SIEMENS

SIMATIC

PRODAVE MPI/IE V6.0

Handbuch

Ausgabe 05/2005
A5E00388167-01

Sicherheitstechnische Hinweise

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 oder erheblicher Sachschaden eintreten **werden**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Warnung
bedeutet, dass Tod, schwere Körperverletzung oder erheblicher Sachschaden eintreten **können**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht
bedeutet, dass eine leichte Körperverletzung oder ein Sachschaden eintreten können, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht
bedeutet, dass ein Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Achtung
ist eine wichtige Information über das Produkt, die Handhabung des Produktes oder den jeweiligen Teil der Dokumentation, auf den besonders aufmerksam gemacht werden soll.

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 Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Bestimmungsgemäßer Gebrauch

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.

Marken

SIMATIC®, SIMATIC HMI® und SIMATIC NET® sind 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 kann.

Copyright Siemens AG 2005 All rights reserved
Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten, insbesondere für den Fall der Patenterteilung oder GM-Eintragung

Haftungsausschluss
Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Siemens AG
Bereich Automation and Drives
Geschäftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nürnberg

Siemens AG 2005
Technische Änderungen bleiben vorbehalten

Siemens Aktiengesellschaft

A5E00388167-01

Vorwort

Zweck des Handbuchs

Dieses Handbuch gibt Ihnen einen vollständigen Überblick über die Funktionen von PRODAVE MPI/IE V6.0.

Es richtet sich an Personen, die in den Bereichen Projektierung, Inbetriebsetzung und Service von Automatisierungssystemen tätig sind.

Erforderliche Grundkenntnisse

Zum Verständnis des Handbuchs sind allgemeine Kenntnisse auf dem Gebiet der Automatisierungstechnik erforderlich.

Außerdem werden Kenntnisse über die Verwendung von Computern oder PC-ähnlichen Arbeitsmitteln (z. B. Programmiergeräten) unter dem Betriebssystem Windows 2000 bzw. XP vorausgesetzt.

Gültigkeitsbereich des Handbuchs

Das Handbuch ist gültig für das Softwarepaket PRODAVE MPI/IE V6.0.

Weitere Unterstützung

Bei Fragen zur Nutzung der im Handbuch beschriebenen Produkte, die Sie hier nicht beantwortet finden, wenden Sie sich bitte an Ihren Siemens-Ansprechpartner in den für Sie zuständigen Vertretungen und Geschäftsstellen.

Ihren Ansprechpartner finden Sie unter:

http://www.siemens.com/automation/partner

Den Wegweiser zum Angebot an technischen Dokumentationen für die einzelnen SIMATIC Produkte und Systeme finden Sie unter:

http://www.siemens.de/simatic-tech-doku-portal

Den Online-Katalog und das Online-Bestellsystem finden Sie unter:

http://mall.automation.siemens.com/

Trainingscenter

Um Ihnen den Einstieg in das Automatisierungssytem S7 zu erleichtern, bieten wir entsprechende Kurse an. Wenden Sie sich bitte an Ihr regionales Trainingscenter oder an das zentrale Trainingscenter in D 90327 Nürnberg.

Telefon: +49 (911) 895-3200.

Internet: http://www.sitrain.com

A&D Technical Support

Sie erreichen den Technical Support für alle A&D-Produkte

- Über das Web-Formular für den Support Request
  http://www.siemens.de/automation/support-request
- Telefon: + 49 180 5050 222
- Fax: + 49 180 5050 223

Weitere Informationen zu unserem Technical Support finden Sie im Internet unter
http://www.siemens.com/automation/service

Service & Support im Internet

Zusätzlich zu unserem Dokumentations-Angebot bieten wir Ihnen im Internet unser komplettes Wissen online an.

http://www.siemens.com/automation/service&support

Dort finden Sie:

- den Newsletter, der Sie ständig mit den aktuellsten Informationen zu Ihren Produkten versorgt.
- die für Sie richtigen Dokumente über unsere Suche in Service & Support.
- ein Forum in welchem Anwender und Spezialisten weltweit Erfahrungen austauschen.
- Ihren Ansprechpartner für Automation & Drives vor Ort.
- Informationen über Vor-Ort Service, Reparaturen, Ersatzteile. Vieles mehr steht für Sie unter dem Begriff "Leistungen“ bereit.

Inhaltsverzeichnis

1 Einführung

PRODAVE MPI/IE V6.0 stellt Funktionen zur Verfügung, die man in einer Anwender-Applikation kombinieren kann. Die Kombination der Tools erfolgt in den Entwicklungsumgebungen der Programmiersprachen C und Basic.

Über diese Funktionen baut PRODAVE MPI/IE V6.0 den Prozessdatenverkehr über die MPI-Profibus- oder Ethernet-Anschaltung des SIMATIC AS zwischen Automatisierungssystem und PG/PC auf. PRODAVE MPI/IE V6.0 greift bei der Kommunikation auf Funktionen der Komponente S7SDD zu. Die Parametrierung der Anschaltungen erfolgt mit dem Programm "PG/PC-Schnittstelle einstellen" des S7SDD.

Folgende wesentlichen Funktionserweiterungen wurden in PRODAVE MPI/IE V6.0 eingebracht:

- Unterstützung von Standard Ethernet-Karten auf dem PC und allen relevante Ethernet-CPs der SIMATIC S7 (CPU 31x PN, CP343 und CP443 in allen Varianten Standard und Advanced).
- Doppelwort-Unterstützung beim Lesen/Schreiben von Datenbausteinen über Industrial Ethernet (bisher nur bit-, byte-, wort- und blockweise Unterstützung).
- Unterstützung von "Advanced Costumers" mit neuen Funktionen über Industrial Ethernet: Lesen von Bausteinlisten/Headerinformationen für die Diagnose(FCs/FBs) und Statusbilder (OBs/FCs/FBs/DBs), Lesen von Codebausteinen für die Diagnose und Auslesen der Daten des Diagnosepuffers und der Systemzustandslisten (SZL).

Unterstützt werden alle SIMATIC-Automatisierungssysteme:

- S7 400
- S7 300
- S7 200

PRODAVE MPI/IE V6.0 unterstützt nur die Betriebssysteme Windows 2000, Windows XP Home, sowie Windows XP Professional. Zusätzlich werden die jeweils aktuellen Service Packs unterstützt.

Die Dienste über Industrial Ethernet werden nur für Windows 2000 und Windows XP freigegeben!

PRODAVE MPI/IE V6.0 unterstützt weder Fast-User-Switching noch den Terminal-Server-Betrieb!

2 Installation

2.1 Automation License Manager

2.1.1 Nutzungsberechtigung durch den Automation License Manager

Automation Licence Manager

Für die Nutzung von PRODAVE MPI/IE V6.0 wird ein produktspezifischer License Key (Nutzungsberechtigung) benötigt, dessen Installation ab der VV6.0 von PRODAVE MPI/IE V6.0 mit dem Automation License Manager durchgeführt wird.

Der Automation License Manager ist ein Software-Produkt der Siemens AG. Er wird systemübergreifend zur Handhabung von License Keys (technische Repräsentanten von Lizenzen) eingesetzt.

Den Automation License Manager finden Sie:

- auf dem Installationsmedium des jeweiligen Software-Produktes, für den ein License Key benötigt wird, oder
- auf einem separaten Installationsmedium sowie
- auf den Internetseiten des A&D Customer Support der Siemens AG als WebDownLoad.

Im Automation License Manager ist eine Online-Hilfe integriert, die Sie nach der Installation kontextsensitiv über die F1-Taste oder über den Menübefehl **Hilfe > Hilfe zum License Manager** aufrufen können. In dieser Hilfe erhalten Sie detaillierte Informationen zur Funktionalität und Handhabung des Automation License Managers.

Lizenzen

Für die Nutzung von lizenzrechtlich geschützten Programmpaketen von PRODAVE MPI/IE V6.0 werden Lizenzen benötigt. Eine Lizenz wird als Recht zur Nutzung von Produkten vergeben. Die Repräsentanten dieses Rechtes sind:

- das CoL (**C**ertificate **o**f **L**icense) und
- der License Key.

Certificate of License (CoL)

Das im Lieferumfang der jeweiligen Produkte enthaltene "Certificate of License" ist der juristische Nachweis des Nutzungsrechtes. Das jeweilige Produkt darf nur durch den Besitzer des CoL oder beauftragte Personen genutzt werden.

License Keys

Der License Key ist der technische Repräsentant einer Lizenz (elektronischer Lizenzstempel).

Für jede Software, die lizenzrechtlich geschützt ist, wird von der SIEMENS AG ein License Key vergeben. Erst wenn nach dem Starten der Software auf einem Rechner das Vorhandensein eines gültigen License Keys festgestellt wurde, kann die jeweilige Software entsprechend der mit diesem License Key verbundenen Lizenz- und Nutzungsbedingungen genutzt werden.

Hinweise

- Sie können PRODAVE MPI/IE V6.0 zum kurzen kennen lernen von Bedienoberfläche und Funktionsumfang auch ohne License Key verwenden.
- Die uneingeschränkte Nutzung unter Berücksichtigung der lizenzrechtlichen Vereinbarungen ist jedoch nur mit installiertem License Key zulässig und möglich.
- Wenn Sie den License Key **nicht** installiert haben, werden Sie in regelmäßigen Abständen aufgefordert, die Installation vorzunehmen.

License Keys können wie folgt abgelegt sein und zwischen den einzelnen Speichermedien transferiert werden:

- auf License Key-Disketten,
- auf lokalen Festplattenspeichern und
- auf Festplattenspeichern von Rechnern im Netzwerk.

Wenn Software-Produkte installiert sind, für die kein License Key verfügbar ist, so können Sie den License Key-Bedarf ermitteln und anschließend die notwendigen Lizenzen bestellen.

Weiterführende Informationen über die Handhabung von License Keys entnehmen Sie bitte der Online-Hilfe zum Automation License Manager.

Lizenz-Typen

Für Software-Produkte der Siemens AG wird zwischen folgenden anwendungsorientierten Lizenz-Typen unterschieden. Das Verhalten der Software wird durch die für diese Lizenz-Typen unterschiedlichen License Keys gesteuert. Die Art der Nutzung ergibt sich aus dem jeweiligen Certificate of License.

Lizenz-Typ	**Beschreibung**
Single License	Die Nutzung der Software ist zeitlich unbegrenzt auf einem beliebigen Rechner zulässig.
Floating License	Zeitlich unbegrenzte, auf Nutzung über ein Netzwerk bezogenes Nutzungsrecht ("remote"Nutzung) einer Software.
Trial License	Die Nutzung der Software ist beschränkt auf:· • eine Gültigkeit von maximal 14 Tagen, • eine bestimmte Anzahl von Tagen ab der Erstnutzung, • die Nutzung für Tests und zur Validierung (Haftungsausschluss).
Upgrade License	Für ein Upgrade können spezifische Anforderungen an den Systemzustand gefordert sein:· • Mit einer Upgrade License kann eine Lizenz einer "Alt-"Version x auf eine Version >x+... umgestellt werden. • Ein Upgrade kann z. B. durch eine Erweiterung des Mengengerüsts notwendig sein.

2.1.2 Installieren des Automation License Managers

Der Automation License Manager wird über ein MSI-Setup installiert. Die Installations-Software für den Automation License Manager finden Sie auf der PRODAVE MPI/IE V6.0-Produkt-CD.

Sie können den Automation License Manager im Zusammenhang mit PRODAVE MPI/IE V6.0 oder erst zu einem späteren Zeitpunkt installieren.

Hinweise

- Detaillierte Informationen zur Vorgehensweise beim Installieren des Automation License Managers entnehmen Sie bitte der aktuellen Liesmich.wri.
- In der Online-Hilfe zum Automation License Manager erhalten Sie alle benötigten Informationen zur Funktionalität und Handhabung von License Keys.

License Keys später installieren

Wenn Sie die PRODAVE MPI/IE V6.0-Software starten und keine License Keys vorhanden sind, so erhalten Sie eine entsprechende Meldung.

Hinweise

- Sie können PRODAVE MPI/IE V6.0 zum kurzen kennen lernen von Bedienoberfläche und Funktionsumfang auch ohne License Key verwenden.
- Die uneingeschränkte Nutzung unter Berücksichtigung der lizenzrechtlichen Vereinbarungen ist jedoch nur mit installiertem License Key zulässig und möglich.
- Wenn Sie den License Key **nicht** installiert haben, werden Sie in regelmäßigen Abständen aufgefordert, die Installation vorzunehmen.

Zum nachträglichen Installieren von License Keys haben Sie folgende Möglichkeiten:

- Installieren der License Keys von Disketten
- Installieren der License Keys über WebDownLoad (vorherige Bestellung erforderlich)
- Nutzung von im Netzwerk vorhandenen Floating License Keys.

Detaillierte Informationen zur Vorgehensweise entnehmen Sie bitte der Online-Hilfe zum Automation License Manager, die Sie nach der Installation kontextsensitiv über die F1-Taste oder über den Menübefehl **Hilfe > Hilfe zum License Manager** aufrufen können.

Hinweise

- License Keys sind unter Windows 2000/XP nur dann funktionsfähig, wenn sie auf einem Festplattenlaufwerk liegen, auf dem schreibende Zugriffe zugelassen sind.
- Floating Licenses können auch über ein Netzwerk, also "remote" genutzt werden.

2.1.3 Regeln für den Umgang mit License Keys

Vorsicht

Beachten Sie die Hinweise zum Umgang mit License Keys, die in der Online-Hilfe zum Automation License Manager beschrieben sind und in der PRODAVE MPI/IE V6.0-Liesmich.wri auf der CD-ROM. Bei Nichtbeachtung besteht die Gefahr, dass License Keys unwiderruflich verloren gehen.

Die Online-Hilfe zum Automation License Manager können Sie kontextsensitiv über die F1-Taste oder über den Menübefehl **Hilfe > Hilfe zum Automation License Manager** aufrufen.

In dieser Hilfe erhalten Sie alle benötigten Informationen zur Funktionalität und Handhabung von License Keys.

2.2 Installieren von PRODAVE MPI/IE V6.0

PRODAVE MPI/IE V6.0 enthält ein Setup-Programm, das die Installation automatisch durchführt. Eingabeaufforderungen auf dem Bildschirm führen Sie Schritt für Schritt durch den gesamten Installationsvorgang. Es wird mit der unter Windows 2000/XP üblichen Standardprozedur zur Installation von Software aufgerufen.

Die wesentlichen Phasen der Installation sind:

- das Kopieren der Daten auf Ihr Erstellsystem,
- das Einrichten der Treiber für die Kommunikation,
- das Eintragen der Ident-Nr.
- das Installieren der License Keys (falls gewünscht).

Installationsvoraussetzungen

- Betriebssystem
  Microsoft Windows 2000, Windows XP.
- Basishardware
  PC oder Programmiergerät mit
  - Pentium-Prozessor (bei Windows 2000 P 233, bei Windows XP P 333),
  - RAM-Speicherausbau: mindestens 256 MB.
  - Farbmonitor, Tastatur und Maus, die von Microsoft Windows unterstützt werden.

Ein Programmiergerät (PG) ist ein Personal Computer in spezieller industrietauglicher und kompakter Ausführung. Es ist komplett ausgestattet für die Programmierung der SIMATIC-Automatisierungssysteme.

- Speicherkapazität
  Erforderlicher Speicherplatz auf der Festplatte siehe LIESMICH.WRI.
- MPI-Schnittstelle (optional)
  Die MPI-Schnittstelle zwischen Erstellsystem (Programmiergerät oder PC) und Zielsystem ist nur erforderlich, wenn Sie unter PRODAVE MPI/IE V6.0 über MPI mit dem Zielsystem kommunizieren wollen.
  Dazu verwenden Sie entweder
  - einen PC-Adapter und ein Null-Modem-Kabel (RS232), die an die Kommunikationsschnittstelle Ihres Geräts angeschlossen werden oder
  - eine MPI-Baugruppe (z. B. CP 5611), die in Ihrem Gerät installiert wird.

Bei Programmiergeräten ist die MPI-Schnittstelle bereits eingebaut.

Hinweise

Beachten Sie auch die Hinweise zum Installieren von PRODAVE MPI/IE V6.0 in der Datei LIESMICH.WRI.

Die Liesmich-Datei finden Sie in der Startleiste unter Start > Simatic > Produkt-Hinweise.

2.2.1 Vorgehen beim Installieren

Vorbereitungen

Bevor Sie mit der Installation beginnen können, muss das Betriebssystem (Windows 2000 oder XP) gestartet sein.

- Sie benötigen keine externen Datenträger, wenn sich die installierbare PRODAVE MPI/IE V6.0-Software bereits auf der Festplatte des PG befindet.
- Um von CD-ROM zu installieren, legen Sie die CD-ROM in das CD-ROM-Laufwerk ihres PC.

Starten des Installationsprogramms

Gehen Sie zur Installation wie folgt vor:

1. Legen Sie die CD-ROM ein und starten Sie das Setup durch Doppelklick auf die Datei ”setup.exe”.
2. Befolgen Sie Schritt für Schritt die Anweisungen, die Ihnen das Installationsprogramm anzeigt.

Das Programm führt Sie schrittweise durch den Installationsprozess. Sie können jeweils zum nachfolgenden oder vorhergehenden Schritt weiterschalten.

Während des Installationsvorgangs werden Ihnen in Dialogfeldern Fragen angezeigt oder Optionen zur Auswahl angeboten. Lesen Sie bitte die folgenden Hinweise, um die Dialoge schneller und leichter beantworten zu können.

Zum Installationsumfang

Zur Festlegung des Installationsumfangs haben Sie drei Auswahlmöglichkeiten:

- PRODAVE MPI 5.6: alle Funktionen der Version 5.6. Den dazu benötigten Speicherplatz entnehmen Sie bitte der aktuellen Produktinformation.
- PRODAVE MPI/IE V6.0: alle Funktionen der Version V6.0. Den dazu benötigten Speicherplatz entnehmen Sie bitte der aktuellen Produktinformation.

- PRODAVE MPI 5.6 und PRODAVE MPI/IE V6.0: alle Funktionen der Versionen 5.6 und V6.0. Den dazu benötigten Speicherplatz entnehmen Sie bitte der aktuellen Produktinformation.

Zur Ident-Nummer

Sie werden bei der Installation nach einer Ident-Nummer gefragt. Tragen Sie die Ident-Nummer ein. Sie finden diese auf dem Software-Produktschein oder auf der zugehörigen License Key-Diskette.

Zum Installieren von License Keys

Bei der Installation wird überprüft, ob ein entsprechender License Key auf der Festplatte vorhanden ist. Wird kein gültiger License Key erkannt, so erscheint ein Hinweis, dass die Software nur mit vorhandenem License Key benutzt werden kann. Wenn Sie es wünschen, können Sie gleich die License Keys installieren oder aber die Installation von PRODAVE MPI/IE V6.0 fortsetzen und die License Keys zu einem späteren Zeitpunkt nachinstallieren. Im erstgenannten Fall legen Sie die mitgelieferte License Key-Diskette ein, wenn Sie dazu aufgefordert werden.

Zur PG/PC-Schnittstelleneinstellung

Während des Installationsvorgangs wird ein Dialog zur Einstellung der PG/PC-Schnittstelle angezeigt. Lesen Sie dazu "Einstellen der PG/PC-Schnittstelle".

Fehler während der Installation

Folgende Fehler führen zum Abbruch der Installation:

- Wenn sofort nach dem Start von Setup ein Initialisierungsfehler auftritt, so wurde höchst wahrscheinlich *setup* nicht unter Windows gestartet.
- Speicherplatz reicht nicht aus: Sie benötigen abhängig vom Installationsumfang ca. 50 MB freien Speicherplatz auf Ihrer Festplatte für die Software.
- Defekte CD: Wenn Sie feststellen, dass die CD defekt ist, wenden Sie sich bitte an Ihre SIEMENS-Vertretung.
- Bedienungsfehler: Beginnen Sie die Installation erneut und beachten Sie die Anweisungen sorgfältig.

Zum Abschluss der Installation ...

Wenn die Installation erfolgreich war, wird dies durch eine entsprechende Meldung auf dem Bildschirm angezeigt.

Falls bei der Installation Systemdateien aktualisiert wurden, werden Sie aufgefordert, Windows neu zu starten. Nach dem Neustart (Warmstart) können Sie die Oberfläche von PRODAVE MPI/IE V6.0 starten.

Nach einer erfolgreichen Installation ist eine Programmgruppe für PRODAVE MPI/IE V6.0 eingerichtet.

2.2.2 Einstellen der PG/PC-Schnittstelle

Mit den hier gemachten Einstellungen legen Sie die Kommunikation zwischen PG/PC und dem Automatisierungssystem fest. Während des Installationsvorgangs wird ein Dialog zur Einstellung der PG/PC-Schnittstelle angezeigt. Sie können sich den Dialog auch nach der Installation anzeigen lassen, indem Sie das Programm ”PG/PC-Schnittstelle einstellen” in der PRODAVE MPI/IE V6.0-Programmgruppe oder der Systemsteuerung aufrufen. Dadurch ist es möglich, die Schnittstellenparameter auch später unabhängig von einer Installation zu ändern.

Prinzipielles Vorgehen

Zum Betrieb einer Schnittstelle sind erforderlich:

- Einstellungen im Betriebssystem
- eine geeignete Schnittstellenparametrierung

Wenn Sie einen PC mit MPI-Karte oder Kommunikationsprozessoren (CP) einsetzen, so sollten Sie über die "Systemsteuerung" von Windows die Interrupt- und Adressbelegung prüfen, um sicherzustellen, dass es zu keinen Interrupt-Konflikten oder Adressbereichsüberschneidungen kommt.

Unter Windows 2000 und Windows XP wird die ISA-Komponente MPI-ISA-Card nicht mehr unterstützt und nicht mehr zur Installation angeboten.

Um die Parametrierung der PG/PC-Schnittstelle zu vereinfachen, werden Ihnen in Dialogfeldern vordefinierte Sätze von Grundparametern (Schnittstellenparametrierungen) zur Auswahl angeboten.

Parametrieren der PG/PC-Schnittstelle

Gehen Sie folgendermaßen vor (ausführliche Beschreibung in der Online-Hilfe):

1. Doppelklicken Sie in der "Systemsteuerung" von Windows auf "PG/PC-Schnittstelle einstellen".
2. Stellen Sie den "Zugangspunkt der Applikation" auf "S7ONLINE".
3. Wählen Sie in der Liste "Benutzte Schnittstellenparametrierung" die gewünschte Schnittstellenparametrierung aus. Wird die von Ihnen gewünschte Schnittstellenparametrierung nicht angezeigt, müssen Sie zunächst über die Schaltfläche "Auswählen" eine Baugruppe bzw. ein Protokoll installieren. Die Schnittstellenparametrierung wird dann automatisch erstellt. Bei Plug&Play-Systemen können die Plug&Play-fähigen CPs (CP 5611 und CP 5511) nicht manuell installiert werden. Sie werden automatisch in "PG/PC-Schnittstelle einstellen" integriert, wenn Sie die Hardware in Ihrem PG/PC eingebaut haben.
   - Wenn Sie eine Schnittstelle **mit automatischer Erkennung der Busparameter** wählen (z. B. CP 5611 (Auto)), dann können Sie das PG bzw. den PC an MPI bzw. PROFIBUS anschließen, ohne Busparameter einstellen zu müssen. Bei Übertragungsgeschwindigkeiten kleiner 187,5 kBit/s können allerdings Wartezeiten bis zu einer Minute entstehen. **Voraussetzung für die automatische Erkennung**: Am Bus sind Master angeschlossen, die zyklisch Busparameter verteilen. Alle neuen MPI-Komponenten tun das; bei PROFIBUS-Subnetzen darf das zyklische

Verteilen der Busparameter nicht ausgeschaltet sein (voreingestellte PROFIBUS-Netzeinstellung).

- Wenn Sie eine Schnittstelle **ohne automatische Erkennung der Busparameter** wählen, dann lassen Sie sich die Eigenschaften anzeigen und passen sie an das Subnetz an.

Änderungen sind auch dann erforderlich, wenn sich Konflikte mit anderen Einstellungen ergeben (z. B. Interrupt- oder Adressbelegungen). Nehmen Sie in diesem Fall die entsprechenden Änderungen mit der Hardwareerkennung und der Systemsteuerung von Windows vor (siehe unten).

Vorsicht

Eine eventuell angezeigte Schnittstellenparametrierung "TCP/IP" **nicht** entfernen!

Dies könnte den Ablauf von anderen Anwendungen beeinträchtigen.

Prüfen der Interrupt- und Adressbelegung

Wenn Sie einen PC mit MPI-Karte einsetzen, so sollten Sie in jedem Fall prüfen, ob der voreingestellte Interrupt und der voreingestellte Adressbereich frei sind und ggf. einen freien Interrupt und/oder Adressbereich wählen.

Windows 2000

Unter Windows 2000 können Sie sich die Ressourcen

- unter **Systemsteuerung > Verwaltung > Computerverwaltung > System > Systeminformationen > Hardwareressourcen** anzeigen lassen.
- unter **Systemsteuerung > Verwaltung > Computerverwaltung > System > Geräte-Manager > SIMATIC NET > CP-Name > Eigenschaften > Ressourcen** ändern.

Windows XP

Unter Windows XP können Sie sich die Ressourcen

- unter **START > Alle Programme > Zubehör > System > Systemprogramme > Systeminformationen > Hardwareressourcen** anzeigen lassen.
- unter Systemsteuerung > Arbeitsplatz > Eigenschaften > Geräte-Manager > SIMATIC NET > CP-Name > Eigenschaften > Ressourcen ändern.

2.3 Deinstallieren von PRODAVE MPI/IE V6.0

2.3.1 Deinstallieren von PRODAVE MPI/IE V6.0

Benutzen Sie das unter Windows übliche Verfahren zur Deinstallation:

1. Starten Sie unter Windows den Dialog zu Installation von Software durch Doppelklick auf das Symbol "Software" in ”Systemsteuerung”.
2. Markieren Sie den PRODAVE MPI/IE V6.0-Eintrag in der angezeigten Liste der installierten Software. Betätigen Sie dann die Schaltfläche zum "Entfernen" der Software.
3. Falls Dialogfelder "Freigegebene Datei entfernen" erscheinen, so klicken Sie im Zweifelsfall auf die Schaltfläche "Nein":

3 Neue Funktionen ab PRODAVE MPI/IE V6.0

Nachfolgende Tabelle zeigt eine Auflistung aller neuen Funktionen:

Version	PRODAVE MPI/IE V6.0	Beschreibung
Library	PRODAVE6.DLL	
Header	PRODAVE6.H	
Grundfunktionen		
LoadConnection_ex6	x	MPI/Profibus, ISO- oder TCP/IP-Protokoll
UnloadConnection_ex6	x	MPI/Profibus, ISO- oder TCP/IP-Protokoll
SetActiveConnection_ex6	x	MPI/Profibus, ISO- oder TCP/IP-Protokoll
SetPassword_ex6	x	
UnSetPassword_ex6	x	
Funktionen zum Datenverkehr zur S7 300/400		
as_info_ex6	x	
as_zustand_ex6	x	
db_buch_ex6	x	
db_read_ex6	x	erweitert um Doppelworte
db_write_ex6	x	erweitert um Doppelworte
bst_read_ex6	x	"Advanced" Bausteincode auslesen
bst_read_diag_ex6	x	"Advanced" Bausteinlisten/Headerinfo
bst_read_stat_ex6	x	"Advanced" Bausteinlisten/Headerinfo
read_diag_buf_ex6	x	"Advanced" Diagnosepuffer auslesen
field_read_ex6	x	Zusammenfassung aller _field_read
field_write_ex6	x	Zusammenfassung aller _field_write
mb_setbit_ex6	x	Setzten eines Bits
mb_bittest_ex6	x	

Funktionen zum Datenverkehr zur S7 200		
as200_as_info_ex6	x	
as200_as_zustand_ex6	x	
as200_field_read_ex6	x	Zusammenfassung aller _field_read
as200_field_write_ex6	x	Zusammenfassung aller _field_write
as200_mb_setbit_ex6	x	
as200_mb_bittest_ex6	x	
Komfortfunktionen		
GetErrorMessage_ex6	x	
kg_2_float_ex6	x	
float_2_kg_ex6	x	
gp_2_float_ex6	x	
float_2_gp_ex6	x	
testbit_ex6	x	
byte_2_bool_ex6	x	
bool_2_byte_ex6	x	
kf_2_integer_ex6	x	
kf_2_long_ex6	x	erweitert um Doppelworte
swab_buffer_ex6	x	
copy_buffer_ex6	x	
ushort_2_bcd_ex6	x	
bcd_2_ushort_ex6	x	
ulong_2_bcd_ex6	x	erweitert um Doppelworte
bcd_2_ulong_ex6	x	erweitert um Doppelworte
GetLoadedConnections_ex6	x	
TeleService-Funktionen		
ts_dial_ex6	x	
ts_hang_up_dial_ex6	x	
ts_set_ringindicator_ex6	x	
ts_read_info_ex6	x	
ts_hang_up_ring_ex6	x	
ts_get_modem_name_ex6	x	

Funktionen

Alle aufgeführten Funktionen sind aus PRODAVE MPI/IE V6.0-Anwendersicht beschrieben. Allerdings müssen Sie zusätzlich noch zu jeder Funktion die Aufrufkonvention WINAPI hinzufügen:

int **WINAPI** load_tool (char *chConNo*, …).

Die alten PRODAVE MPI 5.6 Libraries (W95_S7.DLL, KOMFORT.DLL) und die neue PRODAVE MPI/IE V6.0 Library (PRODAVE6.LIB) können parallel in einem Anwenderprogramm verwendet werden.

Alte Prodave V5.6 Programme können weiterhin benutzt werden. Allerdings, ist eine Verschränkung der beiden Versionen nicht möglich, d. h. es ist nicht möglich mit einer alten Funktion eine Verbindung zum AS herzustellen, und dann mit einer neuen Funktion Daten zu lesen und umgekehrt. Die Verbindung und das Lesen oder Schreiben der Daten muss homogen, mit der selben Prodave Version, geschehen.

3.1 Grundfunktionen

3.1.1 LoadConnection_ex6

Die Grundfunktion **LoadConnection_ex6** initialisiert den Adapter, prüft ob der Treiber geladen ist, initialisiert die parametrierten Adressen und schaltet die angewählte Schnittstelle aktiv.

Mit **LoadConnection_ex6** werden Transportverbindung via MPI/PB- oder IP-Adressen (TCP/IP-Protokoll) aufgebaut.

int LoadConnection_ex6 (int *ConNr*, char* *pAccessPoint*, int *ConTableLen,* CON_TABLE_TYPE * *pConTable*);

Parameter

ConNr

[in] Nummer der Verbindung (maximal 64 Verbindungen).

pAccessPoint

[in] Zugangspunkt (nullterminiert) des verwendeten Treibers, z. B. "S7ONLINE" für den MPI-Treiber oder 0 (default).

ConTableLen

[in] Länge der vom Anwender bereitgestellten Verbindungstabelle in Byte

pConTable

[in] Zeiger auf Adressliste der angeschlossenen Teilnehmer; 'Adr == 0' wird als Endekennung der Liste gewertet.

```
#pragma pack(1)

typedef union {
    unsigned char Mpi;          // MPI/PB-Stationsadresse (2)
    unsigned char Ip[4];        // IP-Adresse (192.168.0.1)
    unsigned char Mac[6];       // MAC-Adresse (08-00-06-01-AA-BB)
    } CON_ADR_TYPE;

typedef struct {
    CON_ADR_TYPE Adr;           // Verbindungsadresse
    unsigned char AdrType;      // Typ der Adresse: MPI/PB (1), IP (2), MAC (3)
    unsigned char SlotNr;       // Steckplatznummer
```

```
    unsigned char RackNr;          // Nummer des Baugruppenträgers
    } CON_TABLE_TYPE;

#pragma pack()
```

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0.

Sollte ein Fehler auftreten, wird eine Fehlernummer zurückgegeben, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Nach Behebung der Ursache des Fehlers, muss **LoadConnection_ex6** erneut ausgeführt werden, damit eine Verbindung zum AS erstellt wird.

Aufrufbeispiel

Für jede Verbindung wird ein neuer Tabelleneintrag in der Verbindungstabelle angelegt.

```
unsigned short ConNr = 0;                          // erste Verbindung
                                                   (0 ... 63)
char AccessPoint[] = {"S7ONLINE"};                 // Default Zugangspunkt
                                                   S7ONLINE
CON_TABLE_TYPE                                     // Verbindungstabelle
ConTable[MAX_CONNECTION+1] = {0};
CON_TABLE_TYPE * pConTable = ConTable;             // Zeiger auf
                                                   Verbindungstabelle
unsigned short ConTableLen = sizeof(ConTable);     // Länge der
                                                   Verbindungstabelle
int RetValue;
pConTable[ConNo].Adr.Ip[0] = 192;                  // Default IP-Adresse
                                                   (192.168.0.1)
pConTable[ConNo].Adr.Ip[1] = 168;
pConTable[ConNo].Adr.Ip[2] = 0;
pConTable[ConNo].Adr.Ip[3] = 1;
pConTable[ConNo].AdrType = 2;                      // Adresstyp = IP
pConTable[ConNo].SlotNr = 2;                       // Steckplatznummer = 2
pConTable[ConNo].RackNr = 0;                       // Baugruppenträgernummer
                                                   = 0
RetValue = LoadConnection_ex6 (ConNr, &AccessPoint, ConTableLen,
pConTable);
```

Bei Verbindungsfehlern während des Betriebs müssen Sie die Verbindung schließen und neu laden.

Voraussetzungen

	V5.6 load_tool, load_tool_ex	**V6.0 Load_Connection_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		**PRODAVE6.DLL**

Siehe auch:

UnloadConnection_ex6, **SetActiveConnection_ex6**

3.1.2 UnloadConnection_ex6

Die Grundfunktion **UnloadConnection_ex6** deinitialisiert die Verbindungen und den Adapter und muss vor dem Beenden der Applikation aufgerufen werden.

int UnloadConnection_ex6 (unsigned short *ConNr*);

Parameter

ConNr

[in] Nummer der Verbindung, die mit **LoadConnection_ex6** vergeben wurde.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Falls die gerade aktive Verbindung geschlossen wurde, wird eine Warnung (ein Fehler) zurückgegeben.

Aufrufbeispiel

Muss für jede Verbindung in der Verbindungstabelle *ConTable* aufgerufen werden.

```
unsigned short ConNr = 0;                    // erste Verbindung (0 ... 63)
int RetValue;
RetVal = UnloadConnection_ex6 (ConNr);
```

Voraussetzungen

	V5.6 unload_tool	**V6.0 Unload_Connection_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

LoadConnection_ex6, **SetActiveConnection_ex6**

3.1.3 SetActiveConnection_ex6

Die Grundfunktion **SetActiveConnection_ex6** schaltet die Verbindung des PG/PC aktiv, über welche der weitere Datenaustausch stattfinden soll.

int SetActiveConnection_ex6 (unsigned short *ConNr*);

Parameter

ConNr

[in] Nummer einer mit **LoadConnection_ex6** vergebenen Verbindung, die aktiviert werden soll.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned short ConNr = 7;                    // (0 ... 63)
int RetValue;
RetVal = SetActiveConnection_ex6(ConNr);
```

Voraussetzungen

	V5.6 new_ss	**V6.0 SetActiveConnection_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

LoadConnection_ex6, **UnloadConnection_ex6**

3.1.4 SetPassword_ex6

Die Funktion ***SetPassword_ex6*** setzt für eine bestimmte Verbindung mit einer CPU ein Passwort.

int SetPassword_ex6 (unsigned short *ConNr, unsigned char * Password*);

Parameter

ConNr

[in] Verbindungsnummer

Password

[in] Zeiger auf einen UCHAR-String von der Länge 8

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

Falls eine CPU mit einem Passwort gesichert wurde, ist ein Passwort nötig, um Veränderungen am Zustand der CPU durchzuführen. Die Funktion **SetPassword_ex6** muss dazu nach **LoadConnection_ex6** aufgerufen werden.

```
unsigned char passw[8];          // Passwort
unsigned short ConNr = 13;       // Verbindungsnummer
int RetValue;


RetValue = SetPassword_ex6 (ConNr, &passw[0]);
```

Voraussetzungen

	V5.6 new_ss	**V6.0 SetPassword_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

As_zustand_ex6

3.1.5 UnSetPassword_ex6

Die Funktion **Un*SetPassword_ex6*** entfernt das Passwort für eine bestimmte Verbindung.

int UnSetPassword_ex6 (unsigned short *ConNr*);

Parameter

ConNr

[in] Verbindungsnummer

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned short ConNr= 13;       // Verbindungsnummer
int RetValue;


RetValue = UnSetPassword_ex6 (ConNr);
```

Voraussetzungen

	V5.6 new_ss	**V6.0 UnSetPassword_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

As_zustand_ex6

3.1.6 as_info_ex6

Die Funktion **as_info_ex6** liest den Ausgabestand der AS-Software und der PG-Anschaltung, sowie die MLFB-Nummer des AS aus und legt diese als ASCII-String nullterminiert im Übergabepuffer des PG/PC ab.

int as_info_ex6 (unsigned long BufLen, AS_INFO_TYPE * *pInfoBuffer*, unsigned long * *pDatLen*);

Parameter

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pInfoBuffer

[out] Zeiger auf Übergabepuffer mit den zu liefernden AS-Informationen:

```
typedef struct {
  unsigned char Plcas[4];      // Ausgabestand der Hardware
  unsigned char Pgas[2];       // Ausgabestand der Firmware
  char Mlfb[20];               // MLFB des angeschlossenen AS
} AS_INFO_TYPE;
```

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Einige AS können die MLFB oder den FW- und den HW-Ausgabestand nicht ausgeben. In diesem Fall wird die MLFB, falls diese gelesen werden konnte, in *pInfoBuffer* geschrieben, und eine entsprechende Fehlermeldung wird ausgegeben.

Aufrufbeispiel

```
AS_INFO_TYPE Buffer;                    // Übergabepuffer anlegen
AS_INFO_TYPE * pInfoBuffer = &Buffer;   // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);  // Länge des Übergabepuffers
unsigned long DatLen;                   // Datenlänge
int RetValue;
RetValue = as_info_ex6 (BufLen, pInfoBuffer, &DatLen);
```

Voraussetzungen

	V5.6 ag_info	**V6.0 as_info_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as_zustand_ex6

3.1.7 as_zustand_ex6

Die Funktion **as_zustand_ex6** liest den AS-Zustand (RUN oder STOP) und legt die Daten in einem Speicherbereich des PG/PC ab.

int as_zustand_ex6 (unsigned char * *pState*);

Parameter

pState

[out] Übergabepuffer mit den zu liefernden AS-Zuständen (benötigte Länge 1 Byte)

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned char State;                    // Übergabebyte für den zu liefernden
                                        AS-Zustand

int RetValue;

RetValue = as_zustand_ex6 (&State);
```

"Aktueller Betriebszustand" (Werte in Hex)

BZ-ID	Bedeutung
00	Configuration in RUN (EVAL bei R-KiR)
01	STOP (update)
02	STOP (Urlöschen)
03	STOP (Eigeninitialisierung)
04	STOP (intern)
05	Anlauf (Kaltstart)
06	Anlauf (Warmstart)
07	Anlauf (Wiederanlauf)
08	RUN (Solo)
09	RUN (Redundant)
0A	HALT
Bx	ANKOPPELN
Cx	AUFDATEN
0D	Defekt
Ex	TEST
Fx	NOPOWER

Voraussetzungen

	V5.6 ag_zustand	V6.0 as_zustand_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as_info_ex6

3.1.8 db_buch_ex6

Die Funktion **db_buch_ex6** prüft, welche Datenbausteine im AS vorhanden sind.

int db_buch_ex6 (unsigned long BufLen, unsigned short * *pBuchBuffer*, unsigned long * *pDatLen*);

Parameter

BufLen

[in] Länge des von Ihnen bereitgestellten Übergabepuffers in Byte

pBuchBuffer

[out] Zeiger auf Übergabepuffer für die gelesenen Datenbausteine. Die Nummern der vorhandenen Datenbausteine werden als Wortwerte (Bausteinnummer kann 0 sein!) in das Puffer-Array geschrieben.

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Hat **pDatLen** den Wert 0, wurde kein Datenbaustein gefunden.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned short Buffer[MAX_BUFFER];      // Übergabepuffer
unsigned short * pBuchBuffer = Buffer;  // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);  // Länge des Übergabepuffers
unsigned long DatLen;                   // Datenlänge
int RetValue;
RetValue = db_buch_ex6 (BufLen, pBuchBuffer, &DatLen);
```

Voraussetzungen

	V5.6 db_buch	**V6.0 db_buch_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.1.9 db_read_ex6

Die Funktion **db_read_ex6** liest eine Anzahl Datenworte aus einem Datenbaustein im AS aus und transferiert diese in einen Speicherbereich (Puffer) des PG/PC.

Mit **db_read_ex6** können Datenworte wortweise/doppelwortweise (*pAmount* = 1) oder blockweise (*pAmount* > 1) ausgelesen werden.

int db_read_ex6 (unsigned short *BlkNr*, unsigned char *DatType*, unsigned short *StartNr*, unsigned long * p*Amount*, unsigned long BufLen, unsigned char * *pReadBuffer*, unsigned long * *pDatLen*)

Parameter

BlkNr

[in] Nummer des Datenbausteins

DatType

[in] Typ der zu lesenden Daten:

0x02 = BYTE, 0x04 = WORD, 0x06 = DWORD default: *DatType* = 0x02

StartNr

[in] Anfangsnummer des ersten zu lesenden Datenwortes/-doppelwortes

pAmount

[in/out] Anzahl der zu lesenden Datenbytes/-worte/-doppelworte

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pReadBuffer

[out] Zeiger auf Übergabepuffer für die gelesenen Datenworte/-doppelworte

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Existiert der Datenbaustein nicht, so wird dies durch einen Rückgabewert = Fehlernummer angegeben.

Sollen mehr Daten gelesen werden als im Datenbaustein vorhanden sind, so werden nur die tatsächlich gelesenen Daten in *pReadBuffer* geschrieben, die Anzahl (in Bytes) der tatsächlich gelesenen Daten wird in *pAmount* geschrieben und eine Fehlermeldung "Bausteingrenze überschritten" zurückgegeben.

Aufrufbeispiel

```
unsigned short BlkNr = 0;                    // Bausteinnummer = 0
unsigned char DatType = 0x04;                // Datentyp = Wort
unsigned short StartNr = 2;                  // Erstes zu lesendes Wort = 2
int Amount = 6;                              // Anzahl zu lesende Worte = 6
unsigned char Buffer[MAX_BUFFER];            // Übergabepuffer
unsigned char * pReadBuffer = Buffer;        // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);       // Länge des Übergabepuffers
unsigned long DatLen;                        // Datenlänge
int RetValue;
RetValue = db_read_ex6 (BlkNr, DatType, StartNr, &Amount, BufLen,
pReadBuffer, &DatLen);

unsigned short * wBuffer = (unsigned         // wortweiser Zugriff
short *)Buffer;
unsigned long * dwBuffer = (unsigned         // doppelwortweiser Zugriff
long *)Buffer;
```

Achtung

Beim Zugriff auf einen Datenbaustein sind die Datenworte im Puffer nicht nach Intel-Notation (low Byte - high Byte), sondern in STEP5-Notation (high Byte - low Byte) abgelegt. Bei einer Weiterverarbeitung der Daten muss dies beachtet werden. Zum Tauschen von Bytes stehen die Funktionen **kf_2_integer_ex6, kf_2_long_ex6** und **swab_buffer_ex6** zur Verfügung.

Voraussetzungen

	V5.6 db_read	**V6.0 db_read_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

db_write_ex6, kf_2_integer, kf_2_long_ex6, swab_buffer_ex6

3.1.10 db_write_ex6

Die Funktion **db_write_ex6** schreibt eine Anzahl Datenworte, die in einem definierten Speicherbereich (Puffer) im PG/PC liegen, in einen Datenbaustein im AS.

Mit **db_write_ex6** können Datenworte wortweise/woppelwortweise (*pAmount* = 1) oder blockweise (*pAmount* > 1) geschrieben werden.

int db_write_ex6 (unsigned short *BlkNr*, unsigned char *DatType*, unsigned short *StartNr*, unsigned long * p*Amount*, unsigned long BufLen, unsigned char * *pWriteBuffer*);

Parameter

BlkNr

[in] Nummer des Datenbausteins

DatType

[in] Typ der zu schreibenden Daten:

0x02 = BYTE, 0x04 = WORD, 0x06 = DWORD default: *DatType* = 0x02

StartNr

[in] Anfangsnummer des ersten zu schreibenden Datenwortes/-doppelwortes

pAmount

[in/out] Anzahl der zu schreibenden Datenworte/-doppelworte

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pWriteBuffer

[out] Zeiger auf Übergabepuffer für die geschriebenen Daten-Worte/Doppelworte

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Existiert der Datenbaustein nicht, so wird dies durch einen Rückgabewert = Fehlernummer angegeben.

Sollen mehr Daten geschrieben werden als Platz im Datenbaustein vorhanden ist, so wird eine entsprechene Fehlernummer zurückgegeben, die Anzahl (in Bytes) der möglichen Daten wird in *pAmount* geschrieben und eine Fehlermeldung "Bausteingrenze überschritten" zurückgegeben.

Aufrufbeispiel

```
unsigned short BlkNr = 0;                    // Bausteinnummer = 0
unsigned char DatType = 0x04;                // Datentyp = Wort
unsigned short StartNr = 2;                  // Erstes zu schreibendes Wort = 2
int Amount = 6;                              // Anzahl zu schreibende Worte = 6
unsigned char pWriteBuffer
[MAX_BUFFER];                                // Übergabepuffer
unsigned char *pBuffer = Buffer;             // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);       // Länge des Übergabepuffers
int RetValue;
RetValue = db_write_ex6 (BlkNr, DatType, StartNr, &Amount, BufLen,
pWriteBuffer);

unsigned short * wBuffer = (unsigned         // wortweiser Zugriff
short *)Buffer;
unsigned long * dwBuffer = (unsigned         // doppelwortweiser Zugriff
long *)Buffer;
```

Voraussetzungen

	V5.6 db_write	**V6.0 db_write_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

db_read_ex6, kf_2_integer, kf_2_long_ex6, swab_buffer_ex6

3.1.11 bst_read_diag_ex6

Die Funktion **bst_read_diag_ex6** liest Bausteinlisten und Headerinformationen für die Diagnose (FCs/FBs).

Die Informationen werden in einer Struktur übergeben und beinhalten, welche Bausteine vorhanden sind und den Zeitstempel jedes Bausteins, um geänderte Bausteine zu erkennen (Online/Offline-Vergleich).

int bst_read_diag_ex6 (int B*lkType,* unsigned short *StartNr*, unsigned long * p*Amount*, unsigned long BufLen, BST_DIAG_TYPE * *pDiagBuffer*, unsigned long * *pDatLen*);

Sollen alle Bausteine gelesen werden, muss StartNr und pAmount gleich 0 gesetzt werden. Nach dem Ausführen der Funktion steht in pAmount die Anzahl der gelesenen Bausteine.

Wird der Baustein mit der Startnummer (StartNr) nicht gefunden, wird der nächst höhere Baustein als Startbaustein verwendet.

Parameter

BlkType

[in] Typ der zu lesenden Bausteine:

BlkType: 0x08 = OB, 0x0a = DB, 0x0c = FC, 0x0e = FB default: BlkTyp = 0x0a

StartNr

[in] Anfangsnummer des ersten zu lesenden Bausteins

Amount

[in/out] Anzahl der zu lesenden Bausteine

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pDiagBuffer

[out] Zeiger auf Übergabepuffer für die gelesenen Daten

```
typedef struct {
  unsigned short BlkNr;          // Bausteinnummer
  struct tm BlkTimestamp;        // Zeitstempel zu der Bausteinnummer
} BST_DIAG_TYPE;
```

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Sollen mehr Bausteine gelesen werden als im AS vorhanden sind, so werden nur die tatsächlich gelesenen Diagnoseinformationen in *pDiagBuffer* geschrieben, die Anzahl der gelesenen Bausteine wird in pAmount geschrieben und eine Fehlermeldung "Bausteingrenze überschritten" zurückgegeben.

Aufrufbeispiel

```
int BlkType = 0x0a;                          // Bausteintyp = DB
unsigned short StartNr = 2;                  // Erster zu lesender Baustein = 2
int Amount = 4;                              // Anzahl zu lesende Bausteine = 4
BST_DIAG_TYPE Buffer[MAX_BLK];               // Übergabepuffer Array mit
                                             Bausteinlisten
BST_DIAG_TYPE * pDiagBuffer =                // Adresse des Übergabepuffers
&Buffer;
unsigned long BufLen = sizeof(Buffer);       // Länge des Übergabepuffers
unsigned long DatLen;                        // Datenlänge
int RetValue;
RetValue = bst_read_diag_ex6 (BlkType, StartNr, &Amount, BufLen, pDiagBuffer,
&DatLen);
```

Voraussetzungen

		V6.0 bst_read_diag_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

bst_read_stat_ex6, bst_read_ex6, read_diag_buf_ex6

3.1.12 bst_read_stat_ex6

Die Funktion **bst_read_stat_ex6** liest Bausteinlisten und Headerinformationen für die Statusbilder (OBs/FCs/FBs/DBs).

int bst_read_stat_ex6 (int *BlkType,* unsigned long BufLen, BST_STAT_TYPE * *pStatBuffer*, unsigned long * *pDatLen*);

Parameter

BlkType

[in] Typ des zu lesenden Bausteins:

BlkType: 0x08 = OB, 0x0a = DB, 0x0c = FC, 0x0e = FB default: *BlkTyp* = 0x0a

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pStatBuffer

[out] Übergabepuffer für die gelesenen Daten

```
typedef struct {
  unsigned short BlkType;          // Bausteintyp
  unsigned short BlkNumber;        // Bausteinnummer
  char BlkName[8];                 // Bausteinname
  unsigned char BlkVersion;        // Bausteinversion
  unsigned long BlkLength;         // Bausteinlänge
  struct tm BlkTimestamp1;         // Zeitstempel 1 zu der Bausteinnummer
  struct tm BlkTimestamp2;         // Zeitstempel 2 zu der Bausteinnummer
  unsigned char blkSecurity[4];    // Bausteinpasswort und Schutzstufe
} BST_STAT_TYPE;
```

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
int BlkType = 0x0a;                          // Bausteintyp = DB
BST_STAT_TYPE Buffer[MAX_BLK];               // Übergabepuffer gelesene Daten
BST_STAT_TYPE * pStatBuffer = &Buffer;       // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);       // Länge des Übergabepuffers
unsigned long DatLen;                        // Datenlänge
int RetValue;
RetValue = bst_read_stat_ex6 (BlkType, BufLen, pStatBuffer, &DatLen);
```

Voraussetzungen

		V6.0 bst_read_stat_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

bst_read_diag_ex6, bst_read_ex6, read_diag_buf_ex6

3.1.13 bst_read_ex6

Die Funktion **bst_read_ex6** liest Bausteine in einen Puffer ein. "Code"-Bausteine werden komplett eingelesen (Header und Daten); bei OBs und DBs wird nur der Header gelesen.

int bst_read_ex6 (int *BlkType, unsigned short blkNr*, unsigned long BufLen, unsigned char * pReadBuffer, unsigned long * *pDatLen*);

Parameter

BlkType

[in] Typ des zu lesenden Bausteins:

BlkType: 0x08 = OB, 0x0a = DB, 0x0c = FC, 0x0e = FB default: *BlkTyp* = 0x0a

BlkType

[in] Bausteinnummer des zu lesenden Bausteins

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pReadBuffer

[out] Zeiger auf Übergabepuffer für die gelesenen Daten

pDatLen

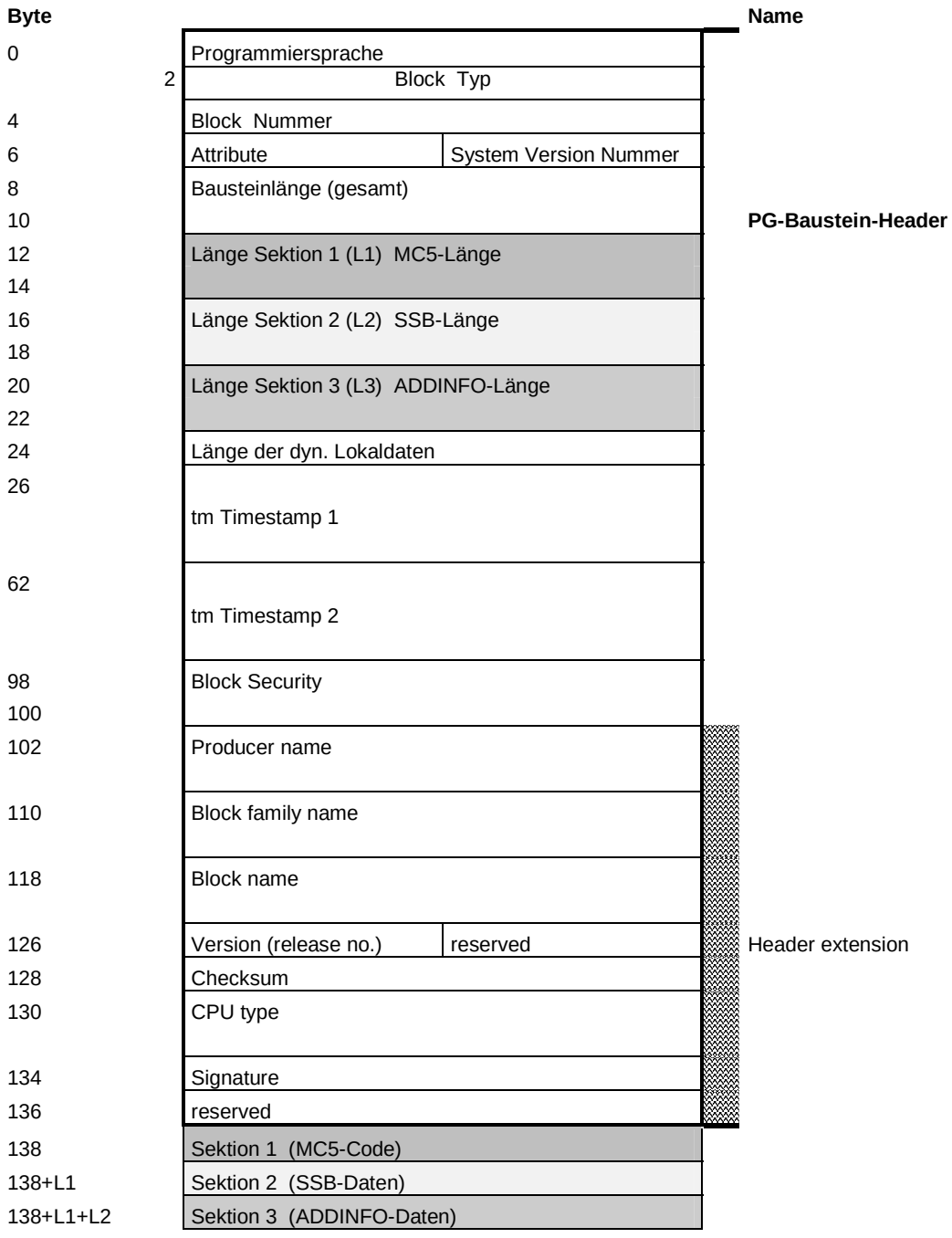
[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Bei Know-How geschützten Bausteinen ist das Lesen nicht möglich. In diesem Fall wird eine entsprechende Fehlernummer zurückgegeben.

Baustein


Byte
Name
0
Programmiersprache
2
Block Typ
4
Block Nummer
6
Attribute
System Version Nummer
8
Bausteinlänge (gesamt)
10
PG-Baustein-Header
12
Länge Sektion 1 (L1) MC5-Länge
14
16
Länge Sektion 2 (L2) SSB-Länge
18
20
Länge Sektion 3 (L3) ADDINFO-Länge
22
24
Länge der dyn. Lokaldaten
26
tm Timestamp 1
62
tm Timestamp 2
98
Block Security
100
102
Producer name
110
Block family name
118
Block name
126
Version (release no.)
reserved
Header extension
128
Checksum
130
CPU type
134
Signature
136
reserved
138
Sektion 1 (MC5-Code)
138+L1
Sektion 2 (SSB-Daten)
138+L1+L2
Sektion 3 (ADDINFO-Daten)

Header des gelesenen Bausteins:

```
typedef struct {
    unsigned char ProgLang;              // Programiersprache
    unsigned char BlkType;               // Bausteintyp
    unsigned short BlkNumber;            // Bausteinnummer
    unsigned char Attribute[8];          // Attribute
    unsigned char BlkVersion;            // Bausteinversion
    unsigned long BlkLength;             // Bausteinlänge
    unsigned long Length[3];             // Datenlänge
    unsigned char DynLen;                // Länge der dyn. lokal Daten
    struct tm BlkTimestamp1;             // Zeitstempel 1 zur Bausteinnummer
    struct tm BlkTimestamp2;             // Zeitstempel 2 zur Bausteinnummer
    unsigned char blkSecurity [4];       // Baustein Schutzstufe
    unsigned char ProducerName[8];       // Producer Name
    unsigned char BlkFamName[8];         // Baustein Familien Name
    unsigned char BlkName[8];            // Bausteinname
    unsigned char Version;               // Dies ist eine Anwenderversion und
                                          hat nicht mit der Bausteinversion zu
                                          tun
    unsigned short chckSum;              // Checksumme
    unsigned long CPUType;               // CPU-Typ
    unsigned short signature;            // signature
} BST_HEADER_TYPE;
```

Aufrufbeispiel

```
int BlkType = 0x0a;                          // Bausteintyp = DB
unsigned short blkNr = 34;                   // Bausteinnummer 34
BST_HEADER_TYPE
Buffer[MAX_BUFFER];                          // Übergabepuffer gelesene Daten
BST_HEADER_TYPE *pReadBuffer =
Buffer;                                      // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);       // Länge des Übergabepuffers
unsigned long DatLen;                        // Datenlänge
int RetValue;
RetValue = bst_read_ex6 (BlkType, blkNr, BufLen, &Buffer[0], &DatLen);
```

Voraussetzungen

		V6.0 bst_read_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

bst_read_diag_ 60, bst_read_stat_ex6, read_diag_buf_ex6

3.1.14 read_diag_buf_ex6

Die Funktion **read_diag_buf_ex6** liest Daten des Diagnosepuffers und der SZLs, um bestimmte Informationen über den Status einer CPU zu erfahren.

int read_diag_buf_ex6 (unsigned long BufLen, unsigned char * *pDiagBuffer*, unsigned long * *pDatLen*);

Parameter

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pDiagBuffer

[out] Zeiger auf Übergabepuffer für die gelesenen Daten

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Hinweis

Bei den Auskünften kann die Anzahl, die maximal lieferbar ist, vom Betriebszustand abhängig sein (produktspezifische Festlegung). Beispielsweise kann eine CPU alle Infos nur im Betriebszustand STOP liefern.

Im Betriebszustand RUN wird nur eine produktspezifische Teilmenge geliefert.

Beispiel eines Diagnoseeintrags

```
typedef struct {
  unsigned short EventID;          // EreignisID
  unsigned char EventInfo[20];          // Info zum Ereignis
  unsigned char Timestamp[8];          // Zeitstempel des Ereignisses
} DIAG_BUFFER_TYPE;
```

Aufrufbeispiel

```
Unsigned char Buffer[MAX_BUFFER];      // Übergabepuffer gelesene Daten
(DIAG_BUFFER_TYPE) &Buffer[0];         // Struktur über Puffer legen
unsigned long BufLen = sizeof(Buffer); // Länge des Übergabepuffers
unsigned long DatLen;                  // Datenlänge
int RetValue;
RetValue = read_diag_buf_ex6 (BufLen, pDiagBuffer, &DatLen);
```

Voraussetzungen

		V6.0 read_diag_buf_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

bst_read_diag_ex6, bst_read_stat_ex6, bst_read_ex6

3.1.15 field_read_ex6

Die Funktion **field_read_ex6** liest eine Anzahl Bytes oder Worte aus dem AS aus und transferiert diese in einen Übergabepuffer des PG/PC.

int field_read_ex6 (char *FieldType*, unsigned short *BlkNr*, unsigned short *StartNr*, unsigned long *Amount*, unsigned long BufLen, unsigned char * *pBuffer*, unsigned long * *pDatLen*);

Parameter

FieldType

[in] Werte-Typ als ASCII-Zeichen: Datenbyte (d/D), Eingangsbyte (e/E), Ausgangsbyte (a/A), Merkerbyte (m/M), Timerwort (t/T), Zählerwort (z/Z)

BlkNr

[in] Bausteinnummer (nur bei Datenbausteinen anzugeben), default = 0

StartNr

[in] Anfangsnummer des ersten zu lesenden Bytes oder Worts

Amount

[in] Anzahl der zu lesenden Bytes oder Worte

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pBuffer

[out] Übergabepuffer für die gelesenen Bytes oder Worte

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
char FieldType = 'D'                          // Wertetyp = Datenbyte (D)
unsigned short BlkNr = 0;                     // Bausteinnummer = 0
unsigned short StartNr = 2;                   // Erstes zu lesendes Wort = 2
int Amount = 6;                               // Anzahl zu lesende Worte = 6
unsigned char Buffer[MAX_BUFFER];             // Übergabepuffer
unsigned char *pBuffer = Buffer;              // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);        // Länge des Übergabepuffers
unsigned long DatLen;                         // Datenlänge
int RetValue;
RetValue = field_read_ex6 (FieldType, BlkNr, StartNr, Amount, BufLen, pBuffer,
&DatLen);
unsigned short * wBuffer = (unsigned          // wortweiser Zugriff
short *)Buffer;
```

Voraussetzungen

	V5.6 d|e|a|m|t|z_field_read	**V6.0 field_read_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

field_write_ex6

3.1.16 field_write_ex6

Die Funktion **field_write_ex6** schreibt eine Anzahl Bytes oder Worte aus einem Übergabepuffer des PG/PC in das AS.

int field_write_ex6 (char *FieldType*, unsigned short *BlkNr*, unsigned short *StartNr*, unsigned long *Amount,* unsigned long BufLen, unsigned char * *pBuffer*);

Parameter

FieldType

[in] Wertetyp als ASCII-Zeichen: Datenbyte (d/D), Eingangsbyte (e/E), Ausgangsbyte (a/A), Merkerbyte (m/M), Timerwort (t/T), Zählerwort (z/Z)

BlkNr

[in] Bausteinnummer (nur bei Datenbausteinen anzugeben), default = 0

StartNr

[in] Anfangsnummer des ersten zu schreibenden Bytes oder Worts

Amount

[in] Anzahl der zu schreibenden Bytes oder Worte

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pBuffer

[out] Zeiger auf Übergabepuffer für die zu schreibenden Bytes oder Worte

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
char FieldType = 'D'                    // Wertetyp = Datenbyte (D)
unsigned short BlkNr = 0;               // Bausteinnummer = 0
unsigned short StartNr = 2;             // Erstes zu schreibendes Wort = 2
int Amount = 6;                         // Anzahl zu schreibende Worte = 6
unsigned char Buffer[MAX_BUFFER];       // Übergabepuffer
unsigned char *pBuffer = Buffer;        // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);  // Länge des Übergabepuffers
int RetValue;
RetValue = field_write_ex6 (FieldType, BlkNr, StartNr, Amount, BufLen, pBuffer);

unsigned short * wBuffer = (unsigned    // wortweiser Zugriff
short *)Buffer;
```

Voraussetzungen

	V5.6 d|e|m|z_field_write	**V6.0 field_write_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

field_read_ex6

3.1.17 mb_setbit_ex6

Die Funktion **mb_setbit_ex6** setzt einen Merker im AS auf 1 oder 0. Es wird nicht geprüft, ob das Merkerbit im verwendeten AS vorhanden ist.

int mb_setbit_ex6 (unsigned short *MbNr*, unsigned short *BitNr,* unsigned char *Value*);

Parameter

MbNr

[in] Nummer des Merkerbytes

BitNr

[in] Bitnummer im Merkerbyte

Value

[in] zu setzender Wert (0/1)

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned short MbNr = 2;                    // Merkerbyte Nummer 2
unsigned short BitNr = 4;                   // Bit Nummer 4
unsigned char Value = 1;                    // zu setzender Wert = 1
int RetValue;
RetValue = mb_setbit_ex6 (MbNr, BitNr, Value);
```

Voraussetzungen

	V5.6 mb_setbit, mb_resetbit	**V6.0 mb_setbit_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

mb_bittest_ex6

3.1.18 mb_bittest_ex6

Die Funktion **mb_bittest_ex6** prüft ein Bit in einem angegebenen Merkerbyte auf den angegebenen Wert und liefert im Übergabepuffer den Zustand des angegebenen Bits.

int mb_bittest_ex6 (unsigned short *MbNr*, unsigned short *BitNr,* int * *pValue*);

Parameter

MbNr

[in] Nummer des Merkerbytes

BitNr

[in] Bitnummer im Merkerbyte

pValue

[out] Zeiger auf Übergabebyte mit dem getesteten Bitwert

Value == TRUE	Bit gesetzt oder 1
Value == FALSE	Bit nicht gesetzt oder 0

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned short MbNr = 2;          // Merkerbyte Nummer 2
unsigned short BitNr = 4;         // Bit Nummer 4
int Value;                        // Übergabebyte für den zu testenden
                                  Bitwert
int RetValue;
RetValue = mb_bittest_ex6 (MbNr, BitNr, &Value);
```

Voraussetzungen

	V5.6 mb_bittest	**V6.0 mb_bittest_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

mb_setbit_ex6

3.2 Funktionen zum Datenverkehr zur S7-200

3.2.1 as200_as_info_ex6

Die Funktion **as200_as_info_ex6** liest den Ausgabestand der AS-Software und der PG-Anschaltung, sowie den PLC-Typ des AS aus und legt diese als ASCII-String nullterminiert im Übergabepuffer des PG/PC ab.

int as200_as_info_ex6 (unsigned long BufLen, AS200_INFO_TYPE * *pBuffer*, unsigned long * *pDatLen*);

Parameter

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pBuffer

[out] Zeiger auf Übergabepuffer mit den zu liefernden AS-Informationen

```
typedef struct {
  unsigned char Firmware[4];        // FW-Ausgabestand
  unsigned char ASIC[4];            ASIC-Ausgabestand
  char Mlfb[16];                    // MLFB des angeschlossenen AS
    } AS200_INFO_TYPE;
```

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
AS200_INFO_TYPE Buffer;                    // Übergabepuffer anlegen
AS200_INFO_TYPE * pBuffer = &Buffer;       // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);     // Länge des Übergabepuffers
unsigned long DatLen;                      // Datenlänge
int RetValue;
RetValue = as200_as_info_ex6 (BufLen, pBuffer, &DatLen);
```

Voraussetzungen

	V5.6 as200_ag_info	**V6.0 as200_as_info_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as200_as_zustand_ex6

3.2.2 as200_as_zustand_ex6

Die Funktion **as200_as_zustand_ex6** liest den AS-Zustand (RUN oder STOP) aus dem AS aus und legt die Daten in einem Speicherbereich des PG/PC ab.

int as200_as_zustand_ex6 (unsigned char * *pState*);

Parameter

pState

[out] Übergabepuffer mit den zu liefernden AS-Zuständen (benötigte Länge 1 Byte)

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned char State;                    // Übergabebyte für den zu liefernden
                                        AS-Zustand
int RetValue;
RetValue = as200_as_zustand_ex6 (&State);
```

"Aktueller Betriebszustand" (Werte in Hex)

BZ-ID	Bedeutung
00	STOP
01	RUN

Voraussetzungen

	V5.6 as200_ag_zustand	V6.0 as200_as_zustand_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as200_as_info_ex6

3.2.3 as200_field_read_ex6

Die Funktion **as200_field_read_ex6** liest eine Anzahl Bytes aus dem AS aus und transferiert diese in einen Übergabepuffer des PG/PC.

int as200_field_read_ex6 (char Field*Type*, unsigned short *BlkNr*, unsigned short *StartNr*, unsigned long *Amount*, unsigned long BufLen, unsigned char * *pBuffer*, unsigned long * *pDatLen*);

Parameter

FieldType

[in] Wertetyp als ASCII-Zeichen: Eingangsbyte (e/E), Ausgangsbyte (a/A), Merkerbyte (m/M), Sondermerkerbyte (s/S), Variablenspeicherbyte (v/V), Timerwert (t/T), Zählerwert (z/Z)

BlkNr

[in] Bausteinnummer (nur bei Datenbausteinen anzugeben), default = 0

StartNr

[in] Anfangsnummer des ersten zu lesenden Bytes oder Werts

Amount

[in] Anzahl der zu lesenden Bytes oder Werte

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pBuffer

[out] Übergabepuffer für die gelesenen Bytes oder Worte

pDatLen

[out] Zeiger auf Länge der im Puffer gelieferten Daten in Byte. Bei Fehler "bereitgestellter Übergabepuffer zu klein für Daten" steht hier die benötigte Größe des Puffers.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
char FieldType = 0x45                    // Wertetyp = Eingangsbyte (E)
unsigned short BlkNr = 0;                // Bausteinnummer = 0
unsigned short StartNr = 2;              // Erstes zu lesendes Wort = 2
int Amount = 6;                          // Anzahl zu lesende Worte = 6
unsigned char Buffer[MAX_BUFFER];        // Übergabepuffer
unsigned char *pBuffer = Buffer;         // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);   // Länge des Übergabepuffers
unsigned long DatLen;                    // Datenlänge
int RetValue;
RetValue = as200_field_read_ex6 (FieldType, BlkNr, StartNr, Amount, BufLen,
pBuffer, &DatLen);
```

Voraussetzungen

	V5.6 as200_e|a|…_field_read	**V6.0 as200_field_read_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as200_field_write_ex6

3.2.4 as200_field_write_ex6

Die Funktion **as200_field_write_ex6** schreibt eine Anzahl Bytes aus einen Übergabepuffer des PG/PCs in das AS.

int as200_field_write_ex6(char Field*Type*, unsigned short *BlkNr*, unsigned short *StartNr*, unsigned long *Amount,* unsigned long BufLen, unsigned char * *pBuffer*);

Parameter

FieldType

[in] Wertetyp als ASCII-Zeichen: Eingangsbyte (e/E), Ausgangsbyte (a/A), Merkerbyte (m/M), Sondermerkerbyte (s/S), Variablenspeicherbyte (v/V), Timerwert (t/T), Zählerwert (z/Z)

BlkNr

[in] Bausteinnummer (nur bei Datenbausteinen anzugeben), default = 0

StartNr

[in] Anfangsnummer des ersten zu schreibenden Bytes oder Werts

Amount

[in] Anzahl der zu schreibenden Bytes oder Werte

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pBuffer

[out] Übergabepuffer für die zu schreibenden Bytes oder Werte

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
char FieldType = 0x45                    // Wertetyp = Eingangsbyte (E)
unsigned short BlkNr = 0;                // Bausteinnummer = 0
unsigned short StartNr = 2;              // Erstes zu schreibendes Wort = 2
int Amount = 6;                          // Anzahl zu schreibende Worte = 6
unsigned char Buffer[MAX_BUFFER];        // Übergabepuffer
unsigned char *pBuffer = Buffer;         // Adresse des Übergabepuffers
unsigned long BufLen = sizeof(Buffer);   // Länge des Übergabepuffers
int RetValue;
RetValue = as200_field_write_ex6 (FieldType, BlkNr, StartNr, Amount, BufLen,
pBuffer);
```

Voraussetzungen

	V5.6 as200_e|m|…_field_write	**V6.0 as200_field_write_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as200_field_read_ex6

3.2.5 as200_mb_setbit_ex6

Die Funktion **as200_mb_setbit_ex6** setzt einen Merker imr AS auf 1 oder 0. Es wird nicht geprüft, ob das Merkerbit im verwendeten AS vorhanden ist.

int as200_mb_setbit_ex6 (unsigned short *MbNr*, unsigned short *BitNr,* unsigned char *Value*);

Parameter

MbNr

[in] Nummer des Merkerbytes

BitNr

[in] Bitnummer im Merkerbyte

Value

[in] zu setzender Wert (0/1)

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned short MbNr = 2;                    // Merkerbytenummer 2
unsigned short BitNr = 4;                   // Bitnummer 4
unsigned char Value = 1;                    // zu setzender Wert = 1
int RetValue;
RetValue = as200_mb_setbit_ex6 (MbNr, BitNr, Value);
```

Voraussetzungen

	V5.6 as200_mb_setbit, …_resetbit	V6.0 as200_mb_setbit_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as200_mb_bittest_ex6

3.2.6 as200_mb_bittest_ex6

Die Funktion **as200_mb_bittest_ex6** prüft ein Bit in einem angegebenen Merkerbyte und liefert im Übergabepuffer den Zustand des angegebenen Bits.

int as200_mb_bittest_ex6 (unsigned short *MbNr*, unsigned short *BitNr,* int * *pValue*);

Parameter

MbNr

[in] Nummer des Merkerbytes

BitNr

[in] Bitnummer im Merkerbyte

pValue

[out] Zeiger auf Übergabebyte mit dem getesteten Bitwert

Value == TRUE	Bit gesetzt oder 1
Value == FALSE	Bit nicht gesetzt oder 0

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage_ex6**).

Aufrufbeispiel

```
unsigned short MbNr = 2;          // Merkerbytenummer 2
unsigned short BitNr = 4;         // Bitnummer 4
int Value;                        // Übergabebyte für den zu testenden
                                  Bitwert
int RetValue;
RetValue = as200_mb_bittest_ex6 (MbNr, BitNr, &Value);
```

Voraussetzungen

	V5.6 as200_mb_bittest	V6.0 as200_mb_bittest_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

as200_mb_setbit_ex6

3.3 Komfortfunktionen

3.3.1 GetErrorMessage_ex6

Die Funktion **GetErrorMessage_ex6** liefert zu einer Fehlernummer den zugehörigen Fehlertext als nullterminierten Character-String.

int GetErrorMessage_ex6 (int *ErrorNr*, unsigned long BufLen, unsigned char * *pBuffer*);

Parameter

ErrorNr

[in] Fehlernummer

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pBuffer

[out] Zeiger auf Übergabepuffer mit den Fehlertexten

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0. Bei Fehler liefert die Funktion folgende Werte:

1. Datei ERROR.DAT ist nicht vorhanden oder kann nicht geöffnet werden.
2. Fehler beim Lesen der Datei ERROR.DAT.
3. Falscher Aufbau der Datei ERROR.DAT.
4. Zu dieser Fehlernummer existiert kein Fehlertext.
5. Zu viele Fehlertexte in der Datei ERROR.DAT.

Aufrufbeispiel

```
int ErrorNr = 0x303;                          // "Bausteingrenze überschritten,
                                                 Anzahl korrigieren"
unsigned char Buffer[MAX_BUFFER];             // Übergabepuffer für Fehlertext
unsigned long BufLen = sizeof(Buffer);        // Pufferlänge
unsigned char *pBuffer = Buffer;              // Adresse des Übergabepuffers
int RetValue;
RetValue = GetErrorMessage_ex6 (ErrorNr, BufLen, pBuffer);
```

In ERROR.DAT sollten nicht mehr als 512 Fehlertexte gespeichert werden. Der Fehlertext darf 200 Zeichen nicht überschreiten.

Bei Übergabe der Fehlernummer 0 kann in „Buffer“ der Dateiname der zu ladenden Fehlertextdatei übergeben werden (z. B. eine englische oder eine deutsche Fehlertextdatei). Wurde kein gültiger Dateiname oder ein NULL-Pointer übergeben, wird die Datei ERROR.DAT im aktuellen Verzeichnis gelesen. Deshalb sollte sichergestellt sein, dass die Datei ERROR.DAT vorhanden ist und im gleichen Verzeichnis wie das Programm liegt.

Die Datei ERROR.DAT wird beim ersten Aufruf der Funktion **GetErrorMessage_ex6** gelesen und die Texte in einem Feld gespeichert.

Es ist empfehlenswert, kurz nach Programmbeginn die Funktion **GetErrorMessage_ex6** mit error_no = 0 aufzurufen, um die Datei ERROR.DAT zu laden. Somit ist bei weiteren Aufrufen dieser Funktion die Ausführungszeit nahezu konstant.

Aufbau der Fehlertext-Datei:

[Fehlernummer als ASCII-Hex]: [Fehlertext]

Beispiel für Fehlertexte in ERROR.DAT:

00E1:	zu viele offene Kanäle
00E9:	sin_serv.exe nicht gestartet
00CE:	Baugruppe nicht gefunden
00CF:	Treiber nicht geladen
00CA:	keine Ressourcen verfügbar
00F1:	kein globaler DOS-Speicher verfügbar
0101:	Verbindung nicht aufgebaut / parametriert
010a:	negative Quittung empfangen / Timeout-Fehler
010c:	Daten nicht vorhanden oder gesperrt
0201:	falsche Schnittstelle angegeben
0202:	Maximalanzahl Schnittstellen überschritten
0203:	Toolbox schon installiert
0204:	Toolbox schon mit anderen Verbindungen installiert
0205:	Toolbox nicht installiert
0206:	Handle kann nicht gesetzt werden
0207:	Datensegment kann nicht gesperrt werden
0209:	Datenfeld fehlerhaft !
0302:	Baustein zu klein, Datenwort ist nicht vorhanden
0303:	Bausteingrenze überschritten, Anzahl korrigieren
0310:	Baugruppe nicht gefunden
0311:	Hardwarefehler
...	
...	
FFFE:	unbekannter Fehler FFFE hex
FFFF:	Timeout-Fehler. Schnittstelle überprüfen

Voraussetzungen

	V5.6 error_message	**V6.0 GetErrorMessage_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.3.2 kg_2_float_ex6

Die Funktion **kg_2_float_ex6** wandelt einen S5-Gleitpunktwert in einen Wert vom Typ float (IEEE-Format) um.

int kg_2_float_ex6 (unsigned long *kg,* float * *pieee*);

Parameter

kg

[in] S5-Gleitpunktwert

pieee

[out] Zeiger auf Floatwert

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine 1.

Aufrufbeispiel

```
unsigned long kg = S5_FLOAT_VALUE;
float ieee;
int RetValue;
RetValue = kg_2_float_ex6 (kg, &ieee);    // in ieee wird der IEEE Gleitpunktwert geliefert
```

Voraussetzungen

	V5.6 kg_to_float	**V6.0 kg_2_float_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

float_2_kg_ex6

3.3.3 float_2_kg_ex6

Die Funktion **float_2_kg_ex6** wandelt einen Wert vom Typ float (IEEE-Format) in einen S5-Gleitpunktwert um.

int float_2_kg_ex6 (float *ieee,* unsigned long * *pkg*);

Parameter

ieee

[in] Floatwert

pkg

[out] Zeiger auf S5-Gleitpunktwert

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine 1.

Aufrufbeispiel

```
float ieee = FLOAT_VALUE;
unsigned long kg;
int RetValue;
RetValue = float_2_kg_ex6 (ieee, &kg);    // in kg wird der S5-Gleitpunktwert
                                          geliefert
```

Voraussetzungen

	V5.6 float_to_kg	**V6.0 float_2_kg_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

kg_2_float_ex6

3.3.4 gp_2_float_ex6

Die Funktion **gp_2_float_ex6** wandelt einen S7-Gleitpunktwert in einen Wert vom Typ float (IEEE-Format) um.

int gp_2_float_ex6 (unsigned long *gp,* float * *pieee*);

Parameter

gp

[in] S7-Gleitpunktwert

pieee

[out] Zeiger auf Floatwert

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine 1.

Aufrufbeispiel

```
unsigned long gp = S7_FLOAT_VALUE;
float ieee;
gp_2_float_ex6 (gp, &ieee);          // in ieee wird der IEEE-Gleitpunktwert geliefert
```

Voraussetzungen

	V5.6 gp_to_float	**V6.0 gp_2_float_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

float_2_gp_ex6

3.3.5 float_2_gp_ex6

Die Funktion **float_2_gp_ex6** wandelt einen Wert vom Typ float (IEEE-Format) in einen S7-Gleitpunktwert um.

int float_2_gp_ex6 (float *ieee,* unsigned long * *pgp*);

Parameter

ieee

[in] Floatwert

pgp

[out] Zeiger auf S7-Gleitpunktwert

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine 1.

Aufrufbeispiel

```
float ieee = FLOAT_VALUE;
unsigned long gp;
float_2_gp_ex6 (ieee, &gp);          // in gp wird der S7-Gleitpunktwert
                                     geliefert
```

Voraussetzungen

	V5.6 float_to_gp	V6.0 float_2_gp_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

gp_2_float_ex6

3.3.6 testbit_ex6

Die Funktion **testbit_ex6** prüft, ob ein angegebenes Bit in einer Bytevariablen gesetzt ist. Die Bytevariable und die Bitnummer werden der Funktion als Parameter übergeben.

int testbit_ex6 (char *Value,* int *BitNr*);

Parameter

Value

[in] Wert der Bytevariablen

BitNr

[in] zu testendes Bit in der Bytevariablen

Rückgabewerte

Rückgabewert TRUE: Bit ist gesetzt (oder 1)

Rückgabewert FALSE: Bit ist nicht gesetzt (oder 0)

Aufrufbeispiel

keines

Voraussetzungen

	V5.6 testbit	**V6.0 testbit_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.3.7 byte_2_bool_ex6

Die Funktion **byte_2_bool_ex6** wandelt ein Byte in acht logische Werte (PC-Darstellung) um.

void byte_2_bool_ex6 (unsigned char *Value*, int * *pBuffer*);

Parameter

Value

[in] Bytewert

pBuffer

[out] Zeiger auf Puffer mit 8 umgewandelten logischen Werten

Rückgabewerte

keine

Aufrufbeispiel

Der übergebene Zeiger sollte auf ein unsigned char-Feld mit folgendem Aufbau zeigen:

Buffer[0]	*Buffer*[1]	*Buffer*[2]	*Buffer*[3]	*Buffer*[4]	*Buffer*[5]	*Buffer*[6]	*Buffer*[7]
bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7

```
#define SIZE_BOOL_BYTE 8
int Buffer[SIZE_BOOL_BYTE];          // Puffer für 8 logische Werte
unsigned char Value = 1;
byte_2_bool_ex6 (Value, Buffer);     // in Buffer[0] wird TRUE geliefert,
                                     Buffer[1]-[7] FALSE
```

Voraussetzungen

	V5.6 byte_boolean	**V6.0 byte_2_bool_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

bool_2_byte_ex6

3.3.8 bool_2_byte_ex6

Die Funktion **bool_2_byte_ex6** wandelt acht logische Werte (PC-Darstellung) in ein Byte um.

unsigned char bool_2_byte_ex6 (int * *pBuffer*);

Parameter

pBuffer

[in] Zeiger auf Puffer mit acht logischen Werten

Rückgabewerte

Umgewandelter Bytewert

Aufrufbeispiel

Der übergebene Zeiger sollte auf ein unsigned char-Feld mit folgendem Aufbau zeigen:

Buffer[0]	*Buffer*[1]	*Buffer*[2]	*Buffer*[3]	*Buffer*[4]	*Buffer*[5]	*Buffer*[6]	*Buffer*[7]
bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7

```
#define SIZE_BOOL_BYTE 8
int Buffer[SIZE_BOOL_BYTE] = {0};          // Puffer mit 8 logischen Werten
Buffer[0] = TRUE ;
unsigned char RetValue;
RetValue = bool_2_byte_ex6 (Buffer);       // RetValue hat den Wert 1
```

Voraussetzungen

	V5.6 boolean_byte	**V6.0 bool_2_byte_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

byte_2_bool_ex6

3.3.9 kf_2_integer_ex6, kf_2_long_ex6

Die Funktionen vertauschen das High- und das Low-Byte eines übergebenen Werts.

Mit **kf_2_integer_ex6** werden die High- und Low-Bytes eines 16-Bit Werts vertauscht.

Mit **kf_2_long_ex6** werden die High- und Low-Bytes eines 32-Bit Werts vertauscht.

unsigned short kf_2_integer_ex6 (unsigned short w*Value*);

unsigned long kf_2_long_ex6 (unsigned long dw*Value*);

Parameter

wValue

[in] 16-Bit-Werte

dwValue

[in] 32-Bit-Werte

Rückgabewerte

kf_integer liefert einen 16-Bit-Wert mit vertauschten Bytes.

kf_long_ex6 liefert einen 32-Bit-Wert mit vertauschten Bytes.

Aufrufbeispiel

Beispiel: (Doppelwort)

0	1	2	3
…Umwandlung…			
3	2	1	0

Voraussetzungen

	V5.6 kf_integer	**V6.0 kf_2_integer_ex6, kf_2_long_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

swab_buffer_ex6

3.3.10 swab_buffer_ex6

Die Funktion **swab_buffer_ex6** vertauscht die High- und Low-Bytes eines übergebenen Puffers.

void swab_buffer_ex6 (unsigned char * *pBuffer*, unsigned long *Amount*);

Parameter

pBuffer

[in/out] Zeiger auf den Puffer, in dem die Bytes getauscht werden

Amount

[in] Anzahl der zu tauschenden Bytes

Rückgabewerte

keine

Aufrufbeispiel

Intern wird die Standard C Funktion **void_swab(char * src, char * dest, int n)** aufgerufen.

a	b	…Umwandlung…	b	a
c	d		d	c
x	y		y	x

Voraussetzungen

	V5.6 swab_buffer	V6.0 swab_buffer_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

kf_2_integer_ex6, kf_2_long_ex6, copy_buffer_ex6

3.3.11 copy_buffer_ex6

Die Funktion **copy_buffer_ex6** kopiert eine Anzahl Bytes von einem Puffer in einen anderen.

void copy_buffer_ex6 (unsigned char * *pTargetBuffer*, unsigned char * *pSourceBuffer*, unsigned long *Amount*);

Parameter

pTargetBuffer

[out] Zeiger auf dem Puffer, in den die Bytes kopiert werden

pSourceBuffer

[in] Zeiger auf dem Puffer, aus dem die Bytes geholt werden

Amount

[in] Anzahl der zu kopierenden Bytes

Rückgabewerte

keine

Aufrufbeispiel

Intern wird die Standard C-Funktion **void *memcpy(char * dest, char * src, size_t count)** aufgerufen.

Voraussetzungen

	V5.6 copy_buffer	**V6.0 copy_buffer_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

swab_buffer_ex6

3.3.12 ushort_2_bcd_ex6, ulong_2_bcd_ex6

Die Funktionen sind Wandelroutinen, die eine Anzahl von Dualwerten in BCD-Werte umwandeln.

Mit **ushort_2_bcd_ex6** werden 16-Bit Werte (Worte) gewandelt.

Mit **ulong_2_bcd_ex6** werden 32-Bit Werte (Doppelworte) gewandelt.

void ushort_2_bcd_ex6 (unsigned short * *pwValues*, unsigned long *Amount*, int *InBytechange*, int *OutBytechange*);

void ulong_2_bcd_ex6 (unsigned long * *pdwValues*, unsigned long *Amount*, int *InBytechange*, int *OutBytechange*);

Parameter

pwValues

[in/out] Zeiger auf 16-Bit-Dualwerte

pdwValues

[in/out] Zeiger auf 32-Bit-Dualwerte

Amount

[in] Anzahl Werte

InBytechange

[in] TRUE oder FALSE

OutBytechange

[in] TRUE oder FALSE

Rückgabewerte

Nach Aufruf der Funktion zeigt *pwValues* auf 16-Bit BCD-Werte, *pdwValuese* auf 32-Bit BCD-Werte.

Aufrufbeispiel

Mit der Funktion können Sie z. B. Zähler setzen oder Zeitfunktionen versorgen.

Ist der Parameter 'InBytechange' gesetzt (1) so werden vor dem Wandeln in einen BCD-Wert, die High und Low Bytes vertauscht. Wird statt dessen der Parameter 'OutBytechange' gesetzt, so werden die High- und Low-Bytes erst nach der Wandlung vertauscht.

Wird keines der beiden Bytechange-Argumente gesetzt, so wird keine High-Low-Byte-Vertauschung durchgeführt.

Der verfügbare Zahlenbereich für 16-Bit BCD-Werte ist +999 bis –999.

Der verfügbare Zahlenbereich für 32-Bit BCD-Werte ist +9 999 999 bis –9 999 999.

Voraussetzungen

	V5.6 USHORT_2_bcd	**V6.0 ushort_2_bcd_ex6, ulong_2_bcd_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

bcd_2_ushort_ex6, bcd_2_ulong_ex6

3.3.13 bcd_2_ushort_ex6, bcd_2_ulong_ex6

Die Funktionen sind Wandelroutinen, die eine Anzahl von BCD-Werten in Dualwerte umwandeln.

Mit **bcd_2_ushort_ex6** werden 16-Bit Werte (Worte) gewandelt.

Mit **bcd_2_ulong _ex6** werden 32-Bit Werte (Doppelworte) gewandelt.

void bcd_2_ushort_ex6 (unsigned short * *pwValues*, unsigned long *Amount*, int *InBytechange*, int *OutBytechange*);

void bcd_2_ulong_ex6 (unsigned long * *pdwValues*, unsigned long *Amount*, int *InBytechange*, int *OutBytechange*);

Parameter

pwValues

[in/out] Zeiger auf 16-Bit BCD-Werte

pdwValues

[in/out] Zeiger auf 32-Bit BCD-Werte

Amount

[in] Anzahl Werte

InBytechange

[in] TRUE oder FALSE

OutBytechange

[in] TRUE oder FALSE

Rückgabewerte

Nach dem Aufruf der Funktion zeigt *pwValues* auf 16-Bit Dualwerte, *pdwValues* auf 32-Bit Dualwerte.

Aufrufbeispiel

Mit der Funktion können Sie z. B. Zähler setzen oder Zeitfunktionen versorgen.

Ist der Parameter InBytechange gesetzt (1), so werden vor dem Wandeln in einen Dualwert die High und Low Bytes vertauscht. Wird statt dessen der Parameter OutBytechange gesetzt, so werden die High und Low Bytes erst nach der Wandlung vertauscht.

Wird keines der beiden Bytechange-Argumente gesetzt, so wird keine High-Low-Byte-Vertauschung durchgeführt.

Der verfügbare Zahlenbereich für 16-Bit BCD-Werte ist +999 bis –999.

Der verfügbare Zahlenbereich für 32-Bit BCD-Werte ist +9 999 999 bis –9 999 999.

Voraussetzungen

	V5.6 bcd_2_USHORT	**V6.0 bcd_2_ushort_ex6, bcd_2_ulong_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Siehe auch:

ushort_2_bcd_ex6, ulong_2_bcd_ex6

3.3.14 GetLoadedConnections_ex6

Füllt ein Array mit 64 Boolwerten. Damit erhält der Anwender die Information, welche der 64 Verbindungsnummern (*ConNr)* schon besetzt sind, bzw welche Verbindungen schon initialisiert sind.

void GetLoadedConnections_ex6(unsigned long BufLen, int *pBuffer)

Parameter

BufLen

[in] Länge des vom Anwender bereitgestellten Übergabepuffers in Byte

pBuffer

[out] Zeiger auf ein Array mit Boolwerten

Rückgabewerte

keine

Aufrufbeispiel

keines

Voraussetzungen

		V6.0 GetLoadedConnections_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.4 TeleService-Funktionen

3.4.1 ts_dial_ex6

Die Funktion **ts_dial_ex6** wählt eine entfernte Station über Modem an und baut die Verbindung zum TS-Adapter auf.

int ts_dial_ex6 (char * *cModemName*, char * *cStandort*, char * *cTelNo*, char * *cUserName*, char * *cPassword*, HANDLE *WindowHandle*, unsigned int *Message*, WPARAM *wParam*, char * *Res1*);

Parameter

cModemName

[in] Name des zu verwendenden Modems, einstellbar in Systemsteuerung / Modems / Wahlparameter.

cStandort

[in] Name des zu Modem Standortes, einstellbar in Systemsteuerung / Modems / Wahlparameter.

cTelNo

[in] Telefonnummer, die vom angeschlossenen Moden angewählt wird.

cUserName

[in] Hier wird der Benutzername angegeben, der im anzuwählenden TS-Adapter parametriert wurde.

cPassword

[in] Hier wird das Passwort angegeben, welches im anzuwählenden TS-Adapter parametriert wurde.

WindowHandle

[in] Hier kann ein Fenster-Handle übergeben werden.

Message

[in] Nachricht, die an das Fenster geschickt wird, wenn die Verbindung aufgebaut oder der Timeout abgelaufen ist.

wParam

[in] Parameter für die Nachricht.

Res1

[in] Reserviert für spätere Erweiterungen, muss jetzt mit NULL vorbelegt werden.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage**).

Hinweis

Wenn die Funktion **ts_dial_ex6** asynchron aufgerufen wird, wird im Fehlerfall keine Fehlernummer ausgegeben. Informieren Sie in diesem Fall den Bediener und lassen Sie ihn entscheiden, ob der die Funktion nochmals ausführen möchte, oder ob er detaillierte Fehlerinformationen benötigt. Für detaillierte Informationen verwenden Sie den Synchronaufruf.

Aufrufbeispiel

keines

Voraussetzungen

	V5.6 ts_dial	**V6.0 ts_dial_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.4.2 ts_hang_up_dial_ex6

Die Funktion **ts_hang_up_dial_ex6** bricht die momentan stehende Verbindung oder einen momentan laufenden asynchronen Wählvorgang ab.

int ts_hang_up_dial_ex6 (void);

Parameter

keine

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage**).

Aufrufbeispiel

keines

Voraussetzungen

	V5.6 ts_hang_up_dial	**V6.0 ts_hang_up_dial_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.4.3 ts_set_ringindicator_ex6

Die Funktion **ts_set_ringindicator_ex6** initialisiert das unterlagerte System für Anrufannahme, Verbindungsaufbau und Benachrichtigung (Ringindication).

int ts_set_ringindicator_ex6 (char * *ModemName*, char * *NumberOfRings*, HANDLE *WindowHandle*, unsigned int *Message*, WPARAM *wParam*, char * *Res1*);

Parameter

ModemName

[in] Name des für die Ringindication zu verwendenden Modems, einstellbar in Systemsteuerung / Modems.

NumberOfRings

[in] Anzahl der Klingelzeichen, bis das Modem abnehmen soll.

WindowHandle

[in] Hier kann ein Fenster-Handle übergeben werden.

Message

[in] Nachricht, die an das Fenster geschickt wird, wenn die Verbindung aufgebaut oder der Timeout abgelaufen ist.

wParam

[in] Parameter für die Nachricht.

Res1

[in] Reserviert für spätere Erweiterungen, muss jetzt mit NULL vorbelegt werden.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage**).

Aufrufbeispiel

keines

Voraussetzungen

	V5.6 ts_set_ringindicator	**V6.0 ts_set_ringindicator_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.4.4 ts_read_info_ex6

Die Funktion **ts_read_info_ex6** liefert Informationen über die alarmauslösende Station.

int ts_read_info_ex6 (void * *IventId*, unsigned char * *MpiAdr*);

Parameter

IventId

[in] Zeiger auf ein 16 Byte langes Feld. Hier werden Informationen der alarmauslösenden Station eingetragen.

MpiAdr

[in] Zeiger auf ein Byte. Hier wird die MPI-Adresse der alarmauslösenden Station eingetragen.

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage**).

Aufrufbeispiel

keines

Voraussetzungen

	V5.6 ts_read_info	**V6.0 ts_read_info_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.4.5 ts_hang_up_ring_ex6

Die Funktion **ts_hang_up_ring_ex6** bricht die vom TS-Adapter aufgebaute Verbindung ab.

int ts_hang_up_ring_ex6 (void);

Parameter

keine

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage**).

Aufrufbeispiel

keines

Voraussetzungen

	V5.6 ts_hang_up_ring	**V6.0 ts_hang_up_ring_ex6**
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

3.4.6 ts_get_modem_name_ex6

Die Funktion **ts_get_modem_name_ex6** liefert alle Namen der dem System bekannten Modems.

int ts_get_modem_name_ex6 (int *ModemId*, char * *Buffer*, int * *BufferLen*);

Parameter

ModemId

[in] Modem-ID 0 ... n

Buffer

[out] Zeiger auf den Puffer für Modemname

BufferLen

[in/out] Zeiger auf die Länge des Puffers; Enthält nach dem Aufruf die tatsächliche Länge des Strings mit dem Modemnamen.

Achtung

BufferLen muss vor dem Aufruf der Funktion ausreichend groß für den zu erwartenden Modemnamen gewählt werden!

Rückgabewerte

Wenn keine Fehler aufgetreten sind, liefert die Funktion als Rückgabewert eine 0, andernfalls eine Fehlernummer, die anhand der Fehlertabelle ausgewertet werden kann (siehe **GetErrorMessage**).

Aufrufbeispiel

keines

Voraussetzungen

	V5.6 ts_get_modem_name	V6.0 ts_get_modem_name_ex6
Windows:		2000, XP
Header:		PRODAVE6.H
Library:		PRODAVE6.DLL

Glossar

Automatisierungssystem (AS)

Ein Automatisierungssystem ist eine Speicherprogrammierbare Steuerung, die aus einem Zentralgerät, einer CPU und diversen Ein-/Ausgabegruppen besteht.

Datenbaustein (DB)

Datenbausteine sind Bausteine, die Daten und Parameter enthalten, mit denen das Anwenderprogramm arbeitet. Sie enthalten im Gegensatz zu allen anderen Bausteinen keine Anweisungen.

Multi Point Interface (MPI)

Die Mehrpunktfähige Schnittstelle ist die Programmiergeräte-Schnittstelle von SIMATIC S7/M7. Damit können von zentraler Stelle aus programmierbare Baugruppen, Text Displays und Operator Panels erreicht werden.

Die Teilnehmer an der MPI können miteinander kommunizieren.

Programmiergerät (PG)

Personal Computer in spezieller industrietauglicher und kompakter Ausführung. Ein PG ist komplett ausgestattet für die Programmierung der SIMATIC-Automatisierungssysteme.

Point-to-Point Interface (PPI)

Bei der Punkt-zu-Punkt-Kopplung bildet der Kommunikationsprozessor die Schnittstelle zwischen einer Speicherprogrammierbaren Steuerung und einem Kommunikationspartner.

Betriebszustand ANKOPPELN

Der Betriebszustand ANKOPPELN wird von der Reserve-CPU eines H-Systems erreicht, wenn das H-System sich im Systemzustand Ankoppeln befindet. Master-CPU und Reserve-CPU vergleichen den Speicherausbau und die Inhalte der Ladespeicher. Werden Unterschiede im Anwenderprogramm festgestellt, aktualisiert die Master-CPU das Anwenderprogramm der Reserve-CPU.

Betriebszustand ANLAUF

Der Betriebszustand ANLAUF wird beim Übergang vom Betriebszustand STOP in den Betriebszustand RUN durchlaufen. Kann ausgelöst werden durch:

- Betätigen des Betriebsartenschalter oder
- nach Netz-Ein oder
- durch Bedienung am Programmiergerät.

Man unterscheidet zwischen den Anlaufarten Kaltstart, Neustart und Wiederanlauf.

Betriebszustand AUFDATEN

Der Betriebszustand AUFDATEN wird von der Reserve-CPU eines H-Systems erreicht, wenn das H-System sich im Systemzustand Aufdaten befindet. Die Master-CPU aktualisiert die dynamischen Daten der Reserve-CPU.

Betriebszustand FEHLERSUCHE

Betriebszustand der Reserve-CPU eines H-Systems, in dem die CPU einen vollständigen Selbsttest durchführt. Gleichzeitig befindet sich die Master-CPU im Betriebszustand RUN.

Betriebszustand HALT

Der Betriebszustand HALT wird aus dem Betriebszustand RUN durch Anforderung vom PG erreicht. In diesem Betriebszustand sind spezielle Testfunktionen möglich.

Betriebszustand RUN

Im Betriebszustand RUN wird das Anwenderprogramm bearbeitet, das Prozessabbild wird zyklisch aktualisiert. Alle digitalen Ausgänge sind freigegeben.

Betriebszustand STOP

Der Betriebszustand STOP wird erreicht durch:

- Betätigung des Betriebsartenschalters
- durch einen internen Fehler auf der Zentralbaugruppe
- durch Bedienung am Programmiergerät

Alle Baugruppen werden in einen sicheren Zustand geschaltet.

Bei S7: Im Betriebszustand STOP wird das Anwenderporgramm nicht bearbeitet. Bestimmte Programmierfunktionen sowie Bedien- und Beobachtfunktionen sind aber möglich.

Bei M7: Im Betriebszustand STOP können Anwenderprogramme weiterbearbeitet werden.

Codebaustein

Ein Codebausteinen ist bei SIMATIC S7 ein Baustein, der einen Teil des STEP 7-Anwenderprogramms enthält. Im Gegensatz dazu enthält ein Datenbaustein nur Daten. Es gibt folgende Codebausteine: Organisationsbausteine (OB), Funktionsbausteine (FB), Funktionen (FC), Systemfunktionsbausteine (SFB) und Systemfunktionen (SFC).

Funktion (FC)

Eine Funktion (FC) ist gemäß IEC 1131-3 ein Codebaustein ohne statische Daten. Eine Funktion bietet die Möglichkeit der Übergabe von Parametern im Anwenderprogramm. Dadurch eignen sich Funktionen zur Parametrierung von häufig wiederkehrenden komplexen Funktionen, z. B. Berechnungen.

Funktionsbaustein (FB)

Ein Funktionsbaustein (FB) ist gemäß IEC 1131-3 ein Codebaustein mit statischen Daten (Daten, statisch). Da ein FB über ein Gedächtnis (Instanz-Datenbaustein) verfügt, kann auf seine Parameter (z. B. Ausgänge) zu jeder Zeit an jeder beliebigen Stelle im Anwenderprogramm zugegriffen werden.

Merker (M)

Speicherbereich im Systemspeicher einer SIMATIC S7-CPU. Auf ihn kann schreibend und lesend zugegriffen werden (bit-, byte-, wort- und doppelwortweise). Der Merkerbereich kann vom Anwender zum Speichern von Zwischenergebnissen verwendet werden.

Offline

Offline bezeichnet den Betriebszustand, bei dem das Programmiergerät keine Verbindung mit dem Automatisierungssystem hat (physikalisch, logisch).

Online

Online bezeichnet den Betriebszustand, bei dem das Programmiergerät mit dem Automatisierungssystem verbunden ist (physikalisch, logisch).

Organisationsbaustein (OB)

Organisationsbausteine bilden die Schnittstelle zwischen dem Betriebssystem der CPU und dem Anwenderprogramm. In den Organisationsbausteinen wird die Reihenfolge der Bearbeitung des Anwenderprogramms festgelegt.

Systemfunktion (SFC)

Eine Systemfunktion (SFC) ist eine im Betriebssystem der CPU integrierte Funktion, die bei Bedarf im STEP 7-Anwenderprogramm aufgerufen werden kann.

Systemfunktionsbaustein (SFB)

Ein Systemfunktionsbaustein (SFB) ist ein im Betriebssystem der CPU integrierter Funktionsbaustein, der bei Bedarf im STEP 7-Anwenderprogramm aufgerufen werden kann.

Systemdatenbaustein (SDB)

System-Datenbausteine sind Datenbereiche in der S7-CPU, die Systemeinstellungen und Baugruppenparameter enthalten. Die Systemdatenbausteine werden mit der STEP 7 Basissoftware erzeugt und geändert.

Zentralbaugruppe (CPU)

Die CPU (central processing unit) ist eine Zentralbaugruppe des Automatisierungssystems, in der das Anwenderprogramm gespeichert und bearbeitet wird. Sie beinhaltet Betriebssystem, Bearbeitungseinheit und Kommunikations-Schnittstellen.

Index