AnaGate Motion Benutzerhandbuch

Analytica GmbH

AnaGate Motion Benutzerhandbuch

Analytica GmbH

Veröffentlicht 22. April 2010

Zusammenfassung

Das AnaGate Motion Manual umfasst die Beschreibung der Hardware-Komonenten und der CANopen-Schnittstelle der Anagate Motion Serie.

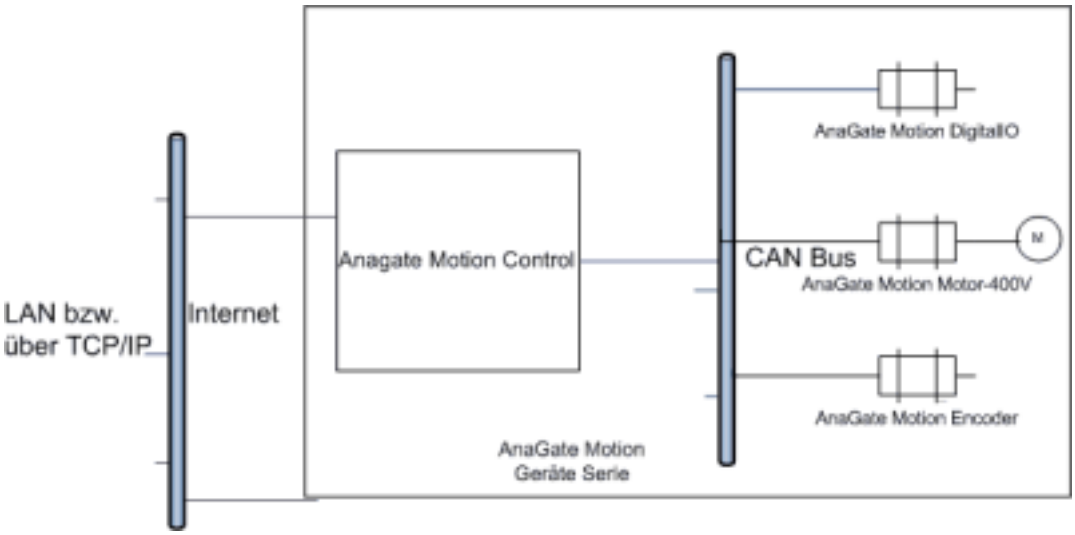
Inhaltsverzeichnis

Tabellenverzeichnis

Beispiele

Einleitung

Die Geräteserie AnaGate Motion besteht aus verschiedenen Modulen, die die Ansteuerung von unterschiedlicher Motortypen (24Vdc bzw. 400Vac) erlaubt. Eine Erweiterung der Basisfunktionalität ist über optionale Zusatzmodule gewährleistet (digitale und analog Eingänge bzw. Ausgänge, Encode, etc.). Sämtliche Module verfügen über einen CANopen-Anschluß, über die die Module gesteuert werden können.



Module

Control Modul	Dieses Modul dient der Steuerung der einzelnen Module (Motor/Digital IO/ etc.). Für die Kommunikation mit einer übergeordneten Steuerung enthält dieses Modul einen Ethernet Anschluß. Zum Anschluss von weiteren CAN/ CANopen Geräte ist noch ein weiterer CAN Bus vorhanden. Als Basis für die Erstellung eigener Anwendungen läuft auf dem Modul ein Linux Betriebssystem (Kernel 2.6.27) mit 400MHz ARM9 CPU, 64MB RAM und 256MB Flash. Eigene Anwendungen können mittels einer einfachen API unter der Programmiersprache C/C++ oder über die Scriptsprache LUA erstellt werden.
Control Modul LCD	Wie das Control Modul, jedoch mit integriertem monochromen LCD Display (128x64 Punkte) und Touchscreeninterface. Bei diesem Modul ist dann anstelle von 2 CAN Bussen nur noch einer verfügbar.
CANopen Kopfmodul	Dieses Modul dient der Steuerung der einzelnen Module (Motor/Digital IO/ etc.). Für die Kommunikation mit einer übergeordneten Steuerung enthält

	dieses Modul einen Ethernet Anschluß. Zum Anschluss von weiteren CAN/ CANopen Geräte ist noch ein weiterer CAN Bus vorhanden.
Motor 24Vdc BT Modul	Dieses Modul dient der Ansteuerung von 24V Brush Type Motoren. Über eine flexibel einstellbare PWM kann die Sollgeschwindigkeit der Motoren eingestellt werden. Dieses Modul gibt es in zwei verschiedenen Varianten: • 2 Motoren mit jeweils max. 10A Dauerstrom • 4 Motoren mit jeweils max. 3A Dauerstrom Jede dieser Varianten besitzt zusätzlich noch 2 Lichtschranken- und Encodereingänge.
Motor 24Vdc BL Modul	Mittels dieses Moduls können 24V Brush Less (EC) Motoren mit integrierter Kommutierungselektronik (z.B. von Maxon, ebmpapst) angesteuert werden. Es werden insgesamt 2 Motoren mit einer maximalen Leistung von 150W unterstützt. Das Modul besitzt zusätzlich noch 2 Lichtschranken- und Encodereingänge.
Motor 400Vac Modul	Über dieses Modul können insgesamt 2x 400V Motoren bis zu einer max. Leistung von 2kW angesteuert werden. Hierbei dient die Steuerung als intelligenter Ein-/Ausschalter für die Motoren mit einer integrierten Strommessung (1 Phase). Auf dem Modul sind ebenfalls zwei Lichtschranken- und Encodereingänge vorhanden.
Digital IO	Durch die Benutzung des Standardprotokolls CANopen können auch Fremdprodukte an die AnaGate Motion Geräteserie angeschlossen werden. Dies wird auch im Falle von Frequenzumrichtern (FU) für die Ansteuerung von 400V Motoren so durchgeführt, um die vielen verschiedenen Kundenwünsche realisieren zu können.
Encoder	Dieses Modul besitzt insgesamt 4x Binär-Encoder Eingänge um optische/ mechanische Encoder anzuschließen. Über den integrierten Quadratur Counter können Signale bis zu max. 10MHz aufgenommen werden.

Das AnaGate Motion Manual umfasst die Beschreibung der Hardware der verschiedenen Module sowie der CANopen Schnittstelle der AnaGate Motion Serie.

Im folgenden geht das Dokument auf die aufgelisteten Bereiche ein:

- Hardware
- CANOpen Allgemein
- CANOpen DSP402 (Geräteprofil für Motrorsteuerung)
- CANOpen DSP401 (Geräteprofil für Digital IO)
- Spezielle Eigenschaften

Impressum

Analytica GmbH
Vorholzstraße 36
76137 Karlsruhe
Germany
Fon 0721-43035-0
Fax 0721-43035-20

www.analytica-karlsruhe.de [http://www.analytica-karlsruhe.de]

Technischer Support

Die Hardware-Serie AnaGate Motion und die vorhandenen Programmierschnittstellen werden von der Analytica GmbH entwickelt und unterstützt. Technische Unterstützung kann wie folgt angefordert werden:

Internet EMail

Für technische Unterstützung Über Internet, senden Sie bitte eine EMail an

<support@analytica-karlsruhe.de>

Helfen Sie uns bei der optimalen Unterstützung, und halten Sie stets folgende Informationen bereit, wenn Sie mit dem Support in Verbindung treten.

- AnaGate Motion Hardware-Serie und Geräte-Versionsnummer
- Name und Version des verwendeten Betriebssystems

Hardwarebeschreibung der AnaGate Motion Serie

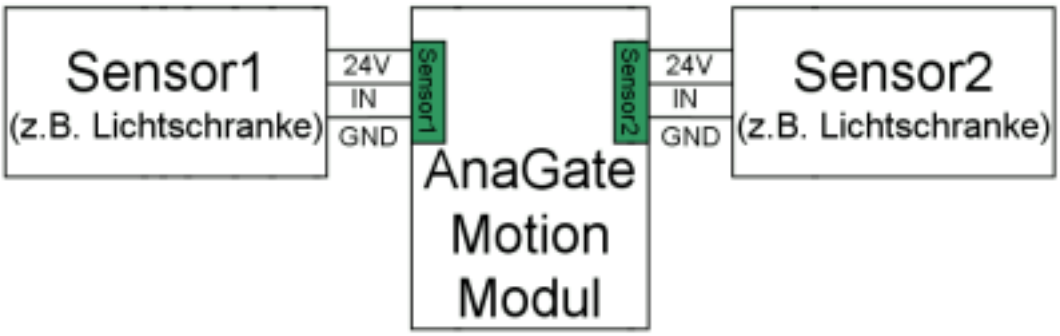
Inhaltsverzeichnis

Module der AnaGate Motion Serie

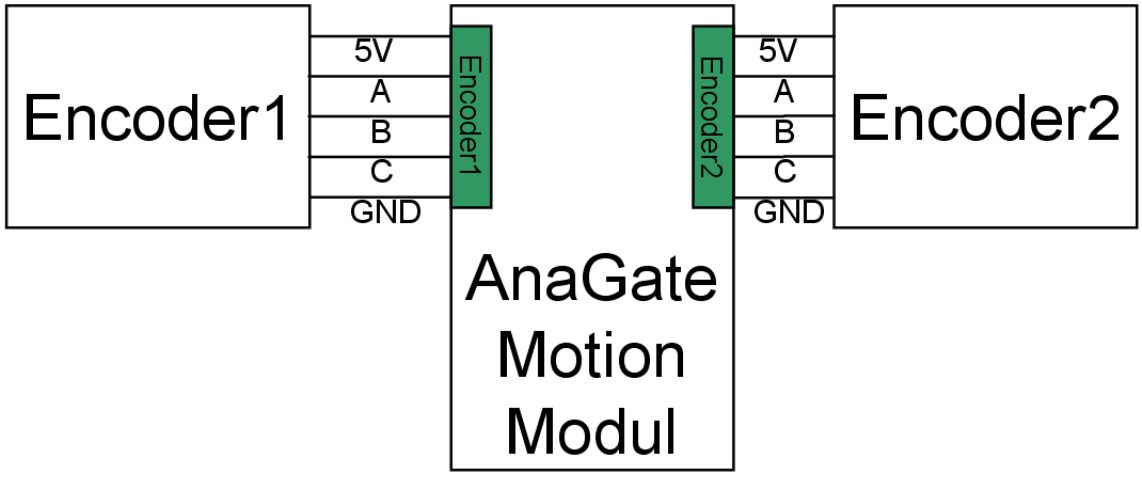
Nachfolgend werden die verschiedenen AnaGate Motion Module mit ihren Hardwareschnittstellen beschrieben.

- Control Modul
- CANopen Kopfmodul
- Motor 24Vdc BL (Brushless)
- Motor 24Vdc BT (Brushtype)
- Motor 400Vac
- Digital IO Modul

Beschaltung Sensoreingänge



Beschaltung Encodereingänge



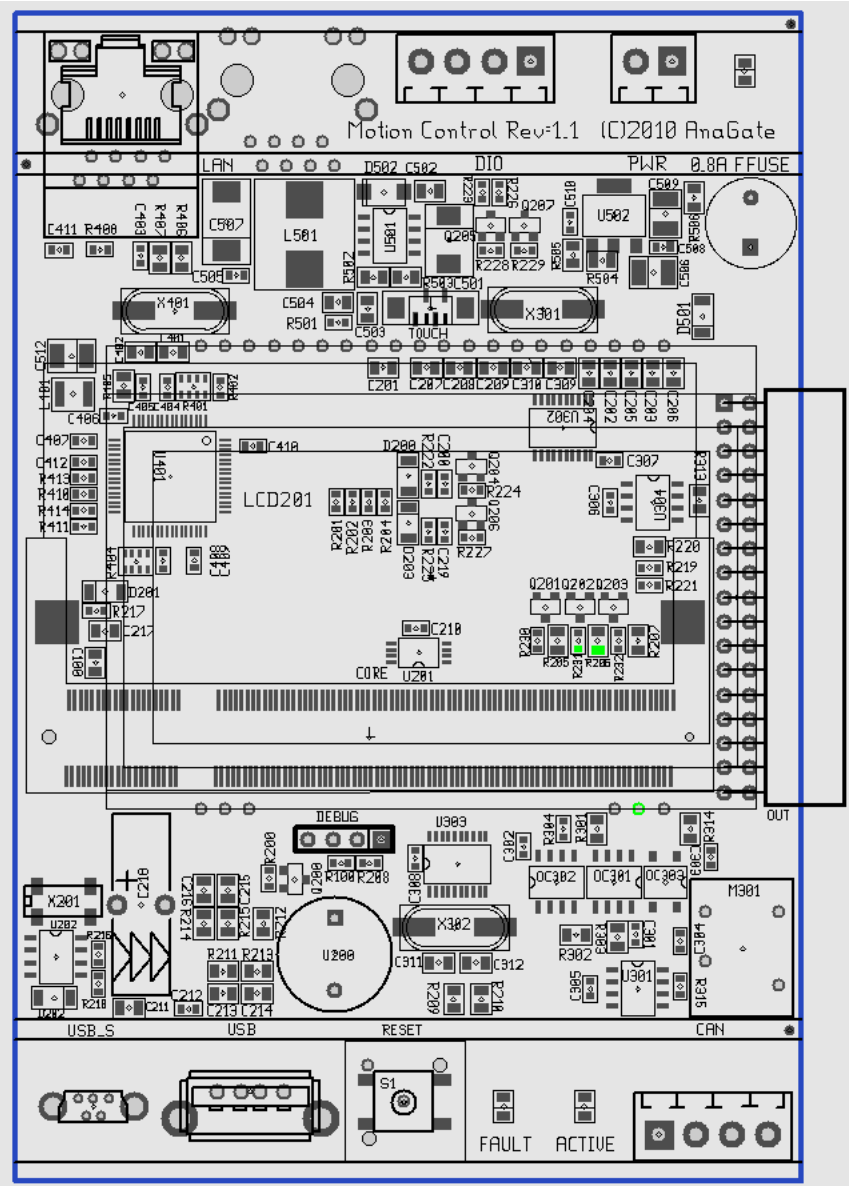
Steuereinheit (Kontrollmodul)

Datenblatt Control Modul

Tabelle 1. Technische Daten (Control Modul)

Maße:	L x B x H	To be defined
	Gewicht	To be defined
Spannungsversorgung	Eingangsspannung	24V DC
Temperaturbereich	Industrial	-20 ... +70°C
Schutzklasse		IP20
Prozessor	ARM9 (32 bit)	
Betriebssystem	Linux (Kernel 2.6.20)	
LAN Interface	Baudrate TCP/IP Schnittstellen	10/100 Mbps statische oder dynamisch (DHCP) IP Adresse RJ45 Buchse
LAN Interface	Baudrate TCP/IP Schnittstellen	10/100 Mbps statische oder dynamisch (DHCP) IP Adresse RJ45 Buchse
CAN Bus	Baudrate CAN Controller Schnittstellen	10, 20, 50, 62.5, 100, 125, 250, 500, 800 kbps bzw. 1 Mbps einstellbar per Software. 2x Microchip MCP2515 1x 4polige Steckerklemme ; 1x interner Bus
EG Richtlinien	RoHS, CE	
Software	Konfiguration	Web-Oberfläche
Programmierung		C/C++ und LUA Software kann auf dem Control Modul oder auf einem Windows/Linux PC ablaufen

Beschaltung Steuereinheit (Kontrollmodul)


Motion Control Rev:1.1 (C)2010 AnaGate
LAN
DIO
PWR 0.8A FFUSE
LCD201
TOUCH
CORE
DEBUG
OUT
USB_S
USB
RESET
CAN
FAULT
ACTIVE

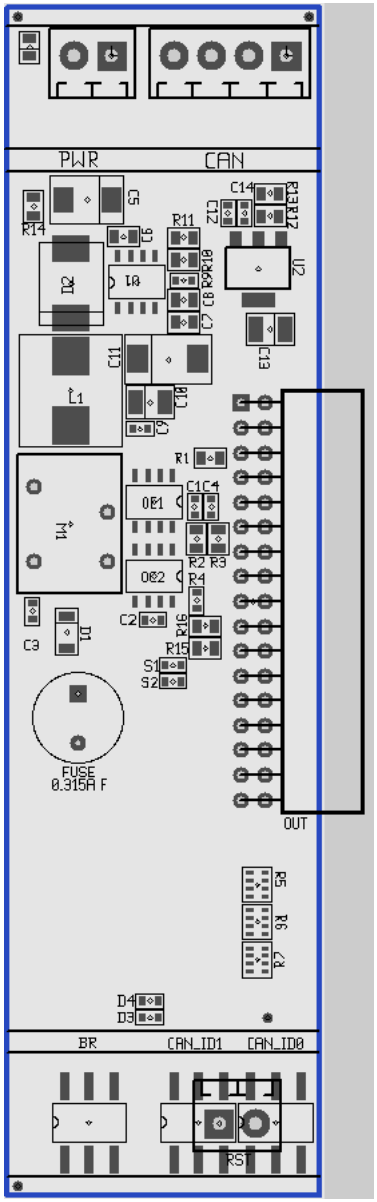
CANopen Kopfmodul

Datenblatt CANopen Kopfmodul

Tabelle 2. Technische Daten (CANopen Kopfmodul)

Maße:	L x B x H	To be defined
	Gewicht	To be defined
Spannungsversorgung	Eingangsspannung	24V DC
Temperaturbereich	Industrial	-20 ... +70°C
Schutzklasse		IP20
Prozessor	ARM9 (32 bit)	
Betriebssystem	Linux (Kernel 2.6.20)	
LAN Interface	Baudrate TCP/IP Schnittstellen	10/100 Mbps statische oder dynamisch (DHCP) IP Adresse RJ45 Buchse
LAN Interface	Baudrate TCP/IP Schnittstellen	10/100 Mbps statische oder dynamisch (DHCP) IP Adresse RJ45 Buchse
CAN Bus	Baudrate CAN Controller Schnittstellen	10, 20, 50, 62.5, 100, 125, 250, 500, 800 kbps bzw. 1 Mbps einstellbar per Software. 2x Microchip MCP2515 1x 4polige Steckerklemme ; 1x interner Bus
EG Richtlinien	RoHS, CE	
Software	Konfiguration	Web-Oberfläche
Programmierung		C/C++ und LUA Software kann auf dem Control Modul oder auf einem Windows/Linux PC ablaufen

Beschaltung CANopen Kopfmodul


PWR
CAN
OUT
BR
CAN_ID1
CAN_ID0
RST
FUSE
0.315A F

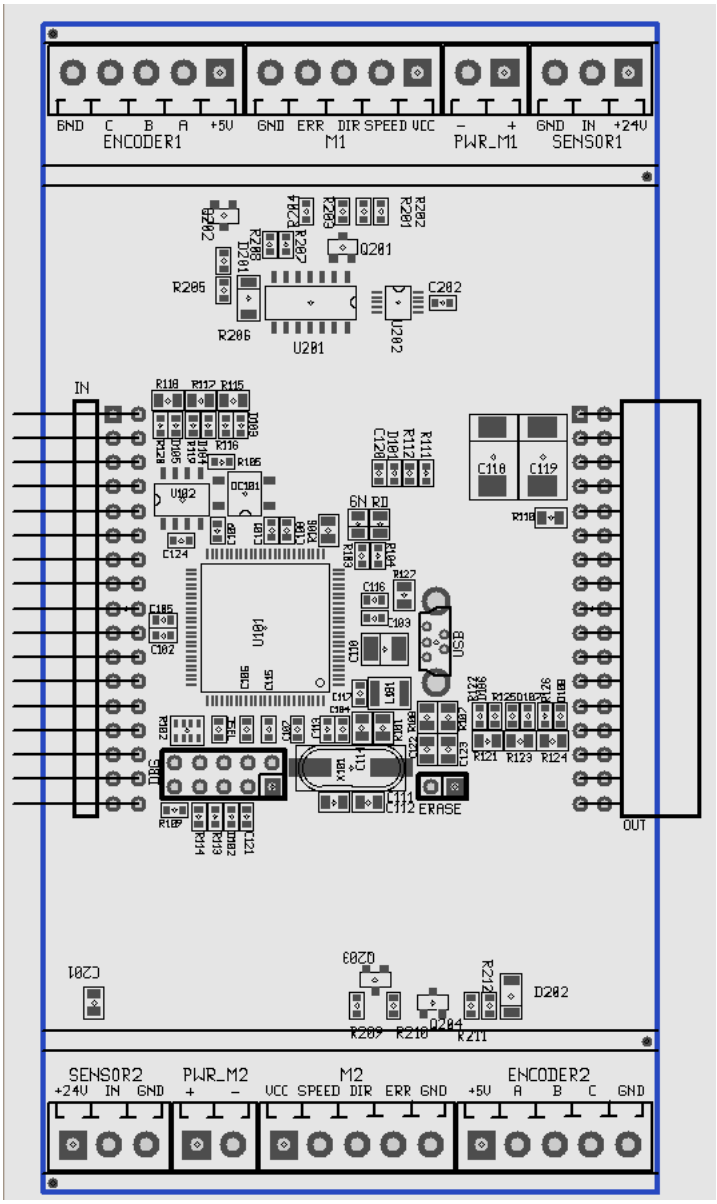
Motor 24Vdc BL (Brushless)

Datenblatt 24Vdc BL (Brushless)

Tabelle 3. Technische Daten (24Vdc BL Modul)

Maße:	L x B x H	To be defined
	Gewicht	To be defined
Spannungsversorgung	Eingangsspannung	24V DC
Temperaturbereich	Industrial	-20 ... +70°C
Schutzklasse		IP20
Prozessor	ARM7 (32 bit)	55MHz
Motor	Anzahl Schnittstellen Versorgungsspannung V_{out} I_{signal} Betriebsarten	2 12 - 24V 0 - 10V <50mA Geschwindigkeit in % instellbar ; Rampe
2x Sensoreingänge (z.B. Lichtschranke)		24V Pegel
2x Encodereingänge		5V Pegel
CAN Bus	Baudrate CAN Controller Schnittstellen	10, 20, 50, 100, 125, 250, 500, 800 bzw. 1000 kbps einstellbar über Codierschalter. 1x integriert in CPU 1x interner Bus
EG Richtlinien	RoHS, CE	
Software Schnittstelle		CANopen® DSP402

Beschaltung 24Vdc BL (Brushless)


GND C B A +5V
ENCODER1
GND ERR DIR SPEED VCC
M1
– +
PWR_M1
GND IN +24V
SENSOR1
IN
OUT
USB
ERASE
SENSOR2
+24V IN GND
PWR_M2
+ –
M2
VCC SPEED DIR ERR GND
ENCODER2
+5V A B C GND

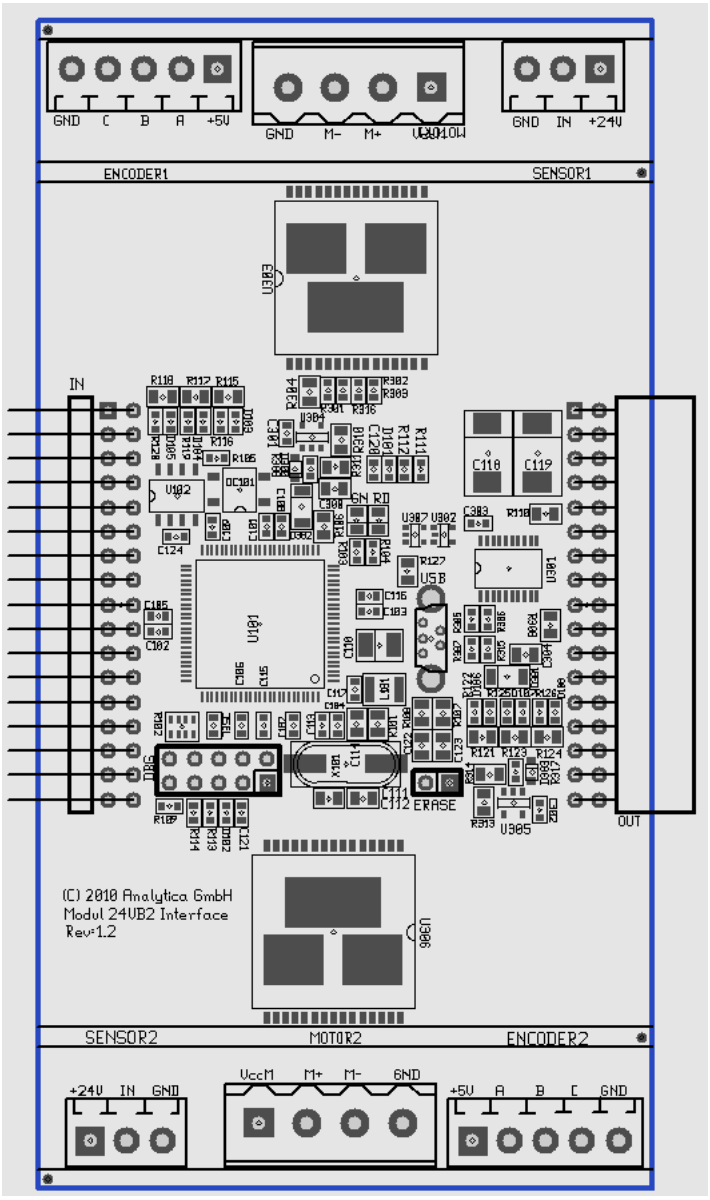
Motor 24Vdc BT (Brushtype)

Datenblatt 24Vdc BT (Brushtype)

Tabelle 4. Technische Daten (24Vdc BT Modul)

Maße:	L x B x H	To be defined
	Gewicht	To be defined
Spannungsversorgung	Eingangsspannung	24V DC
Temperaturbereich	Industrial	-20 ... +70°C
Schutzklasse		IP20
Prozessor	ARM7 (32 bit)	55MHz
Motor	Anzahl Schnittstellen Treiber (H-Brücke) Versorgungsspannung I_{max} PWM Frequenz Betriebsarten	2 ST VNH3SP30 9 - 30V 30A 1 - 10KHz Geschwindigkeit in % instellbar ; Rampe
2x Sensoreingänge (z.B. Lichtschranke)		24V Pegel
2x Encodereingänge		5V Pegel
CAN Bus	Baudrate CAN Controller Schnittstellen	10, 20, 50, 100, 125, 250, 500, 800 bzw. 1000 kbps einstellbar über Codierschalter. 1x integriert in CPU 1x interner Bus
EG Richtlinien	RoHS, CE	
Software Schnittstelle		CANopen® DSP402

Beschaltung 24Vdc BT (Brushtype)


GND C B A +5V
GND M- M+ VccM
GND IN +24V
ENCODER1
SENSOR1
IN
OUT
USB
ERASE
(C) 2010 Analytica GmbH
Modul 24VB2 Interface
Rev:1.2
SENSOR2
MOTOR2
ENCODER2
+24V IN GND
VccM M+ M- GND
+5V A B C GND

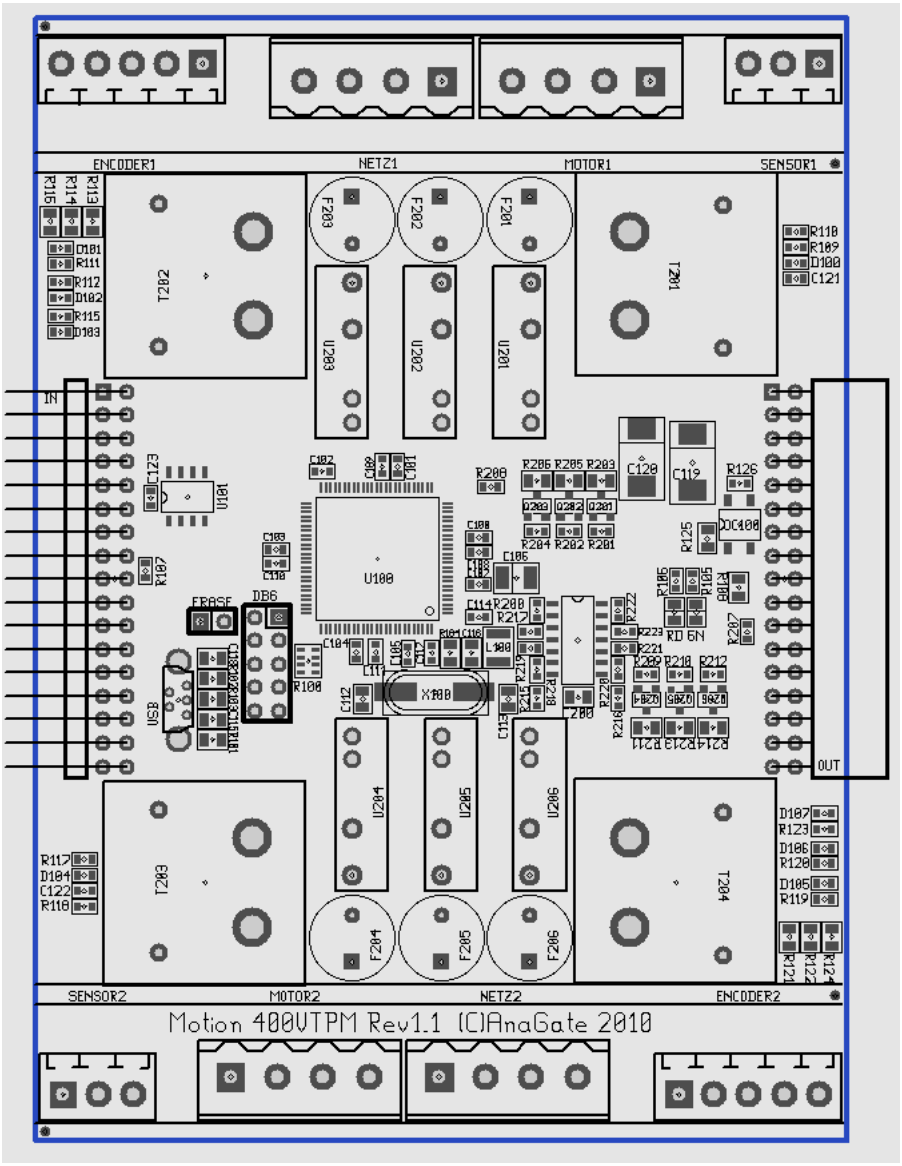
Motor 400Vac

Datenblatt 400Vac

Tabelle 5. Technische Daten (400Vac Modul)

Maße:	L x B x H	To be defined
	Gewicht	To be defined
Spannungsversorgung	Eingangsspannung	24V DC
Temperaturbereich	Industrial	-20 ... +70°C
Schutzklasse		IP20
Prozessor	ARM7 (32 bit)	55MHz
Motor	Anzahl Schnittstellen Typ Spannung Leistung Betriebsarten sonstiges	2 3-Phasen Motor 400V <2KW An/Aus Stromüberwachung
2x Sensoreingänge (z.B. Lichtschranke)		24V Pegel
2x Encodereingänge		5V Pegel
CAN Bus	Baudrate CAN Controller Schnittstellen	10, 20, 50, 100, 125, 250, 500, 800 bzw. 1000 kbps einstellbar über Codierschalter. 1x integriert in CPU 1x interner Bus
EG Richtlinien	RoHS, CE	
Software Schnittstelle		CANopen® DSP402

Beschaltung 400Vac


ENCODER1
NETZ1
MOTOR1
SENSOR1
IN
OUT
ERASE
DB6
USB
SENSOR2
MOTOR2
NETZ2
ENCODER2
Motion 400VTPM Rev1.1 (C)AnaGate 2010

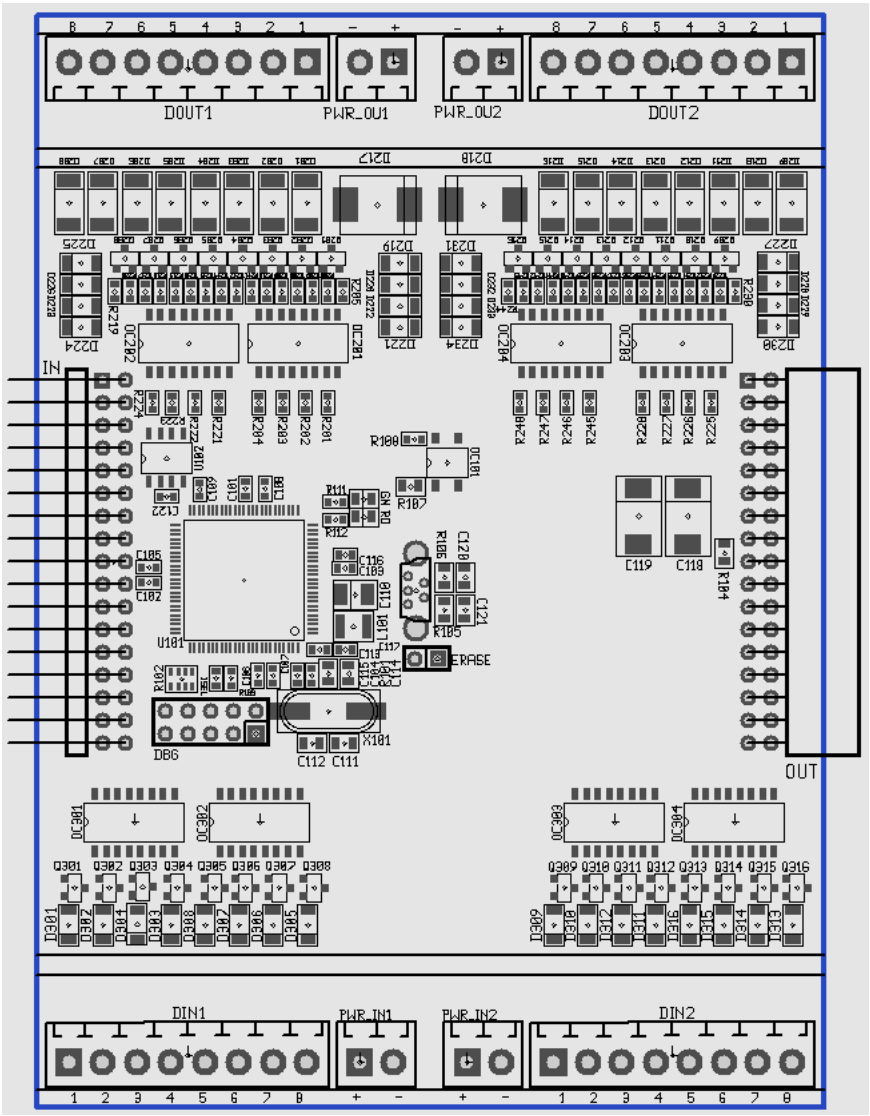
Digital IO Modul

Datenblatt Digital IO

Tabelle 6. Technische Daten (Digital IO Modul)

Maße:	L x B x H	To be defined
	Gewicht	To be defined
Spannungsversorgung	Eingangsspannung	24V DC
Temperaturbereich	Industrial	-20 ... +70°C
Schutzklasse		IP20
Prozessor	ARM7 (32 bit)	55MHz
Schnittstellen	Eingang Ausgang	2x 8 galvanisch getrennt ; 12-48V 2x 8 galvanisch getrennt ; 12-48V ;I_{max} 0,35mA
CAN Bus	Baudrate CAN Controller Schnittstellen	10, 20, 50, 100, 125, 250, 500, 800 bzw. 1000 kbps einstellbar über Codierschalter. 1x integriert in CPU 1x interner Bus
EG Richtlinien	RoHS, CE	
Software Schnittstelle		CANopen® DSP401

Beschaltung Digital IO


DOUT1
PWR_OU1
PWR_OU2
DOUT2
IN
OUT
DIN1
PWR_IN1
PWR_IN2
DIN2
ERASE

Anhang A. Versionshistorie

Tabelle A.1. Versionshistorie Version history

Version	Datum	Beschreibung
1.0.0	22.04.2010	Initiale Version der AnaGate Motion Hardware Dokumentation
1.0.1	10.02.2011	Erweiterte Version der AnaGate Motion Hardware Dokumentation

Software Schnittstellen

Inhaltsverzeichnis

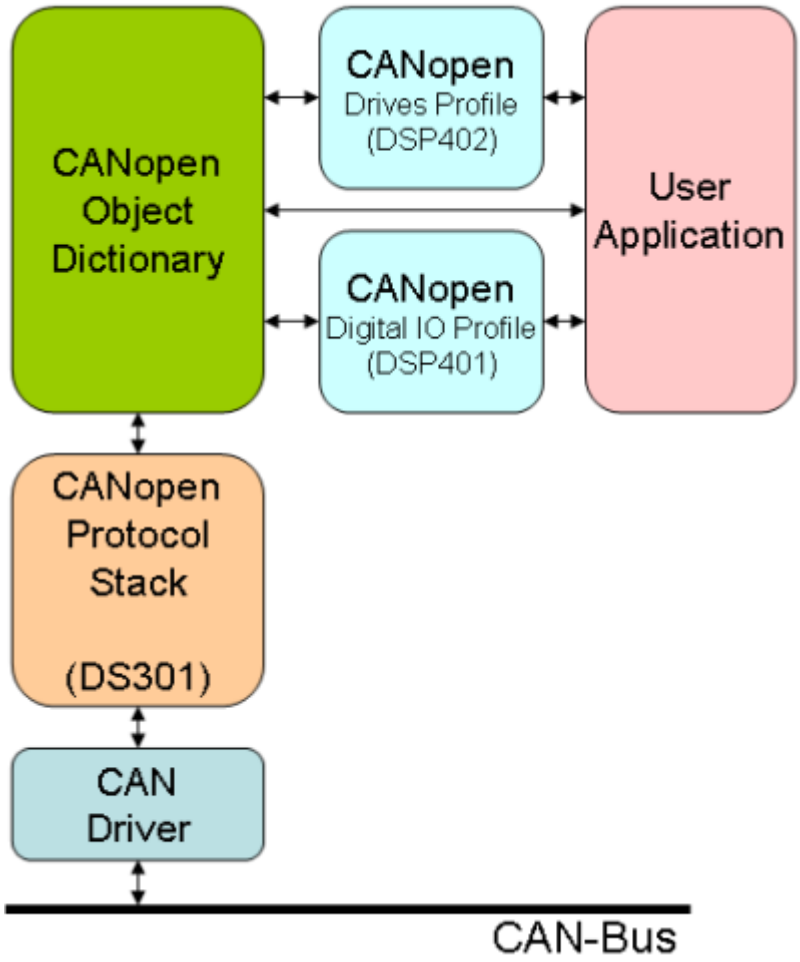
Einführung in CANopen

CANopen ist ein auf CAN basierendes Kommunikationsprotokoll, welches hauptsächlich in der Automatisierungstechnik und zur Vernetzung innerhalb komplexer Geräte verwendet wird.

In diesem Dokument wird nur kurz auf die verschiedenen CANopen Komponenten eingegangen. Weitere tiefergehende Informationen zu den standardisierten Komponenten findet man z.B. unter:

- CiA [http://www.can-cia.org]

Um den schematischen Aufbau des Gesamtaufbaus (CAN-Bus, CANopen-Objektverzeichnis und Geräteprofile) zu verdeutlichen sollte die nachfolgende Grafik einen groben Überblick verschaffen.



Das Protokoll basiert auf:

- dem Kommunikationsprofil

  Spezifiziert die zugrundegelegten Kommunikationsmechanismen.

- den Geräte- oder Applikationsprofilen

  Definition der wichtigsten in der industriellen Automatisierungstechnik eingesetzten Gerätetypen

  - Antriebssteuerungen (z.B. AnaGate Motion "Motor 24Vdc BT Modul")

- digitale und analoge Ein/Ausgabemodule (z.B. AnaGate Motion "Digital IO")
- Regler
- Encoder

Im Geräteprofil werden Funktionalität und Parameter von Standardgeräten des jeweiligen Typs festgelegt. Mit Hilfe dieser standardisierten Profile kann mit derselben Logik auf verschiedene CANopen Geräte zugegriffen werden. Dadurch erreicht man eine weitgehende Produktserien-/Herstellerunabhängigkeit durch die Möglichkeit des Austauschens von Geräten verschiedener Produktserien/Hersteller.

Zentrales Element der CANopen-Standards ist die Beschreibung der Gerätefunktionalität über ein "Objektverzeichnis".

CANopen unterscheidet zwischen zwei Datenübertragungstechniken:

- Den schnellen Austausch kurzer Prozessdaten über sogenannte "Prozessdatenobjekte" (PDOs, Process Data Objects)
- Den Zugriff auf Einträge des Objektverzeichnisses über sog. "Servicedatenobjekte" (SDOs, Service Data Objects)

Innerhalb eines PDOs können maximal 8 Byte Nutzdaten übertragen werden. Prozessdatenobjekte können ereignisorientiert, zyklisch oder auf Anforderung als Broadcastobjekte ohne zusätzlichen Protokolloverhead übertragen werden. In Verbindung mit einer Synchronisationsnachricht kann das Senden sowie die Übernahme von PDOs netzwerkweit synchronisiert werden ("synchrone PDOs"). Die Zuordnung von Anwendungsobjekten auf ein PDO ist über im Objektverzeichnis abgelegte Datenstrukturen ("PDO-Mapping") einstellbar.

Die Übertragung von SDOs erfolgt als bestätigter Dienst mit jeweils zwei CAN-Objekten in Form einer Punkt-zu-Punkt Verbindung zwischen zwei Netzknoten. Die Adressierung des entsprechenden Objektverzeichniseintrages erfolgt durch Angabe von Index und Subindex des Eintrags innerhalb des Datenfeldes der CAN-Nachricht. Über das SDO-Protokoll können Daten unbegrenzter Länge übertragen werden, was allerdings mit einem zusätzlichen Protokolloverhead verbunden ist.

Über die globale Zeit-Nachricht kann die Systemzeit aller CANopen Geräte synchroniosiert werden.

Für die Meldung von Gerätefehlern werden standardisierte ereignisorientierte Alarmnachrichten ("Emergency-Messages") versendet

Zur Kontrolle des CANopen Gerätezustandes steht eine Netzwerkmanagement-Funktionalität (NMT) zur Verfügung. Die Funktionalität der CANopen Geräte wird mit den Diensten "Node-Guarding", "Heartbeat" und "Boot-Up" überwacht.

Allgemein

Funktionscode

Die Definition der Funktion einer Nachricht erfolgt automatisch anhand des CAN-Identifier indem die höchsten 4 Bit des Identifiers als Funktionscode und die restlichen 7 Bits als Node-ID betrachtet werden(siehe [25]). PDO's haben demnach eine höhere Priorität als die SDO's. Die höchste Priorität ist für das Netzwerk-Management (NMT) vorgesehen, mit dem das Netzwerk gestartet und gestoppt wird. Die anderen definierbaren Funktionen, wie EMERGENCY, SYNC usw., sind optional. Ein ‚Minimum Capability Device`, also ein Knoten mit minimalem Funktionsumfang, muß zumindest die Dienste bzw. COB's NMT, SDO und PDO unterstützen.

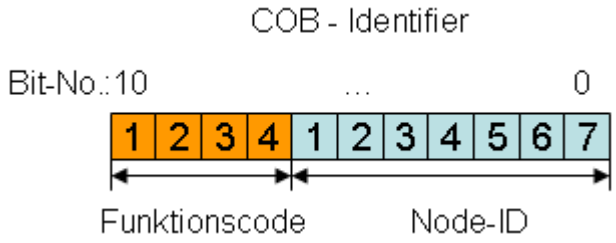


Tabelle 8. Funktionscode

Objekt	Identifier	Kommentar
NMT	0x000	Netzmanagement
SYNC	0x080	globale Sync-Nachricht
TIME STAMP	0x100	globale Zeit-Nachricht
Emergency	0x081 - 0x0FF	Notfall-Nachrichten
T-PDO1	0x181 - 0x1FF	Default Transmit PDO 1
R-PDO1	0x201 - 0x27F	Default Receive PDO 1
T-PDO2	0x281 - 0x2FF	Default Transmit PDO 2
R-PDO2	0x301 - 0x37F	Default Receive PDO 2
T-PDO3	0x381 - 0x3FF	Default Transmit PDO 3
R-PDO3	0x401 - 0x47F	Default Receive PDO 3
T-PDO4	0x481 - 0x4FF	Default Transmit PDO 4
R-PDO4	0x501 - 0x57F	Default Receive PDO 4
Default SDO	0x581 - 0x5FF	Objektverzeichnis-Zugriff (Server -> Client)
Default SDO	0x601 - 0x67F	Objektverzeichnis-Zugriff (Client -> Server)
Default SDO Request	0x6E0	Zur Abstimmung von ggf. mehrfach vorhandenen SDO-Managern
NMT Error Control	0x701 - 0x77F	Fehlersituationen bei Netzmanagement (Node Guarding, Heartbeat und Boot-Up)

Objektverzeichnis

Das Objektverzeichnis beschreibt die komplette Funktionalität der CANopen-Geräte und ist in Tabellenform organisiert. Im Objektverzeichnis sind nicht nur die standardisierten Datentypen und Objekte des CANopen-Kommunikationsprofils sowie der Geräteprofile enthalten, sondern gegebenenfalls auch hersteller-spezifische Objekte und Datentypen. Die Adressierung der Einträge erfolgt mit Hilfe eines 16-Bit-Indizes

(Reihenadresse der Tabelle, maximal 65536 Einträge) und eines 8-Bit-Subindizes (Spaltenadresse der Tabelle, maximal 256 Einträge). Somit lassen sich zusammengehörige Objekte leicht gruppieren. Die Struktur dieses CANopen Objektverzeichnisses ist in der folgenden Tabelle dargestellt.

Tabelle 9. Objektverzeichnis

0x0000	nicht verwendet
0x0001 - 0x001F	statische Datentypen
0x0020 - 0x003F	komplexe Datentypen
0x0040 - 0x005F	herstellerspezifische Datentypen
0x0060 - 0x025F	profilspezifische Datentypen
0x0260 - 0x0FFF	reserviert
0x1000 - 0x1FFF	Kommunikationsprofil (DS-301)
0x2000 - 0x5FFF	herstellerspezifische Parameter
0x6000 - 0x9FFF	Parameter aus den standardisierten Geräteprofilen
0xA000 - 0xAFFF	standardisierte Netzwerkvariablen
0xB000 - 0xBFFF	standardisierte Systemvariablen
0xC000 - 0xFFFF	reserviert

Kommunikationsarten

SDO Service Data Objects

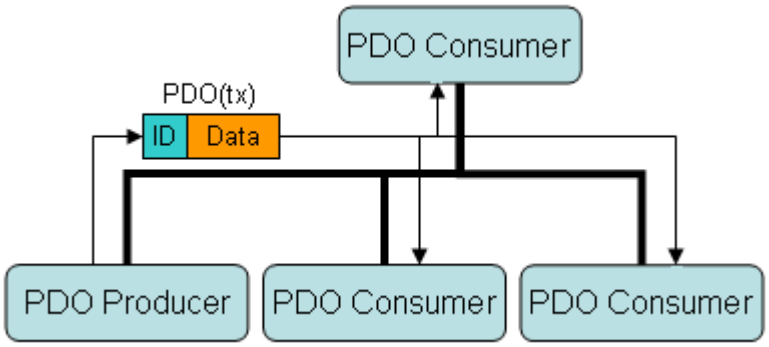
Service Data Objects (SDOs) werden für Änderungen im Objektverzeichnis und für Statusabfragen verwendet. Jedes CANopen Gerät verfügt über mindestens einen SDO Kanal, dem zwei CAN-Identifier zugeordnet sind. SDOs nutzen das "Multiplexed Domain Transfer Protocol" der CAL Spezifikation. Mit diesem Protokoll lassen sich Daten beliebiger Länge übertragen, wobei die Daten gegebenenfalls auf mehrere CANNachrichten aufgeteilt (segmentiert) werden. In der ersten CAN-Nachricht des SDO sind vier der acht verfügbaren Bytes mit Protokollinformationen belegt. Für Zugriffe auf Objektverzeichniseinträge mit bis zu 4 Byte Länge (z.B. Integer16, Integer32, Float) genügt folglich eine einzige CAN-Nachricht (Expedited Transfer). Bei Datenlängen größer 4 Byte erfolgt eine segmentierte Übertragung, bei der alle auf die erste CAN-Nachricht folgenden Segmente des SDO jeweils 7 Byte Nutzdaten enthalten können. Das letzte Segment enthält eine Ende-Kennung. Ein SDO wird immer bestätigt übertragen, daß heißt der Empfang einer jeden Nachricht wird durch den Empfänger quittiert. Mit Einführung der CANopen Spezifikation 4.0 ist auch ein beschleunigter SDO-Transfer möglich. Bei diesem wird nicht mehr jedes Segment bestätigt, sondern es wird nur noch der Empfang einer Gruppe von Segmenten quittiert. Somit erhält man einen enormen Bandbreitenzuwachs für die Übertragung von großen Datenmengen.

PDO Process Data Objects

Für die Übertragung von Prozeßdaten steht der Mechanismus des Process Data Object (PDO) zur Verfügung. Jedes Prozeßdaten produzierende oder konsumierende CANopen-Gerät verfügt deshalb über mindestens ein PDO. PDOs verwenden das "Stored Event Protocol" des CAN Application Layers (siehe [1]). Dabei stehen dem Anwender alle 8 Datenbytes einer CAN-Botschaft zur Verfügung. Die Übertragung einer PDO erfolgt unbestätigt, da letztendlich die CAN-Verbindungsschicht die fehlerfreie Übertragung einer Nachricht sicherstellt. Außerdem sind bestätigte Dienste in zeitkritischen Applikationen nicht wünschenswert, weil sie die Busbandbreite signifikant reduzieren.

Somit sind PDOs "CAN pur", ohne den geringsten Protokoll-Overhead durch CANopen! Die Broadcast-Eigenschaften von CAN bleiben voll erhalten, eine Nachricht kann also von allen Teilnehmern empfangen und ausgewertet werden. Die Kommunikation mit Prozeßdatenobjekten entspricht dem "Producer-Consumer" Modell. Daten werden von ihrem "Produzent" (beispielsweise einer Ein-Ausgabebaugruppe)

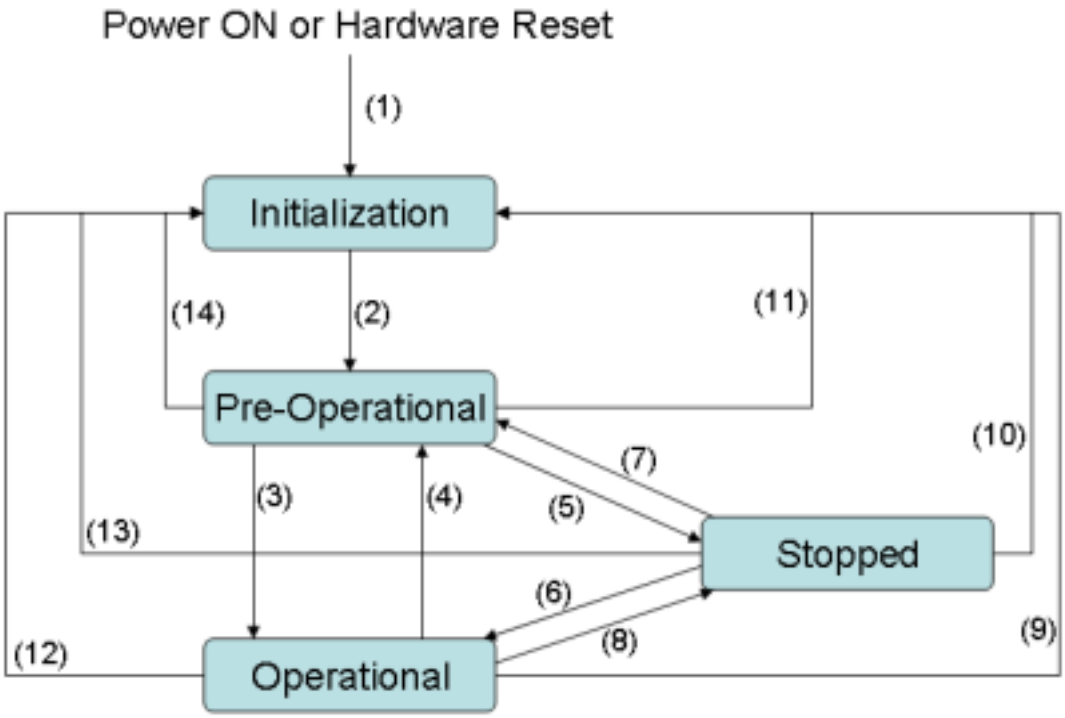
auf den Bus gelegt und von einer beliebigen Anzahl von " Konsumenten" empfangen und verarbeitet (siehe [27]). Das statische Master/Slave - Verhalten der meisten anderen Systeme ist hier nicht vorhanden. Hier liegt einer der entscheidenden Vorteile von CANopen. Die PDOs lassen sich wahlweise ereignisgesteuert oder synchronisiert übertragen, wie es etwa in der Antriebstechnik erforderlich ist. Durch den ereignisgesteuerten Modus kann darauf verzichtet werden, ständig das gesamte Prozeßabbild zu übertragen. Es genügt, die Änderung der Daten zu kommunizieren. Damit lassen sich sowohl die Busbelastung als auch die Reaktionszeit des Netzes auf ein Minimum reduzieren. Es wird dadurch eine hohe Kommunikationsleistung bei vergleichsweiser geringer Baudrate erzielt.



NMT-Dienste

Tabelle 10. NMT-Dienste zur Gerätekontrolle

Kommando	**Beschreibung**	**Zustands-übergang**
-	Automatische Initialisierung nach dem Einschalten	(1)
-	Beendigung der Initialisierung --> PRE-OPERATIONAL	(2)
0x01	*Start Remote Node* Teilnehmer soll in den Zustand OPERATIONAL wechseln und damit den normalen Netzbetrieb starten	(3),(6)
0x02	*Stop Remote Node* Teilnehmer soll in den Zustand STOPPED übergehen und damit seine Kommunikation stoppen. Eine aktive Verbindungsüberwachung bleibt aktiv.	(5),(8)
0x80	*Enter PRE-OPERATIONAL* Teilnehmer soll in den Zustand PRE-OPERATIONAL gehen. Alle Nachrichten außer PDOs können verwendet werden.	(4),(7)
0x81	*Reset Node* Werte der Profilparameter des OV auf Default-Werte setzen. Danach Übergang in den Zustand RESET COMMUNICATION.	(9),(10),(11)
0x82	*Reset Communication* Teilnehmer soll in den Zustand RESET COMMUNICATION gehen und die Werte der Kommunikationsparameter des OV mit Default-Werten laden. Danach Übergang in den Zustand INITIALIZATION, erster Zustand nach Einschalten.	(12),(13),(14)



Emergency Dienst

Nachrichten vom Typ "Emergency" werden verwendet, um Fehler eines Gerätes zu signalisieren. In dem Emergency-Telegramm wird ein Code übertragen, der den Fehler eindeutig identifiziert (definiert im Kommunikationsprofil DS-301 sowie in den jeweiligen Geräteprofilen DSP-40x).

Tabelle 11. Emergency Dienst

Code	Bedeutung
0x00XX	Kein Fehler
0x10XX	Nicht definierter Fehlertyp
0x20XX	Stromfehler
0x30XX	Spannungsfehler
0x40XX	Temperaturfehler
0x50XX	Fehler an der Hardware
0x60XX	Fehler an der Software
0x70XX	Zusatzmodule
0x80XX	Kommunikation
0x90XX	Externer Fehler
0xFF00	Gerätespezifisch

Sync Dienst

Nachrichten vom Typ "Sync" werden verwendet um die Prozessdaten innerhalb des CANopen Netes zu synchronisieren. Die Sync-Nachricht wird vom CANopen Master versendet und je nachdem wie die CANopen Slaves konfiguriert sind senden diese Ihre aktuellen Prozessdaten (PDO´s) auf den CAN-Bus.

Kommunikationsprofil (DS-301)

In diesem Kommunikationsprofil werden die unterstützten Objekteinträge der MotionControl-Serie erklärt.

Objekt 0x1000: Gerätetyp (Device type)

Legt den Gerätetyp fest

Tabelle 12. Objekt: Gerätetyp

Attribute	Wert
Index	0x1000
Subindex	0x00
Name	Gerätetyp
Objektcode	VAR
Datentyp	UNSIGNED32
Zugriff	ro
Wert	0x00
PDO-Mapping	nein

Objekt 0x1001: Fehlerregister (Error register)

Informationen über den Fehlerzustand

Tabelle 13. Objekt: Fehlerregister

Attribute	Wert
Index	0x1001
Subindex	0x00
Name	Fehlerregister
Objektcode	VAR
Datentyp	UNSIGNED8
Zugriff	ro
Wert	Siehe Wertdefinition
PDO-Mapping	nein

Tabelle 14. Struktur des Fehlerregisters

Bit	Bedeutung
0	Allgemeiner Fehler

Objekt 0x1008: Hersteller Gerätebezeichnung

Stellt die Gerätebezeichnung des Moduls bereit

Tabelle 15. Objekt: Hersteller Gerätebezeichnung

Attribute	Wert
Index	0x1008

Attribute	Wert
Subindex	0x00
Name	Hersteller Gerätebezeichnung
Objektcode	VAR
Datentyp	VISIBLE_STRING
Zugriff	ro
PDO-Mapping	nein

Objekt 0x1009: Hersteller Hardwareversion

Stellt die Hardwareversion des Moduls bereit

Tabelle 16. Objekt: Hersteller Hardwareversion

Attribute	Wert
Index	0x1009
Subindex	0x00
Name	Hersteller Hardwareversion
Objektcode	VAR
Datentyp	VISIBLE_STRING
Zugriff	ro
PDO-Mapping	nein

Objekt 0x100A: Hersteller Softwareversion

Stellt die Softwareversion des Moduls bereit

Tabelle 17. Objekt: Hersteller Softwareversion

Attribute	Wert
Index	0x100A
Subindex	0x00
Name	Hersteller Softwareversion
Objektcode	VAR
Datentyp	VISIBLE_STRING
Zugriff	ro
PDO-Mapping	nein

Objekt 0x1018: Identität

Stellt die Identität (AnbieterID, Produktcode und Revisionsnummer) des Moduls bereit

Tabelle 18. Objekt: Anbieter ID

Attribute	Wert
Index	0x1018
Subindex	0x00
Name	Anbieter ID

Attribute	Wert
Objektcode	VAR
Datentyp	UNSIGNED32
Zugriff	ro
PDO-Mapping	nein
Wert	0x2EB (Analytica GmbH)

Tabelle 19. Objekt: Produkt Code

Attribute	Wert
Index	0x1018
Subindex	0x01
Name	Produkt Code
Objektcode	VAR
Datentyp	UNSIGNED32
Zugriff	ro
PDO-Mapping	nein
Wert	• 0x01 BT-Modul • 0x02 BL-Modul • 0x03 400V-Modul • 0x10 DigIO-Modul

Tabelle 20. Objekt: Revisionsnummer

Attribute	Wert
Index	0x1018
Subindex	0x02
Name	Revisionsnummer
Objektcode	VAR
Datentyp	UNSIGNED32
Zugriff	ro
PDO-Mapping	nein

PDO Objekte (Parameter, Mapping)

Die Eigenschaften der PDO's sind spezifisch auf das Geräteprofil angepasst und werden dort erläutert

Geräteprofile

Die CANopen Geräteprofile beschreiben das "was" der Kommunikation. In ihnen sind die Dateninhalte herstellerunabhängig festgelegt, die Grundfunktionalität der entsprechenden Geräteklasse (z.B. Ein-Ausgabebaugruppen, Servoantriebe oder Freqeunzumrichter) läßt sich also einheitlich ansprechen. Damit sind Geräte, die dem gleichen Geräteprofil folgen, untereinander austauschbar. Obwohl bereits viele optionale Parameter in den Profilspezifikationen beschrieben sind, bieten die Geräteprofile zusätzlich Raum für herstellerspezifische Funktionserweiterungen und sind damit "future-proof".

Geräteprofil DSP402

(Drives and motion control)

Dieses Geräteprofil wird für alle "Motor"-Module verwendet.

DSP402 Zustandsmaschine

Ist der Antrieb als CANopen-Slave im Zustand OPERATIONAL, können antriebsspezifische Funktionen angesprochen werden. Der Zustandsautomat [34] gibt einen Überblick über die im Antriebsprofil DSP 402 definierten Zustände und die Übergänge zwischen diesen Zuständen. Der aktuelle Zustand kann dem Zustandswort (ZSW) entnommen werden. Im Zustandsautomat [34] sind die Zustände durch Rechtecke mit gerundeten Ecken dargestellt. Dabei ist jeweils die Binärdarstellung des ZSW angegeben. Ein Bit, welches mit 'x' gekennzeichnet ist, ist für die Zustandsermittlung nicht relevant. Die Zustandsübergänge sind durch Pfeile gekennzeichnet, an denen die Bedingung für den entsprechenden Übergang angegeben ist. In den meisten Fällen handelt es sich dabei um ein Kommando im Steuerwort (STW), welches ebenfalls durch gewisse Bitkombinationen definiert ist. Mit 'x' gekennzeichnete Bits sind dabei nicht relevant. STW.x bezeichnet Bit x des STW, STW.7: 0->1 bedeutet "steigende Flanke STW.7". In einer Fehlersituation gelangt man aus jedem der Zustände im eingezeichneten Rechteck in den Zustand "Fehlerreaktion aktiv".

Neben den Informationen zu Zuständen und Zustandsübergängen enthalten STW und ZSW noch weitere Bedeutungen, die zum Teil von der aktuellen Betriebsart abhängen. Das Steuer- und Zustandswort wird deshalb in den einzelnen Betriebsarten genauer spezifiziert.

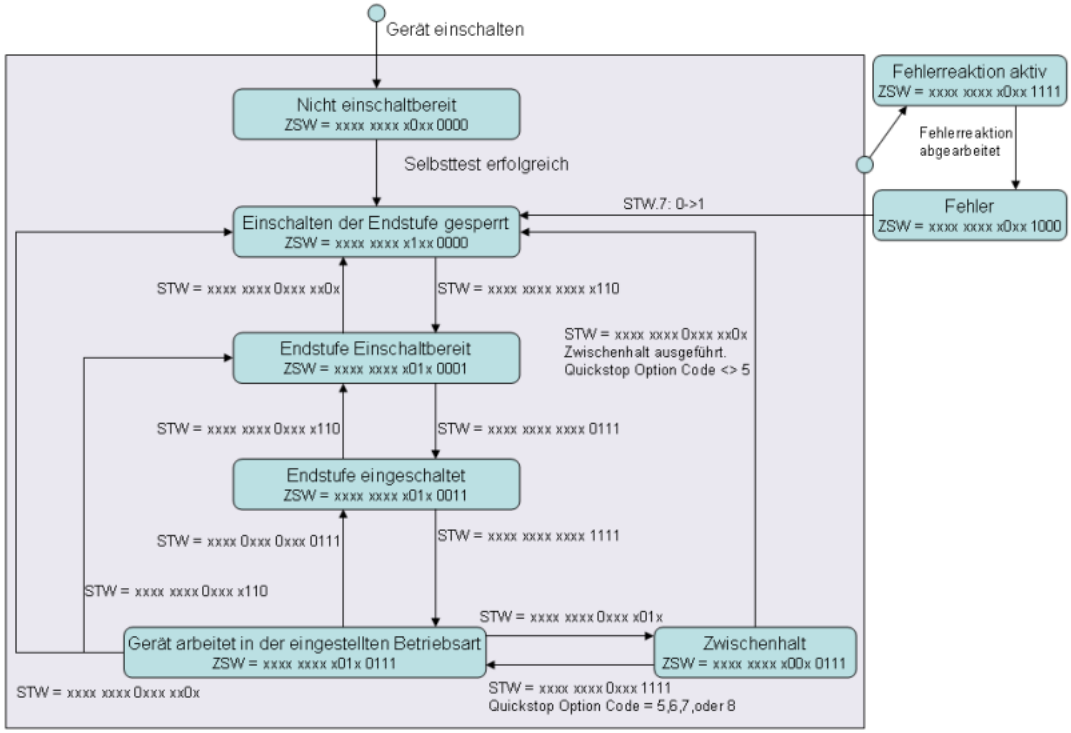
Tabelle 21. DSP402 Zustände

Zustand	Bedingung	Beschreibung
Nicht einschaltbereit (Not ready to switch on)	ZSW = xxxx xxxx x0xx 0000	Der Antrieb befindet sich in der Initialisierungsphase.
Einschaltsperre (Switch on disabled)	ZSW = xxxx xxxx x1xx 0000	Initialisierung des Antriebs beendet. Die Endstufe kann nicht eingeschaltet werden.
Einschaltbereit (Ready to switch on)	ZSW = xxxx xxxx x01x 0001	Die Endstufe kann eingeschaltet werden. Antriebsparameter können geändert werden.
Eingeschaltet (Switched on)	ZSW = xxxx xxxx x01x 0011	Endstufe eingeschaltet.
Betriebsbereit (Operation enabled)	ZSW = xxxx xxxx x01x 0111	Der Antrieb kann gemäß der eingestellten Betriebsart verwendet werden.
Zwischenhalt (Quick stop active)	ZSW = xxxx xxxx x00x 0111	Der Antrieb führt einen Zwischenhalt aus. Ist der Quick Stop Option Code (Objekt 0x605A) 5, bleibt der Antrieb danach in diesem Zustand.
Fehlerreaktion (Fault reaction active)	ZSW = xxxx xxxx x0xx 1111	Es liegt ein Fehler vor. Die Fehlerbehandlung wird ausgeführt.
Fehler (Fault)	ZSW = xxxx xxxx x0xx 1000	Die Fehlerbehandlung ist abgeschlossen.

Der Übergang zwischen diesen Zuständen erfolgt zum Teil über interne Ereignisse, zum Teil über Kommandos, die im Steuerwort übergeben werden:

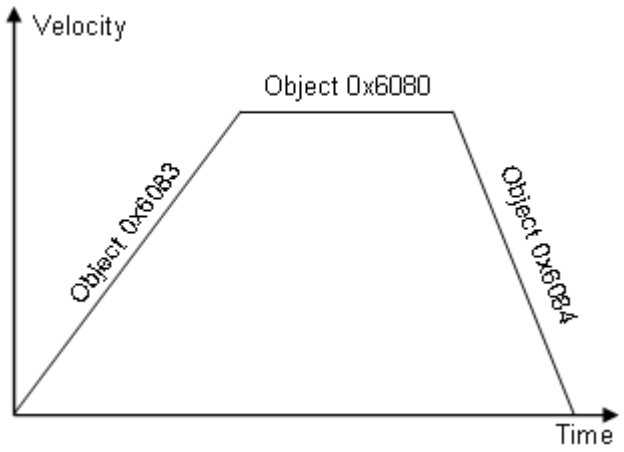
Tabelle 22. DSP402 Zustandsübergänge

Übergang	Ereignis / Kommando	Beschreibung
Start -> Nicht einschaltbereit	Der Antrieb wird eingeschaltet / rückgesetzt	Der Antrieb führt eine Initialisierung durch.
Nicht einschaltbereit -> Einschaltsperre	Die Initialisierung des Antriebs ist beendet.	Die Kommunikation wird aktiviert.
Nicht einschaltbereit -> Einschaltbereit	Der Antrieb empfängt das Kommando "Shutdown": STW = xxxx xxxx 0xxx x110	
Einschaltbereit -> Eingeschaltet	Der Antrieb empfängt das Kommando "Switch On": STW = xxxx xxxx 0xxx 0111	Die Endstufe wird freigegeben
Eingeschaltet -> Betriebsbereit	Der Antrieb empfängt das Kommando "Enable Operation": STW = xxxx xxxx 0xxx 1111	
Betriebsbereit -> Eingeschaltet	Der Antrieb empfängt das Kommando "Disable Operation": STW = xxxx xxxx 0xxx 0111	
Eingeschaltet -> Einschaltbereit	Der Antrieb empfängt das Kommando "Shutdown": STW = xxxx xxxx 0xxx x110	Die Antriebseinheit wird deaktiviert.
Einschaltbereit -> Einschaltsperre	Der Antrieb empfängt das Kommando "Quick Stop": STW = xxxx xxxx 0xxx x01x	
Betriebsbereit -> Einschaltbereit	Der Antrieb empfängt das Kommando "Shutdown": STW = xxxx xxxx 0xxx x110	Die Antriebseinheit wird abgeschaltet.
Betriebsbereit -> Einschaltsperre	Der Antrieb empfängt das Kommando "Disable Voltage": STW = xxxx xxxx 0xxx xx0x	Die Antriebseinheit wird abgeschaltet. Befindet sie sich in Bewegung, wird sie schnellstmöglich angehalten und anschließend deaktiviert.
Eingeschaltet -> Einschaltsperre	Der Antrieb empfängt das Kommando "Disable Voltage": STW = xxxx xxxx 0xxx xx0x oder das Kommando "Quick Stop" STW = xxxx xxxx 0xxx x01x	Die Antriebseinheit wird abgeschaltet.
Betriebsbereit -> Zwischenhalt aktiv	Der Antrieb empfängt das Kommando "Quick Stop": STW = xxxx xxxx 0xxx x01x	Die Antriebseinheit wird gestoppt.
Zwischenhalt -> Einschaltsperre	Der Zwischenhalt ist ausgeführt, oder es wird das Kommando "Disable Voltage" empfangen: STW = xxxx xxxx 0xxx xx0x	Die Antriebseinheit wird dann schnellstmöglich gestoppt und anschließend abgeschaltet.
Aus jedem Zustand -> Fehlerreaktion	Im Antrieb ist ein Fehler aufgetreten.	Die entsprechende Fehlerbehandlung wird ausgeführt.
Fehlerreaktion -> Fehler	Die Fehlerbehandlung ist beendet.	
Fehler -> Einschaltsperre	Der Fehler wurde mit einer steigenden Flanke im Bit 7 des STW quittiert.	



DSP402 Betriebsart: Geschwindigkeitsrampe

In der Betriebsart Geschwindigkeitsrampe kann die vorgegebene Geschwindigkeit mit einer Rampe (lineare Steigerung der Geschwindigkeit) angefahren werden. Dabei wird die Steigung der Rampe durch die zwei Parameter "Beschleunigung" bzw "Bremsbeschleunigung" angegeben. Beim Setzen einer neuen Geschwindigkeit, die Größer als die zuvor eingestellte Geschwindigkeit ist, wird der Parameter "Beschleunigung" benutzt um die eingestellte Geschwindigkeit zu erreichen. Im anderen Fall wenn die neue Geschwindigkeit niegriger ist wird der Parameter "Bremsbeschleunigung" benutzt



DSP402 Steuerwort (STW)

Durch das Steuerwort des Geräteprofiles DSP402 wird der interne Zustandsautomat programmiert. Dabei hat jeder Motor, bei Modulen mit mehr als einem Motoranschluß, seinen eigenen Zustandsautomat die jeweils in unterschiedlichen Zuständen sein können.

Tabelle 23. DSP402 Steuerwort (STW)

Bit	Steuerwort
0	Betriebsbereit schalten
1	Spannung freischalten
2	Quick stop
3	Betriebsart ausführen
4-6	nicht benutzt
7	Übergang 0->1: Fehler rücksetzen
8	Halt
9-15	nicht benutzt

DSP402 Zustandswort (ZSW)

Durch das Zustandswort des Geräteprofiles DSP402 wird der aktuelle interne Zustand abgebildet. Dabei hat jeder Motor, bei Modulen mit mehr als einem Motoranschluß, seinen eigenen Zustand die jeweils in unterschiedlichen sein können.

Tabelle 24. DSP402 Zustandswort (ZSW)

Bit	Zustandswort
0	Einschaltbereit
1	Betriebsbereit
2	Betriebsart aktiviert
3	Fehler liegt vor
4	Spannung eingeschaltet
5	Zwischenhalt aktiv
6	Nicht betriebsbereit
7	Warnung liegt vor
8-9	nicht benutzt
10	Zielgeschwindigkeit erreicht
11-15	nicht benutzt

Parameter der Geschwindigkeitsrampe im Objektverzeichnis

Die nachfolgend angegebenen Einträge des Objektverzeichnises beziehen sich immer auf den 1. Motor. Bei Modulen mit mehreren Motoranschlüssen muß für jeden weiteren Motor der Offset 0x800 (hex) dazuaddiert werden.

Beispiele:

Motorgeschwindigkeit 2. Motor — 0x6080 + 0x800 = 0x6880 (Adresse der Motorgeschwindigkeit des 2. Motors)

beschleunigung 3. Motor — 0x6083 + (2 * 0x800) = 0x7083 (Adresse der Beschleunigung des 3. Motors)

Tabelle 25. DSP402 Parameter der Geschwindigkeitsrampe

Index	Bedeutung
0x6040	Steuerwort

Index	Bedeutung
0x6041	Zustandswort
0x607E	Richtungsumkehr
0x6080	Motorgeschwindigkeit
0x6083	Beschleunigung
0x6084	Bremsbeschleunigung

Objekt 0x6040: Steuerwort

In diesem Objekt werden Kommandos an die Antriebssteuerung übermittelt

Tabelle 26. DSP402 Objekt: Steuerwort

Attribute	Wert
Index	0x6040
Subindex	0x00
Name	Steuerwort
Objektcode	VAR
Datentyp	UNSIGNED16
Zugriff	rw
Standardwert	0x00
Min	0x00
Max	0xFFFF
PDO-Mapping	ja
Beschreibung	siehe

Objekt 0x6041: Zustandswort

In diesem Objekt überträgt die Antriebssteuerung Zustandsinformationen

Tabelle 27. DSP402 Objekt: Zustandswort

Attribute	Wert
Index	0x6041
Subindex	0x00
Name	Zustandswort
Objektcode	VAR
Datentyp	UNSIGNED16
Zugriff	ro
Standardwert	0x00
Min	0x00
Max	0xFFFF
PDO-Mapping	ja
Beschreibung	siehe

Objekt 0x607E: Richtungsumkehr

Legt die Polarität für die Drehbewegung des Motors fest

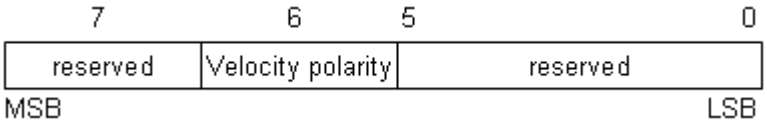


Tabelle 28. DSP402 Objekt: Polarität

Attribute	Wert
Index	0x607E
Subindex	0x00
Name	Polarity
Objektcode	VAR
Datentyp	UNSIGNED8
Zugriff	rw
Standardwert	0x00
PDO-Mapping	nein
Beschreibung	Bit6: • 0: Richtungsumkehr nicht eingeschaltet. Standardrichtung des Motors. • 1: Richtungsumkehr eingeschaltet. Polarität des Motors ist vertauscht. Dreht entgegengesetzt zur Standardrichtung.

Objekt 0x6080: Motorgeschwindigkeit

Legt die Geschwindigkeit des Motors fest

Tabelle 29. DSP402 Objekt: Motorgeschwindigkeit

Attribute	Wert
Index	0x6080
Subindex	0x00
Name	Motorgeschwindigkeit
Objektcode	VAR
Datentyp	UNSIGNED32
Zugriff	rw
Standardwert	0x00
Min	0x00
Max	0x03E8
PDO-Mapping	nein
Beschreibung	Dieser Parameter gibt in Promille die neue Endgeschwindigkeit an. Je nach dem ob diese neue Endgeschwindigkeit größer oder kleiner als die vorherige ist wird die Rampe mit dem Wert der "Beschleunigung" oder "Bremsbeschleunigung" angefahren.

Objekt 0x6083: Beschleunigung

Legt die Beschleunigung des Motors fest

Tabelle 30. DSP402 Objekt: Beschleunigung

Attribute	Wert
Index	0x6083
Subindex	0x00
Name	Beschleunigung
Objektcode	VAR
Datentyp	UNSIGNED32
Zugriff	rw
Standardwert	0x00
Min	0x00
Max	0xFFFFFFFF
PDO-Mapping	nein
Beschreibung	Dieser Parameter gibt die Steigung (Beschleunigung) für die Rampe an. Die Einheit für den Parameter sind Millisekunden, die für eine Rampe von 0‰-1000‰ Geschwindigkeit, benötigt werden. Daraus ergibt sich, daß bei Angabe des Wertes 0 keine Rampe abgefahren, sondern die Endgeschwindigkeit sofort gesetzt wird. Bei Angabe des Wertes 1000(0x03E8) werden die kompletten 1000‰ innerhalb einer Sekunde abgefahren und bei Verwendung des maximalen Wertes von 0xFFFFFFFF währe die langsamste Rampe von ca. 49 Tagen erreicht.

Objekt 0x6084: Bremsbeschleunigung (Verzögerung)

Legt die Bremsbeschleunigung (Verzögerung) des Motors fest

Tabelle 31. DSP402 Objekt: Beschleunigung (Verzögerung)

Attribute	Wert
Index	0x6084
Subindex	0x00
Name	Beschleunigung (Verzögerung)
Objektcode	VAR
Datentyp	UNSIGNED32
Zugriff	rw
Standardwert	0x00
Min	0x00
Max	0xFFFFFFFF
PDO-Mapping	nein
Beschreibung	Dieser Parameter gibt das Gefälle (Bremsbeschleunigung) für die Rampe an. Die Einheit für den Parameter sind Millisekunden, die für eine Rampe von 0‰-1000‰ Geschwindigkeit, benötigt werden. Daraus ergibt sich, daß bei Angabe des Wertes 0 keine Rampe abgefahren, sondern die Endgeschwindigkeit sofort gesetzt wird. Bei Angabe des Wertes 1000(0x03E8) werden die kompletten 1000‰ innerhalb einer Sekunde abgefahren und bei Verwendung des maximalen Wertes von 0xFFFFFFFF währe die langsamste Rampe von ca. 49 Tagen erreicht.

T-PDO1

Mit Hilfe dieser PDO werden die Statuswörter der bis zu vier implementierten Motorprofile übermittelt. Diese PDO wird bei jeder empfangenen Sync-Nachricht vom CANopen Slave erzeugt und versendet. Ein einzelnes Statuswort umfasst 16Bit (2Byte).

Tabelle 32. DSP402 T-PDO1

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x60410010 Statuswort Motor 1		0x68410010 Statuswort Motor 2		0x70410010 Statuswort Motor 3		0x78410010 Statuswort Motor 4	

T-PDO2

Mit Hilfe dieser PDO werden die Errorcodes der bis zu vier implementierten Motorprofile übermittelt. Diese PDO wird bei jeder empfangenen Sync-Nachricht vom CANopen Slave erzeugt und versendet. Ein einzelner Errorcode umfasst 16Bit (2Byte).

Tabelle 33. DSP402 T-PDO2

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x603F0010 Errorcode Motor 1		0x683F0010 Errorcode Motor 2		0x703F0010 Errorcode Motor 3		0x783F0010 Errorcode Motor 4	

T-PDO3

Mit Hilfe dieser PDO wird der Strom der Motoren in 1/100A übermittelt. Diese PDO wird periodisch alle 100ms versendet.

Tabelle 34. DSP402 T-PDO3

Byte 1	Byte 2	Byte 3	Byte 4
0x60780010 Strommessung Motor 1		0x68780010 Strommessung Motor 2	
Motor 1 Phase U(L1)	Motor 1 Phase W(L3)	Motor 2 Phase U(L1)	Motor 2 Phase W(L3)

T-PDO4

Mit Hilfe dieser PDO werden die aktuellen Werte/Zählerstände der Digitalen Sensor-/Encodereingänge übermittelt. Diese PDO wird bei jeder Änderung eines Digitalen Eingangs vom CANopen Slave erzeugt und versendet.

Tabelle 35. DSP402 T-PDO4

Byte 1								Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	
S1 State	S2 State	E1 State C	E1 State B	E1 State A	E2 State C	E2 State B	E2 State A	E1 Value	E2 Value	E1 Count A	E1 Count B	E2 Count A	E2 Count B	S1 Count (4Bit)	S2 Count (4Bit)

Abkürzungen der Liste: S1:Sensor1 / S2:Sensor2 / E1:Encoder1 / E2:Encoder2

Beschreibung:

S1 State: Status des Eingangs Sensor1

S2 State: Status des Eingangs Sensor2

E1 State A: Status des Eingangs Encoder1 Port A

E1 State B: Status des Eingangs Encoder1 Port B

E1 State C: Status des Eingangs Encoder1 Port C

E2 State A: Status des Eingangs Encoder2 Port A

E2 State B: Status des Eingangs Encoder2 Port B

E2 State C: Status des Eingangs Encoder2 Port C

E1 Value: Wert des Encoder1-Eingangs. Wird hoch/heruntergezählt je nachdem in welcher Richtung sich z.B. der Motor dreht.

E2 Value: Wert des Encoder1-Eingangs. Wird hoch/heruntergezählt je nachdem in welcher Richtung sich z.B. der Motor dreht.

E1 Count A: Anzahl der positiven Flankenwechsel am Port A des Encoder 1.

E1 Count B: Anzahl der positiven Flankenwechsel am Port B des Encoder 1.

E2 Count A: Anzahl der positiven Flankenwechsel am Port A des Encoder 2.

E2 Count B: Anzahl der positiven Flankenwechsel am Port B des Encoder 2.

S1 Count: Anzahl der positiven Flankenwechsel am Sensor 1.

S2 Count: Anzahl der positiven Flankenwechsel am Sensor 2.

R-PDO1

Mit Hilfe dieser PDO werden die Zustandsmaschinen der einzelnen Motoren in die jeweiligen Zustände versetzt. Diese PDO wird vom CANopen Master erzeugt und an den CANopen Slave mit der angegebenen Node-ID gesendet.

Tabelle 36. DSP402 R-PDO1

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x60400010 Controlword Motor 1		0x68400010 Controlword Motor 2		0x70400010 Controlword Motor 3		0x78400010 Controlword Motor 4	

Geschwindigkeitsrampe ausführen

Geräteprofil DSP401

(Generic I/O modules)

Dieses Geräteprofil wird für das "DigitalIO"-Modul verwendet.

Objekt 0x8000: Digital Input

Beinhaltet den aktuellen Status der 16 Digitalen Eingänge

Tabelle 37. DSP401 Objekt: Digital Input

Attribute	Wert
Index	0x8000

Attribute	Wert
Subindex	0x00 ; 0x01
Name	Digital Input
Objektcode	VAR
Datentyp	UNSIGNED8
Zugriff	ro
Standardwert	Wert der Eingänge
PDO-Mapping	ja (T-PDO4)
Beschreibung	Subindex: • 0x00: LSB - untere 8Bits (DIN1). • 0x01: MSB - obere 8Bits (DIN2).

Objekt 0x8001: Digital Output

Beinhaltet den aktuellen Status der 16 Digitalen Ausgänge

Tabelle 38. DSP401 Objekt: Digital Output

Attribute	Wert
Index	0x8001
Subindex	0x00 ; 0x01
Name	Digital Output
Objektcode	VAR
Datentyp	UNSIGNED8
Zugriff	rw
Standardwert	0x00
PDO-Mapping	ja (R-PDO4)
Beschreibung	Subindex: • 0x00: LSB - untere 8Bits (DOUT1). • 0x01: MSB - obere 8Bits (DOUT2).

T-PDO4

Mit Hilfe dieser PDO werden die Zustände der 16 Digitalen Eingänge übermittelt. Diese PDO wird bei jeder Änderung eines Digitalen Eingangs vom CANopen Slave erzeugt und versendet.

Tabelle 39. DSP401 T-PDO4

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Digital Input LSB (DIN1)	Digital Input MSB (DIN2)						

R-PDO3

Mit Hilfe dieser PDO werden die 16 Digitalen Ausgänge mit den aktuellen Daten gesetzt. Diese PDO wird vom CANopen Master erzeugt und an den CANopen Slave mit der angegebenen Node-ID gesendet.

Tabelle 40. DSP401 R-PDO3

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Digital Output LSB (DOUT1)	Digital Output MSB (DOUT2)						

Gerätebeschreibung - EDS und DCF

Das "Electronic Data Sheet" (EDS) beschreibt die Funktionalität eines CANopen-Gerätes in maschinenlesbarer Form. In dem EDS sind alle Objekte aufgeführt, die unterstützten Baudraten, der Hersteller und viele weitere Angaben. Die EDS ist jedoch nur eine Schablone für das Gerät, denn es sind nicht die Werte eines Objektes enthalten.

Das "Device Configuration File" (DCF) ist identisch zur EDS aufgebaut und enthält zusätzlich die Werte eines jeden Objektes.

Das Format der beiden Beschreibungsdateien ist an das *.ini-Format von Windows angelehnt. Für den Anwender ist die Einbindung eines neuen Gerätes in das Netzwerk recht einfach: Hardware anschliessen und die Diskette mit der zugehörigen EDS- oder DCF-Datei in den Master einlesen. Somit ist die Betriebsbereitschaft hergestellt.

AnaGate Motion -> CANOpen

CANopen Schnittstelle für die verschiedenen AnaGate Motion Module

- Motor 24Vdc BL (Brushless)
- Motor 24Vdc BT (Brushtype)
- Motor 400Vac
- Digital IO Modul

Motor 24Vdc BL (Brushless)

Dieses Modul verwendet das CANopen Geräteprofil DSP402.

Motor 24Vdc BT (Brushtype)

Dieses Modul verwendet das CANopen Geräteprofil DSP402.

Motor 400Vac

Dieses Modul verwendet das CANopen Geräteprofil DSP402.

Digital IO Modul

Dieses Modul verwendet das CANopen Geräteprofil DSP401.

Beispiel CANopen - Lua

In diesem Beispiel wird eine Verbindung zu einem AnaGate Motion Control Modul aufgebaut. Dieses Control Modul ist über den internen Bus mit einem AnaGate Motion "Motor"-Modul verbunden. Sollte die Verbindung fehlschlagen wird eine Fehlermeldung ausgegeben und das Script beendet. Bei erfolgreicher Verbindung wird der Motor mit 50% der maximalen Geschwindigkeit gestartet und wieder gestoppt.

- Verbindung zu dem AnaGate Motion Control Modul aufbauen
- CANopen-Funktionalität auf AnaGate Motion Control Modul einschalten
- CANopen Slave mit der NodeID 0x02 in Status "OPERATIONAL" setzen
- Zustandsautomat 1.Motors im CANopen Slave mit der NodeID 0x02 in den Status "Betriebsbereit" schalten
- Geschwindigkeit ohne Rampe auf 50% setzen
- Pause
- Rampe für abschalten setzen.
- Geschwindigkeit Rampe auf 0% setzen.
- Verbindung beenden

Beispiel 1. CANopen- LUA Skriptbeispiel

```
--************************************************
function printf(...)
   io.write(string.format(...))
   io.flush();
end
--************************************************
function main()
   nNodeID = 0x02;

   -- Open
   nRC, nHandle = LS_CANOpenDevice( false, true, 5001, "10.1.2.160", 5000 );
   if (nRC ~= 0) then
      print(LS_CANErrorMessage(nRC));
      exit();
   end;

   -- CANopen-Funktionalität auf AnaGate Motion Control Modul einschalten
   nRC = LS_CANopenSetConfig(nHandle, 1);

   -- CANopen Slave mit der NodeID 0x02 in Status "OPERATIONAL" setzen
   nRC = LS_CANopenSendNMT(nHandle, nNodeID, ENMTType.NMT_TYPE_OPERATIONAL);
   LS_Sleep(20);

   -- Zustandsautomat in Status "Einschaltbereit" setzen. Funktionscode 4=R-PDO1
   nRC = LS_CANopenSendPDO(nHandle, nNodeID, 4, 2, {0x06, 0x00});
   LS_Sleep(20);

   -- Zustandsautomat in Status "Eingeschaltet" setzen. Funktionscode 4=R-PDO1
   nRC = LS_CANopenSendPDO(nHandle, nNodeID, 4, 2, {0x07, 0x00});
   LS_Sleep(20);

   -- Zustandsautomat in Status "Betriebsbereit" setzen. Funktionscode 4=R-PDO1
   nRC = LS_CANopenSendPDO(nHandle, nNodeID, 4, 2, {0x0F, 0x00});
   LS_Sleep(20);


   -- Geschwindigkeit auf 50% setzen
   LS_CANopenSendSDOWrite(nHandle, nNodeID, ESDOType.SDO_TYPE_WRITE_4,
                          0x6080, 0x00, 0xF4010000);

   -- Pause
   LS_Sleep(5000);

   -- Bremsbeschleunigung setzen
   LS_CANopenSendSDOWrite(nHandle, nNodeID, ESDOType.SDO_TYPE_WRITE_4,
                          0x6084, 0x00, 0xE8030000);

   -- Geschwindigkeit auf 0% setzen - Motor sollte innerhalb 500ms stehen
   LS_CANopenSendSDOWrite(nHandle, nNodeID, ESDOType.SDO_TYPE_WRITE_4,
                          0x6080, 0x00, 0x00000000);

   -- Verbindungen beenden
   LS_CANCloseDevice(nHandle);
end;
```

Anhang B. Begriffe / Abkürzungen

CAN	Controller Area Network (CAN) is a serial bus system originally developed by the Robert Bosch GmbH. It is internationally standardized by ISO 11898-1. CAN has been implemented by many semiconductor manufacturers.
CANopen	Family of profiles for embedded networking in industrial machinery, medical equipment, building automation (e.g. lift control systems, electronically controlled doors, integrated room control systems), railways, maritime electronics, truck-based superstructures, off-highway and off-road vehicles, etc.
CiA	CAN in Automation e.V. [http://www.can-cia.org]
COB-ID	The COB-ID is the object specifying the CAN identifier and additional parameters (valid/-invalid bit, remote frame support bit, frame format bit) for the related communication object.
COB (communication object)	A communication object consists of one or more CAN messages with a specific functionality, e.g. PDO, SDO, Emergency, Time, or Error Control.
communication profile	A communication profile defines the content of communication objects such as Emergency, Time, Sync, Heartbeat, NMT, etc. in CANopen.
CRC (cyclic redundancy check)	The cyclic redundancy check (CRC) is performed by a polynomial implemented in the transmitting as well as in the receiving CAN modules. The cyclic redundancy check in the CAN data frame and CAN remote frame is a number derived from, and stored or transmitted with, a block of data in order to detect corruption. By recalculating the CRC and comparing it to the value originally transmitted, the receiver can detect some types of transmission errors.
device profile	A device profile defines the device-specific application data and communication capability based on the related higher-layer protocol. For more complex devices these profiles may provide a finite state automaton (FSA), which enables standardized device control.
EDS (electronic data sheet)	Electronic data sheets describe the functionality of a device in a standardized manner. CANopen and DeviceNet use different EDS formats.
EMCY (emergency)	Pre-defined communication service in CANopen mapped into a single 8-byte data frame containing a 2-byte standardized error code, the 1-byte error register, and 5-byte manufacturer-specific information. It is used to communicate device and application failures.
FSA (finite state automaton)	An FSA is an abstraction to describe the behavior of a black box. It is composed of a several states, transitions between those states, and actions.
function code	First four bits of the CAN identifier in the CANopen pre-defined identifier set indicating the function of the communication object (e.g. TPDO_1 or error control message).
galvanic isolation	Galvanic isolation in CAN networks is performed by optocouplers or transformers placed between CAN controller and CAN transceiver chip.

gateway	Device with at least two network interfaces transforming all seven OSI (open system interconnection) protocol layers, e.g. CANopen-to-Ethernet gateway or CANopen-to-DeviceNet gateway.
heartbeat	CANopen and DeviceNet use the heartbeat message to indicate that a node is still alive. This message is transmitted periodically.
index	16-bit address to access information in the CANopen object dictionary; for array and records the address is extended by an 8-bit subindex.
NMT (network management)	Entity responsible for the network boot-up procedure and the optional configuration of nodes. It also may include node-supervising functions such as node guarding.
node	Assembly, linked to the CAN network, capable of communicating across the network according to the CAN protocols.
node guarding	Mechanism used in CANopen and CAL to detect bus-off or disconnected devices, which is part of the error control mechanisms. The NMT master sends a remote frame to the NMT slave that is answered by the corresponding error control message.
Node-ID	Unique identifier for a device required by different CAN-based higher-layer protocols in order to assign CAN identifiers to this device, e.g. in CANopen or DeviceNet. Using the pre-defined connection sets of CANopen or DeviceNet, the node-ID is part of the CAN identifier.
object dictionary	The object dictionary is the heart of any CANopen device. It enables access to all data types used in the device, to the communication parameters, as well as to the process data and configuration parameters.
PDO (process data object)	Communication object defined by the PDO communication parameter and PDO mapping parameter objects. It is an unconfirmed communication service without protocol overhead.
PDO mapping	In PDOs, there may be mapped up to 64 objects. The PDO mapping is described in the PDO mapping parameters.
ro	readonly: Objektverzeichniseintrag kann nur gelesen werden.
rpm	Umdrehungen pro Minute.
rw	readwrite: Objektverzeichniseintrag kann gelesen und geschrieben werden.
SDO (service data object)	The SDO is a confirmed communication service that provides access to all entries in the CANopen object dictionary. An SDO uses two 8-byte CAN messages with different identifiers. The SDO may transmit segmented any amount of data. Each segment (segmented SDO) or a number of segments is confirmed (SDO block transfer).
SDO block transfer	SDO block transfer is a CANopen communication service for increasing the speed of downloading data from the CANopen device. In SDO block transfer, the confirmation is sent after the reception of a number of SDO segments.
sec	Sekunden.
SPS (PLC)	Speicherprogrammierbare Steuerung

STW	Steuerwort
subindex	8-bit sub-address to access the sub-objects of arrays and records in a CANopen object dictionary.
SYNC message	Dedicated CANopen message forcing the receiving nodes to sample the inputs mapped into synchronous TPDOs. Receiving this message causes the node to set the outputs to values received in the previous synchronous RPDO.
TIME message	Standardized message in CANopen containing the time as a 6-byte value given as ms after midnight and days after 1st January 1984.
ZSW	Zustandswort

Anhang C. Versionshistorie CANopen

Tabelle C.1. Versionshistorie CANopen Version history CANopen

Version	Datum	Beschreibung
1.0.0	22.04.2010	Initiale Version der CANopen Dokumentation
1.0.1	10.02.2011	Erweiterte Version der CANopen Dokumentation