

---

# **MBS-SERVO-Verstärker**

**Technische Dokumentation**

**L. Egloff**

**2.9.96**

**Version: 1.50**

# Inhaltsverzeichnis

<b>1. ALLGEMEINES</b>	<b>4</b>
<b>2. POSITIONSREGLER</b>	<b>5</b>
<b>2.1. Blockschema</b>	<b>5</b>
<b>2.2. Technische Daten</b>	<b>5</b>
2.2.1. Hardware	5
2.2.2. Software	6
<b>2.3. Detailbeschreibung</b>	<b>8</b>
2.3.1. Ablaufsteuerung	8
2.3.2. Sollwertgenerator	11
2.3.3. Konvertierung der Positionsabweichung	14
2.3.4. Integrator	14
2.3.5. Fuzzy-Controller	15
2.3.6. 15-Bit PWM-Ausgang aus 2*8-Bit	22
2.3.7. Service-Software FuzzyProg Servo für Windows 3.x	22
<b>3. CAN-BUS-KOMMUNIKATION</b>	<b>23</b>
<b>3.1. CAN Schnittstelle</b>	<b>23</b>
<b>3.2. Aufbau CAN-Meldung</b>	<b>23</b>
<b>3.3. Murrelektronik MBS-Protokoll (nicht implementiert)</b>	<b>24</b>
3.3.1. Allgemeines	24
3.3.2. Message-Identifizier	24
3.3.3. Data-Bytes	25
3.3.4. Message-Typen	25
3.3.5. Control-Command-Message	26
3.3.6. Request-Command-Messages	27
3.3.7. Data-Messages	29
<b>3.4. Selectron Selecan-DP Protokoll</b>	<b>33</b>
3.4.1. Allgemeines	33
3.4.2. Message-Descriptor	33
3.4.3. Service-Messages	33
3.4.4. Service-Messages (nicht Selectron kompatibel)	37
3.4.5. Data-Messages	42
<b>4. RS232-KOMMUNIKATION</b>	<b>45</b>
<b>4.1. Allgemeines</b>	<b>45</b>
4.1.1. Schnittstelle	45
4.1.2. Telegrammstruktur	45
4.1.3. Read Identifikation	47
4.1.4. Read Status	47
4.1.5. Write Operating-Mode	49
4.1.6. Write Parameter	50
4.1.7. Read Parameter	52



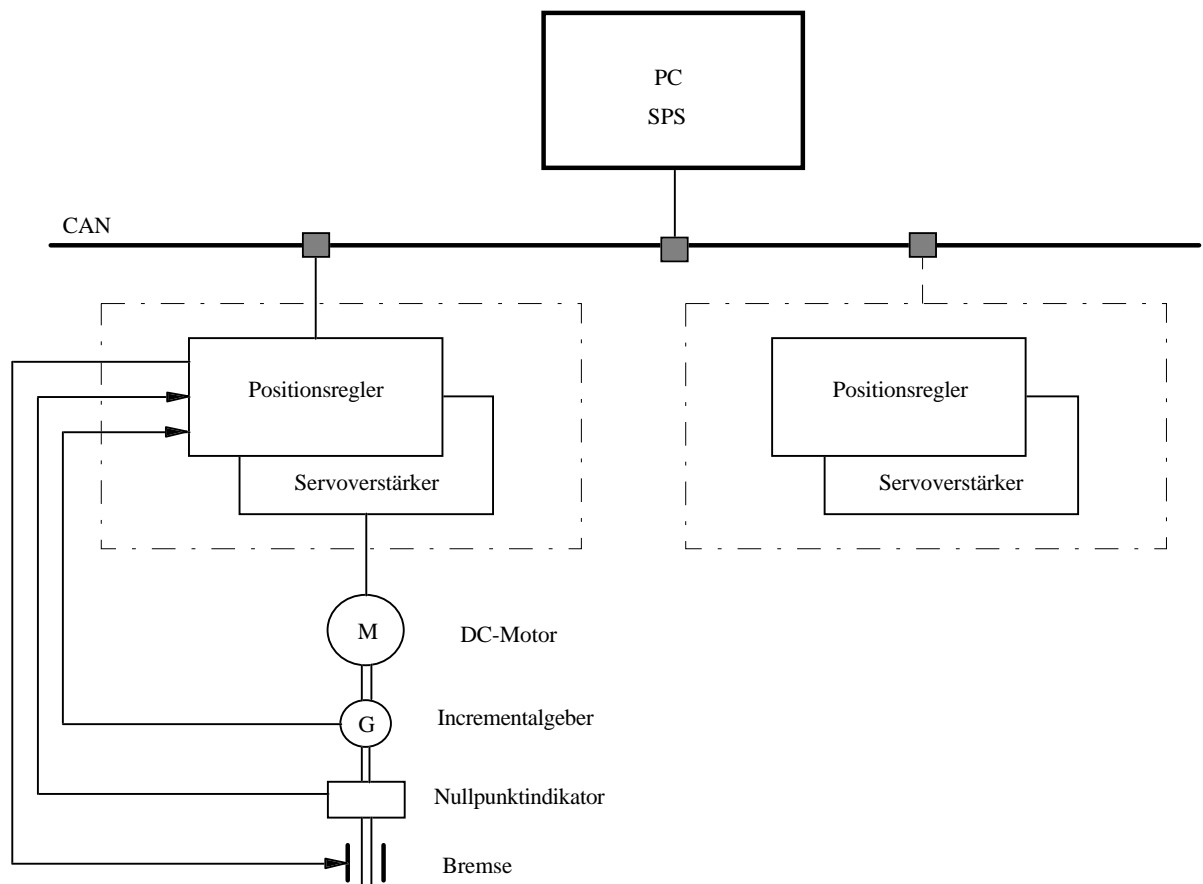
4.1.8. Save Parameter	54
4.1.9. Write Memory-Command	55
4.1.10. Read Memory-Command	56
4.1.11. Read Fuzzy-Status	57
4.1.12. Write Assist Parameter	58
<b>5. SERVO-VERSTÄRKER</b>	<b>59</b>
5.1. Anwendung	59
5.2. Funktion	59
<b>6. INBETRIEBNAHME</b>	<b>60</b>
6.1. Voraussetzungen	60
<b>6.2. Anschlüsse</b>	<b>60</b>
6.2.1. Anschlußbelegung Klemmenblock links	61
6.2.2. Anschlußbelegung Klemmenblock rechts	61
6.2.3. Anschluss Mess-System	61
6.2.4. Anschluss RS232	61
6.2.5. Anschluss CAN	61
6.2.6. Abgriff Strom-Monitor	62
<b>6.3. Einstellungen am Modul</b>	<b>62</b>
6.3.1. Potentiometer Speed Geschwindigkeitsbegrenzung	62
6.3.2. Potentiometer Gain Regelverstärkung	62
6.3.3. Potentiometer Current Strombegrenzung	63
6.3.4. Potentiometer DIG-S Speed mit Inkrementalgeber	63
<b>6.4. Diagnose LED</b>	<b>63</b>
6.4.1. IN 1 gelb	63
6.4.2. IN 2 gelb	63
6.4.3. Endschalter Minus gelb	63
6.4.4. Endschalter PLUS gelb	63
6.4.5. OUT 1 gelb	63
6.4.6. OUT 2 grün	63
6.4.7. NULL grün	63
6.4.8. RUN grün	63
6.4.9. Übertemperatur (Temp) rot	64
6.4.10. Rücksetzung durch ENABLE. Kurzschluss (Curr) rot	64
6.4.11. ACTIV grün	64
6.4.12. POWER grün	64
<b>6.5. Strommonitor</b>	<b>64</b>

# 1. Allgemeines

Der **Fuzzy-Positionsregler** ist ein Einachspositionsregler für Servoantriebe, der mit Fuzzy-Control realisiert wurde.

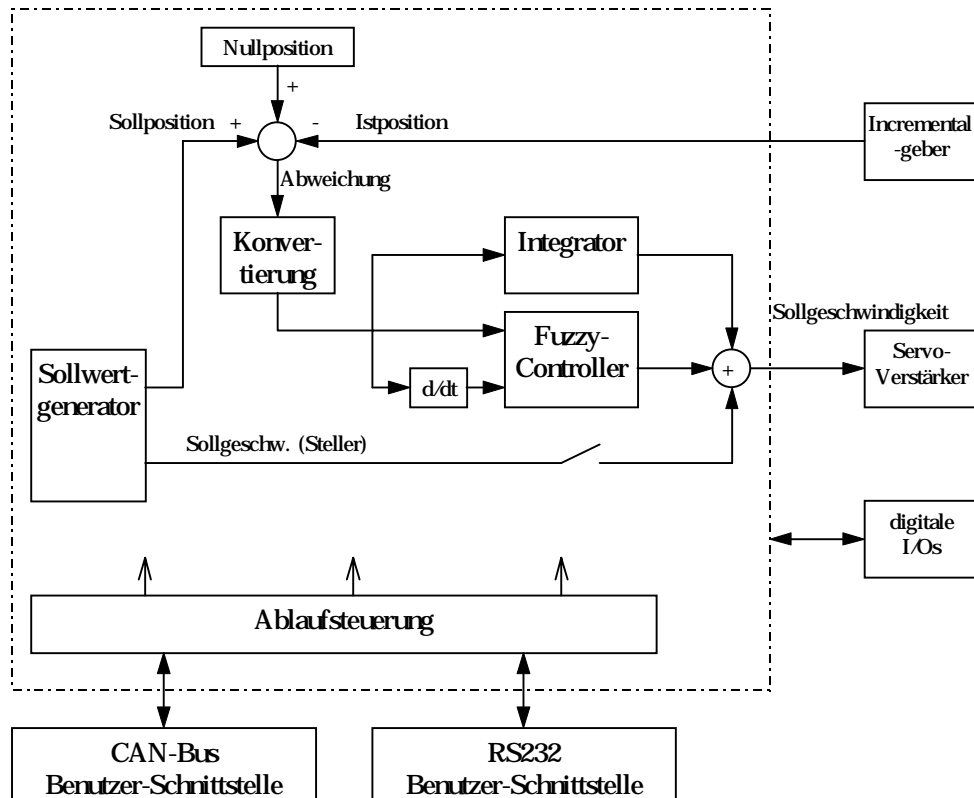
Das Antriebssystem besteht aus dem Fuzzy-Positionsregler und dem Leistungsteil. Die Hardware und Software des Fuzzy-Positionsreglers sind allgemein aufgebaut und können somit für verschiedene Antriebssysteme (Elektro, Hydraulik und Pneumatik) verwendet werden.

Über die integrierte Feldbus-Schnittstelle (CAN-Bus) können mehrere Fuzzy-Positionsregler zu einem Mehrachssystem zusammengeschaltet werden.



## 2. Positionsregler

### 2.1. Blockschema



### 2.2. Technische Daten

#### 2.2.1. Hardware

- Prozessoreinheit:
  - 80C592 Mikroprozessor, 16MHz
  - 64 kByte EPROM für Betriebssystem
  - 32 kByte RAM mit Batteriepufferung für Daten und für die Speicherung von Fahrbefehlen (max. 1000 Stück)
  - 256 Byte EEPROM für Systemkonfiguration und Fuzzy-Controller Parameter
- Incrementalgeberinterface :
  - Stromversorgung 5V für Inkrementalgeber
  - RS485 Empfänger für Signale A, iA, B, iB, C, iC
  - 4 fach Signalauswertung mit Richtungsdiskriminator
  - 14 Bit Vor-/Rückwärtszähler



- Digitaleingänge:
  - Nullpunktindikator, extern, 24VDC
  - Antriebseinheit-Freigabe, extern, 24VDC
  - Endschalter-CW, extern, 24VDC
  - Endschalter-CCW, extern, 24VDC
  - Fehler-Leistungsteil, intern, TTL-Signal
- Digitale Ausgänge:
  - Bremse auf/zu, extern, 24VDC/0,5A
  - Fehler-Antriebseinheit, 24VDC/0,5A
  - Freigabe-Servoverstärker, intern, TTL-Signal
- Sollgeschwindigkeitsausgabe:
  - 2 PWM TTL-Signale (PWM und iPWM)
  - 15-Bit Auflösung durch Mischen von zwei PWM-Signalen (+/- 16383 Digit)
- CAN-Schnittstelle:
  - ISO-Standard DIS 11898, galvanisch getrennt
  - Bitrate einstellbar: 20, 50 100, 125, 250, 500 1000kBit/s
  - Modulzahl: max. 30
  - Protokoll: Murrelektronik "MBS" oder Selectron "Selecant-DP"
- Serielle Schnittstelle:
  - RS-232
  - 2400 Baud
  - 8 Datenbits
  - keine Paritätsbit
  - 1 Stopbit

## 2.2.2. Software

- Fahrbefehle:
  - Handbetrieb mit Vorgabe der Geschwindigkeit und Rampenzeit
  - Nullpunkt anfahren mit Vorgabe der Richtung, Geschwindigkeit und Rampenzeit
  - Punkt zu Punkt-Betrieb mit Vorgabe der Geschwindigkeit, Position und Rampenzeit
  - relativer Punkt zu Punkt-Betrieb mit Vorgabe der Geschwindigkeit, Position und Rampenzeit
  - Multipunktbetrieb mit Vorgabe von Mehrfachpunkten (Zeitraster und Fahrweg)
- Weitere Befehle
  - Stoppen mit Bremsrampe
  - Not-Stopp
  - Digitaler Ausgang (Bremse) Ein/Aus
  - Positionsregelung Ein/Aus
- Fahrbefehl-Speicher:
  - bis zu 1000 komplette Fahrbefehlen
- Systemkonfiguration:
  - max. Drehzahl
  - Incremente pro Umdrehung
  - System-Drehrichtung
  - max. erlaubte Regelabweichung (Positionsdivergenz)
  - Steller Ein/Aus
  - Integrationskonstante
- Fuzzy-Controller Parameterisierung
  - Veränderung der Fuzzy-Controller Parameter (Zugehörigkeitsfunktionen, Regeln)
  - Auslesen sämtlicher Variablen
- Einstellprogramm für:
  - Drehzahl
  - Drehrichtung
  - Offsetabgleich nicht erforderlich



- Positionsregler (Fuzzy-Controller):
  - 2 Eingänge: Positionsdivergenz, Ableitung der Positionsdivergenz
  - 1 Ausgang: Sollgeschwindigkeit
  - 5 bzw. 3 lineare Zugehörigkeitsfunktionen der Eingänge
  - 5 Singletons für den Ausgang
  - 15 Fuzzy-Regeln frei konfigurierbar
  - Abtastzeit = 5 ms
- Sollwertgenerator:
  - Erzeugung des optimalen Fahrprofils
  - Berechnung der Sollposition und der Sollgeschwindigkeit für den Positionsregler
  - Berechnung der Positionsdivergenz und deren Ableitung über die Zeit
  - Überwachung der max. erlaubten Positionsdivergenz (Regelabweichung)
  - 32 Bit breite Soll- und Istposition
  - Anfahr- und Bremsrampe mit  $\sin^2$ -Funktion
  - Rampenzeit (50/100/250/500ms/1/2,5/5s)
  - Maximalgeschwindigkeit 5.000 bis 1000.000 Inc/s ( $U/s * Inc/U$ )
  - Geschwindigkeit: 1-90% der Maximalgeschwindigkeit
- CAN Kommunikation:
  - Murrelektronik MBS Protokoll oder Selectron DP Protokoll
  - Abfrage der Systemkonfiguration
  - Ausführung eines Fahrbefehls direkt oder aus Fahrbefehlsspeicher
  - Diagnose und Statusdaten
- Serielle Kommunikation
  - PC Bedienung mit Windows3.x-Software "FPServo"
  - Eingabe der Systemkonfiguration
  - Einstellprogramm ausführen
  - Eingabe und Speicherung von Fahrbefehlen
  - Ausführung eines Fahrbefehls direkt oder aus Fahrbefehlsspeicher
  - Fuzzy-Controller Parameterisierung
  - Diagnose
  - Aufzeichnung und Ausführung von Befehlssequenzen

## 2.3. Detailbeschreibung

### 2.3.1. Ablaufsteuerung

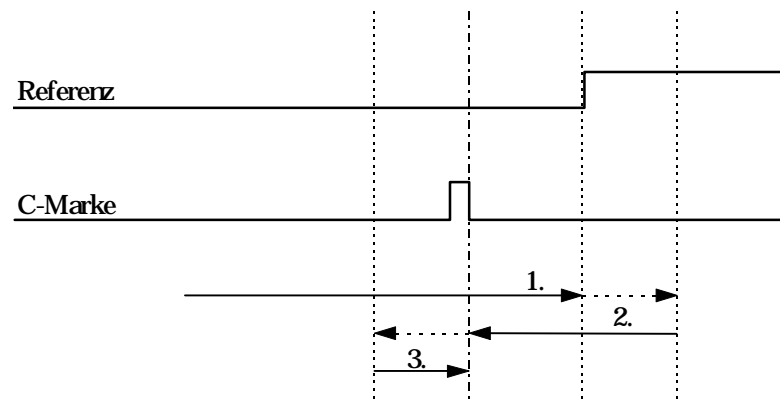
#### 2.3.1.1. Die Befehle

<b>OFF</b>	Regler ausgeschaltet, es erfolgt keine Positionsregelung.
<b>ON</b>	Regler eingeschaltet, die aktuelle Position wird geregelt.
<b>RIDE</b>	Absolute Positionierung, der Motor wird von der aktuellen Position bis zur eingestellten Position bewegt.
<b>RIDEREL</b>	Relative Positionierung, der Motor wird um den eingestellten Weg weiterbewegt.
<b>ZERO +/-</b>	Wegachse Nullen (beide Richtungen)

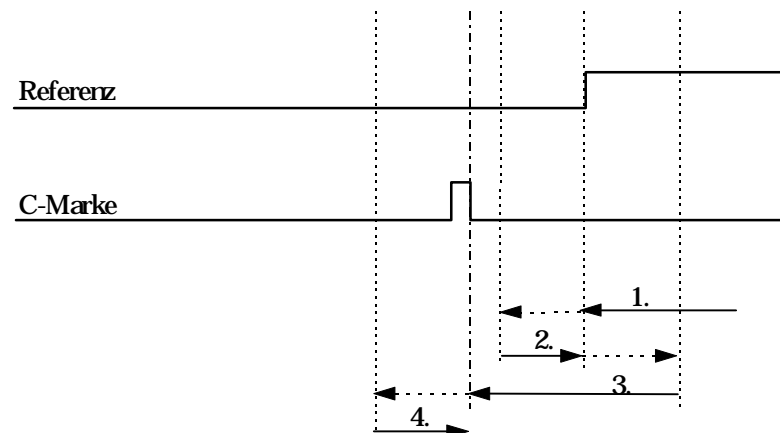
Es stehen 2 Arten der Nullung zur Verfügung, die jeweils in 2 Richtungen betrieben werden können.

#### a.) Referenz- und C-Marke

Ausgangspunkt außerhalb Referenz

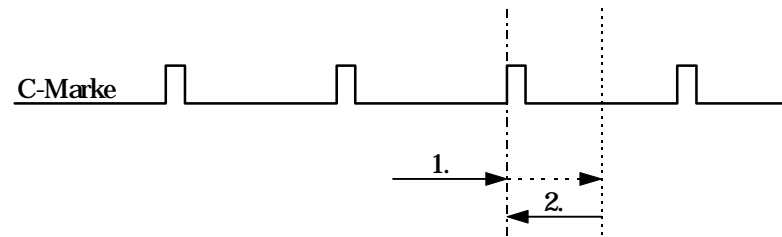


Ausgangspunkt innerhalb Referenz





## b.) C-Marke (ohne Referenz)



**STOP**

Die Bewegung wird mit der selben Rampe, mit der Sie begonnen wird, beendet.

**FSTOP**

Die Bewegung wird sofort mit der schnellsten Rampe beendet.

**OUT\_2\_ON**

Aktiviert den Ausgang 2.

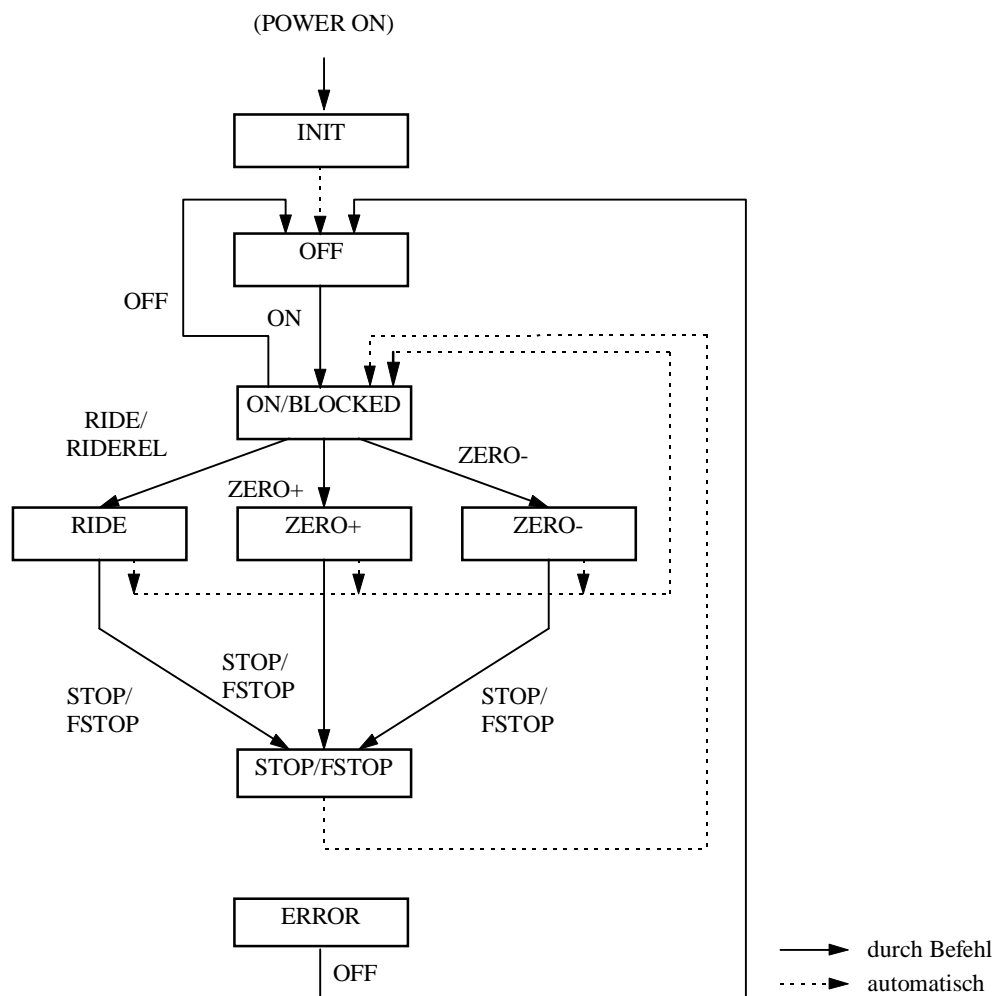
**OUT\_2\_OFF**

Deaktiviert den Ausgang 2.

## 2.3.1.2. Befehls/Status-Tabelle

Operating-Mode Status	OFF	ON	RIDE	ZERO+	ZERO-	STOP	FSTOP	OUT_2_ON OUT_2_OFF
NOP	erlaubt	erlaubt	erlaubt	erlaubt	erlaubt	erlaubt	erlaubt	erlaubt
OFF	erlaubt	erlaubt						erlaubt
ON, BLOCKED	erlaubt	erlaubt	erlaubt	erlaubt	erlaubt			erlaubt
RIDE						erlaubt	erlaubt	erlaubt
ZERO+						erlaubt	erlaubt	erlaubt
ZERO-						erlaubt	erlaubt	erlaubt
STOP							erlaubt	erlaubt
FSTOP								erlaubt
INIT								
ERROR	erlaubt							erlaubt

## 2.3.1.3. Zustandsdiagramm (Reglerstatus-Wechsel)



## 2.3.2. Sollwertgenerator

### 2.3.2.1. Geschwindigkeitsvorgabe (Steller)

Aus der Sollposition, Rampenzeit (Anfahr- und Bremsrampe) und der Sollgeschwindigkeit wird jenes Geschwindigkeitsprofil ermittelt, dessen Fläche unter der Kurve der Sollposition minus Istposition entspricht.

Dazu werden folgende Berechnungen durchgeführt:

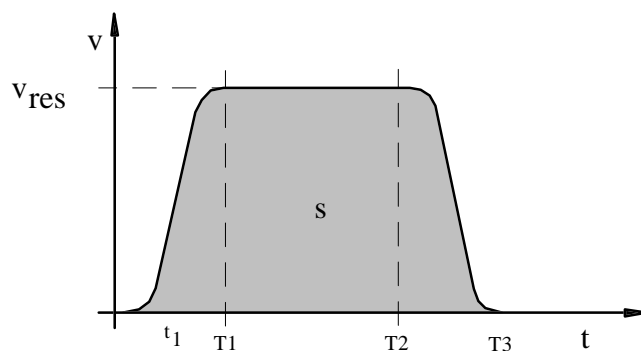
```
setpos = sollpos - istpos;           // Fahrweg
vres = vmax · vend;                 // Geschwindigkeit in Inc/s mal 100
step = 1000 · TA / t1;              // Schrittweite, mit der die Tabelle durchlaufen wird
T1 = t1;                            // Ende der Anfahrphase
T2 = T1 + setpos / vres · 100000 - t1; // Ende der Konstantfahrphase
```

Der Faktor 100000 resultiert aus der Umrechnung von

$v_{res}$

(inc/s) in inc/ms und der Multiplikation mit 100%.

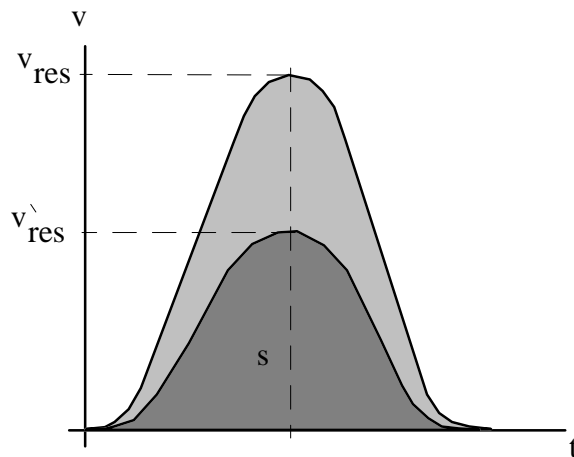
```
T3 = T2 + T1;                        // Ende der Bremsphase
```



Die Anfahr- und Bremsrampen entsprechen einer  $\sin^2$ -Funktion, um "weiche" Übergänge der einzelnen Phasen zu erhalten und somit ein möglichst ruhiges Positionieren des Motors zu erzielen.

Nach den Sollwertvorgaben müssen 2 Kriterien geprüft und ggf. das Profil korrigiert werden.

- Die Sollposition ist so klein, daß  $v_{res}$  nicht erreicht werden kann ( $T2 = T1$ ):



Folgerung:  $v_{res}$  wird automatisch nach unten korrigiert

$$v_{res} = 1000 \cdot setpos / t1 \cdot v_{end};$$

Der Faktor 1000 resultiert aus der Umrechnung von  $t_1$  (ms) in s.

$$T2 = T1;$$

$$T3 = 2 \cdot T1;$$

- Wenn die Zeit der konstanten Geschwindigkeit ( $T2-T1$ ) nicht durch die Abtastzeit teilbar ist, so wird sie auf die nächste Abtastzeit verlängert und  $v_{res}$  so nach unten korrigiert, daß der Weg gleich bleibt.

$$T2 = T2 + TA - ((T2 - T1) \% TA);$$

$$v_{res} = v_{res} \cdot T2_{old} / T2;$$

$$T3 = T2 + T1;$$

Sind diese Berechnungen durchgeführt (jedesmal nach einer Benutzereingabe), so werden nach einem Start die folgenden Phasen durchlaufen.

## Anfahrphase:

$$v(t) = v_{res} \cdot \sin^2(\pi/2 \cdot t/t_1);$$

Die Faktoren " $\sin^2(\pi/2 \cdot t/t_1)$ " werden in 1000 Schritten ( $t/t_1 = 1/1000 \dots 1$ ) in eine Tabelle eingetragen, zuvor jedoch mit 114 (90% von 127) multipliziert. Die Tabelle enthält somit Werte von 0-114 (signed 8 Bit), die dem Sollwertgenerator während der Laufzeit zur Verfügung stehen.

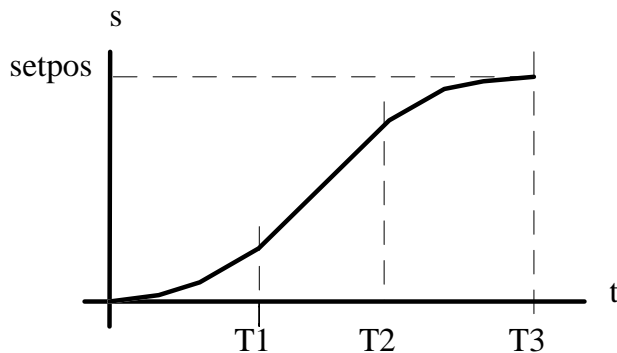
Zur Berechnung der momentanen Sollgeschwindigkeit wird der Wert von  $v_{end}$  auf 16 Bit erweitert (1% ... 655, 100% ... 65500) und mit dem Tabellenwert multipliziert. Daraus resultiert ein 24 Bit Ergebnis ( $16 \cdot 8$ ) von welchem die unteren 16 Bit verworfen und nur die oberen 8 Bit verwendet werden.

**Konstantfahrphase:**  $v(t) = v_{end} \cdot 114$ ; (90% von 127)

**Bremsphase:** siehe "Rampe auf"

## 2.3.2.2. Wegvorgabe

Der Sollwert für die Position wird aus dem Integral der Geschwindigkeit über die Zeit ermittelt.



### Anfahrphase:

Für das Anfahren gilt folgender Zusammenhang:

$$s(t) = v_{max} / 2 \cdot t \cdot [1 - (T_v / \pi) \cdot \sin(\pi / T_v)]; \quad \text{mit } T_v = t_1 / t \text{ und } t = n \cdot Ta;$$

Die Faktoren " $n \cdot Ta / 2 \cdot [1 - (T_v / \pi) \cdot \sin(\pi / T_v)]$ " werden in 1000 Schritten ( $T_v = 1000 \dots 1$ ) in eine Tabelle eingetragen, zuvor jedoch mit einer Konstanten multipliziert, um Integerwerte zu erhalten.

Zur Berechnung der momentanen Sollposition wird der Tabellenwert (16 Bit) mit einer Konstanten, resultierend aus  $v_{res}$  (32 Bit), multipliziert. Dabei werden beim 48 Bit - Ergebnis nur die höchstwertigen 32 Bit für die Sollposition berücksichtigt. Dadurch wird die Kompensation der anderen Faktoren erreicht.

### Konstantfahrphase:

Während der Phase konstanter Geschwindigkeit wird immer ein konstantes Wegstück zu der aktuellen Sollposition addiert. Daraus ergibt sich ein Wegverlauf mit konstanter Geschwindigkeit.

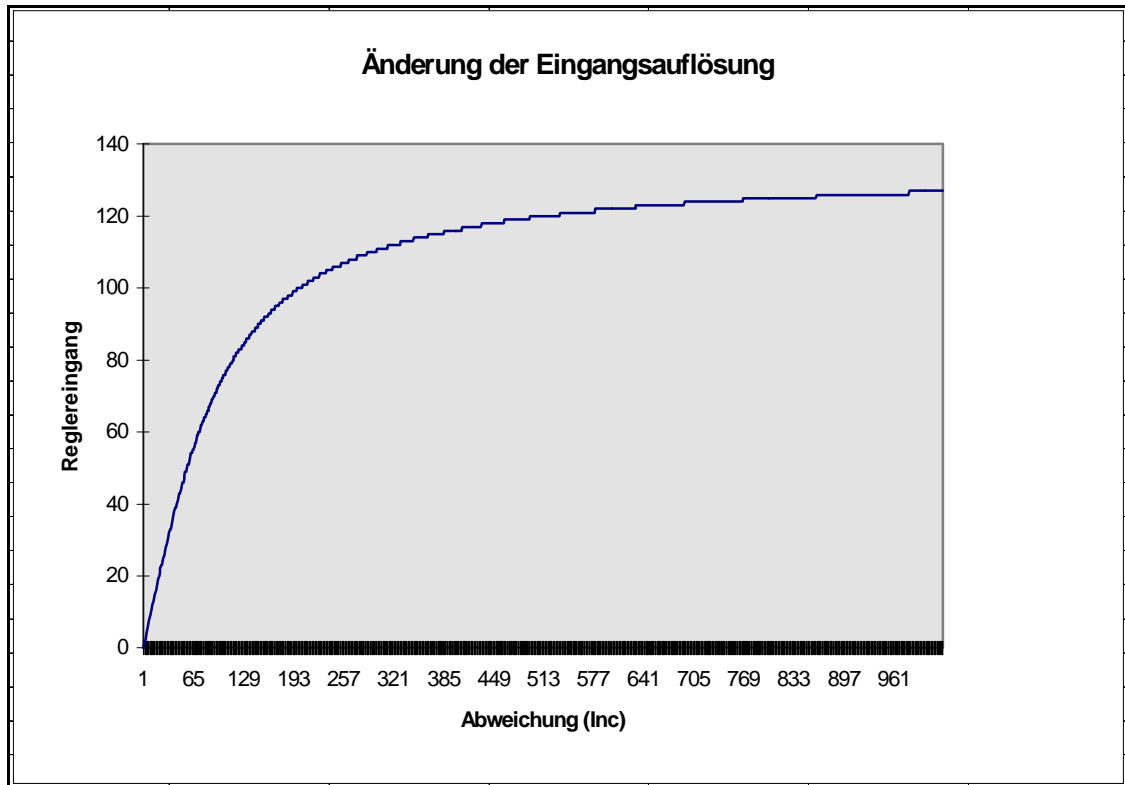
### Bremsphase:

Bei der Bremsphase wird der aus der Tabelle errechnete Wert (wie bei der Anfahrphase in umgekehrter Richtung) laufend von der Endposition (setpos) subtrahiert, so daß am Ende kein Fehler auftritt.

### 2.3.3. Konvertierung der Positionsabweichung

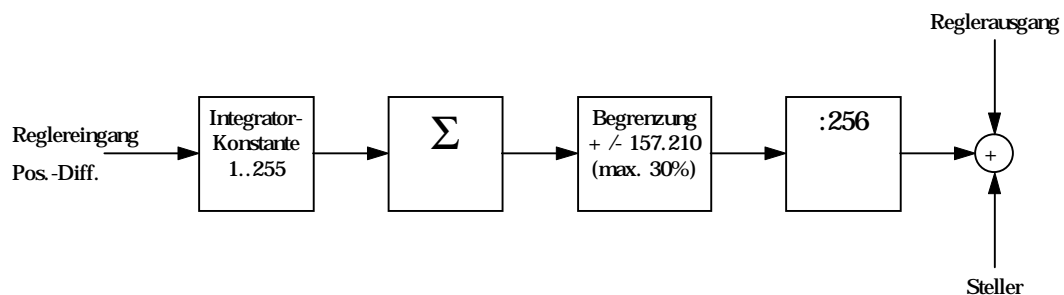
Da für den Eingang der Positionsabweichung aus Geschwindigkeitsgründen nur 8 Bit zur Verfügung stehen, werden Abweichungen von bis zu +/-1023 Incrementen lt. folgender Tabelle in den Reglereingang (max. +/- 127) umgerechnet.

Der Funktionsverlauf ist durch  $f(x)=k1*\arctan(k2*x)$  bestimmt, wobei folgende Konstanten bestimmt wurden:  $k1 = 85.911$ ,  $k2 = 0.012$ . Diese Funktion besitzt den Vorteil, daß sie in einem relativ großen Bereich um den Nullpunkt linear ist.



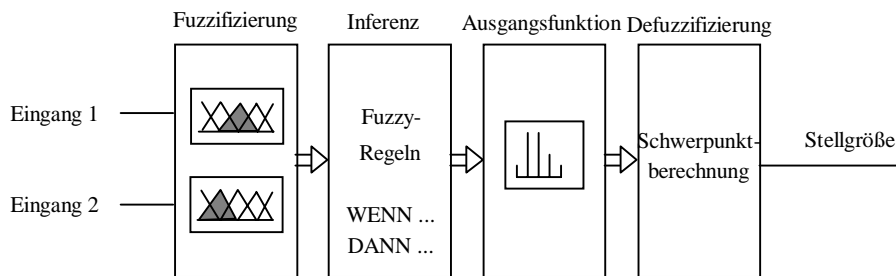
### 2.3.4. Integrator

Um bleibende Regelabweichungen zu vermeiden wurde folgender Integrator implementiert.



## 2.3.5. Fuzzy-Controller

### 2.3.5.1. Blockschema



### 2.3.5.2. Funktionsbeschreibung

#### a) Was ist Fuzzy-Control ?

Bei herkömmlichen Regelungsverfahren beschreibt man die Übertragungsfunktion zwischen Ein- und Ausgängen des Reglers genau und quantitativ mit Hilfe logischer und numerischer Gleichungen. Bei der Fuzzy-Regelung benutzt man statt dessen Ausdrücke, die die Mehrdeutigkeit der Umgangssprache enthalten. Dazu wird ein Verfahren verwendet, mit dem sich Wissen über Regeln formalisieren läßt. Die Eingangswerte für die Regelung sind Meßgrößen aus dem Prozeß und Führungsgrößen. Ein Fuzzy-Controller ermittelt daraus durch Vergleich mit der Wissensbasis die richtige Stellgröße.

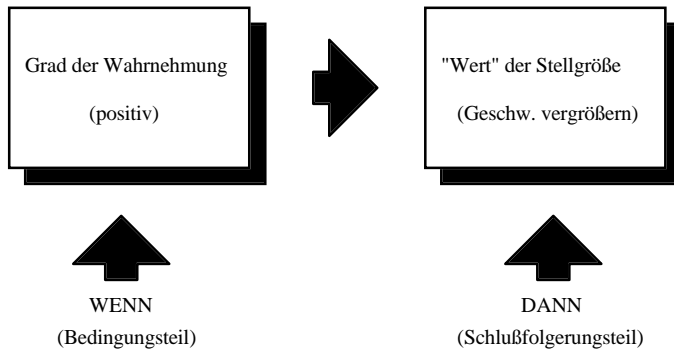
Da, wie gesagt, sowohl die Eingangs- als auch die Ausgangssignale eines Fuzzy-Controllers numerisch exakte Größen sind, andererseits aber die Übertragungsfunktion linguistische Elemente enthält, muß ein Verfahren existieren, um letztere zu quantifizieren.

#### Beispiel:

Zur Positionsregelung eines DC-Servos könnte man das Wissen, das jedermann unbewußt anwendet, wie folgt als Text formulieren.

- Ist die Positionsdivergenz positiv, so ist die Geschwindigkeit zu erhöhen.
- Ist die Positionsdivergenz null, so ist die Geschwindigkeit beizubehalten.
- Ist die Positionsdivergenz negativ, so ist die Geschwindigkeit zu reduzieren.

All diesen Sätzen ist gemeinsam, daß aus dem Grad der Wahrnehmung der Regelgröße die Stellgröße qualitativ abgeleitet wird. Dies gilt für alle Situationen, in denen man Fuzzy-Controller entwerfen will. Das Expertenwissen muß verbal dargestellt werden.



Die oben gezeigte Art, Wissen zu formulieren wird in der Fuzzy-Logik als "Regeln" bezeichnet. Regeln weisen bestimmte, immer wiederkehrende Merkmale auf.

Kombinationen von Bedingungen und Schlußfolgerungen werden der Reihe nach einzeln aufgeführt. Als Gerüst ergibt sich folgende Form:

Bedingung B1 -> Schlußfolgerung F1

Bedingung B2 -> Schlußfolgerung F2

Bedingung B3 -> Schlußfolgerung F3

Häufig führt in der Praxis erst die Kombination mehrerer Bedingungen zu einer oder auch mehreren Schlußfolgerungen.

Bedingung B1, B2 -> Schlußfolgerung F1, F2

usw.

Solche Regeln bilden die Grundlage der Wissensbasis. Bringt man die Regeln in die WENN - DANN - Form, so ergibt sich nachfolgende Darstellung. Sprachlich gesehen werden wie in der klassischen Logik die Bedingungen ebenso wie die Schlußfolgerungen miteinander UND verknüpft, weil z.B. alle Bedingungen einer Regel gleichzeitig erfüllt sein müssen, damit die Regel sich auswirken kann oder - wie man sagt - "zündet". Die Regeln selbst werden über ODER verknüpft, damit jede einzelne eine Chance hat, mit ihrer Schlußfolgerung zur Stellgröße beizutragen.

WENN (B1 UND B2) DANN (F1 UND F2)

ODER

WENN (B3 UND B4) DANN (F3 UND F4)

ODER

WENN (B1 UND B4) DANN (F5 UND F1)

usw.

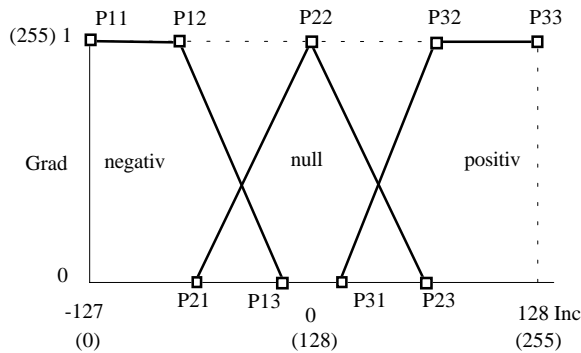
Die Verknüpfung unscharfer Mengen verlangt selbstverständlich andere Kalküle als die UND-, ODER-, usw. Operatoren der klassischen scharfen Logik, auf die hier nicht näher eingegangen wird.



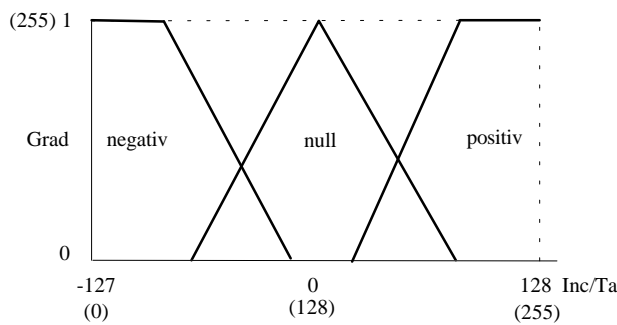
## b) Zugehörigkeitsfunktionen:

Im Bedingungs- und Schlußfolgerungsteil wird der Wert von Variablen, die eigentlich scharf meßbar sind, verbal umschrieben. Man redet in diesem Zusammenhang auch von linguistischen Variablen, die fuzzifiziert werden. Ihre Darstellung erfolgt mit Hilfe der erwähnten Zugehörigkeitsfunktion. Für alle Ein-,Ausgänge muß eine Zugehörigkeitsfunktion festgelegt werden.

### Eingang 1, Positionsdifferenz



### Eingang 2, Änderung Positionsdifferenz



Die Werte in der Klammer entsprechen der Rechenwerte, mit denen die Software rechnet. Die physikalischen Werte werden am Eingang umgerechnet.  $T_a = 5\text{ms}$ , Abtastzeit der Reglers. Der Punkt P11 liegt immer auf der x-Position = (0) und Punkt P33 auf der x-Position = (255).

Jeder der Terme "negativ", "null" und "positiv" benennt eine unscharfe Menge, die durch eine eigene Kurve definiert ist. Auf der waagrechten Achse ist die linguistische Variable Positionsdifferenz bzw. Änderung der Positionsdifferenz aufgetragen. Die senkrechte Achse zeigt den Grad, mit dem eine bestimmte Positionsdifferenz bzw. Änderung der Positionsdifferenz zu einer bestimmten unscharfen Menge gehört.

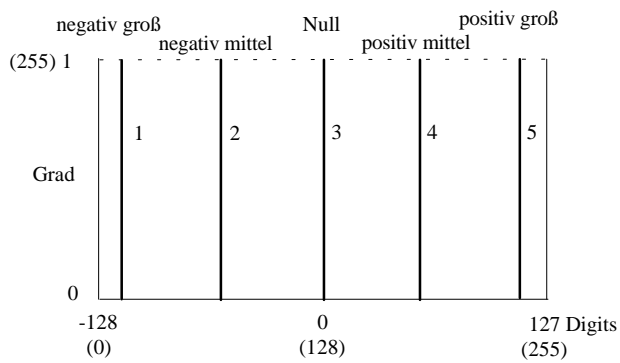
### Beispiel:

Eine Positionsdifferenz von 0 Inc gehört mit dem Grad 1 zur Menge "null". 50 Inc dagegen sind nur noch zum Grade 0.3 "null", aber bereits ebenfalls zum Grade 0.4 "positiv".

Am häufigsten verwendet werden in der Praxis Zugehörigkeitsfunktionen in Dreiecks- oder Trapezform.

## Ausgang, Stellgröße:

Im Schlußfolgerungsteil wird die Stellgröße, ganz ähnlich wie die Eingangsgrößen, verbal beschrieben. In vielen Fällen reicht hier eine Beschreibung der Zugehörigkeitsfunktion einer Variablen über Standardfunktionen und zwar über einfache Delta-Funktionen mit dem Wert 1 (sog. Singletons). Durch das Benutzen dieser Singletons wird der Rechenaufwand bei der Ermittlung der scharfen Stellgröße mit der Schwerpunktmethod stark reduziert, und trotzdem resultieren aus diesen Vereinfachungen praktisch keine regelungstechnischen Nachteile.



Die Werte in der Klammer entsprechen der Rechenwerte, mit denen die Software rechnet. Am Ausgang wird auf den Wert umgerechnet. Die Singletons (1..5) können beliebige x-Positionen (0..255) besitzen.

## c) Fuzzy-Regeln und Fuzzy-Logik, Inferenz

Nachdem nun die Terme des Bedingungs - und des Schlußfolgerungsteils in Form von Zugehörigkeitsfunktionen definiert sind, ist ein Formalismus (auch Operator genannt) erforderlich, der die Verknüpfung zwischen diesen unscharfen Mengen herstellt.

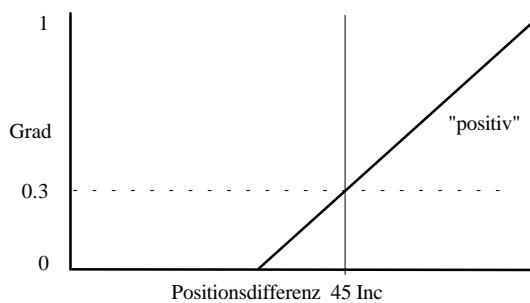
### Das Konzept des Bedingungsteils:

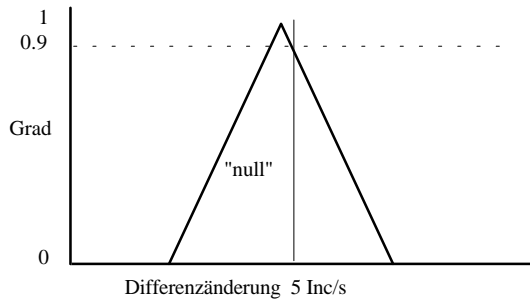
In dem Fall, wo eine Regel mehrere Bedingungen besitzt, muß als erstes aus den einzelnen Zugehörigkeitsgraden ein einziger Zugehörigkeitsgrad gewonnen werden. Im Laufe der Zeit sind eine ganze Reihe von Operatoren entwickelt worden, die dies leisten. Einer der einfachsten und meistverbreiteten Operatoren ist der MIN-MAX-Operator, der im folgenden Beispiel erläutert wird.

Angenommen, es gelte folgende Regel:

WENN Positionsdiffenz "positiv" UND Differenzänderung "null" DANN...

Weiter wird angenommen, daß die gemessene Differenz 45 Incremente und die Differenzänderung 5 Inc/Ta beträgt.





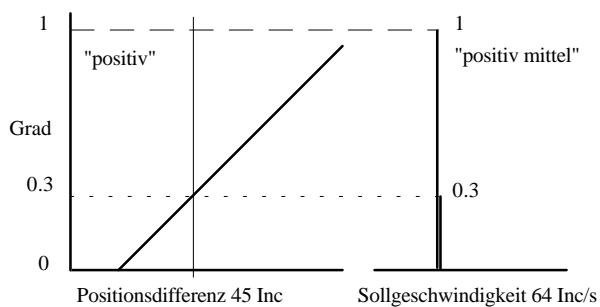
Betrachtet man nun den Term "positiv" in der Zugehörigkeitsfunktion für die Positionsdifferenz und den Term "null" in der Zugehörigkeitsfunktion der Differenzänderung, so erkennt man, 45 Inc gehört zu einem Grad 0.3 zur Fuzzy-Menge "positiv", 5 Inc/s gehört zu einem Grad 0.9 zur Fuzzy-Menge "null". Der MIN-MAX-Operator besagt nun, daß der zusammengefaßte Zugehörigkeitsgrad aller über UND verknüpfte Bedingungen unserer Regel dem Minimum der einzelnen Zugehörigkeitsgrade entspricht somit also 0.3 beträgt.

## Das Konzept des Schlußfolgerungsteils:

Der Erfüllungsgrad einer Regel muß sich im Zugehörigkeitsgrad der Schlußfolgerungsterme widerspiegeln. Beispielsweise heißt die Regel weiter:

... DANN ist die Sollgeschwindigkeit "positiv mittel".

Der Term "positiv mittel" entspricht in der Zugehörigkeitsfunktion für die Sollgeschwindigkeit einem Singleton bei 64 Inc/s. Genau wie das Ergebnis der Bedingungen ein Erfüllungsgrad von 0.3 war, muß nun auch dieser Singleton in einer Höhe von 0.3 abgeschnitten werden. Wäre die Regel gänzlich erfüllt gewesen (Erfüllungsgrad 1), dann besäße auch die Geschwindigkeit von 64 Inc/s einen Zugehörigkeitsgrad von 1 zum Term "positiv mittel". Bei einem Erfüllungsgrad von 0 dieser Regel würde eine Geschwindigkeit von 64 Inc/s überhaupt nichts zum späteren Gesamtergebnis beitragen.

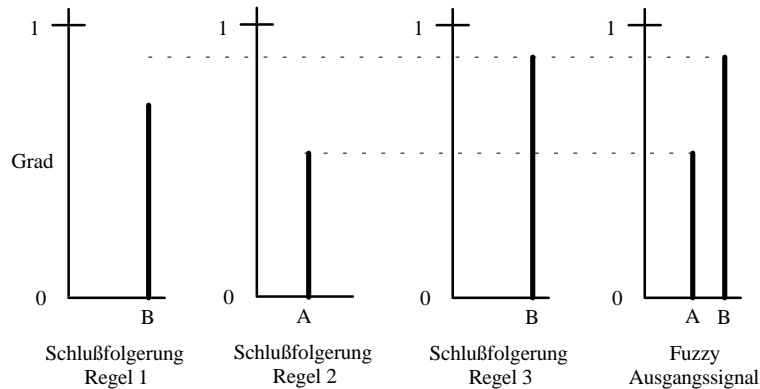


## Zusammenführen mehrerer Regeln:

Das Ergebnis jeder Regel sind - wie oben geschildert - mehr oder weniger abgeschnittene Terme der Ausgangsvariablen, bei denen es sich in diesem Fall um Singletons handelt. Um nun zu einer Zusammenfassung der Schlußfolgerungen aller Regeln zu kommen, müssen diese abgeschnittenen Singletons überlagert werden. Hierbei kommt der zweite Teil des MIN-MAX-Operators zu Anwendung:

Man sucht nämlich für jeden Term in allen Schlußfolgerungen nach dessen Maximalwert. So kommt man zu einer einzigen Zugehörigkeitsfunktion für alle Regeln, die man auch als Fuzzy-Ausgangssignal bezeichnet. Den Maximalwert jedes Termes zu nehmen ist plausibel, da alle Regeln über eine ODER-Aussage miteinander verknüpft sind. Damit setzt sich für jeden Term die Regel mit dem höchsten Erfüllungsgrad durch.

Den bis hierher geschilderten Prozess der Abarbeitung mehrerer Regeln bis hin zur Erzeugung des Fuzzy Ausgangssignals nennt man "Inferenz".



### d) Die Bestimmung der Stellgröße

Die nach dem Zusammenführen erhaltene Zugehörigkeitsfunktion ist das Ergebnis der Verarbeitung von fuzzifizierten Eingangssignalen im Regelwerk. Was nun noch fehlt, ist die Berechnung einer definierten Stellgröße. Der Schritt zur Berechnung einer scharfen Stellgröße heißt "Defuzzifizierung". Dafür gibt es verschiedene Rechenmethoden, von denen die Schwerpunktmethode bei Singletons im folgenden erläutert wird.

Für jeden Singleton des Fuzzy-Ausgangssignals wird der Wert seiner linguistischen Variablen mit seiner Höhe (seinem Grad) multipliziert. Alle diese Produkte werden aufaddiert. Weiterhin wird die Summe aller Grade gebildet. Das Ergebnis - es entspricht dem Schwerpunkt aller beteiligten Singletons auf der waagrechten Achse - wird der Stellgröße zugewiesen.

### 2.3.5.3. Fuzzy-Regeln

Der Fuzzy-Controller besteht in diesem Fall aus 2 Eingängen ( $D_s$ ,  $dD_s/dt$ ) und einem Ausgang der zur Sollgeschwindigkeit des Sollwertgenerators hinzuaddiert wird.

Die beiden linguistischen Eingangsvariablen bestehen aus jeweils 3 Zugehörigkeitsfunktionen für **N**-egativ, **Z**-ero und **P**-ositiv.

Der Ausgang besteht aus 5 Singletons mit den Werten **N**-egativ, **NM**-negativ medium, **Z**-ero, **PM**-positiv medium und **P**-ositiv.

Folgende Regeln wurden aufgestellt:

WENN	$D_s=N$	UND	$dD_s/dt=Z$	DANN	$y=NM$
WENN	$D_s=N$	UND	$dD_s/dt=P$	DANN	$y=Z$
WENN	$D_s=N$	UND	$dD_s/dt=N$	DANN	$y=N$
WENN	$D_s=Z$	UND	$dD_s/dt=Z$	DANN	$y=Z$
WENN	$D_s=Z$	UND	$dD_s/dt=P$	DANN	$y=PM$
WENN	$D_s=Z$	UND	$dD_s/dt=N$	DANN	$y=NM$
WENN	$D_s=P$	UND	$dD_s/dt=Z$	DANN	$y=PM$
WENN	$D_s=P$	UND	$dD_s/dt=P$	DANN	$y=P$
WENN	$D_s=P$	UND	$dD_s/dt=N$	DANN	$y=Z$

Nach dem Prüfen dieser Regeln wird  $y$  mit der MinMax-Methode gebildet und mit der Schwerpunktmethode defuzzifiziert.

#### 2.3.5.4. Programmstruktur

Eingänge lesen

Ausgangs - Singletons löschen

for (alle Regeln)

```
{  
  if (Regel erfüllt)  
  {  
    minimalen Grad ermitteln;  
    if (Grad > bereits Eingetragener Wert)  
    {  
      Grad in entsprechenden Singleton eintragen;  
    }  
  }  
}
```

Stellgröße = Summe der Produkte (Grad \* x) / Summe aller Grade;

### 2.3.6. 15-Bit PWM-Ausgang aus 2\*8-Bit

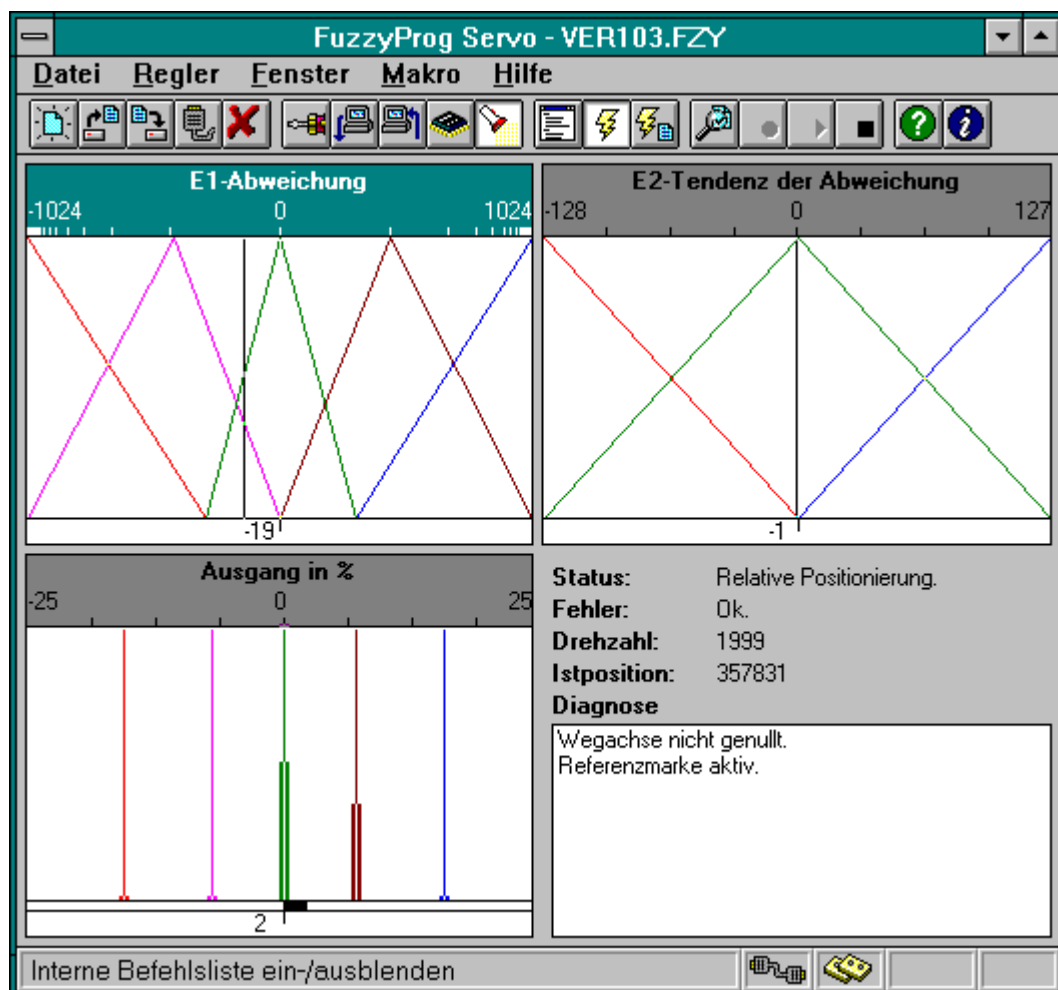
Da nur 8-Bit PWM-Ausgänge zur Verfügung stehen, wurden 2 Ausgänge so "gemischt", daß die Summe einen 15-Bit Ausgang ergibt. Der Steller wird dabei immer nur in 8-Bit Auflösung berechnet. Der Reglerausgang und der I-Anteil erreichen aber 15-Bit Auflösung.

### 2.3.7. Service-Software FuzzyProg Servo für Windows 3.x

FuzzyProg ist ein Service- bzw. Testprogramm, das auf jedem IBM kompatiblen PC unter Windows 3.x läuft.

Folgende Funktionen werden dabei unterstützt:

- Graphische Einstellung der Reglerparameter sowie der Fuzzy-Regeln
- Einstellung des Reglersystems incl. Handbetrieb zur Bestimmung von Drehrichtung, Zählrichtung, max. Drehzahl und Incremente/Umdrehung
- Programmieren des internen Befehlsspeichers im batteriegepufferten RAM des Moduls
- Reglereinstellungen im PC abspeichern
- Befehlssequenzen ausführen und im PC speichern (Makros)
- Online-Anzeige der Istwerte Pos.Diff., d/dt, Reglerausgang, Istposition, Reglerstatus, Modulstatus, Drehzahl, Fehler und Wirkungsgrade der Fuzzy-Regeln all 500ms
- Durch Drücken der Taste F1 steht über die kontextsensitive Online-Hilfe jederzeit Hilfe zur Verfügung.



## 3. CAN-Bus-Kommunikation

### 3.1. CAN Schnittstelle

Schnittstelle:	ISO 11898 High-Speed Interface		
Anschlußstecker:	Flachstecker für 10 pol. Flachkabel oder 9pol. D-Sub Stecker mit Adaptermodul.		
Abschlußwiderstand:	120Ohm schaltbar		
Potentialtrennung:	gegenüber Elektronik		
Übertragungsrate:	1 MBit/s, 500kBit/s, 250kBit/s, 125kBit/s, 100kBit/s, 20kBit/s		
Kabellänge:	1 MBit/s	max.	20m
	500kBit/s	max.	90m
	250kBit/s	max.	200m
	125kBit/s	max.	400m
	100kBit/s	max.	500m
	20kBit/s	max.	1000m
Teilnehmerzahl:	max. 30 Slave-Module, Moduladresse einstellbar mittels Adreßschalters.		

### 3.2. Aufbau CAN-Meldung

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	Descriptor-Byte 1
ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0	Descriptor-Byte 2
								Data-Byte 1
								Data-Byte 2
								Data-Byte 3
								Data-Byte 4
								Data-Byte 5
								Data-Byte 6
								Data-Byte 7
								Data-Byte 8

Descriptor:	ID0..10:	Message-Identifizier (0..2047)
	RTR-Bit:	nicht benutzt, immer 0
	DLC0..3:	Anzahl Data-Bytes (0..8)
Data-Byte1..8:	Nutzdaten, variable Länge 0 bis 8 Bytes	

## 3.3. Murrelektronik MBS-Protokoll (nicht implementiert)

### 3.3.1. Allgemeines

Beim MBS-Protokoll handelt es sich um ein reines Master/Slave Kommunikations-Konzept. Der Master (SPS, PC oder MBS-Busmodul) ist für die Kommunikationsaufnahme verantwortlich und der Slave (Servo) antwortet nur nach einer Masteranforderung.

Die CAN-Schnittstelle wird laufend überwacht und bei einem CAN-Timeout von über 200ms wird ein Auftrag, der über die CAN-Schnittstelle gesendet wurde, abgebrochen. Im Debug-Mode ist die Überwachung ausgeschaltet.

### 3.3.2. Message-Identifizier

ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID
10	9	8	7	6	5	4	3	2	1	0

0	0	0	0	0	1	1					Modul 1 (Slave-Modul-Nr.)
0	0	0	0	1	0	1					Modul 2
0	0	0	1	0	0	1					Modul 3
0	0	1	0	0	0	1					Modul 4
0	1	0	0	0	0	1					Modul 5
0	0	0	0	1	1	0					Modul 6
0	0	0	1	0	1	0					Modul 7
0	0	1	0	0	1	0					Modul 8
0	1	0	0	0	1	0					Modul 9
0	0	0	1	1	0	0					Modul 10
0	0	1	0	1	0	0					Modul 11
0	1	0	0	1	0	0					Modul 12
0	0	1	1	0	0	0					Modul 13
0	1	0	1	0	0	0					Modul 14
0	1	1	0	0	0	0					Modul 15
1	0	0	0	0	1	1					Modul 16
1	0	0	0	1	0	1					Modul 17
1	0	0	1	0	0	1					Modul 18
1	0	1	0	0	0	1					Modul 19
1	1	0	0	0	0	1					Modul 20
1	0	0	0	1	1	0					Modul 21
1	0	0	1	0	1	0					Modul 22



1	0	1	0	0	1	0					Modul 23
1	1	0	0	0	1	0					Modul 24
1	0	0	1	1	0	0					Modul 25
1	0	1	0	1	0	0					Modul 26
1	1	0	0	1	0	0					Modul 27
1	0	1	1	0	0	0					Modul 28
1	1	0	1	0	0	0					Modul 29
1	1	1	0	0	0	0					Modul 30
0	1	1	1	1	1	1	1	0	x	x	Broadcast Command-Message von Master an alle Slave-Module 1..15
1	1	1	1	1	1	1	1	0	x	x	Broadcast Command-Message von Master an alle Slave-Module 16..30
							0				Data Message
							1				Command Message
								0	x	x	Message von Master an Slave (Frame-Nr.: 0..3)
								1	x	x	Message von Slave an Master (Frame-Nr.: 4..7)

### 3.3.3. Data-Bytes

Jede Message kann 0 bis 8 Data-Bytes besitzen. Die Anzahl der Daten-Bytes (0..8) wird im Control-Feld (4 Bit) übertragen.

### 3.3.4. Message-Typen

Die Meldungen werden wie folgt eingeteilt:

- **Control-Command Messages**  
Es wird eine Gruppe von Slave-Modulen (Broadcast-Command) oder nur ein Slave-Modul (Direkte Moduladressierung) von Master aus gesteuert. Die angesprochenen Slave-Module senden keine Message zurück.
- **Request-Command Messages**  
Der Master sendet einem Slave-Modul eine Request-Message. Das angesprochene Slave-Modul sendet die angeforderten Systemdaten zurück.
- **Data-Messages**  
Der Master sendet Data-Output-Message an ein Slave-Modul. Das angesprochene Slave-Modul bestätigt sie mit der Data-Input-Message inklusive Status-Byte.



### 3.3.5. Control-Command-Message

Diese Message wird von jedem Slave-Modul unterstützt.

#### Master nach Slave

- Modul-Adresse: 1..30 oder Broadcast (1..15/16..30)
- Command/Data-Flag: 1 (Command-Message)
- Frame-Nr.: 0
- Data-Byte1: Command-Code
  - 0 = Reset Modul
  - 1-5 nicht belegt
  - 6 = Set Debug Mode
  - 7 = ClearDebug Mode
  - 8-255 nicht belegt

#### Slave nach Master: keine Message

### 3.3.6. Request-Command-Messages

#### 3.3.6.1. Identifikation

Es wird die Identifikation des Slaves gelesen.

##### **Master nach Slave**

- Modul-Adresse: 1..30
- Command/Data-Flag: 1 (Command-Message)
- Frame-Nr.: 1
- Data-Byte1: Request-Code  
0 = Identification

##### **Slave nach Master**

- Modul-Adresse: 1..30
- Command/Data-Flag: 1 (Command-Message)
- Frame-Nr.: 4
- Data-Byte1: Modul-Gruppe (8)
- Data-Byte2: Modul-Variante (1..x)
- Data-Byte3: Software-Version (1..99)
- Data-Byte4: Software-Revision.(0..99)

##### **Modul-Variante**

CAN Protokoll (Einer-Stelle):

x1 = Murrelektronik

x2 = Selectron DP

Antriebsart (Zehner-Stelle):

0x = DC-Servo-Antrieb mit Inkrementalgeber

1x = .....

#### 3.3.6.2. Data-Configuration

Es wird die Einteilung und Größe der Daten-Messages des Slaves gelesen.

##### **Master nach Slave**

- Modul-Adresse: 1..30
- Command/Data-Flag: 1 (Command-Message)
- Frame-Nr.: 1
- Data-Byte1: Request-Code  
1 = Data-Configuration (Anzahl Input/Output Bytes)



## Slave nach Master

- Modul-Adresse: 1..30
- Command/Data-Flag: 1 (Command-Message)
- Frame-Nr.: 5
- Data-Byte1: 28h Anzahl Data-Bytes (8), Data-Output-Frame 0(\*)
- Data-Byte2: 81h Anzahl Data-Bytes (1), Data-Input-Frame 4(\*\*)
- Data-Byte3: 0h Anzahl Data-Bytes (0), Data-Output-Frame 1
- Data-Byte4: 28h Anzahl Data-Bytes (8) Data-Input-Frame 5(\*)
- Data-Byte5: 0h Anzahl Data-Bytes (0), kein Data-Output-Frame 2
- Data-Byte6: 0h Anzahl Data-Bytes (0), kein Data-Input-Frame 6
- Data-Byte7: 0h Anzahl Data-Bytes (0), kein Data-Output-Frame 3
- Data-Byte8: 0h Anzahl Data-Bytes (0), kein Data-Input-Frame 7

(\*) Frames-Konsistenz

(\*\*) Data-Byte1 von Data-Input-Frame 4 = Diagnose-Byte

## 3.3.6.3. Diagnose

Es wird das Diagnose-Byte des Slaves gelesen.

## Master nach Slave

- Modul-Adresse: 1..30
- Command/Data-Flag: 1 (Command-Message)
- Frame-Nr.: 1
- Data-Byte1: Request-Code  
2 = Diagnose

## Slave nach Master

- Modul-Adresse: 1..30
- Command/Data-Flag: 1 (Command-Message)
- Frame-Nr.: 6
- Data-Byte1: Diagnose

## **Diagnose**

- Bit 0 = Modul im Zustand ERROR (näheres siehe Fehlernummer)
- Bit 1 = nicht belegt
- Bit 2 = nicht belegt
- Bit 3 = nicht belegt
- Bit 4 = nicht belegt
- Bit 5 = nicht belegt
- Bit 6 = reserviert
- Bit 7 = reserviert

### 3.3.7. Data-Messages

#### 3.3.7.1. Write Operation-Mode

Es wird der Operating-Mode mit den entsprechenden Parametern gesendet.

##### Master nach Slave

- Modul-Adresse: 1..30
- Command/Data-Flag: 0 (Data-Message)
- Frame-Nr.: 0
- Data-Byte1: Operating-Mode

##### ***Operating Mode NOP, OFF, ON, STOP, FSTOP, OUT2\_ON, OUT2\_OFF***

- Data-Byte2-7: nicht verwendet
- Data-Byte8: Frame-Counter, 0... keine zyklische Übertragung  
1-255... zyklische Übertragung (Frame-Übernahme erfolgt nur bei Änderung des Frame-Counters)

##### ***Operating Mode RIDE, RIDEREL, ZERO+, ZERO-***

- Data-Byte2: Sollposition LSB (Incremente)
- Data-Byte3: Sollposition LSB+1
- Data-Byte4: Sollposition LSB+2
- Data-Byte5: Sollposition MSB
- Data-Byte6: Geschwindigkeit in % der max. Geschwindigkeit (1..100)
- Data-Byte7: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms,  
4..1s, 5..2.5s, 6..5s
- Data-Byte8: Frame-Counter, 0...keine zyklische Übertragung  
1-255... zyklische Übertragung (Frame-Übernahme erfolgt nur bei Änderung des Frame-Counters)

##### ***Operating Mode MEMCMD***

- Data-Byte2: Befehlsnummer in der Tabelle (0-65535) LSB
- Data-Byte3: Befehlsnummer in der Tabelle (0-65535) MSB
- Data-Byte4-7: nicht verwendet
- Data-Byte8: Frame-Counter, 0...keine zyklische Übertragung  
1-255... zyklische Übertragung (Frame-Übernahme erfolgt nur bei Änderung des Frame-Counters)

##### Slave nach Master

- Modul-Adresse: 1..30
- Command/Data-Flag: 0 (Data-Message)
- Frame-Nr.: 4
- Data-Byte1: Diagnose



## Operating-Mode

0	= NOP	keine Aktion
1	= OFF	Regler aus
2	= ON	Regler ein, Sollposition = Istposition, keine Bewegung
3	= RIDE	Absolute Positionierung
4	= RIDEREL	Relative Positionierung
5	= ZERO+	Nullen mit Referenz, vorwärts
6	= ZERO-	Nullen mit Referenz, rückwärts
7	= ZERO2+	Nullen ohne Referenz, vorwärts
8	= ZERO2-	Nullen ohne Referenz, rückwärts
9	= STOP	Fahrbewegung wird gestoppt (mit definierter Rampe)
10	= FSTOP	Fahrbewegung wird gestoppt (schnellst möglich)
11	= OUT2_ON	Ausgang 2 ein
12	= OUT2_OFF	Ausgang 2 aus
13	= MEMCMD	Befehl aus Befehlsliste ausführen
14	= MANUAL	Handbetrieb



## 3.3.7.2. Read Status

Es wird der Status des Slaves abgefragt.

### Master nach Slave

- Modul-Adresse: 1..30
- Command/Data-Flag: 0 (Data-Message)
- Frame-Nr.: 1
- keine Daten

### Slave nach Master

- Modul-Adresse: 1..30
- Command/Data-Flag: 0 (Data-Message)
- Frame-Nr.: 5
- Data-Byte1: Modulstatus
- Data-Byte2: Reglerstatus
- Data-Byte3: Fehlernummer
- Data-Byte4: Istposition LSB (Incremente)
- Data-Byte5: Istposition LSB+1
- Data-Byte6: Istposition LSB+2
- Data-Byte7: Istposition MSB
- Data-Byte8: Operating-Mode-Status
  - 0... letzter Operating-Mode wurde übernommen
  - 1... warten auf Befehlsausführung(\*)
  - 2... letzter Operating-Mode nicht akzeptiert

(\*) In diesem Zustand wird mit Write-Operating-Mode ein anstehender Befehl überschrieben.

### **Modulstatus**

- Bit 0 = Modul im Zustand ERROR (näheres siehe Fehlernummer)
- Bit 1 = System nicht freigegeben (RUN disable) (1 = nicht freigegeben)
- Bit 2 = Endschalter negativ (1 = aktiv)
- Bit 3 = Endschalter positiv (1 = aktiv)
- Bit 4 = Wegachse nicht genullt (1 = nicht genullt)
- Bit 5 = Referenzmarke (1 = aktiv)
- Bit 6 = Ausgang2 (Out2) Status, (1 = aktiv)
- Bit 7 = nicht belegt

### **Reglerstatus**

- 1 = OFF Regler aus
- 2 = ON Regler ein, keine Bewegung
- 3 = RIDE Absolute Positionierung
- 4 = RIDEREL Relative Positionierung
- 5 = ZERO+ Nullen mit Referenz, vorwärts
- 6 = ZERO- Nullen mit Referenz, rückwärts
- 7 = ZERO2+ Nullen ohne Referenz, vorwärts
- 8 = ZERO2- Nullen ohne Referenz, rückwärts
- 9 = STOP Fahrbewegung wird gestoppt (mit definierter Rampe)
- 10 = FSTOP Fahrbewegung wird gestoppt (schnellst möglich)
- 14 = MANUAL Handbetrieb
- 20 = INIT Regler Initialisierung
- 21 = INITEND Ende der Regler Initialisierung
- 22 = SAVE Parameter werden in EEPROM gespeichert



23      = BLOCKED      Regler blockiert (Endschalter oder keine Freigabe)  
255     = ERROR        Regler-Fehler, nur mit Befehl OFF wird Zustand verlassen

## Fehlernummer

0        = kein Fehler  
1        = Servoverstärker-Fehler  
2        = Positionsgeber Fehler (Overrun)  
3        = Reglerabweichung zu groß  
4        = Checksummenfehler im batteriegepufferten RAM  
5        = Checksummenfehler im E<sup>2</sup>PROM



## 3.4. Selectron Selecan-DP Protokoll

### 3.4.1. Allgemeines

Beim DP-Protokoll handelt es sich um ein Master/Slave Kommunikations-Konzept. Der Master (SPS oder PC) ist für die Kommunikationsaufnahme verantwortlich und der Slave (Servo) antwortet nach einer Masteranforderung oder bei einem Ereignis (Fehler, Auftrag ausgeführt, Auftrag nicht angenommen).

#### Standby-Mode

Nach Power-On geht die CAN-Schnittstelle in Standby-Mode und sendet alle 1s die Meldung „Modul-Type“ an den Master und ist somit betriebsbereit. Nach Empfang der Meldung „Start-Communication“ wechselt die CAN-Schnittstelle in den RUN-Mode. Im Standby-Mode werden keine „Data-Messages“ angenommen bzw. gesendet.

#### Run-Mode

Im Run-Mode werden alle Meldungen vom Master angenommen. Wird während 5s nicht mehr über den CAN kommuniziert, so wechselt die CAN-Schnittstelle in den Standby-Mode. Nach Empfang der Meldung „Reset-Communication“ wechselt die CAN-Schnittstelle in den Standby-Mode.

### 3.4.2. Message-Descriptor

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	Descriptor-Byte 1
Prio2	Prio1	Prio0	Adr4	Adr3	Adr2	Adr1	Adr0	
ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0	Descriptor-Byte 2
Spez 2	Spez 1	Spez 0	0					

Prio 0..2: Priorität und Richtung der Meldungen auf dem Bus

Adr 0..4: Moduladresse (0..29)

Spez0..2: Modul spezifisch

DLC 0..3: Anzahl Bytes Nutzdaten (0..8)

### 3.4.3. Service-Messages

#### 3.4.3.1. Modul-Type

Das Servo-Modul sendet im CAN-Standby-Mode alle 1s den Servo-Modul-Typ an den Master und ist somit betriebsbereit für die CAN-Kommunikation.



## Slave nach Master

### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	1	0	0	0	Descriptor-Byte 2

### Nutzdaten:

- Data-Byte1: 20H Command HB
- Data-Byte2: 15H Command LB
- Data-Byte3: 8FH Modultype
- Data-Byte4: 0 nicht verwendet
- Data-Byte5: xx Software-Revision
- Data-Byte6: xx Software-Version
- Data-Byte7: 0 nicht verwendet
- Data-Byte8: 0 nicht verwendet

### 3.4.3.2. Start Communication

Der Master setzt die CAN-Schnittstelle in den CAN-Run-Mode.

## Master nach Slave

### Descriptor:

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	1	0	Descriptor-Byte 2

### Nutzdaten:

- Data-Byte1: 01H Command HB
- Data-Byte2: 50H Command LB

### 3.4.3.3. Startup Timeout

Der Master sendet die Startup-Timeout Zeit.

Wird im Run-Mode während der eingestellten Timeout-Zeit der Slave über CAN-Schnittstelle nicht angesprochen, so nimmt die CAN-Schnittstelle den CAN-Stanby-Mode ein und der letzte Fahrbefehl wird abgebrochen.

## Master nach Slave

### Descriptor:

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	1	0	0	Descriptor-Byte 2

**Nutzdaten:**

- Data-Byte1: 02H Command HB
- Data-Byte2: 12H Command LB
- Data-Byte3: Timeout LB
- Data-Byte4: Timeout HB

**Timeout:**

Zeit in ms (5..1000ms), einstellbar in 5ms Schritten. 0ms bedeutet Timer ausgeschaltet (Run-Mode wird nicht verlassen). Die Standardeinstellung nach Power-Up ist 200ms.

### 3.4.3.4. Reset Communication

Der Master setzt die CAN-Schnittstelle in den CAN-Standby-Mode.

**Master nach Slave****Descriptor:**

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	1	0	Descriptor-Byte 2

**Nutzdaten:**

- Data-Byte1: 01H Command HB
- Data-Byte2: 30H Command LB

### 3.4.3.5. Set Event-Mode

Das Servo-Modul wird in Event-Mode geschaltet. Im Event-Mode wird der Modul-Status nur bei einem Reglerstatus-Wechsel gesendet.

Die Standardeinstellung nach Power-UP ist Poll-Mode.

**Master nach Slave****Descriptor:**

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	1	0	Descriptor-Byte 2

**Nutzdaten:**

- Data-Byte1: 01H Command HB
- Data-Byte2: 40H Command LB

### 3.4.3.6. Set Poll-Mode

Das Servo-Modul wird in Event-Mode geschaltet. Im Poll-Mode wird der „Modul-Status“ unmittelbar nach Empfang der Message „Write Oprating-Mode“ gesendet.

Die Standardeinstellung nach Power-UP ist Poll-Mode.

#### Master nach Slave

##### **Descriptor:**

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	1	0	Descriptor-Byte 2

##### **Nutzdaten:**

- Data-Byte1: 01H Command HB
- Data-Byte2: 41H Command LB

### 3.4.3.7. Start Modul-Status Update

Das Servo-Modul wird aufgefordert den Modul-Status dem Master zu senden.

#### Master nach Slave

##### **Descriptor:**

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	1	0	Descriptor-Byte 2

##### **Nutzdaten:**

- Data-Byte1: 01H Command HB
- Data-Byte2: 61H Command LB



### 3.4.4. Service-Messages (nicht Selectron kompatibel)

Die folgenden Master-Messages werden mit einer Slave-Message bestätigt. Eine neue Service-Master-Message darf erst nach Bestätigung der vorangehenden Service-Master-Message gesendet werden.

#### 3.4.4.1. Befehlsliste-Tabellenindex setzen

Der Master setzt den Tabellenindex (0...999) der Befehlsliste.

##### Master nach Slave

###### Descriptor:

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	1	1	Descriptor-Byte 2

###### Nutzdaten:

- Data-Byte1: F0H Command HB
- Data-Byte2: Index LB
- Data-Byte3: Index HB

##### Slave nach Master

###### a) Index gültig, Index wurde übernommen

###### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	0	0	0	1	Descriptor-Byte 2

###### Nutzdaten:

- Data-Byte1: F0H Command HB

###### b) Index ungültig, Index wurde nicht übernommen

###### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	0	0	0	1	Descriptor-Byte 2

###### Nutzdaten:

- Data-Byte1: FFH Command HB



## 3.4.4.2. Befehlsliste-Tabellenindex lesen

Der Master liest den Tabellenindex der Befehlsliste. Der Tabellenindex wird unmittelbar nach der Anfrage vom Slave ausgesendet.

### Master nach Slave

#### Descriptor:

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	0	1	Descriptor-Byte 2

#### Nutzdaten:

- Data-Byte1: F1H Command HB

### Slave nach Master

#### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	0	0	1	1	Descriptor-Byte 2

#### Nutzdaten:

- Data-Byte1: F1H Command HB
- Data-Byte2: Index LB
- Data-Byte3: Index HB

## 3.4.4.3. Befehl (Operating-Mode) in die Befehlsliste schreiben

Der Master schreibt ein Befehl in die Befehlsliste. Als Position dient der Tabellenindex.

### Master nach Slave

#### Descriptor:

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	1	0	0	0	Descriptor-Byte 2

#### Nutzdaten:

- Data-Byte1: F2H Command HB
- Data-Byte2: Operating-Mode

#### ***Operating Mode NOP, OFF, ON, STOP, FSTOP, OUT2\_ON, OUT2\_OFF***

- Data-Byte3-8: nicht verwendet, leer



## Operating Mode RIDE, RIDEREL, ZERO+, ZERO-

- Data-Byte3: Sollposition LSB (Incremente)
- Data-Byte4: Sollposition LSB+1
- Data-Byte5: Sollposition LSB+2
- Data-Byte6: Sollposition MSB
- Data-Byte7: Geschwindigkeit in % der max. Geschwindigkeit (1..100)
- Data-Byte8: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms, 4..1s, 5..2.5s, 6..5s

## Operating-Mode

0	= NOP	keine Aktion
1	= OFF	Regler aus
2	= ON	Regler ein, Sollposition = Istposition, keine Bewegung
3	= RIDE	Absolute Positionierung
4	= RIDEREL	Relative Positionierung
5	= ZERO+	Nullen mit Referenz, vorwärts
6	= ZERO-	Nullen mit Referenz, rückwärts
7	= ZERO2+	Nullen ohne Referenz, vorwärts
8	= ZERO2-	Nullen ohne Referenz, rückwärts
9	= STOP	Fahrbewegung wird gestoppt (mit definierter Rampe)
10	= FSTOP	Fahrbewegung wird gestoppt (schnellst möglich)
11	= OUT2_ON	Ausgang 2 ein
12	= OUT2_OFF	Ausgang 2 aus
13	= MEMCMD	Befehl aus Befehlsliste ausführen

## Slave nach Master

b) Index gültig. Befehl wurde abgespeichert

### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	0	0	0	1	Descriptor-Byte 2

### Nutzdaten:

- Data-Byte1: F2H Command HB

b) Index ungültig. Befehl wurde nicht abgespeichert

### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	0	0	0	1	Descriptor-Byte 2

### Nutzdaten:

- Data-Byte1: FFH Command HB

## 3.4.4.4. Befehl lesen

Der Master liest ein Befehl (Operating-Mode) aus der Befehlsliste. Der Befehl wird unmittelbar nach der Anfrage vom Slave ausgesendet.

### Master nach Slave

#### Descriptor:

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	0	1	Descriptor-Byte 2

#### Nutzdaten:

- Data-Byte1: F3H Command HB

### Slave nach Master

#### a) Daten und Index gültig

#### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	1	0	0	0	Descriptor-Byte 2

#### Nutzdaten:

- Data-Byte1: F3H Command HB
- Data-Byte2: Operating-Mode  
**Operating Mode NOP, OFF, ON, STOP, FSTOP, OUT2\_ON, OUT2\_OFF**
  - Data-Byte3-8: nicht verwendet, leer
- **Operating Mode RIDE, RIDEREL, ZERO+, ZERO-**
  - Data-Byte3: Sollposition LSB (Incremente)
  - Data-Byte4: Sollposition LSB+1
  - Data-Byte5: Sollposition LSB+2
  - Data-Byte6: Sollposition MSB
  - Data-Byte7: Geschwindigkeit in % der max. Geschwindigkeit (1..100)
  - Data-Byte8: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms, 4..1s, 5..2.5s, 6..5s

#### b) Daten oder Index ungültig, Befehl konnte nicht ausgelesen werden

#### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	0	0	0	1	Descriptor-Byte 2

#### Nutzdaten:





- Data-Byte1: FFH Command HB

### 3.4.4.5. Sollposition lesen

Der Master liest aktuelle Sollposition. Die Sollposition wird unmittelbar nach der Anfrage vom Slave ausgesendet.

#### Master nach Slave

##### Descriptor:

0	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	0	0	0	1	Descriptor-Byte 2

##### Nutzdaten:

- Data-Byte1: F4H Command HB

#### Slave nach Master

##### Descriptor:

0	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	0	1	0	1	Descriptor-Byte 2

##### Nutzdaten:

- Data-Byte1: F4H Command HB
- Data-Byte2: Sollposition LSB (Incremente)
- Data-Byte3: Sollposition LSB+1
- Data-Byte4: Sollposition LSB+2
- Data-Byte5: Sollposition MSB

## 3.4.5. Data-Messages

### 3.4.5.1. Write Operation-Mode

Es wird der Operating-Mode (Auftrag) mit den entsprechenden Parametern an das Servo-Modul gesendet.

#### Master nach Slave

#### Descriptor:

1	0	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
1	1	1	0	1	0	0	0	Descriptor-Byte 2

#### Nutzdaten:

- Data-Byte1: Operating-Mode  
**Operating Mode NOP, OFF, ON, STOP, FSTOP, OUT2\_ON, OUT2\_OFF**
- Data-Byte2-7: nicht verwendet, leer
- Data-Byte8: Frame-Counter, 0.. keine zyklische Übertragung, Frame-Übername erfolgt immer.  
1-255.. zyklische Übertragung, Frame-Übernahme erfolgt nur bei Änderung des Frame-Counters.

#### **Operating Mode RIDE, RIDEREL, ZERO+, ZERO-**

- Data-Byte2: Sollposition LSB (Incremente)
- Data-Byte3: Sollposition LSB+1
- Data-Byte4: Sollposition LSB+2
- Data-Byte5: Sollposition MSB
- Data-Byte6: Geschwindigkeit in % der max. Geschwindigkeit (1..100)
- Data-Byte7: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms, 4..1s, 5..2.5s, 6..5s
- Data-Byte8: Frame-Counter, 0.. keine zyklische Übertragung, Frame-Übername erfolgt immer.  
1-255... zyklische Übertragung, Frame-Übernahme erfolgt nur bei Änderung des Frame-Counters.

#### **Operating Mode MEMCMD**

- Data-Byte2: Befehlsnummer in der Tabelle (0-65535) LSB
- Data-Byte3: Befehlsnummer in der Tabelle (0-65535) MSB
- Data-Byte4-7: nicht verwendet
- Data-Byte8: Frame-Counter, 0.. keine zyklische Übertragung, Frame-Übername erfolgt immer.  
1-255... zyklische Übertragung, Frame-Übernahme erfolgt nur bei Änderung des Frame-Counters.



## Operating-Mode

0	= NOP	keine Aktion
1	= OFF	Regler aus
2	= ON	Regler ein, Sollposition = Istposition, keine Bewegung
3	= RIDE	Absolute Positionierung
4	= RIDEREL	Relative Positionierung
5	= ZERO+	Nullen mit Referenz, vorwärts
6	= ZERO-	Nullen mit Referenz, rückwärts
7	= ZERO2+	Nullen ohne Referenz, vorwärts
8	= ZERO2-	Nullen ohne Referenz, rückwärts
9	= STOP	Fahrbewegung wird gestoppt (mit definierter Rampe)
10	= FSTOP	Fahrbewegung wird gestoppt (schnellst möglich)
11	= OUT2_ON	Ausgang 2 ein
12	= OUT2_OFF	Ausgang 2 aus
13	= MEMCMD	Befehl aus Befehlsliste ausführen

### 3.4.5.2. Modul-Status

Es wird der Status des Servo-Moduls an den Master gesendet.

Der Modul-Status wird nach folgenden Regeln gesendet:

- Unmittelbar nach Empfang „Start-Modul-Status Update“
- Poll-Mode: unmittelbar nach Empfang von „Write Operating-Mode“
- Event-Mode: Nach Ausführung des Befehls "Write Operating-Mode" oder bei einem ungültigen Operating-Mode Befehls.

### Slave nach Master

#### Descriptor:

1	1	1	Adr4	Adr3	Adr2	Adr1	Adr0	Descriptor-Byte 1
0	1	0	0	1	0	0	0	Descriptor-Byte 2

#### Nutzdaten:

- Data-Byte1: Modulstatus
- Data-Byte2: Reglerstatus
- Data-Byte3: Fehlernummer
- Data-Byte4: Istposition LSB (Incremente)
- Data-Byte5: Istposition LSB+1
- Data-Byte6: Istposition LSB+2
- Data-Byte7: Istposition MSB
- Data-Byte8: Operating-Mode-Status  
0... letzter Operating-Mode wurde übernommen  
1... warten auf Befehlsausführung(\*)  
2... letzter Operating-Mode nicht akzeptiert

(\*) In diesem Zustand wird mit Write-Operating-Mode ein anstehender Befehl überschrieben.



## Modulstatus

- Bit 0 = Modul im Zustand ERROR (näheres siehe Fehlernummer)
- Bit 1 = System nicht freigegeben (RUN disable) (1 = nicht freigegeben)
- Bit 2 = Endschalter negativ (1 = aktiv)
- Bit 3 = Endschalter positiv (1 = aktiv)
- Bit 4 = Wegachse nicht genullt (1 = nicht genullt)
- Bit 5 = Referenzmarke (1 = aktiv)
- Bit 6 = Ausgang2 (Out2) Status, (1 = aktiv)
- Bit 7 = nicht belegt

## Reglerstatus

- 1 = OFF Regler aus
- 2 = ON Regler ein, keine Bewegung
- 3 = RIDE Absolute Positionierung
- 4 = RIDEREL Relative Positionierung
- 5 = ZERO+ Nullen mit Referenz, vorwärts
- 6 = ZERO- Nullen mit Referenz, rückwärts
- 7 = ZERO2+ Nullen ohne Referenz, vorwärts
- 8 = ZERO2- Nullen ohne Referenz, rückwärts
- 9 = STOP Fahrbewegung wird gestoppt (mit definierter Rampe)
- 10 = FSTOP Fahrbewegung wird gestoppt (schnellst möglich)
- 14 = MANUAL Handbetrieb
- 20 = INIT Regler Initialisierung
- 21 = INITEND Ende der Regler Initialisierung
- 22 = SAVE Parameter werden in EEPROM gespeichert
- 23 = BLOCKED Regler blockiert (Endschalter oder keine Freigabe)
- 255 = ERROR Regler-Fehler, nur mit Befehl OFF wird Zustand verlassen

## Fehlernummer

- 0 = kein Fehler
- 1 = Servoverstärker-Fehler
- 2 = Positionsgeber Fehler (Overrun)
- 3 = Reglerabweichung zu groß
- 4 = Checksummenfehler im batteriegepufferten RAM
- 5 = Checksummenfehler im E<sup>2</sup>PROM

# 4. RS232-Kommunikation

## 4.1. Allgemeines

### 4.1.1. Schnittstelle

RS-232 Schnittstelle, serielle Übertragung ohne Handshake, 2400 Baud, 8 Datenbits, kein Paritätsbit, 1 Stopbit

### 4.1.2. Telegrammstruktur

Zwischen PC und Servo-Modul wird mit einem blockorientierten Protokoll über die RS232-Schnittstelle kommuniziert. Der PC ist der Master und das Servo-Modul der Slave.

Jede Kommunikation wird vom PC aus mit dem Sendetelegramm gestartet. Das I/O-Modul beantwortet jedes Sendetelegramm mit dem Antworttelegramm, bevor ein neues Sendetelegramm vom PC abgesetzt wird. Das Antworttelegramm muß innerhalb einer einstellbaren Timeoutzeit eintreffen

Bei einem Fehler oder Timeout versucht der PC das gleiche Sendetelegramm 5mal hintereinander abzusetzen, bis die Übertragung abgebrochen und eine Fehlermeldung ausgelöst wird.

#### Sendetelegramm:

adrlo	adrhi	bid	arg1	arg2	lenlo	lenhi	Datenblock	cs
-------	-------	-----	------	------	-------	-------	------------	----

<b>adrlo:</b>	Zieladresse (Low-Byte):					1		
<b>adrhi:</b>	Zieladresse (High-Byte):					0		
<b>bid:</b>	Block-ID						0..255, Beginnt mit 0 und wird mit jedem neuen Telegramm inkrementiert.	
<b>arg1:</b>	Telegrammart					1...x		
<b>arg2:</b>	nicht benutzt					0		
<b>lenlo:</b>	Länge des Datenblockes (LB)					0..100		
<b>lenhi:</b>	Länge des Datenblockes (HB)					0		
<b>Datenblock:</b>	siehe Telegrammart							
<b>cs:</b>	Checksumme, XOR über Telegrammkopf und Datenblock.							



## Antworttelegramm:

adrlo	adrhi	bid	arg1	arg2	lenlo	lenhi	Datenblock	cs
-------	-------	-----	------	------	-------	-------	------------	----

- adrlo:** Zieladresse (Low-Byte): 1
- adrhi:** Zieladresse (High-Byte): 0
- bid:** Block-ID 0..255, Nummer des Sendetelegramms.
- arg1:** Telegrammart
- arg2:** Kurzstatus  
Bit 0: ungültiges Telegramm  
Bit 1: Übertragungsfehler Sendetelegramm  
Bit 2: aktueller Reglerstatus erlaubt Sendetelegramm nicht  
Bit 3: Modul im Fehler-Zustand  
Bit 4: 0  
Bit 5: 0  
Bit 6: 0  
Bit 7: 0
- lenlo:** Länge des Datenblockes (LB) 0..100
- lenhi:** Länge des Datenblockes (HB) 0
- Datenblock:** siehe Telegrammarten
- cs:** Checksumme, XOR über Telegrammkopf und Datenblock.



### 4.1.3. Read Identifikation

Es wird die Identifikation des Slaves gelesen.

#### Master nach Slave

- Telegrammart (arg1): 1
- Datenblock: keine Daten

#### Slave nach Master

- Telegrammart (arg1): 1
- Data-Byte1: Modul-Gruppe (8)
- Data-Byte2: Modul-Variante (1..x)
- Data-Byte3: Software-Version (1..99)
- Data-Byte4: Software-Revision.(0..99)

#### **Modul-Variante**

CAN Protokoll (Einer-Stelle):

- x1 = Murrelektronik
- x2 = Selectron DP

Antriebsart (Zehner-Stelle):

- 0x = DC-Servo-Antrieb mit Inkrementalgeber
- 1x = .....

### 4.1.4. Read Status

Es wird der Status des Slaves abgefragt.

#### Master nach Slave

- Telegrammart (arg1): 2
- Datenblock: keine Daten

#### Slave nach Master

- Telegrammart (arg1): 2
- Data-Byte1: Modulstatus
- Data-Byte2: Reglerstatus
- Data-Byte3: Fehlernummer
- Data-Byte4: Istposition LSB (Incremente)
- Data-Byte5: Istposition LSB+1
- Data-Byte6: Istposition LSB+2
- Data-Byte7: Istposition MSB
- Data-Byte8: Wegänderung (Inc) in 50ms LB (für Drehzahlberechnung)
- Data-Byte9: Wegänderung (Inc) in 50ms HB
- Data-Byte10: Fuzzy-Regler-Eingang 1, Positionsabweichung (ds = Soll-Sist) LSB



- Data-Byte11: Fuzzy-Regler-Eingang 1, Positionsabweichung MSB
- Data-Byte12: Fuzzy-Regler-Eingang 2, Änderung der Positionsabweichung (ds/dt)
- Data-Byte13: Fuzzy-Regler-Ausgang, Stellgröße LB
- Data-Byte14: Fuzzy-Regler-Ausgang, Stellgröße HB (0..-100%, 32767..+100%)
- Data-Byte15: Gesamte Stellgröße LB
- Data-Byte16: Gesamte Stellgröße HB (0..-100%, 32767..+100%)
- Data-Byte17: Fuzzy-Regler-Ausgang, Singleton1, Anteil (0..255)
- Data-Byte18: Fuzzy-Regler-Ausgang, Singleton2, Anteil (0..255)
- Data-Byte19: Fuzzy-Regler-Ausgang, Singleton3, Anteil (0..255)
- Data-Byte20: Fuzzy-Regler-Ausgang, Singleton4, Anteil (0..255)
- Data-Byte21: Fuzzy-Regler-Ausgang, Singleton5, Anteil (0..255)

## Modulstatus

- Bit 0 = Modul im Zustand ERROR (näheres siehe Fehlernummer)
- Bit 1 = System nicht freigegeben (RUN disable) (1 = nicht freigegeben)
- Bit 2 = Endschalter negativ (1 = aktiv)
- Bit 3 = Endschalter positiv (1 = aktiv)
- Bit 4 = Wegachse nicht genullt (1 = nicht genullt)
- Bit 5 = Referenzmarke (1 = aktiv)
- Bit 6 = Ausgang2 (Out2) Status, (1 = aktiv)
- Bit 7 = nicht belegt

## Reglerstatus

- 1 = OFF Regler aus
- 2 = ON Regler ein, keine Bewegung
- 3 = RIDE Absolute Positionierung
- 4 = RIDEREL Relative Positionierung
- 5 = ZERO+ Nullen mit Referenz, vorwärts
- 6 = ZERO- Nullen mit Referenz, rückwärts
- 7 = ZERO2+ Nullen ohne Referenz, vorwärts
- 8 = ZERO2- Nullen ohne Referenz, rückwärts
- 9 = STOP Fahrbewegung wird gestoppt (mit definierter Rampe)
- 10 = FSTOP Fahrbewegung wird gestoppt (schnellst möglich)
- 14 = MANUAL Handbetrieb
- 20 = INIT Regler Initialisierung
- 21 = INITEND Ende Regler Initialisierung
- 22 = SAVE Parameter werden in EEPROM gespeichert
- 23 = BLOCKED Regler blockiert (Endschalter oder keine Freigabe)
- 255 = ERROR Regler-Fehler, nur mit Befehl OFF wird Zustand verlassen

## Fehlernummer

- 0 = kein Fehler
- 1 = Servoverstärker-Fehler
- 2 = Positionsgeber Fehler (Overrun)
- 3 = Reglerabweichung zu groß
- 4 = Checksummenfehler im batteriegepufferten RAM
- 5 = Checksummenfehler im E<sup>2</sup>PROM





### 4.1.5. Write Operating-Mode

Der Operating-Mode (Kommando) mit den entsprechenden Parametern wird gesendet.

#### Master nach Slave

- Telegrammart (arg1): 3
- Data-Byte1: Operating-Mode  
***Operating Mode NOP, OFF, ON, STOP, FSTOP, OUT2\_ON, OUT2\_OFF***

- Data-Byte2-7: nicht verwendet

#### ***Operating Mode RIDE, ZERO+, ZERO-***

- Data-Byte2: Sollposition LSB (Incremente)
- Data-Byte3: Sollposition LSB+1
- Data-Byte4: Sollposition LSB+2
- Data-Byte5: Sollposition MSB
- Data-Byte6: Geschwindigkeit in % der max. Geschwindigkeit (1..100)
- Data-Byte7: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms, 4..1s, 5..2.5s, 6..5s

#### ***Operating Mode MEMCMD***

- Data-Byte2: Befehlsnummer in der Tabelle (0-65535) LSB
- Data-Byte3: Befehlsnummer in der Tabelle (0-65535) MSB
- Data-Byte4-7: nicht verwendet

#### Slave nach Master

- Telegrammart (arg1): 3
- Datenblock: keine Daten

#### **Operating-Mode**

0	= NOP	keine Aktion
1	= OFF	Regler aus
2	= ON	Regler ein, Sollposition = Istposition, keine Bewegung
3	= RIDE	Absolute Positionierung
4	= RIDEREL	Relative Positionierung
5	= ZERO+	Nullen mit Referenz, vorwärts
6	= ZERO-	Nullen mit Referenz, rückwärts
7	= ZERO2+	Nullen ohne Referenz, vorwärts
8	= ZERO2-	Nullen ohne Referenz, rückwärts
9	= STOP	Fahrbewegung wird gestoppt (mit definierter Rampe)
10	= FSTOP	Fahrbewegung wird gestoppt (schnellst möglich)
11	= OUT2_ON	Ausgang 2 ein
12	= OUT2_OFF	Ausgang 2 aus
13	= MEMCMD	Befehl aus Befehlsliste ausführen
14	= MANUAL	Handbetrieb



## 4.1.6. Write Parameter

Es werden Parameter an den Slave gesendet. Die Parameter werden nur im Reglerstatus OFF angenommen.

### Master nach Slave

- Telegrammart (arg1): 4
- Data-Byte1: Maximal erlaubte Reglerabweichung LSB (Incr)
- Data-Byte2: " MSB
- Data-Byte3: max. Drehzahl LSB (1..65535 Umdr./min)
- Data-Byte4: max. Drehzahl MSB
- Data-Byte5: Geberauflösung (Incr/Umdr.) LSB (100..65535 Inc/Umdr.)
- Data-Byte6: Geberauflösung (Incr/Umdr.) MSB
- Data-Byte7: Zählrichtung Encoder (0 = vorwärts, 1 = rückwärts)
- Data-Byte8: Richtung Positionsregler (0 = vorwärts, 1 = rückwärts)
- Data-Byte9: Richtung Geschwindigkeitsregler (0 = vorwärts, 1 = rückwärts)
- Data-Byte10: Integratorzeit (0..255)
- Data-Byte11: Steller (0 = Aus, 1 = Ein)
- Data-Byte12: Fuzzy-Regelfreigabe (Regel 1..8)
- Data-Byte13: Fuzzy-Regelfreigabe (Regel 9..16)
- Data-Byte14: Fuzzy-Regler-Eingang1, Term1, Punkt2, x-Wert (0..255)
- Data-Byte15: Fuzzy-Regler-Eingang1, Term1, Punkt3, x-Wert (0..255)
- Data-Byte16: Fuzzy-Regler-Eingang1, Term2, Punkt1, x-Wert (0..255)
- Data-Byte17: Fuzzy-Regler-Eingang1, Term2, Punkt2, x-Wert (0..255)
- Data-Byte18: Fuzzy-Regler-Eingang1, Term2, Punkt3, x-Wert (0..255)
- Data-Byte19: Fuzzy-Regler-Eingang1, Term3, Punkt1, x-Wert (0..255)
- Data-Byte20: Fuzzy-Regler-Eingang1, Term3, Punkt2, x-Wert (0..255)
- Data-Byte21: Fuzzy-Regler-Eingang1, Term3, Punkt3, x-Wert (0..255)
- Data-Byte22: Fuzzy-Regler-Eingang1, Term4, Punkt1, x-Wert (0..255)
- Data-Byte23: Fuzzy-Regler-Eingang1, Term4, Punkt2, x-Wert (0..255)
- Data-Byte24: Fuzzy-Regler-Eingang1, Term4, Punkt3, x-Wert (0..255)
- Data-Byte25: Fuzzy-Regler-Eingang1, Term5, Punkt1, x-Wert (0..255)
- Data-Byte26: Fuzzy-Regler-Eingang1, Term5, Punkt2, x-Wert (0..255)
- Data-Byte27: Fuzzy-Regler-Eingang2, Term1, Punkt2, x-Wert (0..255)
- Data-Byte28: Fuzzy-Regler-Eingang2, Term1, Punkt3, x-Wert (0..255)
- Data-Byte29: Fuzzy-Regler-Eingang2, Term2, Punkt1, x-Wert (0..255)
- Data-Byte30: Fuzzy-Regler-Eingang2, Term2, Punkt2, x-Wert (0..255)
- Data-Byte31: Fuzzy-Regler-Eingang2, Term2, Punkt3, x-Wert (0..255)
- Data-Byte32: Fuzzy-Regler-Eingang2, Term3, Punkt1, x-Wert (0..255)
- Data-Byte33: Fuzzy-Regler-Eingang2, Term3, Punkt2, x-Wert (0..255)
- Data-Byte34: Fuzzy-Regler-Ausgang, Singleton1, x-Wert LB
- Data-Byte35: Fuzzy-Regler-Ausgang, Singleton1, x-Wert HB (0..32767)
- Data-Byte36: Fuzzy-Regler-Ausgang, Singleton2, x-Wert LB
- Data-Byte37: Fuzzy-Regler-Ausgang, Singleton2, x-Wert HB (0..32767)
- Data-Byte38: Fuzzy-Regler-Ausgang, Singleton3, x-Wert LB



- Data-Byte39: Fuzzy-Regler-Ausgang, Singleton3, x-Wert HB (0..32767)
- Data-Byte40: Fuzzy-Regler-Ausgang, Singleton4, x-Wert LB
- Data-Byte41: Fuzzy-Regler-Ausgang, Singleton4, x-Wert HB (0..32767)
- Data-Byte42: Fuzzy-Regler-Ausgang, Singleton5, x-Wert LB
- Data-Byte43: Fuzzy-Regler-Ausgang, Singleton5, x-Wert HB (0..32767)
- Data-Byte44: Fuzzy-Regel 1, Ausgangsterm (0-4)
- Data-Byte45: Fuzzy-Regel 2, Ausgangsterm (0-4)
- Data-Byte46: Fuzzy-Regel 3, Ausgangsterm (0-4)
- Data-Byte47: Fuzzy-Regel 4, Ausgangsterm (0-4)
- Data-Byte48: Fuzzy-Regel 5, Ausgangsterm (0-4)
- Data-Byte49: Fuzzy-Regel 6, Ausgangsterm (0-4)
- Data-Byte50: Fuzzy-Regel 7, Ausgangsterm (0-4)
- Data-Byte51: Fuzzy-Regel 8, Ausgangsterm (0-4)
- Data-Byte52: Fuzzy-Regel 9, Ausgangsterm (0-4)
- Data-Byte53: Fuzzy-Regel 10, Ausgangsterm (0-4)
- Data-Byte54: Fuzzy-Regel 11, Ausgangsterm (0-4)
- Data-Byte55: Fuzzy-Regel 12, Ausgangsterm (0-4)
- Data-Byte56: Fuzzy-Regel 13, Ausgangsterm (0-4)
- Data-Byte57: Fuzzy-Regel 14, Ausgangsterm (0-4)
- Data-Byte58: Fuzzy-Regel 15, Ausgangsterm (0-4)
- Data-Byte59: CAN-Übertragungsrate
- Data-Byte60: CAN-Adresse (0..29)
- Data-Byte61: Offsetkorrektur LB
- Data-Byte62: Offsetkorrektur HB (-999..+999)

## **Slave nach Master**

- Telegrammart (arg1): 4
- Datenblock: keine Daten

## **Ausgangsterme**

0 = negativ groß  
1 = negativ mittel  
2 = null  
3 = pos mittel  
4 = pos groß

## **CAN-Übertragungsrate**

0 = 20kBit/s  
1 = 50kBit/s  
2 = 100kBit/s  
3 = 125kBit/s  
4 = 250kBit/s  
5 = 500kBit/s  
6 = 1 MBit/s



#### 4.1.7. Read Parameter

Es werden Parameter des Slaves ausgelesen.

##### Master nach Slave

- Telegrammart (arg1): 5
- Datenblock: keine Daten

##### Slave nach Master

- Telegrammart (arg1): 5
- Data-Byte1: Maximal erlaubte Reglerabweichung LSB (Incr)
- Data-Byte2: " MSB
- Data-Byte3: max. Drehzahl LSB (1..65535 Umdr./min)
- Data-Byte4: max. Drehzahl MSB
- Data-Byte5: Geberauflösung (Inc/Umdr.) LSB (100..65535 Inc/Umdr.)
- Data-Byte6: Geberauflösung (Inc/Umdr.) MSB
- Data-Byte7: Zählrichtung Encoder (0 = vorwärts, 1 = rückwärts)
- Data-Byte8: Richtung Positionsregler (0 = vorwärts, 1 = rückwärts)
- Data-Byte9: Richtung Geschwindigkeitsregler (0 = vorwärts, 1 = rückwärts)
- Data-Byte10: Integratorzeit (0..255)
- Data-Byte11: Steller (0 = Aus, 1 = Ein)
- Data-Byte12: Fuzzy-Regelfreigabe (Regel 1..8)
- Data-Byte13: Fuzzy-Regelfreigabe (Regel 9..16)
- Data-Byte14: Fuzzy-Regler-Eingang1, Term1, Punkt2, x-Wert (0..255)
- Data-Byte15: Fuzzy-Regler-Eingang1, Term1, Punkt3, x-Wert (0..255)
- Data-Byte16: Fuzzy-Regler-Eingang1, Term2, Punkt1, x-Wert (0..255)
- Data-Byte17: Fuzzy-Regler-Eingang1, Term2, Punkt2, x-Wert (0..255)
- Data-Byte18: Fuzzy-Regler-Eingang1, Term2, Punkt3, x-Wert (0..255)
- Data-Byte19: Fuzzy-Regler-Eingang1, Term3, Punkt1, x-Wert (0..255)
- Data-Byte20: Fuzzy-Regler-Eingang1, Term3, Punkt2, x-Wert (0..255)
- Data-Byte21: Fuzzy-Regler-Eingang1, Term3, Punkt3, x-Wert (0..255)
- Data-Byte22: Fuzzy-Regler-Eingang1, Term4, Punkt1, x-Wert (0..255)
- Data-Byte23: Fuzzy-Regler-Eingang1, Term4, Punkt2, x-Wert (0..255)
- Data-Byte24: Fuzzy-Regler-Eingang1, Term4, Punkt3, x-Wert (0..255)
- Data-Byte25: Fuzzy-Regler-Eingang1, Term5, Punkt1, x-Wert (0..255)
- Data-Byte26: Fuzzy-Regler-Eingang1, Term5, Punkt2, x-Wert (0..255)
- Data-Byte27: Fuzzy-Regler-Eingang2, Term1, Punkt2, x-Wert (0..255)
- Data-Byte28: Fuzzy-Regler-Eingang2, Term1, Punkt3, x-Wert (0..255)
- Data-Byte29: Fuzzy-Regler-Eingang2, Term2, Punkt1, x-Wert (0..255)
- Data-Byte30: Fuzzy-Regler-Eingang2, Term2, Punkt2, x-Wert (0..255)
- Data-Byte31: Fuzzy-Regler-Eingang2, Term2, Punkt3, x-Wert (0..255)
- Data-Byte32: Fuzzy-Regler-Eingang2, Term3, Punkt1, x-Wert (0..255)
- Data-Byte33: Fuzzy-Regler-Eingang2, Term3, Punkt2, x-Wert (0..255)
- Data-Byte34: Fuzzy-Regler-Ausgang, Singleton1, x-Wert LB



- Data-Byte35: Fuzzy-Regler-Ausgang, Singleton1, x-Wert HB (0..32767)
- Data-Byte36: Fuzzy-Regler-Ausgang, Singleton2, x-Wert LB
- Data-Byte37: Fuzzy-Regler-Ausgang, Singleton2, x-Wert HB (0..32767)
- Data-Byte38: Fuzzy-Regler-Ausgang, Singleton3, x-Wert LB
- Data-Byte39: Fuzzy-Regler-Ausgang, Singleton3, x-Wert HB (0..32767)
- Data-Byte40: Fuzzy-Regler-Ausgang, Singleton4, x-Wert LB
- Data-Byte41: Fuzzy-Regler-Ausgang, Singleton4, x-Wert HB (0..32767)
- Data-Byte42: Fuzzy-Regler-Ausgang, Singleton5, x-Wert LB
- Data-Byte43: Fuzzy-Regler-Ausgang, Singleton5, x-Wert HB (0..32767)
- Data-Byte44: Fuzzy-Regel 1, Ausgangsterm (0-4)
- Data-Byte45: Fuzzy-Regel 2, Ausgangsterm (0-4)
- Data-Byte46: Fuzzy-Regel 3, Ausgangsterm (0-4)
- Data-Byte47: Fuzzy-Regel 4, Ausgangsterm (0-4)
- Data-Byte48: Fuzzy-Regel 5, Ausgangsterm (0-4)
- Data-Byte49: Fuzzy-Regel 6, Ausgangsterm (0-4)
- Data-Byte50: Fuzzy-Regel 7, Ausgangsterm (0-4)
- Data-Byte51: Fuzzy-Regel 8, Ausgangsterm (0-4)
- Data-Byte52: Fuzzy-Regel 9, Ausgangsterm (0-4)
- Data-Byte53: Fuzzy-Regel 10, Ausgangsterm (0-4)
- Data-Byte54: Fuzzy-Regel 11, Ausgangsterm (0-4)
- Data-Byte55: Fuzzy-Regel 12, Ausgangsterm (0-4)
- Data-Byte56: Fuzzy-Regel 13, Ausgangsterm (0-4)
- Data-Byte57: Fuzzy-Regel 14, Ausgangsterm (0-4)
- Data-Byte58: Fuzzy-Regel 15, Ausgangsterm (0-4)
- Data-Byte59: CAN-Übertragungsrate
- Data-Byte60: CAN-Adresse (0..29)
- Data-Byte61: Offsetkorrektur LB
- Data-Byte62: Offsetkorrektur HB (-999..+999)



#### 4.1.8. Save Parameter

Die momentan aktuellen Parameter werden im E<sup>2</sup>PROM gespeichert. Dieser Befehl ist nur im Zustand OFF erlaubt.

##### Master nach Slave

- Telegrammart (arg1): 6
- Datenblock: keine Daten

##### Slave nach Master

- Telegrammart (arg1): 6
- Datenblock: keine Daten



## 4.1.9. Write Memory-Command

Ein beliebiger Befehl (Operating-Mode) wird als Eintrag einer Befehlsliste im batteriegepufferten RAM gespeichert, jedoch nicht ausgeführt. Die einzelnen Befehle der Liste sind durchnummeriert und können über CAN oder RS232 abgerufen werden (Operating-Mode MEMCMD). Dieser Befehl ist nur im Zustand OFF erlaubt.

### Master nach Slave

- Telegrammart (arg1): 7
- Data-Byte1: Tabellenindex LSB
- Data-Byte2: Tabellenindex MSB
- Data-Byte3: Operating-Mode

#### ***Operating Mode NOP, OFF, ON, STOP, FSTOP, OUT2\_ON, OUT2\_OFF***

- Data-Byte4-9: nicht verwendet

#### ***Operating Mode RIDE, ZERO+, ZERO-***

- Data-Byte4: Sollposition LSB (Incremente)
- Data-Byte5: Sollposition LSB+1
- Data-Byte6: Sollposition LSB+2
- Data-Byte7: Sollposition MSB
- Data-Byte8: Geschwindigkeit in % der max. Geschwindigkeit (1..100)
- Data-Byte9: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms, 4..1s, 5..2.5s, 6..5s

### Slave nach Master

- Telegrammart (arg1): 7
- Datenblock: keine Daten

### **Operating-Mode**

0	= NOP	keine Aktion
1	= OFF	Regler aus
2	= ON	Regler ein, Sollposition = Istposition, keine Bewegung
3	= RIDE	Absolute Positionierung
4	= RIDEREL	Relative Positionierung
5	= ZERO+	Nullen mit Referenz, vorwärts
6	= ZERO-	Nullen mit Referenz, rückwärts
7	= ZERO2+	Nullen ohne Referenz, vorwärts
8	= ZERO2-	Nullen ohne Referenz, rückwärts
9	= STOP	Fahrbewegung wird gestoppt (mit definierter Rampe)
10	= FSTOP	Fahrbewegung wird gestoppt (schnellst möglich)
11	= OUT2_ON	Ausgang 2 ein
12	= OUT2_OFF	Ausgang 2 aus



#### 4.1.10. Read Memory-Command

Ein bestimmter Befehl, welcher über den Index selektiert wird, wird aus der Befehlsliste gelesen.

##### Master nach Slave

- Telegrammart (arg1): 8
- Data-Byte1: Tabellenindex LSB
- Data-Byte2: Tabellenindex MSB

##### Slave nach Master

- Telegrammart (arg1): 8
- Data-Byte1: Operating-Mode

***Operating Mode NOP, OFF, ON, STOP, FSTOP, OUT2\_ON, OUT2\_OFF***

- Data-Byte6-11: nicht verwendet

***Operating Mode RIDE, ZERO+, ZERO-***

- Data-Byte6: Sollposition LSB (Incremente)
- Data-Byte7: Sollposition LSB+1
- Data-Byte8: Sollposition LSB+2
- Data-Byte9: Sollposition MSB
- Data-Byte10: Geschwindigkeit in % der max. Geschwindigkeit (1..100)
- Data-Byte11: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms, 4..1s, 5..2.5s, 6..5s

##### **Operating-Mode**

0	= NOP	keine Aktion
1	= OFF	Regler aus
2	= ON	Regler ein, Sollposition = Istposition, keine Bewegung
3	= RIDE	Absolute Positionierung
4	= RIDEREL	Relative Positionierung
5	= ZERO+	Nullen mit Referenz, vorwärts
6	= ZERO-	Nullen mit Referenz, rückwärts
7	= ZERO2+	Nullen ohne Referenz, vorwärts
8	= ZERO2-	Nullen ohne Referenz, rückwärts
9	= STOP	Fahrbewegung wird gestoppt (mit definierter Rampe)
10	= FSTOP	Fahrbewegung wird gestoppt (schnellst möglich)
11	= OUT2_ON	Ausgang 2 ein
12	= OUT2_OFF	Ausgang 2 aus





#### 4.1.11. Read Fuzzy-Status

Liest den Wirkungsgrad der einzelnen Fuzzy-Regeln (0-100%) für die ONLINE-Anzeige.

##### Master nach Slave

- Telegrammart (arg1): 9
- Datenblock: keine Daten

##### Slave nach Master

- Telegrammart (arg1): 9
- Data-Byte1: Wirkungsgrad Fuzzy-Regel 1 (0-100%)
- Data-Byte2: Wirkungsgrad Fuzzy-Regel 2 (0-100%)
- Data-Byte3: Wirkungsgrad Fuzzy-Regel 3 (0-100%)
- Data-Byte4: Wirkungsgrad Fuzzy-Regel 4 (0-100%)
- Data-Byte5: Wirkungsgrad Fuzzy-Regel 5 (0-100%)
- Data-Byte6: Wirkungsgrad Fuzzy-Regel 6 (0-100%)
- Data-Byte7: Wirkungsgrad Fuzzy-Regel 7 (0-100%)
- Data-Byte8: Wirkungsgrad Fuzzy-Regel 8 (0-100%)
- Data-Byte9: Wirkungsgrad Fuzzy-Regel 9 (0-100%)
- Data-Byte10: Wirkungsgrad Fuzzy-Regel 10 (0-100%)
- Data-Byte11: Wirkungsgrad Fuzzy-Regel 11 (0-100%)
- Data-Byte12: Wirkungsgrad Fuzzy-Regel 12 (0-100%)
- Data-Byte13: Wirkungsgrad Fuzzy-Regel 13 (0-100%)
- Data-Byte14: Wirkungsgrad Fuzzy-Regel 14 (0-100%)
- Data-Byte15: Wirkungsgrad Fuzzy-Regel 15 (0-100%)

##### **Die Fuzzy-Regeln:**

Nr.		ds (IstPos-		Tendenz von ds		Ausgang
1	Wenn	pos. groß	und	neg. groß	dan	konfigurierbar
2	Wenn	pos. groß	und	null	dan	konfigurierbar
3	Wenn	pos. groß	und	pos. groß	dan	konfigurierbar
4	Wenn	pos. mittel	und	neg. groß	dan	konfigurierbar
5	Wenn	pos. mittel	und	null	dan	konfigurierbar
6	Wenn	pos. mittel	und	pos. groß	dan	konfigurierbar
7	Wenn	Null	und	neg. groß	dan	konfigurierbar
8	Wenn	Null	und	null	dan	konfigurierbar
9	Wenn	Null	und	pos. groß	dan	konfigurierbar
10	Wenn	neg. mittel	und	neg. groß	dan	konfigurierbar
11	Wenn	neg. mittel	und	null	dan	konfigurierbar
12	Wenn	neg. mittel	und	pos. groß	dan	konfigurierbar
13	Wenn	neg.groß	und	neg. groß	dan	konfigurierbar
14	Wenn	neg.groß	und	null	dan	konfigurierbar
15	Wenn	neg.groß	und	pos. groß	dan	konfigurierbar

#### 4.1.12. Write Assist Parameter

Die Parameter für das Einstellungsprogramm (Handbetrieb) werden an den Regler gesendet. Dieser Frame kann nur im Zustand OFF gesendet werden.

##### Master nach Slave

- Telegrammart (arg1): 10
- Data-Byte1: Geschwindigkeit in % der max. Geschwindigkeit (0-200)
 

0	...	-100%
100	...	0%
200	...	+100%
- Data-Byte2: Rampenzeit: 0..50ms, 1..100ms, 2..250ms, 3..500ms, 4..1s, 5..2.5s, 6..5s
- Data-Byte3: max. Drehzahl LSB (1..65535 Umdr./min)
- Data-Byte4: max. Drehzahl MSB
- Data-Byte5: Geberauflösung (Inc/Umdr.) LSB (100..65535 Inc/Umdr.)
- Data-Byte6: Geberauflösung (Inc/Umdr.) MSB
- Data-Byte7: Zählrichtung Encoder (0 = vorwärts, 1 = rückwärts)
- Data-Byte8: Richtung Positionsregler (0 = vorwärts, 1 = rückwärts)
- Data-Byte9: Richtung Geschwindigkeitsregler (0 = vorwärts, 1 = rückwärts)
- Data-Byte10: Offsetkorrektur LB
- Data-Byte11: Offsetkorrektur HB (-999..+999)

##### Slave nach Master

- Telegrammart (arg1): 10
- Datenblock: keine Daten

## 5. Servo-Verstärker

---

### 5.1. Anwendung

Das MBS-SD - Modul ist eine komplette Einheit bestehend aus:

Positioniersteuerung mit CAN- und RS232-Anschluss und dem Servoverstärker.

Das Modul ist für Positioniervorgänge mit Hydraulikservoventilen, Proportionalventilen und Gleichstromservomotoren verwendbar.

*Einsatzbereich:* Die MBS-SD-Module sind für schnelle Positioniervorgänge, Zustellungsabläufe in mehreren Achsen und Handlingfunktionen über die CAN-Schnittstelle auch bei Mehrachsfunken bestens geeignet.

---

### 5.2. Funktion

Über die serielle Schnittstelle kann der Motor Software-geführt konfiguriert werden. Am Modul bestehen noch Möglichkeiten zur Feinabstimmung des Servoantriebs. Dieser Teil wird im Kapitel „Inbetriebnahme“ genauer beschrieben.

Der MBS-SD Servoverstärker ist ein im 4-Quadranten-Schaltbetrieb arbeitender Gleichstrom-Servoverstärker, der im Leistungsbereich von bis zu 300 Watt eingesetzt werden kann.

Im Netzteil werden die Versorgungsspannung, die interne Betriebsspannung und die Kühlkörpertemperatur überwacht. In jedem Leistungsteil wird zusätzlich der Kurzschluss-Strom überwacht. Das Modul stellt externe Eingänge für Endschalter, Sicherheitsfunktionen und Überwachungsfunktionen zur Verfügung.

Integrierte Schutzschaltungen sperren die Endstufe bei unzulässigem Betrieb.

Jeder unzulässige Zustand wird mit einer Fehlermeldung angezeigt.

Der Ausgang des MBS-SD Servoverstärker ist kurzschlussfest, und zudem auch gegen Kurzschluss zwischen Ausgang und Masse (0V) geschützt.

Eine Beschreibung des CAN - Schnittstellenprotokolls wird auf einer Diskette mitgeliefert.

## 6. Inbetriebnahme

### 6.1. Voraussetzungen

Für die Inbetriebnahme eines MBS - Moduls sind folgende Geräte notwendig:

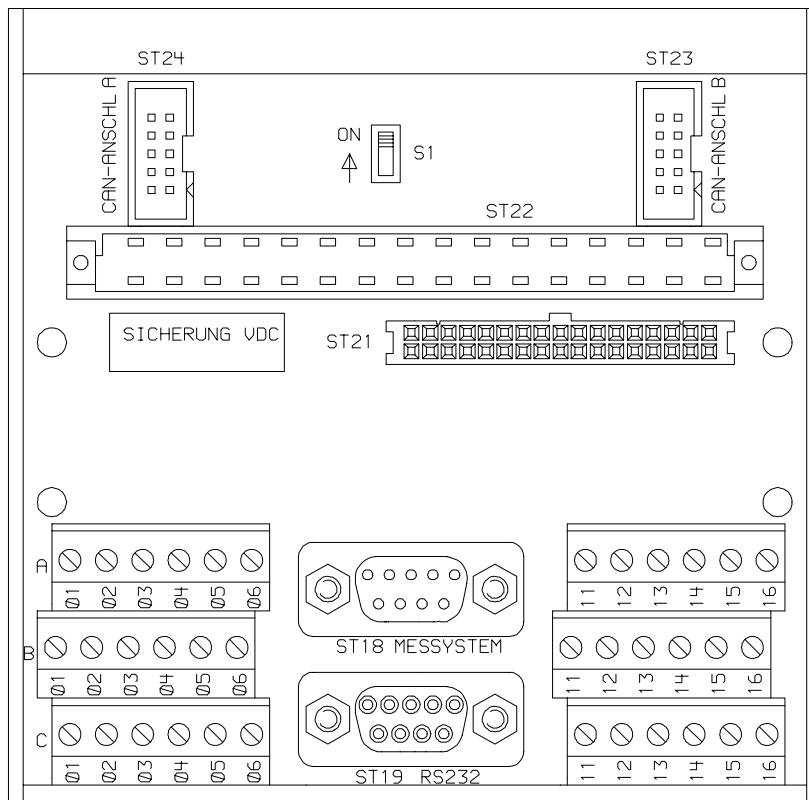
- Externes Netzteil
- Motor mit Inkrementgeber
- PC inkl. MBS-SD Servo - Software

### 6.2. Anschlüsse

Zum Anschluss der Versorgungsspannungen sowie der Ein- und Ausgänge, sind zwei Klemmenblöcke zu je 18 Pole vorgesehen. Sie sind in Schraub- oder Federzugkrafttechnik erhältlich.

Bei Verwendung von Schraubklemmen sollten die Kabelenden 10mm, bei Federzugklemmen 11.5mm aboliert werden. Auf einen ausreichenden Kabelquerschnitt ist zu achten!

Die Klemmen sind direkt bezeichnet.



### 6.2.1. Anschlußbelegung Klemmenblock *links*

	REIHE A	REIHE B	REIHE C
01	VAC1	VAC2	ERDE
02	SPEISUNG POS	SPEISUNG NEG	ERDE
03	MOTOR ANSCHLUSS A	MOTOR ANSCHLUSS B	ERDE
04	MOTOR ANSCHLUSS A	MOTOR ANSCHLUSS B	ERDE
05	TACHO IN/OUT	NC	ELEKTRONIK-GND
06	SOLLWERT POS	SOLLWERT NEG	ELEKTRONIK-GND

Beim Anschluß muß darauf geachtet werden, daß der PE-Anschluß niederohmig mit dem Schutzleiter verbunden wird.

### 6.2.2. Anschlußbelegung Klemmenblock *rechts*

	REIHE A	REIHE B	REIHE C
01	+24VDC (SPS)	EINGANG 1 (FREIGABE)	GND (SPS)
02	+24VDC (SPS)	EINGANG 2 (REFERENZ-SCHALTER)	GND (SPS)
03	+24VDC (SPS)	EINGANG 3 (ENDSCHALTER -)	GND (SPS)
04	+24VDC (SPS)	EINGANG 4 (ENDSCHALTER +)	GND (SPS)
05	+24VDC (SPS)	AUSGANG 1 (SYSTEM AKTIV)	GND (SPS)
06	+24VDC (SPS)	AUSGANG 2 (ANWENDER)	GND (SPS)

### 6.2.3. Anschluss Mess-System

*D-Sub 9 Stift*

Pin 1	VDD (+5VDC AUSGANG)	Pin 4	SIGNAL C	Pin 7	SIGNAL C INV.
Pin 2	SIGNAL A	Pin 5	SIGNAL A INV.	Pin 8	NC
Pin 3	SIGNAL B	Pin 6	SIGNAL B INV.	Pin 9	ELEKTRONIK-GND

### 6.2.4. Anschluss RS232

*D-Sub 9 Buchse*

Pin 1	NC	Pin 4	NC	Pin 7	INT. VERB. PIN8
Pin 2	TXD	Pin 5	RS232 GND	Pin 8	INT. VERB. PIN7
Pin 3	RXD	Pin 6	NC	Pin 9	NC

### 6.2.5. Anschluss CAN

*Flachband Siftleiste 10 polig*

Pin 1	NC	Pin 6	CAN-SIGNAL HIGH
Pin 2	NC	Pin 7	CAN-GND
Pin 3	CAN-GND	Pin 8	CAN-GND
Pin 4	CAN-GND	Pin 9	NC
Pin 5	CAN-SIGNAL LOW	Pin 10	NC

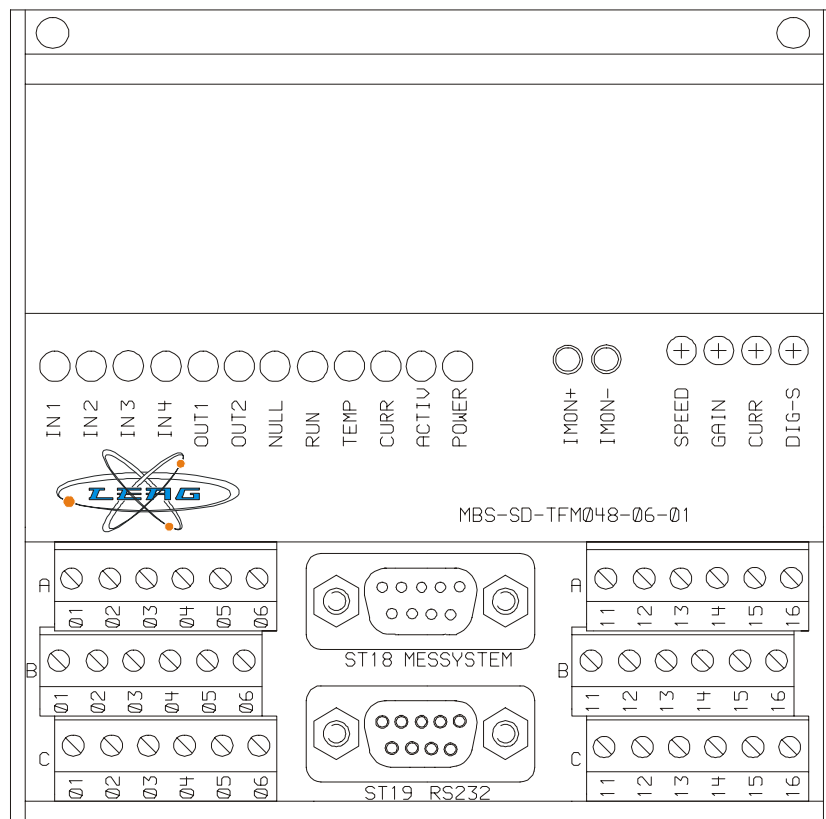
### 6.2.6. Abgriff Strom-Monitor

Steckbuchsen 2mm

IMON+ rot

IMON- schwarz

## 6.3. Einstellungen am Modul



### 6.3.1. Potentiometer Speed      Geschwindigkeitsbegrenzung

Rechtsdrehung (CW) = erhöhte Verstärkung (niedrigere Geschwindigkeit). Die Justierung ist notwendig, um die Istgeschwindigkeit auf den Sollwerteingang einzustellen.

Diese Einstellung ist nur notwendig, wenn das System mit einem analogen Tacho ausgerüstet ist. Wird kein Tacho verwendet, so muss das Potentiometer im Gegenuhrzeigersinn an den Anschlag gedreht werden.

### 6.3.2. Potentiometer Gain      Regelverstärkung

Es dient zum Abgleich des Frequenzgangs des Geschwindigkeitsreglers.

Das Drehen im Uhrzeigersinn (CW) ergibt eine Erhöhung der Hochfrequenzverstärkung. Die Einstellung muss mit *grosser Sorgfalt* durchgeführt werden, da eine höhere AC - Verstärkung auch zu höherer Stromwelligkeit, und somit zu einer erhöhten Motortemperatur führt. Optimierung der „Sprungantwort“ mit diesem Potentiometer, soll unter Beobachtung des Strommonitorsignals auf einem Oszilloskop durchgeführt werden.



### 6.3.3. Potentiometer Current Strombegrenzung

Mit diesem Potentiometer kann der Spitzenstrom eingestellt werden. Wenn man im Uhrzeigersinn bis zum Anschlag dreht, erhält man den maximalen Strom.

Änderungen können durchgeführt werden, indem man den Ausgang des Strommonitors beobachtet.

### 6.3.4. Potentiometer DIG-S Speed mit Inkrementalgeber

Diese Geschwindigkeitseinstellung hat in Verbindung mit dem Tachosignal keinen Einfluss, solange Jumper-2 nicht gesetzt, und kein Inkrementalgeber angeschlossen ist. -> Im Uhrzeigersinn steigt die Drehzahl.

---

## 6.4. Diagnose LED

### 6.4.1. IN 1 gelb

Freigabe - Eingang (Pin11 B): Muss gesetzt sein, damit der Antrieb automatisch läuft.

### 6.4.2. IN 2 gelb

Referenz - Schalter: Bei „Referenzfahren“ kann der externe Referenz - Schalter mit einbezogen werden.

### 6.4.3. Endschalter „Minus“ gelb

Sobald der Antrieb den Endschalter „Minus“ auslöst und dabei den Eingang deaktiviert, wird der Antrieb in dieser Bewegungsrichtung blockiert.

### 6.4.4. Endschalter „Plus“ gelb

Sobald der Antrieb den Endschalter „Plus“ auslöst und dabei den Eingang deaktiviert, wird der Antrieb in dieser Bewegungsrichtung blockiert.

### 6.4.5. OUT 1 gelb

System in Betrieb

### 6.4.6. OUT 2 grün

Frei wählbar. z.B. Bremse...

### 6.4.7. NULL grün

Referenzmarke des Inkrementalgebers (C-Signal)

### 6.4.8. RUN grün

Blinkt, wenn Steuerung in Betrieb



#### **6.4.9. Übertemperatur (Temp) rot**

Zu kleine Kühlung. Bei grossem Leistungsbedarf eventuell mit Fremdbelüftung nachhelfen.

Bei langen Zuleitungen zum Motor, und anschliessendem Kurzschluss, kann das MBS-SD diesen Fehler nicht mehr als „Overcurrent“ erkennen (Zuleitungsinduktivität). Dadurch wird der Regler auch überhitzt.

Dieser Fehler bleibt solange gespeichert bis er mittels dem Enable-Signal zurückgestellt wird. Er wird auch durch den „Power-on Reset“ zurückgesetzt.

#### **6.4.10. Rücksetzung durch ENABLE. Kurzschluss (Curr) rot**

Diese Funktion wird durch ein Kurzschluss am Ausgang, Kurzschluss gegen Masse oder Wicklungsschluss im Motor ausgelöst. Schaltet der Regler aus, so bleibt er gespeichert bis das Enable-Signal zurückgestellt wird.

#### **6.4.11. ACTIV grün**

Servoverstärkerteil ist in Funktion.

#### **6.4.12. POWER grün**

Interne Speisung o.k.

---

### **6.5. Strommonitor**

Zeigt im Verhältnis zum Ausgangsstrom ein Analogsignal ( $5A = ca. 3.5 V$ )

Das exakte Verhältnis wird auf der Diskette angegeben. Zu jedem Bauteil wird eine Diskette mitgeliefert.

Schaffhausen, 6.09.96 / LE

LEAG Antriebstechnik AG, Schaffhausen