DRIVECOM

DriveServer

Version 1.1, 19.Oktober.01 09:42

Herausgeber: DRIVECOM Nutzergruppe e.V.

Postfach 1102, D-32817 Blomberg

Telefon : 0 52 35 / 3-4 18 64

Fax : 0 52 35 / 3-4 18 62

Internet : http://www.DRIVECOM.org

Verfasser:

Albach	Lust	Lahnau
Arlt	Lenze	Hameln
Hadlich	ifak system GmbH	Magdeburg
Iwanitz	Softing GmbH	München
Krumsiek	Phoenix Contact	Blomberg
Leurs	Rexroth Indramat	Lohr am Main
Mirbach	Lenze	Hameln
Müller	Phoenix Contact	Blomberg
Pollmeier	ESR Pollmeier	Ober-Ramstadt
Riedl	ifak Magdeburg e.V.	Magdeburg
Schleicher	Indramat-Refu	Metzingen
Schnurbusch	Lenze	Hameln
Ziegler	SEW-EURODRIVE	Bruchsal

Anmerkungen und Kommentare sind an Herrn Stefan Pollmeier gl@esr-pollmeier.de zu richten.

Änderungshistorie:

Version	Name	Firma	Kommentar
0.0.22	Riedl	ifak	Erstellung
0.1	Mirbach	Lenze	- dreistufige Adressierung (Busserver) - Identifikation des Bussystems durch den DriveServer - Verschiedene Netze bzw. Stränge an einem Busserver
1.0	Mirbach	Lenze	- Klärung des bitweisen Zugriffs an der DriveServer-Schnittstelle zum OPC-Client - Streichung des bitweisen Zugriffs an der Busserver-Schnittstelle
1.0a	Mirbach	Lenze	Ergänzung des Kapitels 5 um den Typ VT_ARRAY
1.1	Mirbach	Lenze	Ergänzung der Kapitel 3.2.1.1 und 3.3.2.1 um die Belange des Profidrive-Parameterkanals. Bei nicht indizierten Parametern darf der Subindex nicht angegeben werden. Aufgrund der nun hergestellten Kompatibilität zum Profibus entsteht in diesem Punkt eine Inkompatibilität zwischen den Versionen 1.0 und 1.1.

Inhalt

1 Einleitung

1.1 Übersicht

Intelligente modulare Systeme, die Komponenten aus Mechanik, Elektronik und Software sowie Sensorik und Aktorik enthalten, bestimmen heute die Produktionsstrategien. Moderne Antriebsregler führen beispielsweise eigenständig umfangreiche technologische Regelungen durch und übernehmen Aufgaben von Steuerungen. Mit standardisierten Feldbussystemen wie Profibus können diese modularen Maschinenkonzepte einfach realisiert werden.

Die Problematik. Die Darstellung eines Antriebs aus Sicht eines Anwenders unterscheidet sich fundamental von der Darstellung eines Antriebs aus Kommunikationssicht. Ein Anwender arbeitet mit den Parametern des Gerätes, normalerweise bezeichnet durch Namen. Aus Kommunikationssicht werden Parameter durch Zahlenpaare (z. B. Index, Subindex; Slot, Index) bezeichnet. Viele Anwender akzeptieren und nutzen diese Darstellungsweise, jedoch mit der Einschränkung, daß die Kommunikationssicht in der Regel an das verwendete Kommunikationsprotokoll bzw. -profil gebunden ist. D. h. je nach Protokoll kommt es zu Varianz. Diese Situation resultiert in einem erheblichen Engineeringaufwand. Ein schneller geräteorientierter Datenzugriff würde die Engineeringleistungen verbessern und damit die Kosten mindern.

Deshalb hat es sich die DRIVECOM-Nutzergruppe zur Aufgabe gemacht, die Kommunikations-schnittstellen für den Zugriff auf Antriebe zu standardisieren. Basierend auf dem Schnittstellenstandard OPC in der Version 2.0 wird die Spezifikation für einen **DriveServer** erarbeitet. Das innovative Konzept sieht eine Vereinheitlichung der Präsentation und des Zugriffs auf Geräte und deren Funktionen vor.

Die Verwendung der OPC-Schnittstelle in der Prozessautomation hat dazu geführt, daß die Anpassungen an spezifische Geräte und Hardware nicht mehr in jeder Applikation realisiert werden muß, sondern von den Herstellern von Automatisierungsgeräten selbst durchgeführt werden kann. Da der Zugriff auf die Prozessdaten vereinheitlicht und die softwaretechnische Anbindung auf dem PC durch den Einsatz von OLE festgelegt wurde, entfallen Anpassungen an die herstellerspezifischen Schnittstellen. Dieses war bisher im wesentlichen auf die Kommunikationsnetzwerke beschränkt. Die Spezifikation eines DriveServers nutzt diesen Ansatz für allgemeine Automatisierungsgeräte.

Einbindung in die OPC-Architektur. Der DriveServer basiert auf der OPC-Technologie. Er bildet eine Schicht zwischen einem Anwendungsprogramm mit OPC Client-Schnittstelle und den Kommunikationsmedien. Die Anbindung an die Kommunikationsmedien kann entweder herstellerspe-zifisch im DriveServer implementiert werden oder erfolgt über einen Kommunikations-OPC-Server.

Der Ansatz des DriveServers kapselt die Geräteeigenschaften funktional. Alle Geräteeigenschaften, deren Repräsentation und deren Zugriff erfolgt über eine funktionale Schnittstelle. Dadurch bleibt das Know-how der Geräte, deren Beschreibungsform und Zugriffsmechanismen gekapselt. Der Zugriff erfolgt mit Standardmitteln. So werden keine Parameter und deren Inhalt, sondern deren Zugriff in einem Namensbildungsverfahren festgelegt. Dieses Vorgehen hat den wesentlichen Vorteil, daß die Gerätehersteller zunächst und auf Dauer ihre eigenen Implementierungen, die sich in den Geräten wiederfinden, beibehalten können.

Um die Vielzahl der verfügbaren und zukünftigen Kommunikationsmedien für eine Kommmunikation zum Antrieb nutzen zu können, basiert die DriveServer-Spezifikation auf einer deutlichen Trennung zwischen Kommunikationsfunktionalität und DriveServer-Funktionalität. Deswegen kommuniziert der DriveServer über ein OPC-Interface. Diese offene Architektur des Drive-Servers gewährleistet die gewünschte Flexibilität der Feldbussysteme und der verwendeten Antriebskomponenten – ein weiterer entscheidener Schritt hin zu offenen und durchgängigen Automatisierungslösungen.

Die hier vorgestellte OPC-basierte Integration von Antrieben in Engineeringsysteme soll einen Maßstab in der Welt der Feldbuskommunikation setzen. Seine im wesentlichen geräteunabhängige

Architektur öffnet dem DriveServer sofort ein breites Einsatzfeld. Setzt sich das DriveServer-Konzept auch für andere Feldbuskomponenten durch, wird es für allgemeine Device-Server die Basis bilden.

Dieses Dokument dient dazu, eine einheitliche Anwendungsschnittstelle (Application Interface) zum Zugriff auf Antriebe zu beschreiben. Ziel der Spezifikationsarbeiten soll sein, durch eine Darstellung der Antriebsvariablen bzw. der firmenspezifischen Funktionen im Namensraum der OPC-Server ein Plug and Play zu ermöglichen, analog der Treibersoftware eines Druckers.

2 Architektur

2.1 Übersicht

Die DriveServer Architektur bewegt sich vollständig im Rahmen der OPC-Spezifikation, um die Kompatibilität zu anderen Servern zu erhalten. Die beim DriveServer vorgenommen Spezifikationen können von OPC-Servern implementiert werden, sie sind jedoch auch vollständig mit Standard OPC-Mitteln einhaltbar. Der DriveServer ist zunächst selbst Client eines Kommunikationsservers, im folgenden Busserver genannt, da die Kommunikationsschnittstellen typischerweise zu einem Feldbus bestehen. Der Busserver kann jedoch grundsätzlich Server eines beliebigen Übertragungsmediums sein.

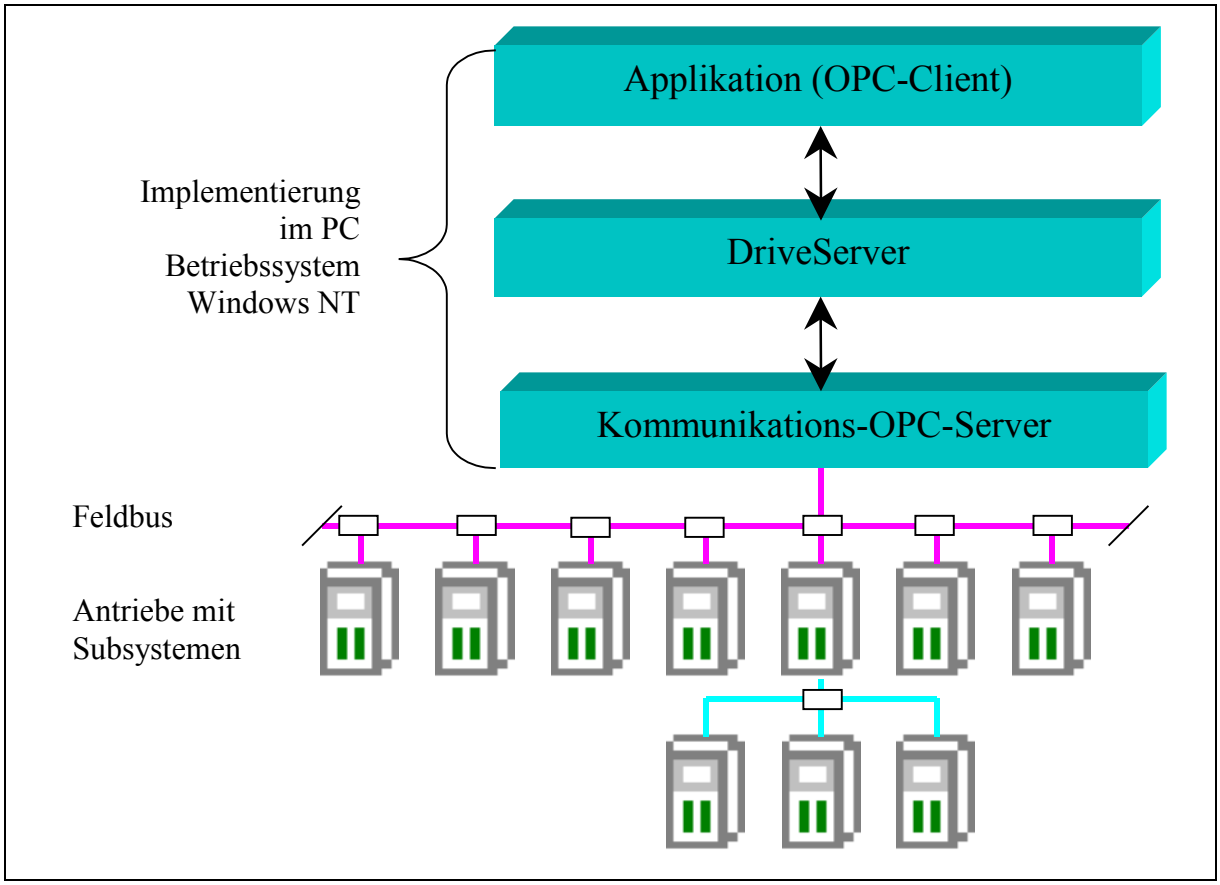

Abbildung 1: Architektur der OPC-Server

2.2 Anordnung der Komponenten

Es ist möglich, die in Abbildung 1 dargestellten SW-Komponenten auf einem einzelnen Rechner oder auf mehreren miteinander verbundenen Rechnern (Rechnerverbund) zu installieren. Dazu müssen die Server die in den OPC- und (D)COM-Spezifikationen festgelegten Regeln zur Registrierung auf einem Rechner bzw. Rechnerverbund einhalten. In dieser Spezifikation wird auf die verteilte Installation nicht eingegangen, da diese durch DCOM-Standardmechanismen definiert ist.

Ein sehr gebräuchlicher Einsatzfall ist der Zugriff eines DriveServers auf ein oder mehrere Busserver. Mit welchen Busservern der DriveServer in Verbindung tritt, kann konfiguriert werden. Dabei ist es dem Hersteller des DriveServers freigestellt, auf welche Art der Zugriff auf Busserver konfiguriert werden kann.

Der DriveServer hat sich beim Busserver mittels der Funktion `IOPCCommon::SetClientName` entsprechend der der OPC-Spezifikation anzumelden (z.B. „`DriveServer_XY.EXE`“ oder „`\\192.168.10.2\DriveServer_XY.EXE`“). Der Busserver kann damit ggf. Rückschlüsse auf den Client ziehen.

Als OPC-Trennzeichen wird der Punkt („.“) definiert. Dieses Zeichen darf sich folglich nicht in den Namen irgendwelcher Parameter befinden.

2.3 Kommunikationsstruktur

Der DriveServer bildet die gerätebezogenen Zugriffe der Applikation auf feldbusbezogene Zugriffe ab. DriveServer und Kommunikations-OPC-Server nehmen laut OPC-Spezifikation den Zugriff über den Namen vor. Diese Namen können je nach Server-Implementierung vorkonfiguriert oder dynamisch angelegt werden. Die Kommunikation über Namen und die Abbildung der Bus- und Gerätearchitektur wird in Abbildung 2 erläutert.

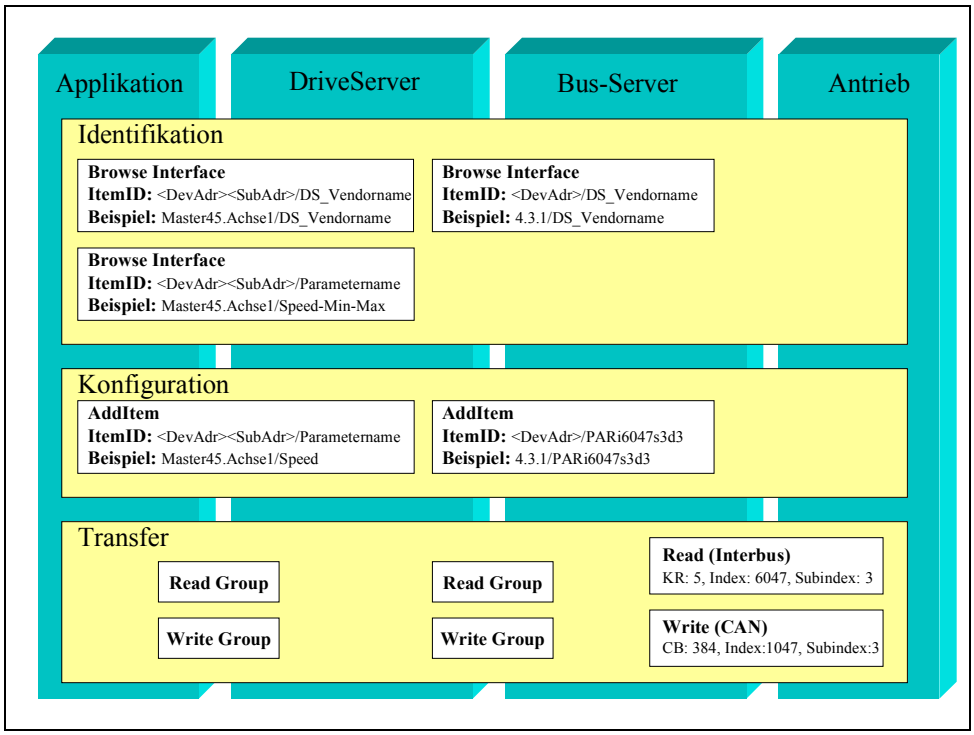

Abbildung 2 : Kommunikationsstruktur

An der Schnittstelle zwischen Applikation, DriveServer und Busserver sind drei Phasen zu betrachten:

- Die Identifikation der Geräte

  Der DriveServer fragt über das Browse Interface des Busservers nach verfügbaren Geräten am Bus. Diese Abfragen erfolgen einheitlich nach den unter 3.3.2 festgelegten Regeln für den Namensraum. Über einen herstellerspezifischen Identifikationsalgorithmus, der im DriveServer implementiert ist, werden die Gerätetypen bestimmt und im Browse Interface des DriveServers zur Anzeige gebracht. Eine Applikation kann mit Hilfe dieser Informationen die Geräte auswählen.

- Die Definition von Items als Zugriffsspezifikation (Konfiguration)

  Nachdem die Applikation weiß, mit welchem Gerät sie zusammen arbeiten möchte, können im DriveServer eine oder mehrere OPC-Gruppen angelegt werden. In diesen OPC-Gruppen werden OPC-Items angelegt, die den Geräteparametern entsprechen. Der DriveServer kann anhand der

vorherigen Indentifikation der Geräte seinen eigenen Namensraum (anhand einer Gerätebeschreibung) aufbauen. Dies versetzt ihn in die Lage, eventuell nicht auflösbare Geräteparameteranfragen abzuweisen. Fehlerhafte Anfragen können so zuverlässig vermieden werden.

In der Regel werden die im DriveServer angelegten OPC-Gruppen und OPC-Items in den betreffenden Busservern ebenfalls angelegt. Die OPC-Itemnamen entsprechen hierbei jedoch den Bus- und geräteinternen Adressen. Kann solch eine OPC-Gruppe oder OPC-Item im Busserver nicht angelegt werden, muß auch der DriveServer diese Objekte der Applikation als nicht erzeugbar melden.

- Die I/O-Operationen (Transfer)

  Nach der Konfiguration der Server kann der eigentliche Datentransfer von der Applikation ausgelöst werden. Hierbei werden auf den Gruppen des DriveServers Lese- bzw. Schreibbefehle ausgeführt, wodurch alle in der OPC-Gruppe enthaltenen OPC-Items kommuniziert werden. Dies bedeutet, daß diese Befehle an die unterliegenden Busserver weitergeleitet werden und diese die Parameterwerte mit den Geräten austauschen.

2.4 DriveServer Architektur

Der DriveServer stellt im Sinne der OPC-Definition eine OPC-Serverschnittstelle und eine OPC-Client-Schnittstelle zur Verfügung. Er besteht aus drei Hauptkomponenten:

- Die Komponente, die die OPC-Server-Schnittstelle implementiert.
- Eine Komponente, die die DRIVECOM-Eigenschaften realisiert und somit als Schnittstelle für alle Antriebsgeräte gleich ist, jedoch unterschiedliche Implementierungen zuläßt.
- Die Komponente, die die herstellerspezifischen Eigenschaften der Geräte kapselt.

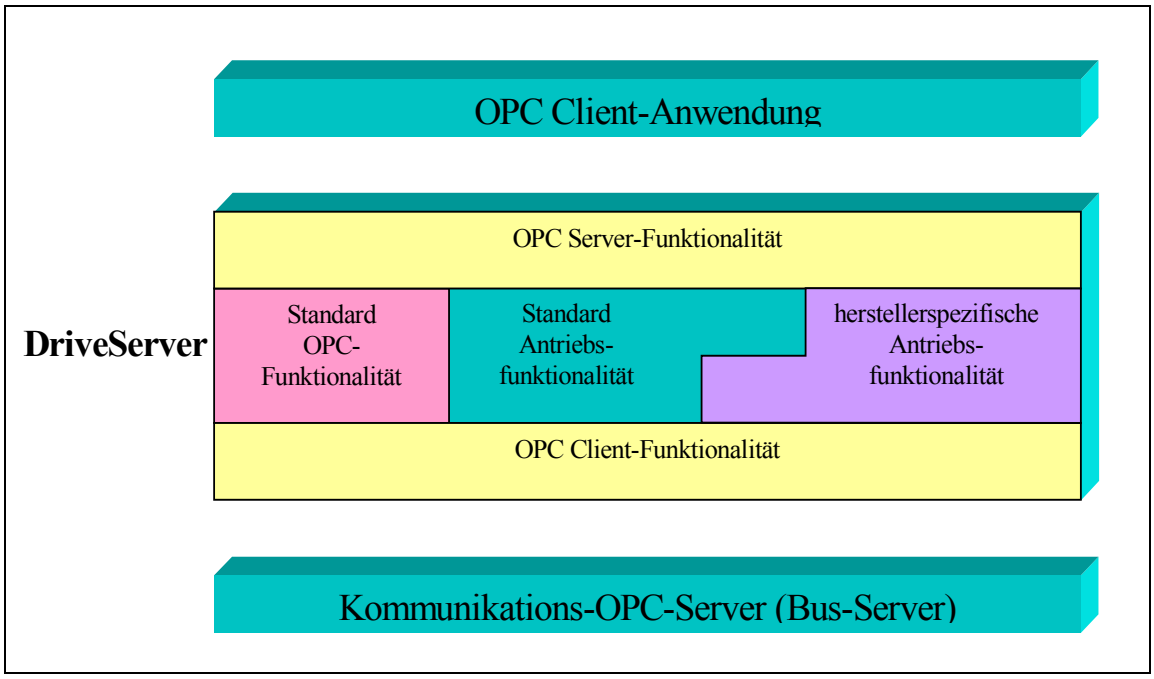

Abbildung 3: DriveServer Architektur

2.5 Busserver-Architektur

Der Busserver übernimmt in der vorgestellten Architektur die Kommunikation mit den Geräten. Er ist in der Regel ein busspezifischer OPC-Server, d.h. die Umsetzung der Lese- und Schreibanforderungen ist im Rahmen dieser Spezifikation nicht festzulegen.

Auch muß der Busserver nicht direkt auf einen Bus zugreifen. Z. B. ist ein OPC Server denkbar, der auf TCP/IP aufsetzt und nur bestimmte Sockets des Betriebssystems nutzt oder der mit anderen (D)COM Komponenten kommuniziert, die Geräte simulieren.

Die Anforderungen an den Busserver beschränken sich auf die Standard-OPC-Serverfunktionalität. Das Browse-Interface muß jedoch implementiert sein. Auch sollte der Busserver dynamisch konfigurierbar sein, d.h. zur Laufzeit sollen OPC-Items hinzugefügt werden können, die nicht zur Entwurfszeit konfiguriert worden sind. Zusätzlich müssen die Forderungen nach 3.3 erfüllt werden.

3 Funktionalität

3.1 Übersicht

Die OPC-Spezifikation “Data Access 2.0” definiert den Aufbau und das Verhalten von Interfaces. Bewußt nicht definiert sind dort alle Belange des Namesraumes eines Servers (Struktur, ItemIds usw.). Daraus ergibt sich ein weiter Spielraum, der für Entwickler Platz für Kreativität und für den Nutzer eine gewisse Unsicherheit bedeutet.

Genau dieser Spielraum, beim Aufbau des Namesraumes und bei der Semantik von ItemIds, soll in der vorliegenden DriveServer-Spezifikation in gewisser Weise eingeengt werden. Auf diese Weise werden Automatismen ermöglicht und dem Nutzer ein einfacherer Umgang (Wiedererkennungswert) gewährleistet.

3.2 DriveServer Funktionalität

Ein Driveserver bietet jede notwendige Funktionalität, um auf einen Antrieb zuzugreifen und ihn zu handhaben.

3.2.1 Übersicht

Die Identifikation der Geräte kann automatisiert über das Browse-Interface des DriveServers erfolgen. Zu jedem Gerät gibt es genau ein identifizierendes <keyword_tag>. Dieses <keyword_tag> wird in dieser Spezifikation mit ‘DS_Vendorname‘ vorgeschlagen, es kann jedoch auch ein anderer String festgelegt werden. Der Client des DriveServers greift auf den nach OPC-Spezifikation flat formatierten (siehe S.10) Namensraum des Driveservers zu und filtert an Hand des identifizierenden <keyword_tag> den Namensraum nach möglichen Geräteadressen. Per Definition legt der vollständige Text des gefilterten Namens, ohne den Kennstring (‚DS_Vendorname‘) die Geräteadresse fest und wird im folgenden <device_tag> genannt. Die Syntax dieses Textes ist serverspezifisch. Da dieser Text einen eindeutigen Zugriffspfad darstellt, wird er von den Clients anstelle einer Geräteadresse verwendet. Der Client benötigt damit die eigentliche busspezifische Geräteadresse zum Zugriff auf die Geräte nicht.

Der DriveServer stellt in einem OPC-Namensraum die von ihm erkannten Antriebe dar. Dieser Namensraum ist hierarchisch aufgebaut, d.h. ein Namensraum ist ähnlich der Dateistruktur in einem Rechner (Verzeichnisse/Dateien) gegliedert (Branches/Leafs). Ein Branch entspricht einem Gliederungspunkt (Verzeichnis) und ein Leaf entspricht einem Datenpunkt im OPC Server (Datei).

Dieser Namensraum kann über das `IOPCBrowseServerAddressSpace`-Interface abgefragt werden. Dabei kann der Client sich durch die verschiedenen Gliederungsebenen des Namensraums bewegen (ähnlich wie bei einem Verzeichniswechsel) und die Leafs einzelner Gliederungspunkte abfragen. Es besteht auch die Möglichkeit, den Namensraum (vollständig oder teilweise) flat formatiert abzufragen. Dabei werden die Leaf-Namen gemeinsam mit den Namen der übergeordneten Gliederungspunkte dargestellt (ähnlich den absoluten Dateinamen) .

Beispiel: Es gibt 2 Geräte am Bus, die auf unterschiedlichen Adressen liegen.
Hierarchischer Namensraum:

```
Bus Server Name
      Adresse 1
            DS_Vendorname
            DS_Devicename
            DS_DeviceID
            ...
```

```
        Adresse 2
              DS_Vendorname
              DS_Devicename
              DS_DeviceID
              ...
```

flat formatiert:

```
Bus Server Name.Adresse 1.DS_Vendorname
Bus Server Name.Adresse 1.DS_Devicename
Bus Server Name.Adresse 1.DS_DeviceID
...
Bus Server Name.Adresse 2.DS_Vendorname
Bus Server Name.Adresse 2.DS_Devicename
Bus Server Name.Adresse 2.DS_DeviceID
...
```

Werden durch den DriveServer mehrere Busserver angesprochen, so kann im Namensraum des DriveServers eine entsprechende Gliederungsebene eingeführt werden. In dieser Ebene entspricht jeder Gliederungspunkt einem Busserver. Den jeweiligen Bus-Server-Gliederungspunkten können die Antriebe, zu denen die einzelnen Busserver eine Kommunikationsbeziehung aufbauen, zugeordnet werden. Der beim Browsen angezeigte String für die Busserver ist frei wählbar und kann (implementierungsabhängig) konfiguriert werden. Die Bezeichnung der Hierarchiestufe für die Geräteadresse wird durch Konkatenation des Wertes des OPC-Items `DS_BusPort`, „-“ und des Wertes des OPC-Items `DS_DeviceID` des Busservers gewonnen.

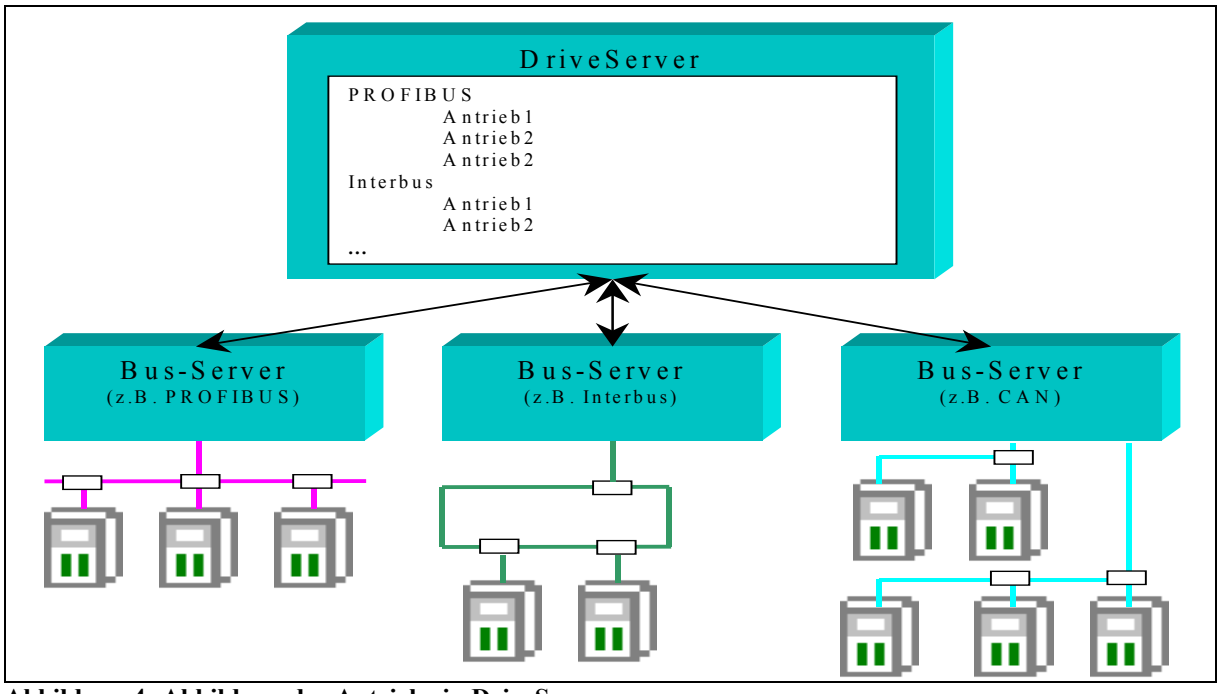

Abbildung 4: Abbildung der Antriebe in DriveServer

In der Hierarchie des Namensraumes spiegelt sich die Struktur der Anlage teilweise wider. Slaveantriebe sind hierarchisch unter ihren jeweiligen Masterantrieben angeordnet.

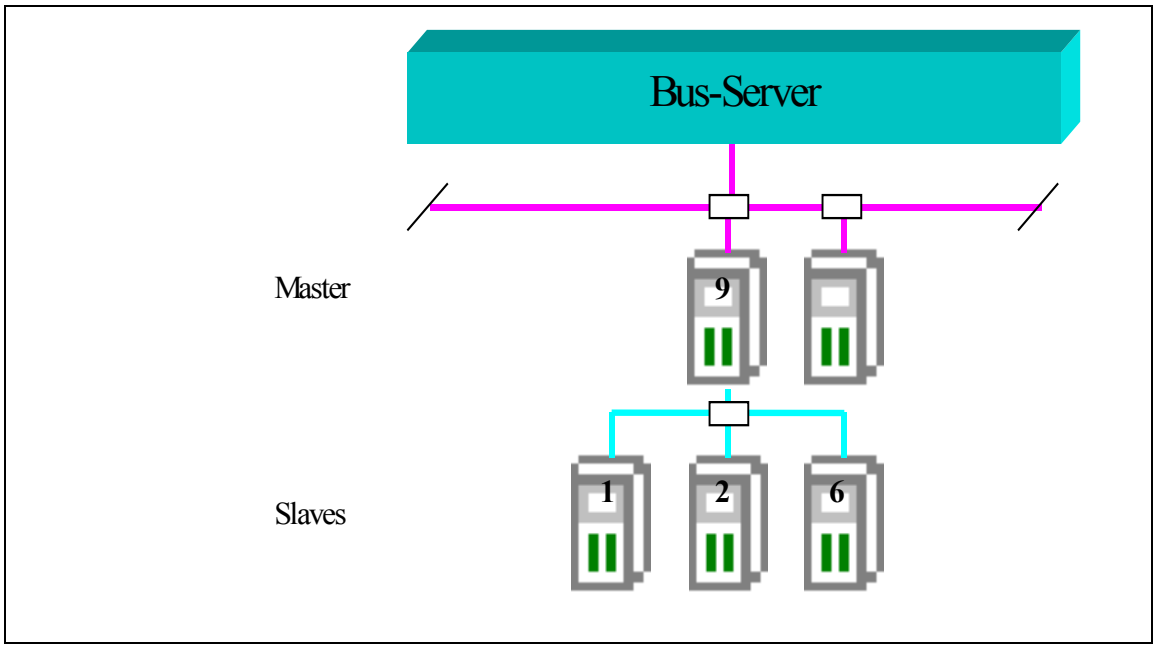

Abbildung 5: Antrieb mit Subsystemen

Beispiel für den Namensraum eines Antriebs mit Subsystemen:

```
Bus Server Name
      9
                  DS_Vendorname
                  DS_Devicename
                  DS_DeviceID
                  ...
                  1
                        DS_Vendorname
                        DS_Devicename
                        DS_DeviceID
                        ...
                  2
                        DS_Vendorname
                        DS_Devicename
                        DS_DeviceID
                        ...
```

Die Initialisierung des Namensraumes ist implementierungsspezifisch (herstellerspezifisch) und wird auch als Konfiguration des Servers bezeichnet. Zwischen zwei Arten der Konfiguration kann man unterscheiden:

- Statische Konfiguration: Sie kann z.B. durch Einlesen einer Anlagenkonfiguration bzw. von Gerätebeschreibungen beim Start des Servers erfolgen. Es werden in der Regel beliebige symbolische Namen (nach DriveServer-Spezifikation oder herstellerspezifisch) vergeben.

- Dynamische Konfiguration: z.B. durch ItemIds mit einer definierten Syntax.

Jedes Gerät ist im Namensraum über ein <keyword_tag> eindeutig zu identifizieren. Alle Parameter eines Gerätes sind als Blätter eines gerätespezifischen Astes auffindbar. D.h. im flat formatierten Namensraum ergibt sich folgende Struktur für alle Parameter:

```
<item_tag>          := <device_tag><parameter_tag>
```

Das <device_tag> wird vom DriveServer selbst vergeben und enthält alle Zeichen und Strings, die der DriveServer zur Identifikation eines Gerätes benötigt, inklusive einer möglichen Trennung zwischen gerätespezifischer Zeichenkette und dem <parameter_tag>.

Der String <device_tag> kann direkt einer Parameteradresse vorangestellt werden, um einen Parameter eines Gerätes zu ermitteln.

3.2.1.1 Reservierte Namen (DriveServerspezifische Namen)

Diese Spezifikation reserviert spezifische Namen (Schreibweise beachten!) für bestimmte Standardfunktionalität.

Jedes Gerät verfügt über Standardparameter, welche zur Identifizierung eines Gerätes durch den Client benutzt werden können. Der Datentyp der folgenden Parameter ist VARIANT. Das darin enthaltene Datum mit seinen Datentyp kann durch den Hersteller frei belegt werden, da jeder OPC-Client in der Lage ist, den Datentyp entsprechend zu interpretieren und ggf. umzuwandeln.

Parameter	Bedeutung	m/o
DS_Devicename	Gerätename	M
DS_DeviceID	Antriebsreglerkennung (Busaddresse, URL, ...)	M
DS_Vendorname	Herstellername	M
DS_Devicetype	Gerätetyp	O
DS_Vendorcode	Herstellercode	O
DS_ProfileID	Profilkennung	O
DS_BusSystem	Kennung des Feldbussystems in Form einer Component Category (Registry Format), Definition in Kapitel 4	O
DS_BusPort	Busnetz bzw. –strang, Definition in Kapitel 3.3.2.1	O

Beispiele :

```
Bus Server Name.9.DS_ProfileID (DS_DeviceID = 9)
Bus Server Name.127-168-0-2.DS_ProfileID (DS_DeviceID = 127-168-0-2)
Bus Server Name.COM1-9.DS_ProfileID (DS_BusPort = COM1, DS_DeviceID = 9)
```

Namenskonvention

Die einzelnen Parameter eines Antriebs können durch eine definierte Syntax angesprochen werden. Für alle Parameter, auch die mit symbolischem Namen, wird zusätzlich folgende Namenskonvention definiert.

Der optionale bitweise Zugriff auf Prozeß- oder Parameterdaten kann vom DriveServer auf bestimmte Parameter beschränkt werden.

```
<parameter_tag>      := <parameter_channel> | <process_channel>
<parameter_channel>  := "PAR"
                        <index_info>[<subindex_info>][<datatype_info>]
                        [<bit_info>][<length_info>]
                        [<extra_info>]
<process_channel>    := <process_cannel_info>
                        <index_info><subindex_info>[<bit_info>]
                        [<datatype_info><length_info>[<extra_info>]

<process_channel_info> :=    "OUT" | "IN"
<index_info>         := (I|i)[0-9]+
<subindex_info>      := (S|s)[0-9]+
<bit_info>           := (B|b)[0-9]+
<datatype_info>      := (D|d)(VT_UI1 | VT_ARRAY | ... )
                        (Die einzusetzenden Zahlenwerte sind in Kapitel 5 aufgeführt)
<length_info>        := (L|l)[0-9]+
<extra_info>         := (X|x)<String>
```

`index_info:` Index des Objekts (beginnt bei „0“)

`subindex_info:` Beim Prozeßdatenkanal Zugriff auf ein Byte, falls das Objekt aus mehreren Bytes besteht, ‚0‘ bedeutet, daß auf alle Bytes des Objekts zugegriffen wird.

Beim Parameterdatenkanal gibt der Subindex die Position in indizierten Feldern an. Handelt es sich bei dem angesprochenen Objekt nicht um einen indizierten Parameter, so wird kein Subindex angegeben.

`bit_info:` Bitposition im Objekt, gezählt beginnend vom LSB als Bit Nr. 0; ist die Bitposition angegeben, muß auch die Länge definiert sein.

`length_info:` Länge der geforderten Daten in Bits

Beispiele :

`PARI1000S0:` Indizierte Parametervariable mit Index 1000 und Subindex 0. Der Datentyp ist dem DriveServer aufgrund der Gerätebeschreibung bekannt und muß daher nicht angegeben werden.

`pari120d3 :` Nicht-indizierte Parametervariable mit Index 120, Datentyp 3 (entspricht `VT_I4`: 4-Byte-Integer)

`pari10s40d8 :` Parametervariable mit Index 10 und Subindex 40, Datentyp 8 (entspricht `VT_BSTR`: Zeichenkette)

`INI2S0D3:` IN-Prozeßvariable auf Prozeßdatenkanal 2. Die gesamte Prozeßvariable ist als 4-Byte-Integer `(VT_I4)` zu interpretieren.

`INI2S0D3L16:` IN-Prozeßvariable auf Prozeßdatenkanal 2. Die unteren 16-Bits der Variablen sollen auf eine 4-Byte-Integer-Variable abgebildet werden.

Annahme: die Breite des Prozeßdatenkanals beträgt 4 Byte.

Quelle: `MSB XXXXXXXX XXXXXXXX PONMLKJI HGFEDCBA LSB`

Ziel: `MSB XXXXXXXX XXXXXXXX PONMLKJI HGFEDCBA LSB`

`INI1S2D3B2L4 :` IN-Prozeßvariable auf Prozeßdatenkanal 1. Byte 2 soll beginnend mit Bit 2 und einer Länge von 4 Bit auf eine 4-Byte-Integer-Variable `(VT_I4)` abgebildet werden.

Annahme: die Breite des Prozeßdatenkanals beträgt 4 Byte.

Quelle: `MSB XXXXXXXX XXXXXXXX XXDCBAXX XXXXXXXX LSB`

Ziel: `MSB XXXXXXXX XXXXXXXX XXXXXXXX XXXXDCBA LSB`

`INI1S0D3B10L4 :` IN-Prozeßvariable auf Prozeßdatenkanal 1. Die Bits 10, 11, 12, 13 der Prozeßdaten sollen auf eine 4-Byte-Integer-Variable (`VT_I4`) abgebildet werden.

Annahme: die Breite des Prozeßdatenkanals beträgt 4 Byte.

Quelle: `MSB XXXXXXXX XXXXXXXX XXDCBAXX XXXXXXXX LSB`

Ziel: `MSB XXXXXXXX XXXXXXXX XXXXXXXX XXXXDCBA LSB`

`PARI150S0D3B1L4:` Parametervariable mit Index 150 und Subindex 0. Beginnend mit Bit 1 sollen 4 Bits auf eine 4-Byte-Integer-Variable (`VT_I4`) abgebildet werden.

Quelle: `MSB - XXXXXXXX XXXDCBAX - LSB`

Ziel: `MSB XXXXXXXX XXXXXXXX XXXXXXXX XXXXDCBA LSB`

mit vollständigem Pfad:

```
Bus Servername.9.pari10s40d3
Bus Servername.9.ini1s0d3l4
```

Der Parameterzugriff anhand dieser Syntax kann intern im DriveServer auf die symbolischen Namen gemappt werden (oder vice versa).

3.2.1.2 Struktur des Namensraumes

Es wird in dieser Spezifikation davon ausgegangen, daß der Namensraum hierarchisch aufgebaut ist.

Mehrachsantriebe

Der DriveServer hat die Aufgabe, die Gerätearchitektur der Antriebsgeräte zu kapseln. Zu den Eigenschaften der Antriebsgeräte gehört es, Subsysteme bilden zu können, die hier unter dem Begriff Mehrachsenantriebe geführt werden.

Mehrere Antriebe können untereinander durch verschiedene Kommunikationssysteme verbunden werden. Diese sind zum Teil herstellerspezifisch. Die Kommunikation zwischen den Antrieben erfolgt über herstellerspezifische Protokolle. Diese gilt es zu kapseln.

Der DriveServer stellt anderen Applikationen eine Systemarchitektur zur Verfügung, welche die Geräte des Subsystems direkt adressierbar macht. Die Architektur wird durch den im DriveServer bildbaren Namensraum zugreifbar gemacht. Beim Lese- und Schreibzugriff auf die Items erfolgt im DriveServer die Abbildung auf das Geräteprotokoll.

Der DriveServer testet beim Anlegen seines internen Namensraumes die bekannten Geräte, die er über das Browse-Interface des Busservers identifiziert hat. Er erkennt die Masterantriebe und identifiziert die Slaveantriebe über ein herstellerspezifisches Protokoll.

Mit den gewonnenen Informationen wird im DriveServer für jedes erkannte Gerät eine Unterhierarchie angelegt, das die Geräteidentifizierung durch eine Applikation ermöglicht. Dabei wird ein Subsystem ebenfalls durch ein <keyword_tag> markiert.

3.2.2 Parametersatztransfer

Für den Parametersatztransfer gibt es eine Subhierarchie zu jedem Antrieb:

Parameter	Bedeutung	m/o
DS_ParameterSets	Anzahl der Parametersätze im Antrieb	O
DS_ParameterSet[0..9]+	Subhierarchie für je einen Parametersatz für Transferaktionen	O
DS_ParameterSetAll	Subhierarchie Transferaktionen aller Parametersätze	O

```
DS_ParameterSet[0-9]+
            Filename
            Action
            State
            Result
DS_ParameterSetAll
            Filename
            Action
            State
            Result
```

Für jeden Parametersatz eines Antriebs gibt es diese Subhierarchie. Die einzelnen Parametersätze werden durch eine Numerierung eindeutig identifiziert.

- Das Item zum Blatt ‘Filename‘ dient der Beschreibung eines vollständigen Dateinamens. Die Datei enthält die herstellerspezifischen Transferdaten. Das Dateiformat ist ebenfalls herstellerspezifisch. Filename ist optional. Wird kein Filename angegegeben, werden die aktuellen Daten des DriveServers zum Antrieb geschrieben bzw. vom Antrieb überschrieben.

- Das Item zum Blatt ‘Action‘ gibt Art des Transfers an. Es können folgende Strings auf das Item geschrieben werden:

  - ”UPLOAD“
  - ”DOWNLOAD”
  - ”VERIFY”
  - ”COMPARE”

  Ein Schreiben auf das Item startet den jeweiligen Vorgang. Die Ergebnisse von VERIFY und COMPARE werden in Dateien gespeichert, deren Name aus dem Wert des Items zum Blatt ‘Filename‘ gewonnen und mit der passenden Erweiterung versehen wird. Das Format ist herstellerspezifisch.

- Auf das Item zum Blatt ‘State‘ kann nur lesend zugegriffen werden. Es gibt Auskunft über den aktuellen Zustand der Transferoperation. Folgende Stati sind definiert:

  - ”READY”
  - ”RUNNING”
  - ”UNDEFINED”

  Ein Schreiben auf das ‘Action‘-Item löst nur in dem Fall eine Aktion aus, wenn ‘State‘ den Wert ”READY” aufweist.

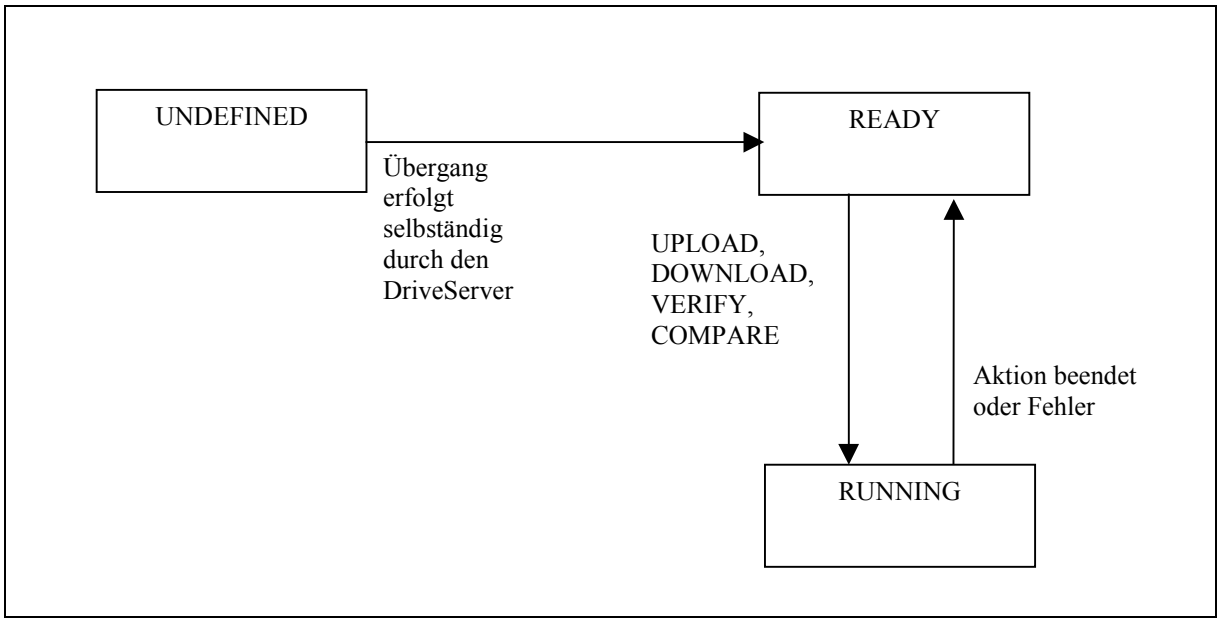

Abbildung 6: Zustandsübergange Parametersatztransfer

- Auf das Item zum Blatt ‘Result‘ kann nur lesend zugegriffen werden. Es gibt Auskunft über den Erfolg der abgeschlossenen Aktion. Dieses Item sollte erst ausgewertet werden, wenn ‘State‘ den Wert ”READY” aufweist. ‘Result‘ hat den Wertebereich und die Codierung von HRESULT.

3.2.3 Anwendungsprogramme des Antriebs

Für Anwendungsprogramme gibt es eine Subhierarchie zu jedem Antrieb:

Parameter	Bedeutung	m/o
DS_Programs	Subhierarchie für Programme im Antrieb	O

```
DS_Programs
      ProgramName1
            Filename
            Action
            State
            Result
            CodeAttribute
      ProgramName2
            Filename
            ...
      ...
```

Für jedes Programm eines Antriebs gibt es diese Subhierarchie. Die einzelnen Programme werden durch einen eindeutigen, frei vergebbaren Knotennamen (hier z.B. `ProgramName1`, `ProgramName2`) identifiziert.

- Das Item zum Blatt '`Filename`' dient der Beschreibung eines vollständigen Dateinamens. Die Datei enthält die herstellerspezifischen Programmdaten. Das Dateiformat ist ebenfalls herstellerspezifisch. `Filename` ist optional.

- Das Item zum Blatt '`Action`' gibt Art des Transfers an. Es können folgende Strings auf das Item geschrieben werden:

  - "`UPLOAD`"
  - "`DOWNLOAD`"
  - "`PREPARE`"
  - "`START`"
  - "`STOP`"
  - "`HALT`"

  Ein Schreiben auf das Item startet den jeweiligen Vorgang. Die einzelnen Werte können in Abhängigkeit des Status geschrieben werden (siehe Abbildung 7).

- Auf das Item zum Blatt '`State`' kann nur lesend zugegriffen werden. Es gibt Auskunft über den aktuellen Zustand der Transferoperation. Folgende Stati sind definiert:

  - "`INITIALIZED`"

    Kein Programmcode ist in den Antrieb geladen worden. Dieses ist der Default-Status.

  - "`LOADED`"

Der Programmcode ist in den Antrieb geladen worden. Das Programm kann mittels "PREPARE" zum Start vorbereitet werden.

- "RUNNING"

  Das Programm wird ausgeführt.

- "STOPPED"

  Das Programm wurde beendet. Das Programm kann jederzeit wieder gestartet werden. Es beginnt am Programmanfang.

- "HALTED"

  Das Programm wurde angehalten. Es kann jederzeit wieder gestartet werden. Es beginnt an der Stelle der Unterbrechung fortzufahren.

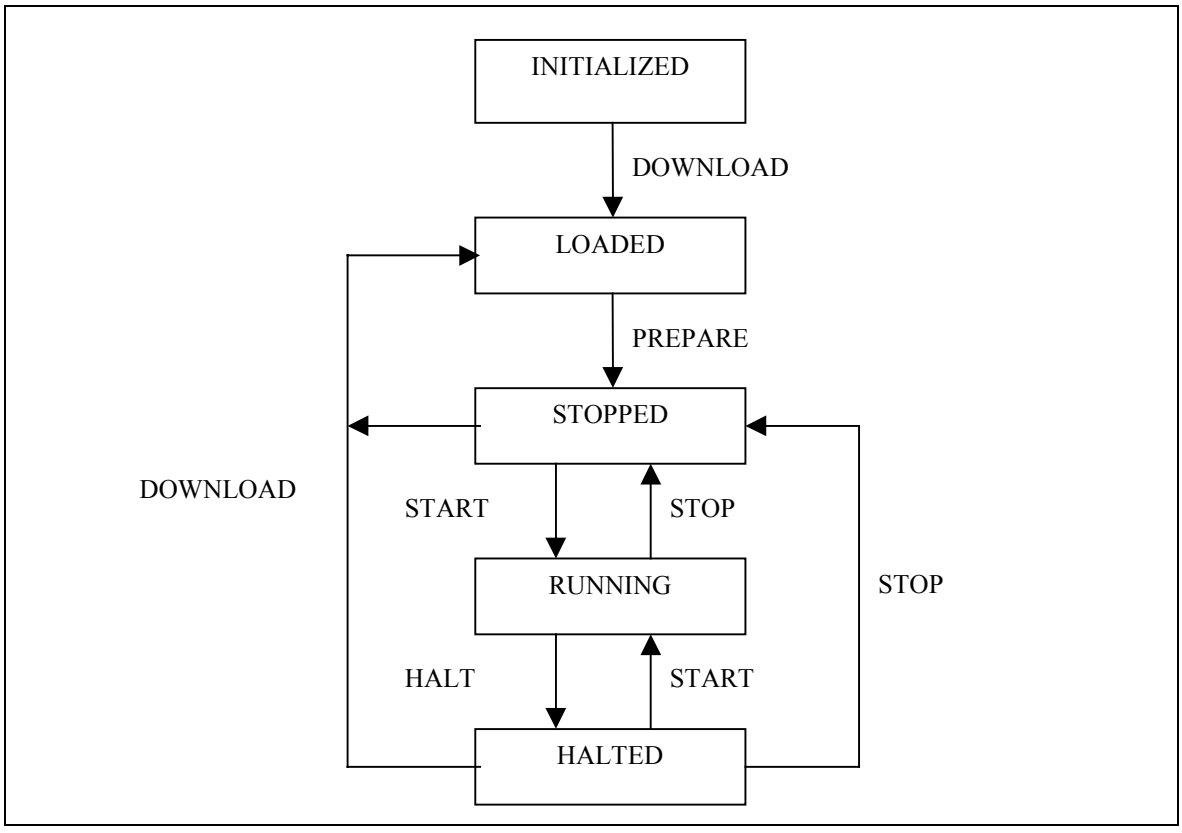

Abbildung 7: Zustandsübergange eines Programms

- Auf das Item zum Blatt 'Result' kann nur lesend zugegriffen werden. Es gibt Auskunft über den Erfolg der abgeschlossenen Aktion. 'Result' hat den Wertebereich und die Codierung von HRESULT.

- Das Item zum Blatt 'CodeAttribute' gibt die Startart des Programms nach einem Neustart des Antriebs an (Wertebereich: VT_I4):

Wert	Bedeutung
0	Nach einem Neustart des Antriebs startet das Programm nicht selbständig.
1	Nach einem Neustart des Antriebs startet das Programm selbständig.
2 ... EFFF	Reserviert
FFFF ... -1	Herstellerspezifisch

Die Art und Weise der Übertragung der Programme und deren Bedienung erfolgt herstellerspezifisch.

3.2.4 Identifikation eines Antriebs

Der DriveServer identifiziert im Busserver die verfügbaren Antriebe mit folgendem Algorithmus:

1. Browsen des Namensraumes im Busserver. Dabei erfolgt ein Suchen nach dem `<keyword_tag>` `DS_Vendorname`. Die Ausgabe des Suchergebnisses sollte `flat` formatiert erfolgen.

2. Zu den gefundenen Blättern sind die Item-Ids abzufragen und in gerätespezifischen und parameterspezifischen Anteil zu zerlegen.

3. Der DriveServer legt im Busserver die zur Identifikation des Antriebs benötigten Items an und wertet diese herstellerspezifisch aus.

4. Nach der Auswertung können diese Items im Busserver freigegeben werden.

Der DriveServer ist nun dafür verantwortlich, die von ihm als bedienbar erkannten Geräte seinen Clients zu präsentieren.

3.3 Busserver Funktionalität

3.3.1 Übersicht

Ein Busserver bietet die Kommunikation von Anlagenwerten über ein Kommunikationsmedium. Busserver sind immer herstellerspezifisch und liegen nicht im Fokus dieser Spezifikation. Damit ein DriveServer optimal mit dem jeweiligem Busserver kooperieren kann, muß ein Busserver jedoch gewisse Anforderungen erfüllen.

- Ein Busserver muß das optionale OPC-Interface ‚`IOPCBrowseServerAddressSpace`‘ unterstützen.

- Es sollte möglich sein, den Namensraum dynamisch zu erweitern. Z.B. durch das Anlegen eines OPC-Items, für das kein Item im Namensraum definiert ist. Es ist dann möglich, OPC-Items anzulegen, die nicht zuvor durch statische Konfiguration dem Namensraum des Busservers bekannt gemacht worden sind.

- Der Busserver muß sich mit den entsprechenden Component Categories für die unterstützten Bussysteme registrieren

3.3.2 Namensraum

Der Namensraum des Busservers ist herstellerspezifisch, nach kommunikationsspezifischen Aspekten aufgebaut.

Damit ein DriveServer mit einem Busserver zusammenarbeiten kann, müssen folgende Regeln eingehalten werden:

- Die reservierten Namen des Busservers müssen eingehalten werden.
- Der Namensraum muß vom DriveServer aus `flat` formatiert durchsucht werden können.

Die Initialisierung des Namensraumes ist implementierungsspezifisch (herstellerspezifisch). Es wird zwischen zwei Arten der Konfiguration unterschieden:

- Statische Konfiguration: Sie kann z. B. durch Einlesen einer Anlagenkonfiguration beim Start des Busservers erfolgen. Es werden in der Regel beliebige symbolische Namen (nach DriveServer-Spezifikation oder herstellerspezifisch) vergeben.
- Dynamische Konfiguration: z.B. durch ItemIds mit einer definierten Syntax.

Jedes Gerät ist im Namensraum über ein `<keyword_tag>` eindeutig zu identifizieren.

Alle Parameter eines Gerätes sind als Blätter eines gerätespezifischen Astes auffindbar.

3.3.2.1 Reservierte Namen

Diese Spezifikation reserviert spezifische Namen für bestimmte Standardfunktionalität.

Busserver spezifische Namen

Jedes Gerät verfügt über Standardparameter, welche zur Identifizierung eines Gerätes durch den Client (DriveServer) benutzt werden können (Schreibweise beachten!).

Parameter	Bedeutung	m/o
`DS_Devicename`	Gerätename	M
`DS_DeviceID`	Antriebsreglerkennung (Busadresse)	M
`DS_Vendorname`	Herstellername	M
`DS_Devicetype`	Gerätetyp	O
`DS_Vendorcode`	Herstellercode	O
`DS_BusSystem`	Kennung des Feldbussystems in Form einer Component Category (Registry Format), Definition in Kapitel 4	O
`DS_BusPort`	Busnetz bzw. –strang	O

Anmerkungen:

Der Parameter `DS_DeviceID` muß für den jeweiligen Busstrang (Busserver können mehrere Busstränge unterstützen z.B. mehrere PC-Karten, mehrere Ports an einer PC-Karte) eindeutig sein, da er zum Aufbau des DriveServer-Namensraumes verwendet wird. Kommen bei einem Busserver mehrere Busnetze bzw. – stränge vor, ist die `DS_DeviceID` nicht mehr eindeutig, da an jedem Busstrang z.B. die Geräteadresse „5“ vorkommen kann. Bei mehreren Bussträngen am Busserver muß der Busserver in dem Parameter `DS_BusPort` den Busstrang kennzeichnen, so daß der Namensraumaufbau im DriveServer über `DS_DeviceID` und `DS_BusPort` eindeutig ist.

Das Zeichen "." darf innerhalb der Parameter nicht verwendet werden, da es als OPC-Trennzeichen verwendet wird.

Der Wert der Parameter `DS_Devicetype, DS_DeviceID, DS_BusSystem, DS_BusPort` sowie `DS_Vendorcode` ist sprachenunabhängig, da er zur Adressierung und Identifikation benötigt wird.

Unterstützt ein Busserver mehrere Bussysteme, so kann aus den in der Registry eingetragenen Component Categories nicht ermittelt werden, an welchem Bus das Gerät angeschlossen ist. Dieses ist aber mit dem Parameter `DS_BusSystem` möglich.

Namenskonvention

Für Parameter ohne symbolischen Namen wird die folgende Namenskonvention definiert. Abhängig davon, für welches Bussystem der Busserver steht, sind 2-stufige oder 3-stufige Adressierung zu unterscheiden.

Bei den 2-stufig zu adressierenden Bussytemen (Beispiel: CANopen) werden die Parameter über Index und Subindex angesprochen. Diese Adressierung entspricht auch der Anwendersicht auf den DriveServer. Die 3-stufige Adressierung findet z.B. bei DeviceNet Anwendung. Dabei ist dann die <index_info> als Instanz, und die <subindex_info> als Attribut zu verstehen. Für die Klasse wird eine zusätzliche Kennung definiert. Diese darf nur bei Bussystemen verwendet werden, bei denen eine 3-stufige Adressierung eingesetzt wird.

```
<parameter_tag>      := <parameter_channel> | <process_channel>
<parameter_channel>  := "PAR"
                        [<class_info>]<index_info>[<subindex_info>]
                        <datatype_info>[<length_info>]
                        [<extra_info>]
<process_channel>    := <process_cannel_info>
                        <index_info><subindex_info>
                        <datatype_info><length_info>[<extra_info>]

<process_channel_info> :=    "OUT" | "IN"
<class_info>         := (C|c)[0-9]+
<index_info>         := (I|i)[0-9]+
<subindex_info>      := (S|s)[0-9]+
<datatype_info>      := (D|d)(VT_UI1 | VT_ARRAY | ... )
                        (Die einzusetzenden Zahlenwerte sind in Kapitel 5 aufgeführt)
<length_info>        := (L|l)[0-9]+
<extra_info>         := (X|x)<String>
```

`class_info:` Objektklasse; nur bei 3-stufiger Adressierung

`index_info:` 2-stufige Adressierung: Index des Objekts (beginnt bei „0")

3-stufige Adressierung: Instanz einer Klasse

Prozeßdaten: Nummer des Prozeßdatenkanals

`subindex_info:`	2-stufige Adressierung: Subindex eines Objektes (beginnt bei „0“). Handelt es sich bei dem angesprochenen Objekt nicht um einen indizierten Parameter, so ist kein Subindex anzugeben. 3-stufige Adressierung: Attribut einer Instanz Prozeßdaten: Zugriff auf ein Byte, falls das Objekt aus mehreren Bytes besteht. ‚0‘ bedeutet, daß auf alle Bytes des Objekts zugegriffen wird
`length_info:`	Länge der geforderten Daten in Bits

Beispiele :

`INI1S0D3:`	IN-Prozeßvariable auf Prozeßdatenkanal 1. Das Objekt soll vollständig auf ein `VT_I4` abgebildet werden.
`pari24564D3 :`	Parametervariable mit Index 24564, Datentyp 3 (entspricht `VT_I4`)
`pari24566S0D3 :`	Indizierte Parametervariable mit Index 24566 und Subindex 0, Datentyp 3 (entspricht `VT_I4`)
`parc100i10s40d8:`	Parametervariable Klasse 100, Instanz 10, Attribut 40 in einem 3-stufig zu adressierenden Bussystem; als Datentyp `VT_BSTR` darzustellen

mit vollständigem Pfad:

```
OPC://PROGID- Bus Server /DP://brd0.seg0.dev9/ini1s0d3l4
OPC://PROGID- Bus Server /DP://brd0.seg0.dev9/pari24566s0d3
```

3.3.2.2 Struktur des Namensraumes

Es wird in dieser Spezifikation davon ausgegangen, daß der Namensraum hierarchisch aufgebaut ist. Er ist so gegliedert, daß alle Parameter eines Gerätes als Blätter eines gerätespezifischen Astes auffindbar sind. D.h. im `flat` formatierten Namensraum ergibt sich folgende Struktur für alle Zweignamen:

```
<item_tag>          := <device_tag><parameter_tag>
```

Das `<device_tag>` enthält alle Zeichen und Strings, die der OPC Server zur Identifikation eines Gerätes benötigt, inklusive einer möglichen Trennung zwischen gerätespezifischer Zeichenkette und dem `<parameter_tag>`.

4 Registrierung

Die DRIVECOM definiert die folgenden Component Categories. Es sind die CATIDs und die zugehörige Beschreibung aufgeführt.

“ DriveServer Version 1.0”

```
CATID_DriveServer10 = {276BC1C0-0BC1-11D4-A78F-525405F5B2CF}
```

“ Busserver Version 1.0”

```
CATID_BusServer10 = {276BC1C1-0BC1-11D4-A78F-525405F5B2CF}
```

“ **Busserver CANopen**”

```
CATID_CANOPEN = {3CF19FF1-907B-11d4-B4A7-0050DA3F121C}
```

“ **Busserver PROFIBUS-PA**”

```
CATID_PROFIBUS-PA = {3CF19FF2-907B-11d4-B4A7-0050DA3F121C}
```

“ **Busserver PROFIBUS-DP**”

```
CATID_PROFIBUS-DP = {3CF19FF3-907B-11d4-B4A7-0050DA3F121C}
```

“ **Busserver PROFIBUS-FMS**”

```
CATID_PROFIBUS-FMS = {3CF19FF4-907B-11d4-B4A7-0050DA3F121C}
```

“ **Busserver INTERBUS**”

```
CATID_INTERBUS = {3CF19FF5-907B-11d4-B4A7-0050DA3F121C}
```

“ **Busserver DeviceNet**”

```
CATID_DEVICENET = {3CF19FF6-907B-11d4-B4A7-0050DA3F121C}
```

“ **Busserver LON**”

```
CATID_LON = {3CF19FF7-907B-11d4-B4A7-0050DA3F121C}
```

Es wird gefordert, daß die Server bei Ihrer Installation jede von ihnen unterstützte Category zunächst in der Registry eintragen (`HKEY_CLASSES_ROOT\Component Categories`). Zusammen mit dem CATID-Wert wird die textuelle Beschreibung mit gespeichert.

Jeder Busserver muß alle Busprotokolle, die er unterstützt, als CATID eintragen. Wenn keine CATID zur Kennzeichnung eines Busprotokolls vorhanden ist, wird davon ausgegangen, daß dem Busserver die unter 3.3.2 definierte Syntax ausreicht, um den gewünschten Parameter anzusprechen und keine weitere Protokollumsetzung im DriveServer notwendig ist. Der Server selbst deklariert seine Unterstützung einer CATID durch die entsprechenden Einträge unterhalb des Schlüssels ‚`Implemented Categories`‘ in der Registry.

5 Variant-Datentypen

In der Namenskonvention werden die Microsoft-Variant-Datentypen verwendet. Der Vollständigkeit halber sind die in Frage kommenden Typen in der nachfolgenden Tabelle zusammengestellt.

Varianttyp	Wert	Beschreibung
VT_EMPTY	0	Untypisiert
VT_I2	2	2-Byte-Integer mit Vorzeichen
VT_I4	3	4-Byte-Integer mit Vorzeichen
VT_R4	4	4-Byte-Gleitkommazahl
VT_R8	5	8-Byte-Gleitkommazahl
VT_CY	6	Währung
VT_DATE	7	Datum
VT_BSTR	8	Zeichenkette
VT_BOOL	11	Boolean, True=-1, False=0
VT_UI1	17	1-Byte-Integer ohne Vorzeichen

Der Datentyp VT_ARRAY beinhaltet neben den Daten auch Informationen über den Datentyp der Feldelemente und die Feldgrenzen. Die Angabe des Datentyps geschieht durch Kombination der Werte VT_ARRAY und der Varianttypen aus obiger Tabelle durch bitweises Oder. Z.B. ergibt sich als Wert für den Datentyp "Feld von 1-Byte-Integer ohne Vorzeichen" 8209 (= 8192 Oder 17). Die Länge des Feldes ist variabel.

Varianttyp	Wert	Beschreibung
VT_ARRAY	8192	Feld

6 Glossar

Bus-Server	OPC-Server, der eine Kommunikation mit Geräten ermöglicht. Der Namensraum und die verfügbaren Items des Servers sind kommunikationsspezifisch
`<device_tag>`	Zeichenkette, die den gerätespezifischen Teil einer ItemID repräsentiert
DriveServer	OPC-Server, der eine Kommunikation mit Geräten (Antrieben) ermöglicht. Der Namensraum und die verfügbaren Items des Servers sind gerätespezifisch
`<item_Tag>`	Zeichenkette, die eine vollständige Parameteradresse repräsentiert und als ItemID verwendet werden kann
`<keyword_tag>`	Schlüsselwort, anhand dessen im Namensraum Geräte erkannt und das zum Zugriff notwendige `<device_tag>` ermittelt werden kann.
Namensraum	Übersicht über die dem OPC-Server bekannten Datenpunkte. Ein Namensraum kann hierarchisch strukturiert oder auch `flat` formatiert sein, d.h. alle verfügbaren Datenpunkte werden in einer Liste dargestellt
`<parameter_tag>`	Zeichenkette, die den parameterspezifischen Teil einer ItemID repräsentiert

7 Literatur

OPC OLE for Process Control, Data Access Custom Interface Standard, Version 2.0

www.opcfoundation.org