



# CyberServo CO9110 Einachssteuerung

## Handbuch

15.09.2008

---

### Kommunikation

Schnittstelle: .....	3
Kommandos: .....	3
Quittierung: .....	4
Parameter: .....	4
Einschalten: .....	4
Bootloader: .....	5

Befehlsübersicht.....	2
-----------------------	---

## Befehlsübersicht

AC (Acceleration).....	6
AD (Address) .....	6
AM (After Move) .....	8
BG (Begin) .....	8
BJ (Begin Joined).....	8
BN (Burn) .....	9
BP (Beep) .....	10
BR (Brake) .....	10
CE (Clear Error).....	11
DB (Dead Band) .....	11
DP (Define Position) .....	12
DT (Define Target Position).....	12
EJ (Error Limit Joined) .....	13
ER (Error Limit).....	13
GC (Get Control) .....	14
IL (Integral Limit) .....	14
JR (Joined Reference Mode) .....	15
KD (Differential Factor) .....	16
KI (Integral Factor).....	17
KP (Proportional Factor).....	17
LM (Limit Mode) .....	18
MD (Mode).....	19
MO (Motor Off) .....	20
MT (Motor Type) .....	20
OF (Offset).....	21
PA (Position Absolute).....	22
PB (PWM Output Begin).....	22
PID-Regler.....	22
PO (PWM Output).....	23
PR (Position Relative).....	23
RB (Reibung) .....	24
RC (Remote Control) .....	25
RE (Reference Error).....	25
RJ (Reference Joined) .....	26
RF (Reference).....	27
RM (Remote Mode).....	28
RO (Reference Offset) .....	28
RV (Reference Velocity) .....	29
SF (Set Filter) .....	29
SP (Speed) .....	30
SR (Stop Ramp) .....	30
ST (Stop) .....	31
TB (Tell Burned) .....	31
TE (Tell Error).....	32
TO (Timeout) .....	32
TP (Tell Position) .....	32
TS (Tell Status) .....	33
VE (Version) .....	34
WD (Window).....	34



**Quittierung:**

Alle Kommandos, die mit der richtigen Moduladresse gesendet werden, werden von der Steuerung wie folgt quittiert:

Befehl erfolgreich:	Adr. + „>“ + ¶
Ausnahme: AM,TB,TE,TP,TS,TB,VE	Adr. + Parameter (LSB...HSB) + „>“ + ¶
Befehl nicht bekannt:	Adr. + „?“ + ¶ (wenn Mode bit6 = 1)
Befehl bekannt, aber falsche Anzahl der Parameter:	Adr. + „?“ + ¶ (wenn Mode bit6 = 1)

Die Adresse (Adr.) wird nur gesendet, wenn der Mode (MD Highbyte bit6, Vgl. MD) entsprechend gesetzt ist.

Alle Kommandos mit einer gemeinsamen Moduladresse werden nicht quittiert. Ebenso erfolgt bei Verwendung einer gemeinsamen Adresse keine Antwort auf Abfragebefehle wie TP,TS

**Parameter:**

Alle mit BN im EEPROM gespeicherten Parameter stehen nach dem Wiedereinschalten automatisch zur Verfügung. Auch nach Durchführung einer Referenzfahrt werden diese erneut geladen.

Einzelne im SRAM gespeicherte Parameter können mit der Kombination Adresse + Kommando + "?" + cr abgefragt werden, wobei auch hier, die Ausgabe LSB..HSB formatiert ist. Ausgenommen hiervon sind die Kommandos: AD, BR, PA, PR und DP.

Bsp: (Adr = XA)

Befehl:  
XAKP?¶

Modulantwort:  
KP=8000>¶            KP = 128

**Einschalten:**

Aktuelle Position = 0  
Motor Off Zustand  
Bremsen (falls vorhanden) = ON

**Bootloader:**

Ab der Version CyberServo CO9110 V01.10 ist in die Firmware des Servomoduls ein Bootloader integriert, mit dessen Hilfe sich Updates der Firmware über die RS485 – Schnittstelle realisieren lassen. Hierzu sind eine Reihe weiterer Befehle integriert, die nicht Bestandteil dieser Beschreibung sein sollen. Alle Updates sind abwärtskompatibel.

Für ein Firmwareupdate wenden Sie sich bitte an die aj Cybertron GmbH.

## Befehle:

### AC (Acceleration)

---

Stellt die Beschleunigung für eine Fahrt ein.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)  
Auflösung: 1qc/1000ms

Der Beschleunigungswert wird nicht in der Maßeinheit qc/s<sup>2</sup> angegeben. Ein Beschleunigungswert von 50 (50qc/1000ms) bedeutet, dass die aktuelle Geschwindigkeit alle 20ms um 1 erhöht wird.

Bsp: (Adr = XA)  
Acc Soll dec = 50qc/1000ms  
hex [HSB..LSB] = 0032  
hex [LSB..HSB] = 3200

Befehl:  
XAAC3200¶ entspr. 50qc/1000ms  
Byte0 = 32 [LSB]  
Byte1 = 00 [HSB]

Modulantwort:  
XA>¶

Vgl. SP (Speed)

### AD (Address)

---

Stellt die Modul-Adresse ein, wobei der Ascii-code genutzt wird.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Bsp: (alte Adr = XA)  
Soll Adresse = "XB"  
hex Asc("X") = 58 entspr. HSB  
hex Asc("B") = 42 entspr. LSB

Befehl:  
XAAD5842¶ stellt die Adresse "XB" ein

Modulantwort:  
XB>¶

Um die neue Adresse dauerhaft zu speichern, muss sie mit BN (Burn) ins EEPROM gebrannt werden.

Vgl. BN (Burn)

**Gemeinsame Adresse:**

Das zweite Byte einer gemeinsamen Adresse ist immer "0" (Null). Besitzen Sie also 2 Module mit den Adressen "XA" und "XB", dann ist die gemeinsame Adresse "X0".

Bsp:

Befehl:

```
X0PA00000000¶ die Module XA und XB fahren zur Position 0
```

Modulantwort:

```
keine
```

Besitzen Sie 2 Module mit den Adressen "XA" und "YA", dann existiert keine gemeinsame Adresse für beide Module. Das Modul "XA" besitzt die gemeinsame Adresse "X0" und das Modul "YA" besitzt die gemeinsame Adresse "Y0".

**Auswahl der Adresse:**

Es ist immer empfehlenswert, bei der Vergabe einer Adresse auf Zeichen zurückzugreifen, die a) einmalig im System und b) nicht im Zeichensatz des Hexadezimalcodes enthalten sind. Verwenden Sie also nicht ausschließlich Ziffern von 0..9 bzw. Buchstaben von A..F.

**Erstinbetriebnahme:**

Die Moduladresse ist im EEPROM gebrannt. Ist dieses jedoch noch leer, dann besitzen alle Adressbytes des EEPROM den Wert \$FF, so auch die Moduladresse.

Ist die gewünschte Moduladresse "XA", dann stellt man diese wie folgt ein:

```
ÿÿAD4158¶
```

```
wobei ÿ = chr(255)
```

```
AD = Befehl
```

```
41 = hex(asc("A"))
```

```
58 = hex(asc("X"))
```

## AM (After Move)

---

Fragt das Modul ab, ob die aktuelle Fahrt beendet wurde.  
Die Antwort des Moduls gibt nur Auskunft darüber, ob die Fahrt beendet wurde oder nicht, also ob sich die Achse noch bewegt oder nicht. Auch wenn die Fahrt durch einen Fehler (Error Limit, Timeout) beendet wurde, dann antwortet das Modul mit 1 (entspr. Fahrt beendet).  
Alternativ steht der Befehl "Tell Status" (TS) zur Verfügung.

Bsp: (Adr = XA)

Befehl:  
XAAM¶

Modulantwort:  
XA0>¶ Fahrt ist nicht beendet  
XA1>¶ Fahrt ist beendet

(Vgl. TS)

## BG (Begin)

---

Startet die Fahrt zu der mit PA bzw. PR übergebenen Position.

Bsp: (Adr = XA)

Befehl:  
XABG¶

Modulantwort:  
XA>¶ Befehl verstanden  
XA#¶ wenn Fahrt beendet (Vlg. MD)

## BJ (Begin Joined)

---

Startet die Fahrt zu der mit PA bzw. PR übergebenen Position, wobei während der Fahrt der mit EJ eingestellte Wert als Schleppfehler überwacht wird. Bei Überschreitung dieses Wertes geht das Modul in den Motor-Off Zustand und meldet dies dem zweiten angekoppelten Modul, damit auch dieses abschaltet.

Bsp: (Adr = XA)

Befehl:  
XABJ¶

Modulantwort:  
XA>¶ Parameter gebrannt  
XA#¶ wenn Fahrt beendet (Vlg. MD)

## BN (Burn)

---

Brennt die Parameter ins EEPROM.

Die so gebrannten Parameter werden nach dem Wiedereinschalten des Moduls in das SRAM geladen und aktiviert. Ebenso werden diese Parameter nach jeder Referenzfahrt geladen.

Bsp: (Adr = XA)

Befehl:  
XABN¶

Modulantwort:  
XA>¶ Parameter gebrannt

Der Brennvorgang kann bis zu 1 Sekunde dauern. Die Antwort erfolgt erst nach diesem Vorgang.:

Folgende Parameter werden durch den Befehl BN gebrannt:

KP  
KI  
KD  
IL  
AC  
SP  
MD  
ER  
DB  
TO  
OF  
RB  
WD  
SF  
RV  
MT  
RO  
RE  
LM  
PO  
EJ und natürlich  
AD

(Vgl. TB)

## BP (Beep)

---

Erzeugt ein akustisches Signal durch Ansteuerung des Motors

Ein Beep – Kommando wird nur im Motor-Off Zustand ausgeführt. Die Höhe des PWM-Signals stellt dabei die Amplitude dar. Die Frequenz wird durch eine Verzögerung der Richtungsumkehr erreicht, die in Schritten von 3,5 µs angegeben wird.

Parameter Länge: 4 Byte (Kommunikation: 8 Byte)

Bsp: (Adr = XA)

Befehl:

```
XABP7A6F5000¶
--          7A   Verzögerung (x 3.5 µs)
--          6F   PWM-Signal (entspr. Amplitude)
--          --   50   (LSB..HSB)
--          00   $0050 = Länge Beep (80ms)
```

Modulantwort:

```
XA>¶
```

Verzögerung Bsp:

```
$7A = 122 dez.
d.h. alle 122*3.5µs = 427µs erfolgt ein Pegelwechsel
d.h. die Periodendauer beträgt 427µs x 2 = 854µs
d.h. die so eingestellte Frequenz beträgt ca. 1.17 kHz
```

## BR (Brake)

---

Schaltet die Bremse (falls vorhanden)

Die Bremse ist Low-Aktiv, d.h. im Ruhezustand bzw. nach dem Einschalten ist die Achse gebremst (BR = 1)

Parameter Länge: 1 Byte (Kommunikation: 2 Byte)

Bsp: (Adr = XA)

Befehl:

```
XABR01¶   Schaltet die Bremse ein, d.h. die Achse ist
           gebremst
XABR00¶   Schaltet die Bremse aus, d.h. die Achse kann
           verfahren werden
```

Modulantwort:

```
XA>¶
```

## CE (Clear Error)

---

Löscht die gespeicherten Fehlerzustände

Die Daten von TE (Tell Error) sowie die Flags für Bits 1 (error limit) und Bit 2 (timeout) von TS (Tell Status) werden gelöscht.

Bsp: (Adr = XA)

Befehl:  
XACE¶ Fehler löschen

Modulantwort:  
XA>¶

## DB (Dead Band)

---

Setzt das Totband

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Das Totband beschreibt eine Toleranz um die auszuregelnde Position, innerhalb derer keine Regelung stattfindet. Erst bei Überschreitung des Totbandes setzt die Regelung ein.

Das Totband ist bei Motor Off deaktiv. Am Ende einer Fahrt bzw. durch das Kommando ST wird es automatisch aktiviert

Bsp: (Adr = XA)  
Totband Soll dec = 5qc  
hex [HSB..LSB] = 0005  
hex [LSB..HSB] = 0500

Befehl:  
XADB0500¶ Totband = ± 5qc

Modulantwort:  
XA>¶

## DP (Define Position)

---

Setzt die aktuelle Position auf den übergebenen Wert. Dabei werden Ist- und Sollposition gleichzeitig gesetzt.

Parameter Länge: 4 Byte (Kommunikation: 8 Byte)

Bsp: (Adr = XA)  
IstPosition dec = 100 qc  
hex [HSB..LSB] = 00000064  
hex [LSB..HSB] = 64000000

Befehl:  
XADP64000000¶ Position = 100qc

Modulantwort:  
XA>¶

## DT (Define Target Position)

---

Setzt die aktuelle Sollposition auf den übergebenen Wert. Dabei wird die Istposition im gleichen Verhältnis zur vorigen Sollposition gesetzt, so dass im ST-Zustand die Regelung nicht unterbrochen wird.

Dieses Kommando wird vom Controller nur im MO - und im ST - Zustand akzeptiert. Während einer Fahrt antwortet das Modul mit "XA?¶" !

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Bsp: (Adr = XA)  
SollPosition(alt) dec = 1000 qc  
IstPosition(alt) dec = 990 qc  
SollPosition(neu) dec = 0 qc  
IstPosition(neu) dec = -10 qc

hex [HSB..LSB] = 00000000  
hex [LSB..HSB] = 00000000

Befehl:  
XADT00000000¶ Position = 0qc

Modulantwort:  
XA>¶ ok  
XA?¶ nicht akzeptiert

Dieses Kommando verwendet man hauptsächlich in Dreheinheiten.

## EJ (Error Limit Joined)

---

Setzt die maximal zulässige Regelabweichung einer mit BJ initiierten Fahrt. Beim Auftreten dieses Fehlers schaltet das Modul in den Motor-Off Zustand und meldet dies per Handshake dem gekoppelten zweiten Modul, damit auch dieses abschaltet.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Die Regelabweichung ist die Differenz des Regler-Sollwertes zum Istwert. EJ wird nur während der mit BJ gestarteten Fahrt überprüft, nicht im Stop.

Bsp: (Adr = XA)  
Max. Regelkabw. = 50 qc  
hex [HSB..LSB] = 0032  
hex [LSB..HSB] = 3200

Befehl:  
XAEJ3200¶ max. gekoppelte Regelabweichung = 50

Modulantwort:  
XA>¶

## ER (Error Limit)

---

Setzt die maximal zulässige Regelabweichung.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Die Regelabweichung ist die Differenz des Regler-Sollwertes zum Istwert. Sie wird im ST-Zustand sowie während einer Fahrt überprüft.

Bsp: (Adr = XA)  
Max. Regelkabw. = 500 qc  
hex [HSB..LSB] = 01F4  
hex [LSB..HSB] = F401

Befehl:  
XAERF401¶ max. Regelabweichung = 500

Modulantwort:  
XA>¶

## GC (Get Control)

---

Liest Reglerdaten aus.

GC liest aktuelle Reglerdaten aus, die zur Auswertung, Parametrierung ect. herangezogen werden können. Es werden die Regelabweichung, das PWM und die Regelrichtung ausgegeben

Bsp: (Adr = XA)

Befehl:  
XAGC¶

Modulantwort:  
2C013201>¶  
-- PWM Dir [00..01] = positiv  
-- PWM [00..FF] = 50  
---- Regelabweichung [LSB..HSB] = 300

Die Antwort erfolgt ohne die Rückgabe der Adresse

## IL (Integral Limit)

---

Setzt den Maximalwert des Integralanteils des PID-Reglers.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Der Maximalwert begrenzt den I-Anteil des PID-Reglers, so dass dieser den hier eingestellten Wert über die Zeit nicht überschreitet.

Bsp: (Adr = XA)  
Max. Int.Anteil = 255  
hex [HSB..LSB] = 00FF  
hex [LSB..HSB] = FF00

Befehl:  
XAILFF00¶ max. Integralanteil = 255

Modulantwort:  
XA>¶

Beachten Sie, dass das Stellsignal des PID-Reglers eine Auflösung von 1/128 besitzt.

## JR (Joined Reference Mode)

Setzt die Referenzrichtung der Achse für die *gemeinsame* Referenzfahrt sowie den Referenzmodus (Vgl. Kommando RJ).

Parameter Länge: 1 Byte (Kommunikation: 2 Byte)

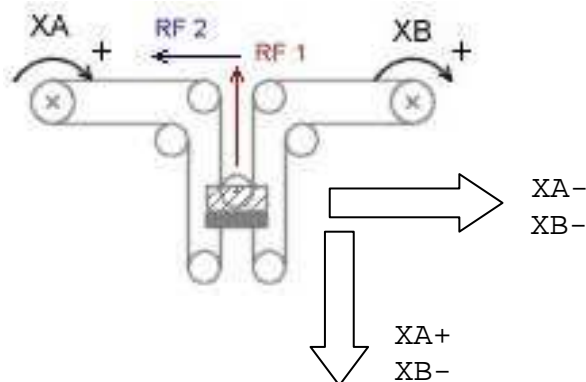
bit0 = 0 -> Ref- (direction negative)  
 bit0 = 1 -> Ref+ (direction positive)  
 bit1 = 0 -> Master  
 bit1 = 1 -> Slave  
 bit2 = 0 -> Automaster ON  
 bit2 = 1 -> Automaster OFF

Die Referenzrichtung ist abhängig vom Anschluß der Motoren und des Parameters von MT (vgl. MT). Der Zusammenschluß zweier Module zu einer Fahrteinheit (Portal o.ä.) setzt dabei immer eine Richtungsvorgabe voraus.

Eine gemeinsame Referenzfahrt erfolgt grundsätzlich gegen einen Limit 1 -Schalter eines der beiden Module

Bsp.

Aus der Darstellung (vgl. RJ) schlußfolgern sich die Fahrtrichtungen:



Die Referenzfahrten sollen nun wie dargestellt zunächst in der Z-Ebene (RF1) und anschließend in X-Ebene (RF2) erfolgen. Im Automaster-Modus (bit2 = 0) ergibt sich folgender Befehlssatz:

Bsp: (Adr1 = XA, Adr2 = XB)

RF1:	Antwort	Beschreibung
XAJR00¶	XA>¶	Ref -
XBJR01¶	XB>¶	Ref +
X0RJ¶	(keine)	Start gemeinsame Referenz
RF2:	Antwort	Beschreibung
XAJR01¶	XA>¶	Ref +
XBJR01¶	XB>¶	Ref +
X0RJ¶	(keine)	Start gemeinsame Referenz

Im Automaster-Modus wird automatisch das Modul zum Master, dessen Limit1 - Schalter zuerst den Anschlag detektiert. Ab dann steuert dieses Modul die weitere Referenzfahrt.

Jedoch ist der Automaster-Modus nicht für alle Fälle geeignet. Soll z.B. in der Z-Ebene referiert werden, während der Limitschalter der X-Ebene schon bedämft ist, führt die so angestoßene Referenzfahrt zu einem Fehler.

Hier setzen das bit1 und bit2 des JR-Parameters an. Voraussetzung ist hier die Kenntnis darüber, welcher Limitschalter an welchem Modul angeschlossen ist:

Bsp: (Adr1 = XA, Adr2 = XB)

RF1 -> XA Limit1  
RF2 -> XB Limit1

RF1:	Antwort	Beschreibung
XAJR04¶	XA>¶	Ref -; Master; Auto OFF
XBJR07¶	XB>¶	Ref +; Slave ; Auto OFF
X0RJ¶	(keine)	Start gemeinsame Referenz

RF2:	Antwort	Beschreibung
XAJR07¶	XA>¶	Ref +; Slave ; Auto OFF
XBJR05¶	XB>¶	Ref +; Master; Auto OFF
X0RJ¶	(keine)	Start gemeinsame Referenz

## KD (Differential Factor)

---

Setzt den Differentialanteil des PID-Reglers.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Die Auflösung des Differentialanteils des PID-Reglers beträgt 1/128.

Bsp: (Adr = XA)

Diff. Anteil = 32  
hex [HSB..LSB] = 0020  
hex [LSB..HSB] = 2000

Befehl:

XAKD2000¶ Differentialanteil = 32

Modulantwort:

XA>¶

## KI (Integral Factor)

---

Setzt den Integralanteil des PID-Reglers

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Die Auflösung des Integralanteils des PID-Reglers beträgt 1/128.

Bsp: (Adr = XA)  
Diff. Anteil = 8  
hex [HSB..LSB] = 0008  
hex [LSB..HSB] = 0800

Befehl:  
XAKI0800¶ Integralanteil = 8

Modulantwort:  
XA>¶

## KP (Proportional Factor)

---

Setzt den Proportionalanteil des PID-Reglers

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Die Auflösung Proportionalanteils des PID-Reglers beträgt 1/128.

Bsp: (Adr = XA)  
Prop. Anteil = 256  
hex [HSB..LSB] = 0100  
hex [LSB..HSB] = 0001

Befehl:  
XAKP0001¶ Proportionalanteil = 256

Modulantwort:  
XA>¶

## LM (Limit Mode)

---

Setzt die Polarität der Limitschalter und aktiviert / deaktiviert sie.

Parameter Länge: 1 Byte (Kommunikation: 2 Byte)

Die Verdrahtung der Limitschalter kann so realisiert werden, dass diese im aktiven Zustand auf Masse oder auf High TTL-Pegel schalten. In der Regel wird für den Low-aktiven Zustand ein interner Pull-Up Widerstand von ca. 100kΩ auf den Signaleingang geschaltet, was durch bit0 bzw. bit1 des Limit Mode realisiert wird. Im High-aktiven Zustand sollten die externen Pull-Down Widerstände vorgesehen sein.

Ebenfalls verwendet werden können Open Collector Lichtschranken, die einen Pull-Up fordern, trotzdem aber High-aktiv schalten.

Die Limitschalter sind bei normaler Anwendung zu aktivieren, weil sie nur dann abgefragt werden. Im Sonderfall, bspw. bei Einsatz einer Rotationsachse ist der Limitschalter nur während der Referenzfahrt bedeutend. Um bspw. pro Umdrehung das Auslösen eines Fehler zu unterbinden, lassen sich die Limitschalter einzeln deaktivieren.

Bei einer Referenzfahrt sind die Limitschalter automatisch aktiviert.

Konfigurieren der Limitschalter:

```

bit0 = 0 -> Limit 1 = clear pullup
bit0 = 1 -> Limit 1 = set pullup
bit1 = 0 -> Limit 2 = clear pullup
bit1 = 1 -> Limit 2 = set pullup
bit2 = 0 -> Limit 1 = disabled
bit2 = 1 -> Limit 1 = enabled
bit3 = 0 -> Limit 2 = disabled
bit3 = 1 -> Limit 2 = enabled
bit4 = 0 -> Limit 1 = active low
bit4 = 1 -> Limit 1 = active high
bit5 = 0 -> Limit 2 = active low
bit5 = 1 -> Limit 2 = active high

```

Bsp: (Adr = XA)

```

Limit1      = aktiv bei Masse-Signal      = low active
Limit2      = aktiv bei high TTL-Signal   = high active
Limit1      = aktiviert
Limit2      = aktiviert
hex [HSB..LSB] = 2D
hex [LSB..HSB] = 2D

```

Befehl:

```

XALM2D¶      Limit1 = set pull up, low active, enabled
              Limit2 = clear pull up, high active, enabled

```

Modulantwort:

```

XA>¶

```

## MD (Mode)

---

Stellt die Verhaltensweise der Steuerung ein.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Über das Mode - Kommando werden verschiedene Verhaltensweisen der Steuerung bei auftretenden Fehlerzuständen sowie für die Meldung solcher Fehler eingestellt. Die Parameter sind dabei binärcodiert.

### LOWBYTE:

bit0=0:	sende „#“ wenn Fahrt beendet AUS
bit0=1:	sende „#“ wenn Fahrt beendet EIN
bit1=0:	default (Empfang von „#“ wird ignoriert)
bit1=1:	ST(Stop) bei Empfang von „#“
bit2=0:	on error limit motor off
bit2=1:	on error limit servo here
bit3=0:	on timeout motor off
bit3=1:	on timeout servo here
bit4=0:	on error limit sende „e“ AUS
bit4=1:	on error limit sende „e“ EIN
bit5=0:	on timeout sende „t“ AUS
bit5=1:	on timeout sende „t“ EIN
bit6=0:	on error command sende „?“ AUS
bit6=1:	on error command sende „?“ EIN
bit7=0:	on limit switch sende „l“ (left), „r“(right) AUS
bit7=1:	on limit switch sende „l“ (left), „r“(right) EIN

### HIGHBYTE:

bit0=0:	on overtemp sende „o“ AUS
bit0=1:	on overtemp sende „o“ EIN
bit1=0:	on limit1 motor off
bit1=1:	on limit1 servo here
bit2=0:	on limit2 motor off
bit2=1:	on limit2 servo here
bit3=0:	on reference ok sende „h“ (home) AUS
bit3=1:	on reference ok sende „h“ (home) EIN
bit4=0:	on error limit and motor off set brake on AUS
bit4=1:	on error limit and motor off set brake on EIN
bit5=0:	on limit switch (1/2) and motor off set brake on AUS
bit5=1:	on limit switch (1/2) and motor off set brake on EIN
bit6=0:	on answer send address first AUS
bit6=1:	on answer send address first EIN
bit7=0:	ST on begin + MO on pos ok AUS
bit7=1:	ST on begin + MO on pos ok EIN

Bsp: (Adr = XA)

Sende „#“ bei Fahrt beendet  
 Sende „e“ bei Schleppfehler  
 Sende „t“ bei Timeout

hex [HSB..LSB] = 0031  
 hex [LSB..HSB] = 3100

Befehl:

XAMD3100¶

Modulantwort:

XA>¶

## MO (Motor Off)

---

Schaltet die Regelung aus. Nach diesem Kommando kann der Motor frei bewegt werden.

Bsp: (Adr = XA)

Befehl:  
XAMO¶

Modulantwort:  
XA>¶

(Vgl. ST)

## MT (Motor Type)

---

Setzt die Zählrichtung des Inkrementalgebers und die Richtung des PWM-Signals. Der Parameter ist binärcodiert.

Parameter Länge: 1 Byte (Kommunikation: 2 Byte)

bit0=0: Incr Zähler +  
bit0=1: Incr Zähler -  
bit1=0: PWM DIR +  
bit1=1: PWM DIR -

Bsp: (Adr = XA)

Befehl:  
XAMT03¶      Incr. Zähler -  
                 PWM DIR -

Modulantwort:  
XA>¶

## OF (Offset)

---

Stellt einen Offset in Abhängigkeit zur Regelrichtung ein.

Parameter Länge:	2 Byte (Kommunikation: 4 Byte)	
Bereich theoretisch:	-32768 .. 32767	[\$8000 .. \$7FFF]
Bereich praktisch: (bei 8bit-PWM)	-255 .. 255	[\$FF01 .. \$00FF]

### OF muss <= RB sein !

Dieses Kommando wirkt zusammen mit RB. OF ist nur im geregelten Zustand aktiv. Es dient einer Gewichtskompensation.

Bei einer negativen Auslenkung des Motors folgt eine positive Ausregelung, um die Auslenkung zu kompensieren. Ist  $OF > 0$ , dann wird OF in diesem Zustand der Reibung(RB) aufaddiert. Bei einer positiven Auslenkung mit folgender negativen Ausregelung wird OF dementsprechend von der Reibung subtrahiert. Damit wird eine Verschiebung des Losbrechmomentes in positive bzw. negative Richtung erzeugt. Praktisch stellt man so jeweils ein RB(+) für positive und ein RB(-) für negative Ausregelung ein.

*Annahme:  $KP=128, KI=0, KD=0, DB=0, RB=10, OF=5$*

*Pos-Abweichung = -1qc: -> (+) Ausregelung:*  
*PWM(+)* = 1 (PID-Regler) + 10(RB) + 5(OF) = 16

*Pos-Abweichung = +1qc: -> (-) Ausregelung:*  
*PWM(-)* = 1 (PID-Regler) + 10(RB) - 5(OF) = 6

Bsp: (Adr = XA)  
 Offset = -4  
 hex [HSB..LSB] = FFFC  
 hex [LSB..HSB] = FCFE

Befehl:  
 XAOFFCFE¶ Offset = -4

Modulantwort:  
 XA>¶

Vgl. RB (Reibung)

## PA (Position Absolute)

---

Setzt die absolute Sollposition der Achse. Die Fahrt wird durch den anschließenden Befehl BG gestartet.

Parameter Länge: 4 Byte (Kommunikation: 8 Byte)  
 Bereich: -2147483648.. 2147483647 [\$80000000..\$7FFFFFFF]

Bsp: (Adr = XA)  
 Soll Position Abs. = 1000  
 hex [HSB..LSB] = 000003E8  
 hex [LSB..HSB] = E8030000

Befehl:  
 XAPAE8030000¶ Position = 1000

Modulantwort:  
 XA>¶ Kommando verstanden

## PB (PWM Output Begin)

---

Stellt das durch den Befehl PO übergebene PWM-Signal am Reglerausgang bereit. Dieser Befehl begründet sich beim Einsatz mehrerer Controller, die durch eine gemeinsame Adresse angesprochen werden

Bsp: (Adr = XA)

Befehl:  
 XAPB¶

Modulantwort:  
 XA>¶

## PID-Regler

---

Der PID-Regler berechnet die Stellgröße [F(t)] (PWM-Signal) aus der Regelabweichung U(t) zum Zeitpunkt t nach dem Algorithmus:

$$F_{(t)} = \frac{1}{128} * \left( kp * U_{(t)} + kd * (U_{(t)} - U_{(t-1)}) + ki * \sum_{i=1}^t U_i \right)$$

Die Auflösung beträgt 1/128.

Die Richtungsabhängigkeit des Reglerausgangs wird durch das Richtungssignal PWM DIR gesetzt, so dass die Stellgrößen für den Proportional- und den Differentialanteil absolute Größen darstellen. Aufgrund dieses Verhaltens wird der Differentialanteil auf <=0 begrenzt. Sobald  $U_i = 0$  ist, wird der Integralanteil auf Null und  $i=t$  gesetzt. Die Begrenzung des I-Anteils über IL betrifft den Integralanteil innerhalb der Auflösung.

## PO (PWM Output)

---

Bereitet ein PWM-Signal am Reglerausgang vor, dass dann durch das Kommando PB initiiert wird.

Parameter Länge:	2 Byte (Kommunikation: 4 Byte)	
Bereich theoretisch:	-32768 .. 32767	[\$8000 .. \$7FFF]
Bereich praktisch: (bei 8bit-PWM)	-255 .. 255	[\$FF01 .. \$00FF]

Dieses Kommando löst ein MO-Zustand aus

Dieses Kommando kann dazu verwendet werden, die Drehrichtung des Motors festzustellen sowie für die Parametrierung von RB und OF.

```
Bsp: (Adr = XA)
      PWM           = 10
      PWM DIR       = -1
      -> PWM        = -10
      hex [HSB..LSB] = FFF6
      hex [LSB..HSB] = F6FF
```

```
Befehl:
      XAPOF6FF¶           PWM = -10           Ausführung erst durch PB
```

```
Modulantwort:
      XA>¶
```

## PR (Position Relative)

---

Setzt die Sollposition relativ zur Istposition der Achse. Die Fahrt wird durch den anschließenden Befehl BG gestartet.

Parameter Länge:	4 Byte (Kommunikation: 8 Byte)	
Bereich:	-2147483648.. 2147483647	[\$80000000..\$7FFFFFFF]

```
Bsp: (Adr = XA)
      Soll Position Rel. = -1000
      hex [HSB..LSB]     = FFFFFFFC18
      hex [LSB..HSB]     = 18FCFFFF
```

```
Befehl:
      XAPR18FCFFFF¶           Position = IstPosition - 1000
```

```
Modulantwort:
      XA>¶           Kommando verstanden
```

## RB (Reibung)

---

Setzt das Losbrechmoment des Motors.

Parameter Länge:	2 Byte (Kommunikation: 4 Byte)	
Bereich theoretisch:	-32768 .. 32767	[\$8000 .. \$7FFF]
Bereich praktisch:	-255 .. 255	[\$FF01 .. \$00FF]
	(bei 8bit-PWM)	

Dieser Parameter wird dem PWM aufaddiert, sobald das PWM  $\neq 0$  ist, um bei einer Auslenkung eine schnelle Reaktion des Motors auszulösen. (vgl. OF)

Einstellen der Reibung(mit Offset ab Version m128V01.02):

$RB = 0: OF = 0$             *setzen*  
 $PO = 1..X$                 *PO schrittweise erhöhen, bis die Achse anfängt, sich zu bewegen*  
 $Wert1 = PO - 1$             *merken (entspr. RB+)*  
 $PO = -1..-X$                *PO schrittweise erniedrigen, bis die Achse anfängt, sich zu bewegen*  
 $Wert2 = PO + 1$             *merken (entspr. RB-)*  
 $RB = (Abs(Wert1) + Abs(Wert2)) / 2$   
 $OF = RB - Abs(Wert2)$

*Bsp.: Wert1 = 15: Wert2 = -5*  
 $RB = (Abs(15) + Abs(-5)) / 2 = 10$   
 $OF = 10 - Abs(-5) = 5$

Bsp: (Adr = XA)  
 Reibung = 20  
 hex [HSB..LSB]            = 0014  
 hex [LSB..HSB]            = 1400

Befehl:  
 XARB1400¶                Reibung = 20

Modulantwort:  
 XA>¶

## RC (Remote Control)

---

Über den mit RC übergebenen Parameter wird im Remote Mode (vgl RM) das PWM Signal und vorzeichenabhängig die Richtung des Signals zum Verstärkereingang gesetzt. Damit läßt sich die Bewegung des Motors praktisch fernsteuern.

Als Rückgabe erfolgt die Position wie bei TP.

Durch die Verwendung eines 8-bit PWM Signals wird der Parameter in den Bereich von -256 bis 255 eingegrenzt.

Parameter = 20		Parameter = -20	
hex [HSB..LSB]	= 0014	hex [HSB..LSB]	= FFEC
hex [LSB..HSB]	= 1400	hex [LSB..HSB]	= ECFE
Befehl:      Antwort (bspw):		Befehl:      Antwort (bspw):	
XARC1400¶	XA28000000>¶	XARCECFE¶	XAD8FFFFFF>¶

## RE (Reference Error)

---

Definiert die maximal zulässige Regelabweichung während der Referenzfahrt

Parameter Länge:      2 Byte (Kommunikation: 4 Byte)

Beschreibung:  
Die Achse kann

- a)      gegen Limit 1 oder
- b)      gegen den mit RE eingestellten Schleppfehler referiert werden.

Im Fall a) sollte RE ähnlich dem eingestellten ER sein. Ist RE zu klein gewählt und während der Referenzfahrt tritt dieser Schleppfehler auf, dann erkennt die Steuerung diese Position als Referenzpunkt RP1. (Vgl. RF,RO)

Im Fall b) stellt man mit dem Kommando RE die Empfindlichkeit ein, mit der die Steuerung den Anschlag als Referenzpunkt RP1 erkennt. (Vgl. RF,RO)

Bsp: (Adr = XA)  
Reference Error = 150  
hex [HSB..LSB] = 0096  
hex [LSB..HSB] = 9600

Befehl:  
XARE9600¶              Reference Error = 150

Modulantwort:  
XA>¶

## RJ (Reference Joined)

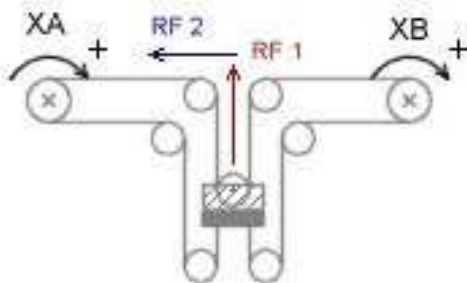
---

Die über eine gemeinsame Adresse angesprochenen Controller führen eine gemeinsame Referenzfahrt entsprechend des mit RJ übergebenen Modi aus.

Die gemeinsame Referenzfahrt verlangt das Vorhandensein von Limit1-Schaltern (vgl. RF).

Beide Controller referieren zunächst in vorgegebener Referenzrichtung (vgl. JR) solange, bis der Schalter Limit1 eines der beiden Controller gesetzt wird (AutoMaster -> vgl. JR). Dieser Controller wird nun automatisch zum Master und steuert den weiteren Verlauf der Referenzfahrt von Master und Slave.

Die Grafik soll den Aufbau eines Ausleger-Portals schematisch darstellen. Um so ein Portal in dieser Konfiguration zu referieren, sind 2 Referenzfahrten (RF1, RF2) notwendig, die wie folgt aussehen könnten.



Kommando:	Antwort:
RF1: XAJR00¶	XA>¶
XBJR01¶	XB>¶
X0RJ¶	
RF2: XAJR01¶	XA>¶
XBJR01¶	XB>¶
X0RJ¶	

Der Erfolg der Referenzfahrt (RF1, RF2) kann mit TS jedes einzelnen Controllers abgefragt werden. In bestimmten Fällen ist im AutoMaster-Modus (vgl. JR) auch eine zusätzlich Fahrt zwischen RF1 und RF2 notwendig, wenn z.B. der Limitschalter 1 von RF2 in der Referenzfahrt RF1 detektiert wird.

## RF (Reference)

---

Führt eine Referenzfahrt aus und setzt anschließend die Istposition auf 0.

Beschreibung:

Die Referenzfahrt geht standardmäßig gegen Limit1. Sie beinhaltet 2 Fahrten. Als erstes wird in der mit RV eingestellten Geschwindigkeit gegen Limit1 gefahren. Wird der Schalter betätigt, dann stoppt die Achse an diesem Punkt (RP1). Anschließend erfolgt eine langsame Fahrt aus dem Limitschalter (Limit1) heraus bis zum mit RO eingestellten Offset. Diese zweite Fahrt wird unterbrochen, sobald der Schalter (Limit1) nicht mehr betätigt ist. An dieser Stelle stoppt die Achse und setzt die Istposition auf Null. Die Referenzfahrt ist beendet.

Bei Referenz gegen Anschlag (wenn kein Limit1 zur Verfügung steht), bewegt sich die Achse mit der mit RV eingestellten Geschwindigkeit in negative Richtung bis zum Anschlag (RP1), den die Steuerung als Schleppfehler erkennt. Die Empfindlichkeit wird hierbei über das Kommando RE eingestellt. Anschließend fährt die Achse in entgegengesetzter Richtung um den mit RO eingestellten Betrag zurück und setzt die Istposition auf Null. Die Referenzfahrt ist beendet.

Die Unterscheidung, ob eine Referenz gegen Limit1 oder gegen Anschlag ausgeführt wird, ist allein abhängig von der Parametrierung. Ist RE so klein gewählt, dass ein Schleppfehler während der Fahrt zum Limit1 auftritt, dann wird dieser Punkt als RP1 (fehlerhaft) interpretiert. (Sichtkontrolle)

Nach jeder Referenzfahrt werden die im EEPROM gespeicherten Parameter erneut geladen und gesetzt. (Vgl. TB)

Der Erfolg der Referenzfahrt kann mit TS bzw. auch mit AM abgefragt werden. Nach der Referenzfahrt ist der Status (TS) bit0=1. Ist Mode (MD) Highbyte bit3=1 gesetzt worden, dann wird ein „h“ (home) nach Beendigung der Referenz gesendet.

Bsp: (Adr = XA)

Befehl:

XARF¶ Ausführen einer Referenzfahrt

Modulantwort:

XA>¶ Kommando wird ausgeführt

XAh¶ Referenzfahrt beendet (Vgl. MD)

## RM (Remote Mode)

---

Aktiviert / Deaktiviert den Remote Modus.

Beschreibung:

Im Remote Modus kann das Modul praktisch ferngesteuert werden, bspw. durch einen Joystick oder ähnliches. Die Vorgabe erfolgt dabei über das Kommando "RC"

Bsp: (Adr = XA)

Befehl:

```
XARM01¶ Aktivieren des Remote Modus
XARM00¶ Deaktivieren des Remote Modus
```

Modulantwort:

```
XA>¶ Kommando wird ausgeführt
```

## RO (Reference Offset)

---

Stellt den Wert in Quadcounts ein, um die die Achse nach Erreichen des Limitschalters bzw. bei Referenz gegen Schleppabstand des RE (ab dem Punkt RP1 (Vgl. RF)) zurückfährt.

Parameter Länge: 4 Byte (Kommunikation: 8 Byte)

Beschreibung:

*Referenzfahrt gegen Limitschalter*

Nach Erreichen des Limitschalters wird eine langsame Positionsfahrt zur Position des eingestellten RO ausgeführt. Wird dabei der Limitschalter wieder deaktiviert, dann wird diese Fahrt unterbrochen und die Position auf Null gesetzt. Hierbei sollte also der RV ausreichend hoch gesetzt werden (Bsp. 5000)

*Referenzfahrt gegen Schleppfehler (RE: Reference Error):*

Nach Erreichen eines Anschlages (Empfindlichkeit vgl.. RE) wird eine langsame Positionsfahrt zur Position des eingestellten RO ausgeführt. Anschließend wird die Position auf Null gesetzt  
(Vgl. RF,RE)

Bsp: (Adr = XA)

```
Reference Offset = 5000
hex [HSB..LSB]   = 00001388
hex [LSB..HSB]   = 88130000
```

Befehl:

```
XAR088130000¶ Reference Offset = 5000
```

Modulantwort:

```
XA>¶
```

## RV (Reference Velocity)

---

Stellt die Geschwindigkeit der Referenzfahrt ein.

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Dieses Kommando stellt die Geschwindigkeit der ersten Fahrt der Referenz ein. Die erste Fahrt ist die Fahrt in den Limit1 bzw. gegen den Anschlag in negativer Richtung (Zählrichtung).

Bsp: (Adr = XA)  
Ref. Geschwindigkeit = 2000  
hex [HSB..LSB] = 07D0  
hex [LSB..HSB] = D007

Befehl:  
XARVD007¶ Reference Velocity = 2000

Modulantwort:  
XA>¶

## SF (Set Filter)

---

Definiert die Verzögerung des Interrupts eines Inkrementaleinganges. Liegt nach dieser Zeitverzögerung immernoch die Signaländerung an, dann wird die Istposition gezählt.

Parameter Länge: 1 Byte (Kommunikation: 2 Byte)

Beschreibung:

Durch die zeitliche Verzögerung der Auswertung lassen sich Störungen, die als Peaks auf der Leitung auftreten können, herausfiltern. Die Dauer der korrekten Abarbeitung eines Interrupts beträgt bei einer Taktfrequenz von 16 MHz 3,50 µs. Durchschnittlich kann man von der Faustformel ausgehen:

$$\text{Verz} = 3,50 \mu\text{s} + 0,25 * (\text{SF}) \mu\text{s}$$

Bsp: (Adr = XA)  
Filter = 2

Befehl:  
XASF02¶ Filter = 2

Modulantwort:  
XA>¶

## SP (Speed)

---

Stellt die Geschwindigkeit für eine Fahrt ein  
Parameter Länge = 4 Byte (Kommunikation: 8 Byte)

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)  
Auflösung: 1qc/1000ms

Bsp: (Adr = XA)  
Speed Soll dec = 5000qc/1000ms  
hex [HSB..LSB] = 00001388  
hex [LSB..HSB] = 88130000

Befehl:  
XASP88130000¶ enpspr. 5000qc/1000ms

Modulantwort:  
XA>¶

Vgl. AC (Acceleration)

## SR (Stop Ramp)

---

Stoppt die Fahrt mit der mit AC eingestellten Rampe

Nur während einer Fahrt

Im Motor Off - Zustand hat der Befehl keine Auswirkung, also auch beim Fahren mit PO.

Vgl. ST (Stop)

Bsp: (Adr = XA)

Befehl:  
XASR¶

Modulantwort:  
XA>¶ Kommando verstanden  
XA#¶ wenn Fahrt beendet (Vlg. MD)

## ST (Stop)

---

Schaltet die Positionsregelung ein  
 Sollposition = Istposition  
 Während einer Fahrt wird der Motor abrupt (ohne Bremsrampe) gestoppt

Bsp: (Adr = XA)

Befehl:  
 XAST¶

Modulantwort:  
 XA>¶

Vgl. MO (Motor Off)  
 SR (Stop Ramp)

## TB (Tell Burned)

---

Listet alle mit BN gebrannten Parameter auf

Bsp: (Adr = XA)

Befehl:  
 XATB¶

Modulantwort:

KP=0002¶	KP	=	512
KI=0100¶	KI	=	1
KD=0001¶	KD	=	256
IL=0002¶	IL	=	512
AC=1000¶	AC	=	16
SP=0080¶	SP	=	32768
MD=F101¶	MD	=	497
ER=D007¶	ER	=	2000
DB=0000¶	DB	=	0
TO=8813¶	TO	=	5000
OF=0000¶	OF	=	0
RB=0600¶	RB	=	6
WD=3200¶	WD	=	50
SF=02¶	SF	=	2
RV=F401¶	RV	=	500
MT=00 ¶	MT	=	0
RO=E3080000¶	RO	=	1000
RE=3C00¶	RE	=	60
LM=00 ¶	LM	=	0
PO=0000¶	PO	=	0
>¶			

## TE (Tell Error)

---

Zeigt den zuletzt aufgetretenen Schleppfehler an.

Bsp: (Adr = XA)

Befehl:

```
XATE¶
```

Modulantwort:

```
XAE407>¶ = 2020
```

(Vgl. CE)

## TO (Timeout)

---

Setzt den Timeout zur Erreichung der Sollposition nach Beendigung des errechneten Profils

Parameter Länge: 2 Byte (Kommunikation: 4 Byte)

Bsp: (Adr = XA)

```
Timeout = 5000 ms
```

```
hex [HSB..LSB] = 1388
```

```
hex [LSB..HSB] = 8813
```

Befehl:

```
XATO8813¶
```

Modulantwort:

```
XA>¶
```

(Vgl. WD)

## TP (Tell Position)

---

Zeigt die Istposition an.

Bsp: (Adr = XA)

Befehl:

```
XATP¶
```

Modulantwort:

```
XA204E0000>¶ Istposition = 20000
```

## TS (Tell Status)

---

Zeigt den Status der Achse

Der zurückgegebene Parameter ist binärcodiert.

### LOWBYTE:

bit0 = 0	system not referenced
bit0 = 1	system referenced
bit1 = 0	no error limit
bit1 = 1	error limit
bit2 = 0	no timeout
bit2 = 1	timeout
bit3 = 0	not in position movement
bit3 = 1	in position movement
bit4 = 0	motor on or movement
bit4 = 1	motor off or pwm out (PO)
bit5 = 0	brake on (low active)
bit5 = 1	brake off
bit6 = 0	limit1 = 0 (ref-limit)
bit6 = 1	limit1 = 1 (ref-limit)
bit7 = 0	limit2 = 0
bit7 = 1	limit2 = 1

### HIGHBYTE:

bit0 = 0	no overtemp
bit0 = 1	overtemp
bit1 = 0	no joined error limit
bit1 = 1	joined error limit
bit2 = 0	no remote mode
bit2 = 1	remote mode

Die Flags für bit1 und bit2 (LOWBYTE) werden mit CE bzw RF zurückgesetzt.

Bsp: (Adr = XA)

Befehl:

```
XATS¶
```

Modulantwort:

```
XA0501>¶
```

```
0501 low: b 00000101 high: b 00000001
```

```
low bit0=1: referiert
```

```
bit2=1: timeout
```

```
bit3=0: nicht in Fahrt
```

```
high bit0=1: Übertemperatur
```

## VE (Version)

---

Zeigt die Version der Software

Bsp: (Adr = XA)

Befehl:  
XAVE¶

Modulantwort:

XAm128V01.10>¶            Version = m128V01.10

## WD (Window)

---

Setzt das Fenster, innerhalb dessen sich die Istposition befinden muss, um die Fahrt als erfolgreich abgeschlossen zu melden

Parameter Länge:        2 Byte (Kommunikation: 4 Byte)

Beschreibung:

Nach Ablauf einer PA(PR)-Fahrt (Profil) wird der Positionsfehler (PosErr = Sollpos – Istpos) mit Window verglichen. Ist PosErr ≤ Window, dann wird ein „#“ als Quittung (falls mode\_bit0=1) gesendet. Ist PosErr > Window, dann wird nach Ablauf des Timeout (Vgl. TO) ein „t“ (falls mode bit5=1) gesendet, solange bis dahin die Window-Funktion nicht erfüllt ist.

Bsp: (Adr = XA)

Window = 20  
hex [HSB..LSB]            = 0014  
hex [LSB..HSB]            = 1400

Befehl:

XAWD1400¶

Modulantwort:

XA>¶

(Vgl. TO)