SIEMENS

SIMATIC NET

S7-Programmierschnittstelle

Band 1 von 1

C79000-G8900-C077 Ausgabe 03

SIEMENS

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 der Druckschrift werden jedoch regelmäßig überprüft. Notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen vorbehalten.

We have checked the contents of this manual for agreement with the hardware described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcome.

Technical data subject to change.

Nous avons vérifié la conformité du contenu du présent manuel avec le matériel et le logiciel qui y sont décrits. Or, des divergences n'étant pas exclues, nous ne pouvons pas nous porter garants pour la conformité intégrale. Si l'usage du manuel devait révéler des erreurs, nous en tiendrons compte et apporterons les corrections nécessaires dès la prochaine édition. Veuillez nous faire part de vos suggestions.
Nous nous réservons le droit de modifier les caractéristiques techniques.

Siemens Aktiengesellschaft

Elektronikwerk Karlsruhe
Printed in the Federal Republic of Germany

SIEMENS

SIMATIC NET

S7-Programmierschnittstelle

Beschreibung C79000-B8900-C077/03

Hinweis

Wir weisen darauf hin, daß der Inhalt dieser Betriebsanleitung nicht Teil einer früheren oder bestehenden Vereinbarung, Zusage oder eines Rechtsverhältnisses ist oder diese abändern soll. Sämtliche Verpflichtungen von Siemens ergeben sich aus dem jeweiligen Kaufvertrag, der auch die vollständige und allein gültige Gewährleistungsregel enthält. Diese vertraglichen Gewährleistungsbestimmungen werden durch die Ausführungen dieser Betriebsanleitung weder erweitert noch beschränkt.

Wir weisen außerdem darauf hin, daß aus Gründen der Übersichtlichkeit in dieser Betriebsanleitung nicht jede nur erdenkliche Problemstellung im Zusammenhang mit dem Einsatz dieses Gerätes beschrieben werden kann. Sollten Sie weitere Informationen benötigen oder sollten besondere Probleme auftreten, die in der Betriebsanleitung nicht ausführlich genug behandelt werden, können Sie die erforderliche Auskunft über die örtliche Siemens-Niederlassung anfordern.

Allgemeines

Dieses Gerät wird mit Elektrizität betrieben. Beim Betrieb elektrischer Geräte stehen zwangsläufig bestimmte Teile dieser Geräte unter gefährlicher Spannung.

WARNUNG !

!

Bei Nichtbeachtung der Warnhinweise können deshalb schwere Körperverletzungen und/oder Sachschäden auftreten.

Nur entsprechend qualifiziertes Personal sollte an diesem Gerät oder in dessen Nähe arbeiten. Dieses Personal muß gründlich mit allen Warnungen und Instandhaltungsmaßnahmen gemäß dieser Betriebsanleitung vertraut sein.

Der einwandfreie und sichere Betrieb dieses Gerätes setzt sachgemäßen Transport, fachgerechte Lagerung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Anforderung an die Qualifikation des Personals

Qualifiziertes Personal im Sinne dieser Betriebsanleitung bzw. der Warnhinweise sind Personen, die mit Aufstellung, Montage, Inbetriebsetzung und Betrieb dieses Produktes vertraut sind und über die ihrer Tätigkeit entsprechenden Qualifikation verfügen, wie z.B.:

- Ausbildung oder Unterweisung bzw. Berechtigung, Stromkreise und Geräte bzw. Systeme gemäß den aktuellen Standards der Sicherheitstechnik ein- und auszuschalten, zu erden und zu kennzeichnen
- Ausbildung oder Unterweisung gemäß an den aktuellen Standards der Sicherheitstechnik in Pflege und Gebrauch angemessener Sicherheitsausrüstungen
- Schulung in Erster Hilfe

S7-Programmierschnittstelle

SIMATIC S7-Systemkomponenten kommunizieren miteinander über das S7-Kommunikationsprotokoll. Um PG-/PC-Anwenderprogrammen den Zugang zu SIMATIC S7-Systemkomponenten zu ermöglichen, wurde die SAPI-S7-Programmierschnittstelle entwickelt. Mit ihrer C-Programmierschnittstelle können Sie einfach und flexibel auf die Daten von SIMATIC S7-Systemkomponenten zugreifen.

Der vorliegende Band beschreibt das S7-Protokoll und dessen Programmierung.

Notizen

1 Die SAPI-S7-Schnittstelle

Im vorliegenden Kapitel lernen Sie die Vorteile des S7-Kommunikationsprotokolls gegenüber anderen Protokollen und die S7-Programmierschnittstelle für PC/PG (SAPI-S7) kennen.

1.1 Vorteile von S7 gegenüber anderen Protokollen

Vorteile von S7

Die Vorteile von S7 gegenüber anderen Protokollen liegen in der

- ➢ geringen Prozessor- und Busbelastung.
  Das S7-Protokoll ist für die SIMATIC-Kommunikation optimiert.
- ➢ Einfachheit.
  Die S7-Protokollelemente sind an die SIMATIC-Bedürfnisse angepaßt und daher einfach anwendbar.
- ➢ Schnelligkeit.
  S7 ist im Vergleich zu anderen Ebene-7-Automatisierungsprotokollen, wie z. B. MMS, sehr performant.

1.2 Vorteile der SAPI-S7-Programmierschnittstelle

Was bedeutet SAPI-S7?

Die Abkürzung SAPI-S7 steht für:

- SAPI - **S**imple **A**pplication **P**rogrammers **I**nterface - einfache Programmierschnittstelle,
- S7 - das Ebene-7-Kommunikationsprotokoll für SIMATIC S7-Systeme.

Was ist SAPI-S7?

SAPI-S7

- stellt eine einfache C-Programmierschnittstelle dar,
- bietet den Zugang zu den S7-Diensten auf PCs und PGs und
- ist verfügbar als C-Library und wird mit Siemens-Treibern und -Anschaltungen betrieben.

Vorteile der SAPI-S7-Programmier-schnittstelle

Die SAPI-S7-Programmierschnittstelle bietet die nachfolgenden Vorteile:

- ➢ SAPI-S7 ist eine einfache und zugleich flexible und mächtige Schnittstelle, die pro Auftrag nur einen Übergabeparameterblock benötigt und eine einfache Pufferverwaltung ermöglicht. SAPI-S7 bietet die Voraussetzungen für geringe Einarbeitungsaufwände und -zeiten.
- ➢ Die SAPI-S7-Programmierschnittstelle ist asynchron ausgelegt, da asynchrone Aufrufe performanter sind (mehrere Aufträge können gleichzeitig bearbeitet werden), jederzeit den Empfang von Statusmeldungen erlauben und sich ideal für die Erstellung ereignisgesteuerter Programme, z. B. unter Windows, eignen.
- ➢ SAPI-S7 wickelt Sequenzdienste, wie den aktiven oder passiven S7-Verbindungsaufbau, der aus vielen Einzelschritten besteht, automatisch ab. Man muß sich deshalb nicht um die einzelnen Sequenzen und Fehlerbehandlung in den Einzelschritten kümmern. Eine Programmerstellung wird damit erheblich einfacher.
- ➢ SAPI-S7 unterstützt die Fehlersuche mit Hilfe eines integrierten Trace, der die wichtigsten Übergabeparameter und Rückgabewerte in eine Datei schreibt. Der Trace ist für jede einzelne Dienstklasse ein- und ausschaltbar.
- ➢ Die SAPI-S7-Programmierschnittstelle ist kompatibel zu anderen SAPI-Programmierschnittstellen, d. h. der Portierungsaufwand von SAPI-S7-Applikationen zu Applikationen anderer SAPI-Programmierschnittstellen (z. B. SAPI-FMS) ist gering.
- ➢ Die Strukturen der SAPI-S7-Programmierschnittstelle sind so ausgelegt, daß mit Byte- oder Word-Alignment übersetzte Anwenderprogramme problemlos auf die einzelnen Komponenten zugreifen können. Damit sind auch die Voraussetzungen für die Nutzung, z. B. durch BASIC- oder Pascal-Programme, geschaffen.

2 Prinzipien der Programmierschnittstelle

Im folgenden Kapitellernen Sie die Prinzipien der SAPI-S7-Programmierschnittstelle kennen:

- ➢ Mit asynchronen Aufrufen können Sie eine Performance-Steigerung erreichen und ereignisgesteuerte Applikationen erstellen.
- ➢ Durch die Trennung von Empfangsaufrufen und ereignisspezifischen Bearbeitungsfunktionen wird die SAPI-S7-Programmierschnittstelle vereinfacht.
- ➢ Durch die Library-interne Abarbeitung von Sequenzdiensten werden Anwenderprogramme vereinfacht.
- ➢ Durch eine An- und eine Abmeldefunktion werden der Multi-CP- und der Multi-User-Betrieb gewährleistet.
- ➢ Durch einen integrierten Trace können Sie zu Debug-Zwecken den Programmablauf und die Funktionsfähigkeit der Library und der Anwenderprogramme kontrollieren.

Am Ende des Kapitels sind Sie mit den Prinzipien der S7-Programmierschnittstelle soweit vertraut, daß Sie die Schnittstelle zur Programmerstellung verstehen können.

2.1 Synchrone und asynchrone Auftragsabwicklung

Synchrone Auftragsabwicklung

Bei der synchronen Auftragsabwicklung (Bild 2.1) bleibt ein Anwenderprogramm vom Zeitpunkt des Auftrags (Request) bis zum Zeitpunkt der Auftragsbestätigung (Confirmation) blockiert. Zwischenzeitlich eintreffende Ereignisse können nur bearbeitet werden, falls sie interruptgesteuert ablaufen. Diese Betriebsart ist bei SAPI-S7 nicht implementiert.

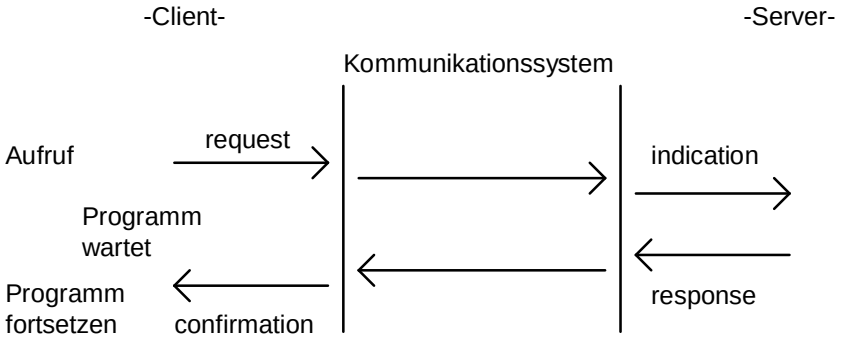

Bild 2.1: synchrone Betriebsart

Asynchrone Auftragsabwicklung

Bei der asynchronen Auftragsabwicklung (Bild 2.2) kann ein Anwenderprogramm nach einem erfolgreichen Auftrag jedes beliebige Ereignis entgegennehmen und geeignet darauf reagieren. Die Auftragsbestätigung erfolgt erst mit einem expliziten Empfangsaufruf.

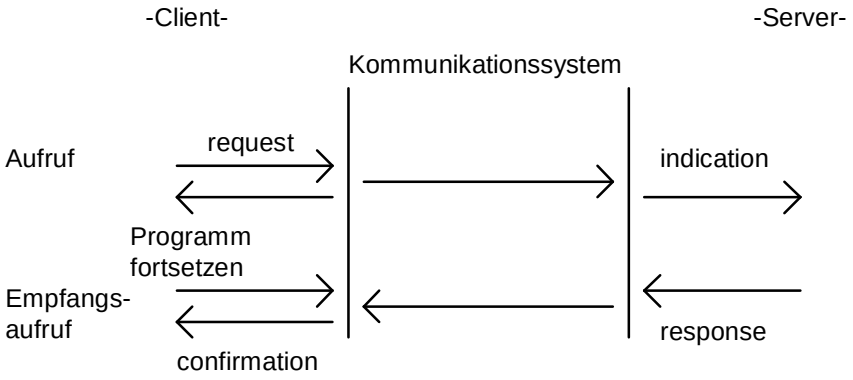

Bild 2.2: asynchrone Betriebsart

2.2 Vorteile der asynchronen Betriebsart

Asynchrone Aufträge als Grundprinzip der SAPI-S7-Programmierschnittstelle

Die SAPI-S7-Programmierschnittstelle ist asynchron ausgelegt,

- ➢ da mit asynchronen Aufrufen ein höherer Datendurchsatz erreicht werden kann (mehrere Aufträge können gleichzeitig bearbeitet werden);
- ➢ damit die Applikation nicht blockiert wird und sich um andere Dinge, wie z. B. den Empfang von Statusmeldungen, kümmern kann;
- ➢ da asynchrone Mechanismen sich ideal für die Erstellung ereignisgesteuerter Programme, z. B. unter Windows, eignen.

Aus Sicht der Anwendung ist zu entscheiden, ob die angesprochenen Dienste voneinander abhängen. Ist dies der Fall, so muß die Durchführung eines Auftrags bestätigt werden, bevor der nächste Auftrag abgesetzt werden kann. Beispielsweise muß eine Verbindung aufgebaut sein, bevor über diese Verbindung Daten transferiert werden können.

2.3 Empfangsaufruf und Bearbeitungsfunktionen

Wie wird eine empfangene Nachricht bearbeitet?

Eine von den unterlagerten Protokollschichten empfangene Nachricht wird vom Anwenderprogramm in den beiden folgenden Schritten abgearbeitet (Bild 2.3):

- ➢ Die Funktion 's7_receive()' holt zunächst eine vorhandene Nachricht ab. Der Return-Wert gibt Auskunft über das empfangene Ereignis. In der Header-Datei 'SAPI_S7.H' ist je Ereignis eine Konstante definiert, z. B. zeigt der Wert 'S7_INITIATE_IND' an, daß die Partnerstation den Aufbau einer S7-Verbindung wünscht. Bei 'S7_NO_MSG' wurde keine Nachricht empfangen.
- ➢ Im nächsten Schritt rufen Sie die entsprechende Bearbeitungsfunktion für die empfangene Nachricht auf: das mit 'S7_INITIATE_IND' gekennzeichnete Ereignis bedarf des Aufrufs der Bearbeitungsfunktion 's7_get_initiate_ind()'.

Mit Hilfe der Empfangsfunktion kann pollend auf ein Ereignis gewartet werden. Zwischen Empfang und Aufruf der Bearbeitungsfunktion darf allerdings kein weiteres Ereignis beim Kommunikationssystem abgeholt werden.

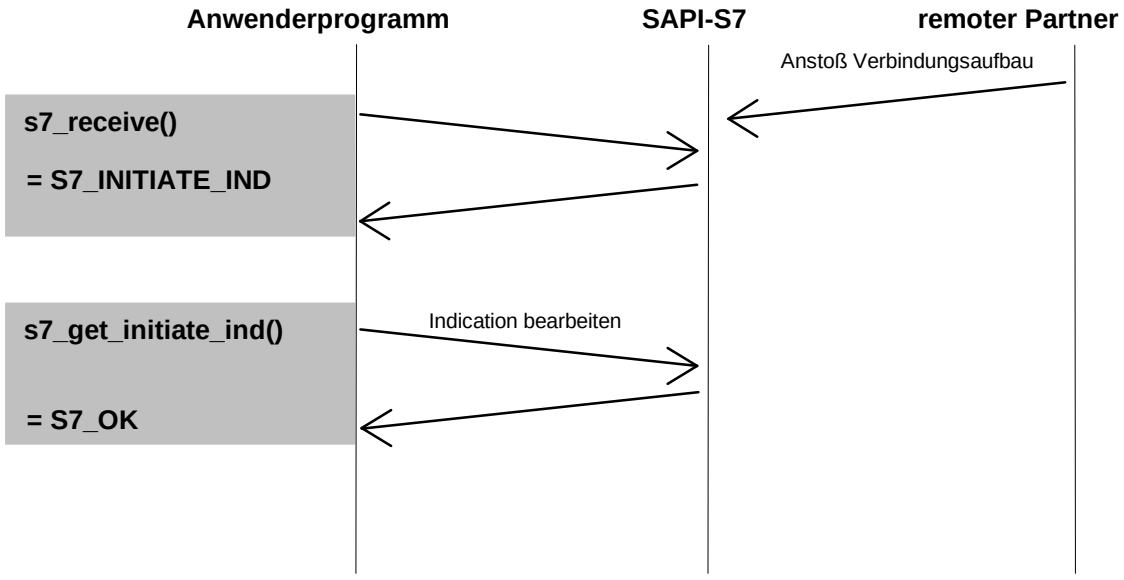

Bild 2.3: Nachrichten empfangen und bearbeiten

Vorteile der Aufteilung

Eine Aufteilung der Ereignisbearbeitung in eine Empfangsfunktion und eine ereignisspezifische Bearbeitungsfunktion:

- ➢ erlaubt eine einfachere und übersichtlichere Programmierschnittstelle. Nur die für das empfangene Ereignis relevanten Daten und Parameter werden den Funktionen übergeben,
- ➢ ermöglicht eine problemlose Erweiterung der SAPI-S7-Programmierschnittstelle um zusätzliche Ereignisse durch Bereitstellen von entsprechenden Bearbeitungsfunktionen,
- ➢ bewirkt eine Trennung von Indication (Empfangsaufruf) und Response (Bearbeitungsfunktion). Dadurch ist es möglich, Anwenderdaten für die Response aufzubereiten.

2.4 Handhabung der S7-Verbindungsmanagement-Dienste

Unterstützung durch die S7-Library

Die SAPI-S7-Library unterstützt Sie beim aktiven und passiven Verbindungsaufbau, der sich aus vielen Einzelschritten zusammensetzt. Bei jedem Schritt wird auf Erfolg und Mißerfolg geprüft. Im Fehlerfall wird sichergestellt, daß schon zuvor erfolgreich durchgeführte Einzelschritte korrigiert werden. Sie müssen hierzu nach Absetzen des entsprechenden Requests lediglich die Empfangsfunktion mehrfach bis zum Erhalt der Abschlußquittung aufrufen; Bild 2.4 zeigt diesen Ablauf.

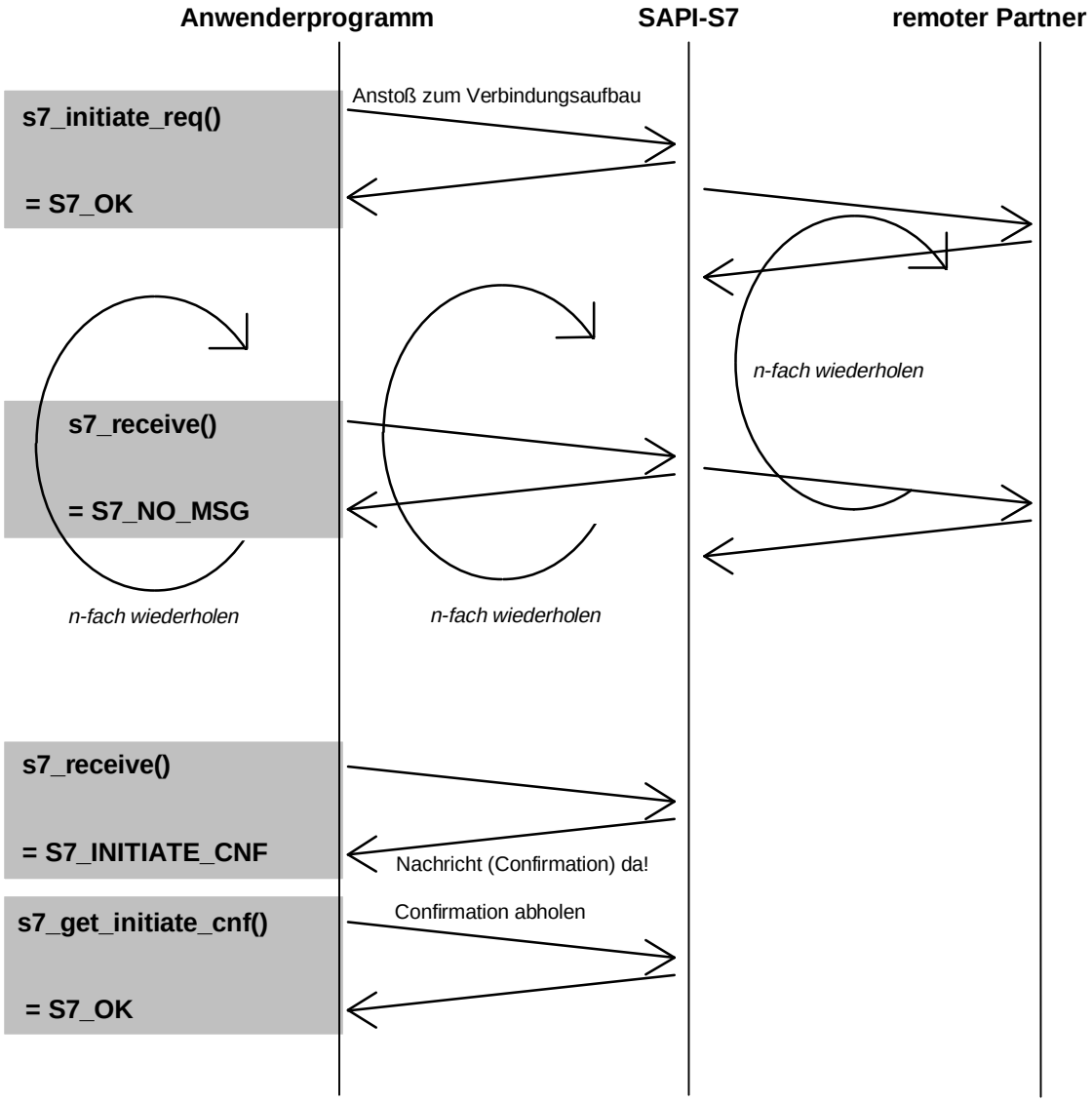

Bild 2.4: aktiver Verbindungsaufbau

2.5 Fehlermeldekonzept

Das dreistufige Fehlermelde-konzept der SAPI-S7-Programmier-schnittstelle

Die SAPI-S7-Programmierschnittstelle unterstützt ein dreistufiges Fehlermeldekonzept:

- ➢ Erfolg oder Mißerfolg werden zwecks einfacher Fehlerbehandlung lediglich durch den Return-Wert des Aufrufes angezeigt.
- ➢ Die Fehlerursachen sind nach IEC 1131 (**I**nternational **E**lectrotechnical **C**ommission) genormt und reduzieren die Anzahl der Fehlerkennungen auf ein handhabbares Maß.
- ➢ Für genauere Fehleranalysen stehen detailliertere Fehlercodes und -meldungen zur Verfügung.

Die Return-Werte der SAPI-S7-Programmier-schnittstelle

Für die Return-Werte der einzelnen SAPI-S7-Aufrufe stehen in der Header-Datei 'SAPI_S7.H' Defines bereit:

- ➢ Der Wert 'S7_OK' zeigt die fehlerfreie Durchführung eines Aufrufes an.
- ➢ Mit dem Wert 'S7_ERR_RETRY' wird ein Fehler bei der Ausführung des angeforderten Dienstes angezeigt. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
- ➢ Mit dem Wert 'S7_ERR' wird ebenfalls ein Fehler bei der Ausführung des angeforderten Dienstes angezeigt. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

Die IEC-Fehlerkennungen

Die S7-Library stellt die Funktionen 's7_last_iec_err_no()' und 's7_last_iec_err_msg()' zum Auslesen einer Fehlernummer und einer Fehlermeldung bereit. Diese Fehlerursachen sind nach IEC 1131 (**I**nternational **E**lectrotechnical **C**ommission) genormt und reduzieren die Anzahl der Fehlerkennungen auf ein handhabbares Maß.

Die detaillierten Fehlercodes

Für detailliertere Fehleranalysen sind an der SAPI-S7-Programmierschnittstelle die Funktionen 's7_last_detailed_err_no()' und 's7_last_detailed_err_msg()' realisiert. Sie beschreiben die Fehlerquellen genauer und liefern auch Hinweise zur Behebung der aufgetretenen Fehler.

2.6 Der Trace

Was versteht man unter dem Trace?

Der Trace stellt eine einfache und trotzdem effektive Debug-Hilfe für die S7-Library dar. Dadurch werden Daten und Ereignisse in einen Umlaufpuffer und eine Trace-Datei abgelegt.

Der Trace kann den verschiedensten Anwendungsfällen angepaßt werden:

- ➢ der Name der Trace-Datei kann beliebig vorgegeben werden,
- ➢ die Dienstklassen, für die der Trace aktiviert werden soll, können eingestellt werden,
- ➢ die Trace-Tiefe (Trace ausgeschaltet, Trace nur für negative Ereignisse eingeschaltet etc.) kann bestimmt werden,
- ➢ das Target des Trace gibt an, ob eine Trace-Datei neu angelegt, eine alte Trace-Datei verwendet oder in den internen Umlaufpuffer geschrieben werden soll.

Nutzen des Trace

Während des laufenden Betriebs schreibt die S7-Library wichtige Daten in die Trace-Datei. Der Inhalt der Datei

- ➢ dient als Protokoll über sämtliche Aktionen, die von der S7-Library durchgeführt wurden,
- ➢ erlaubt eine Ablaufkontrolle sowohl der Applikation als auch der Library selbst,
- ➢ ermöglicht ein einfaches Debuggen und Überprüfen der Daten und Parameter an der SAPI-S7-Programmierschnittstelle.

Weiter können Sie die Trace-Datei selbst beschreiben. Durch solche Eintragungen ist es möglich, wichtige Daten zu Kontrollzwecken zu sichern, den Programmablauf zu kontrollieren bzw. mit den Eintragungen durch die Library zu synchronisieren.

Zeitverhalten

Der Trace mit der Ablage der Daten in die Trace-Datei verlangsamt das Anwendungsprogramm. Deshalb gibt es zusätzlich die Möglichkeit, den Umlaufpuffer als Schnappschuß in eine Datei zu schreiben.

2.7 Die Mini-DB

Zweck der Mini-DB

Mit Hilfe der Mini-DB (Mini-**D**aten**b**asis) können Abweichungen vom Standard-Betriebsfall eingestellt und detaillierte Informationen ausgelesen werden. Die Mini-DB stellt einen Datenbereich innerhalb der S7-Library dar,

- ➢ in dem immer wieder (in der Regel unverändert) benötigte Daten abgelegt sind,
- ➢ in dem protokollspezifische Daten abgelegt sind bzw. im laufenden Betrieb abgelegt werden,
- ➢ in dem die Kennungen des zuletzt aufgetretenen Fehlers hinterlegt werden,
- ➢ der durch Funktionen wie 's7_mini_db_get()', 's7_mini_db_set()' etc. auslesbar und veränderbar ist.

Vorteile der Mini-DB

Die S7-Library ermöglicht einen Betrieb mit Default-Einstellungen für den Trace, den Aufbau von S7-Verbindungen (aktiv und passiv) etc. Diese Default-Einstellungen sind für einen Großteil der Anwendungsfälle ausreichend, können aber durch einen Mini-DB-Aufruf jederzeit geändert und an die jeweiligen Anforderungen angepaßt werden.

Die Verwendung einer Mini-DB in der S7-Library bietet folgende Vorteile:

- ➢ Die Programmierschnittstelle kann auf wenige Übergabeparameter vereinfacht werden. Die (eventuell immer unverändert) benötigten Werte werden aus der Mini-DB gelesen. Dies erhöht die Performance und vereinfacht das Handling der SAPI-S7-Programmierschnittstelle.
- ➢ Die hohe Flexibilität, die das Protokoll S7 bietet, bleibt erhalten. Eine Änderung der Einstellungen und deren Anpassung an die eigenen Bedürfnisse ist jederzeit durch Mini-DB-Aufrufe möglich.

2.8 Multi-CP- und Multi-User-Betrieb

Multi-CP-Betrieb

Unter einem Multi-CP-Betrieb versteht man die Fähigkeit, von einer Applikation auf einem Rechner

- ➢ mehrere CPs und
- ➢ die mit den verschiedenen CPs per Vernetzung verbundenen Partnerstationen

ansprechen zu können.

Multi-User-Betrieb

Bei einem Multi-User-Betrieb können die angebotenen Dienste und Ressourcen von mehreren Applikationen eines Rechners gleichzeitig, d. h. ohne gegenseitige Beeinflussung, in Anspruch genommen werden.

Voraussetzungen

Voraussetzung für einen Multi-CP- und Multi-User-Betrieb ist, daß Requests und Confirmations einerseits sowie Indications und Responses andererseits unmißverständlich einander und auch der zugehörigen Applikation zugeordnet werden können.

Diese Voraussetzung wird erreicht durch eine Anmeldefunktion der SAPI-S7-Programmierschnittstelle. Eine Anmeldung erfolgt

- ➢ für einen CP und
- ➢ für ein VFD dieses CP.

Mit dem VFD sind die VFD-eigenen S7-Verbindungen für die Applikation verfügbar (Bild 2.5). Eine weitere Anmeldung für das VFD ist weder von dieser noch von einer anderen Applikation aus möglich, während eine Anmeldung für den CP durchaus mehrfach möglich ist.

2.8.1 Zuordnung von VFD und S7-Verbindungsliste

Zusammenhang VFD und S7-Verbindungsliste

Eine Applikation kann sich bei mehreren VFDs auf einem oder auf verschiedenen CPs anmelden. Multi-CP- und Multi-User-Betrieb setzen voraus, daß ein VFD nach Anmeldung eindeutig einer Applikation zugeordnet werden kann (1 : n - Zuordnung). Mit der Anmeldung beim CP und bei der ausgewählten VFD stehen aus der S7-Verbindungsliste des CPs diejenigen Verbindungen zur Verfügung, die durch Projektierung dem VFD zugeordnet sind. Beispielsweise ist im nachfolgenden Bild eine Kommunikation über die Verbindungen ‘V1’, ‘V2’ und ‘V3’ nach Anmeldung bei ‘CP 1’ und ‘VFD 1’ möglich.

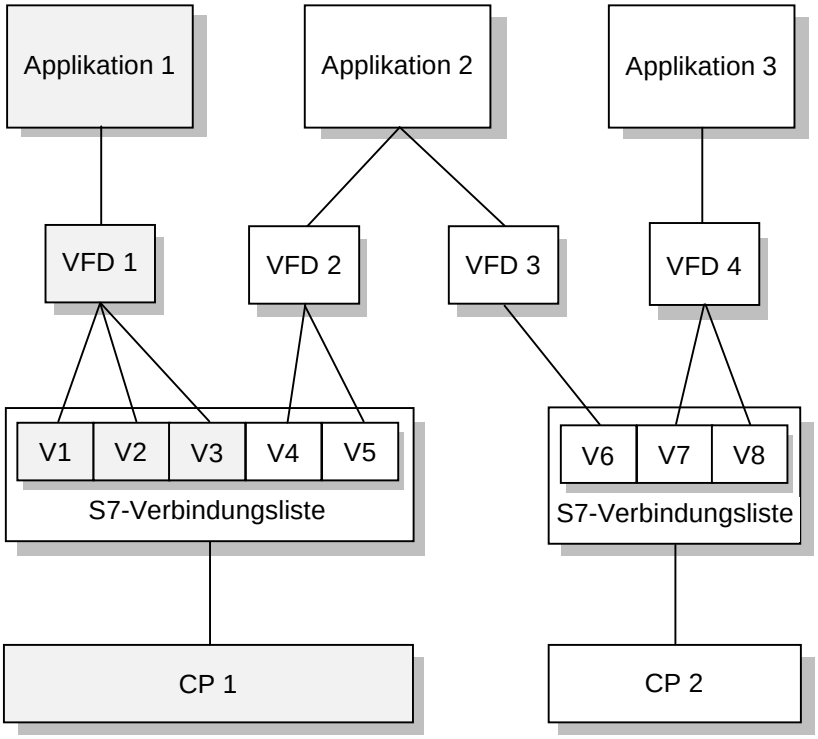

Bild 2.5: Zuordnung der Liste der S7-Verbindungen bei einer Anmeldung an VFD 1 auf CP 1

2.9 Installation und Betriebsvoraussetzungen

Installation

Die zur Installation notwendigen Arbeitsschritte sind den jeweiligen Produktbeschreibungen zu entnehmen und nicht Gegenstand dieser Unterlage.

Betriebsvoraus-setzungen

Für den Betrieb der S7-Library muß sich das Kommunikationssystem in einem betriebsbereiten Zustand befinden, d. h. VFD und S7-Verbindungslisten müssen projektiert worden sein. Diese Aufgaben werden nicht von der SAPI-S7-Programmierschnittstelle übernommen.

3 Die Programmierschnittstelle

Im folgenden Kapitel lernen Sie die praktische Anwendung der S7-Programmierschnittstelle für die Programmiersprache 'C' kennen.

Den Umgang mit den Aufrufen der Programmierschnittstelle, die Ihnen den Zugang zu den Diensten und Objekten von S7 ermöglichen, lernen Sie in den folgenden Unterkapiteln anhand von Beispielprogrammen kennen. Die Beispiele

- beschreiben eindeutig und übersichtlich die einzuhaltenden Reihenfolgen für einen erfolgreichen Kommunikationsverlauf,
- stellen die Programmierschnittstelle Schritt für Schritt vor,
- bauen aufeinander auf, wobei neue Funktionen oder veränderte Programmsegmente des vorhergehenden Beispiels **fettgedruckt** dargestellt werden,
- realisieren eine Fehlerbehandlungsroutine, die von Ihnen auf Ihre jeweiligen Anforderungen angepaßt werden sollte.

Die S7-Funktionen werden in den Beispielprogrammen nicht direkt aufgerufen, sondern in eigene Funktionen (z. B. 'my_init()' für 's7_init()') gekapselt. Dies ist nicht zwingend erforderlich, macht das Programm aber besser geeignet für Debugging, Fehlerausgaben und nachträgliche Erweiterungen.

Am Ende dieses Kapitels wissen Sie,

- welche Dienste im einzelnen auf einem Host-System zur Verfügung stehen,
- für welche Aufgaben die einzelnen Dienste zu verwenden sind,
- welche Kommunikationsabläufe durch eine Dienstanforderung und Dienstbestätigung entstehen,
- welche Aufruf- und Ablaufstruktur Sie allgemein an der S7-Programmierschnittstelle vorfinden.

Sie sind dann mit den Grundlagen der S7-Programmierschnittstelle soweit vertraut, daß Sie die Programmierung einer SAPI-S7-Anwendung angehen können.

3.1 Übersicht über die Programmierschnittstelle

Einteilung der Dienste

Die S7-Programmierschnittstelle bietet folgende Dienste:

- ➢ Administrative Dienste
- ➢ Empfangsaufrufdienst
- ➢ S7-Verbindungsmanagementdienste
- ➢ Variablendienste
- ➢ Blockorientierte Dienste
- ➢ VFD-Dienste

3.1.1 Administrative Dienste

Beschreibung

Als administrative Dienste stehen Funktionen zum Auslesen von Projektierinformationen und zum An- und Abmelden beim Kommunikationssystem zur Verfügung. Außerdem gibt es Funktionen, die Referenzen für symbolische S7-Verbindungsnamen liefern.

Die folgende Tabelle gibt einen Überblick über die administrativen Dienste. Eine genaue Beschreibung finden Sie in Kapitel 3.2 auf Seite 23.

s7_get_device()	Mit dieser Funktion können alle installierten CP-Namen abgefragt werden.
s7_get_vfd()	Mit dieser Funktion können alle projektierten VFDs eines CP abgefragt werden.
s7_init()	Mit dieser Funktion meldet sich das Anwenderprogramm beim Kommunikationssystem an.
s7_get_cref()	Diese Funktion liefert eine Referenz zu einem symbolischen S7-Verbindungsnamen. Dieser Wert selektiert die reale Verbindung im Netz und ist handlicher als der symbolische Name.
s7_get_conn()	Diese Funktion liefert alle symbolischen S7-Verbindungsnamen beim angemeldeten VFD sowie deren Referenz zurück.
s7_shut()	Mit dieser Funktion meldet sich das Anwenderprogramm beim Kommunikationssystem ab.

3.1.2 Empfangsaufrufdienst

Beschreibung Dienst zum Abholen von Nachrichten.

Eine genaue Beschreibung finden Sie in Kapitel 3.3.1 auf Seite 23.

s7_receive()	Mit dieser Funktion holt sich das Anwenderprogramm empfangene Nachrichten beim Kommunikations-system ab.

3.1.3 S7-Verbindungsmanagementdienste

Beschreibung Die S7-Verbindungsmanagementdienste werden zum Auf- und Abbau von S7-Verbindungen benötigt.

Aktiver Verbindungsaufbau Die folgende Tabelle gibt einen Überblick über die Verbindungsmanagementdienste beim aktiven Verbindungsaufbau. Eine genaue Beschreibung finden Sie ab Kapitel 3.4.1 auf Seite 23.

s7_initiate_req()	Diese Funktion leitet einen Auftrag zum aktiven Aufbau einer S7-Verbindung an das Kommunikationssystem weiter.
s7_get_initiate_cnf()	Diese Funktion nimmt das Ergebnis des obigen Aufrufs entgegen.

Passiver Verbindungsaufbau

Die folgende Tabelle gibt einen Überblick über die Verbindungsmanagementdienste beim passiven Verbindungsaufbau. Eine genaue Beschreibung finden Sie ab Kapitel 3.4.3 auf Seite 23.

s7_await_initiate_req()	Diese Funktion bereitet das Kommunikationssystem zum Empfang eines passiven S7-Verbindungsaufbauwunsches vor.
s7_get_await_initiate_cnf()	Diese Funktion nimmt das Ergebnis des obigen Aufrufs entgegen.
s7_get_initiate_ind()	Diese Funktion schließt die Entgegennahme einer Initiate Indication ab.
s7_initiate_rsp()	Der Aufbauwunsch kann mit dieser Funktion akzeptiert oder auch abgelehnt werden.

Verbindungsabbruch

Die folgende Tabelle gibt einen Überblick über die Verbindungsmanagementdienste bei Verbindungsabbruch. Eine genaue Beschreibung finden Sie ab Kapitel 3.4.7 auf Seite 23.

s7_abort()	Diese Funktion bricht eine aufgebaute S7-Verbindung ab.
s7_get_abort_ind()	Diese Funktion schließt die Entgegennahme einer Abort Indication ab.

3.1.4 Variablendienste

Beschreibung

Bei den Variablendiensten stehen Funktionen zum Lesen und Schreiben einer oder mehrerer Variablen zur Verfügung.

Lesen und Schreiben einer Variablen

Die folgende Tabelle gibt einen Überblick über die Variablendienste beim Lesen und Schreiben einer Variablen. Eine genaue Beschreibung finden Sie ab Kapitel 3.5.1 auf Seite 23.

s7_read_req()	Mit dieser Funktion wird ein Variablen-Leseauftrag eingeleitet.
s7_get_read_cnf()	Diese Funktion nimmt den gelesenen Variablenwert entgegen.
s7_write_req()	Diese Funktion leitet einen Variablen-Schreibauftrag ein.
s7_get_write_cnf()	Diese Funktion nimmt das Ergebnis des obigen Aufrufs entgegen.

Lesen und Schreiben mehrerer Variablen

Die folgende Tabelle gibt einen Überblick über die Variablendienste beim Lesen und Schreiben mehrerer Variablen. Eine genaue Beschreibung finden Sie ab Kapitel 3.5.5 auf Seite 23.

s7_multiple_read_req()	Mit dieser Funktion wird ein Auftrag zum Lesen mehrerer Variablen eingeleitet.
s7_get_multiple_read_cnf()	Diese Funktion nimmt die gelesenen Variablenwerte entgegen.
s7_multiple_write_req()	Diese Funktion leitet einen Auftrag zum Schreiben mehrerer Variablen ein.
s7_get_multiple_write_cnf()	Diese Funktion nimmt die Ergebnisse des obigen Aufrufs entgegen.

Zyklisches Lesen mit mehreren Aufrufen anstoßen

Die folgende Tabelle gibt einen Überblick über die Variablendienste beim zyklischen Lesen mit mehreren Aufrufen. Eine genaue Beschreibung finden Sie ab Kapitel 3.5.9 auf Seite 23.

s7_cycl_read_init_req()	Mit dieser Funktion wird der Server veranlaßt, ein zyklisches Lesen von Variablen vorzubereiten.
s7_get_cycl_read_init_cnf()	Diese Funktion nimmt das Ergebnis des obigen Aufrufs entgegen.
s7_cycl_read_start_req()	Mit dieser Funktion wird der Server veranlaßt, ein zyklisches Lesen von Variablen zu starten.
s7_get_cycl_read_start_cnf()	Diese Funktion nimmt das Ergebnis des obigen Aufrufs entgegen.

Zyklische Daten entgegennehmen

Die folgende Tabelle gibt einen Überblick über die Variablendienste beim zyklischem Daten entgegennehmen. Eine genaue Beschreibung finden Sie ab Kapitel 3.5.13 auf Seite 23.

s7_get_cycl_read_ind()	Mit dieser Funktion werden die vom Server gesendeten Daten entgegengenommen.

Zyklisches Lesen anhalten und beenden

Die folgende Tabelle gibt einen Überblick über die Variablendienste beim Anhalten und Beenden des zyklischen Lesens. Eine genaue Beschreibung finden Sie ab Kapitel 3.5.14 auf Seite 23.

s7_get_cycl_read_abort_ind()	Diese Funktion schließt die Entgegennahme einer Cyclic Read Abort Indication ab.
s7_cycl_read_stop_req()	Mit dieser Funktion wird der Server veranlaßt, ein zyklisches Lesen von Variablen anzuhalten.
s7_get_cycl_read_stop_cnf()	Diese Funktion nimmt das Ergebnis des obigen Aufrufs entgegen.
s7_cycl_read_delete_req()	Mit dieser Funktion wird das zyklische Lesen abgebrochen und beim Server abgemeldet.
s7_get_cycl_read_delete_cnf()	Diese Funktion nimmt das Ergebnis des obigen Aufrufs entgegen.

Zyklisches Lesen mit einem Anruf anstoßen

Die folgende Tabelle gibt einen Überblick über den Variablendienst „Zyklisches Lesen mit einem Anruf anstoßen“. Eine genaue Beschreibung finden Sie ab Kapitel 3.5.19 auf Seite 23.

s7_cycl_read()	Mit dieser Funktion wird der Server veranlaßt, ein zyklisches Lesen von Variablen vorzubereiten und sofort zu starten.

3.1.5 Blockorientierte Dienste

Beschreibung

Die blockorientierten Diensten bieten Funktionen zum Datenaustausch von bis zu 65534 Byte. Sie setzen eine zweiseitige Verbindungsprojektierung voraus (STEP 7, COML S7).

Die folgende Tabelle gibt einen Überblick über die blockorientierten Dienste. Eine genaue Beschreibung finden Sie ab Kapitel 3.6.1 auf Seite 23.

s7_bsend_req()	Mit dieser Funktion kann eine Client-Applikation bis zu 64 KByte Daten an eine remote Station senden.
s7_get_bsend_cnf()	Mit dieser Funktion wird das Ergebnis des BSEND-Auftrags entgegengenommen.
s7_brcv_init()	Mit dieser Funktion wird dynamisch Puffer bereitgestellt, um für den Empfang von BSEND-Daten bereit zu sein, die von der remoten Station gesendet werden.
s7_get_brcv_ind()	Mit dieser Funktion werden die vom Partner gesendeten Nettodaten in den angegebenen Speicherbereich kopiert.
s7_brcv_stop()	Mit dieser Funktion wird der von s7_brcv_init belegte Puffer wieder freigegeben, d. h. die Kommunikation zum remoten BSEND ist nicht mehr möglich.

3.1.6 VFD-Dienste

Beschreibung

Die folgende Tabelle gibt einen Überblick über die VFD-Dienste. Eine genaue Beschreibung finden Sie ab Kapitel 3.7.1 auf Seite 23.

s7_vfd_state_req()	Mit dieser Funktion wird die Abfrage des Geräte-/Anwenderstatus initiiert.
s7_get_vfd_state_cnf()	Diese Funktion nimmt den Geräte-/Anwenderstatus entgegen.
s7_get_vfd_ustate_ind()	Mit dieser Funktion wird der vom Server unaufgefordert gesendete Geräte-/Anwenderstatus entgegengenommen.

3.2 Administrative Dienste

Beschreibung des Beispiels

Das nachfolgende Beispiel beschreibt die administrativen Dienste zum An- ('**s7_init()**') und Abmelden ('**s7_shut()**') einer S7-Applikation beim Kommunikationssystem. Eine Kommunikation mit einem lokalen CP und damit auch mit einem remoten CP ist erst nach erfolgreichem Anmelden möglich. Da bei der Anmeldung beim lokalen VFD die VFD-eigenen Ressourcen für die anmeldende Applikation reserviert werden, muß vor dem Programmende für jede Anmeldung auch eine Abmeldung erfolgen.

Die zum Anmelden benötigten Übergabeparameter, wie der CP-Name und der VFD-Name, können mit Hilfe der Funktionen **'s7_get_device()'** bzw. mit **'s7_get_vfd()'** vom Kommunikationssystem erfragt werden.

Beispiel

```
#include "sapi_s7.h"

/* prototypings */
static void my_exit(ord32 cp_descr,char *msg,int32 ret);
static void my_get_cref(ord32 cp_descr,ord16 *cref_ptr);
static void my_init(ord32 *cp_descr_ptr);
static void my_shut(ord32 cp_descr);

/* exit application */
static void my_exit(ord32 cp_descr,char *msg,int32 ret)
{
      printf("\n%s = %lx",msg,ret);
      my_shut(cp_descr);
}

/* get reference for connection 'TEST' */
static void my_get_cref(ord32 cp_descr,ord16 *cref_ptr)
{     int32 ret;

      ret=s7_get_cref(cp_descr,"TEST",cref_ptr);
      if(ret!=S7_OK)
      {
            my_exit(cp_descr,"Error s7_get_cref",ret);
      }
}

/* initialize s7 */
static void my_init(ord32 *cp_descr_ptr)
{     int32 ret;
      char vfd_name[S7_MAX_NAMLEN+1];
      char dev_name[S7_MAX_DEVICELEN+1];
      ord16 number;

      /* only use first device */
       ret=s7_get_device(0,&number,dev_name);
      if((ret!=S7_OK)||(number==0))
      {     /* something has gone wrong */
            printf("\ns7_get_device = %lx, number = %d", ret,number);
            exit(-1);
      }

      /* only use first vfd of first device */
      ret=s7_get_vfd(dev_name,0,&number,vfd_name);
      if((ret!=S7_OK)||(number==0))
      {     /* something has gone wrong */
            printf(     "\ns7_get_vfd = %lx, number = %d", ret,number);
            exit(-2);
      }

      /* initialize s7 */
      ret=s7_init(dev_name,vfd_name,cp_descr_ptr);
      if(ret!=S7_OK)
      {     /* something has gone wrong */
            printf(     "\ns7_init = %lx",ret);
            exit(-3);
      }
}
```

```
/* end communication */
static void my_shut(ord32 cp_descr)
{     int32 ret;

      ret=s7_shut(cp_descr);
      if(ret!=S7_OK)
      {     /* error has occurred -> exit */
            printf(      "\ns7_shut = %lx",ret);
      }

      /* no error has occurred         */
}

/* main */
void main(void)
{     ord32  cp_descr;
      ord16  cref;

      /* initialize s7 */
      my_init(&cp_descr);

      /* get reference for connection 'TEST' */
      my_get_cref(cp_descr,&cref);

      /* end communication */
      my_shut(cp_descr);
}
```

Ablaufdiagramm

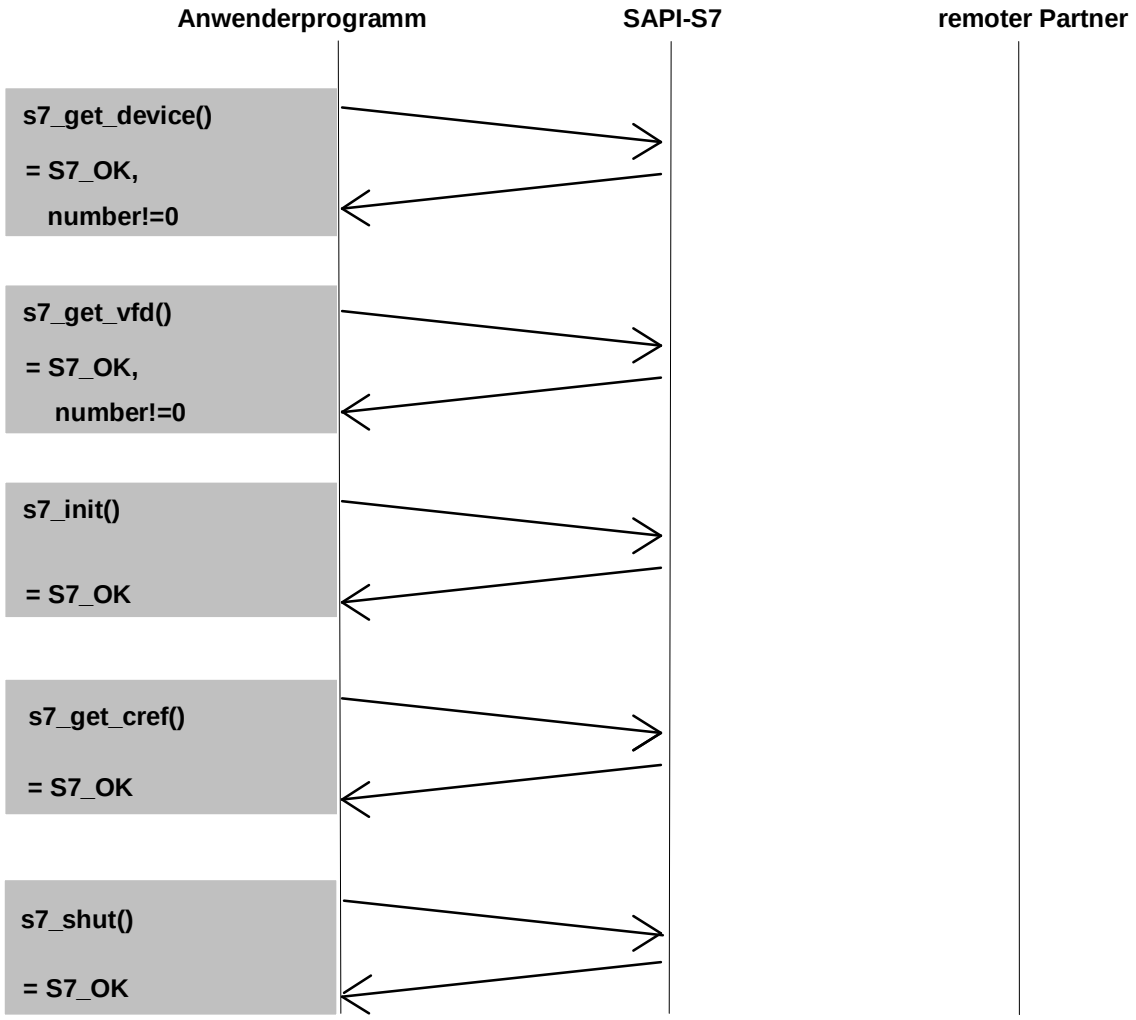

Bild 3.1: Ablaufdiagramm für das Beispiel

3.2.1 s7_get_device

Beschreibung

Mit diesem Aufruf können die konfigurierten Namen der installierten Kommunikationsprozessoren vom Kommunikationssystem abgefragt werden. Die Namen sind für die Anmeldung mit ‘s7_init()’ relevant.

Deklaration

```
int32 s7_get_device(
                ord16  index,          /* Vorgabe  */
                ord16  *number_ptr,    /* Rückgabe */
                char   *dev_name       /* Rückgabe */
                      )
```

Parameter

index	Der Parameter ‘index’ selektiert einen der vorhandenen CPs, die mit 0 beginnend fortlaufend numeriert sind.
number_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ ‘ord16’. Hier wird die Anzahl der installierten CPs zurückgeliefert.
dev_name	Anfangsadresse eines vom Anwenderprogramm bereitgestellten Speicherbereichs, in dem ein konfigurierter CP-Name eingetragen wird. Der Speicherbereich sollte mindestens (S7_MAX_DEVICELEN + 1) Byte groß sein für den maximal S7_MAX_DEVICELEN Byte langen CP-Namen plus String-Endekennung.

Abfrage aller CPs

Zum Abfragen aller CPs verwenden Sie am besten eine Programmschleife. Der Parameter ‘index’ wird als Schleifenvariable verwendet und läuft von 0 bis ‘*number_ptr-1’. Der Parameter ‘*number_ptr’ wird mit 1 vorbesetzt.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.2.2 s7_get_vfd

Beschreibung

Mit diesem Aufruf können die projektierten VFDs eines Kommunikationsprozessors abgefragt werden. Die VFD-Namen sind für die Anmeldung mit 's7_init()' relevant.

Deklaration

```
int32 s7_get_vfd(
                char   *dev_name,      /* Vorgabe  */
                ord16  index,          /* Vorgabe  */
                ord16  *number_ptr,    /* Rückgabe */
                char   *vfd_name       /* Rückgabe */

                  )
```

Parameter

dev_name	Konfigurierter Name des Kommunikationsprozessors, über den kommuniziert werden soll. Für diesen Parameter wird üblicherweise ein mit 's7_get_device()' ausgelesener CP-Name verwendet. Er muß mit einem konfigurierten CP-Namen übereinstimmen (z. B. 'CP_L2_1:').
index	Der Parameter 'index' selektiert eines der vorhandenen VFDs, die mit 0 beginnend fortlaufend numeriert sind.
number_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird die Anzahl der projektierten VFDs zurückgeliefert.
vfd_name	Anfangsadresse eines vom Anwenderprogramm bereitgestellten Speicherbereichs, in dem ein projektierter VFD-Name eingetragen wird. Der Speicherbereich sollte mindestens (S7_MAX_NAMLEN + 1) Byte für den maximal S7_MAX_NAMLEN Byte langen VFD-Namen plus String-Endekennung groß sein.

Abfrage aller VFDs

Zum Abfragen aller VFDs verwenden Sie am besten eine Programmschleife. Der Parameter ‘index’ wird als Schleifenvariable verwendet und läuft von 0 bis ‘*number_ptr-1’. Der Parameter ‘*number_ptr’ wird mit 1 vorbesetzt.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.2.3 s7_init

Beschreibung

Mit diesem Aufruf meldet sich ein Anwenderprogramm bei einem VFD eines unterlagerten Kommunikationssystems an.

Das Anwenderprogramm gibt als Eingabeparameter den konfigurierten Namen des Kommunikationsprozessors, über den er kommunizieren möchte, und den projektierten Namen des VFD vor, deren Ressourcen und Dienste genutzt werden sollen.

Zurück erhält es einen Descriptor 'cp_descr', der bis zur Abmeldung der Applikation bei allen folgenden Aufrufen als Adressierungsparameter für den selektierten CP und das VFD mitgegeben wird.

Bei der Anmeldung wird die Projektierinformation (z. B. die Liste aller S7-Verbindungen) aus einer Datei ausgelesen, deren Name sich aus dem CP-Namen wie folgt ableitet: Entfernen des Doppelpunkts, mit dem der CP-Name abschließt, und Anhängen der Namenserweiterung ‘.LDB’. Dabei wird vorausgesetzt, daß sich die Datei im aktuellen Arbeitsverzeichnis befindet. Als Alternative hierzu kann über Environmentvariablen der Name der Projektierdatei frei vorgegeben werden.

Bei paralleler Benutzung von mehreren VFDs in einem oder mehreren Kommunikationsprozessoren durch eine Applikation sind entsprechend viele Aufrufe 's7_init()' notwendig.

Deklaration

```
int32 s7_init(
            char   *cp_name,       /* Vorgabe  */
            char   *vfd_name,      /* Vorgabe  */
            ord32  *cp_descr_ptr   /* Rückgabe */

            )
```

Parameter	cp_name	Konfigurierter Name des Kommunikationsprozessors, über den kommuniziert werden soll. Um eine bestimmte Baugruppe anzusprechen, verwendet das Anwenderprogramm einen mit dem SIMATIC NET Installations-Tool konfigurierten CP-Namen (z. B. 'CP_L2_1:'). Hier wird üblicherweise ein mit 's7_get_device()' ausgelesener CP-Name verwendet.
	vfd_name	Projektierter Name des lokalen VFD, auf die sich die Applikation anmeldet. Um ein bestimmtes VFD anzusprechen, verwendet das Anwenderprogramm einen mit dem SIMATIC NET Projektier-Tool festgelegten VFD-Namen. Mit dem VFD werden auch die S7-Verbindungen selektiert. Hier wird üblicherweise ein mit 's7_get_vfd()' ausgelesener VFD-Name verwendet.
	cp_descr_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord32'. Hier wird ein Descriptor zur Adressierung des ausgewählten Kommunikationsprozessors und dem VFD abgelegt. Dieser Parameter muß für die weitere Kommunikation über den gewählten Kommunikationsprozessor und das VFD benutzt werden.
Return-Werte	S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
	S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
	S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.2.4 s7_get_cref

Beschreibung

Die S7-Applikation sieht vom Kommunikationssystem nur den symbolischen Namen der S7-Verbindung, der zur Laufzeit unhandlich ist. Deshalb besorgt sich die Applikation über den Aufruf 's7_get_cref()' eine Referenz auf einen symbolischen Verbindungsnamen.

Deklaration

```
int32 s7_get_cref(
                ord32  cp_descr,    /* Vorgabe  */
                char   *conn_name,  /* Vorgabe  */
                ord16  *cref_ptr    /* Rückgabe */
                  )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
conn_name	Symbolischer Name der S7-Verbindung, über die kommuniziert werden soll.
cref_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird die ermittelte Verbindungsreferenz zurückgegeben.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.2.5 s7_get_conn

Beschreibung

Mit diesem Aufruf können die S7-Verbindungen eines VFDs auf einem Kommunikationsprozessor und deren Referenzen abgefragt werden. Dieser Wert selektiert die reale Verbindung im Netz und ist handlicher als der symbolische Name.

Deklaration

```
int32 s7_get_conn(
                ord32  cp_descr,     /* Vorgabe   */
                ord16  index,        /* Vorgabe   */
                ord16  *number_ptr,  /* Rückgabe  */
                ord16  *cref_ptr,    /* Rückgabe  */
                char   *conn_name    /* Rückgabe  */
                )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
index	Der Parameter ‘index’ selektiert eine der vorhandenen S7-Verbindungen, die mit 0 beginnend fortlaufend numeriert sind.
number_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ ‘ord16’. Hier wird die Anzahl der S7-Verbindungen zurückgeliefert.
cref_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird die ermittelte Verbindungsreferenz zurückgegeben.
conn_name	Anfangsadresse eines vom Anwenderprogramm bereitgestellten Speicherbereichs, in dem ein S7-Verbindungsname eingetragen wird. Der Speicherbereich sollte mindestens (S7_MAX_NAMLEN + 1) Byte für den maximal S7_MAX_NAMLEN Byte langen Verbindungsnamen plus String-Endekennung groß sein.

Abfrage aller S7-Verbindungen

Zum Abfragen aller S7-Verbindungen verwenden Sie am besten eine Programmschleife. Der Parameter ‘index’ wird als Schleifenvariable verwendet und läuft von 0 bis ‘*number_ptr-1’. Der Parameter ‘*number_ptr’ wird mit 1 vorbesetzt.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.2.6 s7_shut

Beschreibung

Mit dem Aufruf 's7_shut()' macht die Applikation eine durch den CP-Descriptor gekennzeichnete Anmeldung (‘s7_init()’) beim Kommunikationsprozessor rückgängig. Dabei werden alle beim Anmelden belegten Ressourcen des Kommunikationssystems zurückgegeben und aufgebaute Verbindungen abgebrochen. Dieser Aufruf muß daher unbedingt vor dem Programmende für jede einzelne Anmeldung abgesetzt werden.

Deklaration

```
int32 s7_shut(
                ord32   cp_descr    /* Vorgabe  */
              )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs. Er kennzeichnet die Anmeldung, die rückgängig gemacht werden soll.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.3 Empfangsaufruf

Beschreibung des Beispiels

Als Erweiterung zum Beispiel aus Kapitel 3.2 wird mit Hilfe des Empfangsaufrufs '**s7_receive()**' auf eine eventuell vorhandene Nachricht geprüft. Eine Nachricht wird in diesem einfachen Fall nicht erwartet ('S7_NO_MSG' als Retourniert); das Beispiel dient lediglich als Vorbereitung für nachfolgende Programme.

Beispiel

```
:
:
/* additional prototypings */
static void my_receive(ord32 cp_descr,int32 last_event_expected);

/* receive any message from communication system */
static void my_receive(ord32 cp_descr,int32 last_event_expected)
{     ord16 cref,orderid;
      int32 ret;

      do
      {     ret=s7_receive(cp_descr,&cref,&orderid);
            switch(ret)
            {     case S7_NO_MSG:
                        break;
                  default:
                        printf("\nEvent unexpected: %lx", ret);
                        break;
            }
      }     while(ret!=last_event_expected);
}

/* main */
void main(void)
{     ord32 cp_descr;
      ord16 cref;

      /* initialize s7 */
      my_init(&cp_descr);

      /* get reference for connection 'TEST' */
      my_get_cref(cp_descr,&cref);

      /* receive message */
      my_receive(cp_descr,S7_NO_MSG);

      /* end communication */
      my_shut(cp_descr);
}
```

Ablaufdiagramm

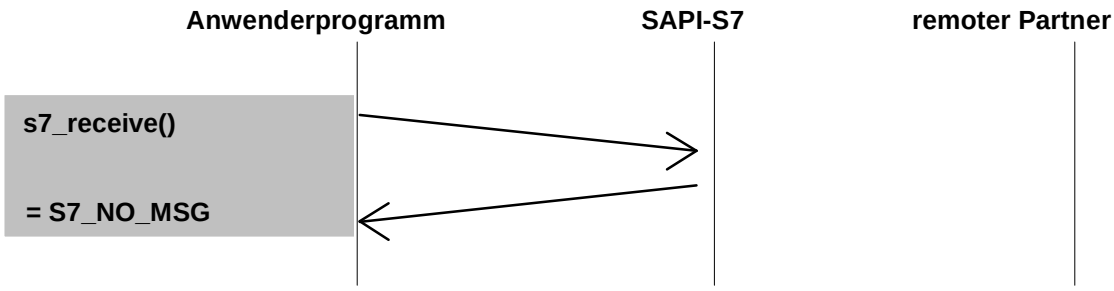

Bild 3.2: Ablaufdiagramm für das Beispiel

3.3.1 s7_receive

Beschreibung

Diese Empfangsfunktion der S7-Library hat die zentrale Aufgabe, vom unterlagerten Kommunikationssystem empfangene Ereignisse zu analysieren und ohne weitere Bearbeitung direkt an die Applikation zu melden.

Der Aufruf 's7_receive()' ist zwingend notwendig, wenn der Client

- ➢ quittierte Aufrufe abgesetzt hat und auf die zugehörige Quittung des Servers wartet,
- ➢ unquittierte Nachrichten vom Netz empfangen will,
- ➢ Sequenzaufträge abgesetzt hat (z. B. Aufbau einer S7-Verbindung), um so die Bearbeitung der Dienstsequenz bis zu deren Ende zu gewährleisten.

Der Rückgabewert der Funktion 's7_receive()' liefert bei Empfang einer Nachricht eine Dienstkennung. Diese identifiziert die Dienstart der empfangenen Antwort (z. B. 'S7_READ_CNF', wenn eine Variable gelesen wurde). Nach Erhalt einer Nachricht mit 's7_receive()' ist der Aufruf der zugehörigen Bearbeitungsfunktion zwingend notwendig (z. B. 's7_get_read_cnf()'). Weitere Empfangsaufrufe werden sonst mit einer Fehlermeldung abgelehnt.

Deklaration

```
int32 s7_receive(
            ord32  cp_descr,      /* Vorgabe  */
            ord16  *cref_ptr,     /* Rückgabe */
            ord16  *orderid_ptr   /* Rückgabe */
               )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs. Er kennzeichnet den Kommunikationsprozessor und das VFD, über die ein Ereignis abgeholt werden soll.
cref_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird die Referenz der S7-Verbindung, auf der eine Indication oder eine Confirmation empfangen wurde, abgelegt. Sie entspricht der S7-Verbindungsreferenz, auf der der Auftrag abgegeben wurde.
orderid_ptr	Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird das Auftragskennzeichen der erhaltenen Quittung abgelegt. Dieses dient dem Anwenderprogramm zur Identifizierung des zuvor abgegebenen Auftrags. Die Order-ID ist bei unquittierten Diensten oder Diensten zum Aufbau einer S7-Verbindung nicht relevant und wird mit einem Default-Wert belegt.

Return-Werte

S7_NO_MSG	Es wurde keine Nachricht empfangen.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.
	Neben diesen Werten werden je nach empfangener Nachricht weitere Dienstkennungen, wie z. B. 'S7_READ_CNF', zurückgeliefert. Die entsprechenden Definitionen finden Sie in der Datei ‘SAPI_S7.H’.

Aufrufart

In Single-Tasking-Betriebssystemen wie MS-DOS wird die Empfangsfunktion pollend aufgerufen. Für Multi-Tasking-Betriebssysteme wie MS-Windows besteht die Möglichkeit, die Empfangsfunktion an einer zentralen Wartestelle einzuklinken, um so alle Ereignisse empfangen zu können.

3.4 S7-Verbindungsmanagement-Dienste

Beschreibung des Beispiels

Als Erweiterung zum Beispiel aus Kapitel 3.3 wird eine S7-Verbindung mit dem symbolischen Namen 'TEST' aufgebaut (‘**s7_initiate_req()**’) und nach Erhalt der Quittung (‘**s7_get_initiate_cnf()**’) wieder abgebrochen (‘**s7_abort()**’).

Beispiel

```
:
:
/* additional prototypings */
static void my_abort(ord32 cp_descr,ord16 cref);
static void my_get_abort_ind(ord32 cp_descr);
static void my_get_initiate_cnf(ord32 cp_descr);
static void my_initiate_req(ord32 cp_descr,ord16 cref);

/* abort connection */
static void my_abort(ord32 cp_descr,ord16 cref)
{     int32 ret;

      if((ret=s7_abort(cp_descr,cref))!=S7_OK)
      {
            my_exit(cp_descr,"Error s7_abort",ret);
      }
}

/* get abort indication */
static void my_get_abort_ind(ord32 cp_descr)
{     int32 ret;

      if((ret=s7_get_abort_ind())!=S7_OK)
      {
            my_exit(     cp_descr,
      }
}

/* get initiate confirmation */
static void my_get_initiate_cnf(ord32 cp_descr)
{     int32 ret;

      if((ret=s7_get_initiate_cnf())!=S7_OK)
      {
            my_exit(     cp_descr,
      }
}

/* initiate connection "TEST" */
static void my_initiate_req(ord32 cp_descr,ord16 cref)
{     int32 ret;

      if((ret=s7_initiate_req(  cp_descr,cref))!=S7_OK)
      {
            my_exit(cp_descr,"Error
s7_initiate_req",ret);
      }
}
```

```
/* receive any message from communication system */
static void my_receive(ord32 cp_descr,int32
last_event_expected)
{     ord16 cref,orderid;
      int32 ret;

      do
      {     ret=s7_receive(cp_descr,&cref,&orderid);
            switch(ret)
            {
                  :
                  :
                  case S7_INITIATE_CNF:
                        my_get_initiate_cnf(cp_descr);
                        my_abort(cp_descr,cref);
                        break;
                  case S7_ABORT_IND:
                        my_get_abort_ind(cp_descr);
                        break;
                  default:
                        printf(     "Event
unexpected",                                          ret);
                        break;
            }
      }     while((ret!=last_event_expected)&&
}

/* main */
void main(void)
{     ord32  cp_descr;
      ord16  cref;

      /* initialize s7 */
      my_init(&cp_descr);

      /* get reference for connection 'TEST' */
      my_get_cref(cp_descr,&cref);

      /* initiate connection */
      my_initiate_req(cp_descr,cref);

      /* receive initiate confirmation */
      my_receive(cp_descr,S7_INITIATE_CNF);

      /* end communication */
      my_shut(cp_descr);
}
```

Ablaufdiagramm für den aktiven Verbindungs-aufbau

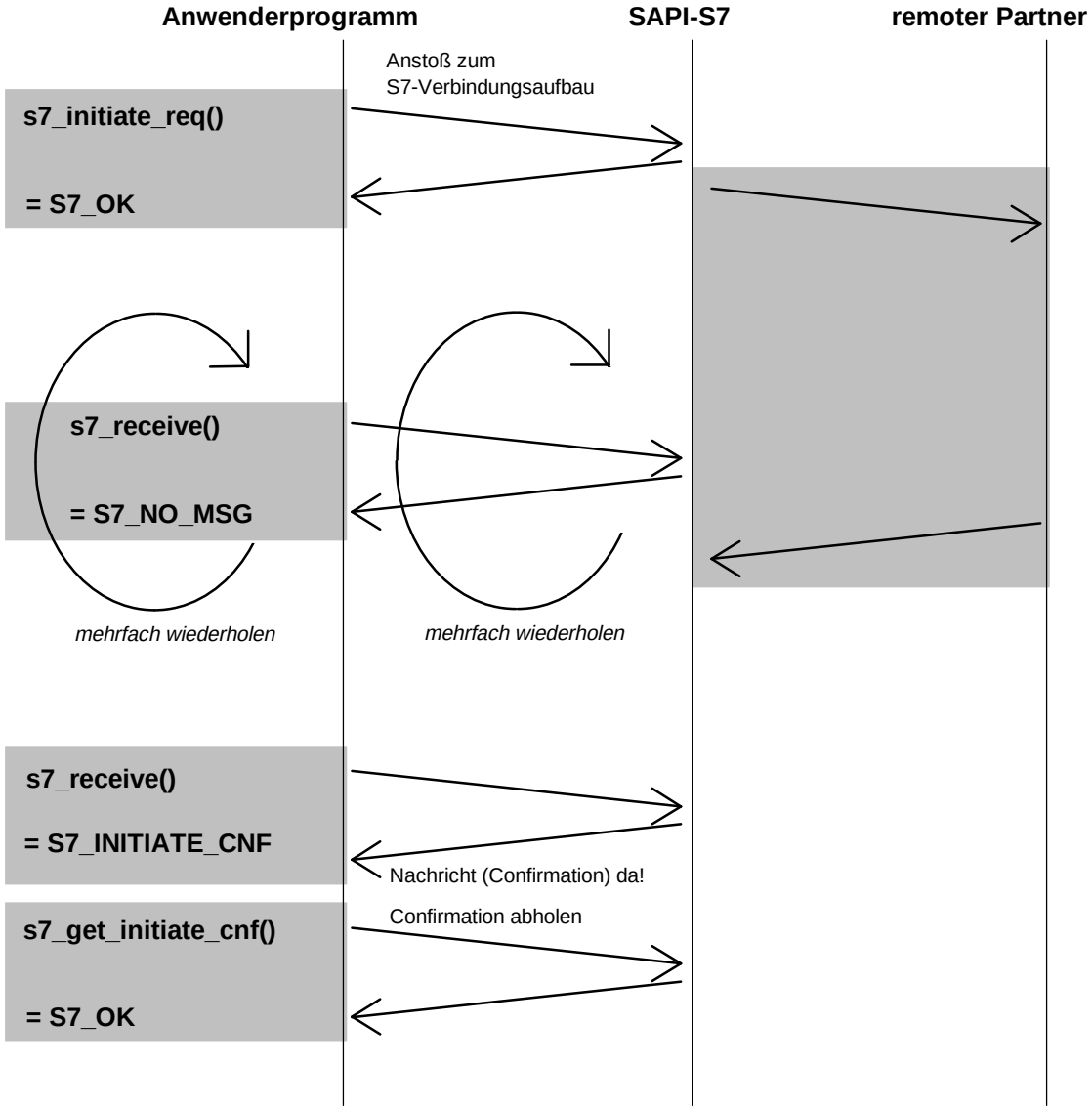

Bild 3.3: Ablaufdiagramm für den aktiven S7-Verbindungsaufbau

Ablaufdiagramm für das Vorbereiten zum passiven Verbindungs-aufbau (nicht Bestandteil des Beispiels)

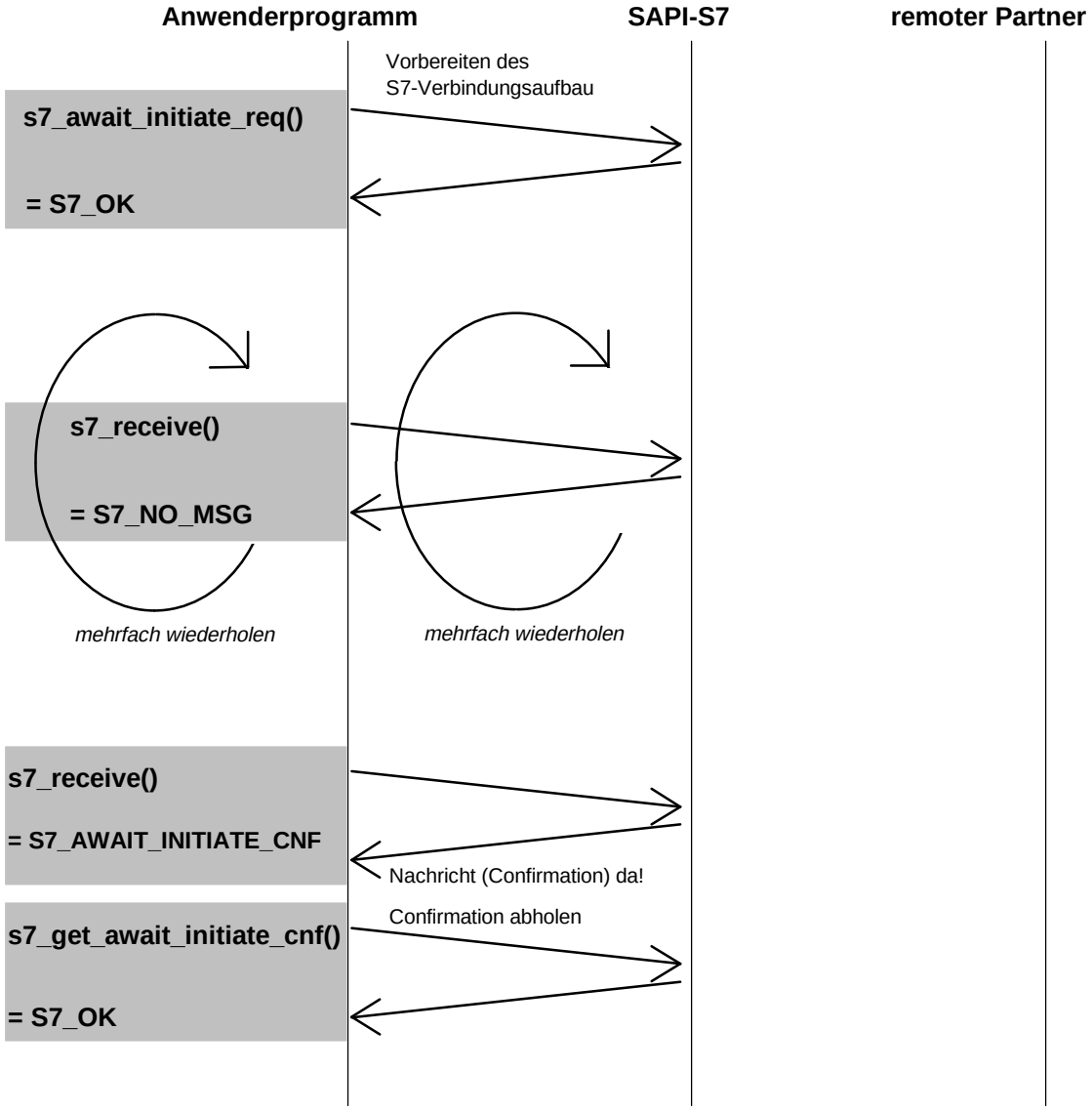

Bild 3.4: Ablaufdiagramm für das Vorbereiten zum passiven S7-Verbindungsaufbau

Ablaufdiagramm zum passiven Verbindungs-aufbau (nicht Bestandteil des Beispiels)

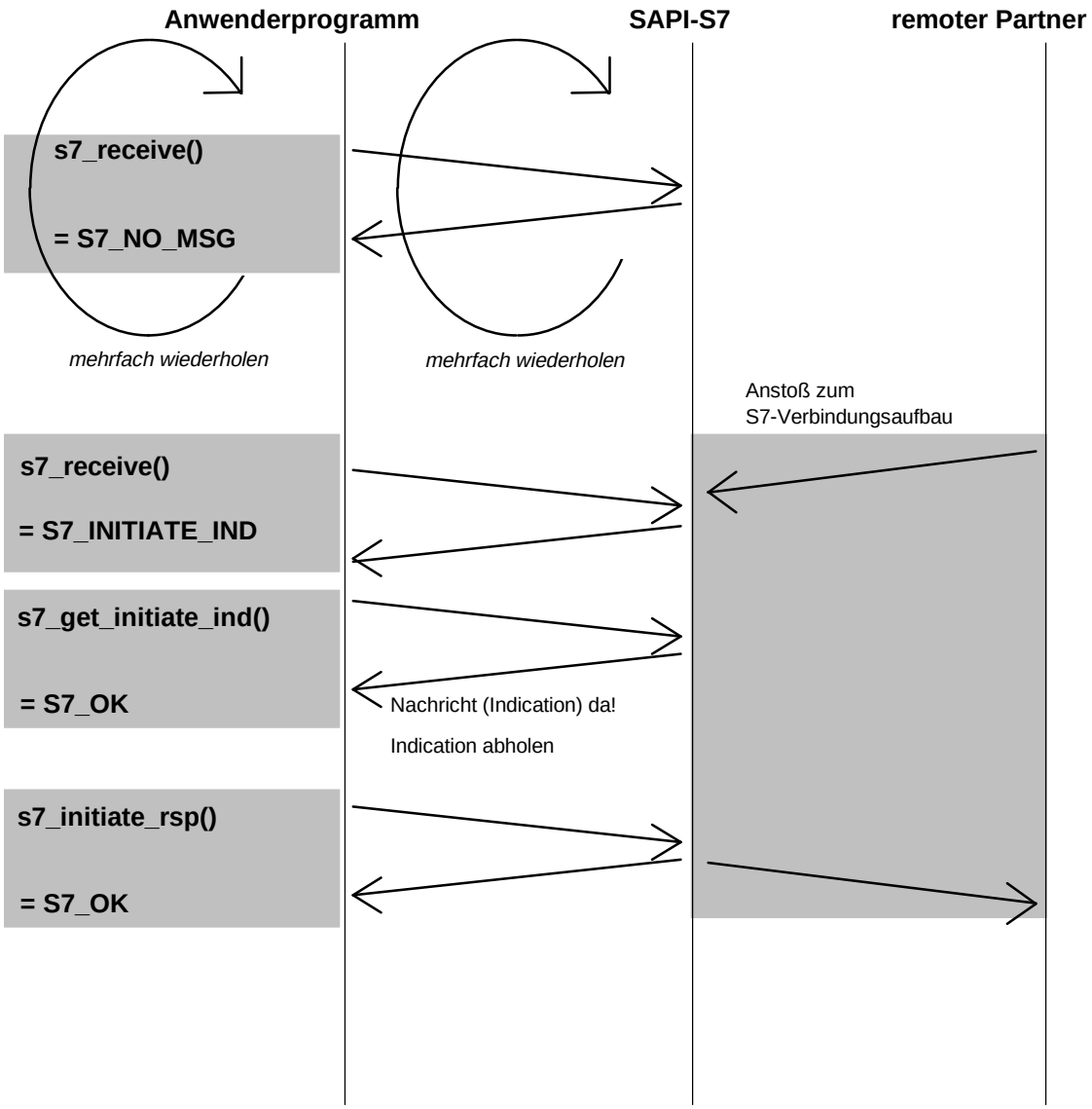

Bild 3.5: Ablaufdiagramm für den passiven S7-Verbindungsaufbau

3.4.1 s7_initiate_req

Beschreibung

Mit dem Aufruf 's7_initiate_req()' wird der Aufbau einer S7-Verbindung angestoßen. Die Initiative zum Aufbau geht dabei vom Anwenderprogramm aus, die notwendigen Parameter werden aus der Mini-DB gelesen. Dort können sie vom Anwenderprogramm ausgelesen, verändert und den jeweiligen Anforderungen angepaßt werden.

Beim Partner (Passivseite) werden die entsprechenden, durch Projektierung festgelegten Leistungsparameter mit den empfangenen Werten des Auftrags verglichen. Die realisierbare Leistungsfähigkeit der S7-Verbindung (Sende-Credit, Empfangs-Credit, PDU Größe, ...) ist durch den jeweils kleineren der beidseitig möglichen Leistungsparameter bestimmt.

Deklaration

```
int32 s7_initiate_req(
                ord32  cp_descr,    /* Vorgabe  */
                ord16  cref         /* Vorgabe  */
                      )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, die aufgebaut werden soll.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.4.2 s7_get_initiate_cnf

Beschreibung

Mit dem Aufruf 's7_get_initiate_cnf()' wird das Ergebnis eines S7-Verbindungsaufbaus entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_INITIATE_CNF', falls der Aufbauwunsch abgearbeitet wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_initiate_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Die ausgehandelten Leistungsparameter (Sende-Credit, Empfangs-Credit, PDU-Größe) werden in der Mini-DB eingetragen und können anschließend mit Mini-DB-Aufrufen ausgelesen werden.

Deklaration

```
int32 s7_get_initiate_cnf(void)
```

Parameter

keine

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.4.3 s7_await_initiate_req

Beschreibung

Mit dem Aufruf 's7_await_initiate_req()' wird das Kommunikationssystem für Verbindungsaufbauwünsche vom remoten Partner vorbereitet. Die Initiate zum Verbindungsaufbau geht dabei von der Partnerstation aus, die notwendigen Parameter werden aus der Mini-DB gelesen. Dort können sie vom Anwenderprogramm ausgelesen, verändert und den jeweiligen Anforderungen angepaßt werden.

Deklaration

```
int32 s7_await_initiate_req(
                ord32  cp_descr,    /* Vorgabe  */
                ord16  cref         /* Vorgabe  */
                          )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, die aufgebaut werden soll.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.4.4 s7_get_await_initiate_cnf

Beschreibung

Mit dem Aufruf 's7_get_await_initiate_cnf()' wird das Ergebnis des 's7_await_initiate_req()'-Auftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_AWAIT_INITIATE_CNF', falls das Kommunikationsystem für einen Verbindungsaufbau vorbereitet wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_initiate_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_await_initiate_cnf(void)
```

Parameter

entfällt

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.4.5 s7_get_initiate_ind

Beschreibung

Mit dem Aufruf 's7_get_initiate_ind()' wird ein Aufbauwunsch einer S7-Verbindung von der remoten Seite entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_INITIATE_IND', falls der remote Partner eine S7-Verbindung aufbauen möchte. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_initiate_ind()' zur internen Bearbeitung in der Library aufgerufen werden.

Mit dem Aufruf 's7_get_initiate_ind()' werden die Leistungsparameter der S7-Verbindung in der Mini-DB abgelegt. Anschließend können weitere Ereignisse vom Kommunikationssystem (z. B. über andere S7-Verbindungen) empfangen werden. Eine Zustimmung oder Ablehnung des Aufbauwunsches kann mit dem Aufruf 's7_initiate_rsp()' zeitlich verzögert erfolgen .

Durch diese Zweiteilung des passiven Aufbaus ist es möglich,

- ➢ bei einer einfachen Aufrufstruktur die Leistungsparameter vor dem Aufbau zu überprüfen und
- ➢ den Aufbauwunsch einer anderen Applikation weiterzureichen, die ihrerseits darüber entscheidet (Applikation hat Gateway-Funktionalität).

Deklaration

```
int32 s7_get_initiate_ind(void)
```

Parameter

keine

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.4.6 s7_initiate_rsp

Beschreibung

Mit dem Aufruf 's7_initiate_rsp()' entscheidet das Anwenderprogramm über den passiven Aufbauwunsch.

Der passive Aufbau einer S7-Verbindung ist zweigeteilt:

- ➢ mit dem Aufruf 's7_get_initiate_ind()' werden die Leistungsparameter in die Mini-DB eingetragen und können dort ausgelesen werden. Anschließend ist es möglich, weitere Ereignisse vom Kommunikationssystem (z. B. über andere S7-Verbindungen) zu empfangen.
- ➢ mit dem Aufruf 's7_initiate_rsp()' wird der Aufbauwunsch zustimmend oder ablehnend beantwortet.

Deklaration

```
int32 s7_initiate_rsp(
                ord32  cp_descr,    /* Vorgabe  */
                ord16  cref,        /* Vorgabe  */
                ord16  accept       /* Vorgabe  */
                      )
```

Parameter

cp_descr	Descriptor des CP, über den der Aufbauwunsch einer S7-Verbindung gemeldet wurde.
cref	Referenz der S7-Verbindung, die aufgebaut werden soll.
accept	Mit diesem Parameter kann der Aufbauwunsch angenommen ('accept=S7_ACCEPT') oder auch abgewiesen werden ('accept= S7_NON_ACCEPT').

Return-Werte	S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
	S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
	S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.4.7 s7_abort

Beschreibung

Mit dem Aufruf 's7_abort()' wird eine bestehende S7-Verbindung unquittiert und nicht verhandelbar abgebrochen.

Das Abholen einer Anwort mittels 's7_receive()' ist nicht notwendig, da es sich hierbei um einen unquittierten Dienst handelt. Noch nicht quittierte Aufträge dieser S7-Verbindung können anschließend eintreffen oder ganz ausbleiben.

Deklaration

```
int32 s7_abort(
                ord32  cp_descr,    /* Vorgabe  */
                ord16  cref         /* Vorgabe  */
              )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, die abgebrochen werden soll.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.4.8 s7_get_abort_ind

Beschreibung

Mit dem Aufruf 's7_get_abort_ind()' wird ein Abbruch einer S7-Verbindung durch eine remote Station oder den unterlagerten CP entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_ABORT_IND', falls durch die remote Partnerstation oder durch den unterlagerten Kommunikationsprozessor die S7-Verbindung abgebrochen wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_abort_ind()' zur internen Bearbeitung in der Library aufgerufen werden.

Bei S7 ist der Abbruch einer S7-Verbindung ein unquittierter Dienst. Noch nicht quittierte Aufträge dieser S7-Verbindung werden nicht mehr bestätigt.

Deklaration

```
int32 s7_get_abort_ind(void)
```

Parameter

keine

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5 Variablendienste

Beschreibung des Beispiels

Als Erweiterung zum Beispiel aus Kapitel 3.4 wird über die aufgebaute S7-Verbindung ein Auftrag zum zyklischen Lesen einer Variablen ('**s7_cycl_read()**') abgesetzt. Die empfangenen Daten werden mit Hilfe der Funktion '**s7_get_cycl_read_ind()**' in den Anwenderspeicher kopiert. Der Zyklus wird mit '**s7_cycl_read_delete_req()**' beendet.

symbolische Variablen-adressierung

Die Adressierung von S7-Variablen erfolgt symbolisch. Diese Art des Zugriffs orientiert sich an der Notation bei S7-Werkzeugen. Sie müssen sich also nicht verschiedene Notationen für Variablenadressen aneignen.

Beispiele:

DB5,X12.1	Datenbaustein 5, Datenbyte 12, Datenbit 1
DB5,B12	Datenbaustein 5, Datenbyte 12
DB5,W10	Datenbaustein 5, Datenwort 10
E1.1,5	5 Eingänge ab Eingangsbit 1.1 bis einschließlich Eingangsbit 1.5
A2.3,7	7 Ausgänge ab Ausgangsbit 2.3 bis einschließlich Ausgangsbit 3.1
MB9,3	3 Merkerbyte ab Merkerbyte 9
DB5,W10,9	9 Worte im Datenbaustein 5, ab Datenwort 10

Syntax:

Die Syntax ist wie folgt definiert (Klein-/Großschreibung irrelevant):

DB<nr>, DI<nr>, <bereich>	<typ>	<index> <index>.<bitnr>	,<anzahl>
vorgeschrieben	optional	vorgeschrieben	optional

Parameterbeschreibung:

DB bzw. DI	Datenbaustein oder Instanzdatenbaustein	
<nr>	Nummer des Datenbausteins oder Instanzdatenbausteins	
<bereich>	**A**	Ausgang
	C	Zähler
	E	Eingang
	M	Merker
	PA	Peripherieausgang
	PE	Peripherieeingang
	T	Timer
	Z	Zähler
<typ>	**B**	Byte (unsigned)
	BYTE	Byte (unsigned)
	CHAR	Byte (signed)
	D	Doppelwort (unsigned)
	DINT	Doppelwort (signed)
	DWORD	Doppelwort (unsigned)
	INT	Word (signed)
	REAL	Floating Point
	W	Wort (unsigned)
	WORD	Wort (unsigned)
	X	Byte für Bereich DB und DI
<index>	Elementnummer bezogen auf den Bausteinanfang	
<bitnr>	Bit innerhalb der Elementnummer	
<anzahl>	Anzahl der Variablen eines Typs, deren Werte ab <adresse> im <bereich> adressiert werden	

Beispiel

```
:
:
/* additional prototypings */
static void my_cycl_read(ord32 cp_descr,ord16 cref,ord16 orderid);
static void my_cycl_read_delete_req(  ord32 cp_descr,
                                      ord16 cref,
                                            ord16 orderid);
static void my_get_cycl_read_delete_cnf(ord32 cp_descr);
static void my_get_cycl_read_ind(ord32 cp_descr);

/* start cyclic read */
static void my_cycl_read(ord32 cp_descr,ord16 cref,ord16 orderid);
{     struct S7_READ_PARA read_para;
      int32 ret;

      read_para.access=S7_ACCESS_SYMB_ADDRESS;
      strcpy(read_para.var_name,“e0,10“);

      ret=s7_cycl_read(cp_descr,cref,orderid,200,1,&read_para);
      if(ret!=S7_OK)
      {
            my_exit(     cp_descr,
                         "Error s7_cycl_read",
                         ret);
      }
}

/* delete cyclic read */
static void my_cycl_read_delete_req(  ord32,cp_descr,ord16 cref,
                                      ord16 orderid)
{     int32 ret;

      ret=s7_cycl_read_delete_req(cp_descr,cref,orderid);
      if(ret!=S7_OK)
      {
            my_exit(     cp_descr,
                         "Error s7_cycl_read",
                         ret);
      }
}

/* get cyclic read delete confirmation */
static void my_get_cycl_read_delete_cnf(ord32 cp_descr)
{     int32 ret;

      if((ret=s7_get_cycl_read_delete_cnf())!=S7_OK)
      {
            my_exit(     cp_descr,
                         "Error s7_get_cycl_read_delete_cnf",
                         ret);
      }
}
```

```
/* get cyclic read indication */
static void my_get_cycl_read_ind(ord32 cp_descr)
{     int32 ret;
      ord16 var_length=10,result;
      char  data_buffer[10];
      char  *value_array[1];

      value_array[0]=data_buffer;

      if((ret=s7_get_cycl_read_ind((  void *)0,
                                          &result,
                                          &var_length,
                                    (void *)value_array))!=S7_OK)
      {
            my_exit(     cp_descr,
                         "Error s7_get_cycl_read_ind",
                         ret);
      }
}


/* receive any message from communication system */
static void my_receive(ord32 cp_descr,int32 last_event_expected)
{     ord16 cref,orderid;
      int32 ret;

      do
      {     ret=s7_receive(cp_descr,&cref,&orderid);
            switch(ret)
            {
                  :
                  :
                  case S7_INITIATE_CNF:
                        my_get_initiate_cnf(cp_descr);
                        my_cycl_read(cp_descr,cref,0);
                        break;
                  case S7_CYCL_READ_IND:
                        my_get_cycl_read_ind(cp_descr);
                        my_cycl_read_delete_req(
                                          cp_descr,
                                          cref,
                                          orderid);
                        break;
                  case S7_CYCL_READ_DELETE_CNF:
                        my_get_cycl_read_delete_cnf(
                                                cp_descr);
                        my_abort(cp_descr,cref);
                        break;
                  default:
                        printf(      "Event unexpected",
                              ret);
                        break;
            }
      }     while( (ret!=last_event_expected)&&
      (ret!=S7_ABORT_IND));
}
```

```
/* main */
void main(void)
{     ord32  cp_descr;
      ord16  cref;

      /* initialize s7 */
      my_init(&cp_descr);

      /* get reference for connection 'TEST' */
      my_get_cref(cp_descr,&cref);

      /* initiate connection */
      my_initiate_req(cp_descr,cref);

      /* receive cyclic read delete confirmation */
      my_receive(cp_descr,S7_CYCL_READ_DELETE_CNF);

      /* end communication */
      my_shut(cp_descr);
}
```

Ablaufdiagramm

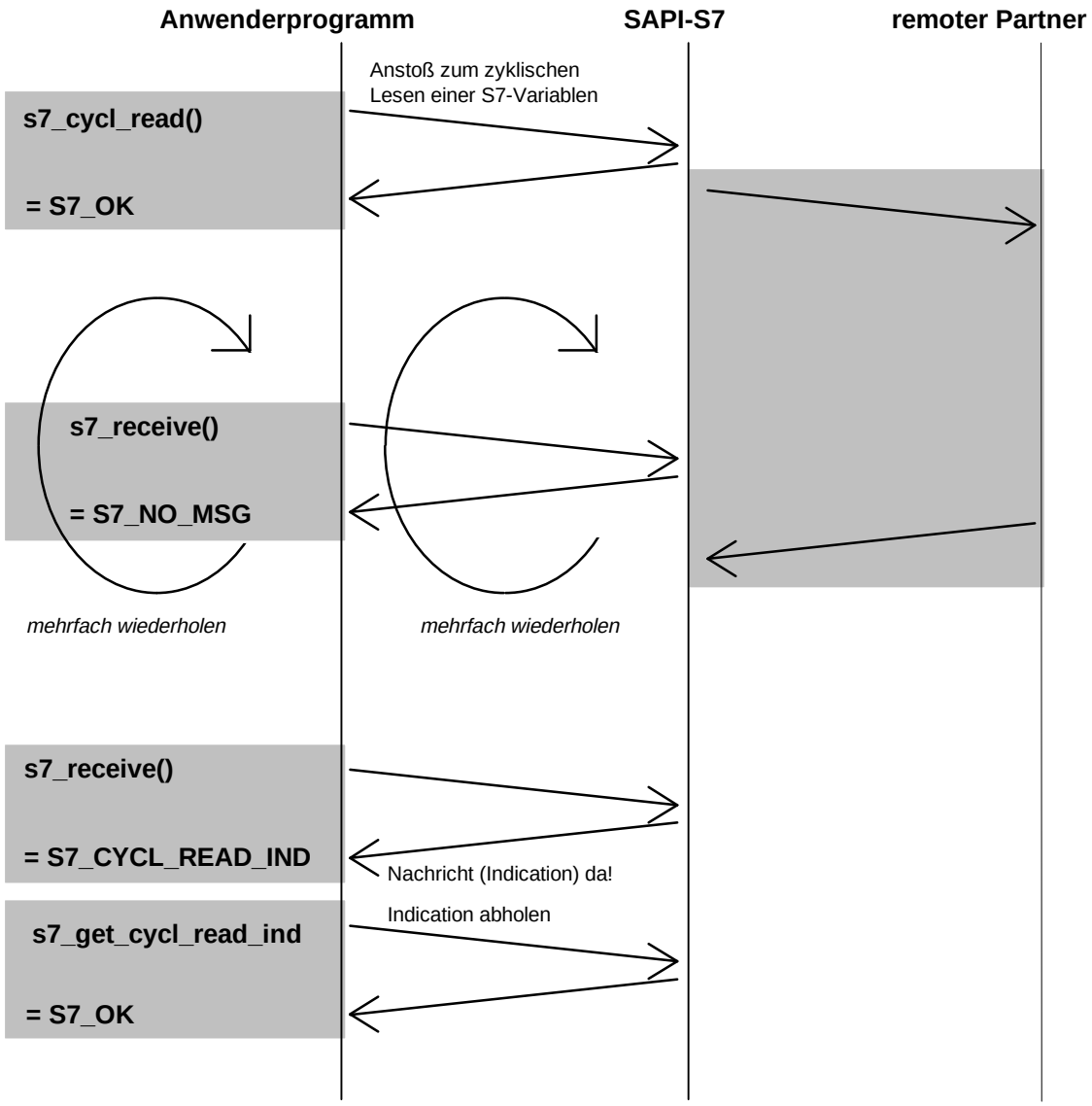

Bild 3.6: Ablaufdiagramm für das Beispiel

3.5.1 s7_read_req

Beschreibung

Mit dem Aufruf 's7_read_req()' kann eine Client-Applikation auf eine Variable eines Servers lesend zugreifen. Der Zugriff auf die Variablen kann nur per symbolischer Adresse erfolgen. Die Daten werden in der Quittung vom Server zum Client übertragen und werden mit Hilfe der entsprechenden Bearbeitungsfunktion für die Abschlußquittung ('s7_get_read_cnf()') in den Anwenderpuffer übertragen.

Deklaration

```
int32 s7_read_req(
                ord32  cp_descr,       /* Vorgabe  */
                ord16  cref,           /* Vorgabe  */
                ord16  orderid,        /* Vorgabe  */
                struct S7_READ_PARA *read_para_ptr
                       /* Vorgabe  */

                    )
```

Parameter

cp_descr — Handle als Rückgabewert des 's7_init()'-Aufrufs.

cref — Referenz der S7-Verbindung, über die der Auftrag gesendet werden soll.

orderid — Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung.

read_para_ptr — Zeiger auf einen vom Anwenderprogramm bereitgestellten Puffer der nachfolgenden Struktur:

```
struct S7_READ_PARA
{       ord16   access;
        char    var_name[S7_MAX_NAMLEN+2];
        ord16   index;
        ord16   subindex;
        ord16   address_len;
        ord8    address[S7_MAX_ADDRESSLEN];
}
```

Der Parameter **'access'** gibt die Zugriffsart an. Mit dem Wert 'S7_ACCESS_SYMB_ADDRESS' wird die symbolische Adresse im Feld ‘var_name’ erwartet.

Der Parameter **'var_name'** gibt die symbolische Adresse der zu lesenden Variablen an und wird ausgewertet, falls der Zugriff auf eine Variable per sym-

bolischer ('access=S7_ACCESS_SYMB_ADDRESS') Adresse erfolgt (Beachten Sie bitte die allgemeinen Hinweise zur Variablenadressierung zu Beginn dieses Kapitels).

Der Parameter **'index'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'subindex'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address_len'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.2 s7_get_read_cnf

Beschreibung

Mit dem Aufruf 's7_get_read_cnf()' wird ein gelesener Variablenwert entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_READ_CNF', falls der Leseauftrag durchgeführt wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_read_cnf()' zur internen Bearbeitung in der Library aufgerufen werden, um die gelesenen Werte in einen Anwenderpuffer zu kopieren.

Deklaration

```
int32 s7_get_read_cnf(
                void *od_ptr,          /* Vorgabe  */
                ord16 *var_length_ptr,/* Vor- und */
                                       /* Rückgabe */
                void   *value_ptr      /* Rückgabe */
                       )
```

Parameter

od_ptr — Der Parameter 'od_ptr' ist aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert und muß mit dem NULL-Pointer belegt werden, d. h die Variablenwerte werden an der SAPI-S7-Programmierschnittstelle in Netzdarstellung übergeben.

var_length_ptr — Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird die Länge des Datenpuffers vorgegeben. Nach dem Aufruf enthält der Parameter die Länge der gelesenen Variablen.

value_ptr — Zeiger auf einen vom Anwenderprogramm bereitgestellten Puffer. Hier wird der gelesene Variableninhalt in der Netzdarstellung abgelegt. Beim Auswerten des Pufferinhalts muß der Datentyp der Variablen beachtet werden.

Es ist darüber hinaus zu berücksichtigen, daß die Variablenwerte byte-aligned, d. h. ohne Füllbytes (Padding-Bytes) zwischen zwei Komponenten, abgelegt werden.

Return-Werte	S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
	S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
	S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.3 s7_write_req

Beschreibung

Mit dem Aufruf 's7_write_req()' kann eine Client-Applikation auf eine Variable eines Servers schreibend zugreifen. Der Zugriff auf die Variablen kann nur per symbolischer Adresse erfolgen. Die Daten werden im Auftrag vom Client zum Server übertragen.

Deklaration

```
int32 s7_write_req(
                ord32  cp_descr,       /* Vorgabe */
                ord16  cref,           /* Vorgabe  */
                ord16  orderid,        /* Vorgabe  */
                struct S7_WRITE_PARA *write_para_ptr
                                       /* Vorgabe  */
                void   *od_ptr         /* Vorgabe  */
                      )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, über die der Auftrag gesendet werden soll.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung.
write_para_ptr	Zeiger auf einen vom Anwenderprogramm bereitgestellten Puffer der nachfolgenden Struktur:

```
struct S7_WRITE_PARA
{       ord16   access;
        char    var_name[S7_MAX_NAMLEN+2];
        ord16   index;
        ord16   subindex;
        ord16   address_len;
        ord8    address[S7_MAX_ADDRESSLEN];
        ord16   var_length;
        ord8    value[S7_MAX_BUFLEN];
}
```

Der Parameter **'access'** gibt die Zugriffsart an. Mit dem Wert 'S7_ACCESS_SYMB_ADDRESS' wird die symbolische Adresse im Feld ‘var_name’ erwartet.

Der Parameter **'var_name'** gibt die symbolische Adresse der zu schreibenden Variablen an und wird ausgewertet, falls der Zugriff auf eine Variable per symbolischer Adresse erfolgt ('access= S7_ACCESS_SYMB_ADDRESS') (Beachten Sie bitte die allgemeinen Hinweise zur Variablenadressierung zu Beginn dieses Kapitels).

Der Parameter **'index'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'subindex'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address_len'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'var_length'** gibt die Anzahl der relevanten und gültigen Bytes im Datenpuffer 'value' an.

Im Puffer **'value'** ist der Wert der zu schreibenden Variable in der Netzdarstellung abgelegt. Beim Auswerten des Pufferinhalts muß der Datentyp der Variablen beachtet werden.

Es ist darüber hinaus zu berücksichtigen, daß die Variablenwerte byte-aligned, d. h. ohne Füllbytes (Padding-Bytes) zwischen zwei Komponenten, übergeben werden müssen.

od_ptr

Der Parameter 'od_ptr' ist aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert und muß mit dem NULL-Pointer belegt werden, d. h. die Variablenwerte werden an der SAPI-S7-Programmierschnittstelle in Netzdarstellung übergeben.

Return-Werte	S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
	S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
	S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.4 s7_get_write_cnf

Beschreibung

Mit dem Aufruf 's7_get_write_cnf()' wird das Ergebnis des Variablen-Schreibauftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_WRITE_CNF', falls der Schreibauftrag durchgeführt wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_write_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_write_cnf(void)
```

Parameter

keine

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.5 s7_multiple_read_req

Beschreibung

Mit dem Aufruf 's7_multiple_read_req()' kann eine Client-Applikation auf eine oder mehrere Variablen eines Servers gleichzeitig lesend zugreifen. Der Zugriff auf die Variablen kann nur per symbolischer Adresse erfolgen. Die Daten werden in der Quittung vom Server zum Client übertragen und werden mit Hilfe der entsprechenden Bearbeitungsfunktion für die Abschlußquittung ('s7_get_multiple_read_cnf()') in die Anwenderpuffer übertragen.

Deklaration

```
int32 s7_multiple_read_req(
                ord32  cp_descr,       /* Vorgabe  */
                ord16  cref,           /* Vorgabe  */
                ord16  orderid,        /* Vorgabe  */
                ord16  number,         /* Vorgabe  */
                struct S7_READ_PARA *read_para_array
                                       /* Vorgabe  */

                        )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der Verbindung, über die der Auftrag gesendet werden soll.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung.
number	Anzahl der Variablen, die gelesen werden sollen.
read_para_array	Zeiger auf ein vom Anwenderprogramm bereitgestelltes Array von insgesamt 'number' Elementen der nachfolgenden Struktur, wobei das i-te Element die i-te Variable beschreibt:

```
struct S7_READ_PARA
{       ord16   access;
        char    var_name[S7_MAX_NAMLEN+2];
        ord16   index;
        ord16   subindex;
        ord16   address_len;
        ord8    address[S7_MAX_ADDRESSLEN];
}
```

Der Parameter **'access'** gibt die Zugriffsart an. Mit dem Wert 'S7_ACCESS_SYMB_ADDRESS' wird die symbolische Adresse im Feld ‘var_name’ erwartet.

Der Parameter **'var_name'** gibt die symbolische Adresse der zu lesenden Variablen an und wird ausgewertet, falls der Zugriff auf eine Variable per symbolischer Adresse erfolgt ('access= S7_ACCESS_SYMB_ADDRESS') (Beachten Sie bitte die allgemeinen Hinweise zur Variablenadressierung zu Beginn dieses Kapitels).

Der Parameter **'index'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'subindex'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address_len'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.6 s7_get_multiple_read_cnf

Beschreibung

Mit dem Aufruf 's7_get_multiple_read_cnf()' werden die gelesenen Variablenwerte entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_MULTIPLE_READ_CNF', falls der Leseauftrag durchgeführt wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_multiple_read_cnf()' zur internen Bearbeitung in der Library aufgerufen werden. Sie kopiert die gelesenen Werte in die Anwenderpuffer.

Deklaration

```
int32 s7_get_multiple_read_cnf(
                void   *od_ptr,        /* Vorgabe  */
                ord16  *result_array,  /* Rückgabe */
                ord16  *var_length_array,
                                       /* Vor- und */
                                       /* Rückgabe */
                void   *value_array    /* Rückgabe */
                                    )
```

Parameter

od_ptr

Der Parameter 'od_ptr' ist aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert und muß mit dem NULL-Pointer belegt werden, d. h die Variablenwerte werden an der SAPI-S7-Programmierschnittstelle in Netzdarstellung übergeben.

result_array

Adresse eines vom Anwenderprogramm bereitgestelltes Arrays vom Typ 'ord16'. Das Array muß mindestens so viele Elemente enthalten, wie Variablen gelesen wurden. Die Array-Elemente enthalten die Zugriffsergebnisse in der Reihenfolge, in der die Variablen beim Auftrag angegeben wurden. Folgende Zugriffsergebnisse sind möglich:

S7_RESULT_OK

Dieser Wert zeigt den fehlerfreien Zugriff auf die Variable an.

S7_RESULT_HW_ERROR

Bei diesem Wert trat ein Hardwarefehler auf.

S7_RESULT_OBJ_ACCESS_DENIED

Der Zugriff auf eine Variable wurde abgelehnt.

S7_RESULT_OBJ_ADDRESS_INVALID

Die angegebene Adresse ist ungültig.

S7_RESULT_OBJ_TYPE_NOT_SUPPORTED

Der Server unterstützt den Datentyp nicht.

S7_RESULT_OBJ_TYPE_INCONSISTENT

Der Datentyp der Variablen ist nicht konsistent.

S7_RESULT_OBJ_NOT_EXISTS

Die anzusprechende Variable existiert nicht.

var_length_array Adresse eines vom Anwenderprogramm bereitgestellten Arrays vom Typ 'ord16'. Das Array muß mindestens so viele Elemente umfassen, wie Variablen gelesen wurden. In den einzelnen Array-Elementen werden die Längen der Datenpuffer vorgegeben. Nach dem Aufruf enthalten die Elemente dieses Arrays die Längen der gelesenen Variablen. Der Wert '0' bedeutet, daß die entsprechende Variable nicht gelesen werden konnte.

value_array Zeigerfeld auf vom Anwenderprogramm bereitgestellte Puffer. In den einzelnen Puffern werden die gelesenen Variablenwerte abgelegt. Auch hier gilt die Reihenfolge, in der die Variablen beim Auftrag angegeben wurden. Beim Auswerten der Pufferinhalte muß der Datentyp der Variablen beachtet werden.

Es ist darüber hinaus zu berücksichtigen, daß die Variablenwerte byte-aligned, d. h. ohne Füllbytes (Padding-Bytes) zwischen zwei Komponenten, übergeben werden müssen.

Return-Werte	S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
	S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
	S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.7 s7_multiple_write_req

Beschreibung

Mit dem Aufruf 's7_multiple_write_req()' kann eine Client-Applikation auf eine oder mehrere Variablen eines Servers gleichzeitig schreibend zugreifen. Der Zugriff auf die Variablen kann nur per symbolischer Adresse erfolgen. Die Daten werden im Auftrag vom Client zum Server übertragen.

Deklaration

```
int32 s7_multiple_write_req(
                ord32  cp_descr,       /* Vorgabe  */
                ord16  cref,           /* Vorgabe  */
                ord16  orderid,        /* Vorgabe  */
                ord16  number,         /* Vorgabe  */
                struct S7_WRITE_PARA *write_para_array,
                                       /* Vorgabe  */
                void   *od_ptr         /* Vorgabe  */
                                       /* Vorgabe  */
                        )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der Verbindung, über die der Auftrag gesendet werden soll.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung.
number	Anzahl der Variablen, die geschrieben werden sollen.
write_para_array	Zeiger auf ein vom Anwenderprogramm bereitgestelltes Array von insgesamt 'number' Elementen der nachfolgenden Struktur, wobei das i-te Element die i-te Variable beschreibt:

```
struct S7_WRITE_PARA
{       ord16   access;
        char    var_name[S7_MAX_NAMLEN+2];
        ord16   index;
        ord16   subindex;
        ord16   address_len;
        ord8    address[S7_MAX_ADDRESSLEN];
        ord16   var_length;
        ord8    value[S7_MAX_BUFLEN];
}
```

Der Parameter **'access'** gibt die Zugriffsart an. Mit dem Wert 'S7_ACCESS_SYMB_ADDRESS' wird die symbolische Adresse im Feld ‘var_name’ erwartet.

Der Parameter **'var_name'** gibt die symbolische Adresse der zu lesenden Variablen an und wird ausgewertet, falls der Zugriff auf eine Variable per symbolischer ('access=S7_ACCESS_SYMB_ADDRESS') Adresse erfolgt (Beachten Sie bitte die allgemeinen Hinweise zur Variablenadressierung zu Beginn dieses Kapitels).

Der Parameter **'index'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'subindex'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address_len'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'var_length'** gibt die Anzahl der relevanten und gültigen Bytes im Datenpuffer 'value' an.

Im Puffer **'value'** ist der Wert der zu schreibenden Variable in der Netzdarstellung abgelegt. Beim Auswerten des Pufferinhalts muß der Datentyp der Variablen beachtet werden.

Es ist darüber hinaus zu berücksichtigen, daß die Variablenwerte byte-aligned, d. h. ohne Füllbytes (Padding-Bytes) zwischen zwei Komponenten, übergeben werden müssen.

od_ptr

Der Parameter 'od_ptr' ist aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert und muß mit dem NULL-Pointer belegt werden, d. h die Variablenwerte werden an der SAPI-S7-Programmierschnittstelle in Netzdarstellung übergeben.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.8 s7_get_multiple_write_cnf

Beschreibung

Mit dem Aufruf 's7_get_multiple_write_cnf()' werden die Ergebnisse eines Variablen-Schreibauftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_MULTIPLE_WRITE_CNF', falls der Schreibauftrag durchgeführt wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_multiple_write_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_multiple_write_cnf(
                ord16 *result_array    /* Rückgabe */
                                  )
```

Parameter

result_array

Adresse eines vom Anwenderprogramm bereitgestellten Arrays vom Typ 'ord16'. Das Array muß mindestens so viele Elemente enthalten, wie Variablen gelesen wurden. Die Array-Elemente enthalten die Zugriffsergebnisse in der Reihenfolge, in der die Variablen beim Auftrag angegeben wurden. Folgende Zugriffsergebnisse sind möglich:

S7_RESULT_OK

Dieser Wert zeigt den fehlerfreien Zugriff auf die Variable an.

S7_RESULT_HW_ERROR

Bei diesem Wert trat ein Hardwarefehler auf.

S7_RESULT_OBJ_ACCESS_DENIED

Der Zugriff auf eine Variable wurde abgelehnt.

S7_RESULT_OBJ_ADDRESS_INVALID

Die angegebene Adresse ist ungültig.

S7_RESULT_OBJ_TYPE_NOT_SUPPORTED

Der Server unterstützt den Datentyp nicht.

S7_RESULT_OBJ_TYPE_INCONSISTENT

Der Datentyp der Variablen ist nicht konsistent.

S7_RESULT_OBJ_NOT_EXISTS

Die anzusprechende Variable existiert nicht.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.9 s7_cycl_read_init_req

Beschreibung

Eine S7-Applikation veranlaßt mit diesem Auftrag den Server, ein zyklisches Lesen von Variablen vorzubereiten. Im Auftrag werden die Zykluszeit, die Anzahl der Variablen sowie die Variablen, deren Werte gelesen werden sollen, übergeben

Deklaration

```
int32 s7_cycl_read_init_req(
                ord32  cp_descr,       /* Vorgabe  */
                ord16  cref,           /* Vorgabe  */
                ord16  orderid,        /* Vorgabe  */
                ord16  cycl_time,      /* Vorgabe  */
                ord16  number,         /* Vorgabe  */
                struct S7_READ_PARA *read_para_array
                                       /* Vorgabe  */
                        )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs
cref	Referenz der Verbindung, über die der Auftrag gesendet werden soll. Über diese Verbindung werden die Variablenwerte übertragen.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung. Dieses Auftragskennzeichen muß bei weiteren Aufträgen wie das Starten (‘s7_cycl_read_start_req()’), Anhalten (‘s7_cycl_read_stop_req()’) und Löschen (‘s7_cycl_read_delete_req()’) des zyklischen Lesens wiederverwendet werden.
cycl_time	Zykluszeit als Vielfaches von 10tel Sekunden. SAPI-S7 rundet auf einen erlaubten Wert ab, das sind: 1, 2 ... 9 sowie 10, 20 ... 90 und 100, 200 ... 900.
number	Anzahl der Variablen, deren Werte zyklisch gelesen werden sollen.

read_para_array Zeiger auf ein vom Anwenderprogramm bereitgestelltes Array von insgesamt 'number' Elementen der nachfolgenden Struktur, wobei das i-te Element die i-te Variable beschreibt:

```
struct S7_READ_PARA
{       ord16   access;
        char    var_name[S7_MAX_NAMLEN+2];
        ord16   index;
        ord16   subindex;
        ord16   address_len;
        ord8    address[S7_MAX_ADDRESSLEN];
}
```

Der Parameter **'access'** gibt die Zugriffsart an. Mit dem Wert 'S7_ACCESS_SYMB_ADDRESS' wird die symbolische Adresse im Feld ‘var_name’ erwartet.

Der Parameter **'var_name'** gibt die symbolische Adresse der zu lesenden Variablen an und wird ausgewertet, falls der Zugriff auf eine Variable per symbolischer ('access=S7_ACCESS_SYMB_ADDRESS') Adresse erfolgt (Beachten Sie bitte die allgemeinen Hinweise zur Variablenadressierung zu Beginn dieses Kapitels).

Der Parameter **'index'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'subindex'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address_len'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.10 s7_get_cycl_read_init_cnf

Beschreibung

Mit dem Aufruf 's7_get_cycl_read_init_cnf()' wird das Ergebnis eines 's7_cycl_read_init_req()'-Auftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_CYCL_READ_INIT_CNF', falls der remote Partner den Auftrag zum Vorbereiten des zyklischen Variablen-Lesens abgearbeitet hat. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_cycl_read_init_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_cycl_read_init_cnf(void)
```

Parameter

entfällt

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.11 s7_cycl_read_start_req

Beschreibung

Eine S7-Applikation veranlaßt mit diesem Auftrag den Server, ein zyklisches Lesen von Variablen zu starten. Voraussetzung hierfür ist, daß der Server mit Hilfe des Aufrufs 's7_cycl_read_init_req()' für diesen Dienst vorbereitet wurde.

Deklaration

```
int32 s7_cycl_read_start_req(
                ord32   cp_descr,    /* Vorgabe  */
                ord16   cref,        /* Vorgabe  */
                ord16   orderid      /* Vorgabe  */
                      )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'-Aufrufs übereinstimmen
cref	Referenz der Verbindung, über die der Auftrag gesendet werden soll. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'-Aufrufs übereinstimmen.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'-Aufrufs übereinstimmen.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.12 s7_get_cycl_read_start_cnf

Beschreibung

Mit dem Aufruf 's7_get_cycl_read_start_cnf()' wird das Ergebnis eines 's7_cycl_read_start_req()'-Auftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_CYCL_READ_START_CNF', falls der remote Partner den Auftrag zum Starten des zyklischen Variablen-Lesens abgearbeitet hat. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_cycl_read_start_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_cycl_read_start_cnf(void)
```

Parameter

entfällt

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.13 s7_get_cycl_read_ind

Beschreibung

Mit dem Aufruf 's7_get_cycl_read_ind()' werden die vom Server gesendeten Daten entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_CYCL_READ_IND', falls vom remoten Partner eine Variable zyklisch gelesen wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_cycl_read_ind()' aufgerufen werden, um die gelesenen Werte in die Anwenderpuffer zu kopieren.

Deklaration

```
int32 s7_get_cycl_read_ind(
                void   *od_ptr,        /* Vorgabe  */
                ord16  *result_array,  /* Rückgabe */
                ord16  *var_length_array,
                                       /* Vor- und */
                                       /* Rückgabe */
                void   *value_array    /* Rückgabe */
                      )
```

Parameter

od_ptr

Der Parameter 'od_ptr' ist aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert und muß mit dem NULL-Pointer belegt werden, d. h die Variablenwerte werden an der SAPI-S7-Programmierschnittstelle in Netzdarstellung übergeben.

result_array

Adresse eines vom Anwenderprogramm bereitgestellten Arrays vom Typ 'ord16'. Das Array muß mindestens so viele Elemente enthalten, wie Variablen gelesen wurden. Die Array-Elemente enthalten die Zugriffsergebnisse in der Reihenfolge, in der die Variablen beim Auftrag angegeben wurden. Folgende Zugriffsergebnisse sind möglich:

S7_RESULT_OK

Dieser Wert zeigt den fehlerfreien Zugriff auf die Variable an.

S7_RESULT_HW_ERROR

Bei diesem Wert trat ein Hardwarefehler auf.

S7_RESULT_OBJ_ACCESS_DENIED

Der Zugriff auf eine Variable wurde abgelehnt.

S7_RESULT_OBJ_ADDRESS_INVALID

Die angegebene Adresse ist ungültig.

S7_RESULT_OBJ_TYPE_NOT_SUPPORTED

Der Server unterstützt den Datentyp nicht.

S7_RESULT_OBJ_TYPE_INCONSISTENT

Der Datentyp der Variablen ist nicht konsistent.

S7_RESULT_OBJ_NOT_EXISTS

Die anzusprechende Variable existiert nicht.

var_length_array Adresse eines vom Anwenderprogramm bereitgestellten Arrays vom Typ 'ord16'. Das Array muß mindestens so viele Elemente umfassen, wie Variablen gelesen wurden. In den einzelnen Array-Elementen werden die Längen der Datenpuffer vorgegeben. Nach dem Aufruf enthalten die Elemente dieses Arrays die Längen der gelesenen Variablen. Der Wert '0' bedeutet, daß die entsprechende Variable nicht gelesen werden konnte.

value_array Zeigerfeld auf vom Anwenderprogramm bereitgestellte Puffer. In den einzelnen Puffern werden die gelesenen Variablenwerte abgelegt. Auch hier gilt die Reihenfolge, in der die Variablen beim Auftrag angegeben wurden. Beim Auswerten der Pufferinhalte muß der Datentyp der Variablen beachtet werden.

Achten Sie darauf, daß die Variablenwerte byte-aligned, d. h. ohne Füllbytes (Padding-Bytes) zwischen zwei Komponenten, übergeben werden müssen.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.14 s7_get_cycl_read_abort_ind

Beschreibung

Mit dem Aufruf 's7_get_cycl_read_abort_ind()' wird eine Cyclic Read Abort Indication entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_CYCL_READ_ABORT_IND', falls das zyklische Variablen-Lesen abgebrochen wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_cycl_read_delete_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Mit Hilfe der in den vorigen Kapiteln beschriebenen Funktionen kann ein zyklisches Variablen-Lesen wieder eingerichtet werden.

Deklaration

```
int32 s7_get_cycl_read_abort_ind(void)
```

Parameter

entfällt

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.15 s7_cycl_read_stop_req

Beschreibung

Eine S7-Applikation veranlaßt mit diesem Auftrag den Server, ein zyklisches Lesen von Variablen anzuhalten. Voraussetzung hierfür ist, daß der Server zum zyklischen Variablen-Lesen bereits erfolgreich aufgefordert wurde.

Deklaration

```
int32 s7_cycl_read_stop_req(
                ord32  cp_descr,    /* Vorgabe  */
                ord16  cref,        /* Vorgabe  */
                ord16  orderid      /* Vorgabe  */
                     )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'- oder ‘s7_cycl_read()’-Aufrufs übereinstimmen.
cref	Referenz der Verbindung, über die der Auftrag gesendet werden soll. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()’- oder ‘s7_cycl_read()’-Aufrufs übereinstimmen.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'- oder ‘s7_cycl_read()’-Aufrufs übereinstimmen.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.16 s7_get_cycl_read_stop_cnf

Beschreibung

Mit dem Aufruf 's7_get_cycl_read_stop_cnf()' wird das Ergebnis eines 's7_cycl_read_stop_req()'-Auftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_CYCL_READ_STOP_CNF', falls der remote Partner den Auftrag zum Beenden des zyklischen Variablen-Lesens abgearbeitet hat. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_cycl_read_stop_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_cycl_read_stop_cnf(void)
```

Parameter

entfällt

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.17 s7_cycl_read_delete_req

Beschreibung

Mit dieser Funktion wird das zyklische Lesen abgebrochen und beim Server abgemeldet.

Deklaration

```
int32 s7_cycl_read_delete_req(
                ord32  cp_descr,    /* Vorgabe  */
                ord16  cref,        /* Vorgabe  */
                ord16  orderid      /* Vorgabe  */
                              )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'- oder ‘s7_cycl_read()’-Aufrufs übereinstimmen.
cref	Referenz der Verbindung, über die der Auftrag gesendet werden soll. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'- oder ‘s7_cycl_read()’-Aufrufs übereinstimmen.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung. Dieser Parameter muß mit dem entsprechenden Parameter des 's7_cycl_read_init_req()'- oder ‘s7_cycl_read()’-Aufrufs übereinstimmen.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.18 s7_get_cycl_read_delete_cnf

Beschreibung

Mit dem Aufruf 's7_get_cycl_read_delete_cnf()' wird das Ergebnis eines 's7_cycl_read_delete_req()'-Auftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_CYCL_READ_DELETE_CNF', falls der remote Partner den Auftrag zum Aufheben des zyklischen Variablen-Lesens abgearbeitet hat. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_cycl_read_delete_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_cycl_read_delete_cnf(void)
```

Parameter

entfällt

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.5.19 s7_cycl_read

Beschreibung

Eine S7-Applikation veranlaßt mit diesem Auftrag den Server, zyklisch Variablen zu lesen. Der Auftrag faßt die Sequenz zusammen, die aus den Aufrufen 's7_cycl_read_init_req()' (Anmelden beim Server) und 's7_cycl_read_start_req()' (Starten des Variablen-Lesens) besteht. Die Zykluszeit, die Anzahl der Variablen und die Variablen, deren Werte gelesen werden sollen, werden vorgegeben.

Wichtig: Bei S7 ist dieser Auftrag ein unquittierter Dienst (im Gegensatz zu den Einzelaufträgen, aus denen sich dieser Aufruf zusammensetzt).

Deklaration

```
int32 s7_cycl_read(
                ord32  cp_descr,       /* Vorgabe  */
                ord16  cref,           /* Vorgabe  */
                ord16  orderid,        /* Vorgabe  */
                ord16  cycl_time,      /* Vorgabe  */
                ord16  number,         /* Vorgabe  */
                struct S7_READ_PARA *read_para_array
                                       /* Vorgabe  */

                )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs
cref	Referenz der Verbindung, über die der Auftrag gesendet werden soll. Über diese Verbindung werden die Variablenwerte übertragen.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung. Dieses Auftragskennzeichen muß bei weiteren Aufträgen wie das Anhalten (‘s7_cycl_read_stop_req()’), Fortsetzen (‘s7_cycl_read_start_req()’) und Löschen (‘s7_cycl_read_delete_req()’) des zyklischen Lesens wiederverwendet werden.
cycl_time	Zykluszeit als Vielfaches von 10tel Sekunden. SAPI-S7 rundet auf einen erlaubten Wert ab, das sind: 1, 2 ... 9 sowie 10, 20 ... 90 und 100, 200 ... 900.
number	Anzahl der Variablen, deren Werte zyklisch gelesen werden sollen.

read_para_array Zeiger auf ein vom Anwenderprogramm bereitgestelltes Array von insgesamt 'number' Elementen der nachfolgenden Struktur, wobei das i-te Element die i-te Variable beschreibt:

```
struct S7_READ_PARA
{       ord16   access;
        char    var_name[S7_MAX_NAMLEN+2];
        ord16   index;
        ord16   subindex;
        ord16   address_len;
        ord8    address[S7_MAX_ADDRESSLEN];
}
```

Der Parameter **'access'** gibt die Zugriffsart an. Mit dem Wert 'S7_ACCESS_SYMB_ADDRESS' wird die symbolische Adresse im Feld ‘var_name’ erwartet.

Der Parameter **'var_name'** gibt die symbolische Adresse der zu lesenden Variablen an und wird ausgewertet, falls der Zugriff auf eine Variable per symbolischer Adresse erfolgt ('access= S7_ACCESS_SYMB_ADDRESS') (Beachten Sie bitte die allgemeinen Hinweise zur Variablenadressierung zu Beginn dieses Kapitels).

Der Parameter **'index'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'subindex'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address_len'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Der Parameter **'address'** ist nicht relevant und nur aus Kompatibilitätsgründen zu anderen SAPI-Schnittstellen realisiert.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.6 Blockorientierte Dienste

Beschreibung

Dem Anwendungsprogrammierer werden sowohl auf PG/PC- als auch auf SIMATIC S7-AS-Seite blockorientierte Dienste angeboten.

Diese Dienste bieten folgende Vorteile:

- ➢ Übertragung größerer Datenmengen
- ➢ Anstoß zur Übertragung kann von der SIMATIC S7-AS ausgehen

Einschränkungen

Die blockorientierten Dienste stehen unter MSDOS und Windows 3.x nicht zur Verfügung

Standard-funktionen

Auf der SIMATIC S7-AS-Seite stehen die beiden Bausteine BSEND und BRCV zur Verfügung. Diese Funktionalität wird auf PG/PC-Seite mit den Funktionen der 'Blockorientierten Dienste' umgesetzt.

Datenaustausch

Für den Datenaustausch zwischen zwei Kommunikationspartnern ist eine Verbindungsprojektierung notwendig. Bei den blockorientierten Diensten gehört immer das Bausteinpaar BSEND und BRCV zusammen. Hierfür ist eine **zweiseitige** Verbindungsprojektierung (COML S7 und STEP 7 NETPRO) notwendig. Im Gegensatz zu den nicht blockorientierten Diensten muß die mit STEP 7 NETPRO erstellte Verbindungsprojektierung in die AS geladen werden. Über eine Verbindung können mehrere Bausteinpaare Daten austauschen.

Beachten Sie bitte bei der Projektierung der Verbindungen die Hinweise in der betreffenden Produktinformation.

Datenmenge

Die Datenmenge beträgt bei diesem Datentransfer bis zu 65534 Byte, unabhängig von der Größe der CPU. Die Segmentierung der Daten wird von den Funktionen selbst übernommen.

Adressierungs-parameter

Der Adressierungsparameter R_ID ist für ein Bausteinpaar (BSEND/ BRCV) festgelegt und innerhalb einer Verbindung eindeutig definiert. Das heißt, es können mehrere BSEND-Bausteine über eine Verbindung senden, aber immer mit einer unterschiedlichen R_ID. Gleiche R_IDs können für weitere Verbindungen verwendet werden.

Einteilung der Funktionen

In der folgende Tabelle sind die blockorientierten Dienste in zwei Gruppen eingeteilt:

- ➢ Gruppe ‘bsend’
- ➢ Gruppe ‘brcv’

Blockorientierte Dienste	Funktionen
bsend	s7_bsend_req()
	s7_get_bsend_cnf()
brcv	s7_brcv_init()
	s7_get_brcv_ind()
	s7_brcv_stop()

In den folgenden Kapiteln werden die oben aufgeführten Funktionen beschrieben.

Beispiel ‘bsend’

```
void main(int argc,char **argv)
{     ord32  cp_descr;
      ord16  cref;
      ord16  orderid;
      ord32  r_id=1;
      int32  ret;
      char   send_buffer[100];
      ord16  err_no;
      const  char   *err_msg_ptr;
...
      Verbindung aufgebaut
...

      /* first BSEND request*/
      ret = s7_bsend_req( cp_descr,cref,orderid,r_id,
                          (void*)send_buffer,sizeof(send_buffer));
      printf(      "s7_bsend_req =0x%x, r_id = 0x%x, len = %d Byte\n",
                   ret, r_id, sizeof(send_buffer));
      while (ret == S7_NO_MSG )
      {
            ret = s7_receive(cp_descr,&cref,&orderid);
            printf( "s7_receive = 0x%x\n", ret);
            switch (ret)
            {

                  /* BSEND confirmation */
                  case S7_BSEND_CNF:
                  {
                        ret = s7_get_bsend_cnf();
                        printf("s7_get_bsend_cnf = 0x%x\n",ret );
                        if(ret == S7_OK)
                        {

                              /* next BSEND request */
                              ret = s7_bsend_req( cp_descr,cref,orderid,
                                                  r_id,
                                          (void*)send_buffer,
                                          sizeof(send_buffer));
                              printf(      "s7_bsend_req =0x%x,
                                          r_id = 0x%x,
                                          len = %d Byte\n",ret,r_id,
                                          sizeof(send_buffer));
                        }
                        break;
                  }
                  default:
                  {
                        ...
                        break;
                  }
            }
      }

      /* end communication */
      my_shut(cp_descr);
}
```

Ablaufdiagramm 'bsend'

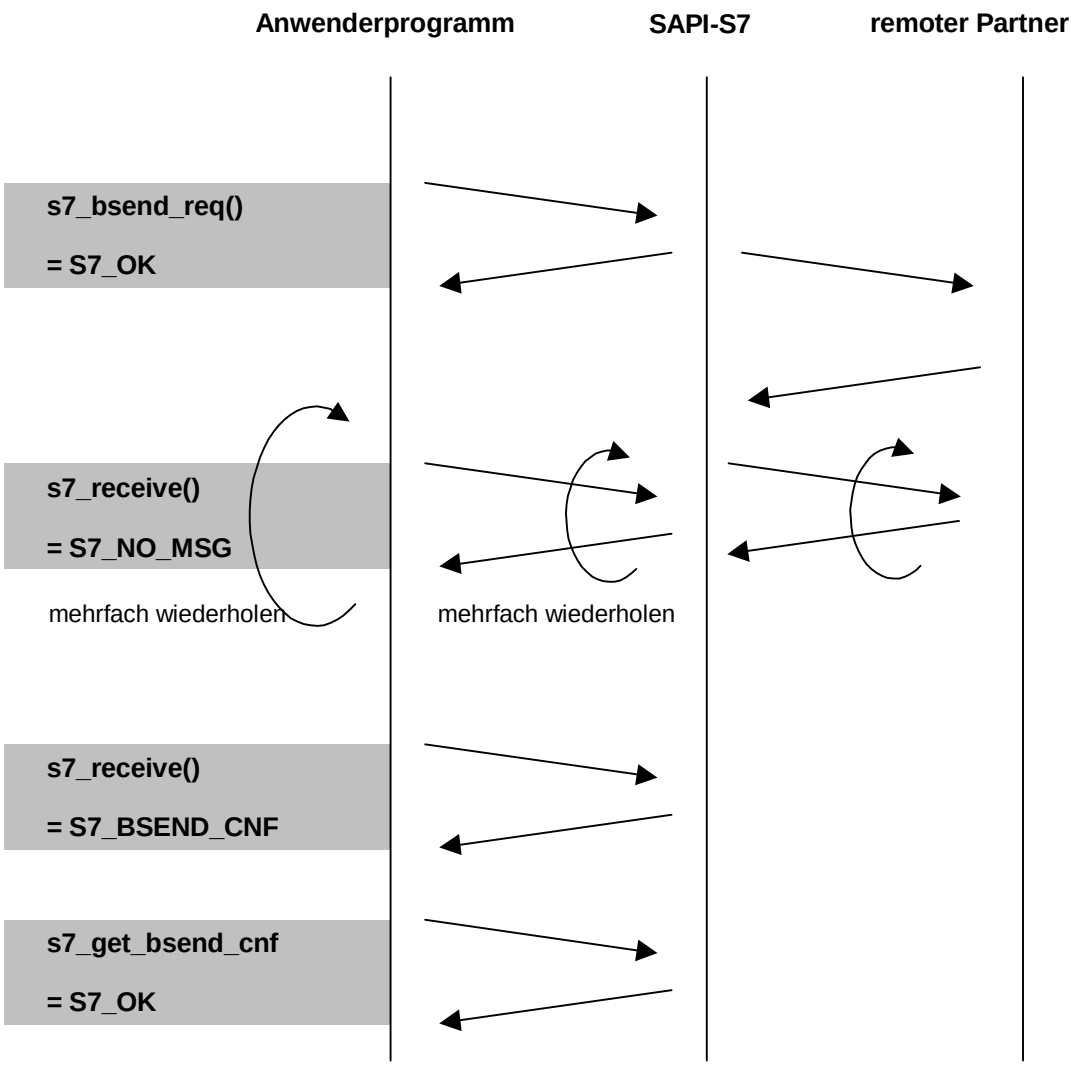

Bild 3.7: Ablaufdiagramm 'bsend'

Beispiel ‘brcv’

```
void main(int argc,char **argv)
{     ord32  cp_descr;
      ord16 cref;
      ord16  orderid;
      ord32 r_id=1;
      int32  ret;
      ord32  ret_id;
      ord16  ret_len;
      char   receive_buffer[65540];
      ord16  err_no;
      const  char   *err_msg_ptr;
...
      Verbindung aufgebaut
...

      /* BRCV initialize */
      ret = s7_brcv_init( cp_descr,cref,r_id);
      printf("s7_brcv_init =0x%x,  r_id = 0x%x\n",ret, r_id;
      while (ret == S7_NO_MSG )
      {
            ret = s7_receive(cp_descr,&cref,&orderid);
            printf( "s7_receive = 0x%x\n", ret);
            switch (ret)
            {

                  /* BRCV indication */
                  case S7_BRCV_IND:
                  {
                        ret = s7_get_brcv_ind(
                                          receive_buffer,
                                          (ord32)sizeof(receive_buffer),
                                          &ret_id,&ret_len);
                        printf(      "s7_get_brcv_ind = 0x%x, r_id = 0x%x,
                                    rec_len = %d Byte\n",ret, ret_id,
                                    ret_len);
                        break;
                  }
                  default:
                  {
                        ...
                        break;
                  }
            }
      }

      /* BRCV stop */
      ret = s7_brcv_stop( cp_descr,cref,r_id);
      printf("s7_brcv_stop = 0x%x",ret);

      /* end communication */
      my_shut(cp_descr);
}
```

Ablaufdiagramm 'brcv'

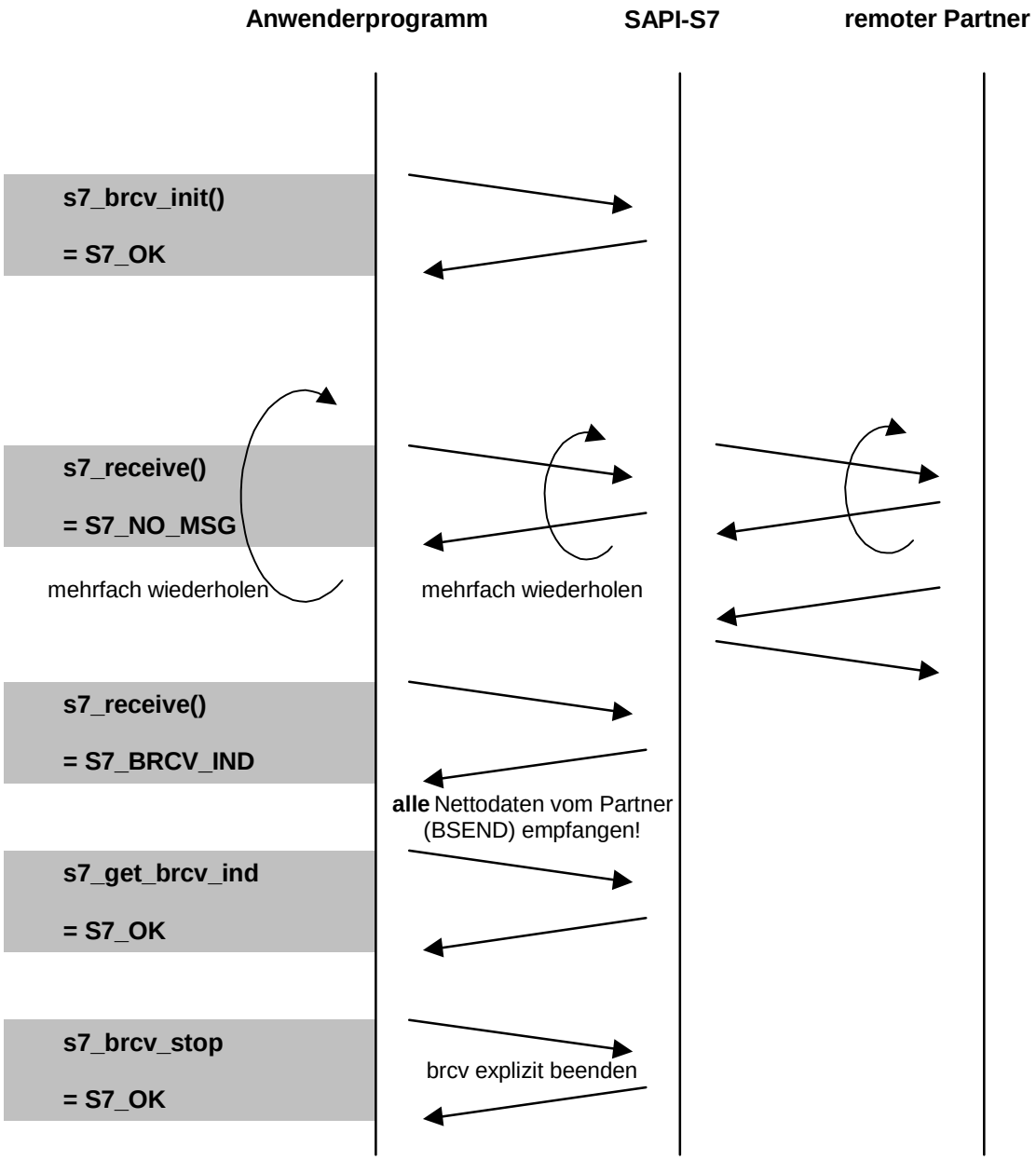

Bild 3.8: Ablaufdiagramm 'brcv'

3.6.1 s7_bsend_req

Beschreibung

Mit dem Aufruf s7_bsend_req kann eine Client-Applikation bis zu 65534 Byte Daten an eine remote Station senden.

Deklaration

```
int32 s7_bsend_req (
                    ord32    cp_descr,    /* Vorgabe */
                    ord16    cref,        /* Vorgabe */
                    ord16    orderid,     /* Vorgabe */
                    ord32    r_id,        /* Vorgabe */
                    void     *buffer_ptr, /* Vorgabe */
                    ord32    buffer_len   /* Vorgabe */
                       )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, über die der Auftrag gesendet werden soll.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung.
r_id	Es werden nur Daten zum Partner über diese Verbindung erfolgreich gesendet, wenn der Adressierungsparameter r_id für die Verbindung eindeutig ist und mit dem remoten R_ID des BRCV übereinstimmt. Wertebereich (hexadezimal): 0 bis FFFF FFFF
*buffer_ptr	Zeiger auf den zu sendenden Adreßbereich
buffer_len	explizite Längenangabe der Nettodaten in Bytes; Wertebereich (hexadezimal): 1 bis 65534

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.

S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.6.2 s7_get_bsend_cnf

Beschreibung

Mit dem Aufruf s7_get_bsend_cnf wird das Ergebnis des BSEND-Auftrags entgegengenommen.

Das Anwenderprogramm erhält beim s7_receive–Aufruf die Anzeige S7_BSEND_CNF, wenn der Sendeauftrag abgearbeitet wurde. Danach muß die zugehörige Bearbeitungsfunktion s7_get_bsend_cnf zur internen Bearbeitung in der Library aufgerufen werden.

Deklaration

```
int32 s7_get_bsend_cnf (void)
```

Parameter

keine

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.6.3 s7_brcv_init

Beschreibung

Mit diesem Aufruf meldet sich die Applikation zum Empfang von BSEND Aufträgen vom Verbindungspartner an. Jedem BSEND Auftrag mit einer bestimmten R_ID des Verbindungspartners muß genau ein s7_brcv_init mit der gleichen R_ID entsprechen.

Deklaration

```
int32 s7_brcv_init (
                    ord32   cp_descr,   /*  Vorgabe  */
                    ord16   cref,       /*  Vorgabe  */
                    ord32   r_id        /*  Vorgabe  */
                       )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, über die der Auftrag gesendet werden soll.
r_id	Es werden nur Daten vom Partner über diese Verbindung empfangen, wenn der Adressierungsparameter r_id für die Verbindung eindeutig ist und mit dem remoten R_ID am BSEND übereinstimmt.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.6.4 s7_get_brcv_ind

Beschreibung

Mit dem Aufruf s7_get_brcv_ind werden die vom Partner gesendeten Nettodaten in den angegebenen Speicherbereich kopiert.

Deklaration

```
int32 s7_get_brcv_ind (
        void    *buffer_ptr,          /* Vorgabe  */
        ord32   buffer_len,           /* Vorgabe  */
        ord32   *r_id_ptr,            /* Rückgabe */
        ord32   *rec_buffer_len_ptr   /* Rückgabe */
                    )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, über die der Auftrag gesendet werden soll.
*buffer_ptr	Der Pointer zeigt auf die Zieladresse, an der die empfangenen Daten abgelegt werden sollen.
buffer_len	Maximale Länge des Empfangspuffers (buffer_ptr); wenn die empfangene Datenlänge größer als die hier angegebene Pufferlänge ist, wird S7_ERR zurückgegeben. In diesem Fall wird im Parameter rec_buffer_len die benötigte Puffergröße angegeben und der detailed error auf S7_ERR_INVALID_DATARANGE_OR_TYPE gesetzt.
*r_id_ptr	*r_id_ptr zeigt auf eine Variable vom Typ ord32, die nach Aufruf der Funktion die R_ID des empfangenen BSEND-Auftrags enthält.
rec_buffer_len_ptr	Gesamte empfangene Datenlänge bei Return-Wert = S7_OK bzw. benötigte Puffergröße, wenn der Empfangsbuffer zu klein ist und der Return-wert = S7_ERR ist.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.

S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.6.5 s7_brcv_stop

Beschreibung

Mit diesem Aufruf meldet sich die Applikation beim Verbindungspartner ab.

Deklaration

```
int32 s7_brcv_stop (
                    ord32   cp_descr,  /*  Vorgabe  */
                    ord16   cref,      /*  Vorgabe  */
                    ord32   r_id       /*  Vorgabe  */
                       )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, über die der Auftrag gesendet werden soll.
r_id	Die beim entsprechenden brcv_init angegebene R_ID.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.7 VFD-Dienste

Beschreibung des Beispiels

Als Erweiterung zum Beispiel aus Kapitel 3.5 wird nach dem Beenden des zyklischen Variable-Lesens noch der Status des remoten VFD abgefragt. Der Status gibt Auskunft über die Betriebsbereitschaft des remoten Kommunikationspartners.

Beispiel

```
:
:
/* additional prototypings */
static void my_get_vfd_state_cnf(ord32 cp_descr);
static void my_get_vfd_ustate_ind(ord32 cp_descr);
static void my_vfd_state_req(ord32 cp_descr,ord16 cref);

/* get vfd state confirmation */
static void my_get_vfd_state_cnf(ord32 cp_descr)
{     ord16 log_state,phy_state;
      ord8  local_detail[3];
      int32 ret;

      ret=s7_get_vfd_state_cnf( &log_state,
                                &phy_state,
                                local_detail);
      if(ret!=S7_OK)
      {
            my_exit(     cp_descr,
                         "Error s7_get_vfd_state_cnf",
                         ret);
      }
}

/* get unsolicited state indication */
static void my_get_vfd_ustate_ind(ord32 cp_descr)
{     ord16 log_state,phy_state;
      ord8  local_detail[3];
      int32 ret;

      ret=s7_get_vfd_ustate_ind(      &log_state,
                                      &phy_state,
                                      local_detail);
      if(ret!=S7_OK)
      {
            my_exit(     cp_descr,
                         "Error s7_get_vfd_ustate_ind",
                         ret);
      }
}

/* send vfd state request */
static void my_vfd_state_req(ord32 cp_descr,ord16 cref)
{     int32 ret;

      ret=s7_vfd_state_req(
                  cp_descr,          /* cp_descr     */
                  cref,0             /* cref,orderid */);
      if(ret!=S7_OK)
      {
            my_exit(     cp_descr,
                         "Error s7_vfd_state_req",
                         ret);
      }
}
```

```
/* receive any message from communication system */
static void my_receive(ord32 cp_descr,int32 last_event_expected)
{     ord16 cref,orderid;
      int32 ret;

      do
      {     ret=s7_receive(cp_descr,&cref,&orderid);
            switch(ret)
            {
                  :
                  :
                  case S7_CYCL_READ_DELETE_CNF:
                        my_get_cycl_read_delete_cnf(
                        cp_descr);
                        my_vfd_state_req(cp_descr,cref);
                        break;
                  case S7_VFD_STATE_CNF:
                        my_get_vfd_state_cnf(cp_descr);
                        my_abort(cp_descr,cref);
                        break;
                  case S7_VFD_USTATE_IND:
                        my_get_vfd_ustate_ind(    cp_descr);
                        break;
                  default:
                        printf(      "Event unexpected",
                              ret);
      break;
            }
      }     while( (ret!=last_event_expected)&&
      (ret!=S7_ABORT_IND));
}

/* main */
void main(void)
{     ord32  cp_descr;
      ord16  cref;

      /* initialize s7 */
      my_init(&cp_descr);

      /* get reference for connection 'TEST' */
      my_get_cref(cp_descr,&cref);

      /* initiate connection */
      my_initiate_req(cp_descr,cref);

      /* receive vfd state confirmation */
      my_receive(cp_descr,S7_VFD_STATE_CNF);

      /* end communication */
      my_shut(cp_descr);
}
```

Ablaufdiagramm

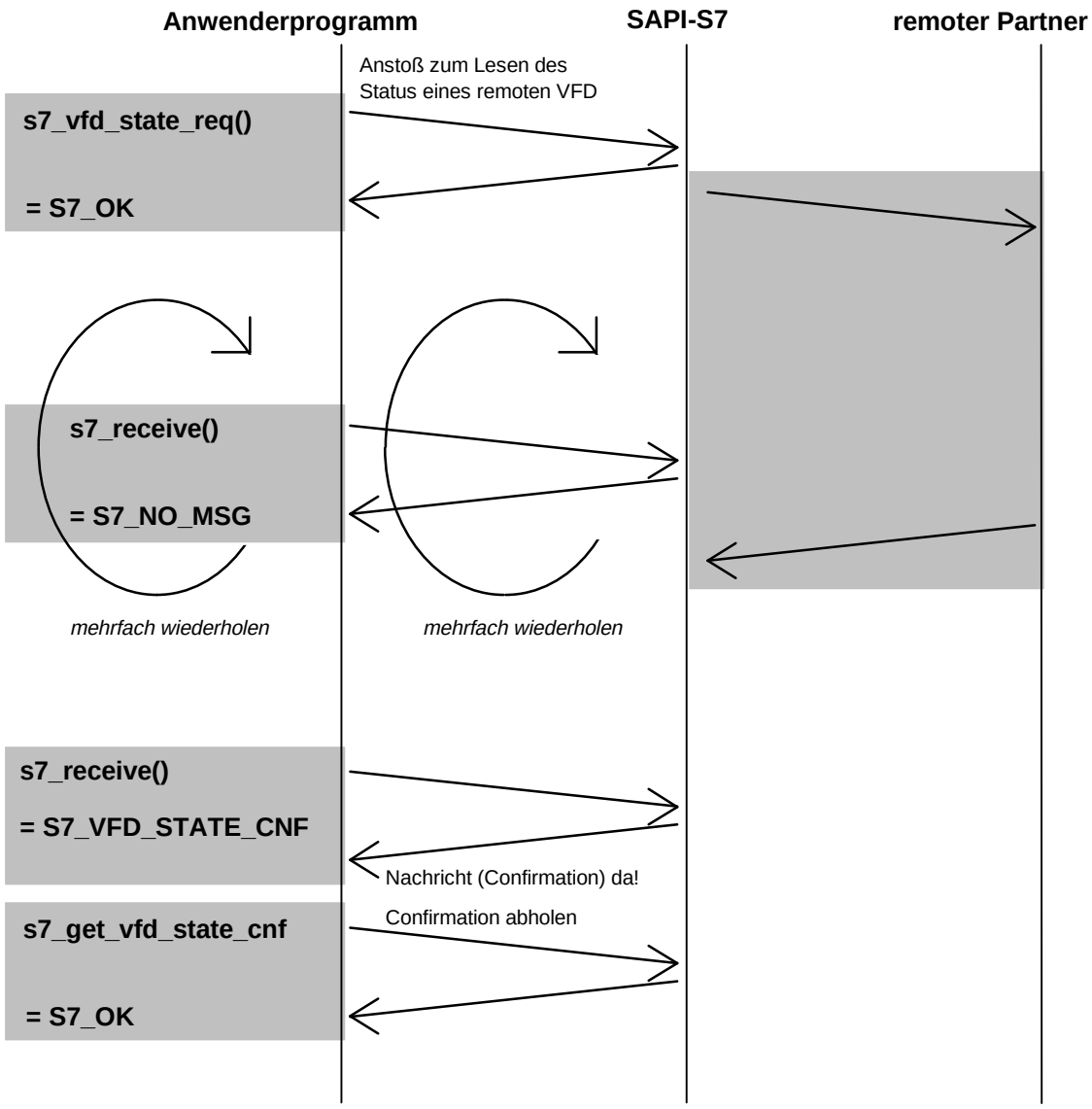

Bild 3.9: Ablaufdiagramm für das Beispiel

3.7.1 s7_vfd_state_req

Beschreibung

Mit dem Aufruf 's7_vfd_state_req()' kann eine Client-Applikation den logischen und physikalischen Status eines anderen (remoten) virtuellen Gerätes (VFD) lesen.

Deklaration

```
int32 s7_vfd_state_req(
      ord32           cp_descr,    /*  Vorgabe  */
      ord16           cref,        /*  Vorgabe  */
      ord16           orderid      /*  Vorgabe  */
           )
```

Parameter

cp_descr	Handle als Rückgabewert des 's7_init()'-Aufrufs.
cref	Referenz der S7-Verbindung, über die der Auftrag gesendet werden soll.
orderid	Auftragskennzeichen zur Identifizierung des abzugebenden Auftrags und der zugehörigen Quittung.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.7.2 s7_get_vfd_state_cnf

Beschreibung

Mit dem Aufruf 's7_get_vfd_state_cnf()' wird das Ergebnis eines VFD-Status-Auftrags entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_VFD_STATE_CNF', falls der Statusauftrag durchgeführt wurde. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_vfd_state_cnf()' zur internen Bearbeitung in der Library aufgerufen werden.

Mit dem Aufruf 's7_get_vfd_state_cnf()' werden die gelesenen Werte (der physikalische, der logische Status des VFD und der lokale Status der Anwendung) in Anwenderpuffer kopiert.

Deklaration

```
int32 s7_get_vfd_state_cnf(
      ord16   *log_state_ptr,       /* Rückgabe */
      ord16   *phy_state_ptr,       /* Rückgabe */
      ord8    *local_detail_ptr     /* Rückgabe */
                                )
```

Parameter

log_state_ptr — Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird der logische Status des VFD vermerkt. Dieser Parameter gibt an, welche Dienste momentan nutzbar sind. Hier ist 'S7_STATE_CHANGES_ALLOWED' als einziger Zustand möglich, d. h. es sind alle Dienste erlaubt.

phy_state_ptr — Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird der physikalische Status des VFD vermerkt. Der Wert dieses Parameters leitet sich aus den Zuständen der Betriebsmittel ab. Im Fall 'S7_OPERATIONAL' ist das VFD voll funktionsfähig, bei 'S7_NEEDS_COMMISSIONING' müssen noch lokale Eingriffe erfolgen, um in einen betriebsbereiten Zustand zu gelangen.

local_detail_ptr — Zeiger auf einen vom Anwenderprogramm bereitgestellten Puffer, der eine Mindestgröße von 3 Byte haben muß. Hier wird der lokale Status der Anwendung und des Gerätes abgelegt. Die Bedeutung der 3 Bytes ist VFD-spezifisch und in der zugehörigen VFD-Beschreibung des Servers nachzulesen.

Return-Werte	S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
	S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
	S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

3.7.3 s7_get_vfd_ustate_ind

Beschreibung

Mit dem Aufruf 's7_get_vfd_ustate_ind()' wird der vom Server unaufgefordert gesendete Geräte-/Anwenderstatus entgegengenommen.

Das Anwenderprogramm erhält beim 's7_receive()'-Aufruf die Anzeige 'S7_VFD_USTATE_IND', falls der remote Partner seinen Status unaufgefordert meldet. Danach muß die zugehörige Bearbeitungsfunktion 's7_get_vfd_ustate_ind()' zur internen Bearbeitung in der Library aufgerufen werden.

Mit dem Aufruf 's7_get_vfd_ustate_ind()' werden der logische und der physikalische Zustand des VFD in Anwenderpuffer übertragen.

Deklaration

```
int32 s7_get_vfd_ustate_ind(
      ord16   *log_state_ptr,       /* Rückgabe */
      ord16   *phy_state_ptr,       /* Rückgabe */
      ord8    *local_detail_ptr     /* Rückgabe */
                                )
```

Parameter

log_state_ptr — Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird der logische Status des VFD vermerkt. Dieser Parameter gibt an, welche Dienste momentan nutzbar sind. Hier ist 'S7_STATE_CHANGES_ALLOWED' als einziger Zustand möglich, d. h. es sind alle Dienste erlaubt.

phy_state_ptr — Adresse einer vom Anwenderprogramm bereitgestellten Variablen vom Typ 'ord16'. Hier wird der physikalische Status des VFD vermerkt. Der Wert dieses Parameters leitet sich aus den Zuständen der Betriebsmittel ab. Im Fall 'S7_OPERATIONAL' ist das VFD voll funktionsfähig, bei 'S7_NEEDS_COMMISSIONING' müssen noch lokale Eingriffe erfolgen, um in einen betriebsbereiten Zustand zu gelangen.

local_detail_ptr — Zeiger auf einen vom Anwenderprogramm bereitgestellten Puffer, der eine Mindestgröße von 3 Byte haben muß. Hier wird der lokale Status der Anwendung und des Gerätes abgelegt. Die Bedeutung der 3 Bytes ist VFD-spezifisch und in der zugehörigen VFD-Beschreibung nachzulesen.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

4 Trace und Mini-DB

Im folgenden Kapitel lernen Sie den Umgang mit den Trace- und den Mini-DB-Aufrufen kennen. Sie erfahren,

- ➢ wie Sie Einträge in den Library-eigenen Trace ermöglichen können.
- ➢ wie Sie Einstellungen in der Mini-DB abfragen oder ändern können.
- ➢ wie Sie Informationen zum zuletzt aufgetretenen Fehler erhalten.

Am Ende des Kapitels können Sie

- ➢ die komplette Funktionalität, die Ihnen S7 bietet, für Ihre Applikation nutzen unter Beibehaltung einer einfachen SAPI-S7-Programmierschnittstelle.
- ➢ Fehlersituationen durch Ihre Applikation erkennen und beseitigen.

4.1 s7_trace

Beschreibung

Mit diesem Aufruf können Eintragungen in den Trace der S7-Library vorgenommen werden. Dadurch ist es möglich, wichtige Daten zu Kontrollzwecken zu sichern, den Programmablauf zu kontrollieren bzw. mit den Trace-Eintragungen zu synchronisieren.

Deklaration

```
void s7_trace(
              char       *msg        /* Vorgabe */
               )
```

Parameter

msg	String mit der Anwendermeldung, die in den Trace eingetragen werden soll. Eine Trace-Zeile kann bis zu 78 Zeichen enthalten, wobei aber 14 Zeichen für die seit der Initialisierung des Trace verstrichene Zeit und die Zeilennummer reserviert sind. Somit ist eine Netto-Datenlänge von 64 Zeichen je Trace-Aufruf erreichbar. Längere Strings werden abgeschnitten.

Rückgabewert

Keiner

4.2 s7_write_trace_buffer

Beschreibung

Bei hochperformanten Anwendungen ist das Schreiben des Trace in eine Datei störend. Trotzdem sollen die Eintragungen in den Trace, z. B. im Fehlerfall, verfügbar sein. Deswegen wurde die Möglichkeit geschaffen, mit Hilfe der Mini-DB den Trace dahingehend zu konfigurieren, daß sämtliche Trace-Einträge in einen internen Umlaufpuffer erfolgen. Die gesamte Information kann dann mit der Funktion ‘s7_write_trace_buffer()’ in eine Datei geschrieben werden und steht somit für Fehleranalysen und Auswertung zur Verfügung.

Deklaration

```
void s7_write_trace_buffer(
                char   *filename    /* Vorgabe  */
                                )
```

Parameter

filename	Name der Datei, in die der interne Umlaufpuffer geschrieben werden soll.

Rückgabewert

Keiner

4.3 s7_mini_db_set

Beschreibung

Mit diesem Aufruf werden Einstellungen in der Mini-DB überschrieben, um den verschiedensten Anforderungen bezüglich Aufbau einer S7-Verbindung, Trace und Fehlerabfragen gerecht zu werden. Für die gewünschten Daten und deren Werte sind nur eine gewisse Anzahl von Kombinationen möglich, die im folgenden noch beschrieben werden.

Deklaration

```
int32 s7_mini_db_set(
                ord16   type,       /* Vorgabe  */
                char    *value      /* Vorgabe  */
                    )
```

Parameter

type	Kennzeichen für die zu ändernde Einstellung. Die möglichen Übergabewerte sind im folgenden beschrieben.
value	Neuer Wert für die zu ändernde Einstellung. Der Wert wird immer als String übergeben. Dafür stehen in der Header-Datei 'SAPI_S7.H' Defines zur Verfügung, die als ASCII-Zeichenkette dargestellten Zahlen entsprechen. Sind Kombinationen von Einzelwerten erlaubt, so müssen zunächst die einzelnen Defines in Ganzzahlen gewandelt, oderverknüpft und das Ergebnis wieder in einen String zurückgewandelt werden.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

Wertekombinationen für den Trace

Der Trace stellt eine einfache und trotzdem effektive Debug-Hilfe für die S7-Library dar. Er kann den verschiedensten Anwendungsfällen angepaßt werden. Die erlaubten Wertekombinationen werden im folgenden ausgehend vom Parameter 'type' beschrieben.

Ab den SAPI-S7 Bibliotheken der Version V 1.371.2002 besteht die Möglichkeit, die Trace-Ausgabe auf mehrere Dateien aufzuteilen. Hierdurch wird ein zu großes Anwachsen der Trace-Datei verhindert. Die Anzahl und Größe der Dateien läßt sich mit den Parametern S7_ MINI_DB_TRACE_MAXFILES und S7_ MINI_DB _TRACE_MAXLINES einstellen.

S7_MINI_DB_TRACE_FILENAME

Mit diesem Parameterwert wird der Name der Trace-Datei bestimmt. Als 'value' wird der Dateiname übergeben (Default: 'S7TRACE.TXT' im aktuellen Arbeitsverzeichnis).

S7_MINI_DB_TRACE_TARGET

Dieser Wert legt das Target für den Trace fest.

Parameter 'value'	Beschreibung
S7_TRACE_TARGET_BUFFER	Die Trace-Einträge werden in einen internen Umlaufpuffer geschrieben (Default-Einstellung).
S7_TRACE_TARGET_OLD_FILE	Die Trace-Einträge werden in eine Datei geschrieben. Eine eventuell schon bestehende Datei bleibt unverändert, anschließende Trace-Einträge werden an die Datei angehängt.
S7_TRACE_TARGET_NEW_FILE	Die Trace-Einträge werden in eine Datei geschrieben. Die Datei wird neu erstellt, anschließende Trace-Einträge werden an die Datei angehängt. Eine überdimensional anwachsende Trace-Datei kann hiermit vermieden werden.

S7_TRACE_TARGET_CONSOLE	Die Trace-Einträge werden an eine andere Applikation weitergeleitet. Das weitere Verhalten ist betriebssystemabhängig.

S7_ MINI_DB_TRACE_MAXFILES

Der Trace ist ein Umlaufpuffer mit einer Anzahl von S7_MINI_DB_TRACE_MAXFILES- Dateien. Ist eine Datei voll beschrieben, wird eine neue Datei angelegt. Sind alle Dateien beschrieben, wird immer die älteste überschrieben.

Es sind Werte zwischen 1 und 999 einstellbar. Der Defaultwert ist **2**.

S7_ MINI_DB _TRACE_MAXLINES

Hiermit wird die Größe der S7_Trace-Dateien festgelegt

Es sind Werte zwischen 1 und 2^2-1 einstellbar. Die Größe sollte dem verfügbaren Speicherplatz angepaßt werden.

Der Defaultwert ist **10.000**.

S7_MINI_DB_TRACE_DEPTH

Hiermit wird die Trace-Tiefe eingestellt.

Parameter 'value'	Beschreibung
S7_TRACE_DEPTH_OFF	Der Trace wird ausgeschaltet.
S7_TRACE_DEPTH_USER	Mit diesem Wert (Default) werden nur die vom Anwenderprogramm durch die Funktion 's7_trace()' vorgegebenen Strings in den Trace eingetragen
S7_TRACE_DEPTH_EXCEPT	Diese Einstellung läßt nur Trace-Einträge nach fehlerhaften Ereignissen oder Ergebnissen zu.
S7_TRACE_DEPTH_INTERFACE	Mit diesem Wert werden darüber hinaus Übergabeparameter an der SAPI-S7-Programmierschnittstelle in den Trace eingetragen. Damit können fehlerhafte Parameter schnell und ohne Debugger erkannt werden.
S7_TRACE_DEPTH_OTHER	Diese Einstellung liefert weitere Informationen.

S7_MINI_DB_TRACE_SELECT

Um den Trace dienstklassenspezifisch ein- oder auszuschalten, steht je Dienstklasse ein Define in der Header-Datei 'SAPI_S7.H' zur Verfügung. Die Defines können wie oben beschrieben kombiniert werden.

Parameter 'value'	Beschreibung
S7_TRACE_SELECT_ADMIN_SERVICES	Der Trace wird aktiviert für die administrativen Dienste.
S7_TRACE_SELECT_CONN_SERVICES	Der Trace wird aktiviert für die S7-Verbindungsmanagement-Dienste.
S7_TRACE_SELECT_VAR_SERVICES	Der Trace wird aktiviert für die Variablendienste.
S7_TRACE_SELECT_CYCL_VAR_SERVICES	Der Trace wird aktiviert für die zyklischen Variablendienste.
S7_TRACE_SELECT_RECEIVE_SERVICES	Der Trace wird aktiviert für den Empfangsaufruf.
S7_TRACE_SELECT_VFD_SERVICES	Der Trace wird aktiviert für die VFD-Dienste.
S7_TRACE_SELECT_OTHER_SERVICES	Der Trace wird aktiviert für die Zugriffsfunktionen auf die Mini-DB.
S7_TRACE_SELECT_PBK_SERVICES	Der Trace wird aktiviert für die 'Blockorientierten Dienste'.
S7_TRACE_SELECT_ALL	Der Trace wird aktiviert für sämtliche Dienstklassen.

S7_MINI_DB_TRACE_NO_LINES

Mit diesem Parameter kann die Anzahl der Zeilen des internen Umlaufpuffers verändert werden. Für speicherplatzintensive Applikationen kann der Umlaufpuffer verkleinert, für das Auftreten von Fehlern zwecks Aufzeichnung der Vorgeschichte vergrößert werden. Die Funktion 's7_write_trace_buffer()' erlaubt das Schreiben des Umlaufpuffers in eine Datei.

Die Änderung dieses Parameters ist nur wirksam, wenn sie vor dem ersten S7-Library-Aufruf geschieht.

Wertekombinationen für den Aufbau von S7-Verbindungen

Für den S7-Verbindungsaufbau können sowohl auf der aktiven als auch auf der passiven Seite verschiedene Verbindungsparameter vorbesetzt werden, die zwischen den beiden Stationen ausgehandelt werden. Die erlaubten Wertekombinationen sind im folgenden ausgehend vom Parameter 'type' beschrieben.

S7_MINI_DB_INIT_REQ_AMQ_CALLING

Mit diesem Wert wird für den aktiven Partner vor dem Verbindungsaufbau festgelegt, wieviele Aufträge mit Quittungen auf dieser Verbindung gleichzeitig empfangen werden können (Default: ‘3’). Hierbei handelt es sich um einen Vorschlagswert, der von der Partnerstation akzeptiert oder heruntergehandelt werden kann. Der ausgehandelte Wert kann mit Hilfe der Funktion ‘s7_mini_db_get()’ ausgelesen werden.

S7_MINI_DB_INIT_REQ_AMQ_CALLED

Mit diesem Wert wird für den aktiven Partner vor dem Verbindungsaufbau festgelegt, wieviele Aufträge mit Quittungen auf dieser Verbindung gleichzeitig gesendet werden können (Default: ‘3’). Hierbei handelt es sich um einen Vorschlagswert, der von der Partnerstation akzeptiert oder heruntergehandelt werden kann. Der ausgehandelte Wert kann mit Hilfe der Funktion ‘s7_mini_db_get()’ ausgelesen werden.

S7_MINI_DB_INIT_REQ_PDU_SIZE

Mit diesem Wert wird für den aktiven Partner vor dem Verbindungsaufbau die maximale Größe einer PDU auf dieser Verbindung vorgegeben (Default: ‘0x100’). Hierbei handelt es sich um einen Vorschlagswert, der von der Partnerstation akzeptiert oder heruntergehandelt werden kann. Der ausgehandelte Wert kann mit Hilfe der Funktion ‘s7_mini_db_get()’ ausgelesen werden.

S7_MINI_DB_INIT_RSP_AMQ_CALLING

Mit diesem Wert wird für den passiven Partner vor dem Verbindungsaufbau festgelegt, wieviele Aufträge mit Quittungen auf dieser Verbindung gleichzeitig gesendet werden können (Default: ‘3’). Ausgehandelt wird das Minimum der entsprechenden Größen auf der aktiven und auf der passiven Seite.

S7_MINI_DB_INIT_RSP_AMQ_CALLED

Mit diesem Wert wird für den passiven Partner beim Verbindungsaufbau festgelegt, wieviele Aufträge mit Quittungen auf dieser Verbindung gleichzeitig empfangen werden können (Default: ‘3’). Ausgehandelt wird das Minimum der entsprechenden Größen auf der aktiven und auf der passiven Seite.

S7_MINI_DB_INIT_RSP_PDU_SIZE

Mit diesem Wert wird für den passiven Verbindungsaufbau die maximale Größe einer PDU auf dieser Verbindung vorgegeben (Default: ‘0x100’). Ausgehandelt wird das Minimum der PDU-Größen auf der aktiven und auf der passiven Seite.

S7_MINI_DB_PERSISTANCE_COUNT

Mit diesem Wert wird die Anzahl der Wiederholungsversuche für einen aktiven Verbindungsaufbau gesetzt (Default: ‘5’). Erst wenn die Partnerstation entsprechend oft den Aufbauwunsch abgelehnt hat, wird der Verbindungsaufbau beendet und dem Anwenderprogramm negativ quittiert.

S7_MINI_DB_ABORT_TIMEOUT

Mit diesem Wert wird die maximale Zeitdauer für Sendewiederholungen festgelegt, falls die remote Station nicht antwortet. Die Festlegung erfolgt in Vielfachen von 51 ms (Default: ‘300’). Der Parameter gilt sowohl für die Verbindungsaufbauphase als auch für die Datentransferphase.

4.4 s7_mini_db_get

Beschreibung

Mit diesem Aufruf werden Einstellungen aus der Mini-DB gelesen. Das Anwenderprogramm gibt ein Kennzeichen für die zu lesende Einstellung vor und erhält einen String zurück, der abhängig vom Kennzeichen interpretiert werden muß.

Deklaration

```
const char *s7_mini_db_get(
                ord16      type        /* Vorgabe  */
                              )
```

Parameter

type — Kennzeichen für die zu lesende Einstellung. Die möglichen Werte sind im folgenden beschrieben.

Rückgabewerte für Trace-Einstellungen

Die Mini-DB erlaubt bezüglich des Trace jederzeit das Lesen aller änderbaren Einstellungen. Dies sind:

S7_MINI_DB_TRACE_FILENAME

Mit diesem Parameter wird der Name der Trace-Datei zurückgegeben.

S7_MINI_DB_TRACE_TARGET

Mit diesem Parameter wird das Target des Traces ausgegeben.

S7_MINI_DB_TRACE_DEPTH

Mit diesem Parameter kann die Trace-Tiefe abgefragt werden.

S7_MINI_DB_TRACE_SELECT

Mit diesem Parameter werden die Dienstklassen, für die der Trace aktiviert wurde, angezeigt.

Die Rückgabewerte entsprechen den Vorgabewerten beim Aufruf 's7_mini_db_set()' und können dort nachgelesen werden.

Rückgabewerte für Aufbau einer S7-Verbindung

Nach Empfang einer Initiate-Confirmation werden von der entsprechenden Bearbeitungsfunktion 's7_get_initiate_cnf()' zwischen Client und Server ausgehandelte Leistungsparameter in die Mini-DB eingetragen.

S7_MINI_DB_INIT_IND_AMQ_CALLING

Mit diesem Wert wird dem passiven Partner nach Erhalt der Initiate-Indication angezeigt, wieviele Aufträge der aktive Partner auf dieser Verbindung gleichzeitig empfangen kann.

S7_MINI_DB_INIT_IND_AMQ_CALLED

Mit diesem Wert wird dem passiven Partner nach Erhalt der Initiate-Indication angezeigt, wieviele Aufträge der aktive Partner auf dieser Verbindung gleichzeitig senden kann.

S7_MINI_DB_INIT_IND_PDU_SIZE

Mit diesem Wert wird dem passiven Partner nach Erhalt der Initiate-Indication angezeigt, wieviele Daten der aktive Partner auf dieser Verbindung empfangen kann.

S7_MINI_DB_INIT_CNF_AMQ_CALLING

Mit diesem Wert wird nach dem aktiven Verbindungsaufbau die Anzahl der Aufträge mit Quittungen ausgelesen, die auf dieser Verbindung gleichzeitig empfangen werden können. Der Wert wurde zwischen den beiden Kommunikationspartnern beim Verbindungsaufbau ausgehandelt.

S7_MINI_DB_INIT_CNF_AMQ_CALLED

Mit diesem Wert wird nach dem aktiven Verbindungsaufbau die Anzahl der Aufträge mit Quittungen ausgelesen, die auf dieser Verbindung gleichzeitig gesendet werden können. Der Wert wurde zwischen den beiden Kommunikationspartnern beim Verbindungsaufbau ausgehandelt.

S7_MINI_DB_INIT_CNF_PDU_SIZE

Mit diesem Wert wird nach dem aktiven Verbindungsaufbau die maximale Größe einer PDU auf dieser Verbindung ausgelesen. Der Wert wurde zwischen den beiden Kommunikationspartnern beim Verbindungsaufbau ausgehandelt.

4.5 s7_last_iec_err_no

Beschreibung

Sämtliche Fehlerkennungen werden von der S7-Library auf ein handhabbares Maß abgebildet, um die Fehlerbehandlung in Applikationen einfacher gestalten zu können. Nach IEC 1131 (**I**nternational **E**lectrotechnical **C**ommission) bestehen Norm-Fehlercodes, die mit diesem Aufruf ausgelesen werden können.

Deklaration

```
ord16 s7_last_iec_err_no(void)
```

Parameter

Keine

Rückgabewerte

Mögliche Rückgabewerte und deren Bedeutungen im einzelnen:

S7_ERR_IEC_NO

Es ist kein Fehler aufgetreten.

S7_ERR_IEC_DATA_TYPE_MISMATCH

Die Datentypen stimmen nicht überein.

S7_ERR_IEC_INVALID_REF

Die angegebene S7-Verbindungsreferenz existiert nicht.

S7_ERR_IEC_LOWER_LAYER

Es ist ein Fehler in den tieferen Schichten aufgetreten.

S7_ERR_IEC_NEG_RESPONSE

Der Client hat eine negative Quittung vom Kommunikationspartner erhalten.

S7_ERR_IEC_NO_ACCESS_TO_REM_OBJECT

Der Zugriff auf ein Objekt wurde abgelehnt.

S7_ERR_IEC_PARTNER_IN_WRONG_STATE

Die Partnerstation befindet sich in einem Zustand, in dem der abgegebene Auftrag nicht bearbeitet werden kann.

S7_ERR_IEC_RECEIVER_DISABLED

Der Server antwortet nicht.

S7_ERR_IEC_RECEIVER_OVERRUN

Die Ressourcen im Server sind erschöpft.

S7_ERR_IEC_RESET_RECEIVED

Es ist eine Anforderung zum Reset eingetroffen.

4.6 s7_last_iec_err_msg

Beschreibung Dieser Aufruf liefert zum vorhandenen IEC-Fehlercode einen String zurück, der den aufgetretenen Fehler beschreibt. Es handelt sich dabei um einen englischsprachigen Fehlerstring, der z. B. für Ausgaben auf einer Operator-Console, in eine Protokolldatei etc. verwendet werden kann.

Deklaration

```
const char *s7_last_iec_err_msg(void)
```

Parameter Keine

4.7 s7_last_detailed_err_no

Beschreibung Mit diesem Aufruf erhält der Aufrufer eine Fehlernummer, die detailliertere Angaben zur Fehlerursache macht, als die genormten IEC-Fehlercodes.

Deklaration

```
ord16 s7_last_detailed_err_no(void)
```

Parameter Keine

Rückgabewerte Mögliche Rückgabewerte und deren Bedeutungen im einzelnen:

S7_ERR_NO_ERROR

Es ist kein Fehler aufgetreten.

S7_ERR_CONN_ABORTED

Die S7-Verbindung wurde abgebrochen.

S7_ERR_CONN_CNF

Die S7-Verbindung konnte nicht aufgebaut werden.

S7_ERR_CONN_NAME_NOT_FOUND

Der angegebene S7-Verbindungsname konnte nicht gefunden werden.

S7_ERR_FW_ERROR

Auf der Anschaltung ist ein Firmware-Fehler aufgetreten.

S7_ERR_INSTALL

Beim Installieren des SIMATIC NET-Treibers bzw. beim Initialisieren der Anschaltung ist ein Fehler aufgetreten, der eine Kommunikation unmöglich macht.

S7_INTERNAL_ERROR

Bei der Kommunikation wurden library-interne Daten überschrieben, die einen weiteren Betrieb der Applikation nicht zulassen.

S7_ERR_INVALID_CONN_STATE

Der abgegebene Auftrag ist im aktuellen Zustand der S7-Verbindung nicht zulässig.

S7_ERR_INVALID_CREF

Die angegebene S7-Verbindungssreferenz ist ungültig.

S7_ERR_INVALID_CYCL_READ_STATE

Der Auftrag ist im aktuellen Status des zyklischen Lese-Auftrag nicht erlaubt.

S7_ERR_INVALID_DATARANGE_OR_TYPE

Eingangsparameter der aufgerufenen Funktion außerhalb des gültigen Wertebereichs

S7_ERR_INVALID_DATA_SIZE

Der vom Anwenderprogramm bereitgestellte Datenpuffer ist zu klein.

S7_ERR_INVALID_ORDERID

Es besteht kein Auftrag mit dem angegebenen Auftragskennzeichen (Parameter 'orderid').

S7_ERR_INVALID_PARAMETER

Ein Übergabeparameter oder ein Vorgabewert in einer übergebenen Struktur ist nicht gültig.

S7_ERR_MAX_REQ

Die maximale Anzahl an Aufträgen mit Quittung, wie beim Verbindungsaufbau ausgehandelt, wurde schon abgesetzt.

S7_ERR_MINI_DB_TYPE

Der Parameter 'type' bei einem Mini-DB-Aufruf ist nicht erlaubt.

S7_ERR_MINI_DB_VALUE

Der Parameter 'value' bei einem Mini-DB-Aufruf ist nicht erlaubt.

S7_ERR_NO_LICENCE

Die für das Produkt notwendige Lizenz konnte nicht gefunden werden.

S7_ERR_NO_RESOURCE

Die zur Verfügung stehenden Ressourcen sind momentan erschöpft.

S7_ERR_NO_SIN_SERV

Der für S7-Applikationen unter Windows notwendige SIMATIC NET Server, der Nachrichten an die zuständige Applikation schickt, konnte nicht gestartet werden.

S7_ERR_OBJ_ACCESS_DENIED

Der Zugriff auf das gewünschte Objekt wurde abgelehnt.

S7_ERR_OBJ_ATTR_INCONSISTENT

Das OD oder die Attribute des angesprochenen Objekts sind inkonsistent.

S7_ERR_OBJ_UNDEFINED

Das Objekt, auf das zugegriffen werden soll, existiert nicht.

S7_ERR_ORDERID_USED

Das beim Aufruf übergebene Auftragskennzeichen (Parameter 'orderid') wird bereits benutzt.

S7_ERR_RECEIVE_BUFFER_FULL

Es wurde zwar eine Nachricht empfangen, die entsprechende Bearbeitungsfunktion aber noch nicht aufgerufen.

S7_ERR_SERVICE_NOT_SUPPORTED

Der angeforderte Dienst wird nicht unterstützt.

S7_ERR_SERVICE_VFD_ALREADY_USED

Die Applikation oder ein anderer Prozeß hat sich schon auf das VFD angemeldet.

S7_ERR_SYMB_ADDRESS

Die im Auftrag übergebene symbolische Adresse ist fehlerhaft.

S7_ERR_SYMB_ADDRESS_INCONSISTENT

Die in der symbolischen Adresse enthaltene Größe der Anwenderdaten und die Größe des Anwenderpuffers sind widersprüchlich.

S7_ERR_TOO_LONG_DATA

Es sollten mehr Daten geschrieben werden als nach Norm zulässig.

S7_ERR_UNKNOWN_ERROR

Es ist ein unbekannter Fehler aufgetreten.

S7_ERR_WRONG_CP_DESCR

Der beim Aufruf angegebene CP-Descriptor ist nicht korrekt.

S7_ERR_WRONG_IND_CNF

Es wurde zu einer empfangenen Nachricht die falsche Bearbeitungsfunktion aufgerufen.

4.8 s7_last_detailed_err_msg

Beschreibung Dieser Aufruf liefert zur vorhandenen, detaillierten Fehlernummer eine Fehlermeldung zurück, die den aufgetretenen Fehler beschreibt und Hinweise zur Fehlerbehebung gibt. Es handelt sich dabei um einen englischsprachigen Fehlerstring, der z. B. für Ausgaben auf einer Operator-Console, in eine Protokolldatei etc. verwendet werden kann.

Deklaration

```
const char *s7_last_detailed_err_msg(void)
```

Parameter Keine

4.9 s7_discard_msg

Beschreibung

Mit diesem Aufruf kann eine empfangene Nachricht verworfen werden, ohne die entsprechende Bearbeitungsfunktion aufgerufen zu haben.

Für jede S7-Verbindung können nur eine maximal festgelegte und beim Aufbau ausgehandelte Anzahl von quittierten Aufträgen gleichzeitig bearbeitet werden. Das Ignorieren von (z. B. unerwarteten) Ereignissen und damit das Fehlen von Quittungen vermindert dauerhaft die Anzahl der gleichzeitig nutzbaren Aufträge mit Quittungen.

Deklaration

```
void s7_discard_msg(void)
```

Parameter

Keine

Notizen

5 Projektierung

In diesem Kapitel

- ➢ lernen Sie die Bedeutung der Projektierung kennen,
- ➢ erhalten Sie einen Überblick über die zum Betrieb notwendigen Projektierungsparameter,
- ➢ finden Sie eine Zusammenfassung der Dienste, die Projektierungsparameter verwenden.

Am Ende des Kapitels sind Sie in der Lage, Ihre SAPI-S7-Applikation auf die Projektierung abzustimmen.

5.1 Bedeutung der Projektierung

Projektierung erhöht die Flexibilität Ihrer Applikation

Um Applikationen von Anpassungen bei Änderungen im Kommunikationssystem (Netz) freizuhalten, erfolgt die Bereitstellung und Zuordnung von S7-Verbindungen mittels Projektierung. Die Projektierung ist eine standardisierte Methode, Adreßparameter usw. für alle Applikationen einstellbar zu machen. In der Regel wird die Inbetriebnahme von Software und deren Integration ins Netz nicht vom Softwareentwickler durchgeführt.

Bei einer Umprojektierung ist darauf zu achten, daß die für die Applikation sichtbaren Projektierungsparameter weiterhin Bestand haben. So muß z. B. der Name einer S7-Verbindung immer noch existieren, auch wenn über diese Verbindung eine andere Partnerstation, u. U. sogar mit einer anderen S7-Verbindungsreferenz, angesprochen werden soll. Anpassungen können unter dieser Randbedingung mit den entsprechenden Projektierungswerkzeugen jederzeit ohne Eingriff in die Anwenderprogramme vorgenommen werden.

In Kapitel 6.5 ist der Zugriff auf die Projektierung beschrieben.

5.2 Dienste mit Projektierdaten

Administrative Dienste

Von den administrativen Diensten benötigt die Anmeldefunktion 's7_init()' zwei Vorgaben:

- zum einen muß der CP-Name, der einen CP identifiziert, vorgegeben werden und mit einem bei der Installation festgelegten Namen übereinstimmen; die Installation wird von den SIMATIC NET Produkten übernommen. Die notwendigen Beschreibungen liegen dem jeweiligen Produkt bei;
- zum anderen wird der Name des lokalen VFD benötigt, auf das sich das Anwenderprogramm anmelden will. Für die Applikation werden durch die Anmeldung die VFD-eigenen S7-Verbindungen zugänglich. Der VFD-Name wird per Projektierung festgelegt.

Die Namen der installierten CPs bzw. die Namen der auf einem CP projektierten VFDs können mit den Aufrufen ‘s7_get_device()’ bzw. ‘s7_get_vfd()’ abgefragt werden.

Managementdienste für S7-Verbindungslisten

Von den Managementdiensten für S7-Verbindungslisten benötigt lediglich der Aufruf 's7_get_cref()' einen Projektierungsparameter in Form des S7-Verbindungsnamens. Diese Funktion liefert die Referenz für eine S7-Verbindung zurück. Eine Applikation sollte im weiteren Verlauf die so ermittelte Referenz verwenden.

Die Namen aller projektierten S7-Verbindungen können mit dem Aufruf ‘s7_get_conn()’ ausgelesen werden.

Notizen

6 SAPI-S7 unter MS-DOS/Windows

Im folgenden Kapitel lernen Sie die betriebssystemspezifischen Eigenschaften der SAPI-S7-Programmierschnittstelle für MS-DOS und Windows kennen. Dabei werden mit der Bezeichnung „Windows“ die Windows-Betriebssysteme Windows 3.x (3.1 und 3.11) im sogenannten „enhanced mode 386“, Windows 95 sowie Windows NT zusammengefaßt, falls nicht eines dieser Systeme explizit benannt wird.

Sie erfahren,

- ➢ für welche Compiler und Speichermodelle die S7-Library unter MS-DOS und Windows zur Verfügung steht,
- ➢ welche Compileroptionen bei der Übersetzung eigener Applikationen sinnvoll sind,
- ➢ welche Linker-Optionen beim Linken Ihrer Programmodule mit der S7-Library notwendig sind,
- ➢ wie Sie den Trace durch Environmentvariablen steuern können, ohne Ihre Applikation ändern zu müssen.

Am Ende des Kapitels können Sie

- ➢ eigene Programmodule übersetzen und zusammen mit der S7-Library zu einem ausführbaren Programm binden,
- ➢ die Trace-Ausgaben Ihrer Applikation steuern.

6.1 Allgemeines

Speichermodelle

Die SAPI-S7-Programmierschnittstelle wird in Form von Libraries zur Verfügung gestellt. Die für die Benutzung der Programmierschnittstelle notwendigen Definitionen sind in der Datei 'SAPI_S7.H' hinterlegt. Es werden Libraries für MS-DOS, Windows 3.x sowie für Windows 95 und Windows NT für verschiedene Compiler angeboten.

Für MS-DOS und Windows 3.x setzen sich die Namen dieser Libraries wie folgt zusammen:

<Speichermodell><Betriebssystem>s7<Compiler>.lib

Die nachfolgende Tabelle erklärt die Komponenten, aus denen sich der Library-Namen zusammen setzt:

Formatbezeichnung	Formateintrag	Bedeutung
<Speichermodell>	l	Large Model
	h	Huge Model
<Betriebssystem>	d	MS DOS
	w	Windows 3.x
<Compiler>	msc	MSC-Compiler 7.0
	tc	Turbo-C-Compiler 1.0
	bc	Borland-C-Compiler 3.1

Beispielsweise handelt es sich bei der Datei ‘LDS7TC.LIB’ um die mit dem Turbo-C-Compiler im Speichermodell ‘Large’ übersetzte Library für das Betriebssystem MS-DOS.

Für das Betriebssystem Windows 3.x steht mit der Datei ‘S7.DLL’ eine DLL-Version (**D**ynamic **L**ink **L**ibrary) zur Verfügung mit den Import-Libraries ‘S7MSC.LIB’ und ‘S7BC.LIB’ für den Microsoft-C- bzw. Borland-C-Compiler.

Für Windows 95 und Windows NT wird eine 32-Bit-DLL 'S732.DLL’ und die zugehörige Importlibrary 'S7MSC.LIB’ zur Verfügung gestellt.

Alignment

Normalerweise werden Variablen vom Compiler in einer Form im Speicher abgelegt, wie es für den Compiler am sinnvollsten erscheint. Hierbei können zwischen Komponenten einer Variablen Lücken entstehen (padding bytes).

Die Strukturen, die an der SAPI-S7-Programmierschnittstelle angeboten werden, sind so ausgelegt, daß mit Byte- oder Word-Alignment übersetzte Anwenderprogramme problemlos auf die einzelnen Komponenten zugreifen können. Double-Word-Alignment wird von der S7-Library nicht unterstützt.

Programmabbruch

Anwenderprogramme müssen sich zwecks Kommunikation beim Kommunikationssystem anmelden, das zur Verwaltung Ressourcen belegt. Wird eine Applikation durch die Tastenkombination 'CTRL+C' abgebrochen, bleiben die Ressourcen für den Prozeß reserviert, und die Anmeldung hat weiterhin Bestand. Um dies zu vermeiden, sollte im Anwenderprogramm ein 'CTRL+C'-Handler implementiert sein, der bei Programmabbruch sämtliche Abmeldungen beim Kommunikationssystem übernimmt.

6.2 Übersetzen und Binden für MS-DOS

Voraussetzungen

In den folgenden Kapiteln werden Generierbeispiele aufgelistet, die Ihnen die notwendigen Compiler- und Linker-Optionen für Ihre Applikationen vermitteln.

Die absoluten und auf Ihrem Zielrechner gültigen Laufwerks- und Pfadangaben müssen von Ihnen in die Generieranweisung eingebracht bzw. im Suchpfad aufgenommen werden.

Arbeiten mit dem MSC-Compiler 7.0

Die S7-Library für den MSC-Compiler 7.0 hat unter MS-DOS den Namen 'LDS7MSC.LIB'. Das folgende Beispiel zeigt, wie ein Beispielprogramm 'BSP.C' mit dem Speichermodell 'Large' für MS-DOS übersetzt und gebunden wird:

```
cl /c /AL /Os /Iinc src\bsp.c
link @bsp.lnk
```

In der Datei 'BSP.LNK' stehen die Anweisungen für den Linker:

```
bsp.obj,
bsp.exe,
bsp.map,
lds7msc.lib+
\msc70\lib\oldnames.lib+
\msc70\lib\llibce.lib
;
```

Arbeiten mit dem MS Visual C++-Compiler 1.0

Wenn Sie den MS Visual C++-Compiler 1.0 für Ihre MS-DOS-Applikationen einsetzen wollen, so können Sie dieselbe S7-Library wie für den MSC-Compiler 7.0 verwenden. Die Generieranweisungen für das Speichermodell 'Large' haben dann unter MS-DOS folgendes Aussehen:

```
cl /c /AL /Os /Iinc src\bsp.c
link @bsp.lnk
```

In der Datei 'BSP.LNK' stehen die Anweisungen für den Linker:

```
bsp.obj,
bsp.exe,
bsp.map,
lds7msc.lib+
\msvc\lib\oldnames.lib+
\msvc\lib\llibce.lib
;
```

Arbeiten mit dem Turbo-C-Compiler 1.0

Für den Turbo-C-Compiler 1.0 liegen für MS-DOS zwei Ausprägungen der S7-Library vor: 'LDS7TC.LIB' für das Speichermodell 'Large' und 'HDS7TC.LIB' für das Speichermodell 'Huge'. Das folgende Beispiel zeigt, wie ein Beispielprogramm 'BSP.C' mit dem Speichermodell 'Large' für MS-DOS übersetzt und gebunden wird:

```
tcc -c -ml -Iinc src\bsp.c
tlink @bsp.lnk.
```

In der Datei 'BSP.LNK' stehen die Anweisungen für den Linker:

```
\tc10\lib\c0l.obj bsp.obj
bsp.exe
bsp.map
\tc10\lib\emu.lib \tc10\lib\mathl.lib \tc10\lib\cl.lib LDS7TC.LIB
```

6.3 Übersetzen und Binden für Windows 3.x

Voraussetzungen

In den folgenden Kapiteln werden Generierbeispiele aufgelistet, die Ihnen die notwendigen Compiler- und Linker-Optionen für Ihre Applikationen vermitteln sollen.

Die absoluten und auf Ihrem Zielrechner gültigen Laufwerks- und Pfadangaben müssen von Ihnen in die Generieranweisung eingebracht bzw. im Suchpfad aufgenommen werden.

Arbeiten mit dem MSC-Compiler 7.0

Die S7-Library für den MSC-Compiler 7.0 hat unter Windows 3.x den Namen 'LWS7MSC.LIB'. Das folgende Beispiel zeigt, wie ein Beispielprogramm 'BSP.C' mit dem Speichermodell 'Large' für Windows 3.x übersetzt und gebunden wird:

```
cl /c /AL /Os /Iinc src\bsp.c
link @bsp.lnk
```

In der Datei 'BSP.LNK' stehen die Anweisungen für den Linker:

```
bsp.obj,
bsp.exe,
bsp.map,
lws7msc.lib+
\msc70\lib\oldnames.lib +
\msc70\lib\llibcew.lib +
\msc70\lib\libw.lib
bsp.def;
```

Die Moduldefinitionsdatei 'bsp.def' hat folgendes Aussehen:

```
NAME        BSP
EXETYPE     WINDOWS
CODE        PRELOAD MOVEABLE DISCARDABLE
DATA        PRELOAD MOVEABLE MULTIPLE
HEAPSIZE    1024
STACKSIZE   10240
EXPORTS
```

Arbeiten mit dem MS Visual C++-Compiler 1.0

Wenn Sie den MS Visual C++-Compiler 1.0 für Ihre Windows-Applikationen einsetzen wollen, so können Sie dieselbe S7-Library wie für den MSC-Compiler 7.0 verwenden. Die Generieranweisungen für das Speichermodell 'Large' haben dann unter Windows 3.x folgendes Aussehen:

```
cl /c /AL /Os /Iinc src\bsp.c
link @bsp.lnk
```

In der Datei 'BSP.LNK' stehen die Anweisungen für den Linker:

```
bsp.obj,
bsp.exe,
bsp.map,
lws7msc.lib+
\msvc\lib\oldnames.lib +
\msvc\lib\llibcew.lib +
\msvc\lib\libw.lib
bsp.def;
```

Die Moduldefinitionsdatei 'bsp.def' hat folgendes Aussehen:

```
NAME        BSP
EXETYPE     WINDOWS
CODE        PRELOAD MOVEABLE DISCARDABLE
DATA        PRELOAD MOVEABLE MULTIPLE
HEAPSIZE    1024
STACKSIZE   10240
EXPORTS
```

Arbeiten mit dem Borland-C-Compiler 3.1

Die S7-Library für den Borland-C-Compiler 3.1 hat unter Windows 3.x den Namen 'LWS7BC.LIB'. Das folgende Beispiel zeigt, wie ein Beispielprogramm 'BSP.C' mit dem Speichermodell 'Large' für Windows 3.x übersetzt und gebunden wird:

```
bcc -c -ml -Os -Iinc src\bsp.c
tlink /Twe @bsp.lnk
```

In der Datei 'BSP.LNK' stehen die Anweisungen für den Linker:

```
\bc31\lib\c0wl.obj bsp.obj
bsp.exe
bsp.map
lws7bc.lib \bc31\lib\mathwl.lib \bc31\lib\import.lib
\bc31\lib\cwl.lib
bsp.def
```

Die Moduldefinitionsdatei 'bsp.def' hat folgendes Aussehen:

```
NAME        BSP
EXETYPE     WINDOWS
CODE        PRELOAD MOVEABLE DISCARDABLE
DATA        PRELOAD MOVEABLE MULTIPLE
HEAPSIZE    1024
STACKSIZE   10240
EXPORTS
```

SAPI-S7 als DLL-Version

Unter dem Betriebssystem Windows 3.x steht neben den oben aufgeführten Libraries auch eine DLL-Version (**D**ynamic **L**ink **L**ibrary) zur Verfügung (Datei ‘S7.DLL’) und die notwendigen Import-Libraries für den MSC-Compiler 7.0 und MS Visual C++-Compiler 1.0 (Datei ‘S7MSC.LIB’) bzw. den Borland-C-Compiler 3.1 (Datei ‘S7BC.LIB’).

Die Generiervorschriften von SAPI-S7-Applikationen, die auf die DLL-Version von SAPI-S7 aufsetzen, ähneln den Vorschriften für Applikationen, die die SAPI-S7-Libraries verwenden. Die obigen SAPI-S7-Libraries ‘LWS7MSC.LIB’ und ‘LWS7BC.LIB’ sind durch die Import-Libraries ‘S7MSC.LIB‘ und ‘S7BC.LIB‘ zu ersetzen. Darüber hinaus muß beim Compilieren von Quelldateien, die SAPI-S7-Funktionen benutzen, das Define ‘S7_DLL’ gesetzt sein.

Wie Sie diese DLL in anderen Programmiersprachen (z. B. BASIC, Pascal) einsetzen, entnehmen Sie bitte den entsprechenden Handbüchern.

6.4 Übersetzen und Binden für Windows 95 und Windows NT

Voraussetzungen

In den folgenden Kapiteln werden Generierbeispiele aufgelistet, die Ihnen die notwendigen Compiler- und Linker-Optionen für Ihre Applikationen vermitteln sollen.

Die absoluten und auf Ihrem Zielrechner gültigen Laufwerks- und Pfadangaben müssen von Ihnen in die Generieranweisung eingebracht bzw. im Suchpfad aufgenommen werden.

Arbeiten mit dem MSVC-Compiler 2.2

Die S7-Importlibrary für den Microsoft Visual C++-Compiler 2.2 hat unter Windows 95 und Windows NT den Namen 'S7MSC.LIB‘. Die zugehörige DLL hat den Namen 'S732.DLL‘. Das folgende Beispiel zeigt, wie ein Beispielprogramm 'BSP.C‘ für Windows 95 übersetzt und gebunden wird. Für Windows NT ist lediglich das Verzeichnis 'SAPI_S7.W95' durch 'SAPI_S7.NT' zu ersetzen:

```
cl /c /MT /W3 /GX /Zp1 /Od -DSTRICT -DWIN32 -DWINDOWS
   -I\sinec\sapi_s7.w95\include -I\msvc20\include src\bsp.c

link /NODEFAULTLIB /OUT:"bsp.exe" @bsp.dat
```

In der Datei 'BSP.DAT' stehen die Anweisungen für den Linker:

```
bsp.obj,
\sinec\sapi_s7.w95\lib\s7msc.lib
\msvc20\lib\kernel32.lib
\msvc20\lib\user32.lib
\msvc20\lib\gdi32.lib
\msvc20\lib\libc.lib
/SUBSYSTEM:windows /MACHINE:I386
```

6.5 Environmentvariablen

Environment, was ist das?

Beim Environment handelt es sich um einen Speicherbereich, in dem die Parameter abgespeichert sind, die über die MS-DOS-Befehle 'path', 'prompt' und 'set' gesetzt werden. Er besteht aus einer Aneinanderreihung von ASCII-Strings, die durch ein NULL-Zeichen abgeschlossen werden. Er wird verwendet, um Informationen über das gesamte Computersystem festzuhalten.

Umgebungsvariablen der SAPI-S7-Library unter Windows 95 und Windows NT

Die Werte der für die SAPI-S7-Library notwendigen Umgebungsvariablen sind mit dem Konfigurationsprogramm „**PG/PC-Schnittstelle einstellen**“ einzustellen. Dieses Programm setzt die Werte unter dem Schlüssel
„HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\SINEC\SAPI_S7“
in der Registrierdatenbank von Windows 95/NT. Erst wenn dort kein Eintrag mit dem gesuchten Namen vorhanden ist, wird die Variable in der Programmumgebung gesucht.

Steuerung des Trace

Der Trace der S7-Library kann durch insgesamt drei Environmentvariablen gesteuert werden. Mit den Variablen 'S7_TRACE_SELECT', 'S7_TRACE_DEPTH' und 'S7_TRACE_TARGET' können die Dienstklassen, für die Eintragungen im Trace erfolgen sollen, die Trace-Tiefe und das Target des Trace eingestellt werden (siehe Datei ‘SAPI_S7.H‘).

Beispiel: set S7_TRACE_DEPTH=104

Das Vorhandensein und die Auswertung der Umgebungsvariablen erfolgt beim Initialisieren des Trace. Trace-Einstellungen, die schon zuvor getätigt wurden, werden hier überschrieben.

Es wird empfohlen, den Trace mit Hilfe der genannten Environmentvariablen und nicht durch die Mini-DB-Aufrufe einzustellen. Damit können die Defaultwerte zu Debug-Zwecken überschrieben werden, ohne daß die Applikation geändert und neu übersetzt werden muß.

Unter Windows 95 und Windows NT kann der Wert der Variablen auch dadurch festgelegt werden, daß Einträge mit den Namen der Umgebungsvariablen in der Registrierdatenbank unter dem Schlüssel "HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\SINEC\SAPI_S7" abgelegt werden.

Wie erfolgt der Zugriff auf die Projektierung?

In dem Konfigurationsprogramm „PG/PC-Schnittstelle einstellen“ läßt sich die Laufwerksbezeichnung, der Pfad und der Name der Projektierdatei einstellen.

Schritt	Vorgehen
1	Wählen Sie in dem Listenfeld „Benutzte Baugruppenparametrierung“ eine Baugruppenparametrierung aus.
2	Klicken Sie auf die Schaltfläche „Eigenschaften“.
3	Wählen Sie das Register „S7-Protokoll“
4	Aktivieren Sie das Kontrollkästchen „S7 aktivieren“
5	Geben Sie in dem Eingabefeld „SAPI S7-Datenbasis“ die Laufwerksbezeichnung, den Pfad und den Namen der Projektierdatei an.

Unter MS-DOS und Windows 3.x versucht die SAPI-S7-Library den Pfad der Projektierdatei einer Umgebungsvariablen zu entnehmen. Der Namen der Umgebungsvariablen entspricht dem Eintrag in der Registrierdatenbank unter Windows 95 und Windows NT, beim CP ‘CP_L2_1:’ also ‘CP_L2_1:_S7LDB’.

Wird beim Aufruf der Funktion ‘s7_init()’ auch keine entsprechende Umgebungsvariable gefunden, so versucht die SAPI-S7-Library die Projektierdaten aus einer Datei im aktuellen Verzeichnis auszulesen. Der Dateiname wird aus dem Übergabeparameter ‘cp_name’ ermittelt, wobei der Doppelpunkt, mit dem der CP-Name abschließt, entfernt und die Namenserweiterung ‘.LDB’ angehängt wird. So wird, z. B. beim Anmelden des CP ‘CP_L2_1:’, die Datei ‘CP_L2_1.LDB’ ausgelesen.

6.6 Der Trace für MS-DOS oder Windows

Allgemein

Für den Trace der SAPI-S7-Programmierschnittstelle ist eine betriebssystemabhängige Einstellung möglich: per Mini-DB-Aufruf kann das Target des Trace auf den Wert 'S7_TRACE_TARGET_CONSOLE' geändert werden. Das weitere Verhalten ist wie bereits beschrieben betriebssystemabhängig und wird im folgenden näher erläutert.

Die Trace-Einstellung für Windows

Unter Windows wird bei der obigen Trace-Einstellung der gesamte Trace auf dem Bildschirm in einem eigenen Fenster ausgegeben. Das Fenster kann an eigene Bedürfnisse angepaßt werden. Die Konfigurationsmöglichkeiten, wie Anzahl der auszugebenden Trace-Zeilen, sind der Online-Hilfe zu entnehmen.

Die Trace-Einstellung für MS-DOS

Bei der oben genannten Trace-Einstellung werden unter dem Betriebssystem MS-DOS keinerlei Trace-Einträge getätigt. Als Single-User-Betriebssystem ist MS-DOS nicht in der Lage, eine zweite Applikation parallel zum S7-Anwenderprogramm zu starten, die den Trace aufbereitet und auf dem Bildschirm darstellt.

6.7 Besonderheiten für Windows

Unterschied MS-DOS-/Windows-Programme

Windows-Applikationen unterscheiden sich von MS-DOS-Programmen u. a. dadurch, daß sie im Hauptprogramm an einer zentralen Stelle alle Ereignisse empfangen und zur Bearbeitung an eine zuständige Windows-Procedure ('WndProc') weitergeben. Diese Windows-Procedure muß bei Programmstart der Windows-Verwaltung zugänglich gemacht werden.

Beispiel einer typischen Windows-Applikation

```
:
:
#define MY_MSG_ID    1500

WndProc(hWnd,msg,...)
{
        :
        :
        switch(msg)
        {
                case ....:     /* init code */
                        s7_init("CP_L2_1:","MY_VFD",&cp_descr);
                        s7_set_window_handle_msg(
                                cp_descr,hWnd,MY_MSG_ID);
                        break;

                case ....:
                        s7_....(cp_descr,...);
                        break;

                case MY_MSG_ID:
                        s7_receive(cp_descr,&cref,&orderid);
                        break;
        }
}
:
:
```

In einer Windows-Applikation muß nach 's7_init()' die Routine 's7_set_window_handle_msg()' mit einem Window-Handle und einer Message-ID aufgerufen werden, damit das unterlagerte Kommunikationssystem eine Meldung an die Applikation schicken kann. Mit Empfang einer Nachricht wird die Applikation durch eine Message informiert. Die aufgerufene Windows-Procedure ruft dann 's7_receive()' und die entsprechende Bearbeitungsfunktion auf.

6.7.1 s7_set_window_handle_msg

Beschreibung

In einem Windows-Programm muß das Anwenderprogramm nach einem erfolgreichen ‘s7_init()’ die Routine ‘s7_set_window_handle_msg()’ aufrufen. Damit wird dem unterlagerten Kommunikationssystem mitgeteilt, an welches Window und mit welcher ID es seine Nachrichten verschicken soll.

Deklaration

```
int32 s7_set_window_handle_msg(
                ord32     cp_descr,   /* Vorgabe  */
                ord32     hWnd,       /* Vorgabe  */
                ord32     msgID       /* Vorgabe  */
                                      )
```

Parameter

cp_descr	Handle als Rückgabewert des ‘s7_init()’-Aufrufs.
hWnd	Handle des Windows, an das die SIMATIC NET-Message geschickt werden soll.
msgID	ID der SIMATIC NET-Message, die an obiges Window geschickt wird.

Return-Werte

S7_OK	Die Funktion konnte ohne Fehler abgearbeitet werden.
S7_ERR_RETRY	Dieser Wert zeigt einen Fehler bei der Ausführung des angeforderten Dienstes an. Dabei handelt es sich um ein temporär aufgetretenes Problem, wie z. B. um einen vorübergehenden Speicherengpaß. Der Aufruf kann ohne Änderung der Übergabeparameter wiederholt werden.
S7_ERR	Dieser Wert zeigt ebenfalls einen Fehler bei der Ausführung des angeforderten Dienstes an. Allerdings handelt es sich hierbei um einen Fehler, der ein wiederholtes Ausführen des Dienstes nicht erlaubt. Hier müssen weitere Schritte zur Fehlerbehebung, wie das Versorgen des Aufrufs mit anderen Parametern, eingeleitet werden.

Notizen

7 Anhang

Im folgenden Kapitel erfahren Sie,

- ➢ welcher S7-Subset von der SAPI-S7-Library abgedeckt wird,
- ➢ welche Randbedingungen beim Betrieb der SAPI-S7-Library zu beachten sind,
- ➢ wie S7-Variablen im allgemeinen und die Standard-Datentypen von S7 dargestellt werden (sowohl im Host als auch auf dem Netz).

Am Ende des Kapitels erhalten Sie einen Überblick über die gängigsten und in diesem Dokument benutzten Abkürzungen sowie ein Literaturverzeichnis.

7.1 Funktionsumfang von SAPI-S7

SAPI-S7 als Subset von S7

Die SAPI-S7-Programmierschnittstelle bietet den Zugang zu den nachfolgend aufgeführten Diensten (Abkürzung: ‘req’ für Request, ‘ind’ für Indication, ‘con’ für Confirmation und ‘rsp’ für Response):

PICS Serial Number: 1 PICS Part 1 Implementation in the system	
System Parameters	Detail
Implementation’s Vendor Name	- (can be set by COML)
Implementation’s Model Name	VFD-Name (can be set by COML)
Implementation’s Revision Identifier	- (can be set by COML)
Vendor Name of S7	Siemens AG
Controller Type of S7	ASPC2
Hardware Release of S7	A_._ (can be found on type plate)
Software Release of S7	V_._
Profile Number	0
Calling S7 User (enter ‘YES’ or ‘NO’)	YES
Called S7 User (enter ‘YES’ or ‘NO’)	YES

PICS Part 2 Supported Services	
Service	Primitive
Initiate	req, con, ind, rsp
Abort	req, ind
Status	req, con
Unsolicited-Status	ind
Read	req, con
Write	req, con

PICS Part 3	
S7 Parameters and Options	Detail
Addressing by names	YES
Maximum length for names	32
Access-Protection Supported	-
Maximum length for Extension	32
Maximum length for Extension Arguments	0

PICS Part 4	
Local Implementation Values	Detail
Maximum length of S7-PDU	1024
Maximum number of Services Outstanding Calling	-
Maximum number of Services Outstanding Called	-
Syntax and semantics of the Execution Argument	-
Syntax and semantics of Extension	-

Randbedingungen

Beim Betrieb der SAPI-S7-Programmierschnittstelle sind folgende Randbedingungen zu beachten:

➢ Je Applikation können bis zu maximal **150** Aufträge **gleichzeitig** bearbeitet werden.

7.2 Besondere Hinweise

Datenkonsistenz, Nutzdatengröße, zyklisches Lesen

Information zu Datenkonsistenz, Nutzdatengröße und zyklisches Lesen können im Handbuch Kommunikation mit SIMATIC (MLFB 6ES7398-8EA00-8AA0) nachgelesen werden.

7.3 Darstellung von S7-Variablen

Byte-Alignment

Die SAPI-S7-Programmierschnittstelle setzt voraus, daß die Variablen byte-aligned im Hauptspeicher abgelegt werden. Das bedeutet, daß keine Füllbytes (Padding-Bytes), z. B. zwischen den einzelnen Komponenten einer Struktur, eingefügt sein dürfen. Dies wird durch entsprechende Compiler-Optionen bzw. durch Pragmas erreicht.

Netzdarstellung der Variablenwerte

Die SAPI-S7-Library liefert die gelesenen bzw. erwartet die zu schreibenden Variablenwerte in Netzdarstellung. Für zukünftige Versionen der Library ist eine Wandlung zwischen Host- und Netzdarstellung vorgesehen. Deshalb sind alle betroffenen Funktionen um einen Übergabeparameter ‘od_ptr’ erweitert, der in der aktuellen Version der Library mit dem NULL-Pointer belegt werden muß.

Dieser Übergabeparameter ‘od_ptr’ wird in einer späteren Ausgabe der Library die Darstellung der Variablen steuern. Sie ist abhängig davon,

- ob eine Wandlung von Host- in Netzdarstellung oder umgekehrt erfolgen soll (momentan noch nicht Bestandteil der SAPI-S7-Library), oder ob auf dem Host die Netzdarstellung gewünscht wird,
- welche Host-CPU (z. B. Intel) vorhanden ist.

7.4 Darstellung der Standard-Datentypen

7.4.1 Darstellung des Datentyps 'Boolean‘'

Netzdarstellung und Wertebereich

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	0	0	0	0	0	0	0	x

Für 'FALSE' hat 'x' den Wert '0', für 'TRUE' den Wert '1'.

Host-Darstellung und Wertebereich (Intel-CPU)

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	x8	x7	x6	x5	x4	x3	x2	x1

Ist mindestens 1 Bit der Bits ‘x8’ bis ‘x1’ gesetzt, so wird der Wert ‘TRUE’ angenommen. Für 'FALSE' müssen alle Bits ‘0’ sein.

7.4.2 Darstellung des Datentyps 'Integer'

Wertebereich

Beim Datentyp 'Integer' muß nach der Länge des Datentyps unterschieden werden.

Datentyp	Wertebereich	Länge
8-Bit-Integer	-128 ... 127	1 Octet
16-Bit-Integer	- 32768 ... 32767	2 Octets
32-Bit-Integer	-2^{31} ... $2^{31}-1$	4 Octets

Netzdarstellung

Netzdarstellung (2er-Komplementdarstellung) für 8-Bit-Integer ('VZ' stellt das Vorzeichenbit dar und hat für negative Zahlen den Wert '1', sonst den Wert '0'):

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	VZ	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Netzdarstellung (2er-Komplementdarstellung) für 16-Bit-Integer ('VZ' stellt das Vorzeichenbit dar und hat für negative Zahlen den Wert '1', sonst den Wert '0'):

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Netzdarstellung (2er-Komplementdarstellung) für 32-Bit-Integer ('VZ' stellt das Vorzeichenbit dar und hat für negative Zahlen den Wert '1', sonst den Wert '0'):

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	VZ	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Host-Darstellung (Intel-CPU)

Host-Darstellung (2er-Komplementdarstellung) für 8-Bit-Integer ('VZ' stellt das Vorzeichenbit dar und hat für negative Zahlen den Wert '1', sonst den Wert '0'):

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	VZ	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Host-Darstellung (2er-Komplementdarstellung) für 16-Bit-Integer ('VZ' stellt das Vorzeichenbit dar und hat für negative Zahlen den Wert '1', sonst den Wert '0'):

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
2	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}

Host-Darstellung (2er-Komplementdarstellung) für 32-Bit-Integer ('VZ' stellt das Vorzeichenbit dar und hat für negative Zahlen den Wert '1', sonst den Wert '0'):

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
3	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
4	VZ	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}

7.4.3 Darstellung des Datentyps 'Unsigned'

Wertebereich

Beim Datentyp 'Unsigned' muß nach der Länge des Datentyps unterschieden werden.

Datentyp	Wertebereich	Länge
8-Bit-Unsigned	0 ... 255	1 Octet
16-Bit- Unsigned	0 ... 65535	2 Octets
32-Bit- Unsigned	0 ... $2^{32}-1$	4 Octets

Netzdarstellung

Netzdarstellung für 8-Bit-Unsigned:

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Netzdarstellung für 16-Bit-Unsigned:

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Netzdarstellung für 32-Bit-Unsigned:

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Host-Darstellung (Intel-CPU)

Host-Darstellung für 8-Bit-Unsigned:

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Host-Darstellung für 16-Bit-Unsigned:

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8

Host-Darstellung für 32-Bit-Unsigned:

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
3	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
4	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}

7.4.4 Darstellung des Datentyps 'Floating Point'

Wertebereich

Wertebereich	Länge
$-3.37*10^{38}$ ... $-8.43*10^{-37}$	4 Octets
0	
$8.43*10^{-37}$ ... $3.37*10^{38}$	

Netzdarstellung

Netzdarstellung ('VZ' ist das Vorzeichen der Mantisse, die schattierten Felder gehören zum Exponenten, die restlichen zur Mantisse):

Octet \ Bit	MSB 8	7	6	5	4	3	2	LSB 1
1	VZ	2^7	2^6	2^5	2^4	2^3	2^2	2^1
2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

Host-Darstellung (Intel-CPU)

Host-Darstellung ('VZ' ist das Vorzeichen der Mantisse, die schattierten Felder gehören zum Exponenten, die restlichen zur Mantisse):

Octet \ Bit	MSB 8	7	6	5	4	3	2	LSB 1
1	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}
2	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
3	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
4	VZ	2^7	2^6	2^5	2^4	2^3	2^2	2^1

Werteberechnung Der Variablenwert wird wie folgt berechnet:

Exponent	Variablenwert
0	$(-1)^{VZ}*((0.\text{Mantisse})*2^{-126})$
ungleich 0	$(-1)^{VZ}*((1.\text{Mantisse})*2^{(\text{Exponent}-127)})$

Beispiel für die Darstellung des Variablenwertes '0,5' im Host als ‘C’-Datentyp ‘float’:

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	1	1	1	1	1	1

7.4.5 Darstellung des Datentyps 'Visible-String'

Wertebereich

Als Zeichen für Variablen vom Typ 'Visible-String' sind alle Buchstaben (Klein- und Großschreibung), Ziffern, der Underscore ('_') und das Dollarzeichen ('$') zugelassen.

Netzdarstellung

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	erstes Zeichen							
2	zweites Zeichen							
..								
n	n-tes Zeichen							

Host-Darstellung (Intel-CPU)

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	erstes Zeichen							
2	zweites Zeichen							
..								
n	n-tes Zeichen							
n+1	NULL-Terminierung							

Eine Variable vom Typ 'Visible-String' entspricht im Host einem String der Programmiersprache 'C', d. h. er ist NULL-terminiert.

7.4.6 Darstellung des Datentyps 'Octet-String'

Netzdarstellung

Eine Variable vom Typ 'Octet-String' wird auf dem Netz wie ein 'Visible-String' dargestellt. Allerdings sind hier sämtliche Bytewerte zulässig.

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	erstes Zeichen							
2	zweites Zeichen							
..								
n	n-tes Zeichen							

Host-Darstellung (Intel-CPU)

Bit / Octet	MSB 8	7	6	5	4	3	2	LSB 1
1	Längeninformation							
2	erstes Zeichen							
..								
n	(n-1)tes Zeichen							
n+1	n-tes Zeichen							

Die Host-Darstellung unterscheidet sich von der Netzdarstellung insofern, daß im ersten Byte die Längeninformation abgelegt ist.

7.4.7 Darstellung des Datentyps 'Bit-String'

Netzdarstellung

Eine Variable vom Typ 'Bit-String' wird auf dem Netz wie folgt dargestellt.

Bit / Octet	MSB							LSB
	8	7	6	5	4	3	2	1
1	7	6	5	4	3	2	1	0
2	15	14	13	12	11	10	9	8
..								

Host-Darstellung (Intel-CPU)

Bit / Octet	MSB							LSB
	8	7	6	5	4	3	2	1
1	Längeninformation							
2	7	6	5	4	3	2	1	0
3	15	14	13	12	11	10	9	8
..								

Die Host-Darstellung unterscheidet sich von der Netzdarstellung insofern, daß im ersten Byte die Längeninformation abgelegt ist.

Glossar

ASCII	**A**merican **S**tandard **C**ode of **I**nformation **I**nterchange - Kodierungsvorschrift für ein Byte große Zeichen.
CP	**C**ommunication **P**rocessor - Kommunikationsbaugruppe für den Einbau in Rechner oder Automatisierungsgeräte.
CREF	**C**onnection **Ref**erence - Verbindungskennung.
default	Standardeinstellung des Auslieferungszustandes von Siemens.
IEC	**I**nternational **E**lectronical **C**ommission - Internationales Normungskomitee.
Kommunikationssystem	Unterlagerte Software und Hardware zur Anbindung an das Kommunikationsnetz.
Multi-CP-Betrieb	mehrere CPs können gleichzeitig in einem PG/PC betrieben werden
Multi-User-Betrieb	mehrere Applikationen auf einem PG/PC
PDU	**P**rotocol **D**ata **U**nit (Protokolldateneinheit) - Nachricht einer ISO/OSI-Schicht.
PG	**P**rogrammier**g**erät.
PROFIBUS	**Pro**cess **Fi**eld **Bus** - Netz für den Zell- und Feldbereich des mittleren Leistungsbereichs mit dem Einsatzschwerpunkt in industrieller Umgebung nach EN 50 170, Volume 2. PROFIBUS.
SAPI	**S**imple **A**pplication **P**rogrammers **I**nterface - einfache Programmierschnittstelle für Kommunikationsprotokolle auf dem PG/PC.
SIMATIC NET	Produktlinie für industrielle Kommunikation von Siemens.
SINEC L2	-> PROFIBUS
S7	SIMATIC S7 ist ein Automatisierungssystem der Siemens AG.
VFD	**V**irtual **F**ield **D**evice - Abbildung eines realen Automatisierungsgerätes mit der Zielsetzung, eine einheitliche Sicht auf ein beliebiges Gerät zu erhalten.

Index