite) – Verwendung von JavaBeans, Fortsetzung

Parameter Name	Typ	Beschreibung
LANGUAGE	string	Die für die Beschriftungen, Meldungen und Diagnoseanzeigen verwendete Sprache kann fest eingestellt werden. Unterstützt werden zur Zeit die Sprachen Deutsch und Englisch. Als Parameter werden die zweistelligen ISO–639 Codes benutzt. Parametriermöglichkeit: Deutsch = “de” Englisch = “en” Wird der Parameter im Applet–Aufruf nicht verwendet, wird die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt (kann bei Windows–Betriebsystemen unter Systemsteuerung – Ländereinstellungen geändert werden). Wird eine nicht unterstützte Sprache angegeben, so wird versucht die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt.
DEBUGLEVEL	int	Stellt den Detail–Level der Debug–Ausgabe in der Java–Konsole ein. Parametriermöglichkeit: 0 = Keine Ausgabe 1 = Alle Meldungen ausgeben 2 = Nur Warn– und Fehlermeldungen ausgeben 3 = Nur Fehlermeldungen ausgeben 4 = Nur Meldungen fataler Fehler ausgeben Wird der Parameter im Applet–Aufruf nicht verwendet, so wird der Level 3 verwendet, d.h. es werden nur Fehlermeldungen ausgegeben.

Tabelle 3-12 Parameter für die symbolische Variablen–Adressierung (**alternativ** zur absoluten Adressierung – **nur bei S7–300/400**)

Parameter Name	Typ	Beschreibung
SYMBOL	string	Symbolischer Name der S7–Variable Die Variable muss mit dem Symboleditor von STEP 7 angelegt und für den Zugriff über den IT–CP projektiert sein (siehe Kap. 3.5).

Tabelle 3-13 Parameter für die indirekte Variablen–Adressierung über ANY–Zeiger (**alternativ** zu SYMBOL)

Parameter Name	Typ	Beschreibung
VARTYPE (Datentyp)	int	Diese Variablentypcodierung im ANY–Zeiger kennzeichnet den Datentyp der zu lesenden Variablen; mögliche Angaben sind: 0x02 BYTE Bytes (8 Bits) 0x01 BOOL Datentyp BOOL (1 Bit) 0x03 CHAR Zeichen (8 Bits) 0x04 WORD Wörter (16 Bits) 0x05 INT Ganzzahlen (16 Bits) 0x06 DWORD Wörter (32 Bits) 0x07 DINT Ganzzahlen (32 Bits) 0x08 REAL Gleitpunktzahlen (32 Bits) 0x09 DATE Datum 0x0A TIME_OF_DAY (TOD) Uhrzeit 0x0B TIME Zeit 0x13 STRING Zeichenkette Die Angabe kann dezimal (z.B. 10) oder hexadezimal (z.B. 0x0A) erfolgen. Hinweis: Die folgenden Datentypen werden im Parametrierdialog zur Auswahl angeboten. Die Übermittlung dieser komplexen Datentypen wird aber nur durch die S7–Beans (siehe Kap. 4) unterstützt. Anhand der S5– bzw. S7–Formatbeschreibung (siehe STEP 7 Online–Hilfe) können diese Formate dann programmtechnisch decodiert und weiterverarbeitet werden. 0x0C S5TIME Datentyp S5TIME 0x0E DATE_AND_TIME (DT) Datum und Zeit (64 Bits) 0x1C COUNTER Zähler 0x1D TIMER Zeitgeber **Anmerkung:** Die Angaben hier gelten für S7–300/400; bzgl. S7–200 siehe /4/
VARCNT (Wiederholfaktor)	int	Anzahl der zu lesenden Variablen; Durch diese Angabe können Sie angeben, ob eine Variable oder ein zusammenhängender Variablenbereich übergeben werden soll. STEP 7 kennzeichnet Felder und Strukturen als Anzahl (hier mit Hilfe des Wiederholfaktors) an Datentypen. Beispiel: Sollen 10 Wörter übergeben werden, muss beim Wiederholfaktor der Wert 10 und beim Datentyp der Wert 04 eingetragen werden.
VARAREA (Speicherbereich)	int	Bereichscodierung zur Kennzeichnung des Speicherbereiches; 0x81 E Speicherbereich der Eingänge 0x82 A Speicherbereich der Ausgänge 0x83 M Speicherbereich der Merker 0x84 DB Datenbaustein Die Angabe kann dezimal (z.B. 131) oder hexadezimal (z.B. 0x83) erfolgen. **Anmerkung:** Die Angaben hier gelten für S7–300/400; bzgl. S7–200 siehe /4/
VARSUBAREA (Teilbereich)	int	Teilbereichscodierung; z.B. zur Angabe der DB–Nummer. (bei S7–200 immer Wert=1)

Tabelle 3-13 Parameter für die indirekte Variablen–Adressierung über ANY–Zeiger, (Suite)(**alternativ** zu SYMBOL), Fortsetzung

Parameter Name	Typ	Beschreibung
VAROFFSET (Byteadresse)	int	Angabe eines Byte–Offsets. Über diese Angabe kann die Variable oder der Variablenbereich innerhalb des angegebenen Speicherbereiches (VARAREA) adressiert werden (beispielsweise die Merker–Nummer).
VARBITOFFSET (Bitadresse)	int	Angabe eines Bit–Offsets. Über diese Angabe kann das Bit einer Variablen vom Typ BOOL adressiert werden.

Hinweise zu den Adressangaben VARTYPE...VAROFFSET

Zum Lesen der Daten wird vom IT–CP die S7–Funktion SFB 14 (GET) genutzt. Entsprechend ist für die Variablenadressierung der Datentyp ANY–Zeiger für die Parameterübergabe an den SFB zu versorgen.

Weitere Erläuterungen zu SFB 14 (GET) finden Sie in der Online–Hilfe zu STEP 7, dort im Anhang der Hilfethemen unter “Format des Parametertyps ANY”; eine ausführliche Darstellung des ANY–Zeigers ist auch in /21/ zu finden.

Wertebereiche und Hantierung des Parameters FORMAT

Im Parameter FORMAT können folgende Kennungen verwendet werden:

Tabelle 3-14 Bedeutung des Parameters Format

Kennung	Anzahl der berücksichtigten Bytes	Darstellung
\	0	Schrägstrich umgekehrt; markiert das nachfolgende Zeichen als Kennung gemäß vorliegender Tabelle. Für die Darstellung von ”\” ist folgende Eingabe erforderlich: ”\\” Beispiel für Variable vom Typ Integer: \I
S	1	Bit–String Interpretiert das zugeordnete Byte als Folge einzeln darzustellender Bits. Beispiel für eine Ausgabe: 01101110
O	1	Oktal
H	1	Hexadezimal
B	1	Unsigned Byte
C	1	Signed Byte
D	4	Unsigned 32
L	4	Signed 32

Tabelle 3-14 Bedeutung des Parameters Format, Fortsetzung

Kennung	Anzahl der berücksichtigten Bytes	Darstellung
W	2	Unsigned 16
I	2	Signed 16
F	4	Floating Point
Z	1	Character
X (n, String1, String2)	1	Binärwert (n= Position 0..7, String1= Zeichenfolge für Wert 1; String 2 = Zeichenfolge für Wert 0) Zur Anwendung bei Ausgabe mehrerer Binärwerte innerhalb eines Byte ist alternativ die Kennung Y zu verwenden! **Beachte!** Zwischen Klammer–auf und dem ersten Komma darf **kein** Leerzeichen stehen; die Eingabe wird sonst nicht erkannt.
Y (n, on, off)	0	Binärwert (n= Position 0..7, String1= Zeichenfolge für Wert 1; String 2 = Zeichenfolge für Wert 0) zur Funktion: der Positionszähler wird, im Gegensatz zur Kennung X, nicht inkrementiert. Y wird verwendet, wenn mehrere Binärwerte innerhalb eines Bytes ausgegeben werden sollen. Erst für die Ausgabe des letzten Binärwertes innerhalb eines Bytes wird die Kennung X verwendet. **Beachte!** Zwischen Klammer–auf und dem ersten Komma darf **kein** Leerzeichen stehen; die Eingabe wird sonst nicht erkannt.
G	1	Inkrementiert den Positionszähler um 1 Byte, ohne Darstellung; Diese Kennung wird benötigt, um Bytes im Variablenstring zu überspringen. Das ist dann notwendig, wenn aufgrund der Datenstruktur – es werden beispielsweise abwechselnd Worte und Byte definiert – Leer–Bytes zu berücksichtigen sind.

Interpretation des Formatstrings

Die Zeichenfolge im Parameter Format legt fest, wie die gelesenen Variablenwerte darzustellen sind. Dabei wird davon ausgegangen, dass Variablenwerte in Form von Byte–Strings gelesen werden.

Bei der Darstellung wird links beginnend die Zeichenfolge im Formatstring interpretiert und dem Variablenstring zugewiesen. Mit jedem zugewiesenen und ausgegebenen Wert wird ein Positionszähler entsprechend der Angabe “Anzahl der berücksichtigten Bytes” in Tabelle 3-14 inkrementiert.

Die Ausgabe erfolgt so lange, bis alle Formatkennungen abgearbeitet sind. Sind nicht alle Bytes zuordenbar, erfolgt keine weitere Ausgabe. Sind jedoch mehr Formatkennungen angegeben, als zugeordnet werden können, erfolgt die Ausgabe so lange, bis die Formatkennungen abgearbeitet sind.

Hinweis

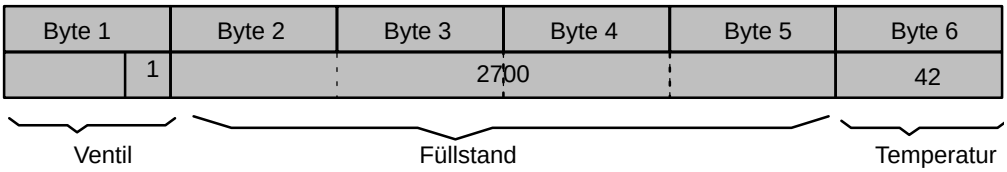
Achten Sie auf exakte Abstimmung des Formatstrings mit der Byte–Ablage im Variablenstring.

Die folgende Darstellung verdeutlicht, wie hier am Beispiel von 3 Variablen diese Zuweisung und Ausgabe erfolgt.

Die Variablen werden hierbei als zusammenhängende Kette von 6 Bytes aus der S7–CPU mit einer Anweisung gelesen. Anschließend werden sie durch die Formatzuweisung den unterschiedlichen Variablentypen zugewiesen und dargestellt.

Formatstring:Ventil: \X(0, auf, zu) Füllstand= \D mm Temperatur= \B °C

gelesener Variablenstring (6 Byte):



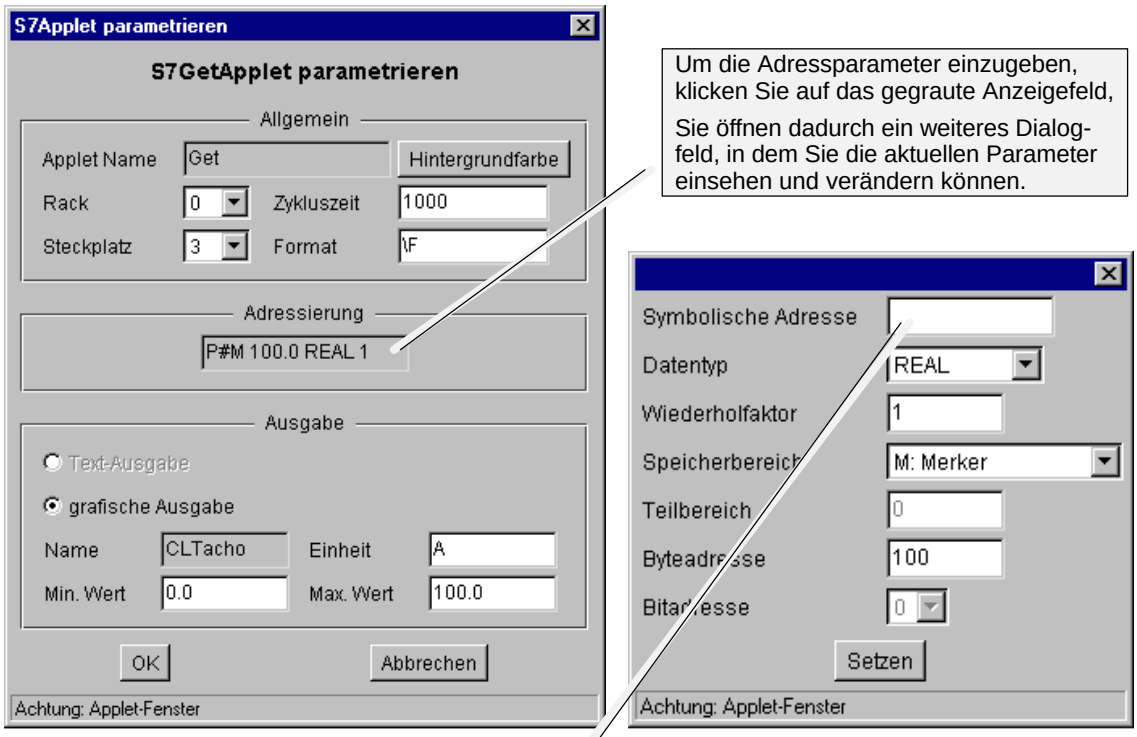
Ausgabe in der HTML–Seite

Ventil: auf Füllstand=2700 mm Temperatur=42 °C

Eine **Fehlermeldung** (siehe Ausgabe in der Java Konsole in Kap. 3.6) wird nur dann erzeugt, wenn ein Zuweisungskonflikt vorliegt. Beispiel: Im Formatstring wird mit der Angabe \D ein Doppelwort definiert, die gelesene Variable umfasst aber nur 2 Byte.

Parametrierhilfen (Bedeutung und Anwendung siehe Kap. 3.4.2)

Die **Online–Parametrierung** für Testzwecke wird unterstützt. Doppelklicken Sie hierzu auf das Ausgabefeld um den Parametrierdialog zu öffnen.



Um die Adressparameter einzugeben, klicken Sie auf das gegraute Anzeigefeld,

Sie öffnen dadurch ein weiteres Dialog-feld, in dem Sie die aktuellen Parameter einsehen und verändern können.

Falls Sie die Variable(n) über ANY -Zeiger indirekt adressieren möchten, lassen Sie das Feld für die Symbolische Adresse leer.

Wenn Sie eine symbolische Adresse eingeben, sind die übrigen Eingabefelder nicht mehr bedienbar. Die zuvor eingestellten Parameter bleiben jedoch erhalten und können durch Löschen der symbolischen Adresse ohne Neueingabe wieder aktiviert werden.

3.4.8 S7GetApplet – Beispiele

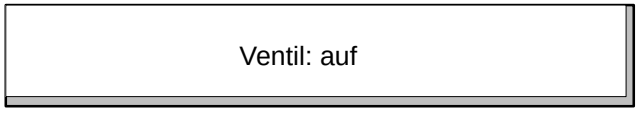
An einfachen Beispielen sollen nachfolgend die beiden Zugriffsmöglichkeiten über die Applet–Parametrierung aufgezeigt werden.

Beispiel 1: Variable in einem Datenbaustein ansprechen

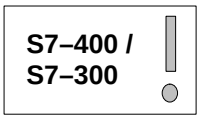
Angenommen wird eine Binärvariable, die den Zustand eines Ventils – auf/zu – beinhaltet. Diese Variable wird mit dem Namen “valve” im DB10 hinterlegt. Der DB10 erhält in der Symboltabelle der CPU den Namen “heater1”.

Die Ausgabe der Variable erfolgt als Zeichenstring.

Ausgabe in der HTML–Seite:



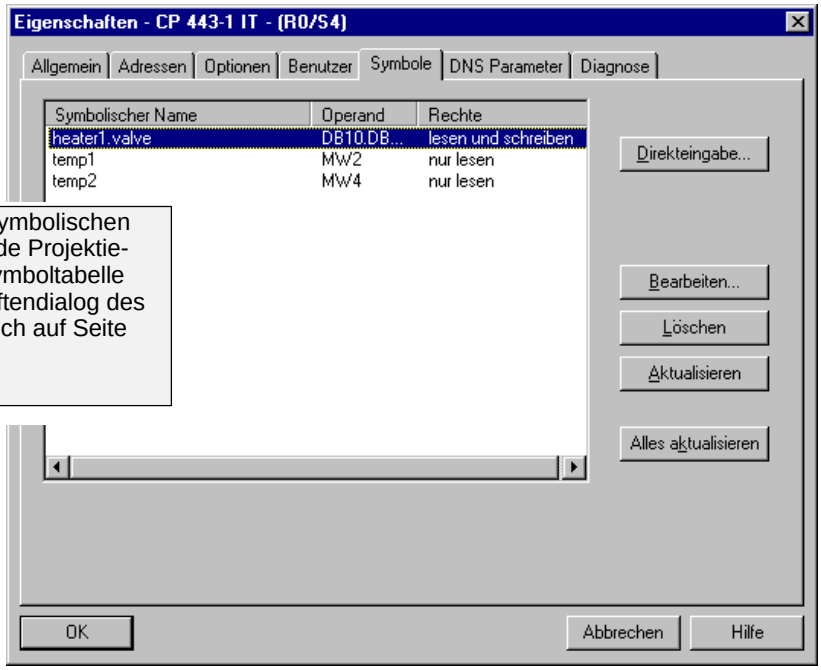
Für den Zugriff auf diese Variable und die Ausgabe in der HTML–Seite wird demnach folgende Applet–Parametrierung benötigt:



a) Zugriff mit symbolischer Adressierung der Variable

```
<P ALIGN=Center><APPLET CODE=”de.siemens.simaticnet.itcp.
   applets.S7GetApplet.class”

CODEBASE=”/applets/” ARCHIVE=”s7applets.jar, s7api.jar”  NAME=”ventil4”
WIDTH=45 HEIGHT=30>

 <PARAM name=”RACK” value=0>  <PARAM name=”SLOT” value=3>
 <PARAM name=”CYCLETIME” value=5000>
<PARAM name=”SYMBOL” value=”heater1.valve”>
<PARAM name=”FORMAT”  value=”Ventil: \X(0,auf,zu)”>
<PARAM name=”BACKGROUNDCOLOR” value=”0xFFFFFF”>
<PARAM name=”EDIT” value=”true”>
</APPLET>
```

Beachten Sie beim Parameter “Format”, dass auf eine Variable vom Typ Byte zugegriffen wird, die an Position “0” den Binärwert enthält.



Voraussetzung für diesen symbolischen Zugriff ist eine entsprechende Projektierung der Variablen in der Symboltabelle der CPU und im Eigenschaftendialog des IT–CP (Lesen Sie hierzu auch auf Seite 58):

b) Zugriff mit indirekter Adressierung der Variable

```
<P ALIGN=Center><APPLET CODE="de.siemens.simaticnet.itcp.
   applets.S7GetApplet.class"

CODEBASE="/applets/" ARCHIVE="s7applets.jar, s7api.jar"  NAME="ventil4"
WIDTH=45 HEIGHT=30>

 <PARAM name="RACK" value=0> <PARAM name="SLOT" value=3>
 <PARAM name="CYCLETIME" value=5000>
<PARAM name="VARTYPE" value=2>
<PARAM name="VARCNT" value =1>
<PARAM name="VARAREA" value=0x84>
<PARAM name="VARSUBAREA" value=10>
<PARAM name="VAROFFSET" value=40>
<PARAM name="FORMAT" value="Ventil: \X(0,auf,zu)">
<PARAM name="BACKGROUNDCOLOR" value="0xFFFFFF">
<PARAM name="EDIT" value="true">
</APPLET>
```

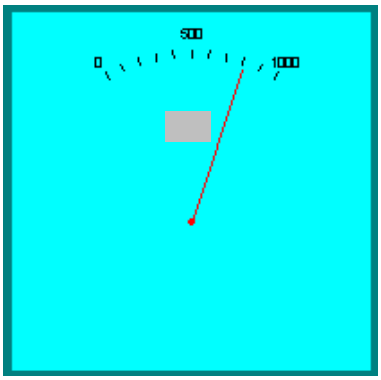
Diese Angaben entsprechen dem ANY–Zeiger:

P#DB10.DBX 40.0 BYTE 1

Beispiel 2: Variable in einem Merkerbereich ansprechen

Angenommen wird ein zu lesendes 16–Bit Merkerwort (MW 12). Der Zugriff erfolgt über indirekte Adressierung der Variablen.

Die Ausgabe der Variablen erfolgt in graphischer Form:



Für den Zugriff auf diese Variable und die Ausgabe in der HTML–Seite wird demnach folgende Applet–Parametrierung benötigt:

```
<P ALIGN=Center><APPLET CODE="de.siemens.simaticnet.itcp.applets.
   S7GetApplet.class"
CODEBASE="/applets/" ARCHIVE="s7applets.jar, s7api.jar"  NAME="speed"
WIDTH=45 HEIGHT=30>
 <PARAM name="RACK" value=0> <PARAM name="SLOT" value=3>
 <PARAM name="CYCLETIME" value=5000>
<PARAM name="VARTYPE" value=5>
<PARAM name="VARCNT" value =1>
<PARAM name="VARAREA" value=0x83>
<PARAM name="VARSUBAREA" value=0>
<PARAM name="VAROFFSET" value=12>

<PARAM name="DISPLAY"  value="CLTACHO">
<PARAM name="MINVAL" value=0><PARAM name="MAXVAL" value=1000>
<PARAM name="BACKGROUNDCOLOR" value="0x00FFFF">
<PARAM name="EDIT" value="true">
</APPLET>
```

Diese Angaben entsprechen dem ANY–Zeiger:

P#MW12 INT 1

Für die graphische Ausgabe wird das S7–JavaBean CLTACHO verwendet.

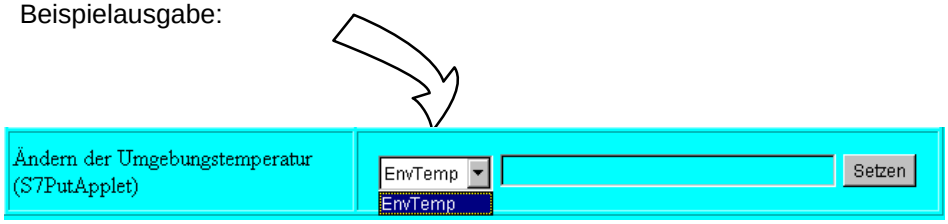
3.4.9 S7PutApplet – Beschreibung

Bedeutung

Das Applet nimmt Variablenwerte per Benutzereingabe entgegen und überträgt diese in die S7–CPU.

Entsprechend der Parametrierung in einem Formatstring werden hierzu im Web Browser 2 Felder zur Auswahl des Datenbereiches und zur Eingabe des Variablenwertes aufgeblendet. Eine zusätzliche Schaltfläche “Setzen” dient dazu, den Schreibvorgang auszulösen.

Beispielausgabe:



Voraussetzungen

Die Benennung der Variablen kann symbolisch oder durch Adressangabe erfolgen.

S7–400 / S7–300

Der symbolische Zugriff setzt eine entsprechende Symbol–Projektierung im IT–CP voraus. Bei symbolischem Zugriff werden die Zugriffsrechte entsprechend der Variablenprojektierung geprüft (siehe Kap. 3.5).

Zugriffsrechte

Unter dem beim Zugriff verwendeten Benutzernamen muss folgendes Zugriffsrecht eingeräumt sein (siehe Dialog “Bearbeiten Benutzereintrag in Kap. 1.4):

- “auf die projektierten Symbole zuzugreifen“ (nur bei symbolischem Zugriff);
- ”Variablen über absolute Adressen zu schreiben” (nur bei absolutem Zugriff);

Aufruf–Tags

CODE=”de.siemens.simaticnet.itcp.applets.S7PutApplet.class”
CODEBASE=”/applets/”
ARCHIVE=”s7applets.jar, s7api.jar”

Parametrierung

Zusätzlich zu den allgemeinen Parametern (siehe Kap. 3.4.1) sind die folgenden funktionsspezifischen Parameter zu belegen:

Tabelle 3-15 Applet spezifische Parameter

Parameter Name	Typ	Beschreibung

Tabelle 3-15 Applet spezifische Parameter, Fortsetzung

Parameter Name	Typ	Beschreibung
SLOT	byte	Slot Nummer (Steckplatz) der angesprochenen Baugruppe (1..18) (bei S7–200 immer R/S=0/0)
RACK	byte	Rack Nummer der angesprochenen Baugruppe (0..7) (bei S7–200 immer R/S=0/0)

Tabelle 3-16 optionale Applet spezifische Parameter

EDIT	bool	Die Online–Parametrierung kann ein– oder ausgeschaltet werden. Parametriermöglichkeit ein = true aus = false Wird der Parameter im Applet–Aufruf nicht verwendet, ist die Online–Parametrierung **standardmäßig ausgeschaltet!**
LANGUAGE	string	Die für die Beschriftungen, Meldungen und Diagnoseanzeigen verwendete Sprache kann fest eingestellt werden. Unterstützt werden zur Zeit die Sprachen Deutsch und Englisch. Als Parameter werden die zweistelligen ISO–639 Codes benutzt. Parametriermöglichkeit: Deutsch = “de” Englisch = “en” Wird der Parameter im Applet–Aufruf nicht verwendet, wird die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt (kann bei Windows–Betriebsystemen unter Systemsteuerung – Ländereinstellungen geändert werden). Wird eine nicht unterstützte Sprache angegeben, so wird versucht die vom Hostsystem verwendete Sprache (oder Englisch, wenn diese nicht verfügbar ist) benutzt.
DEBUGLEVEL	int	Stellt den Detail–Level der Debug–Ausgabe in der Java–Konsole ein. Parametriermöglichkeit: 0 = Keine Ausgabe 1 = Alle Meldungen ausgeben 2 = Nur Warn– und Fehlermeldungen ausgeben 3 = Nur Fehlermeldungen ausgeben 4 = Nur Meldungen fataler Fehler ausgeben Wird der Parameter im Applet–Aufruf nicht verwendet, so wird der Level 3 verwendet, d.h. es werden nur Fehlermeldungen ausgegeben.

Tabelle 3-17 Parameter für die symbolische Daten–Adressierung **(nur bei S7–300/400)**

Parameter Name	Typ	Beschreibung
SYMBOLNUM	int	Anzahl der symbolisch eingebbaren Variablen
SYMBOLn *)	string	Symbolischer Name der S7–Variable. Der Name erscheint in der Auswahlbox des ersten Textfeldes. Die Variable muss mit dem Symboleditor von STEP 7 angelegt und für den Zugriff über IT–CP projektiert sein (siehe Kap. 3.5).
SYMFORMATn *)	string	Die Zeichenfolge im Parameter Format legt fest, wie die eingegebenen Variablenwerte zu interpretieren sind . Entsprechend der Angabe n (Beispiel siehe unten) erfolgt die Zuordnung der Angabe zur S7–Variablen im Eingabefeld.

*) Legende: “n” kennzeichnet eine mit “1” beginnende fortlaufende Numerierung der symbolisch adressierten Variablen innerhalb eines Aufrufes;

Tabelle 3-18 Parameter für die absolute Daten–Adressierung

Parameter Name	Typ	Beschreibung
VARNUM	int	Anzahl der zu schreibenden Variablen;
VARNAMEn *)	string	Variablenname für eine indirekt adressierte Variable; Der Name erscheint bei der Ausgabe in der Auswahlbox des ersten Textfeldes. Max. 256 Zeichen
VARTYPEn *) (Datentyp)	int	Variablentypcodierung im ANY–Zeiger; Diese Variablentypcodierung im ANY–Zeiger kennzeichnet den Datentyp der zu schreibenden Variablen; mögliche Angaben sind: 0x02 BYTE Bytes (8 Bits) 0x01 BOOL Datentyp BOOL (1 Bit) 0x03 CHAR Zeichen (8 Bits) 0x04 WORD Wörter (16 Bits) 0x05 INT Ganzzahlen (16 Bits) 0x06 DWORD Wörter (32 Bits) 0x07 DINT Ganzzahlen (32 Bits) 0x08 REAL Gleitpunktzahlen (32 Bits) 0x09 DATE Datum 0x0A TIME_OF_DAY (TOD) Uhrzeit 0x0B TIME Zeit 0x13 STRING Zeichenkette Hinweis: Die folgenden Datentypen werden im Parametrierdialog zur Auswahl angeboten. Die Übermittlung dieser komplexen Datentypen wird aber nur durch die S7–Beans (siehe Kap. 4) unterstützt. Anhand der S5– bzw. S7–Formatbeschreibung (siehe STEP 7 Online–Hilfe) können diese Formate dann programmtechnisch decodiert und weiterverarbeitet werden. 0x0C S5TIME Datentyp S5TIME 0x0E DATE_AND_TIME (DT) Datum und Zeit (64 Bits) 0x1C COUNTER Zähler 0x1D TIMER Zeitgeber **Anmerkung:** Die Angaben hier gelten für S7–300/400; bzgl. S7–200 siehe /4/

Tabelle 3-18 Parameter für die absolute Daten–Adressierung, Fortsetzung

Parameter Name	Typ	Beschreibung
VARAREAn *) (Speicherbereich)	int	Bereichscodierung zur Kennzeichnung des Speicherbereiches; 0x81 E Speicherbereich der Eingänge 0x82 A Speicherbereich der Ausgänge 0x83 M Speicherbereich der Merker 0x84 DB Datenbaustein Die Angabe kann dezimal (z.B. 131) oder hexadezimal (z.B. 0x83) erfolgen. **Anmerkung:** Die Angaben hier gelten für S7–300/400; bzgl. S7–200 siehe /4/
VARSUBAREAn *) (Teilbereich)	int	Teilbereichscodierung; z.B. zur Angabe der DB–Nummer.
VAROFFSETn *) (Byteadresse)	int	Angabe eines Byte–Offsets. Über diese Angabe kann die Variable oder der Variablenbereich innerhalb des angegebenen Speicherbereiches (VARAREA) adressiert werden (beispielsweise die Merker–Nummer).
VARBITOFFSET (Bitadresse)	int	Angabe eines Bit–Offsets. Über diese Angabe kann das Bit einer Variablen vom Typ BOOL adressiert werden.
VARFORMATn *)	string	Die Zeichenfolge im Parameter Format legt fest, wie die eingegebenen Variablenwerte zu interpretieren sind . Entsprechend der Angabe n (Beispiel siehe Kap. 3.4.10) erfolgt die Zuordnung der Angabe zur S7–Variablen im Eingabefeld.

*) Legende: “n” kennzeichnet eine mit “1” beginnende fortlaufende Numerierung der indirekt (über ANY–Zeiger) adressierten Variablen innerhalb eines Aufrufes;

Wertebereich für den Parameter FORMAT

Im Parameter FORMAT können folgende Kennungen verwendet werden:

Tabelle 3-19 Bedeutung des Parameters Format

Kennung	Anzahl der belegten Bytes	Eingabestring wird wie folgt interpretiert
S	1	Bit–String
O	1	Oktal
H	1	Hexadezimal
B	1	Unsigned Byte
C	1	Signed Byte
D	4	Unsigned 32
L	4	Signed 32
W	2	Unsigned 16
I	2	Signed 16

Tabelle 3-19 Bedeutung des Parameters Format, Fortsetzung

Kennung	Anzahl der belegten Bytes	Eingabestring wird wie folgt interpretiert
Z	n	Character string
F	4	Floating Point 32

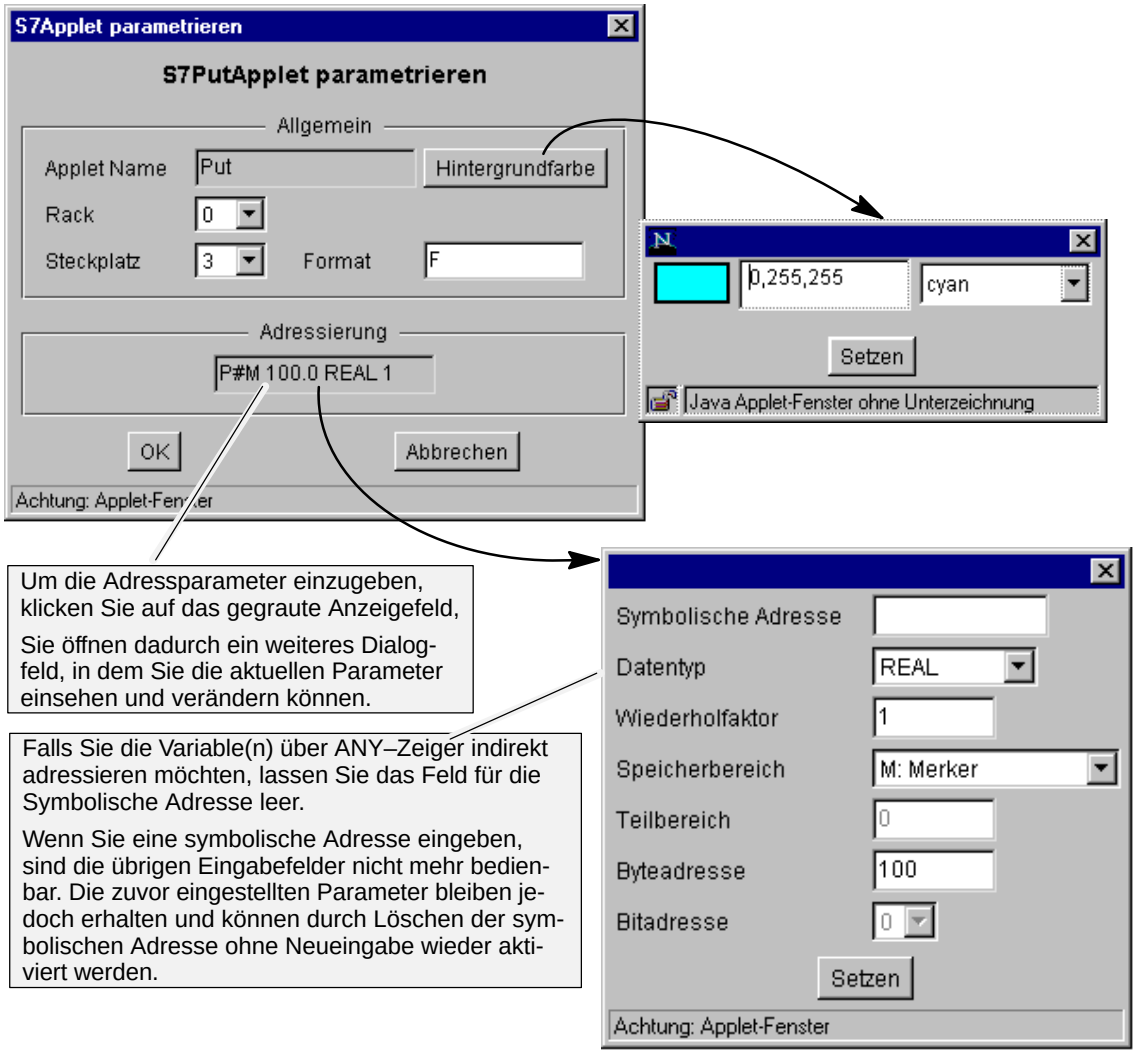
Hinweis

Im Gegensatz zum S7GetApplet wird die Kennung “\” beim S7PutApplet nicht benötigt.

Parametrierhilfen (Bedeutung und Anwendung siehe Kap. 3.4.2)

- Die **Online–Parametrierung** für Testzwecke wird unterstützt. Voraussetzung ist, dass der Applet–Parameter EDIT=true gesetzt wurde.

  Doppelklicken Sie im Darstellungsbereich (**Achtung! nicht** im Eingabefeld für den Variablenwert!) des PutApplet, um folgendes Dialogfeld zu öffnen:



Um die Adressparameter einzugeben, klicken Sie auf das gegraute Anzeigefeld,

Sie öffnen dadurch ein weiteres Dialogfeld, in dem Sie die aktuellen Parameter einsehen und verändern können.

Falls Sie die Variable(n) über ANY–Zeiger indirekt adressieren möchten, lassen Sie das Feld für die Symbolische Adresse leer.

Wenn Sie eine symbolische Adresse eingeben, sind die übrigen Eingabefelder nicht mehr bedienbar. Die zuvor eingestellten Parameter bleiben jedoch erhalten und können durch Löschen der symbolischen Adresse ohne Neueingabe wieder aktiviert werden.

3.4.10 S7PutApplet – Beispiele

Beispiel 1: Eine Variable eingeben

Für die Eingabe einer Variablen, die z.B. als Sollwertvorgabe verwendet wird, wird folgende Applet–Parametrierung benötigt:

S7–400 / S7–300

a) mit symbolischer Adressierung der Variable

```
<APPLET CODE=”de.siemens.simaticnet.itcp.applets.S7PutApplet.class”
CODEBASE=”/applets/”
ARCHIVE=”s7applets.jar, s7api.jar” NAME=”s7_client0” WIDTH=400
HEIGHT=40>
 <PARAM name=”RACK” value=0>
 <PARAM name=”SLOT” value=3>
<PARAM name=”SYMBOLNUM” value=”1”>
<PARAM name=”SYMBOL1” value=”SollwertKessel1”>
<PARAM name=”SYMFORMAT1” value=”I”>
<PARAM name=”BACKGROUNDCOLOR” value=”0x00FFFF”>
<PARAM name=”EDIT” value=”true”>
</APPLET>
```

b) S7–Applet mit indirektem Zugriff:

```
<APPLET CODE=””de.siemens.simaticnet.itcp.applets.S7PutApplet.class”
CODEBASE=”/applets/”
ARCHIVE=”s7applets.jar, s7api.jar” NAME=”s7_client0” WIDTH=400
HEIGHT=40>
 <PARAM name=”RACK” value=0>
 <PARAM name=”SLOT” value=3>
<PARAM name=”VARNUM” value=”1”>
<PARAM name=”VARNAME1” value=“SollwertKessel1”>
<PARAM name=”VARTYPE1” value=2>
<PARAM name=”VARAREA1” value=0x84>
<PARAM name=”VARSUBAREA1” value=0x10>
<PARAM name=”VAROFFSET1” value=40>
<PARAM name=”VARFORMAT1” value=”B”>
<PARAM name=”BACKGROUNDCOLOR” value=”0x00FFFF”>
<PARAM name=”EDIT” value=”true”>
</APPLET>
```

Diese Angaben entsprechen dem ANY–Zeiger:

P#DB10.DBX 40.0 BYTE 1

Beispiel 2: Mehrere Variablen eingeben

Mit einem S7PutApplet können auch mehrere Variablen eingegeben werden. Zusätzlich ist es möglich, die Adressierungsarten – symbolisch oder indirekt – zu mischen.

Im folgenden Beispiel werden 3 Variablen über symbolische Adressierung und eine Variable über indirekte Adressierung angesprochen.

Für die Eingabe mehrerer Variablen bei dieser gemischten Adressierung kann die Applet–Parametrierung dann wie folgt aussehen:

```
<APPLET CODE="de.siemens.simaticnet.itcp.applets.S7PutApplet.class"
CODEBASE="/applets/"

ARCHIVE="s7applets.jar, s7api.jar" NAME="s7_client0" WIDTH=400
HEIGHT=40>

 <PARAM name="RACK" value=0>
 <PARAM name="SLOT" value=3>
```

3 Variablen über symbolische Adressen

```
<PARAM name="SYMBOLNUM" value="3">
<PARAM name="SYMBOL1" value="Sollwert_Kessel1">
<PARAM name="SYMFORMAT1" value="I">
<PARAM name="SYMBOL2" value="Sollwert_Kessel2">
<PARAM name="SYMFORMAT2" value="I">
<PARAM name="SYMBOL3" value="Grenzwert_hoch">
<PARAM name="SYMFORMAT3" value="I">
```

1 Variable über indirekte Adresse

```
<PARAM name="VARNUM" value="1">
<PARAM name="VARNAME1" value="Sollwert_Kessel3">
<PARAM name="VARTYPE1" value=2>
<PARAM name="VARAREA1" value=0x83>
<PARAM name="VARSUBAREA1" value=0x00>
<PARAM name="VAROFFSET1" value=40>
<PARAM name="VARFORMAT1" value="B">

<PARAM name="BACKGROUNDCOLOR" value="0x00FFFF">
<PARAM name="EDIT" value="true">
</APPLET>
```

3.5 Variablen für symbolischen Zugriff projektieren

S7–400 / S7–300

Symbolischer Variablenzugriff vermeidet Projektierfehler

Die S7–Applets S7GetApplet und S7PutApplet ermöglichen den komfortablen symbolischen Variablenzugriff, so wie Sie ihn von der KOP/FUP/AWL–Programmierung mittels Symboltabelle her kennen. Sie werden dadurch von einer oft mühsamen und fehleranfälligen Adresshantierung entlastet.

Sie sehen in folgendem Beispiel, wie für einen Datenbaustein DB 100 eine Namenszuweisung in der Symboltabelle getroffen wird.

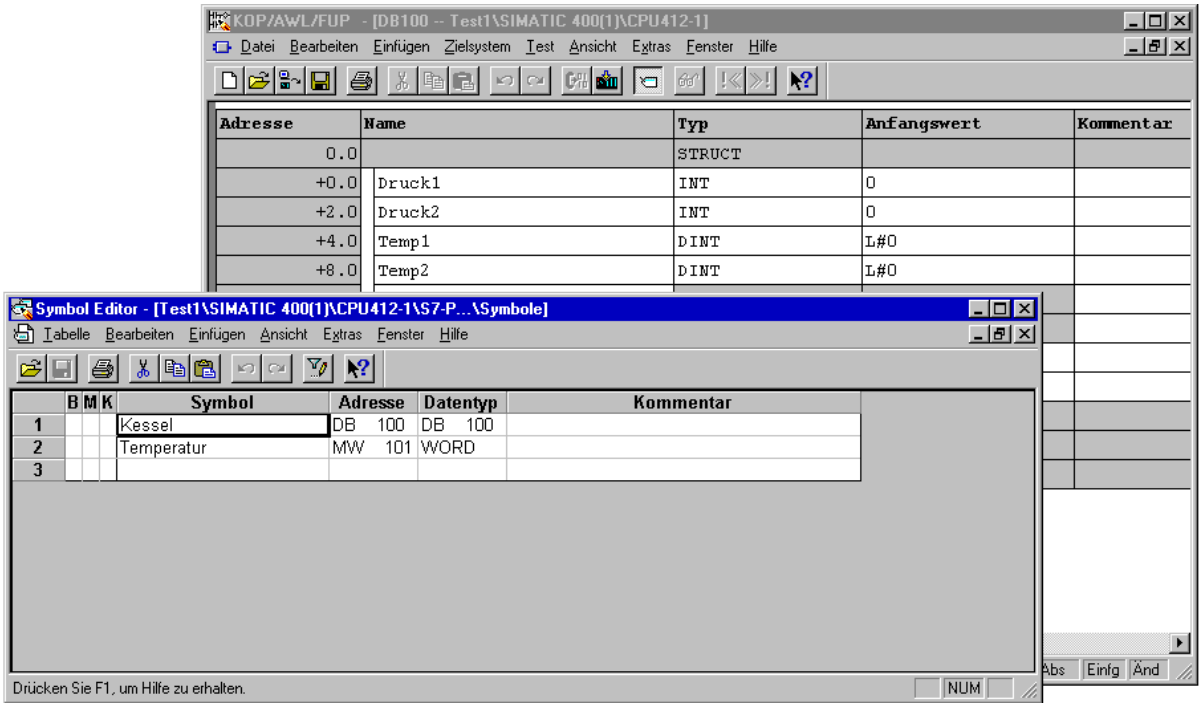


Bild 3-5

Symbole für HTML–Seiten

Um über einen Web Browser per Java–Applet auf Variablen in der S7–CPU zugreifen zu können, müssen dem IT–CP die Namen, Adressen und Zugriffsrechte dieser Variablen bekannt gemacht werden. Sie finden hierzu entsprechende Register im Eigenschaftendialog des IT–CP.

Die Symbole, die Sie hier in der Projektierung angeben, müssen zuvor mit dem STEP 7 Symboleditor in der Symboltabelle deklariert worden sein. Dadurch sind die Zuordnungen der Symbole zu den Variablen bereits getroffen. Mit der hier beschriebenen Projektierung wählen Sie die Symbole aus, auf die per Web Browser ein Zugriff möglich sein soll.

So gehen Sie vor

Wählen Sie unter STEP 7 in HW Konfig den Eigenschaftendialog Ihres IT–CP aus.

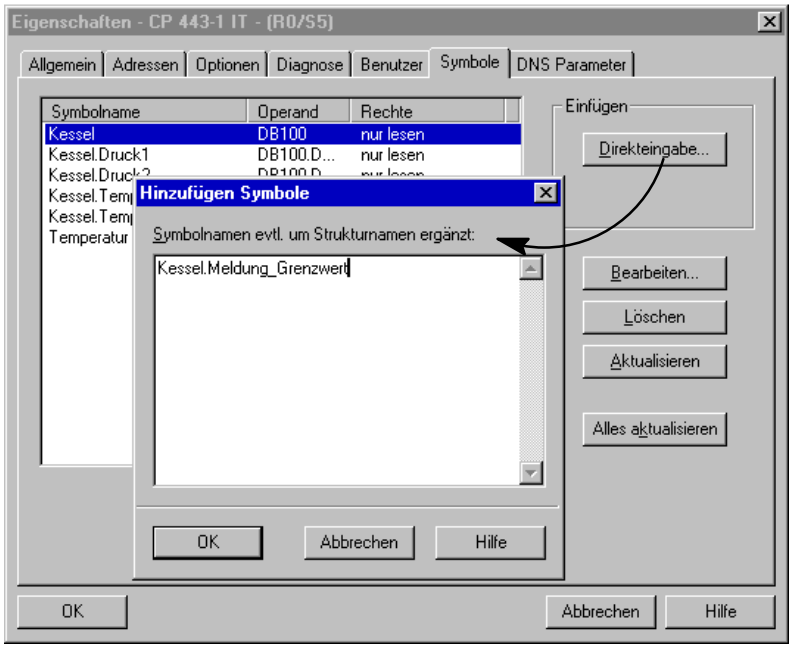


Bild 3-6

Tragen Sie die Symbole der Variablen oder der Strukturelemente ein, auf die ein Zugriff per Web Browser möglich sein soll. Detaillierte Hilfe zu den Schaltflächen und Dialogen gibt die Online–Hilfe.

Beispiele (siehe auch Syntaxregeln für Symbole in der Online–Hilfe)

- Einfache Variablen:
  Kessel
  Temperatur
- Strukturelemente
  Kessel.Druck1
  Kessel.Temperatur

Die Symbole müssen zuvor mit dem Symbol Editor von STEP 7 in der Symboltabelle hinterlegt worden sein! Die Eingabe wird nur bei Übereinstimmung mit dem Eintrag in der Symboltabelle übernommen.

Zugriffsrechte vergeben

Es ist möglich, den symbolisch deklarierten Variablen Zugriffsrechte zuzuweisen, die beim symbolischen Zugriff zusätzlich geprüft werden. Wählen Sie hierzu die Schaltfläche “Bearbeiten...”.

Achtung

Eine hier gesetzte Zugriffsbeschränkung kann nicht durch den Applet–Parameter EDIT außer Kraft gesetzt werden.

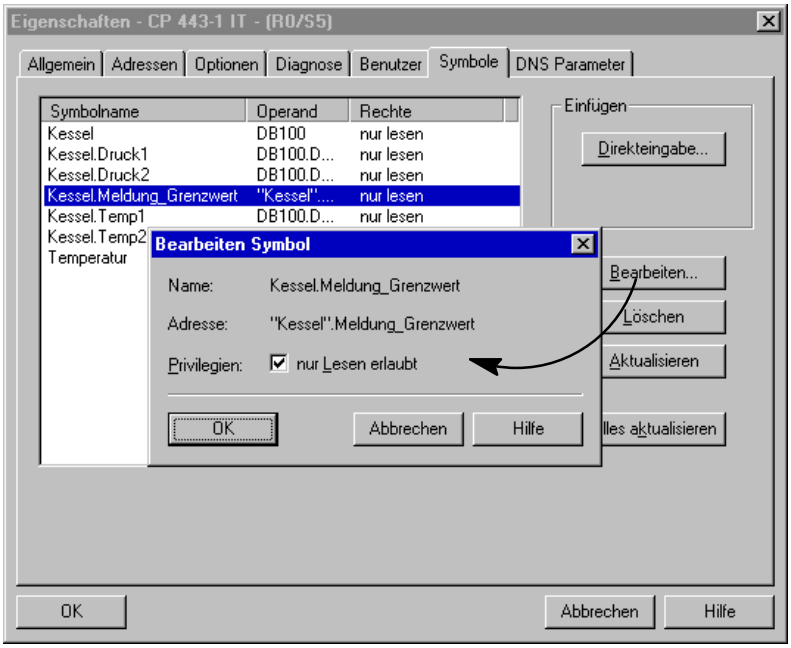


Bild 3-7

Hinweis

Wenn Sie die symbolischen Namen in den S7Beans bzw. S7Applets verwenden, müssen Sie dort die Rack/Steckplatz–Nummer in Klammern anhängen.

Beispiel: Variable ”Temperatur” muß im S7Bean/S7Applet mit ”Temperatur(0/3)” angegeben werden, wenn die Variable in der CPU auf Rack/Steckplatz 0/3 zu finden ist.

Drucken der Variablenliste:

Eine Variablenliste kann zusammen mit den Parametern der Baugruppe IT–CP in HW Konfig gedruckt werden.

3.6 HTML–Seiten testen und anwenden

Testen: Java Konsole

Der Web Browser bietet die Möglichkeit, die Ausführung von Java–Applets über eine Java Konsole zu verfolgen und zu protokollieren.

Die in Ihren HTML–Seiten verwendeten S7–Applets geben standardmäßig nur Fehlermeldungen in der Java Konsole aus. Diese Meldungen geben Ihnen wichtige Aufschlüsse bei unerwarteten Reaktionen in der Darstellung Ihrer HTML–Seite.

Um auch Warnungen oder gar alle verfügbaren Meldungen auf der Konsole auszugeben, kann mittels des Applet–Parameters DEBUGLEVEL die Detailtiefe eingestellt werden (siehe Kap. 4.3):

Parametriermöglichkeit:

0 = Keine Ausgabe

1 = Alle Meldungen ausgeben

2 = Nur Warn– und Fehlermeldungen ausgeben

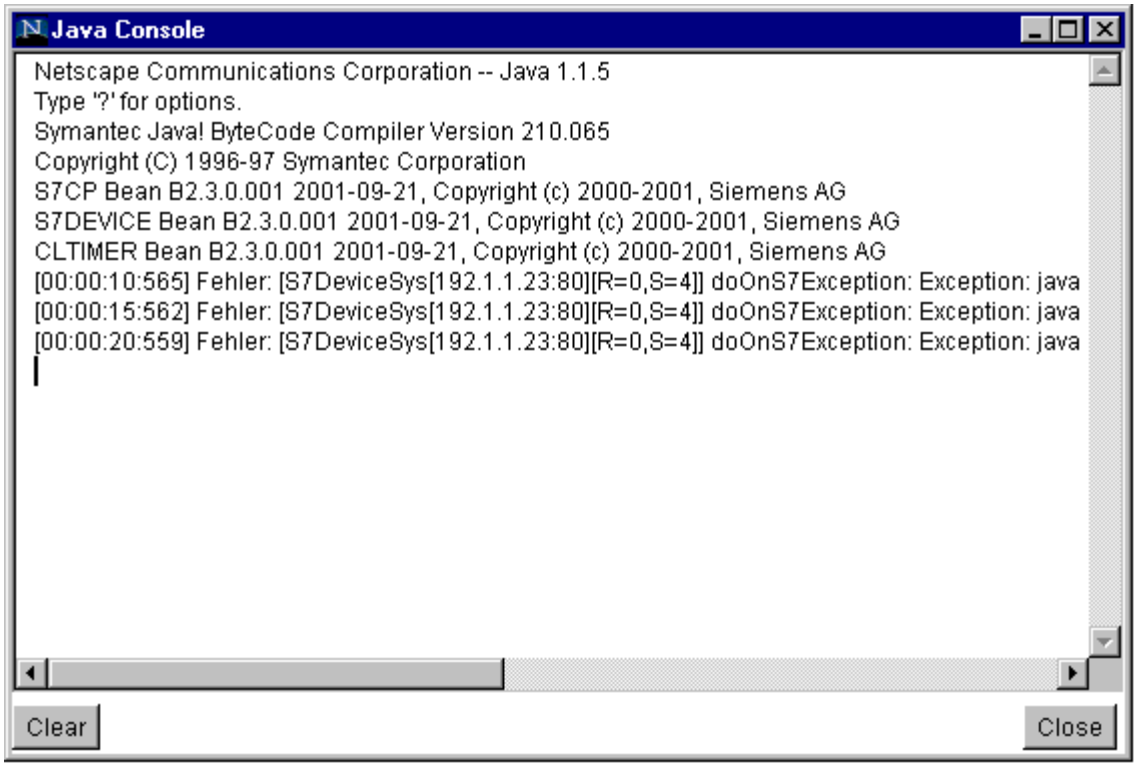
3 = Nur Fehlermeldungen ausgeben (Default–Level)

4 = Nur Meldungen fataler Fehler ausgeben

Wird der Parameter im Applet–Aufruf nicht verwendet, so wird der Level 3 verwendet, d.h. es werden nur Fehlermeldungen ausgegeben.

Achtung

Für den Einsatz sollten Sie Level 3 wählen! Die anderen Level sind nur für Testzwecke anzuwenden.



Laden: FTP für den Transfer von HTML Seiten

Mittels FTP–Funktionen (siehe Kap. 3) können Sie die HTML–Seiten auf den IT–CP laden und die Dateien auf Ihrem IT–CP nach Ihren Anforderungen organisieren.

3.7 JavaScript–Anbindung an die S7Applets

Sie können auch per JavaScript auf die Werte der S7Applets zugreifen. Dazu sind spezielle Methoden in die S7Applets integriert worden, die Sie hier aufgelistet finden:

Tabelle 3-20

S7Applet	Methode	Rückgabe– / Übergabe–Typ
S7GetApplet	getValue ( )	java.lang.Object
S7PutApplet	setValue (param)	java.lang.Object
S7StatusApplet	getState ( ) *)	java.lang.String
S7IdentApplet	getIdent ( ) *)	java.lang.String
– alle S7–Applets –	setRack (rack)	int
– alle S7–Applets –	setSlot (slot)	int

*) Methoden nur bei S7–300/400

In den folgenden drei Kapiteln finden Sie Web–Seiten im HTML–Code, in denen der Zugriff auf die S7Applets per JavaScript in einfachen Beispielen dargestellt wird.

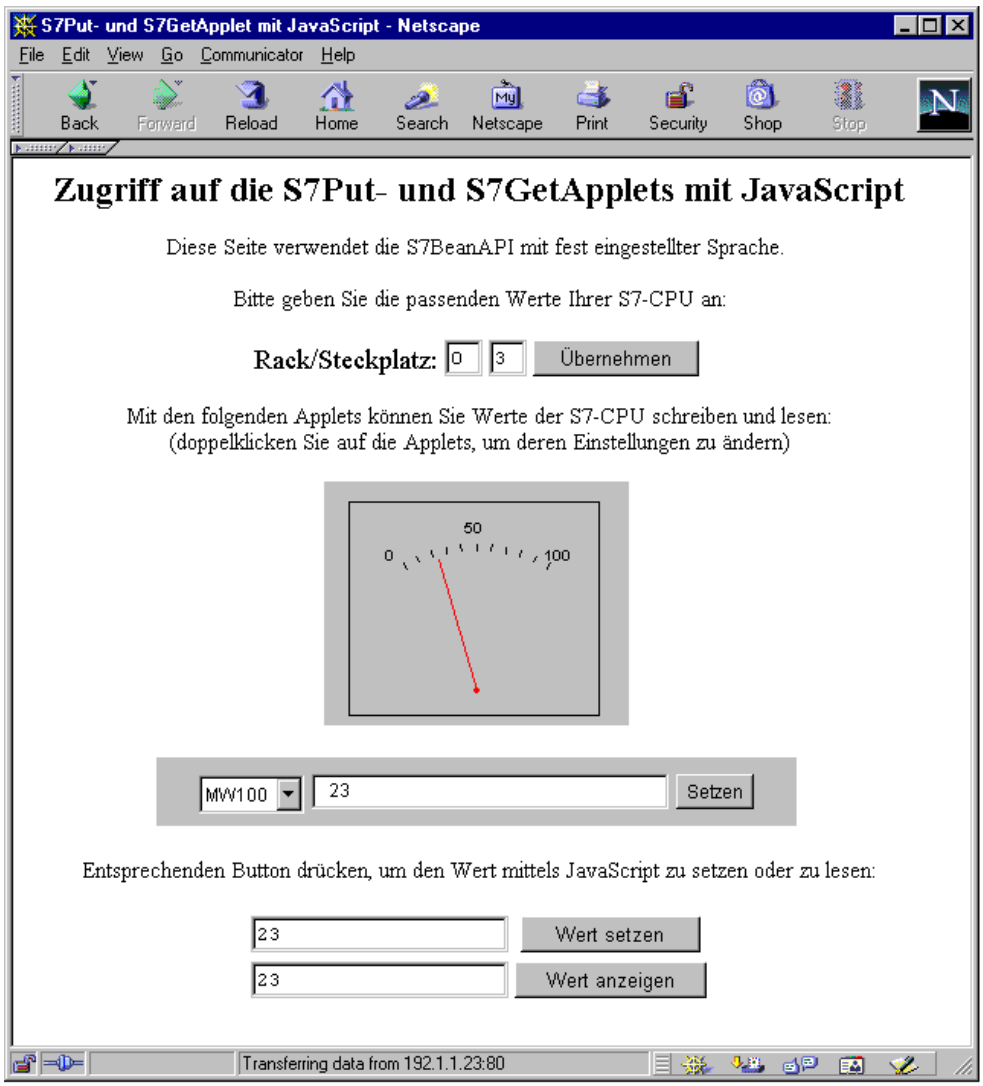
In diesen Beispielen wird der Event Handler onClick verwendet, mit dem Ziel–Links in Formularen definiert werden können. Der Event Handler wird dann aktiviert, wenn der Anwender in der HTML–Seite auf ein Formularfeld klickt.

S7–400 / S7–300

Sie finden die folgenden HTML–Seiten im Filesystem des IT–CP unter /examples/...

3.7.1 S7GetApplet und S7PutApplet

HTML–Beispiel für das Schreiben und Lesen eines Wertes per JavaScript. Das Beispiel beinhaltet ein S7PutApplet zum Setzen eines Wertes und ein S7GetApplet, um den geschriebenen Wert auch verifizieren zu können.



Die Applets sind mittels des Parameters LANGUAGE auf eine feste Sprache eingestellt (im Beispiel in deutsch bzw. englisch). Meldungen in der Javakonsole und Oberflächenelemente des Applets werden daraufhin in deutscher/englischer Sprache angezeigt.

Die Applets sind auch direkt editierbar. D.h. per Doppelklick auf die Appletfläche (Rand) wird ein Konfigurationsdialog geöffnet in dem dann die wichtigsten Applet–Parameter geändert werden können.

Mit den zwei Eingabefeldern Rack/Steckplatz kann die Konfiguration der Hardware angepaßt werden. Dazu werden mittels JavaScript die eingegebenen Werte den

beiden Applets übergeben.

```
<HTML><HEAD>
<meta http–equiv=”Content–Type” content=”text/html; charset=iso–8859–1”>
<TITLE>S7Put– und S7GetApplet mit JavaScript</TITLE>

<script language=”JavaScript”>
<!—
 function setRackSlot(rack, slot) {
                document.Get.setRack(rack);
                document.Get.setSlot(slot);
                document.Put.setRack(rack);
                document.Put.setSlot(slot);

 }
//—>
</script>

</HEAD>

<BODY BGCOLOR=”#FFFFFF”>

<CENTER>
                <H2>Zugriff auf die S7Put– und S7GetApplets mit
                   JavaScript</H2>
                <font size=+0>Diese Seite verwendet die S7BeanAPI mit fest
                  eingestellter Sprache.</font>

<FORM NAME=”config”>
                <font size=+0>Bitte geben Sie die passenden Werte Ihrer S7–CPU
                  an:</font><br><br>
                <font size=+1>Rack/Steckplatz: </font>
                <INPUT TYPE=”text” SIZE=”2” NAME=”rack” VALUE=”0”>
                <INPUT TYPE=”text” SIZE=”2” NAME=”slot” VALUE=”3”>
                <INPUT TYPE=”button” VALUE=”Übernehmen”
                                onClick=”setRackSlot(document.config.rack.value,
                                  document.config.slot.value)”>
</FORM>

<font size=+0>Mit den folgenden Applets können Sie Werte der S7–CPU
  schreiben und lesen:</font><br>
<font size=+0>(doppelklicken Sie auf die Applets, um deren Einstellungen zu
  ändern)</font><br> <br>
<APPLET CODE=”de.siemens.simaticnet.itcp.applets.S7GetApplet.class”
CODEBASE=”/applets/”
                ARCHIVE=”s7applets.jar, s7api.jar” WIDTH=”200” HEIGHT=”160”
NAME=”Get”>
                <PARAM name=”RACK” value= 0>
                <PARAM name=”SLOT” value= 3>
                <PARAM name=”CYCLETIME” value= 10000>
                <PARAM name=”BACKGROUNDCOLOR” value=0xC0C0C0>
                <PARAM name=”LANGUAGE” value=”de”>
```

```
                <PARAM name="DEBUGLEVEL" value=2>
                <PARAM name="DISPLAY" value="CLTacho">
                <PARAM name="MINVAL" value=0>
                <PARAM name="MAXVAL" value=100>
                <PARAM name="EDIT" value="true">

                <PARAM name="VARTYPE"     value=4>
                <PARAM name="VARCNT"      value=1>
                <PARAM name="VARAREA"     value=131>
                <PARAM name="VARSUBAREA"  value=0>
                <PARAM name="VAROFFSET"   value=100>
                <PARAM name="FORMAT"      value="MW100 = \H\H ">
</APPLET>

<br>
<br>

<APPLET CODE="de.siemens.simaticnet.itcp.applets.S7PutApplet.class"
CODEBASE="/applets/"
                ARCHIVE="s7applets.jar, s7api.jar" WIDTH="420" HEIGHT="45"
NAME="Put">
                <PARAM name="RACK" value= 0>
                <PARAM name="SLOT" value= 3>
                <PARAM name="BACKGROUNDCOLOR" value=0xC0C0C0>
                <PARAM name="LANGUAGE" value="de">
                <PARAM name="DEBUGLEVEL" value=2>
                <PARAM name="EDIT" value="true">

                <PARAM name="VARNUM"     value="1">
                <PARAM name="VARNAME1"    value="MW100">
                <PARAM name="VARTYPE1"    value="4">
                <PARAM name="VARAREA1"    value="131">
                <PARAM name="VARSUBAREA1" value="0">
                <PARAM name="VAROFFSET1"  value="100">
                <PARAM name="VARFORMAT1"  value="W">
</APPLET>

<br>

<FORM NAME="form1">
                <font size=+0>Entsprechenden Button drücken, um den Wert
mittels JavaScript zu setzen oder zu lesen:</font><br><br>
                <table width="25%" border="0" align="center">
    <tr>
      <td><INPUT TYPE="text" SIZE="20" NAME="inp"></td>
      <td align="center"><INPUT TYPE="button" VALUE=" Wert setzen "
        onClick="document.Put.setValue(document.form1.inp.value)"></td>
    </tr>
    <tr>
      <td><INPUT TYPE="text" SIZE="20" NAME="out"></td>
      <td align="center"><INPUT TYPE="button" VALUE="Wert anzeigen"
                                onClick="document.form1.out.value = docu-
```

```
ment.Get.getValue()”></td>
    </tr>
                </table>
</FORM>
</CENTER>
</BODY>
</HTML>
```

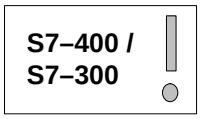
Hinweis

Beachten Sie bitte, dass der Wert, den Sie mit getValue() vom S7GetApplet erhalten, der Wert ist, der dem S7GetApplet zu diesem Zeitpunkt bekannt ist. Der Wert wird nicht direkt aus der SPS abgerufen!

Der Wert, den Sie dem S7PutApplet per setValue() übergeben, wird umgehend an die darunterliegende S7BeansAPI zur Übermittlung an die SPS weitergegeben.

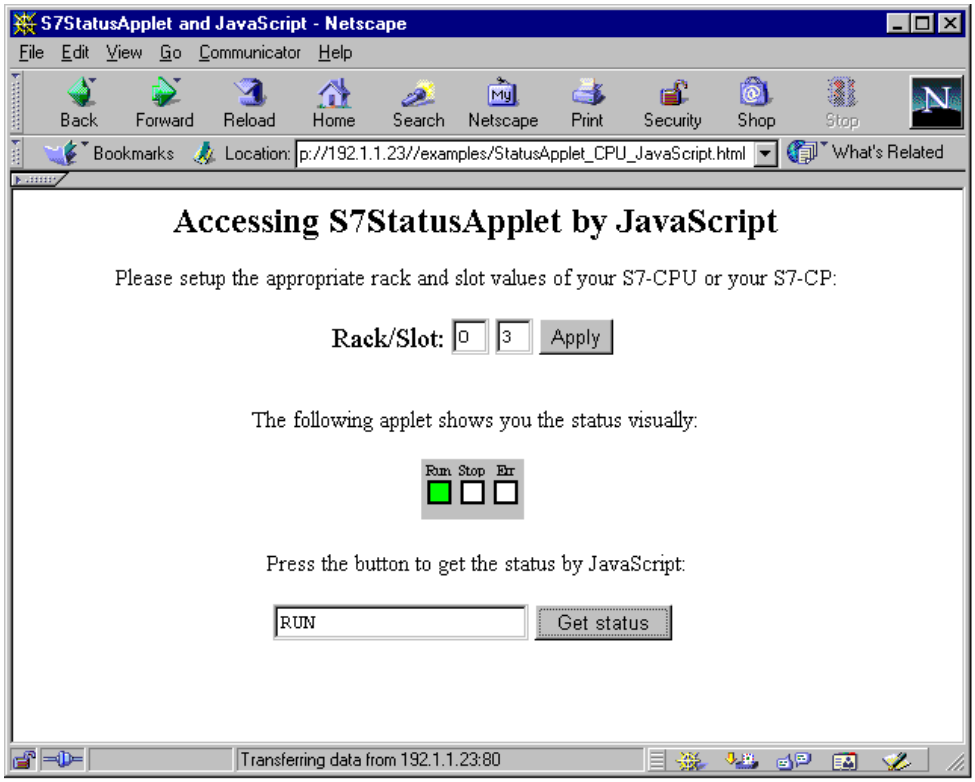
S7GetApplet und S7PutApplet sind zwei getrennt voneinander existierende Programme, die nicht unmittelbar miteinander kommunizieren. D.h., ein Wert, den Sie mit dem S7PutApplet in die SPS geschrieben haben haben, muss nicht unmittelbar dem S7GetApplet bekannt sein.

3.7.2 S7StatusApplet



HTML–Beispiel für den Zugriff auf den Status einer S7–Baugruppe.

Das S7StatusApplet ruft den Status der angegebenen S7–Baugruppe ab und zeigt diesen an.




```
<HTML><HEAD>
<meta http–equiv=”Content–Type” content=”text/html;
charset=iso–8859–1”><TITLE>S7StatusApplet and JavaScript</TITLE>
<script language=”JavaScript”>
<!—
 function setRackSlot(rack, slot) {
                document.State.setRack(rack);
                document.State.setSlot(slot);
 }
//—>
</script>
</HEAD>
<BODY BGCOLOR=”#FFFFFF”>
<CENTER>
                <H2>Accessing S7StatusApplet by JavaScript</H2>
<FORM NAME=”config”>
                <font size=+0>Please setup the appropriate rack and slot values
                   of your S7–CPU or your S7–CP:</font><br> <br>
                <font size=+1>Rack/Slot: </font>
                <INPUT TYPE=”text” SIZE=”2” NAME=”rack” VALUE=”0”>
                <INPUT TYPE=”text” SIZE=”2” NAME=”slot” VALUE=”3”>
                <INPUT TYPE=”button” VALUE=”Apply”
                                onClick=”setRackSlot(document.config.rack.value,
```

```
                                    document.config.slot.value)">
</FORM>
<br>
<font size=+0>The following applet shows you the status visually:</font><br> <br>
<APPLET CODE="de.siemens.simaticnet.itcp.applets.S7StatusApplet.class"
CODEBASE="/applets/"
                ARCHIVE="s7applets.jar, s7api.jar" WIDTH="68" HEIGHT="40"
                  NAME="State">
                <PARAM name="RACK" value= 0>
                <PARAM name="SLOT" value= 3>
                <PARAM name="CYCLETIME" value= 5000>
                <PARAM name="BACKGROUNDCOLOR" value=0xC0C0C0>
</APPLET>
<br>
<FORM NAME="form1">
                <font size=+0>Press the button to get the status by
                   JavaScript:</font><br> <br>
                <INPUT TYPE="text" SIZE="20" NAME="str">
                <INPUT TYPE="button" VALUE="Get status"
                                onClick="document.form1.str.value =
                                  document.State.getState()">
</FORM>
</CENTER>
</BODY>
</HTML>
```

Hinweis

Beachten Sie bitte, dass der Status, den Sie mit getState() vom S7StatusApplet erhalten, der Status ist, der dem S7StatusApplet zu diesem Zeitpunkt bekannt ist. Der Wert wird nicht direkt von der S7–Baugruppe abgerufen!

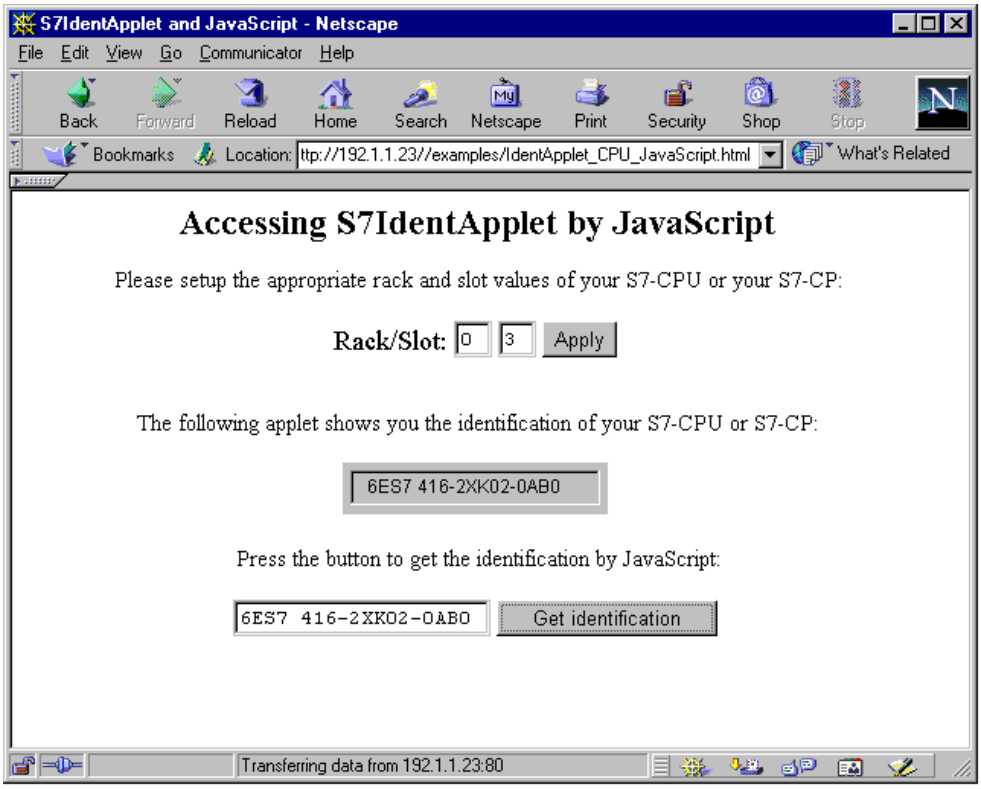
Beachten Sie ferner, dass nicht alle S7–Baugruppen über das gleiche Status–Repertoire verfügen. So unterscheidet der CP 443–1 unter anderem mehrere STOP–Ursachen (z.B. STOP Schalter, STOP intern), während die CPU 416–1 in beiden Fällen STOP intern ausgibt (da sie den Fall STOP Schalter nicht unterscheidet).

3.7.3 S7IdentApplet

S7–400 / S7–300

HTML–Beispiel für den Zugriff auf die MLFB–Nummer einer S7–Baugruppe.

Das S7IdentApplet ruft die MLFB–Nummer der angegebenen S7–Baugruppe ab und zeigt diese an.



```
<HTML><HEAD>
<meta http–equiv=”Content–Type” content=”text/html;
charset=iso–8859–1”><TITLE>
  S7IdentApplet and JavaScript</TITLE>

<script language=”JavaScript”>
<!—
 function setRackSlot(rack, slot) {
                document.Ident.setRack(rack);
                document.Ident.setSlot(slot);

 }
//—>
</script>

</HEAD>
```

```
<BODY BGCOLOR=”#FFFFFF”>

<CENTER>
        <H2>Accessing S7IdentApplet by JavaScript</H2>

<FORM NAME=”config”>
        <font size=+0>Please setup the appropriate rack and slot values of
your S7–CPU
          or your S7–CP:</font><br> <br>
        <font size=+1>Rack/Slot: </font>
        <INPUT TYPE=”text” SIZE=”2” NAME=”rack” VALUE=”0”>
        <INPUT TYPE=”text” SIZE=”2” NAME=”slot” VALUE=”3”>
        <INPUT TYPE=”button” VALUE=”Apply”
          onClick=”setRackSlot(document.config.rack.value,
document.config.slot.
            value)”>

</FORM>

<br>
<font size=+0>The following applet shows you the identification of your S7–CPU or
  ,S7–CP:</font><br> <br>
<APPLET CODE=”de.siemens.simaticnet.itcp.applets.S7IdentApplet.class”
CODEBASE=
  ”/applets/”
        ARCHIVE=”s7applets.jar, s7api.jar” WIDTH=”174”
          HEIGHT=”34” NAME=”Ident”>
        <PARAM name=”RACK” value= 0>
        <PARAM name=”SLOT” value= 3>
        <PARAM name=”BACKGROUNDCOLOR” value=0xC0C0C0>

</APPLET>

<br>

<FORM NAME=”form1”>
        <font size=+0>Press the button to get the identification by
JavaScript:</font><br>
          <br>
        <INPUT TYPE=”text” SIZE=”20” NAME=”str”>
        <INPUT TYPE=”button” VALUE=”Get identification”
                  onClick=”document.form1.str.value =
                    document.Ident.getIdent()”>
</FORM>

</CENTER>
</BODY>
</HTML>
```

3.8 Hilfe zu den S7–Applets

Entnehmen Sie aus der folgenden Aufstellung Hinweise darüber, wodurch es zu Fehlern bei der Verwendung der S7–Applets kommen kann.

Wenn im Betrieb Fehler auftreten, erhalten Sie in der Java Konsole entsprechende Ausgaben (siehe Kap. 2.2 und 3.6).

Alle S7–Applets

- Der Name der Appletklasse, der Parameter CODEBASE oder der Parameter ARCHIVE wurde nicht oder falsch angegeben (Groß–/Kleinschreibung beachten).
- Die Breite und/oder die Höhe des Applets wurde nicht oder mit einem zu großen oder zu kleinen Wert angegeben.
- Die Syntax des Parameter–Tags <PARAM NAME="..." VALUE="..."> ist falsch.
- Ein Parameter wurde vergessen oder falsch geschrieben.
- Der Parameter BACKGROUND für die Hintergrundfarbe des Applets fehlt oder entspricht nicht dem korrekten Wertebereich von 0x000000 bis 0xFFFFFF.
- Der Benutzer besitzt nicht die Rechte, um das Applet auszuführen.
- Die Parameter RACK und SLOT für die Rack–Nummer und die Slot–Nummer entsprechen nicht dem Baugruppenträger und/oder dem Steckplatz, in den die angesprochene Baugruppe tatsächlich gesteckt ist.

Nur S7StatusApplet und S7IdentApplet

- Die angesprochene Baugruppe ist keine CPU / CP.

Nur S7GETApplet und S7StatusApplet

- Der Parameter CYCLETIME für die Zykluszeit fehlt oder entspricht nicht dem Typ Ganzzahl.

Nur S7GETApplet und S7PUTApplet

- Die Symboltabelle wurde nicht gefunden.
- Ein spezifiziertes Symbol wurde nicht in der Symboltabelle gefunden.
- Der ANY–Zeiger enthält ungültige Werte.
- Der Typ eines spezifizierten Parameters entspricht nicht dem erwarteten Typ (z.B. Ganzzahl, Fließkommazahl, Zeichenkette).

Nur S7GETApplet

- Der Formatstring ist syntaktisch nicht korrekt (z.B. durch unbekannte Formatierungszeichen).
- Der Formatstring passt nicht zur Länge der aus der CPU geholten Daten.
- Der maximale Wert und/oder der minimale Wert ist durch die Parameter MINVAL und/oder MAXVAL nicht spezifiziert, falls ein S7–Bean verwendet wird.
- Der Wert im Parameter MAXVAL ist kleiner oder gleich dem Wert im Parameter MINVAL.

Nur S7PUTApplet

- Die Angabe im Parameter SYMBOLNUM und/oder VARNUM entspricht nicht der Anzahl tatsächlich spezifizierter Symbole oder ANY–Zeiger.
- Ein spezifiziertes Symbol ist in der Symboltabelle mit dem ”Nur–Lesen”–Attribut gekennzeichnet

Nur S7StatusApplet

- Die angesprochene Baugruppe ist nicht in der Lage einen Baugruppenstatus zu liefern.
- Falls von mehreren Baugruppen der Status angezeigt werden soll und die Zykluszeit zu kurz gewählt ist, kann es zu häufigen Fehlern beim Verbindungsaufbau kommen.

❑

4 Individuelle Lösungen mit JavaBeans

Oftmals werden Sie eine graphische Darstellung gelesener Prozesswerte gegenüber einer numerischen Darstellung vorziehen. In den Fließbildern von Bedien– und Beobachtungssystemen sind solche Darstellungsformen, wie zum Beispiel Füllstandsanzeigen oder Temperaturskalen, üblich.

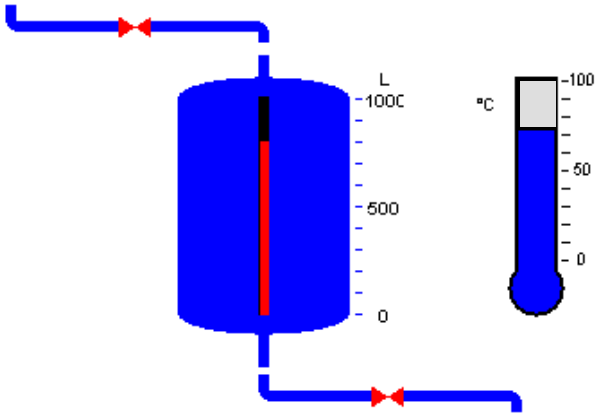


Bild 4-1

Das in Java verfügbare JavaBeans–Konzept ermöglicht es, Objekte (Java Komponenten) zu erstellen und auf einfache Weise zu ausführbaren Programmen zu verbinden. Auch für den Zugriff auf die in einer SIMATIC S7–CPU geführten Prozessdaten ermöglichen JavaBeans sehr flexible Möglichkeiten.

Für Sie als IT–CP Nutzer steht eine S7–Beans Klassenbibliothek (S7BeansAPI) für die Java–Programmierung zur Verfügung. Die in dieser S7–Beans Klassenbibliothek enthaltenen Objektklassen können Sie für einen objektorientierten Zugang zu unterschiedlichen Informationen der SIMATIC S7 und für eine graphische Darstellung von Prozessvariablen nutzen.

4.1 JavaBeans–Konzept und Anwendungsmöglichkeiten

Standard–Anwendung: mit S7GetApplet können Sie bereits S7–Beans nutzen

Bereits für das “S7GetApplet” des IT–CP können Sie über den optionalen Parameter DISPLAY eine gewisse Palette an graphischen Elementen aus der S7–Beans–Klassenbibliothek für die Darstellung in Ihrer HTML–Seite nutzen (siehe Parameter DISPLAY in Kap.3.4.7). Die entsprechend verwendbaren S7–Beans sind in der Tabelle 4-2 im Folgekapitel gekennzeichnet.

Erweiterte Anwendungen: S7–Beans über “Builder Tools” konfigurieren

Um die vollständige Palette der S7–Beans in der S7–Beans Klassenbibliothek nutzen zu können, setzen Sie in Ihrer Java–Programmierumgebung sogenannte Builder–Tools ein.

Mit diesen Builder–Tools haben Sie die Möglichkeit, eine objektorientierte Konfiguration für Ihre Anwendung vorzunehmen. Dies umfasst vereinfacht dargestellt die folgenden Schritte

- die gewünschten S7–Beans auszuwählen;
- die S7–Beans miteinander zu verbinden und damit den Datenfluss festzulegen;
- S7–Beans Parametrierungen festzulegen.

Bereits ohne großen Aufwand können Sie mit dieser Technik komplexe Prozessdarstellungen in Ihrer IDE konfigurieren und programmieren.

Für den versierten Java–Anwender beinhaltet dies nahezu beliebig weit gehende Möglichkeiten, die über den IT–CP erfassbaren Prozessdaten – beispielsweise in Datenbanken oder Management–Informationssystemen – weiter zu verarbeiten.

Hinweis

Zum Entwickeln neuer Applets und eigener Beans beachten Sie die Runtime–Version im Filesystem und die Entwicklungsversion auf der Manual Collection CD.

4.2 Die S7–Beans Klassenbibliothek (S7BeansAPI)

Anwendung: Manual Collection CD

Die S7–Beans Klassenbibliothek befindet sich auf der Manual Collection CD.

Die verfügbaren S7–Beans

Die folgenden Tabellen geben eine Übersicht über die derzeit mitgelieferten S7–Beans. Anhand dieser Tabelle können Sie sich eine Vorstellung über die Gestaltungsmöglichkeiten machen, die Ihnen diese mitgelieferten S7–Beans an die Hand geben.

Zu unterscheiden sind hierbei

- S7–Beans für Geräte

  Für die im SIMATIC S7–Rack ansprechbaren Baugruppen und Software–Objekte werden in der S7BeansAPI diese JavaBeans angeboten. Sie stellen im Programm die Verbindung zu den S7–Beans für die Ein– und Ausgabe (S7–Beans für den Client) her.

- S7–Beans für den Client

  Für die grafische Ausgabe der Prozessdaten in Prozessbildern auf dem Client werden in der S7BeansAPI diese JavaBeans angeboten.

- S7 Utility–Beans für den Client

  Für die zusätzliche Aufbereitung von Daten stehen die Utility–Beans zur Verfügung. Da es sich um reine Konvertierungsfunktionen handelt, gibt es keine unmittelbare grafische Darstellung durch diese Beans.

  Diese S7–Beans befinden sich im JAR–File s7util.jar.

Tabelle 4-1 S7–Beans für Geräte / Objekte – Package = API (ist im JAR–File s7api.jar enthalten)

S7–Bean	Funktion
S7CP	Diese Bean repräsentiert den IT–CP der als Host dient. Eventuell weitere vorhandene IT–CPs sind über S7Device anzusprechen. Diese Bean muss auf jedem Applet zur Adressierung und Speicherung der Hostadresse verwendet werden .
S7Device	S7Device repräsentiert eine beliebige intelligente S7–Baugruppe wie beispielsweise CPU, PROFIBUS–CP, Ethernet–CP, weitere IT–CPs (jedoch in keinem Fall denjenigen IT–CP, der als Host für die Applets dient, der also über den Browser angesprochen werden soll !)

Tabelle 4-1 S7–Beans für Geräte, Fortsetzung/ Objekte – Package = API (ist im JAR–File s7api.jar enthalten), Fortsetzung

S7–Bean	Funktion
S7Variable	Diese Bean repräsentiert Variablen in der S7–CPU.
CLTimer	CLTimer wird für den zyklischen Aufruf von Methoden anderer Beans benötigt. Immer wenn Sie einen Status einer S7–Baugruppe oder eine Prozessvariable dauernd (zyklisch) beobachten wollen, benötigen Sie diese Bean. Hinweis: CLTimer verfügt über **keine** graph. Darstellung

Tabelle 4-2 S7–Beans für den Client – Package = GUI (ist im JAR–File s7gui.jar enthalten)

S7–Bean	Funktion	in S7GetApplet nutzbar	Darstellung
CLTextIn	CLTextIn ist eine Bean zur Eingabe von Text. Dieser Text kann an die Bean S7Variable weitergeleitet werden.	nein	– Eingabefeld –
CLTextOut	CLTextOut ist eine Bean zur textuellen Ausgabe von Werten von Prozessvariablen der Bean.	nein	
CLIdentOut (nur S7–300/400)	CLIdentOut ist eine Bean, die für die textliche Ausgabe einer Identnummer eines IT–CPs oder einer Baugruppe über die Bean S7CP oder S7Device benötigt wird. Hinweis: Nicht alle Baugruppen unterstützen diesen Dienst.	nein	
CLStateLED (nur S7–300/400)	CLStateLED ist eine Bean zur grafischen Darstellung des Status eines IT–CPs oder einer Baugruppe. Die Darstellung erfolgt über die Farbe der LED: • grün: RUN • gelb: STOP • rot: Fehlermeldung der Baugruppe • blau: Verbindungsfehler • grau: Status unbekannt Hinweis: Nicht alle Baugruppen unterstützen diesen Dienst.	nein	

Tabelle 4-2 S7–Beans für den Client – Package = GUI (ist im JAR–File s7gui.jar enthalten), Fortsetzung

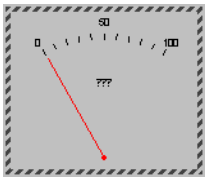
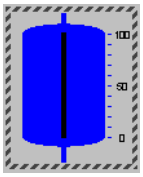
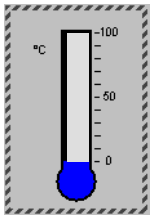
S7–Bean	Funktion	in S7GetApplet nutzbar	Darstellung
CLState3LED (nur S7–300/400)	CLState3LED ist eine Bean zur grafischen Darstellung des Status eines IT–CPs oder einer Baugruppe. Die Darstellung erfolgt über drei LEDs: • grün: RUN • gelb: STOP • rot: Fehlermeldung der Baugruppe • blau: Verbindungsfehler • grau: Status unbekannt Hinweis: Nur intelligente Baugruppen unterstützen diesen Dienst; IO–Baugruppen beispielsweise nicht.	nein	
CLTacho	CLTacho ist eine Bean zur grafischen Darstellung eines Zeigerinstrumentes. Der Zeiger stellt den Wert einer Prozessvariablen dar. Zusätzlich kann das Erreichen eines oberen oder unteren Grenzwertes mittels LED–Anzeige sichtbar gemacht werden. Die Bean ist in ihrer Größe skalierbar.	ja	
CLLevel	CLLevel ist eine Bean zur grafischen Darstellung eines Füllstandswertes einer Prozessvariablen. Zusätzlich kann das Erreichen eines oberen oder unteren Grenzwertes mittels LED–Anzeige sichtbar gemacht werden. Die Bean ist in ihrer Größe skalierbar.	ja	
CLThermo	CLThermo ist eine Bean zur grafischen Darstellung einer Prozessvariablen als Temperaturwert. Zusätzlich kann das Erreichen eines oberen oder unteren Grenzwertes mittels LED–Anzeige sichtbar gemacht werden. Die Bean ist in ihrer Größe skalierbar.	ja	

Tabelle 4-2 S7–Beans für den Client – Package = GUI (ist im JAR–File s7gui.jar enthalten), Fortsetzung

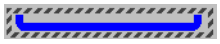
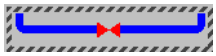
S7–Bean	Funktion	in S7GetApplet nutzbar	Darstellung
CLPipe	CLPipe ist eine Bean zur grafischen Darstellung eines horizontalen oder vertikalen Rohres. Die Farbe des Rohres wechselt mit einem boolschen Wert. Die Bean ist in ihrer Größe skalierbar.	nein	
CLValve	CLValve ist eine Bean zur grafischen Darstellung eines Ventils. Das Ventil mit Zuleitung kann horizontal oder vertikal dargestellt werden. Das Öffnen und Schliessen erfolgt über das Setzen eines boolschen Wertes.	nein	

Tabelle 4-3 S7 Utility–Beans – Package = UTIL (ist im JAR–File s7util.jar enthalten)

S7 Utility–Bean	Funktion
COUNTER	Liefert den Zählerstand (z.B. Z1) als String im Format C#347.
TIMER	Liefert den Wert von Zeiten (z.B. T1) als String im Format S5T#1h3m2s.
DATE	Liefert den S7–Type DATE als String im Format D#2000–12–31.
TIME	Liefert den S7–Type TIME als String im Format T#9h6m6s.
DATEandTIME	Liefert den S7–Type DATE_AND_TIME als String im Format DT#00–12–31–12:31:47.487.
TIMEofDAY	Liefert den S7–Typ Time Of Day als String im Format TOD#9:6:6.127.
S5TIME	Liefert den S7–Type S5TIME als String im Format S5T#1h3m2s.
ConvertNumberSystem	Liefert eine Dezimalzahl wahlweise hexadezimal, oktal oder dual (als String). Außerdem können Dezimalzahlen in BCD–Zahlen umgewandelt werden und umgekehrt.

4.3 S7–Beans verknüpfen

Damit die S7–Beans in Ihrer Anwendung zusammenspielen, müssen diese sinnvoll miteinander verknüpft werden. Mit Werkzeugen, wie beispielsweise Visual Age, das wir Ihnen im Kapitel 5 noch näher beispielhaft vorstellen, können Sie diese Verknüpfung graphisch unterstützt durchführen.

Zunächst sollten Sie jedoch eine Übersicht darüber bekommen, wie die S7–Beans sinnvoll zu verschalten sind.

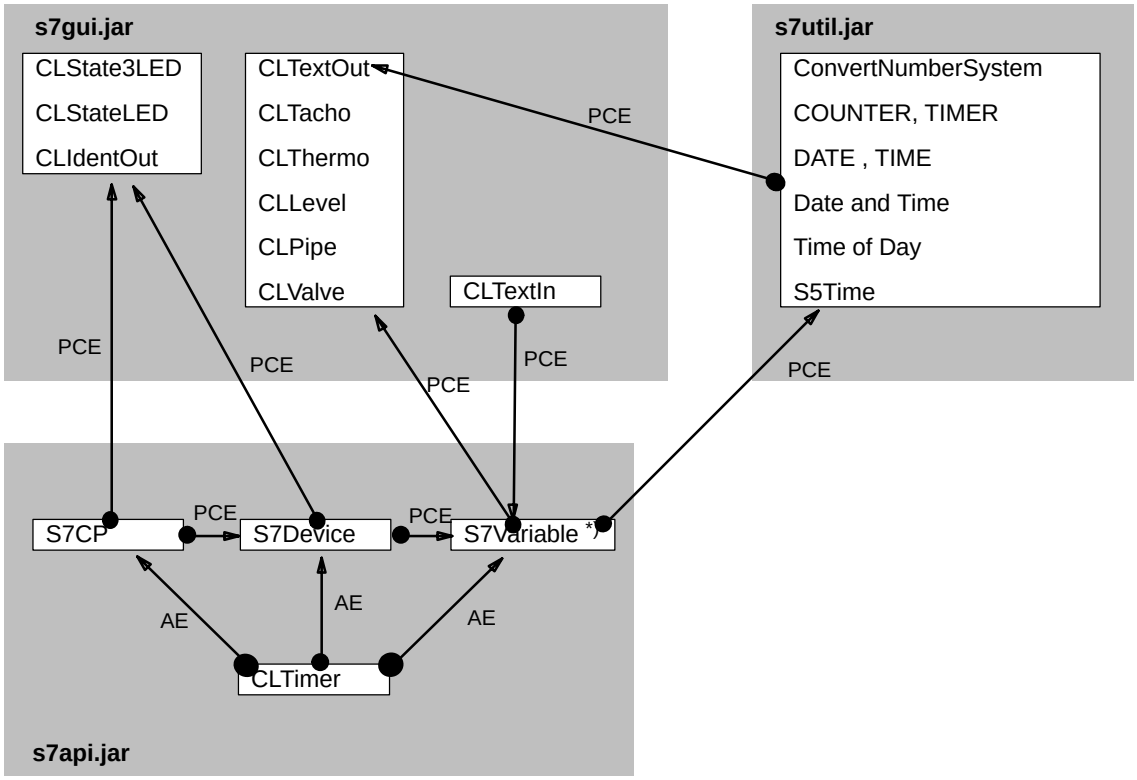
Die folgende Grafik zeigt auf, welche Verbindungen möglich sind; hierbei bedeuten:

AE: ActionEvent

PCE: PropertyChangeEvent

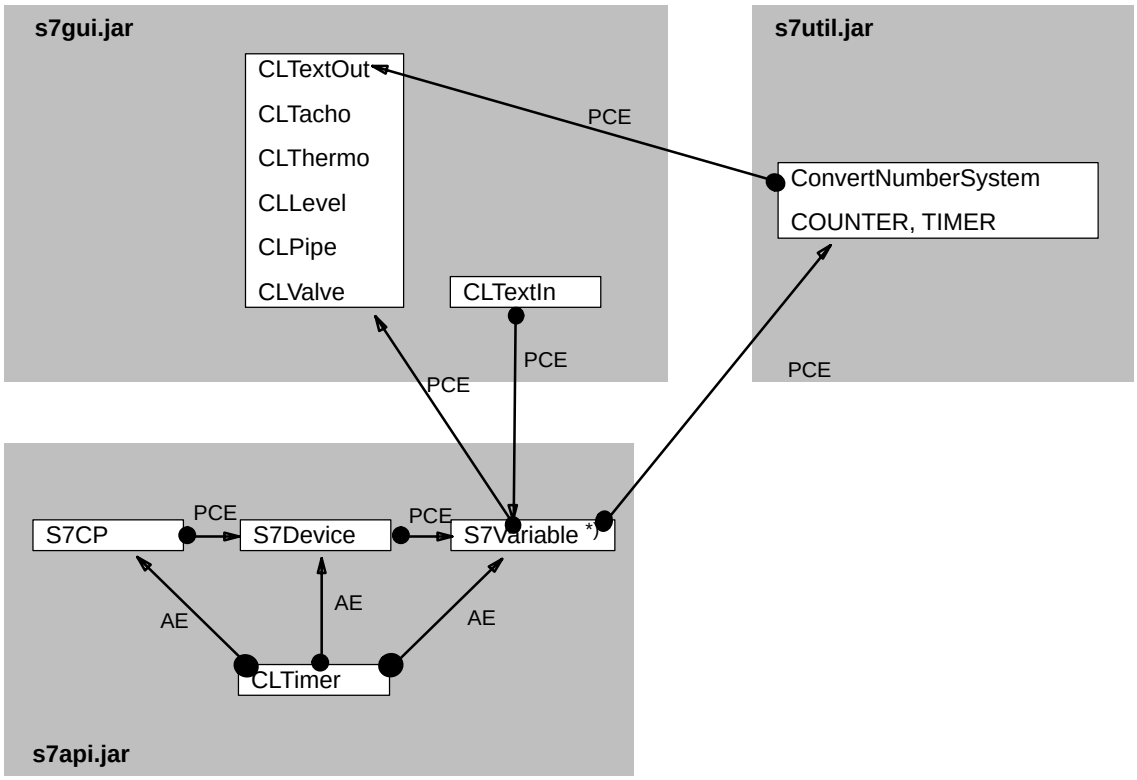
Am Punktende des Pfeils befindet sich dabei jeweils der Produzent des Ereignisses, die Pfeilspitze zeigt auf den Listener.

S7–Beans beim CP 343–1 IT und CP 443–1 IT verknüpfen



*) Bitte beachten Sie zu den Variablentypen bei S7–300 / S7–400 die Tabellen in Kapitel C

S7–Beans beim CP 243–1 IT verknüpfen



*) Bitte beachten Sie zu den Variablentypen bei S7–200 die Tabellen in Kapitel C

❑

5 S7BeansAPI – Schnittstellenbeschreibung

Hier werden die Schnittstellen der ”nicht–sichtbaren” S7API und S7–Beans für Geräte beschrieben. Sie erfahren, wie Sie Instanzen der Beans erzeugen (Konstruktor–Aufrufe) und welche Zugriffsmethoden es gibt.

Die S7–Beans zur Ein– und Ausgabe von Werten übergeben bzw. erhalten ihre Werte per PropertyChangeEvent (siehe Kapitel 4.3: S7–Beans verknüpfen).

Lesen Sie hierzu auch die JavaDoc zur S7BeansAPI, die mit der Manual Collection CD ausgeliefert wurde.

5.1 S7API–Methoden

Seit der S7BeansAPI–Version V2.3 enthält das Package **`de.siemens.simaticnet.itcp.api`** die neue Klasse **S7Api**. Diese Klasse enthält eine Reihe von nützlichen Hilfsfunktionen die im folgenden vorgestellt werden.

5.1.1 Steuerung der verwendeten Sprache

Mittels der Methode **S7Api.setLocale(String newLanguage, String newCountry)** kann zur Laufzeit die für Meldungen und Beschriftungen verwendete Sprache eingestellt werden. Der Parameter **newLanguage** ist dabei ein nach ISO–639 definierter Sprachcode und **newCountry** der Ländercode nach ISO–3166. Die S7BeanApi wird standardmäßig mit englischen und deutschen Ressourcendateien ausgeliefert. Diese Sprachdateien liegen in den jeweiligen JAR–Archiven und tragen die Endung *.properties. Auf Basis dieser mitgelieferten Ressourcen–Dateien ist es aber auch möglich eigene Übersetzungen anzufertigen und somit auch andere Sprachen außer deutsch und englisch zu verwenden. Im folgenden ein Beispielaufruf der Methode **setLocale()** wie er in der **init()**–Methode eines Ihrer Applets stehen könnte:

```
S7Api.setLocale(”de”, ””);
```

Dieser Aufruf setzt die Sprache der S7BeansAPI fest auf neutrales Deutsch, da kein Ländercode angegeben ist.

Hinweis

Wird keine Sprache eingestellt, so verwendet die S7BeansAPI genau wie auch während der Entwicklungsphase in der IDE die Sprache des Hosts, sofern diese auch verfügbar ist.

5.1.2 Einstellen des Detaillevels der Debugausgaben

Um Ressourcenprobleme zu vermeiden, werden Debug–Meldungen in der Java–Konsole werden ab der S7BeansAPI–Version V2.3 standardmäßig nur noch bei Fehlern ausgegeben. Mit der neuen S7Api–Methode **`setDebugLevel(int level)`** kann zur Laufzeit der Detaillevel spezifiziert werden. Dabei können für den Integer–Parameter **level** folgende Werte übergeben werden:

0 = VOID_LEVEL: Keine Meldungen ausgeben

1 = LOG_LEVEL: Alle möglichen Meldungen

2 = WARN_LEVEL: Nur Warnungen und Fehler ausgeben

3 = ERR_LEVEL: Nur Fehlermeldungen ausgeben (Standardwert)

4 = FATAL_LEVEL: Nur schwere, nicht behebbare Fehler ausgeben

Mit folgendem Aufruf werden z.B. nur noch Warnungen und Fehlermeldungen ausgegeben:

```
S7Api.setDebugLevel(2);
```

Hinweis

Die Meldungen (Warnungen, Fehler etc.) werden mit Zeitstempel und Objektinfo ausgegeben. Dadurch können Probleme und das zeitliche Ablaufverhalten besser analysiert werden.

5.1.3 Terminieren der API–Mechanismen

Um Ressourcen–Probleme beim Einsatz der mit den Beans entwickelten Applets zu vermeiden, sollten alle Ressourcen beim Verlassen der zugehörigen HTML–Seite freigegeben und alle Threads gestoppt werden. Da die S7BeansAPI intern statische Ressourcen und Threads für die Kommunikation mit dem IT–CP verwendet, wurde eigens die Methode **terminate()** in der Klasse **S7Api** geschaffen, um diese Ressourcen freizugeben und alle Threads zu stoppen.

Terminate() kann normalerweise nach dem Freigeben aller eigenen Ressourcen in der destroy()–Methode des Applets aufgerufen werden. Da jedoch einige Browser (z.B. Netscape) die destroy()–Methode nicht direkt bei Verlassen der HTML–Seite aufrufen, sondern erst bei Entfernen des Applets aus dem History–Cache des Browsers, ist es ratsam die Ressourcen schon in der stop()–Methode freizugeben und auch gleich terminate() aus der S7Api–Klasse aufzurufen. Dies erfordert allerdings dann ein erneutes Initialisieren sämtlicher Ressourcen in der start()–Methode des Applets, da das Applet z.B. unter Netscape auch durch ein Ändern der Browser–Fenstergröße gestoppt und wieder gestartet wird.

5.2 S7 Beans

5.2.1 S7CP

Beschreibung:

Die S7–Bean S7CP ist die Verbindungskomponente zu dem CP 343–1 IT / CP 443–1 IT, über den Sie Zugang zur SPS erlangen möchten.

(Anmerkung: Aus einem Applet heraus können Sie immer nur auf **den einen** IT–CP zugreifen, von dem das Applet geladen wurde (Sandbox); evtl. weitere IT–CPs in der SPS können Sie nur über die S7–Bean S7Device ansprechen).

Instanz erzeugen – Konstruktor–Aufruf:

Um eine Instanz des S7CPs zu erzeugen, rufen Sie den Standard–Konstruktor auf und setzen anschließend die Properties **host** und **modulName**.

Konstruktor–Aufruf:

Alternativ können Sie auch folgenden Konstruktor aufrufen und die Parameter direkt setzen. Beachten Sie hierbei bitte, dass die Parameter rack und slot in diesem Fall beide auf 0 (Null) gesetzt werden müssen.

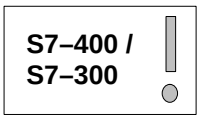
S7CP(String modulName, String host, int rack, int slot)

Properties setzen:

Wenn Sie den Standard–Konstruktor verwendet haben, müssen Sie noch folgende Properties setzen:

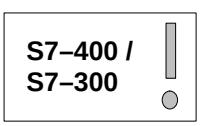
- setModulName(String modulName): Eindeutiger Bezeichner für diese Instanz.
- setHost(String host): IP–Adresse des IT–CP.

Identifikation der Baugruppe (MLFB) abrufen (nur S7–300 / S7–400):



processIdent(): Initiiert die Anforderung der MLFB–Nummer. Die Methode terminiert umgehend. Sobald die angeforderte Information vorhanden ist, wird diese per PropertyChangeEvent (propertyName = identification) an die Listener gesendet.

Statusinformation der Baugruppe abrufen (nur S7–300 / S7–400):



processState(): Initiiert die Anforderung der Statusinformation. Die Methode terminiert umgehend. Sobald die angeforderte Information vorhanden ist, wird diese per PropertyChangeEvent (propertyName = state) an die Listener gesendet.

5.2.2 S7Device

Beschreibung:

Die S7–Bean S7Device repräsentiert eine (intelligente) Baugruppe, wie z.B. eine CPU oder einen CP. Ein S7Device kann auch ein weiterer IT–CP sein, über den sie nicht vom Applet aus auf die SPS zugreifen möchten.

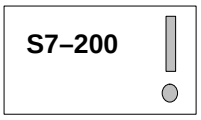
Jedes S7Device muss beim S7CP als PropertyChangeListener angemeldet sein! Sie können mehrere S7Devices bei einem S7CP anmelden.

Instanz erzeugen – Konstruktor–Aufruf:

Um eine Instanz des S7Device zu erzeugen, rufen Sie den Standard–Konstruktor auf und setzen anschließend die Properties *rack, slot* und *modulName.*

Properties setzen:

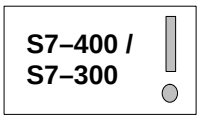
- setModulName(String modulName): Eindeutiger Bezeichner für diese Instanz.
- setRack(int rack): Nummer des Baugruppenträgers.
- setSlot(int slot): Nummer des Steckplatzes.



Achtung

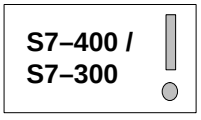
bei S7–200 geben Sie bitte an: Rack =0; Slot =0

Identifikation der Baugruppe (MLFB) abrufen (nur S7–300 / S7–400):



processIdent(): Initiiert die Anforderung der MLFB–Nummer. Die Methode terminiert umgehend. Sobald die angeforderte Information vorhanden ist, wird diese per PropertyChangeEvent (propertyName = identification) an die Listener gesendet.

Statusinformation der Baugruppe abrufen:



processState(): Initiiert die Anforderung der Statusinformation. Die Methode terminiert umgehend. Sobald die angeforderte Information vorhanden ist, wird diese per PropertyChangeEvent (propertyName = state) an die Listener gesendet.

5.2.3 S7Variable

Beschreibung:

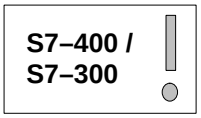
Die S7–Bean S7Variable repräsentiert einen Speicherbereich auf der CPU, den Sie entweder per ANY–Zeiger adressieren oder über eine symbolische Adresse ansprechen können (letztere muss in der Hardware–Konfiguration des IT–CPs eingetragen sein!).

Jede S7Variable muss beim zugehörigen S7Device als PropertyChangeListener angemeldet sein! Sie können mehrere S7Variablen bei einem S7Device anmelden.

Konstruktor–Aufruf:

Um eine Instanz der S7Variable zu erzeugen, rufen Sie den Standard–Konstruktor auf und setzen anschließend die Properties *symbolName* bzw. *s7AnyPointer* und *variableName.*

Properties setzen:



- setSymbolName(String symbolName): Symbolische Adresse.
- setS7Anypointer(S7Anypointer anypointer): ANY–Zeiger Adresse.
- setVariableName(String variableName): Eindeutiger Bezeichner für diese Instanz.

Die symbolische Adressierung und die ANY–Zeiger Adressierung sind alternativ zu verwenden. Werden beide angegeben, hat die symbolische Adresse Vorrang.

Einen Speicherbereich auslesen:

processGet(): Initiiert die Anforderung, einen Speicherbereich auszulesen. Die Methode terminiert umgehend, der Lesevorgang läuft asynchron ab. Sobald die angeforderte Information vorhanden ist, wird diese per PropertyChangeEvent (propertyName: der im Property *variableName* hinterlegte String) an die Listener gesendet.

Object getValue(): Liefert den in der S7Variablen abgelegten Inhalt des Speicherbereichs. Achtung, diese Methode liefert nicht zwangsläufig den aktuellen Inhalt des adressierten Speicherbereiches der SPS! Falls zuvor noch kein Wert mittels *processGet()* ausgelesen wurde, wird hier auch kein Wert zurückgegeben!

Einen Speicherbereich schreiben:

setValue(Object newValue): Initiiert das Schreiben eines Wertes in den Speicherbereich der CPU. Die Methode terminiert umgehend, der Schreibvorgang verläuft asynchron.

5.2.4 CLTimer

Beschreibung:

Die S7–Bean CLTimer erzeugt zyklisch ein ActionEvent und kann somit zum periodischen Auslesen von Statusinformationen und Speicherbereichen als Trigger verwendet werden.

Jede S7–Bean, die periodisch eine Aktion ausführen soll, muss beim CLTimer als ActionListener angemeldet werden. Dabei muss angegeben werden, welche Methode beim Eintreffen des Ereignisses ausgeführt werden soll (bei S7CP und S7Device **processState()** und bei S7Variable **processGet()**). Es können mehrere Beans beim CLTimer als ActionListener angemeldet werde. Es können aber auch mehrere Instanzen von CLTimer (z.B. mit unterschiedlichen Zeiten) verwendet werden.

Konstruktor–Aufruf:

Um eine Instanz von CLTimer zu erzeugen, rufen Sie den Standard–Konstruktor auf und setzen anschließend bei Bedarf das Property *delay.*

Property setzen:

setDelay(int delay): Zykluszeit in Millisekunden (Default–Wert ist 5.000 ms).

CLTimer steuern:

start(): Startet den CLTimer wieder von Neuem nach einem stop()–Aufruf. Nach der Instanziierung startet der CLTimer selbstätig und muss nicht mit start() angestoßen werden.

stop(): Stoppt den CLTimer.

6 Beispiele

6.1 Beispiele mit S7Beans und AWT–Komponenten

In den folgenden Unterkapiteln finden Sie Beispiele, die die Benutzung der S7Beans in eigenen Applets verdeutlichen.

Die Beispiele sind mit dem kompletten Quelltext abgedruckt und auch lauffähig. Lediglich die IP–Adresse des IT–CP und dieRack/Slot–Nummer der S7–CPU müssen im Quelltext noch angepasst werden.

Sie können die Beispiele in einer Java–Entwicklungsumgebung wie VisualAge von IBM oder JBuilder von Borland importieren, oder direkt mit dem Java Development Kit (JDK) kompilieren. Das fertige Applet transferieren Sie dann mittels FTP auf Ihren IT–CP. Zusätzlich brauchen Sie noch eine geeignete HTML–Seite, die als Container für das Applet dienen soll und den Aufruf des Beispiel–Applets ermöglicht.

Im folgenden hierzu eine ganz einfache HTML–Seite, die einfach nur eine Titelzeile darstellt und das Beispiel–Applet 1 startet:

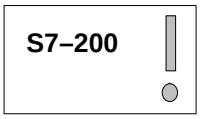
```
<HTML>
<HEAD>
<TITLE>Example 1</TITLE>
</HEAD>
<BODY>
<H1>Example 1</H1>
<APPLET CODE=de.siemens.simaticnet.itcp.example.Example1.class AR-
CHIVE=Examples.jar WIDTH=200 HEIGHT=100>
</APPLET>
</BODY>
</HTML>
```

Für die weiteren Beispiele brauchen Sie nur die Nummer hinter ”Example” anzupassen. Die Angabe ”ARCHIVE” geht übrigens davon aus, dass die HTML–Seite und das entsprechende Beispiel–Applet im gleichen Pfad auf dem IT–CP liegen.

Beispiele unter S7–200 / CP 243–1 IT verwenden

Achtung

Hier beschriebene Beispiele sind prinzipiell für die hier dokumentierten IT–CPs anwendbar. An einigen Stellen werden jedoch Parametrierungen verwendet, die auf S7–300 und S7–400 abgestimmt sind.



Beachten Sie daher bitte für eine Anwendung mit CP 243–1 IT:

- Passen Sie bei der Angabe von Merkerbereichen die Einträge auf die bei S7–200 zulässigen Merkerbereiche an!
- Beachten Sie gekennzeichnete Stellen, beispielsweise die Angaben zu den Steckplätzen, die eine besondere Eingabe erfordern.
- Rack / Slot Nummer der angesprochenen Baugruppe bei S7–200 immer R/S=0/0

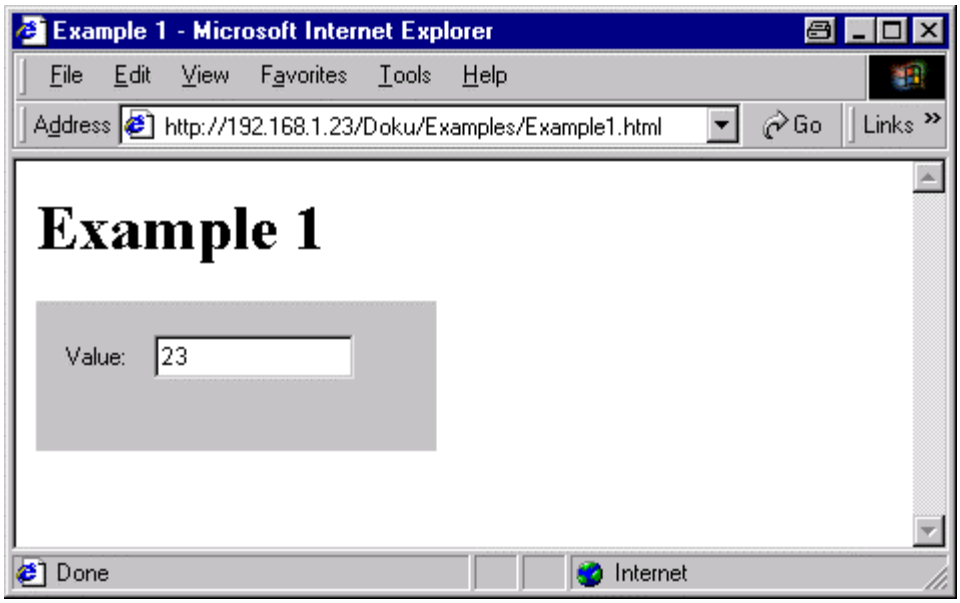
6.1.1 Beispiel 1 – Eine Variable mit S7Beans aus der S7 lesen und anzeigen

Funktionsweise

Es soll eine Variable aus der SIMATIC S7 gelesen und angezeigt werden.

Gelesen wird hierbei das MW 10 als INT.

Darstellung



Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
import de.siemens.simaticnet.itcp.gui.*;
/**
 * Example1.java
 * <p>Überschrift: Beispiel 1 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Ausgabe eines Wertes über CLTextOut Bean mit einer S7Variable.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Ausgabe eines Wertes über CLTextOut Bean mit einer S7Variable.
 * Gelesen wird das Merkerwort MW10.
```

```
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * CLTimer
 * CLTextOut
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example1 extends Applet implements PropertyChangeListener, ActionListener {
   /*-----------------------------------Implementierte Interfaces*/
   /*----------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private CLTextOut cLTextOut1 = null;
   private CLTimer cLTimer1 = null;
   private S7CP s7CP1 = null;
   private S7Device s7Device1 = null;
   private S7Variable s7Variable1 = null;
   /**
    * Wird immer aufgerufen, wenn das Applet initialisiert wird.
    * Dies geschieht sofort, nachdem es geladen ist.
    *
    * @see #start
    * @see #stop
    * @see #destroy
    */
   public void init() {
      super.init();
      // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
      setName("Example1");
      // Setzt den Layoutmanager für diese Komponente.
      setLayout(null);
      // Setzt die Größe des Applet Breite / Höhe
      setSize(426, 240);
      /*-----------Höhe*/
      /*------Breite*/
      // Anlegen einer Instanz für das S7CP Bean.
      // S7CP ist der Ethernet Zugangspunkt in die Station
      s7CP1 = new S7CP();
      // Zuweisen der IP-Adresse
      // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
      s7CP1.setHostString(new HostString ("192.168.1.1:80"));
      /*-----------------------------------------------Angabe der Portnummer
                                                 normalerweise :80*/
      /*-----------------------------------IP-Adresse als String*/
      // Anlegen einer Instanz für das S7Device Bean.
      // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
      s7Device1 = new S7Device();
      // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
      // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
      // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
      // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
      // eingetragen werden.
      // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
      // ######## notwendig                                          ########
      s7Device1.setSlot(2);
      /*----------------Steckplatznummer 2 (int)*/
      // Anlegen einer Instanz für das S7Variable Bean.
      // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
      // werden soll.
      s7Variable1 = new S7Variable();
      // Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
```

```
    s7Variable1.setS7Anypointer(
       new S7Anypointer((int)5, (int)1, (int)131, (int)0, (int)10, (int)0));
    /*-------------------------------------------------------------Bit Nummer 0 ..7*/
    /*-----------------------------------------------------Speicherbereichoffset*/
    /*-------------------------------------------DB Nummer oder '0'*/
    /*-----------------------------------Speicherbereich 131 == M*/
    /*--------------------------Wiederholfaktor 1 .. n*/
    /*------------------Datentyp 5 == INT*/
    // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
    s7Variable1.setVariableName("s7Variable1");
    // Anlegen einer Instanz für das CLTimer Bean.
    // Das CLTimer Bean löst nach Ablauf der Zeit ein PropertyChangeEvent
    // Ereignis aus. Über dieses Ereignis wird eine zyklische Datenaktualisierung
    // realisiert.
    cLTimer1 = new CLTimer();
    // Mit der Methode setDelay() wird das Zeitintervall festgelegt.
    cLTimer1.setDelay(2000);
    /*----------------Zeitintervall in msec. 2000 == 2 sec.*/
    // Anlegen einer Instanz für das CLTextOut Bean.
    // Über das CLTextOut Bean können elementare Variablen vom Anwender eingegeben
    // werden.
    cLTextOut1 = new CLTextOut();
    // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
    cLTextOut1.setName("cLTextOut1");
    // Festlegen der Startposition und Größe der Komponente.
    cLTextOut1.setBounds(0, 0, 200, 45);
    /*-----------------------------Komponentenhöhe*/
    /*--------------------------Komponentenbreite*/
    /*----------------------Startposition Y*/
    /*------------------Startposition X*/
    // Festlegen der Größe des Ausgabefeldes.
    cLTextOut1.setOutFieldSize(100);
    /*-------------------------Länge Ausgabefeld*/
    // Setzt den Beschreibungstext auf die angegebene Zeichenfolge.
    cLTextOut1.setLabel("Value:");
    // Setzt den Dimensionstext auf die angegebene Zeichenfolge.
    cLTextOut1.setUnit("");
    // Einfügen der Komponente ins Applet.
    add(cLTextOut1, cLTextOut1.getName());
    // Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
    // ausgeführt werden, muß sich ein Objekt bei der entsprechenden
    // Ereignisquelle registrieren.
    // Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
    // wobei »XXX« für den entsprechenden Ereignis-Typ steht.
    // Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
    // Interface:
    s7CP1.addPropertyChangeListener(this);
    s7Device1.addPropertyChangeListener(this);
    cLTimer1.addActionListener(this);
    s7Variable1.addPropertyChangeListener(this);
}
/**
 * Wird nach der Initialisierung eines Applets ausgeführt.
 * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
 * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
 *
 * @see #init
 * @see #stop
 * @see #destroy
 */
public void start() {
    super.start();
}
/**
 * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
```

```
     * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
     * einem Browser verlassen wird.
     *
     * @see #init
     * @see #start
     * @see #destroy
     */
    public void stop() {
        super.stop();
    }
    /**
     * Wird immer aufgerufen, wenn das Applet zerstört wird.
     *
     * @see #init
     * @see #start
     * @see #stop
     */
    public void destroy() {
        super.destroy();
        // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
        // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
        S7Api.terminate();
    }
    /**
     * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
     *
     * @param evt PropertyChangeEvent
     */
    public void propertyChange(PropertyChangeEvent evt) {
        // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
        if (evt.getSource() == s7CP1)
            // Wenn JA
            // Ereignis weiterleiten an die S7Device-Instanz
            s7Device1.propertyChange(evt);
        // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
        if (evt.getSource() == s7Device1)
            // Wenn JA
            // Ereignis weiterleiten an die S7Variable-Instanz
            s7Variable1.propertyChange(evt);
        // Abfragen ob Ereignis vom S7Variable ausgelöst wurde.
        if (evt.getSource() == s7Variable1)
            // Wenn JA
            // Dann Ausgabewert an die CLTextOut Instanz übergeben.
            cLTextOut1.propertyChange(evt);
    }
    /**
     * Methode, um Ereignisse für die ActionListener Schnittstelle zu behandeln.
     *
     * @param e java.awt.event.ActionEvent
     */
    public void actionPerformed(ActionEvent e) {
        // Abfragen ob CLTimer ausgelöst hat.
        if (e.getSource() == cLTimer1) {
            // Wenn JA, dann Werte aus dem AG lesen.
            // Der Anstoss zum Lesen erfolgt mittels der Methode processGet()
            // vom S7Variable Bean.
            // Sind die neuen Werte vorhanden, so löst das S7Variable Bean ein
            // PropertyChangeEvent aus.
            s7Variable1.processGet();
        }
    }
}
```

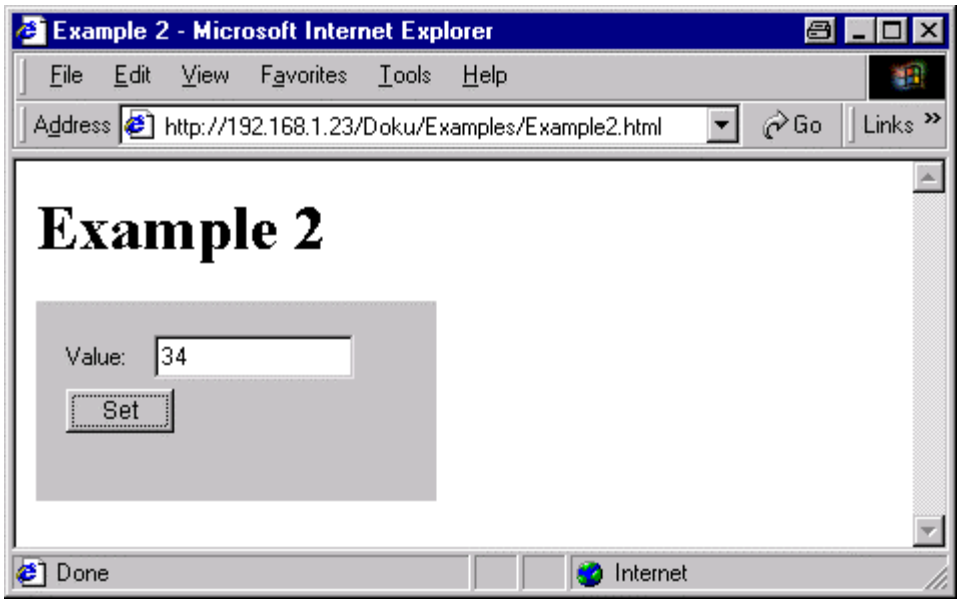
6.1.2 Beispiel 2 – Eine Variable mit S7Beans in die SIMATIC S7 schreiben

Funktionsweise

Es soll eine Variable in die SIMATIC S7 geschrieben werden.

Geschrieben wird hierbei in das MW 10 als INT

Darstellung



Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
import de.siemens.simaticnet.itcp.gui.*;
/**
 * Example2.java
 * <p>Überschrift: Beispiel 2 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Eingabe eines Wertes über CLTextIn Bean mit einer S7Variable.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Eingabe eines Wertes über CLTextIn Bean mit einer S7Variable.
 * Geschrieben wird das Merkerwort MW10.
 *
```

```
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * CLTextIn
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example2 extends Applet implements PropertyChangeListener {
   /*-----------------------------------Implementierte Interfaces*/
   /*----------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private CLTextIn cLTextIn1 = null;
   private S7CP s7CP1 = null;
   private S7Device s7Device1 = null;
   private S7Variable s7Variable1 = null;
   /**
    * Wird immer aufgerufen, wenn das Applet initialisiert wird.
    * Dies geschieht sofort, nachdem es geladen ist.
    *
    * @see #start
    * @see #stop
    * @see #destroy
    */
   public void init() {
      super.init();
      // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
      setName("Example2");
      // Setzt den Layoutmanager für diese Komponente.
      setLayout(null);
      // Setzt die Größe des Applet Breite / Höhe
      setSize(426, 240);
      /*-----------Höhe*/
      /*------Breite*/
      // Anlegen einer Instanz für das S7CP Bean.
      // S7CP ist der Ethernet Zugangspunkt in die Station
      s7CP1 = new S7CP();
      // Zuweisen der IP-Adresse
      // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
      s7CP1.setHostString(new HostString ("192.168.1.1:80"));
      /*-------------------------------------------------Angabe der Portnummer
                                             normalerweise :80*/
      /*-----------------------------------IP-Adresse als String*/
      // Anlegen einer Instanz für das S7Device Bean.
      // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
      s7Device1 = new S7Device();
      // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
      // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
      // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
      // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
      // eingetragen werden.
      // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
      // ######## notwendig                                          ########
      s7Device1.setSlot(2);
      /*----------------Steckplatznummer 2 (int)*/
      // Anlegen einer Instanz für das S7Variable Bean.
      // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
      // werden soll.
      s7Variable1 = new S7Variable();
      // Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
      s7Variable1.setS7Anypointer(
         new S7Anypointer((int)5, (int)1, (int)131, (int)0, (int)10, (int)0));
      /*-------------------------------------------------------------Bit Nummer 0 ..7*/
```

```
    /*-------------------------------------------------Speicherbereichoffset*/
    /*-----------------------------------------DB Nummer oder '0'*/
    /*-----------------------------------Speicherbereich 131 == M*/
    /*--------------------------Wiederholfaktor 1 .. n*/
    /*------------------Datentyp 5 == INT*/
    // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
    s7Variable1.setVariableName("s7Variable1");
    // Anlegen einer Instanz für das CLTextIn Bean.
    // Über das CLTextIn Bean können elementare Variablen vom Anwender eingegeben
    // werden.
    cLTextIn1 = new CLTextIn();
    // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
    cLTextIn1.setName("CLTextIn1");
    // Festlegen der Startposition und Größe der Komponente.
    cLTextIn1.setBounds(0, 0, 200, 45);
    /*----------------------------Komponentenhöhe*/
    /*------------------------Komponentenbreite*/
    /*--------------------Startposition Y*/
    /*------------------Startposition X*/
    // Festlegen der Größe des Eingabefeldes.
    cLTextIn1.setInFieldSize(100);
    /*----------------------Länge Eingabefeld*/
    // Setzt den Beschreibungstext auf die angegebene Zeichenfolge.
    cLTextIn1.setLabel("Value:");
    // Setzt den Dimensionstext auf die angegebene Zeichenfolge.
    cLTextIn1.setUnit("");
    // Einfügen der Komponente ins Applet.
    add(cLTextIn1, cLTextIn1.getName());
    // Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
    // ausgeführt werden, muß sich ein Objekt bei der entsprechenden
    // Ereignisquelle registrieren.
    // Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
    // wobei »XXX« für den entsprechenden Ereignis-Typ steht.
    // Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
    // Interface:
    s7CP1.addPropertyChangeListener(this);
    s7Device1.addPropertyChangeListener(this);
    s7Variable1.addPropertyChangeListener(this);
    cLTextIn1.addPropertyChangeListener(this);
}
/**
 * Wird nach der Initialisierung eines Applets ausgeführt.
 * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
 * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
 *
 * @see #init
 * @see #stop
 * @see #destroy
 */
public void start() {
    super.start();
}
/**
 * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
 * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
 * einem Browser verlassen wird.
 *
 * @see #init
 * @see #start
 * @see #destroy
 */
public void stop() {
    super.stop();
}
/**
```

```
     * Wird immer aufgerufen, wenn das Applet zerstört wird.
     *
     * @see #init
     * @see #start
     * @see #stop
     */
    public void destroy() {
        super.destroy();
        // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
        // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
        S7Api.terminate();
    }
    /**
     * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
     *
     * @param evt PropertyChangeEvent
     */
    public void propertyChange(PropertyChangeEvent evt) {
        // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
        if (evt.getSource() == s7CP1)
            // Wenn JA
            // Ereignis weiterleiten an die S7Device-Instanz
            s7Device1.propertyChange(evt);
        // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
        if (evt.getSource() == s7Device1)
            // Wenn JA
            // Ereignis weiterleiten an die S7Variable-Instanz
            s7Variable1.propertyChange(evt);
        // Abfragen ob Ereignis vom CLTextIn ausgelöst wurde.
        if (evt.getSource() == cLTextIn1)
            // Wenn JA, dann Werte ins AG schreiben.
            // Dann Eingabewert an das S7Variable Instanz übergeben.
            s7Variable1.propertyChange(evt);
    }
}
```

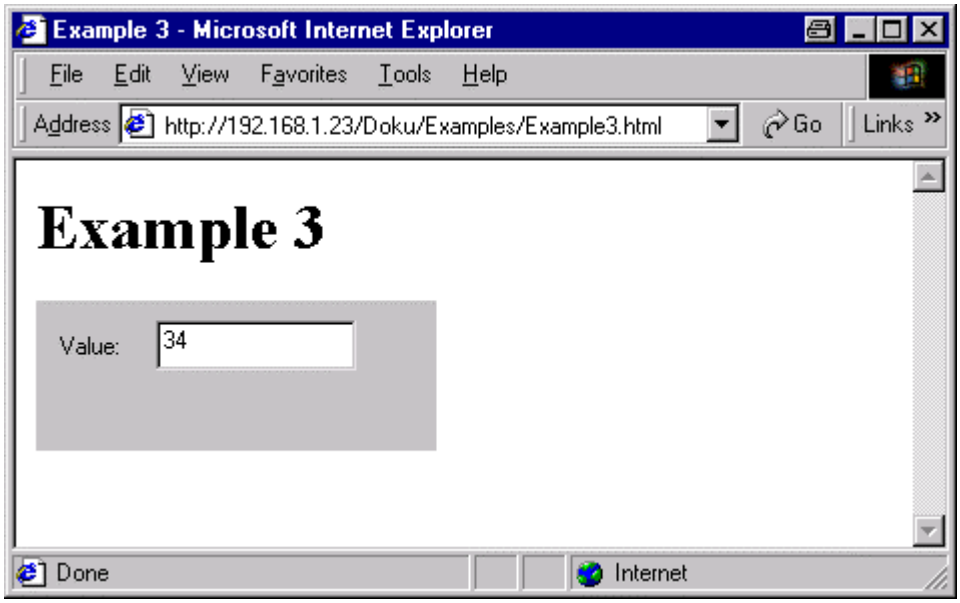
6.1.3 Beispiel 3 – Eine Variable aus der SIMATIC S7 mit AWT–Komponenten lesen und anzeigen

Funktionsweise

Es soll eine Variable aus der SIMATIC S7 mit AWT–Komponenten (Abstract Windowing Toolkit) gelesen und angezeigt werden.

Hierbei wird MW 10 als INT gelesen.

Darstellung



Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
/**
 * Example3.java
 * <p>Überschrift: Beispiel 3 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Ausgabe eines Wertes über AWT-Komponenten mit einer S7Variable.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Ausgabe eines Wertes über AWT-Komponenten mit einer S7Variable.
```

```
 * Gelesen wird das Merkerwort MW10.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * CLTimer
 * AWT Label zur Beschreibung des Eingabefeldes
 * AWT TextField für den Werte
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example3 extends Applet implements PropertyChangeListener, ActionListener {
   /*-----------------------------------Implementierte Interfaces*/
   /*---------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private CLTimer cLTimer1 = null;
   private S7CP s7CP1 = null;
   private S7Device s7Device1 = null;
   private S7Variable s7Variable1 = null;
   private TextField textField1 = null;
   private Label label1 = null;
   /**
    * Wird immer aufgerufen, wenn das Applet initialisiert wird.
    * Dies geschieht sofort, nachdem es geladen ist.
    *
    * @see #start
    * @see #stop
    * @see #destroy
    */
   public void init() {
      super.init();
      // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
      setName("Example3");
      // Setzt den Layoutmanager für diese Komponente.
      setLayout(null);
      // Setzt die Größe des Applet Breite / Höhe
      setSize(426, 240);
      /*-----------Höhe*/
      /*------Breite*/
      // Anlegen einer Instanz für das S7CP Bean.
      // S7CP ist der Ethernet Zugangspunkt in die Station
      s7CP1 = new S7CP();
      // Zuweisen der IP-Adresse
      // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
      s7CP1.setHostString(new HostString ("192.168.1.1:80"));
      /*-------------------------------------------------Angabe der Portnummer
                                             normalerweise :80*/
      /*-----------------------------------IP-Adresse als String*/
      // Anlegen einer Instanz für das S7Device Bean.
      // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
      s7Device1 = new S7Device();
      // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
      // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
      // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
      // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
      // eingetragen werden.
      // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
      // ######## notwendig                                          ########
      s7Device1.setSlot(2);
      /*----------------Steckplatznummer 2 (int)*/
      // Anlegen einer Instanz für das S7Variable Bean.
      // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
```

```
    // werden soll.
    s7Variable1 = new S7Variable();
    // Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
    s7Variable1.setS7Anypointer(
       new S7Anypointer((int)5, (int)1, (int)131, (int)0, (int)10, (int)0));
    /*-------------------------------------------------------------Bit Nummer 0 ..7*/
    /*-----------------------------------------------------Speicherbereichoffset*/
    /*-------------------------------------------DB Nummer oder '0'*/
    /*-----------------------------------Speicherbereich 131 == M*/
    /*--------------------------Wiederholfaktor 1 .. n*/
    /*-------------------Datentyp 5 == INT*/
    // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
    s7Variable1.setVariableName("s7Variable1");
    // Anlegen einer Instanz für das CLTimer Bean.
    // Das CLTimer Bean löst nach Ablauf der Zeit ein PropertyChangeEvent
    // Ereignis aus. Über dieses Ereignis wird eine zyklische Datenaktualisierung
    // realisiert.
    cLTimer1 = new CLTimer();
    // Mit der Methode setDelay() wird das Zeitintervall festgelegt.
    cLTimer1.setDelay(2000);
    /*----------------Zeitintervall in msec. 2000 == 2 sec.*/
    // Anlegen einer Instanz für ein Label.
    label1 = new Label();
    // Setzt den Ausgabetext auf die angegebene Zeichenfolge.
    label1.setText("Value:");
    // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
    label1.setName("Label1");
    // Festlegen der Startposition und Größe der Komponente.
    label1.setBounds(10, 10, 50, 25);
    /*--------------------------Komponentenhöhe*/
    /*-----------------------Komponentenbreite*/
    /*-------------------Startposition Y*/
    /*---------------Startposition X*/
    // Einfügen der Komponente ins Applet.
    add(label1, label1.getName());
    // Anlegen einer Instanz für ein TextField.
    textField1 = new TextField();
    // Setzt den Ausgabetext auf die angegebene Zeichenfolge.
    textField1.setText("");
    // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
    textField1.setName("TextField1");
    // Festlegen der Startposition und Größe der Komponente.
    textField1.setBounds(60, 10, 100, 25);
    /*-------------------------------Komponentenhöhe*/
    /*--------------------------Komponentenbreite*/
    /*----------------------Startposition Y*/
    /*------------------Startposition X*/
    // Einfügen der Komponente ins Applet.
    add(textField1, textField1.getName());
    // Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
    // ausgeführt werden, muß sich ein Objekt bei der entsprechenden
    // Ereignisquelle registrieren.
    // Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
    // wobei »XXX« für den entsprechenden Ereignis-Typ steht.
    // Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
    // Interface:
    s7CP1.addPropertyChangeListener(this);
    s7Device1.addPropertyChangeListener(this);
    cLTimer1.addActionListener(this);
    s7Variable1.addPropertyChangeListener(this);
}
/**
 * Wird nach der Initialisierung eines Applets ausgeführt.
 * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
 * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
```

```
 *
 * @see #init
 * @see #stop
 * @see #destroy
 */
public void start() {
   super.start();
}
/**
 * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
 * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
 * einem Browser verlassen wird.
 *
 * @see #init
 * @see #start
 * @see #destroy
 */
public void stop() {
   super.stop();
}
/**
 * Wird immer aufgerufen, wenn das Applet zerstört wird.
 *
 * @see #init
 * @see #start
 * @see #stop
 */
public void destroy() {
   super.destroy();
   // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
   // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
   S7Api.terminate();
}
/**
 * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
 *
 * @param evt PropertyChangeEvent
 */
public void propertyChange(PropertyChangeEvent evt) {
   // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
   if (evt.getSource() == s7CP1)
      // Wenn JA
      // Ereignis weiterleiten an die S7Device-Instanz
      s7Device1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
   if (evt.getSource() == s7Device1)
      // Wenn JA
      // Ereignis weiterleiten an die S7Variable-Instanz
      s7Variable1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Variable ausgelöst wurde.
   if (evt.getSource() == s7Variable1)
      // Wenn JA, dann sind im Ereignis die neuen Daten aus der Station enthalten.
      // Die neuen Daten werden als Integer von der Methode getNewValue()
      // bereitgestellt.
      // Ausgabe der Werte im Applet
      textField1.setText(evt.getNewValue().toString());
   /*-----------------------------------Integer-Wert in 'String' umwandeln */
   /*---------------------Neuwert lesen*/
   /*---------Gelesenen Wert im Applet anzeigen*/
}
/**
 * Methode, um Ereignisse für die ActionListener Schnittstelle zu behandeln.
 *
 * @param e java.awt.event.ActionEvent
 */
```

```
   public void actionPerformed(ActionEvent e) {
      // Abfragen ob CLTimer ausgelöst hat.
      if (e.getSource() == cLTimer1) {
         // Wenn JA, dann Werte aus dem AG lesen.
         // Der Anstoss zum Lesen erfolgt mittels der Methode processGet()
         // vom S7Variable Bean.
         // Sind die neuen Werte vorhanden, so löst das S7Variable Bean ein
         // PropertyChangeEvent aus.
         s7Variable1.processGet();
      }
   }
}
```

6.1.4 Beispiel 4 – Eine Variable in die SIMATIC S7 mittels AWT–Komponenten schreiben

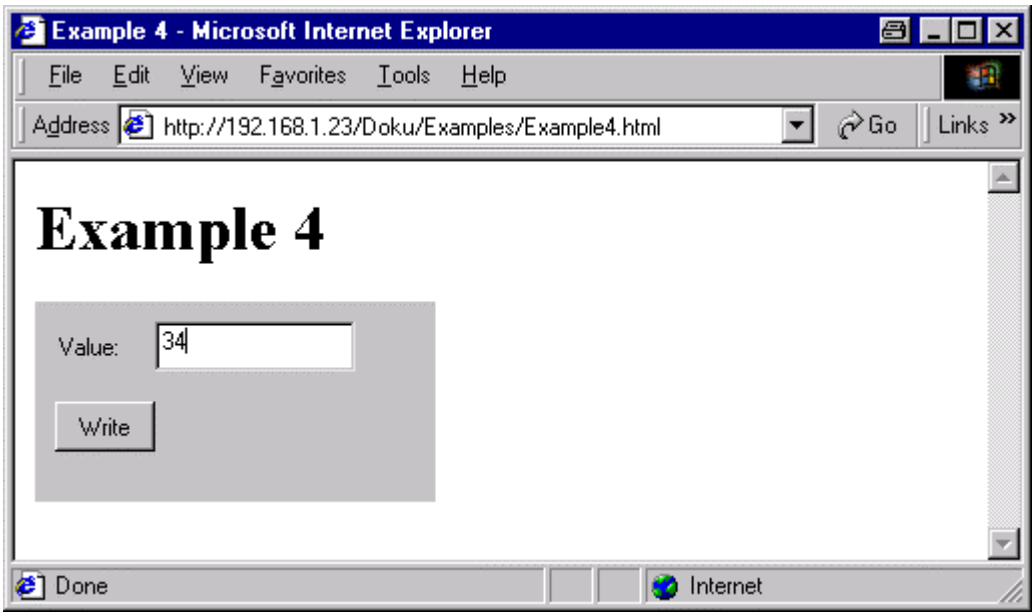
Funktionsweise

Es soll eine Variable in die SIMATIC S7 mittels AWT Komponenten (Abstract Windowing Toolkit) geschrieben werden.

Geschrieben wird hierbei MW 10 als INT.

Der Schreibvorgang wird durch Betätigen des Button ausgelöst.

Darstellung



Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
/**
 * Example4.java
 * <p>Überschrift: Beispiel 4 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Eingabe eines Wertes über AWT-Komponenten mit einer S7Variable.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
```

```
 *
 * Eingabe eines Wertes über AWT-Komponenten mit einer S7Variable.
 * Geschrieben wird das Merkerwort MW10.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * AWT Label zur Beschreibung des Eingabefeldes
 * AWT TextField für den Werte
 * AWT Button zum Schreiben
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example4 extends Applet implements PropertyChangeListener, ActionListener {
   /*-----------------------------------Implementierte Interfaces*/
   /*----------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private S7CP s7CP1 = null;
   private S7Device s7Device1 = null;
   private S7Variable s7Variable1 = null;
   private TextField textField1 = null;
   private Label label1 = null;
   private Button button1 = null;
   /**
    * Wird immer aufgerufen, wenn das Applet initialisiert wird.
    * Dies geschieht sofort, nachdem es geladen ist.
    *
    * @see #start
    * @see #stop
    * @see #destroy
    */
   public void init() {
      super.init();
      // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
      setName("Example4");
      // Setzt den Layoutmanager für diese Komponente.
      setLayout(null);
      // Setzt die Größe des Applet Breite / Höhe
      setSize(426, 240);
      /*-----------Höhe*/
      /*------Breite*/
      // Anlegen einer Instanz für das S7CP Bean.
      // S7CP ist der Ethernet Zugangspunkt in die Station
      s7CP1 = new S7CP();
      // Zuweisen der IP-Adresse
      // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
      s7CP1.setHostString(new HostString ("192.168.1.1:80"));
      /*-------------------------------------------------Angabe der Portnummer
                                              normalerweise :80*/
      /*-----------------------------------IP-Adresse als String*/
      // Anlegen einer Instanz für das S7Device Bean.
      // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
      s7Device1 = new S7Device();
      // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
      // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
      // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
      // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
      // eingetragen werden.
      // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
      // ######## notwendig                                          ########
      s7Device1.setSlot(2);
      /*----------------Steckplatznummer 2 (int)*/
```

```
// Anlegen einer Instanz für das S7Variable Bean.
// Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
// werden soll.
s7Variable1 = new S7Variable();
// Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
s7Variable1.setS7Anypointer(
   new S7Anypointer((int)5, (int)1, (int)131, (int)0, (int)10, (int)0));
/*-------------------------------------------------------------Bit Nummer 0 ..7*/
/*-----------------------------------------------------Speicherbereichoffset*/
/*-------------------------------------------DB Nummer oder '0'*/
/*-----------------------------------Speicherbereich 131 == M*/
/*---------------------------Wiederholfaktor 1 .. n*/
/*-------------------Datentyp 5 == INT*/
// Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
s7Variable1.setVariableName("s7Variable1");
// Anlegen einer Instanz für ein Label.
label1 = new Label();
// Setzt den Ausgabetext auf die angegebene Zeichenfolge.
label1.setText("Value:");
// Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
label1.setName("Label1");
// Festlegen der Startposition und Größe der Komponente.
label1.setBounds(10, 10, 50, 25);
/*---------------------------Komponentenhöhe*/
/*-----------------------Komponentenbreite*/
/*-------------------Startposition Y*/
/*---------------Startposition X*/
// Einfügen der Komponente ins Applet.
add(label1, label1.getName());
// Anlegen einer Instanz für ein TextField.
textField1 = new TextField();
// Setzt den Ausgabetext auf die angegebene Zeichenfolge.
textField1.setText("");
// Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
textField1.setName("TextField1");
// Festlegen der Startposition und Größe der Komponente.
textField1.setBounds(60, 10, 100, 25);
/*-------------------------------Komponentenhöhe*/
/*---------------------------Komponentenbreite*/
/*-----------------------Startposition Y*/
/*-----------------Startposition X*/
// Einfügen der Komponente ins Applet.
add(textField1, textField1.getName());
// Anlegen einer Instanz für einen Button.
button1 = new Button();
// Setzt den Ausgabetext auf die angegebene Zeichenfolge.
button1.setLabel("Write");
// Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
button1.setName("Button1");
// Festlegen der Startposition und Größe der Komponente.
button1.setBounds(10, 50, 50, 25);
/*---------------------------Komponentenhöhe*/
/*-----------------------Komponentenbreite*/
/*-------------------Startposition Y*/
/*---------------Startposition X*/
// Einfügen der Komponente ins Applet.
add(button1, button1.getName());
// Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
// ausgeführt werden, muß sich ein Objekt bei der entsprechenden
// Ereignisquelle registrieren.
// Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
// wobei »XXX« für den entsprechenden Ereignis-Typ steht.
// Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
// Interface:
s7CP1.addPropertyChangeListener(this);
```

```
      s7Device1.addPropertyChangeListener(this);
      s7Variable1.addPropertyChangeListener(this);
      button1.addActionListener(this);
      textField1.addActionListener(this);
   }
   /**
    * Wird nach der Initialisierung eines Applets ausgeführt.
    * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
    * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
    *
    * @see #init
    * @see #stop
    * @see #destroy
    */
   public void start() {
      super.start();
   }
   /**
    * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
    * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
    * einem Browser verlassen wird.
    *
    * @see #init
    * @see #start
    * @see #destroy
    */
   public void stop() {
      super.stop();
   }
   /**
    * Wird immer aufgerufen, wenn das Applet zerstört wird.
    *
    * @see #init
    * @see #start
    * @see #stop
    */
   public void destroy() {
      super.destroy();
      // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
      // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
      S7Api.terminate();
   }
   /**
    * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
    *
    * @param evt PropertyChangeEvent
    */
   public void propertyChange(PropertyChangeEvent evt) {
      // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
      if (evt.getSource() == s7CP1)
         // Wenn JA
         // Ereignis weiterleiten an die S7Device-Instanz
         s7Device1.propertyChange(evt);
      // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
      if (evt.getSource() == s7Device1)
         // Wenn JA
         // Ereignis weiterleiten an die S7Variable-Instanz
         s7Variable1.propertyChange(evt);
      // Abfragen ob Ereignis vom S7Variable ausgelöst wurde.
      if (evt.getSource() == s7Variable1) {
         // Wenn JA, dann sind im Ereignis die neuen Daten aus der Station enthalten.
         // Die neuen Daten werden Integer von der Methode getNewValue()
         // bereitgestellt.
         // Ausgabe der Werte im Applet
         textField1.setText(evt.getNewValue().toString());
```

```
            /*----------------------------------Integer-Wert in 'String' umwandeln */
            /*--------------------Neuwert lesen*/
            /*---------Gelesenen Wert im Applet anzeigen*/
         }
      }
      /**
       * Methode, um Ereignisse für die ActionListener Schnittstelle zu behandeln.
       *
       * @param e java.awt.event.ActionEvent
       */
      public void actionPerformed(ActionEvent e) {
         // Abfragen ob Button Schreiben ausgelöst wurde.
         if (e.getSource() == button1) {
            // Wenn JA, dann Werte ins AG schreiben.
            // Die neuen Daten werden als Integer an die Methode setValue() übergeben.
            // Die Methode setValue() schreibt dann den Wert in die Station.
            // Der Aufruf von setValue() löst automatisch die Methode processGet() zum
            // Lesen des Bereichs wieder aus.
            s7Variable1.setValue(textField1.getText());
            /*------------------------------Einagbewert aus TextField lesen*/
         }
         // Abfragen ob im TextField Return ausgelöst wurde.
         if (e.getSource() == textField1) {
            // Wenn JA, dann Werte ins AG schreiben.
            // Die neuen Daten werden als Integer an die Methode setValue() übergeben.
            // Die Methode setValue() schreibt dann den Wert in die Station.
            // Der Aufruf von setValue() löst automatisch die Methode processGet() zum
            // Lesen des Bereichs wieder aus.
            s7Variable1.setValue(textField1.getText());
            /*------------------------------Einagbewert aus TextField lesen*/
         }
      }
}
```

6.1.5 Beispiel 5 – Ein Button mit der Funktion “Taster”

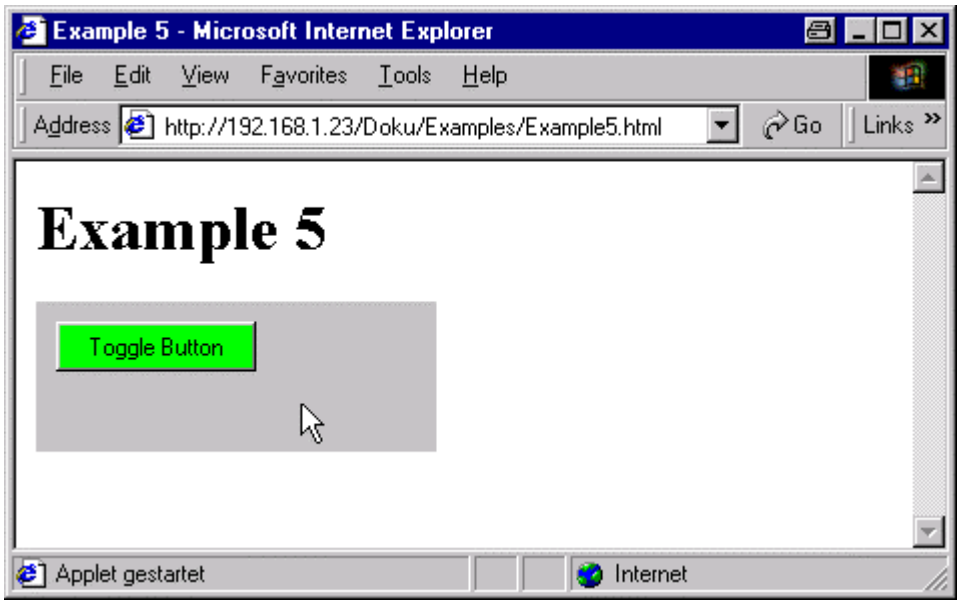
Funktionsweise

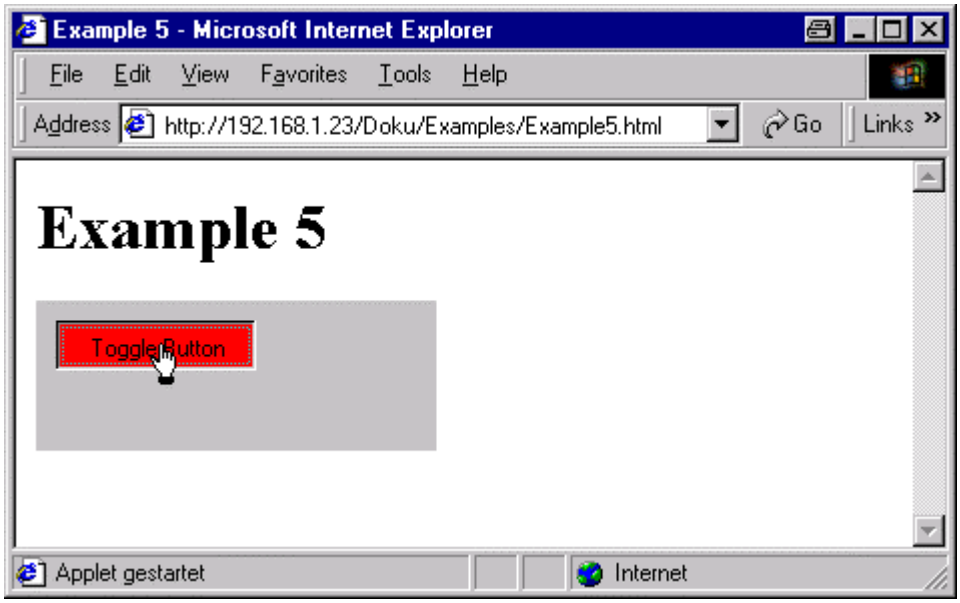
Es soll ein Button mit der Funktion “Taster” eingerichtet werden.

Geschrieben wird hierbei M 10.0 als BOOL.

Dem Merker wird bei Betätigung des Button der Wert “true” übergeben. Beim Loslassen des Button wird dem Merker der Wert “false” übergeben.

Darstellung





Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
/**
 * Example5.java
 * <p>Überschrift: Beispiel 5 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Ein Button mit der Funktion "Taster".</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Ein Button mit der Funktion "Taster".
 * Gelesen und Geschrieben wird das Merkerbit M10.0.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * AWT Button
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example5 extends Applet implements PropertyChangeListener, MouseListener {
   /*----------------------------------Implementierte Interfaces*/
   /*---------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private S7CP s7CP1 = null;
```

```
private S7Device s7Device1 = null;
private S7Variable s7Variable1 = null;
private Button button1 = null;
/**
 * Wird immer aufgerufen, wenn das Applet initialisiert wird.
 * Dies geschieht sofort, nachdem es geladen ist.
 *
 * @see #start
 * @see #stop
 * @see #destroy
 */
public void init() {
   super.init();
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   setName("Example5");
   // Setzt den Layoutmanager für diese Komponente.
   setLayout(null);
   // Setzt die Größe des Applet Breite / Höhe
   setSize(426, 240);
   /*-----------Höhe*/
   /*------Breite*/
   // Anlegen einer Instanz für das S7CP Bean.
   // S7CP ist der Ethernet Zugangspunkt in die Station
   s7CP1 = new S7CP();
   // Zuweisen der IP-Adresse
   // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
   s7CP1.setHostString(new HostString ("192.168.1.1:80"));
   /*-----------------------------------------------Angabe der Portnummer
                                          normalerweise :80*/
   /*----------------------------------IP-Adresse als String*/
   // Anlegen einer Instanz für das S7Device Bean.
   // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
   s7Device1 = new S7Device();
   // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
   // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
   // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
   // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
   // eingetragen werden.
   // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
   // ######## notwendig                                          ########
   s7Device1.setSlot(2);
   /*----------------Steckplatznummer 2 (int)*/
   // Anlegen einer Instanz für das S7Variable Bean.
   // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
   // werden soll.
   s7Variable1 = new S7Variable();
   // Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
   s7Variable1.setS7Anypointer(
      new S7Anypointer((int)1, (int)1, (int)131, (int)0, (int)10, (int)0));
   /*-------------------------------------------------------------Bit Nummer 0 ..7*/
   /*-----------------------------------------------Speicherbereichoffset*/
   /*-------------------------------------------DB Nummer oder '0'*/
   /*----------------------------------Speicherbereich 131 == M*/
   /*--------------------------Wiederholfaktor 1 .. n*/
   /*------------------Datentyp 1 == BOOL*/
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   s7Variable1.setVariableName("s7Variable1");
   // Anlegen einer Instanz für einen Button.
   button1 = new Button();
   // Setzt den Ausgabetext auf die angegebene Zeichenfolge.
   button1.setLabel("Toggle Button");
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   button1.setName("ToggleButton");
   // Festlegen der Startposition und Größe der Komponente.
   button1.setBounds(10, 10, 100, 25);
```

```
    /*----------------------------Komponentenhöhe*/
    /*-------------------------Komponentenbreite*/
    /*--------------------Startposition Y*/
    /*----------------Startposition X*/
    // Setzt die Hintergrundfarbe auf grün.
    button1.setBackground(Color.green);
    // Einfügen der Komponente ins Applet.
    add(button1, button1.getName());
    // Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
    // ausgeführt werden, muß sich ein Objekt bei der entsprechenden
    // Ereignisquelle registrieren.
    // Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
    // wobei »XXX« für den entsprechenden Ereignis-Typ steht.
    // Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
    // Interface:
    s7CP1.addPropertyChangeListener(this);
    s7Device1.addPropertyChangeListener(this);
    s7Variable1.addPropertyChangeListener(this);
    button1.addMouseListener(this);
}
/**
 * Wird nach der Initialisierung eines Applets ausgeführt.
 * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
 * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
 *
 * @see #init
 * @see #stop
 * @see #destroy
 */
public void start() {
    super.start();
}
/**
 * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
 * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
 * einem Browser verlassen wird.
 *
 * @see #init
 * @see #start
 * @see #destroy
 */
public void stop() {
    super.stop();
}
/**
 * Wird immer aufgerufen, wenn das Applet zerstört wird.
 *
 * @see #init
 * @see #start
 * @see #stop
 */
public void destroy() {
    super.destroy();
    // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
    // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
    S7Api.terminate();
}
/**
 * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
 *
 * @param evt PropertyChangeEvent
 */
public void propertyChange(PropertyChangeEvent evt) {
    // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
    if (evt.getSource() == s7CP1)
```

```
            // Wenn JA
            // Ereignis weiterleiten an die S7Device-Instanz
            s7Device1.propertyChange(evt);
        // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
        if (evt.getSource() == s7Device1)
            // Wenn JA
            // Ereignis weiterleiten an die S7Variable-Instanz
            s7Variable1.propertyChange(evt);
        // Abfragen ob Ereignis vom S7Variable ausgelöst wurde.
        if (evt.getSource() == s7Variable1) {
            // Es findet bei diesem Beispiel keine Auswertung der gelesenen Stationswerte
            // statt!
        }
    }
    /**
     * Methode, um Ereignisse für die MouseListener Schnittstelle zu behandeln.
     *
     * mousePressed wird beim Druck einer Maustaste aufgerufen.
     *
     * @param e java.awt.event.ActionEvent
     */
    public void mousePressed(MouseEvent e) {
        // Abfragen ob Button ausgelöst wurde.
        if (e.getSource() == button1) {
            // Wenn JA, dann neuen Wert (true) ins AG schreiben.
            // Die neuen Daten werden als String an die Methode setValue() übergeben.
            // Die Methode setValue() schreibt dann den Wert in die Station.
            // Der Aufruf von setValue() löst automatisch die Methode processGet() zum
            // Lesen des Bereichs wieder aus.
            s7Variable1.setValue(String.valueOf("true"));
            // Setzt die Hintergrundfarbe auf rot.
            button1.setBackground(Color.red);
            // Die Methode waitOnNewData läßt erst nach Ablauf der Wartezeit oder
            // bei Eintreffen neuer Daten einen neuen Aufruf von setValue() zu.
            // Damit werden schnelle Button-Klicks verhindert.
            s7Variable1.waitOnNewData(2000);
            /*------------------------Wartezeit in msec. 2000 == 2 sec.*/
        }
    }
    /**
     * Methode, um Ereignisse für die MouseListener Schnittstelle zu behandeln.
     *
     * mouseReleased wird beim Loslassen einer Maustaste aufgerufen.
     *
     * @param e java.awt.event.ActionEvent
     */
    public void mouseReleased(MouseEvent e) {
        // Abfragen ob Button ausgelöst wurde.
        if (e.getSource() == button1) {
            // Wenn JA, dann neuen Wert (false) ins AG schreiben.
            // Die neuen Daten werden als String an die Methode setValue() übergeben.
            // Die Methode setValue() schreibt dann den Wert in die Station.
            // Der Aufruf von setValue() löst automatisch die Methode processGet() zum
            // Lesen des Bereichs wieder aus.
            s7Variable1.setValue(String.valueOf("false"));
            // Setzt die Hintergrundfarbe auf grün.
            button1.setBackground(Color.green);
            // Die Methode waitOnNewData läßt erst nach Ablauf der Wartezeit oder
            // bei Eintreffen neuer Daten einen neuen Aufruf von setValue() zu.
            // Damit werden schnelle Button-Klicks verhindert.
            s7Variable1.waitOnNewData(2000);
            /*------------------------Wartezeit in msec. 2000 == 2 sec.*/
        }
    }
    // Das Interfaces MouseListener besitzt mehr als eine Methode.
```

```
    // Wenn man nur eine dieser Methoden für die Ereignisverarbeitung benötigt,
    // so müssen die nicht benötigte Methoden als Dummy-Implementierungen bereitgestellt
    // werden.
    public void mouseClicked(MouseEvent e) {
    }
    public void mouseEntered(MouseEvent e) {
    }
    public void mouseExited(MouseEvent e) {
    }
}
```

6.1.6 Beispiel 6 – Ein Button mit der Funktion “Schalter”

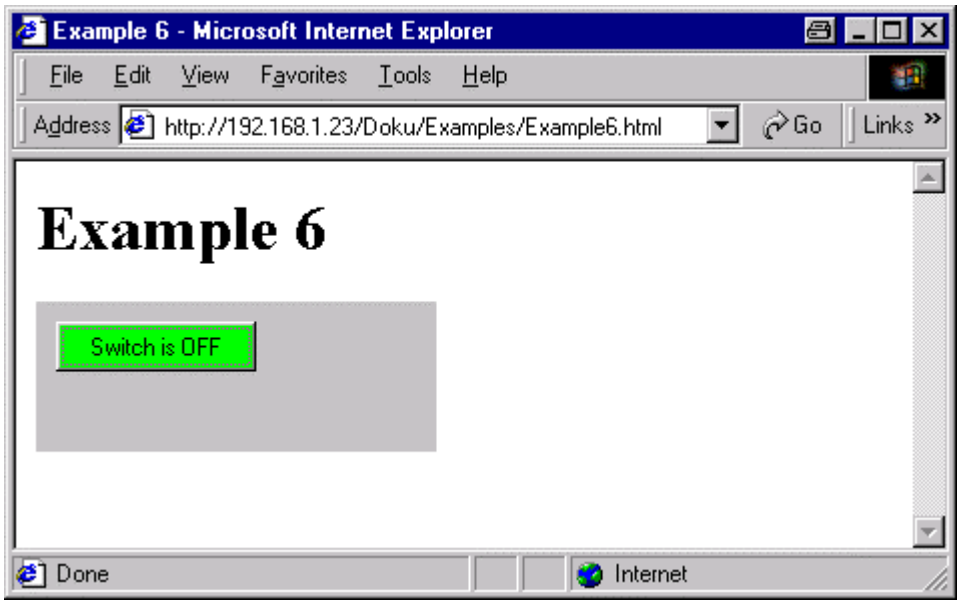
Funktionsweise

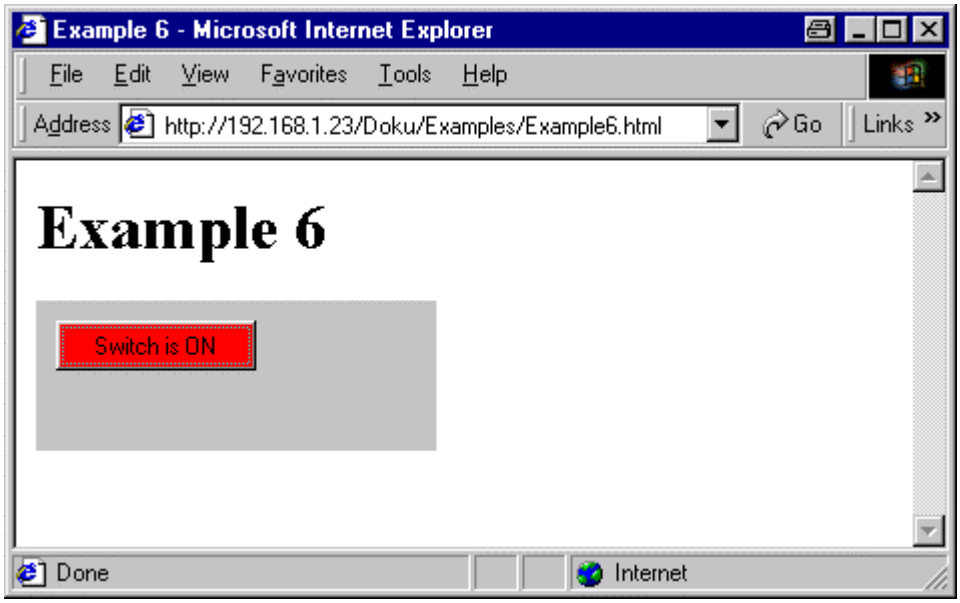
Es soll ein Button mit der Funktion “Schalter” eingerichtet werden.

Geschrieben wird hierbei M 10.0 als BOOL.

Beim Starten des Applets wird der Zustand des Merkers gelesen. Wird der Button betätigt, wird der Zustand invertiert und in den Merker geschrieben.

Darstellung





Quelltext:

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
/**
 * Example6.java
 * <p>Überschrift: Beispiel 6 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Ein Button mit der Funktion "Schalter".</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Ein Button mit der Funktion "Schalter".
 * Gelesen und Geschrieben wird das Merkerbit M10.0.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * AWT Button
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example6 extends Applet implements PropertyChangeListener, MouseListener {
   /*----------------------------------Implementierte Interfaces*/
   /*---------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private S7CP s7CP1 = null;
```

```
private S7Device s7Device1 = null;
private S7Variable s7Variable1 = null;
private Button button1 = null;
/**
 * Wird immer aufgerufen, wenn das Applet initialisiert wird.
 * Dies geschieht sofort, nachdem es geladen ist.
 *
 * @see #start
 * @see #stop
 * @see #destroy
 */
public void init() {
   super.init();
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   setName("Example6");
   // Setzt den Layoutmanager für diese Komponente.
   setLayout(null);
   // Setzt die Größe des Applet Breite / Höhe
   setSize(426, 240);
   /*-----------Höhe*/
   /*------Breite*/
   // Anlegen einer Instanz für das S7CP Bean.
   // S7CP ist der Ethernet Zugangspunkt in die Station
   s7CP1 = new S7CP();
   // Zuweisen der IP-Adresse
   // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
   s7CP1.setHostString(new HostString ("192.168.1.1:80"));
   /*-----------------------------------------------Angabe der Portnummer
                                          normalerweise :80*/
   /*----------------------------------IP-Adresse als String*/
   // Anlegen einer Instanz für das S7Device Bean.
   // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
   s7Device1 = new S7Device();
   // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
   // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
   // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
   // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
   // eingetragen werden.
   // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
   // ######## notwendig                                          ########
   s7Device1.setSlot(2);
   /*----------------Steckplatznummer 2 (int)*/
   // Anlegen einer Instanz für das S7Variable Bean.
   // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
   // werden soll.
   s7Variable1 = new S7Variable();
   // Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
   s7Variable1.setS7Anypointer(
      new S7Anypointer((int)1, (int)1, (int)131, (int)0, (int)10, (int)0));
   /*-------------------------------------------------------------Bit Nummer 0 ..7*/
   /*-----------------------------------------------Speicherbereichoffset*/
   /*-------------------------------------------DB Nummer oder '0'*/
   /*----------------------------------Speicherbereich 131 == M*/
   /*--------------------------Wiederholfaktor 1 .. n*/
   /*------------------Datentyp 1 == BOOL*/
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   s7Variable1.setVariableName("s7Variable1");
   // Anlegen einer Instanz für einen Button.
   button1 = new Button();
   // Setzt den Ausgabetext auf die angegebene Zeichenfolge.
   button1.setLabel("Switch");
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   button1.setName("Switch");
   // Festlegen der Startposition und Größe der Komponente.
   button1.setBounds(10, 10, 100, 25);
```

```
    /*----------------------------Komponentenhöhe*/
    /*-------------------------Komponentenbreite*/
    /*--------------------Startposition Y*/
    /*----------------Startposition X*/
    // Einfügen der Komponente ins Applet.
    add(button1, button1.getName());
    // Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
    // ausgeführt werden, muß sich ein Objekt bei der entsprechenden
    // Ereignisquelle registrieren.
    // Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
    // wobei »XXX« für den entsprechenden Ereignis-Typ steht.
    // Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
    // Interface:
    s7CP1.addPropertyChangeListener(this);
    s7Device1.addPropertyChangeListener(this);
    s7Variable1.addPropertyChangeListener(this);
    button1.addMouseListener(this);
}
/**
 * Wird nach der Initialisierung eines Applets ausgeführt.
 * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
 * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
 *
 * @see #init
 * @see #stop
 * @see #destroy
 */
public void start() {
    super.start();
    // Lesen der definierten S7Variable um den aktuellen Status anzuzeigen.
    // Der Anstoss zum Lesen erfolgt mittels der Methode processGet()
    // vom S7Variable Bean.
    // Sind die neuen Werte vorhanden, so löst das S7Variable Bean ein
    // PropertyChangeEvent aus.
    s7Variable1.processGet();
}
/**
 * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
 * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
 * einem Browser verlassen wird.
 *
 * @see #init
 * @see #start
 * @see #destroy
 */
public void stop() {
    super.stop();
}
/**
 * Wird immer aufgerufen, wenn das Applet zerstört wird.
 *
 * @see #init
 * @see #start
 * @see #stop
 */
public void destroy() {
    super.destroy();
    // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
    // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
    S7Api.terminate();
}
/**
 * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
 *
 * @param evt PropertyChangeEvent
```

```
 */
public void propertyChange(PropertyChangeEvent evt) {
   // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
   if (evt.getSource() == s7CP1)
      // Wenn JA
      // Ereignis weiterleiten an die S7Device-Instanz
      s7Device1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
   if (evt.getSource() == s7Device1)
      // Wenn JA
      // Ereignis weiterleiten an die S7Variable-Instanz
      s7Variable1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Variable ausgelöst wurde.
   if (evt.getSource() == s7Variable1) {
      // Wenn JA, dann sind im Ereignis die neuen Daten aus der Station enthalten.
      // Die neuen Daten werden als Boolean von der Methode getNewValue()
      // bereitgestellt.
      // Abfragen ob der Signalzustand true oder false (VKE1 / VKE0) des gelesenen
      // Werts ist.
      if (((Boolean)evt.getNewValue()).booleanValue()) {
         /*------------------------------Boolean-Wert in 'boolean' umwandeln */
         /*---------------Neuwert lesen*/
         /*---Konvertiert den Neuwert nach Boolean*/
         // Wenn true (VKE1), dann Text und Farbe des Button setzen.
         // Setzt den Ausgabetext auf die angegebene Zeichenfolge.
         button1.setLabel("Switch is ON");
         // Setzt die Hintergrundfarbe auf rot.
         button1.setBackground(Color.red);
      } else {
         // Wenn false (VKE0), dann Text und Farbe des Button setzen.
         // Setzt den Ausgabetext auf die angegebene Zeichenfolge.
         button1.setLabel("Switch is OFF");
         // Setzt die Hintergrundfarbe auf grün.
         button1.setBackground(Color.green);
      }
   }
}
/**
 * Methode, um Ereignisse für die MouseListener Schnittstelle zu behandeln.
 *
 * mousePressed wird beim Druck einer Maustaste aufgerufen.
 *
 * @param e java.awt.event.ActionEvent
 */
public void mousePressed(MouseEvent e) {
   // Abfragen ob Button ausgelöst wurde.
   if (e.getSource() == button1) {
      // Wenn JA, dann neuen Wert ins AG schreiben.
      // Abfragen des letzten Zustands des Stationswertes und invertiert
      // zurückschreiben.
      if (((Boolean)s7Variable1.getValue()).booleanValue()) {
         /*-----------------------------------Boolean-Wert in 'boolean' umwandeln*/
         /*-----------------------letzten Zustand lesen*/
         /*---Konvertiert den letzten Zustand nach Boolean*/
         // Wenn letzter Zustuand true (VKE1), dann neuer Wert false (VKE0).
         // Die neuen Daten werden als String an die Methode setValue() übergeben.
         // Die Methode setValue() schreibt dann den Wert in die Station.
         // Der Aufruf von setValue() löst automatisch die Methode processGet() zum
         // Lesen des Bereichs wieder aus.
         s7Variable1.setValue(String.valueOf(false));
         // Die Methode waitOnNewData läßt erst nach Ablauf der Wartezeit oder
         // bei Eintreffen neuer Daten einen neuen Aufruf von setValue() zu.
         // Damit werden schnelle Button-Klicks verhindert.
         s7Variable1.waitOnNewData(2000);
         /*-----------------------Wartezeit in msec. 2000 == 2 sec.*/
```

```
            } else {
               // Wenn letzter Zustand false (VKE0), dann neuer Wert true (VKE1).
               s7Variable1.setValue(String.valueOf(true));
               s7Variable1.waitOnNewData(2000);
            }
         }
      }
      // Das Interfaces MouseListener besitzt mehr als eine Methode.
      // Wenn man nur eine dieser Methoden für die Ereignisverarbeitung benötigt,
      // so müssen die nicht benötigte Methoden als Dummy-Implementierungen bereitgestellt
      // werden.
      public void mouseReleased(MouseEvent e) {
      }
      public void mouseClicked(MouseEvent e) {
      }
      public void mouseEntered(MouseEvent e) {
      }
      public void mouseExited(MouseEvent e) {
      }
}
```

6.1.7 Beispiel 7 – Ausgabe von mehreren Werten über eine Variable

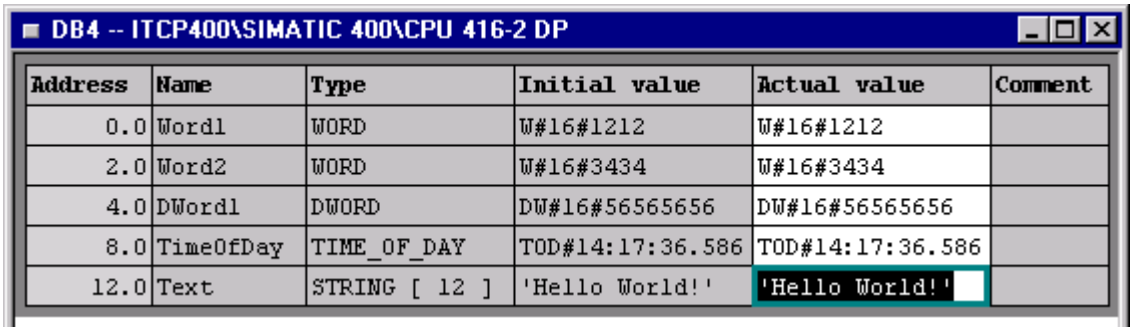
Funktionsweise

Es sollen mehrere Werte über eine Variable ausgegeben werden.

Gelesen wird hierbei aus dem DB4 ab Byte 0 eine Anzahl von 32 Bytes.

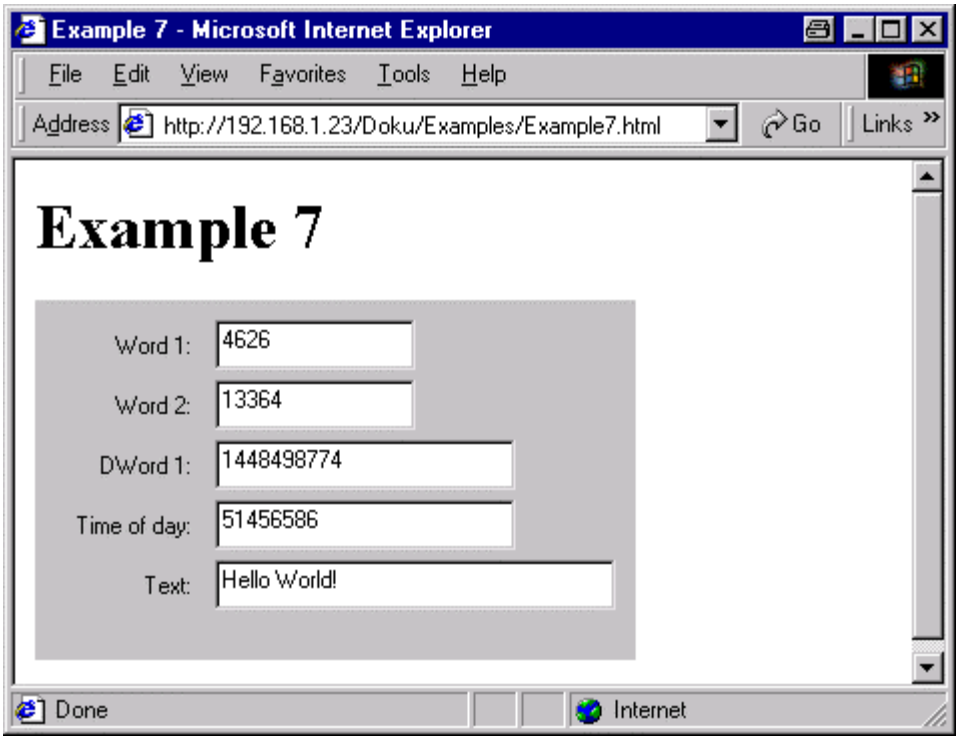
Die Ausgabe erfolgt zyklisch.

Struktur des DB



Address	Name	Type	Initial value	Actual value	Comment
0.0	Word1	WORD	W#16#1212	W#16#1212	
2.0	Word2	WORD	W#16#3434	W#16#3434	
4.0	DWord1	DWORD	DW#16#56565656	DW#16#56565656	
8.0	TimeOfDay	TIME_OF_DAY	TOD#14:17:36.586	TOD#14:17:36.586	
12.0	Text	STRING [12]	'Hello World!'	'Hello World!'	

Darstellung



Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
/**
 * Example7.java
 * <p>Überschrift: Beispiel 7 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Ausgabe von mehreren Werten über AWT-Komponenten mit einer
 * S7Variable.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Ausgabe von mehreren Werten über eine Variable.
 * Gelesen wird aus dem DB4 ab Byte 0 eine Anzahl von 32 Bytes.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * CLTimer
 * AWT Label zur Beschreibung der Ausgabefelder
 * AWT TextField für jeden der 5 Werte
 *
 * Aufbau des Datenbausteins DB4 als AWL-Quelle für STEP7
 * DATA_BLOCK DB 4
 * STRUCT
 * Word1 : WORD  := W#16#1212;
 * Word2 : WORD  := W#16#3434;
 * DWord1 : DWORD  := DW#16#56565656;
 * TimeOfDay : TIME_OF_DAY  := TOD#14:17:36.586;
 * Text : STRING  [12 ] := 'Hello World!';
 * END_STRUCT ;
 * BEGIN
 * END_DATA_BLOCK
 *
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example7 extends Applet implements PropertyChangeListener, ActionListener {
   /*-----------------------------------Implementierte Interfaces*/
   /*----------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private CLTimer cLTimer1 = null;
   private S7CP s7CP1 = null;
   private S7Device s7Device1 = null;
   private S7Variable s7Variable1 = null;
   private TextField tFWord1 = null;
   private Label lWord1 = null;
   private TextField tFWord2 = null;
   private Label lWord2 = null;
   private TextField tFDWord1 = null;
   private Label lDWord1 = null;
   private TextField tFTimeOfDay1 = null;
```

```
private Label lTimeOfDay1 = null;
private TextField tFText1 = null;
private Label lText1 = null;
/**
 * Wird immer aufgerufen, wenn das Applet initialisiert wird.
 * Dies geschieht sofort, nachdem es geladen ist.
 *
 * @see #start
 * @see #stop
 * @see #destroy
 */
public void init() {
   super.init();
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   setName("Example7");
   // Setzt den Layoutmanager für diese Komponente.
   setLayout(null);
   // Setzt die Größe des Applet Breite / Höhe
   setSize(426, 240);
   /*-----------Höhe*/
   /*------Breite*/
   // Anlegen einer Instanz für das S7CP Bean.
   // S7CP ist der Ethernet Zugangspunkt in die Station
   s7CP1 = new S7CP();
   // Zuweisen der IP-Adresse
   // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
   s7CP1.setHostString(new HostString ("192.168.1.1:80"));
   /*-----------------------------------------------Angabe der Portnummer
                                          normalerweise :80*/
   /*----------------------------------IP-Adresse als String*/
   // Anlegen einer Instanz für das S7Device Bean.
   // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
   s7Device1 = new S7Device();
   // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
   // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
   // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
   // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
   // eingetragen werden.
   // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
   // ######## notwendig                                          ########
   s7Device1.setSlot(2);
   /*----------------Steckplatznummer 2 (int)*/
   // Anlegen einer Instanz für das S7Variable Bean.
   // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
   // werden soll.
   s7Variable1 = new S7Variable();
   // Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
   s7Variable1.setS7Anypointer(
      new S7Anypointer((int)2, (int)26, (int)132, (int)4, (int)0, (int)0));
   /*-----------------------------------------------------------Bit Nummer 0 ..7*/
   /*-------------------------------------------------Speicherbereichoffset*/
   /*---------------------------------------------DB Nummer oder '0'*/
   /*----------------------------------Speicherbereich 132 == DB*/
   /*--------------------------Wiederholfaktor 1 .. n*/
   /*------------------Datentyp 2 == Byte*/
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   s7Variable1.setVariableName("s7Variable1");
   // Anlegen einer Instanz für das CLTimer Bean.
   // Das CLTimer Bean löst nach Ablauf der Zeit ein PropertyChangeEvent Ereignis
   // aus. Über dieses Ereignis wird eine zyklische Datenaktualisierung realisiert.
   cLTimer1 = new CLTimer();
   // Mit der Methode setDelay() wird das Zeitintervall festgelegt.
   cLTimer1.setDelay(2000);
   /*----------------Zeitintervall in msec. 2000 == 2 sec.*/
   // Anlegen einer Instanz für ein Label.
```

```
lWord1 = new Label();
// Setzt den Ausgabetext auf die angegebene Zeichenfolge.
lWord1.setText("Word 1:");
// Festlegen der Ausrichtung für den Anzeigetext.
lWord1.setAlignment(Label.RIGHT);
// Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
lWord1.setName("Text1");
// Festlegen der Startposition und Größe der Komponente.
lWord1.setBounds(10, 10, 70, 25);
/*---------------------------Komponentenhöhe*/
/*-------------------------Komponentenbreite*/
/*--------------------Startposition Y*/
/*---------------Startposition X*/
// Einfügen der Komponente ins Applet.
add(lWord1, lWord1.getName());
// Anlegen einer Instanz für ein TextField.
tFWord1 = new TextField();
// Setzt den Ausgabetext auf die angegebene Zeichenfolge.
tFWord1.setText("");
// Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
tFWord1.setName("TextField1");
// Festlegen der Startposition und Größe der Komponente.
tFWord1.setBounds(90, 10, 100, 25);
/*----------------------------Komponentenhöhe*/
/*------------------------Komponentenbreite*/
/*--------------------Startposition Y*/
/*----------------Startposition X*/
// Einfügen der Komponente ins Applet.
add(tFWord1, tFWord1.getName());
lWord2 = new Label();
lWord2.setText("Word 2:");
lWord2.setAlignment(Label.RIGHT);
lWord2.setName("Text2");
lWord2.setBounds(10, 40, 70, 25);
add(lWord2, lWord2.getName());
tFWord2 = new TextField();
tFWord2.setText("");
tFWord2.setName("TextField2");
tFWord2.setBounds(90, 40, 100, 25);
add(tFWord2, tFWord2.getName());
lDWord1 = new Label();
lDWord1.setText("DWord 1:");
lDWord1.setAlignment(Label.RIGHT);
lDWord1.setName("Text3");
lDWord1.setBounds(10, 70, 70, 25);
add(lDWord1, lDWord1.getName());
tFDWord1 = new TextField();
tFDWord1.setText("");
tFDWord1.setName("TextField3");
tFDWord1.setBounds(90, 70, 150, 25);
add(tFDWord1, tFDWord1.getName());
lTimeOfDay1 = new Label();
lTimeOfDay1.setText("Time of day:");
lTimeOfDay1.setAlignment(Label.RIGHT);
lTimeOfDay1.setName("Text4");
lTimeOfDay1.setBounds(10, 100, 70, 25);
add(lTimeOfDay1, lTimeOfDay1.getName());
tFTimeOfDay1 = new TextField();
tFTimeOfDay1.setText("");
tFTimeOfDay1.setName("TextField4");
tFTimeOfDay1.setBounds(90, 100, 150, 25);
add(tFTimeOfDay1, tFTimeOfDay1.getName());
lText1 = new Label();
lText1.setText("Text:");
lText1.setAlignment(Label.RIGHT);
```

```
    lText1.setName("Text5");
    lText1.setBounds(10, 130, 70, 25);
    add(lText1, lText1.getName());
    tFText1 = new TextField();
    tFText1.setText("");
    tFText1.setName("TextField5");
    tFText1.setBounds(90, 130, 200, 25);
    add(tFText1, tFText1.getName());
    // Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
    // ausgeführt werden, muß sich ein Objekt bei der entsprechenden
    // Ereignisquelle registrieren.
    // Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
    // wobei »XXX« für den entsprechenden Ereignis-Typ steht.
    // Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
    // Interface:
    s7CP1.addPropertyChangeListener(this);
    s7Device1.addPropertyChangeListener(this);
    cLTimer1.addActionListener(this);
    s7Variable1.addPropertyChangeListener(this);
}
/**
 * Wird nach der Initialisierung eines Applets ausgeführt.
 * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
 * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
 *
 * @see #init
 * @see #stop
 * @see #destroy
 */
public void start() {
    super.start();
}
/**
 * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
 * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
 * einem Browser verlassen wird.
 *
 * @see #init
 * @see #start
 * @see #destroy
 */
public void stop() {
    super.stop();
}
/**
 * Wird immer aufgerufen, wenn das Applet zerstört wird.
 *
 * @see #init
 * @see #start
 * @see #stop
 */
public void destroy() {
    super.destroy();
    // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
    // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
    S7Api.terminate();
}
/**
 * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
 *
 * @param evt PropertyChangeEvent
 */
public void propertyChange(PropertyChangeEvent evt) {
    // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
    if (evt.getSource() == s7CP1)
```

```
      // Wenn JA
      // Ereignis weiterleiten an die S7Device-Instanz
      s7Device1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
   if (evt.getSource() == s7Device1)
      // Wenn JA
      // Ereignis weiterleiten an die S7Variable-Instanz
      s7Variable1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Variable ausgelöst wurde.
   if (evt.getSource() == s7Variable1) {
      // Wenn JA, dann sind im Ereignis die neuen Daten aus der Station enthalten.
      // Die neuen Daten werden in einem int[] (IntegerArray) von der Methode
      // getNewValue() bereitgestellt.
      // Umladen der Daten in eine lokale Hilfsvariable (myArray)
      int[] myArray = (int[])evt.getNewValue();
      // Deklaration von lokalen Hilfsvariablen zum Umladen der einzelnen
      // Ausgabewerten.
      int word1, word2;
      long dword1, tod;
      char[] text = new char[myArray[13]];
      /*-----------------------------Länge des aktuellen Strings*/
      // Bitte Beschreibung im Hanbuch zum Aufbau eines S7 Strings beachten.
      // Da im Beispiel der Datentyp Byte gewählt wurde, kommen die Stationswerte
      // als einzelne Bytes zurück.
      // Zusammensetzen des ersten Ausgabewortes.
      word1 = (myArray[0]<<8) + (myArray[1]);
      /*---------------------------------zweites Byte laden*/
      /*-------------------Linksverschiebung um 8 Stellen*/
      /*---------------erstes Byte laden*/
      // Zusammensetzen des zweiten Ausgabewortes.
      word2 = (myArray[2]<<8) + (myArray[3]);
      /*---------------------------------zweites Byte laden*/
      /*-------------------Linksverschiebung um 8 Stellen*/
      /*---------------erstes Byte laden*/
      // Zusammensetzen des ersten Ausgabedoppelwortes.
      dword1 = (myArray[4]<<24) + (myArray[5]<<16) +
      /*---------------------------------------Linksverschiebung um 16 Stellen*/
      /*---------------------------------zweites Byte laden*/
      /*-------------------Linksverschiebung um 24 Stellen*/
      /*----------------erstes Byte laden*/
         (myArray[6]<<8) + (myArray[7]);
      /*-----------------------------viertes Byte laden*/
      /*---------------Linksverschiebung um 8 Stellen*/
      /*-----------drittes Byte laden*/
      // Zusammensetzen des zweiten Ausgabedoppelwortes.
      tod = (myArray[8]<<24) + (myArray[9]<<16) +
      /*---------------------------------Linksverschiebung um 16 Stellen*/
      /*---------------------------------zweites Byte laden*/
      /*-----------------Linksverschiebung um 24 Stellen*/
      /*-------------erstes Byte laden*/
         (myArray[10]<<8) + (myArray[11]);
      /*-------------------------------viertes Byte laden*/
      /*---------------Linksverschiebung um 8 Stellen*/
      /*-----------drittes Byte laden*/
      // Laden des AG Textstrings.
      // Es wird jedes Zeichen einzeln aus dem Stations-Bytestrom umgewandelt.
      for (int i = 0; i < myArray[13]; i++) {
         /*-------------------------Anzahl der aktuellen Zeichen*/
         // Umladen des einzelnen Zeichens
         text[i] = (char)myArray[14 + i];
         /*-------------------------n'te Zeichen*/
         /*---------------------Startpunkt im Bytestrom für erstes Zeichen*/
         /*---------wandelt AG-Wert nach char*/
      }
      // Ausgabe der Werte im Applet
```

```
            tFWord1.setText("" + word1);
            tFWord2.setText("" + word2);
            tFDWord1.setText("" + dword1);
            tFTimeOfDay1.setText("" + tod);
            tFText1.setText(new String(text));
         }
      }
      /**
       * Methode, um Ereignisse für die ActionListener Schnittstelle zu behandeln.
       *
       * @param e java.awt.event.ActionEvent
       */
      public void actionPerformed(ActionEvent e) {
         // Abfragen ob CLTimer ausgelöst hat.
         if (e.getSource() == cLTimer1) {
            // Wenn JA, dann Werte aus dem AG lesen.
            // Der Anstoss zum Lesen erfolgt mittels der Methode processGet()
            // vom S7Variable Bean.
            // Sind die neuen Werte vorhanden, so löst das S7Variable Bean ein
            // PropertyChangeEvent aus.
            s7Variable1.processGet();
         }
      }
}
```

6.1.8 Beispiel 8 – Eingabe von mehreren Werten über eine Variable

Funktionsweise

Es soll die Ausgabe und Eingabe von mehreren Werten über eine Variable erfolgen.

Gelesen und Geschrieben wird in dem DB4 ab Byte 0 eine Anzahl von 32 Bytes.

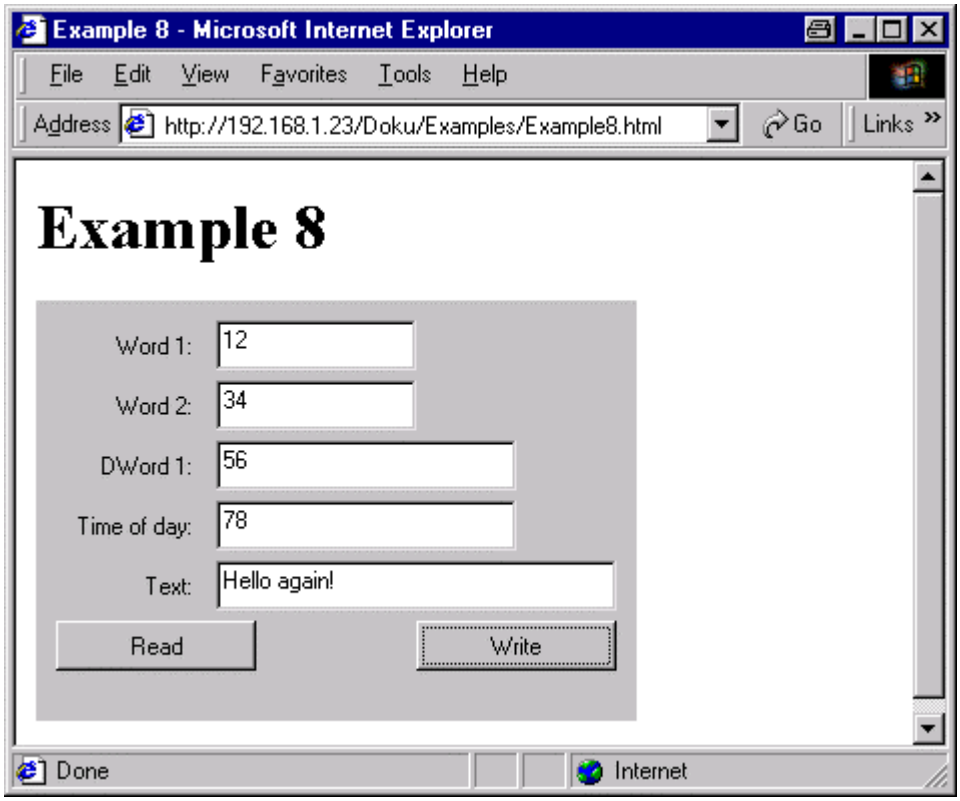
Das Lesen und Schreiben erfolgt über die Betätigung eines Buttons.

Struktur des DB

DB4 -- ITCP400\SIMATIC 400\CPU 416-2 DP

Address	Name	Type	Initial value	Actual value	Comment
0.0	Word1	WORD	W#16#1212	W#16#1212	
2.0	Word2	WORD	W#16#3434	W#16#3434	
4.0	DWord1	DWORD	DW#16#56565656	DW#16#56565656	
8.0	TimeOfDay	TIME_OF_DAY	TOD#14:17:36.586	TOD#14:17:36.586	
12.0	Text	STRING [12]	'Hello World!'	'Hello World!'	

Darstellung



Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
/**
 * Example8.java
 * <p>Überschrift: Beispiel 8 für die Benutzung der ITCP Beans.</p>
 * <p>Beschreibung: Ausgabe und Eingabe von mehreren Werten über AWT-Komponenten mit einer
 * S7Variable.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Ausgabe und Eingabe von mehreren Werten über eine Variable.
 * Gelesen und Geschrieben wird in den DB4 ab Byte 0 eine Anzahl von 32 Bytes.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * AWT Label zur Beschreibung der Ausgabefelder
 * AWT TextField für jeden der 5 Werte
 * AWT Button zum Lesen und Schreiben
 *
 * Aufbau des Datenbausteins DB4 als AWL-Quelle für STEP7
 * DATA_BLOCK DB 4
 * STRUCT
 * Word1 : WORD  := W#16#1212;
 * Word2 : WORD  := W#16#3434;
 * DWord1 : DWORD  := DW#16#56565656;
 * TimeOfDay : TIME_OF_DAY  := TOD#14:17:36.586;
 * Text : STRING  [12 ] := 'Hello World!';
 * END_STRUCT ;
 * BEGIN
 * END_DATA_BLOCK
 *
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example8 extends Applet implements PropertyChangeListener, ActionListener {
   /*-----------------------------------Implementierte Interfaces*/
   /*----------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private S7CP s7CP1 = null;
   private S7Device s7Device1 = null;
   private S7Variable s7Variable1 = null;
   private TextField tFWord1 = null;
   private Label lWord1 = null;
   private TextField tFWord2 = null;
   private Label lWord2 = null;
   private TextField tFDWord1 = null;
   private Label lDWord1 = null;
   private TextField tFTimeOfDay1 = null;
   private Label lTimeOfDay1 = null;
   private TextField tFText1 = null;
```

```
private Label lText1 = null;
private Button button1 = null;
private Button button2 = null;
/**
 * Wird immer aufgerufen, wenn das Applet initialisiert wird.
 * Dies geschieht sofort, nachdem es geladen ist.
 *
 * @see #start
 * @see #stop
 * @see #destroy
 */
public void init() {
   super.init();
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   setName("Example8");
   // Setzt den Layoutmanager für diese Komponente.
   setLayout(null);
   // Setzt die Größe des Applet Breite / Höhe
   setSize(426, 240);
   /*-----------Höhe*/
   /*------Breite*/
   // Anlegen einer Instanz für das S7CP Bean.
   // S7CP ist der Ethernet Zugangspunkt in die Station
   s7CP1 = new S7CP();
   // Zuweisen der IP-Adresse
   // ######## Projektspezifische Anpassung der IP-Adresse notwendig          ########
   s7CP1.setHostString(new HostString ("192.168.1.1:80"));
   /*-----------------------------------------------Angabe der Portnummer
                                          normalerweise :80*/
   /*----------------------------------IP-Adresse als String*/
   // Anlegen einer Instanz für das S7Device Bean.
   // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
   s7Device1 = new S7Device();
   // Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
   // Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
   // Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
   // Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
   // eingetragen werden.
   // ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
   // ######## notwendig                                          ########
   s7Device1.setSlot(2);
   /*----------------Steckplatznummer 2 (int)*/
   // Anlegen einer Instanz für das S7Variable Bean.
   // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
   // werden soll.
   s7Variable1 = new S7Variable();
   // Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
   s7Variable1.setS7Anypointer(
      new S7Anypointer((int)2, (int)26, (int)132, (int)4, (int)0, (int)0));
   /*-------------------------------------------------------------Bit Nummer 0 ..7*/
   /*-----------------------------------------------------Speicherbereichoffset*/
   /*-------------------------------------------DB Nummer oder '0'*/
   /*----------------------------------Speicherbereich 132 == DB*/
   /*--------------------------Wiederholfaktor 1 .. n*/
   /*------------------Datentyp 2 == Byte*/
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   s7Variable1.setVariableName("s7Variable1");
   // Anlegen einer Instanz für ein Label.
   lWord1 = new Label();
   // Setzt den Ausgabetext auf die angegebene Zeichenfolge.
   lWord1.setText("Word 1:");
   // Festlegen der Ausrichtung für den Anzeigetext.
   lWord1.setAlignment(Label.RIGHT);
   // Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
   lWord1.setName("Text1");
```

```
// Festlegen der Startposition und Größe der Komponente.
lWord1.setBounds(10, 10, 70, 25);
/*---------------------------Komponentenhöhe*/
/*-----------------------Komponentenbreite*/
/*-------------------Startposition Y*/
/*---------------Startposition X*/
// Einfügen der Komponente ins Applet.
add(lWord1, lWord1.getName());
// Anlegen einer Instanz für ein TextField.
tFWord1 = new TextField();
// Setzt den Ausgabetext auf die angegebene Zeichenfolge.
tFWord1.setText("");
// Setzt den Namen der Komponente auf die angegebene Zeichenfolge.
tFWord1.setName("TextField1");
// Festlegen der Startposition und Größe der Komponente.
tFWord1.setBounds(90, 10, 100, 25);
/*----------------------------Komponentenhöhe*/
/*-------------------------Komponentenbreite*/
/*--------------------Startposition Y*/
/*----------------Startposition X*/
// Einfügen der Komponente ins Applet.
add(tFWord1, tFWord1.getName());
lWord2 = new Label();
lWord2.setText("Word 2:");
lWord2.setAlignment(Label.RIGHT);
lWord2.setName("Text2");
lWord2.setBounds(10, 40, 70, 25);
add(lWord2, lWord2.getName());
tFWord2 = new TextField();
tFWord2.setText("");
tFWord2.setName("TextField2");
tFWord2.setBounds(90, 40, 100, 25);
add(tFWord2, tFWord2.getName());
lDWord1 = new Label();
lDWord1.setText("DWord 1:");
lDWord1.setAlignment(Label.RIGHT);
lDWord1.setName("Text3");
lDWord1.setBounds(10, 70, 70, 25);
add(lDWord1, lDWord1.getName());
tFDWord1 = new TextField();
tFDWord1.setText("");
tFDWord1.setName("TextField3");
tFDWord1.setBounds(90, 70, 150, 25);
add(tFDWord1, tFDWord1.getName());
lTimeOfDay1 = new Label();
lTimeOfDay1.setText("Time of day:");
lTimeOfDay1.setAlignment(Label.RIGHT);
lTimeOfDay1.setName("Text4");
lTimeOfDay1.setBounds(10, 100, 70, 25);
add(lTimeOfDay1, lTimeOfDay1.getName());
tFTimeOfDay1 = new TextField();
tFTimeOfDay1.setText("");
tFTimeOfDay1.setName("TextField4");
tFTimeOfDay1.setBounds(90, 100, 150, 25);
add(tFTimeOfDay1, tFTimeOfDay1.getName());
lText1 = new Label();
lText1.setText("Text:");
lText1.setAlignment(Label.RIGHT);
lText1.setName("Text5");
lText1.setBounds(10, 130, 70, 25);
add(lText1, lText1.getName());
tFText1 = new TextField();
tFText1.setText("");
tFText1.setName("TextField5");
tFText1.setBounds(90, 130, 200, 25);
```

```
      add(tFText1, tFText1.getName());
      button1 = new Button();
      button1.setLabel("Read");
      button1.setName("Button1");
      button1.setBounds(10, 160, 100, 25);
      add(button1, button1.getName());
      button2 = new Button();
      button2.setLabel("Write");
      button2.setName("Button2");
      button2.setBounds(190, 160, 100, 25);
      add(button2, button2.getName());
      // Neben der Definition der Methoden, die beim Eintreten eines Ereignisses
      // ausgeführt werden, muß sich ein Objekt bei der entsprechenden
      // Ereignisquelle registrieren.
      // Dies geschieht durch Aufruf der addXXXListener-Methode der Ereignisquelle,
      // wobei »XXX« für den entsprechenden Ereignis-Typ steht.
      // Die addXXXListener-Methoden erwarten alle einen Verweis auf das zugehörige
      // Interface:
      s7CP1.addPropertyChangeListener(this);
      s7Device1.addPropertyChangeListener(this);
      s7Variable1.addPropertyChangeListener(this);
      button1.addActionListener(this);
      button2.addActionListener(this);
   }
   /**
    * Wird nach der Initialisierung eines Applets ausgeführt.
    * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
    * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
    *
    * @see #init
    * @see #stop
    * @see #destroy
    */
   public void start() {
      super.start();
   }
   /**
    * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
    * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
    * einem Browser verlassen wird.
    *
    * @see #init
    * @see #start
    * @see #destroy
    */
   public void stop() {
      super.stop();
   }
   /**
    * Wird immer aufgerufen, wenn das Applet zerstört wird.
    *
    * @see #init
    * @see #start
    * @see #stop
    */
   public void destroy() {
      super.destroy();
      // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
      // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
      S7Api.terminate();
   }
   /**
    * Methode, um Ereignisse für die PropertyChangeListener Schnittstelle zu behandeln.
    *
    * @param evt PropertyChangeEvent
```

```
 */
public void propertyChange(PropertyChangeEvent evt) {
   // Abfragen ob Ereignis vom S7CP ausgelöst wurde.
   if (evt.getSource() == s7CP1)
      // Wenn JA
      // Ereignis weiterleiten an die S7Device-Instanz
      s7Device1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Device ausgelöst wurde.
   if (evt.getSource() == s7Device1)
      // Wenn JA
      // Ereignis weiterleiten an die S7Variable-Instanz
      s7Variable1.propertyChange(evt);
   // Abfragen ob Ereignis vom S7Variable ausgelöst wurde.
   if (evt.getSource() == s7Variable1) {
      // Wenn JA, dann sind im Ereignis die neuen Daten aus der Station enthalten.
      // Die neuen Daten werden in einem int[] (IntegerArray) von der Methode
      // getNewValue() bereitgestellt.
      // Umladen der Daten in eine lokale Hilfsvariable (myArray)
      int[] myArray = (int[])evt.getNewValue();
      // Deklaration von lokalen Hilfsvariablen zum Umladen der einzelnen
      // Ausgabewerten.
      int word1, word2;
      long dword1, tod;
      char[] text = new char[myArray[13]];
      /*----------------------------Länge des aktuellen Strings*/
      // Bitte Beschreibung im Hanbuch zum Aufbau eines S7 Strings beachten.
      // Da im Beispiel der Datentyp Byte gewählt wurde, kommen die Stationswerte
      // als einzelne Bytes zurück.
      // Zusammensetzen des ersten Ausgabewortes.
      word1 = (myArray[0]<<8) + (myArray[1]);
      /*---------------------------------zweites Byte laden*/
      /*------------------Linksverschiebung um 8 Stellen*/
      /*---------------erstes Byte laden*/
      // Zusammensetzen des zweiten Ausgabewortes.
      word2 = (myArray[2]<<8) + (myArray[3]);
      /*---------------------------------zweites Byte laden*/
      /*------------------Linksverschiebung um 8 Stellen*/
      /*---------------erstes Byte laden*/
      // Zusammensetzen des ersten Ausgabedoppelwortes.
      dword1 = (myArray[4]<<24) + (myArray[5]<<16) +
      /*-------------------------------------Linksverschiebung um 16 Stellen*/
      /*-----------------------------------zweites Byte laden*/
      /*--------------------Linksverschiebung um 24 Stellen*/
      /*----------------erstes Byte laden*/
         (myArray[6]<<8) + (myArray[7]);
      /*----------------------------vierte Byte laden*/
      /*---------------Linksverschiebung um 8 Stellen*/
      /*-----------drittes Byte laden*/
      // Zusammensetzen des zweiten Ausgabedoppelwortes.
      tod = (myArray[8]<<24) + (myArray[9]<<16) +
      /*-----------------------------------Linksverschiebung um 16 Stellen*/
      /*---------------------------------zweites Byte laden*/
      /*-----------------Linksverschiebung um 24 Stellen*/
      /*-------------erstes Byte laden*/
         (myArray[10]<<8) + (myArray[11]);
      /*-----------------------------vierte Byte laden*/
      /*---------------Linksverschiebung um 8 Stellen*/
      /*-----------drittes Byte laden*/
      // Laden des AG Textstrings.
      // Es wird jedes Zeichen einzeln aus dem Stations-Bytestrom umgewandelt.
      for (int i = 0; i < myArray[13]; i++) {
         /*-------------------------Anzahl der aktuellen Zeichen*/
         // Umladen des einzelnen Zeichens
         text[i] = (char)myArray[14 + i];
         /*---------------------------n'te Zeichen*/
```

```
            /*----------------------Startpunkt im Bytestrom für erstes Zeichen*/
            /*---------wandelt AG-Wert nach char*/
         }
         // Ausgabe der Werte im Applet
         tFWord1.setText("" + word1);
         tFWord2.setText("" + word2);
         tFDWord1.setText("" + dword1);
         tFTimeOfDay1.setText("" + tod);
         tFText1.setText(new String(text));
      }
   }
   /**
    * Methode, um Ereignisse für die ActionListener Schnittstelle zu behandeln.
    *
    * @param e java.awt.event.ActionEvent
    */
   public void actionPerformed(ActionEvent e) {
      // Abfragen ob Button Lesen ausgelöst wurde.
      if (e.getSource() == button1) {
         // Wenn JA, dann Werte aus dem AG lesen.
         // Der Anstoss zum Lesen erfolgt mittels der Methode processGet()
         // vom S7Variable Bean.
         // Sind die neuen Werte vorhanden, so löst das S7Variable Bean ein
         // PropertyChangeEvent aus.
         s7Variable1.processGet();
      }
      // Abfragen ob Button Schreiben ausgelöst wurde.
      if (e.getSource() == button2) {
         // Wenn JA, dann Werte ins AG schreiben.
         // Deklaration von lokalen Hilfsvariablen zum Umladen der einzelnen
         // Schreibwerte.
         String text;
         int iTmp;
         long lTmp;
         int[] myArray = new int[32];
         /*----------------------Größe des Kommunikationsbereich im Beispiel 32 Byte*/
         // Erster Eingabewert in den Bytestrom schreiben (WORD).
         iTmp = Integer.valueOf(tFWord1.getText()).intValue();
         /*---------------------------------------Integer-Wert in 'int' umwandeln */
         /*----------------------------Eingabetextfeld lesen*/
         /*-------------Konvertiert den Inhalt von String in einen Integer-Wert */
         myArray[0] = (iTmp & 0xFF00) >>> 8;
         /*---------------------------Rechtsverschiebung um 8 Stellen und Auffüllen mit
                                 Nullen*/
         /*-------------------Maske für erstes Byte*/
         myArray[1] = iTmp & 0x00FF;
         /*-------------------Maske für zweites Byte*/
         // Zweiter Eingabewert in den Bytestrom schreiben (WORD).
         iTmp = Integer.valueOf(tFWord2.getText()).intValue();
         /*---------------------------------------Integer-Wert in 'int' umwandeln */
         /*----------------------------Eingabetextfeld lesen*/
         /*-------------Konvertiert den Inhalt von String in einen Integer-Wert */
         myArray[2] = (iTmp & 0xFF00) >>> 8;
         /*---------------------------Rechtsverschiebung um 8 Stellen und Auffüllen mit
                                 Nullen*/
         /*-------------------Maske für erstes Byte*/
         myArray[3] = iTmp & 0x00FF;
         /*-------------------Maske für zweites Byte*/
         // Dritter Eingabewert in den Bytestrom schreiben (DWORD).
         lTmp = Long.valueOf(tFDWord1.getText()).longValue();
         /*--------------------------------------Long-Wert in 'long' umwandeln */
         /*---------------------------Eingabetextfeld lesen*/
         /*----------Konvertiert den Inhalt von String in einen Long-Wert */
         myArray[4] = (int)((lTmp & 0xFF000000) >>> 24);
         /*-------------------------------------Rechtsverschiebung um 24 Stellen und
```

```
                                Auffüllen mit Nullen */
/*-------------------------Maske für erstes Byte*/
/*------------nach int wandeln*/
myArray[5] = (int)((lTmp & 0x00FF0000) >>> 16);
/*-------------------------------------Rechtsverschiebung um 16 Stellen und
                             Auffüllen mit Nullen */
/*-------------------------Maske für zweites Byte*/
/*------------nach int wandeln*/
myArray[6] = (int)((lTmp & 0x0000FF00) >>> 8);
/*-------------------------------------Rechtsverschiebung um 8 Stellen und
                                Auffüllen mit Nullen */
/*-------------------------Maske für drittes Byte*/
/*------------nach int wandeln*/
myArray[7] = (int)(lTmp & 0x000000FF);
/*-------------------------Maske für viertes Byte*/
/*------------nach int wandeln*/
// Vierter Eingabewert in den Bytestrom schreiben (TIME_OF_DAY).
lTmp = Long.valueOf(tFTimeOfDay1.getText()).longValue();
/*-----------------------------------------Long-Wert in 'long' umwandeln */
/*--------------------------------Eingabetextfeld lesen*/
/*----------Konvertiert den Inhalt von String in einen Long-Wert */
myArray[8] = (int)((lTmp & 0xFF000000) >>> 24);
/*-------------------------------------Rechtsverschiebung um 24 Stellen und
                                Auffüllen mit Nullen*/
/*-------------------------Maske für erstes Byte*/
/*------------nach int wandeln*/
myArray[9] = (int)((lTmp & 0x00FF0000) >>> 16);
/*-------------------------------------Rechtsverschiebung um 16 Stellen und
                                Auffüllen mit Nullen*/
/*-------------------------Maske für zweites Byte*/
/*------------nach int wandeln*/
myArray[10] = (int)((lTmp & 0x0000FF00) >>> 8);
/*-------------------------------------Rechtsverschiebung um 8 Stellen und
                                 Auffüllen mit Nullen*/
/*-------------------------Maske für drittes Byte*/
/*-------------nach int wandeln*/
myArray[11] = (int)(lTmp & 0x000000FF);
/*-------------------------Maske für viertes Byte*/
/*-------------nach int wandeln*/
// Vierter Eingabewert in den Bytestrom schreiben (STRING).
text = tFText1.getText();
/*-------------Eingabetextfeld lesen und umladen*/
myArray[12] = 18;
/*------------Maximallänge für den String vorbelegen*/
/*------Position der String max. Länge im Bytestrom*/
// Überprüfen der aktuellen Länge mit der max. Länge
if (text.length() > myArray[12]) {
   // Wenn aktuelle Länge größer als max. Länge
   // Dann aktuelle Länge gleich max. Länge
   myArray[13] = myArray[12];
   /*--------------------max. Länge*/
   /*------aktuelle Länge*/
} else {
   // Wenn aktuelle Länge kleiner als max. Länge
   // Dann aktuelle Länge gleich Eingabelänge
   myArray[13] = text.length();
   /*-----------------Eingabelänge*/
   /*------aktuelle Länge*/
}
// Die einzelnen Zeichen in den Bytestrom schreiben
for (int i=0;i<myArray[13];i++) {
   /*--------------------aktuelle Länge*/
   myArray[14+i] = text.charAt(i);
   /*-------------------------n'te Zeichen lesen*/
   /*---------n'te Position im Bytestrom*/
```

```
            /*------Startpunkt im Bytestrom für erstes Zeichen*/
         }
         // Die neuen Daten werden in einem int[] (IntegerArray) an die Methode
         // setValue() übergeben.
         // Die Methode setValue() schreibt dann den Bytestrom in die Station.
         // Der Aufruf von setValue() löst automatisch die Methode processGet() zum
         // Lesen des Bereichs wieder aus.
         s7Variable1.setValue(myArray);
         // Die Methode waitOnNewData läßt erst nach Ablauf der Wartezeit oder
         // bei Eintreffen neuer Daten einen neuen Aufruf von setValue() zu.
         // Damit werden schnelle Button-Klicks verhindert.
         s7Variable1.waitOnNewData(2000);
         /*------------------------Wartezeit in msec. 2000 == 2 sec.*/
      }
   }
}
```

6.2 Beispiel zur Arbeitsweise mit JBuilder

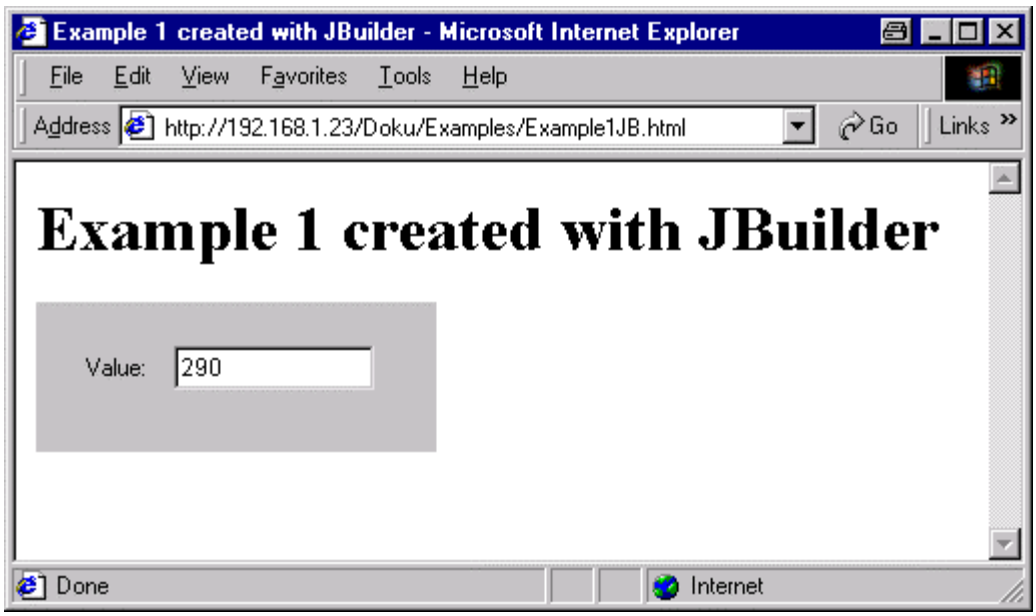
Funktionsweise

Es soll eine Variable aus der SIMATIC S7 gelesen und angezeigt werden.

Gelesen wird hierbei das MW 10 als INT.

Das Beispiel ist mit der Java–Entwicklungsumgebung JBuilder von Borland erstellt und entspricht funktionell dem zuvor in Kap. 6.1.1 gezeigten Beispiel 1. Lediglich der Quelltext ist anders strukturiert.

Darstellung



Quelltext

```
// Durch diese Anweisung werden alle Klassen, in deren Quelltext diese
// Anweisung steht, einem Paket zugeordnet.
package de.siemens.simaticnet.itcp.example;
// Durch den Import eines Pakets oder einer Klasse werden alle
// Vereinbarungen sichtbar gemacht, die von ihrer Zugriffsklasse her in
// anderen Paketen sichtbar sein dürfen.
import java.applet.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
import de.siemens.simaticnet.itcp.gui.*;
/**
 * Example1JB.java
 * <p>Überschrift: Beispiel 1 für die Benutzung der ITCP Beans mit der JBuilder IDE.</p>
 * <p>Beschreibung: Ausgabe eines Wertes über CLTextOut Bean mit einer S7Variable.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
```

```
 * Ausgabe eines Wertes über CLTextOut Bean mit einer S7Variable.
 * Gelesen wird das Merkerwort MW10.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * CLTimer
 * CLTextOut
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class Example1JB extends Applet {
   /*-----------------------------Basis-Klasse Applet*/
   // Deklaration der benötigten Komponenten
   private S7CP s7CP1;
   private S7Device s7Device1;
   private S7Variable s7Variable1;
   private CLTimer cLTimer1;
   private CLTextOut cLTextOut1;
   // Das Applet konstruieren
   public Example1JB() {
   }
   /**
    * Wird immer aufgerufen, wenn das Applet initialisiert wird.
    * Dies geschieht sofort, nachdem es geladen ist.
    *
    * @see #start
    * @see #stop
    * @see #destroy
    */
   public void init() {
      try {
         // Initialisierung der Komponenten
         jbInit();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   // Initialisierung der Komponenten
   private void jbInit() throws Exception {
      // Anlegen einer Instanz für das S7CP Bean.
      // S7CP ist der Ethernet Zugangspunkt in die Station
      s7CP1 = (S7CP) Beans.instantiate(getClass().getClassLoader(),
                                 S7CP.class.getName());
      // Anlegen einer Instanz für das S7Device Bean.
      // Mit S7Device wird der Kommunikationspartner in der Station adressiert.
      s7Device1 = (S7Device) Beans.instantiate(getClass().getClassLoader(),
                                       S7Device.class.getName());
      // Anlegen einer Instanz für das S7Variable Bean.
      // Das S7Variable Bean repräsentiert die Variable die gelesen oder geschrieben
      // werden soll.
      s7Variable1 = (S7Variable) Beans.instantiate(getClass().getClassLoader(),
                                          S7Variable.class.getName());
      // Anlegen einer Instanz für das CLTimer Bean.
      // Das CLTimer Bean löst nach Ablauf der Zeit ein PropertyChangeEvent Ereignis
      // aus. Über dieses Ereignis wird eine zyklische Datenaktualisierung realisiert.
      cLTimer1 = (CLTimer) Beans.instantiate(getClass().getClassLoader(),
                                       CLTimer.class.getName());
      // Anlegen einer Instanz für das CLTextOut Bean.
      // Über das CLTextOut Bean können elementare Variablen vom Anwender eingegeben
      // werden.
      cLTextOut1 = (CLTextOut) Beans.instantiate(getClass().getClassLoader(),
```

```
                                    CLTextOut.class.getName());
// Mit der Methode setDelay() wird das Zeitintervall festgelegt.
cLTimer1.setDelay(2000);
/*-----------------Zeitintervall in msec. 2000 == 2 sec.*/
// Diese von JBuilder generierte Art der Ereignisadapter für innere Klassen
// wird als anonyme Adapter bezeichnet. Sie verhindern, dass eine separate
// Adapterklasse erzeugt wird. Der resultierende Code ist kompakt.
// Erzeugen eines Ereignisadapters für das CLTimer Bean.
cLTimer1.addActionListener(new java.awt.event.ActionListener() {
                         public void actionPerformed(ActionEvent e) {
                            // Aufruf der Ereignismethode
                            cLTimer1_actionPerformed(e);
                         }
                      });
// Die Beschreibung der Variable erfolgt über einen S7 ANY-Pointer
s7Variable1.setS7Anypointer(
   new de.siemens.simaticnet.itcp.api.S7Anypointer(
      (int)5, (int)1, (int)131, (int)0, (int)10, (int)0));
/*-----------------------------------------------Bit Nummer 0 ..7*/
/*---------------------------------------Speicherbereichoffset*/
/*--------------------------------DB Nummer oder '0'*/
/*----------------------Speicherbereich 131 == M*/
/*--------------Wiederholfaktor 1 .. n*/
/*------Datentyp 5 == INT*/
// Erzeugen eines Ereignisadapters für das S7Variable Bean.
s7Variable1.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
   public void propertyChange(PropertyChangeEvent e) {
      // Aufruf der Ereignismethode
      s7Variable1_propertyChange(e);
   }
});
// Die Adressierung erfolgt über die Rahmennummer und Steckplatznummer der
// Baugruppe. Die Vorbesetzung der beiden Methoden zur Adressierung ist '0'.
// Aus diesem Grund kann auf die Rahmennummer verzichtet werden (.setRack(0)).
// Da wir mit der CPU kommunizieren wollen muss der Steckplatz der CPU
// eingetragen werden.
// ######## Projektspezifische Anpassung der Rahmen- und Steckplatznummer ########
// ######## notwendig                                              ########
s7Device1.setSlot(2);
/*----------------Steckplatznummer 2 (int)*/
// Erzeugen eines Ereignisadapters für das S7Device Bean.
s7Device1.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
   public void propertyChange(PropertyChangeEvent e) {
      // Aufruf der Ereignismethode
      s7Device1_propertyChange(e);
   }
});
// Zuweisen der IP-Adresse
// ######## Projektspezifische Anpassung der IP-Adresse notwendig ########
s7CP1.setHostString(
   new de.siemens.simaticnet.itcp.api.HostString ("192.168.1.1:80"));
/*-------------------------------------------------------------Angabe der
                                    Portnummer normalerweise :80*/
/*-----------------------------------------------IP-Adresse als String*/
// Erzeugen eines Ereignisadapters für das S7CP Bean.
s7CP1.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
   public void propertyChange(PropertyChangeEvent e) {
      // Aufruf der Ereignismethode
      s7CP1_propertyChange(e);
   }
});
// Setzt den Dimensionstext auf die angegebene Zeichenfolge.
cLTextOut1.setUnit("");
// Setzt den Beschreibungstext auf die angegebene Zeichenfolge.
cLTextOut1.setLabel("Value:");
```

```
    // Festlegen der Größe des Ausgabefeldes.
    cLTextOut1.setOutFieldSize(100);
    // Einfügen der Komponente ins Applet.
    this.add(cLTextOut1, null);
  }
  /**
   * Wird nach der Initialisierung eines Applets ausgeführt.
   * Bei Browsern wird start() auch dann aufgerufen, wenn eine Seite,
   * auf der sich ein Applet befindet, zum wiederholten Male geladen wird.
   *
   * @see #init
   * @see #stop
   * @see #destroy
   */
  public void start() {
    super.start();
  }
  /**
   * Ein Aufruf erfolgt, wenn der Browser bzw. der Appletviewer zum Icon
   * verkleinert oder eine HTML-Seite mit eingebundenem Applet in
   * einem Browser verlassen wird.
   *
   * @see #init
   * @see #start
   * @see #destroy
   */
  public void stop() {
    super.stop();
  }
  /**
   * Wird immer aufgerufen, wenn das Applet zerstört wird.
   *
   * @see #init
   * @see #start
   * @see #stop
   */
  public void destroy() {
    super.destroy();
    // Mit dieser Methode werden alle S7Bean-Instanzen gelöscht und Threads verworfen.
    // Nach dem Aufruf dieser Methode muss eine Neuinitialisierung durchlaufen werden.
    S7Api.terminate();
  }
  /**
   * Methode, um Ereignisse für den PropertyChangeListener des S7CP Beans zu behandeln.
   *
   * @param e PropertyChangeEvent
   */
  void s7CP1_propertyChange(PropertyChangeEvent e) {
    // Ereignis weiterleiten an die S7Device-Instanz
    s7Device1.propertyChange(e);
  }
  /**
   * Methode, um Ereignisse für den PropertyChangeListener des S7Device Beans zu
   * behandeln.
   *
   * @param e PropertyChangeEvent
   */
  void s7Device1_propertyChange(PropertyChangeEvent e) {
    // Ereignis weiterleiten an die S7Variable-Instanz
    s7Variable1.propertyChange(e);
  }
  /**
   * Methode, um Ereignisse für den PropertyChangeListener des S7Variable Beans zu
   * behandeln.
   *
```

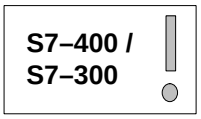
```
     * @param e PropertyChangeEvent
     */
    void s7Variable1_propertyChange(PropertyChangeEvent e) {
        // Dann Ausgabewert an die CLTextOut Instanz übergeben.
        cLTextOut1.propertyChange(e);
    }
    /**
     * Methode, um Ereignisse für den ActionPerformedEvent des CLTimer Beans zu behandeln.
     *
     * @param e ActionPerformedEvent
     */
    void cLTimer1_actionPerformed(ActionEvent e) {
        // Der Anstoss zum Lesen erfolgt mittels der Methode processGet()
        // vom S7Variable Bean.
        // Sind die neuen Werte vorhanden, so löst das S7Variable Bean ein
        // PropertyChangeEvent aus.
        s7Variable1.processGet();
    }
}
```

6.3 Eigene Beans entwickeln

Die S7–Beans halten sich an die Konventionen der JavaBeans und können grundsätzlich über sogenannte Getter– und Setter–Methoden angesprochen werden. Darüber hinaus verstehen die S7–Beans spezielle PropertyChangeEvents. So gibt eine Ausgabe–Bean (wie CLTextOut oder CLLevel) den von einer S7Variable per PropertyChangeEvent erhaltenen Wert direkt an ihre setValue()–Methode und stellt den Wert dar.

Die S7–Bean “S7–Variable” beispielsweise nimmt PCE–Namen InValue an und leitet den damit verknüpften Wert an die CPU weiter.

So erstellen Sie Beans, die mit der S7BeansAPI zusammenspielen (das folgende Beispiel gilt nur für S7–300 / S7–400)



Im folgenden wird die Vorgehensweise anhand der CLIdentOut erläutert.

CLIdentOut ist eine S7–Bean, die für die textliche Ausgabe einer Identnummer eines IT–CPs oder einer Baugruppe über die Bean S7CP oder S7Device benötigt wird.

```
//Deklarationsteil

package de.siemens.simaticnet.itcp.gui;

import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import java.io.Serializable;
import java.util.*;
public class CLIdentOut
extends Container
implements PropertyChangeListener, Serializable {

        private int outFieldSize;
        private String label;
        private Color outFieldColor = newColor (255, 255, 255);
        private Font theFont = new Font (”serif”, 0, 12);
        private String outValue;
        private Label label1 = new Label ();
        private TextField textField1 = new TextField ();
        private static CLGUIMessage international = CLGUIMessage.getInstance();

        public CLIdentOut() {
        try  {
        jbInit();
        }
        catch (Exception ex) {
        ex.printStackTrace();
        }
        }
```

//Erst mal alles notwendige initialisieren:

```
private void jbInit() throws Exception {
        this.setLayout(null);
        this.setSize(new Dimension(230, 60));
        this.setLayout(null);
        this.add(label1, null);
        this.add(textField1, null);


        label1.setAlignment(1);
        label1.setFont(theFont);
        label1.setText("?label?");
        label1.setBounds(new Rectangle(15, 17, 52,
        textField1.getMinimumSize(1).height));
        textField1.setFont(theFont);
        textField1.setEditable(false);
        textField1.setBackground(outFieldColor);
        textField1.setBounds(new Rectangle(66, 17, 160,
        textField1.getMinimumSize(1).height));
}
```

//Hier wird die Beschriftung gesetzt:

```
public void setLabel(String newLabel) {
        label = newLabel;
        label1.setText(newLabel);
        FontMetrics fm = label1.getFontMetrics(label1.getFont());
        int newWidth = fm.stringWidth(label)+ 20;
        if (newWidth != label1.getSize().width) {
                int h = newWidth – label1.getSize().width;
                Point p = new Point(label1.getLocation());
                label1.setBounds(p.x,p.y , newWidth,
                textField1.getMinimumSize(1).height);
                p = textField1.getLocation();
                p.translate(h,0);
                textField1.setBounds(p.x,p.y , textField1.getSize().width,
                textField1.getMinimumSize(1).height);
        }
}
public String getLabel() {
return label1.getText();
}
```

//Die Farbe des TextFields:

```
public void setOutFieldColor(Color newOutFieldColor) {
outFieldColor = newOutFieldColor;
textField1.setBackground(newOutFieldColor);
repaint();
}

public Color getOutFieldColor() {
return  textField1.getBackground();
}
```

//Die Größe des TextFields:

```
public void setOutFieldSize(int newOutFieldSize) {
   //outFieldSize = newOutFieldSize;
        if(textField1.getSize().width != newOutFieldSize){
                int h = newOutFieldSize – textField1.getSize().width;
                Point p = new Point(textField1.getLocation());
                textField1.setBounds(p.x, p.y, newOutFieldSize,
                textField1.getMinimumSize(1).height);
        }

        repaint();
}


public int getOutFieldSize() {
   return textField1.getSize().width;
}
```

//Die Schriftart einstellen:

```
public void setTheFont(Font newFont) {
        theFont = newFont;
        label1.setFont(newFont);
        textField1.setFont(newFont);

        repaint();
}


public Font getTheFont() {
   return theFont;
}

public Dimension getPreferredSize() {
   return new Dimension(230,60);
}
```

Hier folgt nun der spezifische Codeabschnitt, den Sie in entsprechenden eigenen Beans verwenden müssen.

```
//Hier wird überprüft, ob das ankommende Ereignis (PropertyChangeEvent) die
//Baugruppen–Identifikation enthält. Nur dann wird das Event ausgewertet und
//der Wert an setValue() übergeben, damit der Wert ausgegeben kann:

        public void propertyChange(PropertyChangeEvent pce) {
                if (pce.getPropertyName() == ”identification”) {
                        try {
                        setValue(pce.getNewValue().toString());
                        } catch (Exception e) {}
                }
        }
```

//Der übergebene String wird im TextField ausgegeben:

```
        public void setValue(String newOutValue) {
                outValue = newOutValue;
                textField1.setText(outValue);
        }

        public String getValue() {
           return outValue;
        }

        public void addMouseListener(MouseListener listener) {
                super.addMouseListener(listener);
                label1.addMouseListener(listener);
                textField1.addMouseListener(listener);
        }

        public void removeMouseListener(MouseListener listener) {
                super.removeMouseListener(listener);
                label1.removeMouseListener(listener);
                textField1.removeMouseListener(listener);
        }

        public synchronized void paint(Graphics g) {
                super.paint(g);

                Rectangle bl = label1.getBounds();
                Rectangle bt = textField1.getBounds();
                label1.setBounds(bl.x, bl.y, bl.width, textField1.getMinimumSize(1).height);
                textField1.setBounds(bt.x, bt.y, bt.width,
                textField1.getMinimumSize(1).height);
        }

}
```

6.4 Java–Applikationen mit S7Beans

Vorteile

Mit Java–Applikationen eröffnen sich weitaus vielfältigere Möglichkeiten als mit Java–Applets, da die Applikationen nicht den strengen Sicherheitsrichtlinien der Sandbox unterliegen. So lassen sich beispielsweise Zugriffe auf verschiedene SIMATIC–Stationen gleichzeitig realisieren, um Daten dieser Stationen zentral auf einem PC zu sammeln; oder es können Variablenwerte einer S7–CPU auf dem lokalen PC gespeichert und weiterverarbeitet werden.

Aufbau von Java–Applikationen

Der Aufbau der Applikationen ist etwas anders als bei den Applets, da hier statt den Applet–Browser–Schnittstellnmethoden init(), start(), stop() und destroy() die Hauptmethode main() die Kernroutine der Anwendung darstellt:

public static void main(java.lang.String[] args);

Innerhalb dieser öffentlichen statischen Methode können wie im Applet auch wieder Frames und Panels benutzt werden um die grafische Oberfläche darzustellen.

Voraussetzungen

Um Java–Applikationen erstellen und übersetzen zu können benötigen Sie entweder eine Java–Entwicklungsumgebung wie JBuilder von Borland oder VisualAge von IBM oder Sie verwenden die native Toolkette der Java–Entwicklungsumgebung (JDK) von Sun.

Funktionsweise des Beispiels

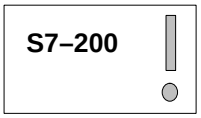
Im folgenden wird eine Java–Beispiel–Applikation vorgestellt, die von einer beliebigen SIMATIC–S7–Station – natürlich ausgestattet mit einem IT–CP und einer Ethernet–Verbindung zum eigenen PC – zyklisch Daten liest und in Dateien auf der lokalen Festplatte ablegt. Die Daten können dabei aus einem beliebigen Datenbaustein mit frei wählbarem Offset und Länge stammen (Achtung! Maximale Länge der übertragbaren Daten aus dem Gerätehandbuch entnehmen) .

Gespeichert werden die Daten beispielsweise auf der lokalen Festplatte in Dateien die als Ringpuffer beschrieben werden. D.h. den Dateien werden vor der Dateinamenserweiterung Nummern angehängt. Ist die letzte Datei (Anzahl ebenfalls einstellbar) geschrieben, wird wieder mit der ersten Datei mit Index 0 begonnen. Die Daten werden dabei binär gelesen und in der Datei wieder binär abgelegt.

Der Log–Vorgang kann auch über einen zentralen Button gestartet und auch wieder gestoppt werden und in der Statuszeile werden Erfolg– bzw. Fehlermeldungen angezeigt.

Das Beispiel als Grundlage für konkrete Anwendungen

Der Quelltext des Beispiels kann als Grundlage für weiterführende Anwendungen genutzt werden, da kurz und knapp das wesentliche beim Einsatz der S7Beans für Datenaustausch und Dateizugriff dargestellt wird. Auf umfangreiches Exceptionhandling und Plausibilitätskontrollen der eingegebenen Daten wird deswegen verzichtet. Das Beispiel setzt für die Oberflächengestaltung Swing–Komponenten ein und sollte deswegen mit einem Java2–Compiler übersetzt und mit einer Java2–Runtime–Umgebung (JRE 1.2.x, 1.3.x oder 1.4.x) gestartet werden.



Achtung

Die S7–200 kennt keine Datenbausteine. Bitte geben Sie deswegen als DB–Nummer immer ”1” an. Damit greifen Sie automatisch auf den ”Variablenspeicher” der S7–200 zu. Die AWL–Quelle im Beispiel muß für die S7–200 angepaßt werden.

Quelltext

```
// Zuordnung der Klassen zum "IT-CP-Example"-Paket
package de.siemens.simaticnet.itcp.example;

// Namensräume der verwendeten Klassen einblenden
import de.siemens.simaticnet.itcp.api.*;
import java.io.*;
import javax.swing.*;
import javax.swing.border.*;
import java.beans.*;
import java.awt.*;
import java.awt.event.*;
/**
 * LogfileDemo.java
 * <p>Überschrift: Beispiel für Einsatz der S7Beans in Java Applikationen.</p>
 * <p>Beschreibung: Zyklisches lesen von Daten aus einer SIMATIC S7 und<br>
 * Schreiben der Daten in Dateien auf dem Lokalen PC.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Organisation: Siemens AG SIMATIC NET</p>
 *
 * Zyklisches lesen von Daten aus einer SIMATIC S7 und
 * Schreiben der Daten auf dem Lokalen PC. Einsatz von
 * SWING-Komponenten unter Java2.
 *
 * Verwendete Komponenten:
 * S7CP
 * S7Device
 * S7Variable
 * CLTimer
 * SWING-Komponenten JFrame, JPanel, JLabel, JTextField und JButton
 *
 * @author ITCP-Team
 * @version 1.0
 *
 */
public class LogfileDemo extends JFrame implements ActionListener, PropertyChangeListener {

   // Deklaration der benötigten Komponenten
   private JPanel JFrameContentPane = null;
   private JPanel LogfileDemoPane = null;
```

```
private JButton btnStartStop = null;
private CLTimer s7Timer = null;
private S7CP s7CP = null;
private S7Device s7Device = null;
private S7Variable s7Variable = null;
private JTextField tfDB = null;
private JTextField tfExtension = null;
private JTextField tfHostname = null;
private JTextField tfLength = null;
private JTextField tfOffset = null;
private JTextField tfPeriod = null;
private JTextField tfPrefix = null;
private JTextField tfRack = null;
private JTextField tfSlot = null;
private JLabel labCPOptions = null;
private JLabel labCPUOptions = null;
private JLabel labData = null;
private JLabel labExtension = null;
private JLabel labHostname = null;
private JLabel labLogOptions = null;
private JLabel labPeriod = null;
private JLabel labPrefix = null;
private JLabel labRackSlot = null;
private boolean bRunning = false;
private int iCounter = 0;
private JLabel labMaxCounter = null;
private JTextField tfMaxCounter = null;
private JLabel labMsg1 = null;
private JLabel labMsg2 = null;
private JPanel StatusBarPane = null;
/**
 * Konstruktor mit der Initialisierung aller Objekte.
 */
public LogfileDemo() {
   // Konstruktor der Basisklasse aufrufen
   super();
   // Setzen wichtiger Fensterattribute
   // Fenstertitel
   setTitle("Logfile Demo");
   // Fenstergröße
   setSize(400, 400);
   // Applikation beenden beim Schließen des Fensters
   setDefaultCloseOperation(EXIT_ON_CLOSE);

   // Initialisieren sämtlicher Oberflächenelemente
   // Objekt anlegen
   labCPOptions = new JLabel();
   // Darzustellender Text festlegen
   labCPOptions.setText("SIMATIC S7 IT-CP Options");
   // Position und Größe einstellen
   labCPOptions.setBounds(20, 15, 181, 20);
   labHostname = new JLabel();
   labHostname.setText("Hostname or IP-Address:");
   labHostname.setBounds(34, 44, 163, 14);
   tfHostname = new JTextField();
   tfHostname.setText("192.168.0.1");
   tfHostname.setBounds(210, 41, 150, 20);
   labCPUOptions = new JLabel();
   labCPUOptions.setText("SIMATIC S7 CPU Options");
   labCPUOptions.setBounds(20, 73, 181, 20);
   labRackSlot = new JLabel();
   labRackSlot.setText("Rack / Slot:");
   labRackSlot.setBounds(34, 102, 173, 14);
   tfRack = new JTextField();
   tfRack.setText("0");
```

```
tfRack.setBounds(210, 99, 40, 20);
tfSlot = new JTextField();
tfSlot.setText("3");
tfSlot.setBounds(265, 99, 40, 20);
labData = new JLabel();
labData.setText("DB / offset / length:");
labData.setBounds(34, 128, 173, 14);
tfDB = new JTextField();
tfDB.setText("17");
tfDB.setBounds(210, 125, 40, 20);
tfOffset = new JTextField();
tfOffset.setText("0");
tfOffset.setBounds(265, 125, 40, 20);
tfLength = new JTextField();
tfLength.setText("8");
tfLength.setBounds(320, 125, 40, 20);
labLogOptions = new JLabel();
labLogOptions.setText("Logging Options");
labLogOptions.setBounds(20, 157, 181, 20);
labPrefix = new JLabel();
labPrefix.setText("Logfile prefix:");
labPrefix.setBounds(34, 186, 173, 14);
tfPrefix = new JTextField();
tfPrefix.setText("Demo");
tfPrefix.setBounds(210, 183, 150, 20);
labExtension = new JLabel();
labExtension.setText("Logfile extension:");
labExtension.setBounds(34, 212, 173, 14);

tfExtension = new JTextField();
tfExtension.setText("dat");
tfExtension.setBounds(210, 209, 150, 20);
labMaxCounter = new JLabel();
labMaxCounter.setText("Max. Logfile Count:");
labMaxCounter.setBounds(34, 238, 173, 14);
tfMaxCounter = new JTextField();
tfMaxCounter.setText("20");
tfMaxCounter.setBounds(210, 235, 40, 20);
labPeriod = new JLabel();
labPeriod.setText("Logging period in seconds:");
labPeriod.setBounds(34, 264, 173, 14);
tfPeriod = new JTextField();
tfPeriod.setText("3");
tfPeriod.setBounds(210, 261, 40, 20);
btnStartStop = new JButton();
btnStartStop.setText("Start Logging");
btnStartStop.setBounds(134, 315, 134, 25);
// Panel anlegen, um die Objekte darauf pixelgenau
// positionieren zu können.
LogfileDemoPane = new JPanel();
LogfileDemoPane.setLayout(null);
LogfileDemoPane.add(labCPUOptions);
LogfileDemoPane.add(labHostname);
LogfileDemoPane.add(tfHostname);
LogfileDemoPane.add(labCPOptions);
LogfileDemoPane.add(labRackSlot);
LogfileDemoPane.add(tfRack);
LogfileDemoPane.add(tfSlot);
LogfileDemoPane.add(labData);
LogfileDemoPane.add(tfDB);
LogfileDemoPane.add(tfOffset);
LogfileDemoPane.add(tfLength);
LogfileDemoPane.add(labLogOptions);
LogfileDemoPane.add(labPrefix);
LogfileDemoPane.add(tfPrefix);
```

```
    LogfileDemoPane.add(labExtension);
    LogfileDemoPane.add(tfExtension);
    LogfileDemoPane.add(labMaxCounter);
    LogfileDemoPane.add(tfMaxCounter);
    LogfileDemoPane.add(labPeriod);
    LogfileDemoPane.add(tfPeriod);
    LogfileDemoPane.add(btnStartStop);
    // Objekte für die Statuszeile anlegen und
    // positionieren
    labMsg1 = new JLabel();
    labMsg1.setBorder(new EtchedBorder());
    labMsg1.setText("Not running   ");
    labMsg2 = new JLabel();
    labMsg2.setBorder(new EtchedBorder());
    labMsg2.setText("");
    StatusBarPane = new JPanel();
    StatusBarPane.setLayout(new BorderLayout());
    StatusBarPane.add(labMsg1, "West");
    StatusBarPane.add(labMsg2, "Center");
    // Panel zur Positionierung des Haupt- und Status-Panels
    // anlegen. Untergeordnete Panels aufnehmen und als
    // Hauptpanel beim Rahmen anmelden
    JFrameContentPane = new JPanel();
    JFrameContentPane.setLayout(new BorderLayout());
    JFrameContentPane.add(StatusBarPane, "South");
    JFrameContentPane.add(LogfileDemoPane, "Center");
    setContentPane(JFrameContentPane);

    // S7Beans für die Datenkommunikation anlegen
    s7CP = new S7CP();
    s7Device = new S7Device();
    s7Variable = new S7Variable();
    s7Timer = new CLTimer();
    // Beim Timer den Autostart-Mechanismus abbrechen
    s7Timer.stop();

    // Registrieren der Objekte bei der Ereignisquelle.
    s7CP.addPropertyChangeListener(this);
    s7Device.addPropertyChangeListener(this);
    s7Variable.addPropertyChangeListener(this);
    s7Timer.addActionListener(this);
    btnStartStop.addActionListener(this);
}

/**
 * Methode, um Ereignisse für die ActionListener Schnittstelle
 * zu behandeln.
 *
 * @param e java.awt.event.ActionEvent
 */
public void actionPerformed(ActionEvent e) {
    // Timer abgelaufen?
    if (e.getSource() == s7Timer)
        // dann neue Daten von SIMATIC S7 holen
        // Sobald die neuen Daten in der S7Variablen angekommen
        // sind, löst diese ihrerseits ein Property-Change-Event aus.
        s7Variable.processGet();
    // Start-/Stop-Button gedrückt?
    if (e.getSource() == btnStartStop)
        // dann entsprechenden Handler aufrufen
        onStartStop();
}

/**
 * Methode, um Ereignisse für die PropertyChangeListener
```

```
 * Schnittstelle zu behandeln.
 *
 * @param evt PropertyChangeEvent
 */
public void propertyChange(PropertyChangeEvent evt) {
   // Ereignis vom S7CP ausgelöst?
   if (evt.getSource() == s7CP)
      // dann Ereignis weiterleiten an die S7Device-Instanz
      s7Device.propertyChange(evt);
   // Ereignis vom S7Device ausgelöst?
   if (evt.getSource() == s7Device)
      // dann Ereignis weiterleiten an die S7Variable-Instanz
      s7Variable.propertyChange(evt);
   // Ereignis vom S7Variable ausgelöst?
   if (evt.getSource() == s7Variable)
      // dann entsprechenden Handler aufrufen
      onNewS7Data(evt);
}

/**
 * Ereignisbehandlungsroutine für das Drücken des
 * Start-/Stop-Buttons.
 */
public void onStartStop() {
   // Falls das Logging gerade läuft
   if (bRunning) {
      // Timer stoppen, damit keine weiteren Daten
      // von der SIMATIC S7 abgeholt und als Datei
      // geschrieben wird.
      s7Timer.stop();
      bRunning = false;

      // Statustext und Button-Text entsprechend setzen
      labMsg1.setText("Not running   ");
      btnStartStop.setText("Start Logging");
   } else {
      // Parametrierung vor dem Start einlesen und S7-Beans
      // entsprechend einstellen.
      // Hostname bzw. IP-Adresse setzen
      s7CP.setHost(tfHostname.getText());

      // Rack-/Slotnummer einstellen
      int iRack = Integer.parseInt(tfRack.getText());
      int iSlot = Integer.parseInt(tfSlot.getText());
      s7Device.setRack(iRack);
      s7Device.setSlot(iSlot);

      // S7-Anypointer im Format "P#DBxx.DBXyy BYTE zz"
      // aus den eingegebenen Daten zusammenstellen
      int iDB = Integer.parseInt(tfDB.getText());
      int iLength = Integer.parseInt(tfLength.getText());
      int iOffset = Integer.parseInt(tfOffset.getText());
      s7Variable.setS7Anypointer(new S7Anypointer(
         S7Anypointer.S7_BYTE, iLength, S7Anypointer.MEM_AREA_DB,
         iDB, iOffset, 0));

      // Timerobjekt auf die eingegebene Zeit in Millisekunden
      // einstellen und starten
      int iPeriod = Integer.parseInt(tfPeriod.getText())*1000;
      s7Timer.setDelay(iPeriod);
      s7Timer.start();
      bRunning = true;

      // Statustext und Button-Text entsprechend setzen
      labMsg1.setText("Running       ");
```

```
            btnStartStop.setText("Stop Logging");
        }
    }

    /**
     * Ereignisbehandlungsroutine für das Eintreffen neuer Daten
     * von der SIMATIC S7.
     *
     * @param evt PropertyChangeEvent von der S7Variablen
     */
    public void onNewS7Data(PropertyChangeEvent evt) {
        // Sicherheitsabfrage, damit keine verspätet eintreffenden Daten
        // mehr in eine Datei geschrieben werden
        if (!bRunning) {
            return;
        }

        // Bei einem Kommunikationsfehler übergibt die S7Variable
        // den Text "S7Error" am Anfang des neuen PCE-Wertes
        if (evt.getNewValue().toString().startsWith("S7Error")) {
            // Fehlermeldung in der Statuszeile anzeigen
            labMsg2.setText(evt.getNewValue().toString());
            return;
        }
        // Aktuellen Dateinamen aus dem eingegebenen Dateiprefix, dem
        // aktuellem Dateiindex und der Dateierweiterung zusammensetzen
        int iMaxCounter = Integer.parseInt(tfMaxCounter.getText());
        String strFilno = "00000000" + iCounter;
        // Maximal notwendige Stellenanzahl für Dateiindex ermitteln
        int iMaxCntWidth = (int)Math.ceil(Math.log(iMaxCounter) /
            Math.log(10.0));
        strFilno = strFilno.substring(strFilno.length() - iMaxCntWidth);
        String strFilename = tfPrefix.getText() + strFilno + "." +
            tfExtension.getText();
        // Datei mit gerade ermitteltem Dateinamen für Schreibzugriffe
        // öffnen und aktuelle Daten der S7Variable als Byte-Array binär
        // in die geöffnete Datei schreiben
        try {
            File file = new File(strFilename);
            FileOutputStream out = new FileOutputStream(file);
            out.write(s7Variable.getDataObjectAsByteArray());
            out.flush();
            out.close();
        } catch (Exception e) {
            // Bei einem Dateifehler Ausgabe der Exception in der
            // Statuszeile
            labMsg2.setText(e.toString());
            return;
        }
        // Erfolgsmeldung in der Statuszeile anzeigen
        labMsg2.setText("Data logged into file " + strFilename);

        // Dateiindex für nächsten Durchgang vorbelegen
        iCounter++;
        if (iCounter >= iMaxCounter) {
            iCounter = 0;
        }
    }

    /**
     * Hautpfunktion der Applikation
     *
     * @param args Array mit den Befehlszeilenargumenten
     */
    public static void main(java.lang.String[] args) {
```

```
      /* Hauptfenster anlegen */
      LogfileDemo aLogfileDemo = new LogfileDemo();
      /* Ereignisbehandlung für "Fenster schließen" registrieren */
      aLogfileDemo.addWindowListener(new WindowAdapter() {
         public void windowClosed(WindowEvent e) {
            System.exit(0);
         };
      });
      // Hauptfenster anzeigen
      aLogfileDemo.setVisible(true);
   }
}
```

Kompilieren und Starten

Sie können den Java–Quelltext in Ihrer Java–Entwicklungsumgebung importieren, übersetzen und starten. Aber auch ohne IDE mit den nativen Java–Tools können Sie das Beispiel übersetzen und starten.

Angenommen, Sie haben die JAR–Files der S7BeansApi und die Quelldateien der Beispiele auf Ihrem Windows–PC im Verzeichnis C:\S7BeansApi abgelegt und das Java–Development–Kit (JDK) im Verzeichnis C:\jdk1.4.x" installiert, dann müssten Sie zum Kompilieren des Beispiels den folgenden Kommandozeilenaufruf angeben:

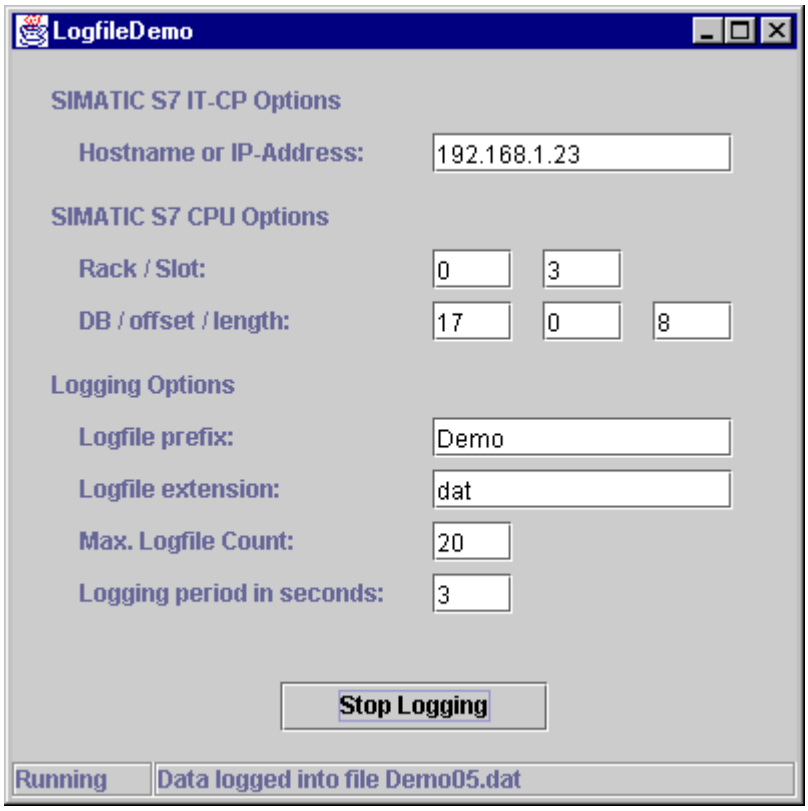
```
C:\S7BeansApi>C:\jdk1.4.x\bin\javac.exe-classpath "C:\S7Bean-
sApi\s7api.jar" -d "." "LogfileDemo.java"
```

Im Classpath muß auf jeden Fall das Java–Archiv der S7Beans "s7api.jar" angegeben werden, da das Beispiel vier Beans aus dieser Bibliothek verwendet. Mit der Parametersequenz –d "." wird angegeben, daß die ausgegebenen Class–Dateien unterhalb des aktuellen Pfades unter Verwendung der Verzeichnisstruktur wie sie sich aus dem Package–Namen ergibt ausgegeben werden. Im Beispiel wäre das der Pfad "C:\ S7BeansApi\de\siemens\simaticnet\itcp\example".

Wurde der Kompiliervorgang erfolgreich durchgeführt, so kann die Applikation nun gestartet werden. Hierzu folgender Kommandozeilenaufruf:

```
C:\S7BeansApi>C:\jdk1.4.x\bin\java.exe-cp ".";"C:\S7BeansApi\s7api.jar"
de.siemens.simaticnet.itcp.example.LogfileDemo
```

Screenshot der gestarteten Java–Applikation



Der im Screenshot dargestellte Fall greift auf eine SIMATIC S7 CPU im Rack 0 Slot 3 über einen IT–CP mit der IP–Adresse 192.168.1.23 zu und holt alle 3 Sekunden 8 Byte aus dem DB17 ab Offset 0 ab. Die Daten werden dabei in Dateien des aktuellen Verzeichnisses mit dem Schema Demo00.dat – Demo19.dat geschrieben.

Damit das Beispiel mit diesen Einstellungen auch ohne Fehlermeldung läuft, muss der DB17 vorhanden und mindestens 8 Byte groß sein.

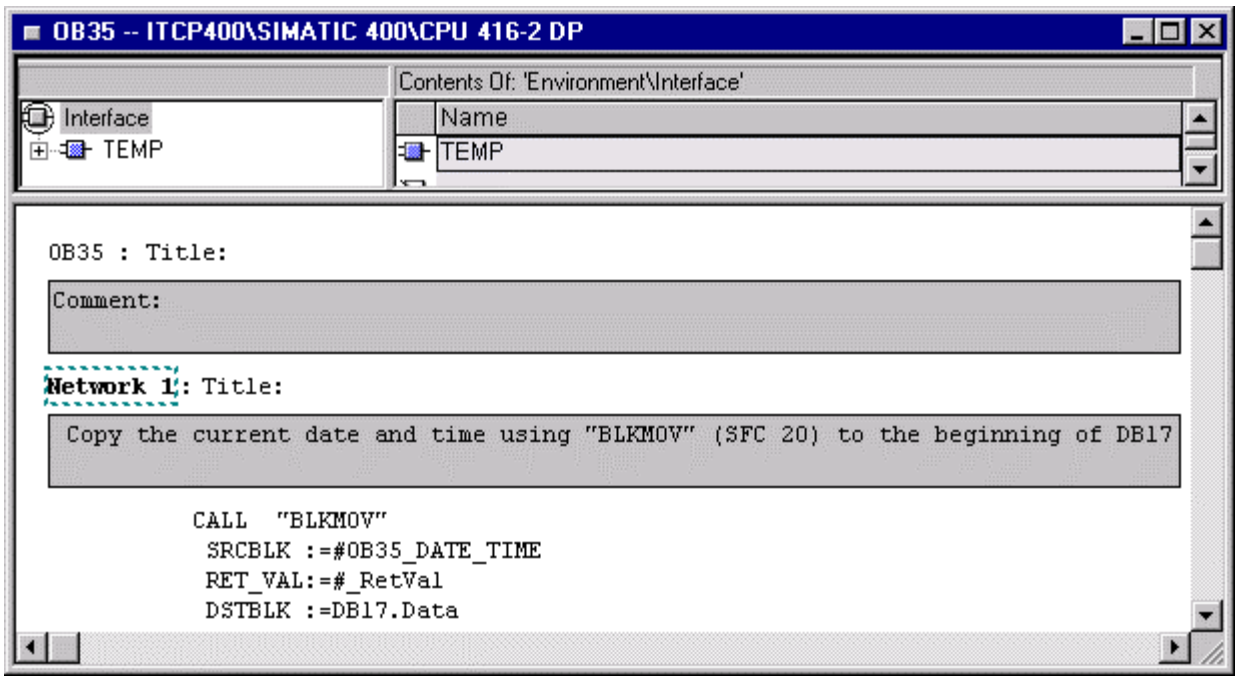
Hierzu folgende AWL–Quelle, die den DB17 bereitstellt und über den Weckarlarm 5 (OB35) alle 100ms das aktuelle CPU–Datum und Uhrzeit hineinkopiert. Nach dem Übersetzen dieser Quelle, hochladen der Bausteine und Starten der Java–Applikation mit den passenden Parametern wird dann alle 3 Sekunden das Datum und die Uhrzeit der CPU in BCD–Kodierung in eine Datei auf dem PC geschrieben.

Text der AWL–Quelle

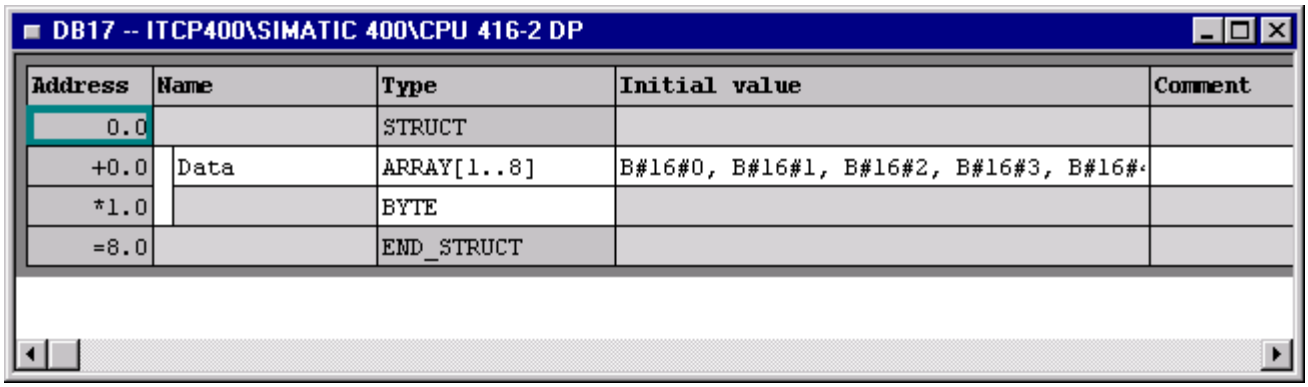
```
DATA_BLOCK DB 17

  STRUCT
   Data : ARRAY  [1 .. 8 ] OF
   BYTE  := B#16#0, B#16#1, B#16#2, B#16#3, B#16#4, B#16#5, B#16#6, B#16#7;
  END_STRUCT ;
```

```
BEGIN
END_DATA_BLOCK

ORGANIZATION_BLOCK OB35

VAR_TEMP
  OB35_EV_CLASS : BYTE ;     // Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event
class 1)
  OB35_STRT_INF : BYTE ;     // 16#36 (OB 35 has started)
  OB35_PRIORITY : BYTE ;     // Priority of OB Execution
  OB35_OB_NUMBR : BYTE ;     // 35 (Organization block 35, OB35)
  OB35_RESERVED_1 : BYTE ;   // Reserved for system
  OB35_RESERVED_2 : BYTE ;   // Reserved for system
  OB35_PHS_OFFSET : INT ;    // Phase offset (integer, milliseconds)
  OB35_RESERVED_3 : INT ;    // Reserved for system
  OB35_EXC_FREQ : INT ;      // Frequency of execution (msec)
  OB35_DATE_TIME : DATE_AND_TIME ;  // Date and time OB35 started
  _RetVal : INT ;            // Return value for BLKMOV
END_VAR
BEGIN
// Copy the current date and time using "BLKMOV" (SFC 20) to the beginning of
DB17
      CALL SFC 20 (
           SRCBLK                  := #OB35_DATE_TIME,
           RET_VAL                 := _RetVal,
           DSTBLK                  := P#DB17.DBX0.0 BYTE 8);

END_ORGANIZATION_BLOCK
```

Screenshot des OB35



Screenshot des DB17



A Property Change Events – Übersicht

Die folgende Tabelle zeigt Ihnen, welche S7Beans welche Property–Change–Events (PCE) erwarten und welche Bedeutung bzw. Typ dabei der mitgeführte Wert *NewValue* haben sollte. Ein ' * ' als PCE–Namen meint dabei, daß der Name nicht geprüft wird, das heißt, alle ankommenden bzw. vorher noch nicht ausgefilterte PCE werden gleich interpretiert.

Tabelle A-1

Empfänger–Bean	PCE–Name	Typ/ Beschreibung des erwarteten PCE–Wertes (NewValue)
S7Variable	inValue	zu setzender Wert für die S7 von Eingabebean z.B. CLTextIn
	*	zu setzender Wert für die S7 (wirft Warnung in Java–Konsole aus)
CLPipe	*	Vom Bean anzuzeigender Wert (Wert > 0 = voll)
CLHPipe	*	Vom Bean anzuzeigender Wert (Wert > 0 = voll)
CLVPipe	*	Vom Bean anzuzeigender Wert (Wert > 0 = voll)
CLLevel	*	Vom Bean anzuzeigender Wert (numerischer Wert)
CLTacho	*	Vom Bean anzuzeigender Wert (numerischer Wert)
CLThermo	*	Vom Bean anzuzeigender Wert (numerischer Wert)
CLValve	*	Vom Bean anzuzeigender Wert (Wert > 0 = offen)
CLTextOut	*	Vom Bean anzuzeigender Wert oder Text
ConvertNumberSystem	*	Vom Bean umzuwandelnder Wert (Dezimalzahl)
COUNTER	*	zu konvertierender und dann weiterzuleitender Wert Der Datentyp bestimmt dabei die Richtung: byte[] : S7Variable –> COUNTER –> Output–Bean String: Input–Bean –> COUNTER –> S7Variable
DATE *)	*	zu konvertierender und dann weiterzuleitender Wert Der Datentyp bestimmt dabei die Richtung: Integer : S7Variable –> DATE –> Output–Bean String: Input–Bean –> DATE –> S7Variable

Tabelle A-1 , Fortsetzung

Empfänger–Bean	PCE–Name	Typ/ Beschreibung des erwarteten PCE–Wertes (NewValue)
DATEandTIME *)	*	zu konvertierender und dann weiterzuleitender Wert Der Datentyp bestimmt dabei die Richtung: byte[]: S7Variable –> DATEandTIME –> Output–Bean String: Input–Bean –> DATEandTIME –> S7Variable
TIME *)	*	zu konvertierender und dann weiterzuleitender Wert Der Datentyp bestimmt dabei die Richtung: Long: S7Variable –> TIME –> Output–Bean String: Input–Bean –> TIME –> S7Variable

*) diese S7–Typen werden bei S7–200 nicht unterstützt.

Die nächste Tabelle zeigt, welche S7Beans welche PCE selbst an ihre Listener senden und welche Bedeutung bzw. Typ dabei der mitgeführte Wert *NewValue* hat.

Tabelle A-2

Sender	PCE–Name	Typ/Beschreibung des gesendeten PCE–Wertes (NewValue)
S7CP	S7CP	Objekt vom Typ S7CP
	slot	Slot als Integer (bei S7–200 immer Wert=0)
	rack	Rack als Integer (bei S7–200 immer Wert=0)
	host	Hostname als String
	s7NetAddress	Objekt vom Typ S7NetAddress
	state	Status als Integer
	identification	Identifikation als String
	moduleName	Modulname als String
S7Device	S7Device	Objekt vom Typ S7Device
	slot	Slot als Integer (bei S7–200 immer Wert=0)
	rack	Rack als Integer (bei S7–200 immer Wert=0)
	host	Hostname als String
	s7NetAddress	Objekt vom Typ S7NetAddress
	state	Status als Integer
	identification	Identifikation als String
	moduleName	Modulname als String
S7Variable	[Variablenname]	Der neue Wert der Variablen nach einem “GET”. Achtung: Der Variablenname wird individuell vom Anwender bei der Projektierung des S7Variable–Beans vergeben
CLTextIn	inValue	String newInValue
ConvertNumberSystem	ConvertNumberSystem	Konvertierter String je nach Einstellung z.B. als Hex– oder Oktal–String
COUNTER *)	COUNTER	Konvertierter String
DATE *)	DATE	Konvertierter String
DATEandTIME *)	DATEandTIME	Konvertierter String
S5TIME *)	S5TIME	Konvertierter String

Tabelle A-2 , Fortsetzung

Sender	PCE–Name	Typ/Beschreibung des gesendeten PCE–Wertes (NewValue)
TIME *)	TIME	Konvertierter String
TIMEofDAY *)	TIMEofDAY	Konvertierter String
TIMER	TIMER	Konvertierter String

*) diese S7–Typen werden bei S7–200 nicht unterstützt.

❑

B Weitere Hinweise / FAQs

B.1 Ressourcenengpässe unter Netscape 4.x

Frage

Nach mehrmaligem Aufruf meiner HTML–Seiten mit meinen auf S7BeansAPI–Basis entwickelten Applets wird der Browser immer langsamer, sammelt immer mehr Systemressourcen an und stürzt dann irgendwann sogar ganz ab. Unter Umständen wird sogar mein Betriebsystem in Mitleidenschaft gezogen. Wie kann ich diese Effekte verhindern?

Antwort

Um Ressourcen–Probleme beim Einsatz der mit den S7Beans entwickelten Applets zu vermeiden, sollten beim Beenden des Applets alle Ressourcen freigegeben und alle Threads gestoppt werden. Da die S7BeansAPI intern statische Ressourcen und Threads für die Kommunikation mit dem IT–CP verwendet, wurde seit der Version V2.3 die Methode terminate() in der Klasse S7Api geschaffen, um diese Ressourcen freizugeben und alle Threads zu stoppen.

Terminate() kann normalerweise nach dem Freigeben aller Ressourcen in der destroy()–Methode des Applets aufgerufen werden. Da jedoch einige Browser (z.B. Netscape) die destroy()–Methode nicht direkt bei Verlassen der HTML–Seite aufrufen, sondern erst bei Entfernen des Applets aus dem History–Cache des Browsers, ist es ratsam die Ressourcen schon in der stop()–Methode des Applets freizugeben und auch gleich terminate() aus der S7Api–Klasse aufzurufen. Dies erfordert allerdings dann ein erneutes Initialisieren sämtlicher Ressourcen in der start()–Methode des Applets, da das Applet z.B. unter Netscape auch durch ein Ändern der Browser–Fenstergröße gestoppt und wieder gestartet wird.

Weiterhin hat sich die Kombination MS Windows NT 4.0 oder
Windows 2000/XPund Internet Explorer 5.5 oder 6.0 als die stabilste Kombination erwiesen. Beim Einsatz von Windows 95 oder 98 bzw. Windows ME können Ressourcen–Probleme im Browser schnell zu Beeinträchtigungen des Betriebsystems führen. Bei schnell wechselnden HTML–Seiten mit vielen Applets ist der Einsatz der Netscape–Browser nicht zu empfehlen, da diese dann schneller Systemressourcen allokieren, als sie wieder freigeben.

Beispiel

Nachfolgend finden Sie einen Abdruck des bereits in Kapitel 6.1.1 vorgestellten Beispieles, das für die Anwendung unter Netscape 4.x angepasst wurde und die oben beschriebenen Probleme vermeidet.

```
package de.siemens.simaticnet.itcp.example;

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import de.siemens.simaticnet.itcp.api.*;
import de.siemens.simaticnet.itcp.gui.*;

/**
 * @author       ITCP-Team
 */
public class Example1Net4x extends Applet implements PropertyChangeListener, ActionListener {
        private CLTextOut cLTextOut1 = null;
        private CLTimer cLTimer1 = null;
        private S7CP s7CP1 = null;
        private S7Device s7Device1 = null;
        private S7Variable s7Variable1 = null;
        private boolean isInit = false;
// flag to remember init is done or not

        /**
         * Initializes the applet.
         *
         * @see #start
         * @see #stop
         * @see #destroy
         */
        public void init() {
                super.init();
                setName("VarianteA");
                setLayout(null);
                setSize(426, 240);

                s7CP1 = new S7CP();
                s7CP1.setHostString(new HostString ("192.168.1.23"));

                s7Device1 = new S7Device();
                s7Device1.setSlot(3);

                s7Variable1 = new S7Variable();
                s7Variable1.setS7Anypointer(new S7Anypointer((int)2, (int)1, (int)131,
                (int)0, (int)10, (int)0));
                s7Variable1.setVariableName("s7Variable1");

                cLTimer1 = new CLTimer();
                cLTimer1.setDelay(2000);

                cLTextOut1 = new CLTextOut();
                cLTextOut1.setName("cLTextOut1");
                cLTextOut1.setBounds(0, 0, 200, 45);
                cLTextOut1.setOutFieldSize(100);
                cLTextOut1.setLabel("Value:");
                cLTextOut1.setUnit("");
                add(cLTextOut1, cLTextOut1.getName());

                /* init connections */
                s7CP1.addPropertyChangeListener(this);
                s7Device1.addPropertyChangeListener(this);
                cLTimer1.addActionListener(this);
                s7Variable1.addPropertyChangeListener(this);

                // initialization is valid
                isInit = true;

        }
```

```
        /**
         * Wird zum Starten des Applets aufgerufen.Diese Methode muß nie direkt aufgerufen werden.
         * Sie wird beim Zugriff auf das Applet-Dokument aufgerufen.
         * @see #init
         * @see #stop
         * @see #destroy
         */
        public void start() {
                super.start();

                // Is a call to init necessary?

                if (!isInit)
                        init();
// applet was stopped before, so call init again
        }
        /**
         * Wird zum Stoppen des Applets aufgerufen.
         * Wird aufgerufen, wenn das Applet-Dokument nicht mehr
         * angezeigt wird. Wird immer aufgerufen, bevor destroy()
         * aufgerufen wird. Diese Methode muß nie direkt aufgerufen werden.
         * @see #init
         * @see #start
         * @see #destroy
         */
        public void stop() {
                super.stop();

                // remove all components from this container
                removeAll();

                // destroy all used resources here
                cLTextOut1 = null;
                cLTimer1 = null;
                s7CP1 = null;
                s7Device1 = null;
                s7Variable1 = null;

                // remember initilization is not done
                isInit = false;

                // terminate the api mechanisms, finalize objects
                // and run the Garbage Collector
                S7Api.terminate();
        }
        /**
         * Cleans up whatever resources are being held. If the applet is active
         * it is stopped.
         *
         * @see #init
         * @see #start
         * @see #stop
         */
        public void destroy() {
                super.destroy();

                // clean up is done in stop method now !
        }

        /**
         * Method to handle events for the PropertyChangeListener interface.
         * @param evt PropertyChangeEvent
         */
        public void propertyChange(PropertyChangeEvent pce) {
                if (pce.getSource() == s7CP1)
```

```
                    s7Device1.propertyChange(pce);
            if (pce.getSource() == s7Device1)
                    s7Variable1.propertyChange(pce);
            if (pce.getSource() == s7Variable1)
                    cLTextOut1.propertyChange(pce);
    }

    /**
     * Method to handle events for the ActionListener interface.
     * @param e java.awt.event.ActionEvent
     */
    public void actionPerformed(ActionEvent ae) {
            if (ae.getSource() == cLTimer1)
                    s7Variable1.processGet();
    }
}
```

C Gegenüberstellung von S7–Typen und Java–Typen

S7–Typen unterscheiden sich in der Regel von den entsprechenden Java–Typen im Wertebereich. Um hier Problemen mit Vorzeichen, etc vorzubeugen, werden die Typen wie in der folgenden Tabelle dargestellt abgebildet. Bei Verwendung von Arrays (Wiederholfaktor (n) > 1) werden in Java die sogenannten ”elementaren” Datentypen verwendet. Ein solches Array (z.B. int[]) wird dann von Java auch als Objekt betrachtet, was bei den einfachen Datentypen sonst nicht der Fall ist. Dies ist für uns insbesondere wichtig für die Übergabeparameter, z.B. bei setValue() von S7Variable.

Tabelle C-1

S7–Typ	Java–Typ	Java–Array
BOOL	Boolean	–
BYTE	Integer	int[]
CHAR	Character	char[]
WORD	Integer	int[]
INT	Integer	int[]
DWORD	Long	long[]
DINT	Long	long[]
REAL	Float	float[]
DATE *)	Integer	int[]
Time Of Day *)	Long	long[]
TIME *)	Long	long[]
S5TIME *)	byte[2]	byte[n*2]
Date And Time	byte[8]	byte[n*8]
STRING	String	–
COUNTER *)	byte[2]	byte[n*2]
TIMER	byte[2]	byte[n*2]

*) diese S7–Typen werden bei S7–200 nicht unterstützt.

Achtung

– Ein Array vom Typ BOOL kann nicht gebildet werden.

– Ein Array vom Typ STRING kann nicht gebildet werden.

D Literaturverzeichnis

/1/ Gerätehandbuch SIMATIC NET CP
Beschreibung der Gerätehandhabung und Installation
SIEMENS AG

/2/ NCM S7 für Industrial Ethernet Handbuch
Bestandteil
– des Handbuch–Paketes NCM S7 für Industrial Ethernet
– der Online–Dokumentation in STEP 7 / Option NCM S7 für Industrial Ethernet
Siemens AG

/3/ Informationstechnologie bei SIMATIC S7 mit CP 343–1 IT und CP 443–1 IT
Bestandteil
– des Handbuch–Paketes NCM S7 für Industrial Ethernet
– der Online–Dokumentation in STEP 7 / Option NCM S7 für Industrial Ethernet
Siemens AG

/4/ Gerätehandbuch CP 243–1 IT
Kommunikationsprozessor für Industrial Ethernet und Informations–Technologie
Projektierung / Programmierung
Siemens AG

/5/ Programmierhilfe für S7 Beans (für Visual Age)
Siemens AG
beziehbar über Internet

Bestellnummern

Die Bestellnummern für die oben genannten SIEMENS–Dokumentationen sind in den Katalogen ”SIMATIC NET Industrielle Kommunikation, Katalog IK10” und ”SIMATIC Automatisierungssysteme SIMATIC S7 / M7 / C7 – Komponenten für die vollintegrierte Automation, Katalog ST70” enthalten.

Diese Kataloge sowie zusätzliche Informationen können bei den jeweiligen SIEMENS–Zweigniederlassungen und Landesgesellschaften angefordert werden.

Einige der hier genannten Dokumente finden Sie auch auf der Manual Collection CD, die jedem S7–CP beiliegt.

Zusätzliche Literaturempfehlungen zum Thema Internet/Web, HTML, Java

/6/ Web–Publishing mit HTML 4
Deborah S.Ray / Eric J.Ray
Sybex Verlag 1998

/7/ Durchblick im Netz
Vom PC–LAN zum Internet
Kauffels, F–J.
Internat. Thomson Publ., 1998
ISBN 3–8266–0413–X

/8/ Campione/ Walrat
The JavaTM Tutorial
Second Edition
Object–Oriented Programming for the Internet
ADDISON–WESLEY, 1998
ISBN 0–201–31007–4

Zum Lernen von Java gibt es inzwischen eine große Auswahl an Büchern; besonders zu empfehlen sind:

/9/ Java in 21 Tagen
von Laura Lemay und Charles L. Perkins
ISBN: 3827255783

/10/ Java in a Nutshell
von David Flanagan
ISBN: 3897211009

/11/ Java Examples in a Nutshell
von David Flanagan
ISBN: 3897211122
(für den schnellen Einstieg mit Programmiererfahrung)

❑

T

U

W

Z