Kommunikations-Referenz

Kommunikations-Referenz **11**

Was Sie für die Programmierung ohne APOSS wissen müssen ... 11
Alle Befehle der Kommunikations-Referenz ... 11

ASCII-Kommando-Referenz **19**

Voraussetzungen und Grundsätze der Befehlsstruktur ... 19
Kurzübersicht ASCII-Kommandos ... 20
Alle ASCII-Kommandos von $P bis _ ... 22
ASCII-Echomodus ... 35
Fehlermeldungen im ASCII-Kommandomodus ... 36

Kommunikations-Referenz

Was Sie für die Programmierung ohne APOSS wissen müssen

Dieser Abschnitt wendet sich an diejenigen Benutzer, die die Positioniersteuerung nicht mit der mitgelieferten PC-Software APOSS ansteuern wollen. Die Kommunikations-Referenz und die ASCII-Kommando-Referenz benötigen Sie nur für den Fall, dass Sie die Steuerung ohne APOSS programmieren wollen.

Voraussetzungen

Es werden Programmierkenntnisse, sowie Kenntnisse über die serielle Datenübertragung vorausgesetzt.

!!!

Die prinzipielle Funktionsweise und die Befehlsstrukturen der Positioniersteuerung sollten durch den Umgang mit der PC-Software APOSS bekannt sein.

Der nachfolgend beschriebene ASCII-Kommandomodus sollte, so weit möglich dem hier beschriebenen vollständigen Kommunikationsmodus vorgezogen werden.

Was sind Befehle und Meldungen?

- Die Kommunikation der Positioniersteuerung mit einem beliebigen Host-Rechner, (wie auch mit der PC-Software APOSS) geschieht über Befehls- und Meldungsstrings.
- Alle Befehle an die Positioniersteuerung (auβer ESC) bestehen aus einem **$** gefolgt von einem Zeichen sowie evtl. Daten.
- Das erste Zeichen eines Befehls (direkt nach dem **$**) gibt Auskunft über die Meldungsart.
- Die Positioniersteuerung antwortet auf jeden Befehl mit einer Rückmeldung, die mit einem Return und einem LF (Linefeed) abgeschlossen wird.
- Befehle und Meldungen bestehen aus ASCII-Strings.
- Zahlen werden als eine Folge von ASCII-Ziffern erwartet, die mit einem Leerzeichen, einem Return oder LF (Linefeed) abgeschlossen sein muss.

Was ist sonst noch wichtig?

- Leerzeichen sind in der folgenden Beschreibung durch einen _ (Unterstrich) dargestellt!
- Groβ- und Kleinschreibung muss beachtet werden!
- Bei den Rückmeldungen ist in Klammern jeweils die durch die PC-Software APOSS erzeugte Klartextmeldung angegeben.
- Serielle Datenübertragung: 9600 Baud, 8 Bit, keine Parität, 1 Stop-Bit

Alle Befehle der Kommunikations-Referenz

ESC

Behebung von Fehlerzuständen und/oder Abbrechen einer Programmausführung. Alle Fehlermeldungen werden gelöscht und die Lageregelung der Antriebe wieder aktiviert.

Mit einer Autokennung versehene Programme werden nach einem Programmabbruch sofort wieder gestartet.

Rückmeldung

B0_#47 (APOSS: Stop)

Bestätigung des Programmabbruchs und Lageregelung aller Antriebe auf der aktuellen Position.

$A_%bu_ Autokennung für das Programm *%bu* setzen. Das betreffende Programm wird künftig automatisch ausgeführt, wenn die Steuerung eingeschaltet oder wenn ein Programmabbruch durchgeführt wurde.

Durch die Vorgabe einer nicht existierenden Programmnummer kann eine bereits definierte Autokennung gelöscht werden.
%bu = Programmnummer

Rückmeldungen

A0_#37_*%bu* (APOSS: Programm Nr. %bu ist mit Autokennung versehen)

Das Programm wurde erfolgreich mit einer Autokennung versehen.

A0_#61_*%bu* (APOSS: Autostart-Programmkennung wurde gelöscht)

Durch Eingabe einer nicht vorhandenen Programmnummer wurde eine vorher gesetzte Autokennung gelöscht.

A1_#38_*%bu* (APOSS: Autokennung für Programm Nr. *%bu* nicht möglich)

Es wurde versucht, eine Autokennung für ein nicht vorhandenes Programm zu setzen. Dabei war auch kein anderes, bereits mit einer Autokennung versehenes Programm vorhanden.

$C Der gesamte Speicher inkl. der Benutzerparameter wird gelöscht und der Originalzustand der Steuerung inkl. der Init-Parameter wird wieder hergestellt.

Rückmeldung

C0_#45 (APOSS: Speicher gelöscht!)

Der gesamte Speicher wurde gelöscht und die Steuerung mit den Init-Einstellungen neu initialisiert.

$D_%u_%bb… Interner Befehl**:** Daten laden (Nicht für den Benutzer vorgesehen!)
Download von Daten in den temporären Speicher der Positioniersteuerung.
%u = Anzahl der Daten, die folgen
%bb… = Datenbytes (insgesamt *%u*)

Rückmeldungen

D0_#39_*%u* (APOSS: *%u* Datenbytes erhalten)

Diese Meldung erscheint nach einem erfolgreich ausgeführten D-Befehl.
%u = Anzahl der erhaltenen Datenbytes

D1_#41_*%u* (APOSS: Keine *%u* Datenbytes mehr frei)

Es war nicht mehr genügend temporärer Speicher vorhanden um die Datenbytes zu speichern.

%u = Anzahl der Datenbytes

D2_#40 (APOSS: Kommunikationsfehler)

Während der Datenübertragung ist ein Fehler aufgetreten.

$E Löschen aller gespeicherten Programme und Freigabe des entsprechenden Speicherplatzes.

Rückmeldungen

E0_#36 (APOSS: Alle Programme wurden gelöscht)
Alle Programme im Speicher der Positioniersteuerung wurden gelöscht und der Speicher wurde wieder freigegeben.

$F_%bu1_%bu2_ Auflistung aller Programmnamen zwischen der Nummer *%bu1* und der Nummer *%bu2*.

%bu1 = Programmnummer (0…254)
%bu2 = Programmnummer (>*%bu1*, 1…254)

Rückmeldungen

M#42_*%bu_%s* (APOSS: Prog. *%bu* ist *%s* ***)

Die angezeigte Programmnummer beinhaltet das angegebene Programm, das mit einer Autokennung versehen ist.

%bu = Programmnummer
%s = Programmname

M#43_*%bu_%s* (APOSS: Prog. *%bu* ist *%s*)

Die angezeigte Programmnummer beinhaltet das angegebene Programm.

%bu = Programmnummer
%s = Programmname

F0_ Die Programmliste ist vollständig, der letzte Eintrag innerhalb des angefragten Programmnummern-Bereichs wurde angezeigt.

F1_#44 (APOSS: Keine Programme mehr vorhanden)

Innerhalb des angefragten Programmnummern-Bereichs befinden sich keine Programme.

$G Ausführen des Programms mit der internen Tmp-Rg 255 und aktuellen Parameter Initialisierung (Direktmodus)

$I Initialisierung der Parameter
Bei einer zu geringen Batteriespannung und einem dadurch resultierenden Verlust der Daten des Speichers ist es nach dem Einschalten der Steuerung notwendig, die Steuerung mit den Init-Parametern neu zu initialisieren. (Ein Datenverlust wird durch einen CRC-Fehler angezeigt.)

Rückmeldungen

I0_#30 (APOSS: Alle Parameter sind neu initialisiert!!!)

Alle Parameter wurden mit den Init-Werten neu initialisiert.

$L Löschen des EEPROMs. EEPROM wird mit Löschvermerk versehen (1 Byte)

Rückmeldungen

L0_#69 (APOSS: EEPROM gelöscht)

EEPROM gelöscht

$M Speichern des RAM-Inhaltes in EEPROM. Alle Dateien, Programme und Parameter werden in das EEPROM gesichert

Rückmeldungen

E#62	(APOSS: Fehler beim Verifizieren)
	Fehler beim Verifizieren
M0_#67	(APOSS: in EEPROM gesichert)
	Erfolgreich in EEPROM gesichert
M1_#68	(APOSS: kein EEPROM vorhanden/Fehler beim Verifizieren)
	Tritt allein auf, falls kein EEPROM vorhanden. Bei einem Fehler beim Verifizieren wird vorher E#62 gemeldet.

$N_n_b Der Befehl vergibt die Slave-Nummer "slavenr" der Steuerung und setzt die Baudrate auf "baudratenr" für die CAN-Kommunikation.

Beim Ausführen des $N-Befehls werden im CAN-Controller der Steuerung zwei Standard-objekte angelegt:

TX-Objekt (Sendeobjekt) mit CAN-Identifikationsnummer ID = 'slavenr' * 2
RX-Objekt (Empfangsobjekt) mit CAN-Identifikationsnummer ID = 'slavenr' * 2 + 1
ID = Identifikationsnummer im CAN-Controller

Syntax $N slavenr baudratenr

Parameter slavenr = Slave-Nummer [1…1000]; dabei werden im CAN Controller zwei Objekte definiert. Wenn slavenr = 9999 werden keine Standard-objekte angelegt.

baudratenr = codierte CAN-Übertragungsrate

Die CAN-Baudrate wird wie folgt codiert:

b = 0	entspricht	5 kBaud
b = 1	entspricht	10 kBaud
b = 2	entspricht	20 kBaud
b = 3	entspricht	50 kBaud
b = 4	entspricht	100 kBaud
b = 5	entspricht	125 kBaud
b = 6	entspricht	250 kBaud
b = 7	entspricht	500 kBaud
b = 8	entspricht	1000 kBaud

Rückmeldung

N0 #82	wenn alles O.K.
N255 #82	wenn CAN-Controller nicht gefunden wurde

! ! ! Dieser Befehl wurde nicht in APOSS eingebaut. Um Slave-Nummern zu setzen muss man ein Terminalprogramm benutzen.

Syntax-Beispiel

$N 5 5	Teilnehmer 5, 125 kB
	TX-Objekt ID 10
	RX-Objekt ID 11

$O_%bu Setzt die Default-Schnittstelle

$P Wechselt in den ASCII-Kommandomodus.
Dieser Modus erlaubt die Eingabe aller Bewegungs- und I/O-Befehle in einer speziellen, leicht verständlichen ASCII-Codierung. Diese Befehlsschnittstelle ist hauptsächlich für die einfache Kommunikation mit einem Host-Rechner vorgesehen.

In diesem Modus wird keine umfangreiche Syntaxprüfung durchgeführt. Der Anwender ist somit selbst für die Korrektheit der Befehle verantwortlich.
Die Befehle des ASCII-Kommandomodus sind im Abschnitt ASCII-Kommando-Referenz ausführlich beschrieben.

Mit Q_ kann dieser Modus wieder verlassen werden.

Hat man den ASCII-Kommandomodus wieder verlassen, ist die zuletzt eingegebene Befehlsfolge (Eingabe zwischen dem letzten G-Befehl und Q) im temporären Speicher.

Rückmeldungen

P0_ASCII _Commandmode	Der ASCII-Kommandomodus wurde erfolgreich gestartet. Mit Q oder ESC kann dieser Modus wieder verlassen werden.
P0_End_ASCII _Commandmode	Der ASCII-Kommandomodus wurde verlassen.

$Q_%s Setzt Projektname auf String (8-fach)

$R_%bu_%bs_ Sichern der temporären Programmdaten mit dem Namen *%bs* unter der Nummer *%bu*.

Ein eventuell bereits unter dieser Nummer gesichertes Programm wird überschrieben.

Falls %bu größer ist als die bisher höchste Nummer eines gespeicherten Programms, werden für die dazwischen liegenden Nummern Leerprogramme erzeugt.

%bu = Programmnummer (max. 254)
%bs = Programmname (max. 8 ASCII-Zeichen)

Rückmeldungen

S0_#34_*%u*	(APOSS: Programm als Nr. *%u* gespeichert) Das Programm wurde erfolgreich gespeichert und kann über die Nummer *%u* angesprochen werden. *%u* = Programmnummer
S1_#35_*%u*	(APOSS: Speichern von Programm Nr. *%u* nicht mehr möglich) Es ist entweder kein gültiges Programm im temporären Speicher vorhanden, oder aber die Programmnummer ist unzulässig.

$S_%bs_ Sichern der temporären Programmdaten, mit dem Namen *%bs*.

%bs = Programmname (max. 8 ASCII-Zeichen)

Rückmeldungen

S0_#34_*%u* (APOSS: Programm als Nr. *%u* gespeichert)

Das Programm wurde erfolgreich gespeichert und kann über die Nummer *%u* angesprochen werden.

%u = zugeteilte Programmnummer

S1_#35_*%u* (APOSS: Speichern von Programm Nr. *%u* nicht mehr möglich)

Es ist entweder kein gültiges Programm im temporären Speicher vorhanden, oder aber alle Programmnummern (max. 254) sind belegt.

$T Schrittweise Programmausführung.

Diese Anweisung kann benutzt werden, wenn das Programm Breakpoints enthält (DEBUG-Befehle). In diesem Fall wird das Programm bis zum nächsten Breakpoint ausgeführt und dann die erreichte Zeilennummer zurückgemeldet.

Rückmeldungen

T0_#31_*%u* (APOSS: Zeile *%u*)

Meldet, dass eine Zeile erfolgreich abgearbeitet wurde.

%u = nächste Zeile, die abgearbeitet wird

T255_ Meldet, dass eine Zeile nicht korrekt ausgeführt wurde.

$U%u1 Liest das Array mit der Nr. %u1
Falls ein Array mit dieser Nummer existiert, werden die Array-Elemente zurückgeliefert.

Rückmeldungen

U0_#72
_%u1_%u2 (APOSS: Feld Nr. %u1 mit Länge %u3)

Die Meldung (72) erfolgt, wenn alles in Ordnung ist.
%u1 enthält die Nummer des gewünschten Arrays (bei 0 beginnend), %u2 gibt die Anzahl Elemente an, die dann anschließend gesendet werden. Die Elemente sind jeweils mit CRLF voneinander getrennt.

U1_#73
_%u1_%u2 (APOSS: Feld Nr. % nicht vorhanden, letztes Feld ist Nr. %)

Die Meldung (73) erfolgt, wenn das gewünschte Array mit der Nummer %u1 nicht existiert. Gleichzeitig erhält man mit %u2 hier die Anzahl der existierenden Arrays

U2_#74
_%u1_%u2 (APOSS: Feld Nr. % ist leer)

Die Meldung (74) erhält man, wenn das gewünschte Array mit der Nummer %u1 leer ist. Dies ist nur der Fall, wenn es das erste nicht existierende Array ist.
%u2 enthält auch hier wieder die Anzahl existierender Felder.

$V %u1_%u2 Beschreibt ein u1-Array mit dem Wert u2

Rückmeldungen

U0_#72
_%u1_%u2
(APOSS: Feld Nr. %u1 mit Länge %u2)

Die Meldung (72) erfolgt, wenn alles in Ordnung ist, d. h. ein Array mit der Nummer %u1 und Größe %u2 existiert oder angelegt werden kann. Anschließend muss der Anwender seine Elemente senden (%u2 Stück).

Als Bestätigung wird dann noch einmal die Meldung U0_#72_%u1_%u2 gesendet. Damit ist der Vorgang abgeschlossen.

U1_#73
_%u1_%u2
(APOSS: Feld Nr. % nicht vorhanden, letztes Feld ist Nr. %)

Die Meldung (73) erfolgt, wenn ein solches Array nicht existiert und es auch nicht das nächste freie ist (Arrays können nur nacheinander neu angelegt werden, nicht in willkürlicher Reihenfolge).

In %u2 wird mitgeteilt, wie viele Arrays existieren.

U3_#75
_%u1_%u2
(APOSS: Nicht genügend Speicher, um Feld %u1 mit Größe %u2 anzulegen)

Die Meldung (75) teilt mit, dass ein solches Array nicht existiert aber angelegt werden könnte, der Speicher dazu aber nicht ausreicht!

U4_#76
_%u1_%u2_%u3
(APOSS: Feld Nr. %u1 hat nicht Größe %u2 sondern %u3)

Die Meldung (76) wird gesendet, wenn ein Array mit Nummer %u1 zwar existiert, aber eine falsche Größe hat. Gewünscht war Größe %u2, tatsächlich hat das Array aber %u2 Elemente.

$X_%bu_ Ausführen des Programms mit der Nummer *%bu*.

Das Programm wird, sofern vorhanden, sofort ausgeführt. Während der Ausführung kann durch ein ESC (ASCII: 27) jederzeit ein Programmabbruch erreicht werden.

%bu = Nummer des auszuführenden Programms
%bu = 255 => Programm im temporären Speicher ausführen

Rückmeldungen

M#32
_%bu_%s
(APOSS: Programm *%bu %s* wird ausgeführt)

Das aufgerufene Programm mit der Nummer *%bu* und dem Namen *%s* wird gerade ausgeführt.

%bu = Programmnummer
%s = Programmname

X0
(APOSS: Programm ohne Probleme ausgeführt)

Programm wurde erfolgreich zu Ende abgearbeitet.

X255
Während der Programmausführung ist ein Fehler aufgetreten.

X254_#33
_%bu
(APOSS: Programm Nr. *%bu* ist nicht vorhanden)

Es wurde versucht ein Programm auszuführen, das nicht im Positioniersteuerungs-Speicher vorhanden ist.

%bu = Programmnummer

B0_#47 (APOSS: Stop)

Durch einen Benutzer wurde ein ESC eingegeben. Dies bewirkt einen sofortigen Programmabbruch und ein Anhalten der Antriebe an der aktuellen Position. Obige Meldung bestätigt einen solchen Vorgang.

E#*%be_%p* Fehlermeldung während der Programmausführung.

Die einzelnen Fehlermeldungen können Sie im Abschnitt Meldungen und Fehlerreferenz im Detail nachlesen.

%be = Fehlernummer
%p = Fehlerparameter

$Y Letztes Programm fortführen.

$? Abfrage des aktuellen Fehlerzustandes.

%be = Aktuelle Fehlernummer

Rückmeldungen

M#53 (APOSS: Kein Fehler aufgetreten)

Es existiert kein aktueller Fehlerzustand.

$$ Liefert SIGN-ON-String (Einschaltmeldung) zurück.

Befehlsunabhängige Rückmeldungen

K1_#48 (APOSS: V24 Fehler)

Ein Überlauf des V24-Kommunikationspuffers hat stattgefunden. Es wurden zu viele Daten übertragen.

K3_#66 *_%c_%bu* (APOSS: $ muss am Anfang eines Kommandos stehen)

Ein Kommando wurde ohne ein $-Zeichen zu Beginn der Befehlssequenz eingegeben.

%c = falsches Befehlszeichen
%bu = ASCII-Code des falschen Zeichens

K2_#46 *_%c_%bu_%lu* (APOSS: Falscher Befehl *%c %bu* um *%lu*)

Es wurde ein unbekannter oder nicht zu interpretierender Befehl eingegeben.

%c = falsches Befehlszeichen
%bu = ASCII-Code des falschen Zeichens
%lu = Stand des internen Timers

ASCII-Kommando-Referenz

Voraussetzungen und Grundsätze der Befehlsstruktur

Dieser Abschnitt ist für Benutzer, die die Positioniersteuerung nicht mit der mitgelieferten PC-Software APOSS ansteuern wollen.

Die Kenntnis der APOSS-Sprachelemente sowie deren Funktionen werden vorausgesetzt.

Der ASCII-Kommandomodus ermöglicht auf einfache Art die Bedienung der Positioniersteuerung von einem beliebigen Host oder Terminal. (Serielle Datenübertragung: 9600 Baud, 8 Datenbit, 1 Stop-Bit, keine Parity)
Der ASCII-Kommandomodus wird durch ein $P gestartet. (siehe auch Abschnitt Kommunikations-Referenz)

Die speziellen Fehlermeldungen des ASCII-Kommandomodus finden Sie im Abschnitt Fehlermeldungen im ASCII-Kommandomodus.

!!! Die APOSS-Software ist – insbesondere seit Version 5 – ständig erweitert worden. Komplexe Anwendungen wie Synchronisieren und CAM-Funktionen können inzwischen mit einfachen Befehlen wie DEC, SYNCC, SYNC etc. realisiert werden. Diese neuen Funktionen lassen sich aber kaum sinnvoll über den ASCII-Mode nutzen. Daher sind sie in der folgenden Beschreibung (noch) nicht enthalten.

Grundsätzliches zum Befehlsaufbau

Alle ASCII-Kommandos haben den gleichen Befehlsaufbau:
Buchstaben-Befehlskennung + eventuell Achskennung + eventuell Parameter

- Groß- und Kleinschreibung der Befehlskennung muss beachtet werden!
- Bei Bewegungsbefehlen können die Achsen nacheinander oder gleichzeitig verfahren werden:
  Mit vorangestelltem **&**-Zeichen wird die Bewegung erst mit dem nächsten Fahrbefehl (einer anderen Achse) ohne vorangestelltem **&** zusammen ausgeführt.
- Zwischen den Befehlsteilen können Leerzeichen eingefügt werden.
- Alle eingegebenen Befehle werden generell bis zur Eingabe des **G**-Befehls zwischengespeichert, wodurch ganze Befehlssequenzen übertragen und abgearbeitet werden können.
- Eine erfolgreiche, fehlerfreie Abarbeitung einer Befehlssequenz (nach dem **G**-Befehl) wird mit einer „X0“-Meldung betätigt.
- Nach dem Verlassen des ASCII-Kommandomodus können die noch nicht durch den **G**-Befehl abgearbeiteten Befehlssequenzen als Programm gespeichert werden. (siehe Abschnitt Kommunikations-Referenz: Befehl: $S)
- Nach dem Verlassen des ASCII-Kommandomodus werden alle Parameter wieder auf ihre ursprüngliche Einstellung zurückgesetzt.

Kapitel IV Kurzübersicht ASCII-Kommandos

Kurzübersicht ASCII-Kommandos

Kurzübersicht in alphabetischer Reihenfolge der Funktion, die folgende ausführliche Referenz in alphabetischer Reihenfolge der Befehlskennung.

Funktion	Befehlsaufbau	Befehlskennung
Abgebrochenes Programm fortsetzen	y	y
Absolute Linearinterpolation	&L%bn x %lp	L
Absolute Positionierung	A %bn x %lp	A
Achsfestlegung für Bahnbewegung	[	[
Achsparameter temporär setzen	S %bn x %bm p %lv	S
Achsstatus holen	a %bn x	a
Analogausgang setzen	v %bn x %bv	v
Analogeingang lesen	d %bA	d
Anfang der Vektorfolge signalisieren	^	^
Anfang der Vektorfolge signalisieren	_	_
ASCII-Kommandomodus aktivieren	$P	$P
ASCII-Kommandomodus verlassen	Q	Q
Ausgang setzen	O %bo %bs	O
Ausgänge byteweise schreiben	B %bo B %bb	B
Bahngeschwindigkeit festlegen	\	\
Befehlsausführung starten	G	G
Benutzerparameter wieder setzen	M %bn x	M
Beschleunigung setzen	C %bn x %wa	C
Bewegungsablauf unterbrechen	s %bn x	s
Drehzahlgeschwindigkeit setzen	D %bn x %wv	D
Drehzahlmodus aktivieren	K %bn x	K
Drehzahlmodus deaktivieren	T %bn x	T
Eingang lesen	I %bi	I
Eingänge byteweise lesen	Y %bi	Y
Fehlermeldung rücksetzen	E	E
Geradenstück definieren	l %lp x	j
Indexposition suchen	i %bn x	i
Interne Systemzeit abfragen	t	t
Kommando-Position abfragen	c %bn x	c
Kreisbogen definieren	b %lp x %lα	b
Lageregelung abschalten	f %bn x	f
Lageregelung aktivieren	n %bn x	n
Maschinennullpunkt anfahren	H %bn x	H
Nullpunkt versetzen	`	`
Permanenten Achsparameter lesen	q %bn x %bm p	q
Permanenten Achsparameter setzen	w %bn x %bm p %lv	w
Positioniergeschwindigkeit setzen	V %bn x %wv	V

Kapitel IV Kurzübersicht ASCII-Kommandos

Funktion	Befehlsaufbau	Befehlskennung
Positionsabfrage	P %bn x	P
RAM sichern	m	m
Realnullpunkt anfahren	N %bn x	N
Realnullpunkt setzen	U %bn x	U
Relative Positionierung	R %bn x %lp	R
Sollposition auf Istposition setzen	x %bn	x
Systemstatus holen	Z %bn x	Z
Temporärnullpunkt löschen	r %bn x	r
Temporärnullpunkt setzen	o %bn x %lp	o
Temporären Achsparameter lesen	p %bn x %bm p	p
Vektorlänge in Benutzereinheiten festlegen	]	]
Warten auf Eingang	X %bi %bs	X
Warten auf Zielposition	F %bn x	F
Warten nach POS-Befehlen ein/aus	J %bs	J
Wartezeit	W %lt	W

Erläuterungen

%ba = Beschleunigung
%bb = Bytewert
%bi = Eingangsnummer
%bm = Parameternummer
%bn = Achsnummer
%bo = Ausgangsnummer
%bs = Zustand (t = true, f = false)
%lt = Zeit (ms)
%lv = Wert
%lp = Position
%wv = Geschwindigkeit
%bA = Analogeingang
%bv = Bytewert (–127 bis 128)
%wa = Beschleunigung
%lα = Winkel im Bogenmaß

Alle ASCII-Kommandos von $P bis _

$P - ASCII-Kommandomodus aktivieren	**Befehl**	$P
	Funktion	ASCII-Kommandomodus aktivieren
	Rückmeldung	P0 ASCII – Command mode
A - Absolute Positionierung	**Befehl**	A %bn x %lp
	Parameter	%bn = Achsnummer %lp = Absolutposition (in BE)
	Funktion	Absolute Positionierung in Benutzereinheiten
	APOSS-Verweis	POSA
	Beispiel	Achse 1 -> 500 BE, Achse 2 -> 2000 BE: (nacheinander) A1x500 A2x2000 G Achse 1 -> 500 BE, Achse 2 -> 2000 BE: (gleichzeitiger Bewegungsbeginn) &A1x500 A2x2000 G
B - Ausgänge byteweise schreiben	**Befehl**	B %bo B %bb
	Parameter	%bo = Nummer des Ausgangsbytes (0 … max. 255, abhängig von der eingesetzten Hardware) %bb = Ausgangszustand (Bytewert: 0…255)
	Funktion	Byte-weises Setzen und Rücksetzen von Ausgängen
	APOSS-Verweis	OUTB
	Beispiel	Bit 7 von Ausgangsbyte 0 setzen, Bit 1 … 6, 8 rücksetzen B0 B64 G
C - Beschleunigung setzen	**Befehl**	C %bn x %wa
	Parameter	%bn = Achsnummer %wa = Beschleunigung (1…Geschwindigkeitsteiler)
	Funktion	Setzen des Beschleunigungswertes im Drehzahl- und Positionier-modus.
	APOSS-Verweis	ACC
	Beispiel	Minimale Beschleunigung für Achse 1 setzen C1x1 G

D – Drehzahl-Geschwindigkeit setzen

Befehl	D %bn x %wv
Parameter	%bn = Achsnummer %wv = Geschwindigkeit (–Geschw.teiler…+Geschw.teiler)
Funktion	Setzen des Geschwindigkeitswertes für den Drehzahlmodus.
APOSS-Verweis	CVEL
Beispiel	Achse 1: Minimale Geschwindigkeit setzen D1x1 G

E - Fehlermeldung rücksetzen

Befehl	E
Parameter	Löschen der aktuellen Fehlermeldung.
Funktion	E#1_%bu %bu = Nummer des gelöschten Fehlers
APOSS-Verweis	ERRCLR, ERRNO
Beispiel	Aktuellen Fehler löschen E G siehe: Rückmeldung: E#1 0 (=> kein Fehler vorhanden)

F - Warten auf Zielposition

Befehl	F %bn x
Parameter	%bn = Achsnummer
Funktion	Bei aktivem NOWAIT Modus warten bis die Achse die Zielposition erreicht hat.
APOSS-Verweis	WAITAX
Beispiel	NOWAIT aktivieren, Achse 1 -> 50000 BE, Warten bis Ziel erreicht, Achse 1 -> 10000 BE Jt A1x50000 F1x A1x10000 G

G - Befehlsausführung starten

Befehl	G
Funktion	Abarbeitung der seit dem letzten G-Befehl (bzw. seit dem Einschalten) übergebenen Befehle starten.
Anmerkung	Alle Befehle werden in einem Puffer zwischengespeichert und erst durch den G-Befehl wird die Abarbeitung der gesamten Befehlssequenz gestartet. Nach der Abarbeitung aller Befehle ist der Befehlspuffer wieder gelöscht.
Rückmeldung	Tritt während der Ausführung der Befehlssequenz ein Fehler auf, so wird eine Fehlermeldung gesandt. Bei fehlerfreier Ausführung der gesamten Befehlssequenz wird ein "X0" gesandt.
Beispiel	Befehlssequenz laden und starten A1x5000 W 1000 A1x1000 W2000 A1x5000 G

H - Maschinennullpunkt anfahren	**Befehl**	H %bn x
	Parameter	%bn = Achsnummer
	Funktion	Maschinennullpunkt (Referenzschalter + Index) anfahren und als Realnullpunkt setzen.
	APOSS-Verweis	HOME
	Beispiel	Maschinennullpunkt Achse 1 anfahren H1x G
I - Eingang lesen	**Befehl**	I %bi
	Parameter	%bi = Eingangsnummer (1…8)
	Funktion	Lesen des an einem digitalen Eingang anliegenden Signalpegels.
	Rückmeldung	M#22_%bi_%bs %bi = Eingangsnummer (1…8) %bs = Eingangszustand (0, 1)
	APOSS-Verweis	IN
	Beispiel	Zustand Eingang 7 einlesen I7 G siehe Rückmeldung: M#22 7 1 (=> Eingang 7 high)
J - Warten nach POS-Befehlen ein/aus	**Befehl**	J %bs
	Parameter	%bs = f => Warten nach POS-Befehlen = t => nicht Warten nach Post-Befehlen
	Funktion	Aktiviert bzw. deaktiviert den NOWAIT Modus, wodurch nach Positionierbefehlen nicht bzw. bis zum Erreichen der Zielposition gewartet wird.
	APOSS-Verweis	NOWAIT ON/OFF
	Beispiel	Künftig nicht nach Positionierbefehlen warten. Jt G
K - Drehzahlmodus aktivieren	**Befehl**	K %bn x
	Parameter	%bn = Achsnummer
	Funktion	Aktivieren des Drehzahlmodus, in dem die Achsen drehzahlgeregelt bewegt werden.
	! ! !	Die Achse dreht nach diesem Befehl permanent mit der vorgegebenen Geschwindigkeit, bis ein neuer Geschwindigkeitswert gesetzt wird oder der Drehzahlmodus deaktiviert wird!
	APOSS-Verweis	CSTART
	Beispiel	Geschwindigkeit setzen, Drehzahlmodus aktivieren D1x-5 K1x G

L - Absolute Linearinterpolation

Befehl	&L %bn1 x %lp1 L %bn2 x %lp2
Parameter	%bn1 = Achsnummer %lp1 = Absolutposition von n1 (in BE) %bn2 = Achsnummer (<> n1) %lp2 = Absolutposition von n2 (in BE)
Funktion	Absolutes Positionieren von zwei oder mehr Achsen, so dass beide Achsen gleichzeitig die Zielposition erreichen.
APOSS-Verweis	LINA
Beispiel	Achse 1, Achse 2 synchron auf Zielposition bewegen &L1x40000 L2x-10000 G

M - Benutzerparameter wieder aktivieren

Befehl	M %bn x
Parameter	%bn = Achsnummer
Funktion	Setzt alle Parameter der angegebenen Achse auf die vor dem Aufruf des ASCII-Modus gültigen Benutzerparameter zurück.
Beispiel	Benutzerparameter der Achse 1 wieder aktivieren M1x G

N - Realnullpunkt anfahren

Befehl	N %bn x
Parameter	%bn = Achsnummer
Funktion	Anfahren des Realnullpunktes.
APOSS-Verweis	ORIGIN bzw. POSA
Beispiel	Achse 1: Realnullpunkt anfahren N1x G

O - Ausgang setzen

Befehl	O %bo %bs
Parameter	%bo = Ausgangsnummer (1…8) %bs = f => Ausgang rückgesetzt = t => Ausgang gesetzt
Funktion	Setzen bzw. Rücksetzen eines digitalen Ausganges.
APOSS-Verweis	OUT
Beispiel	Ausgang 1 setzen, Ausgang 8 rücksetzen O1t O8f G

P - Positionsabfrage

Befehl	P %bn x
Parameter	%bn = Achsnummer
Funktion	Abfragen der aktuellen Position in Quadcounts.
Rückmeldung	M#21_%bn_%lp %bn = Achsnummer %lp = aktuelle Position (in qc)
APOSS-Verweis	APOS
Beispiel	Position Achse 1 abfragen P1x G Rückmeldung: M#21 1 0 (=> Achse 1, Pos.: 0)

Q - ASCII-Kommandomodus verlassen

Befehl	Q
Funktion	Verlassen des ASCII-Kommandomodus
Anmerkung	Alle seit dem letzten G-Befehl eingegebenen Befehle befinden sich nach der Beendigung des ASCII-Kommandomodus noch im temporären Speicher und können durch den $S-Kommunikationsbefehl als Programm gespeichert werden. Während des ASCII-Kommandomodus geänderte Parameter werden wieder auf die ursprüngliche Einstellung zurückgesetzt.
Rückmeldung	P0 End ASCII – Command mode
Beispiel	ASCII-Modus verlassen und letzte Befehlssequenz speichern A1x5000 O7t W2000 H1x Q $S TEST

R - Relative Positionierung

Befehl	R %bn x %lp
Parameter	%bn = Achsnummer %lp = Relative Position (in BE)
Funktion	Relative Positionierung (in Benutzereinheiten) ausgehend von der aktuellen Position.
APOSS-Verweis	POSR
Beispiel	Achse 1 relativ + 500 BE, Achse 2 relativ + 2000 BE: (nacheinander) A1x500 A2x2000 G Achse 1 relativ + 500 BE, Achse 2 relativ + 2000 BE: (gleichzeitiger Bewegungsbeginn) &R1x500 R2x2000 G

S - Achsparameter temporär setzen

Befehl	S %bn x %bm p %lv
Parameter	%bn = Achsnummer %bm = Parameternummer %lv = Parameterwert
Funktion	Wert eines Achsparameters temporär neu setzen.
Anmerkung	Die geänderten Parameterwerte bleiben nur bis zum Verlassen des ASCII-Kommandomodus bzw. bis zum Neusetzen des Parameters gültig. Man spricht in diesem Fall von Programmparametern, da sie nur innerhalb des aktuellen Programms (bzw. in diesem Fall innerhalb des ASCII-Modus) Gültigkeit besitzen.
APOSS-Verweis	SET
Querverweis	ASCII-Kommando: p, q, w Kapitel Software-Referenz, Abschnitt Parameter-Referenz
Beispiel	ORIGIN-Geschwindigkeit temporär neu setzen S1x16p5 G

T - Drehzahlmodus deaktivieren

Befehl	T %bn x
Parameter	%bn = Achsnummer
Funktion	Beenden des Drehzahlmodus und Positioniermodus wieder aktivieren.
! ! !	Dieser Befehl sollte nur bei einer bereits stillstehenden Achse benutzt werden.
APOSS-Verweis	CSTOP
Beispiel	Achse 1 im Drehzahlmodus verfahren, Achse 1 abbremsen Drehzahlmodus beenden K1x D1x5 W2000 G D1x0 W500 G T1x G

U - Realnullpunkt setzen

Befehl	U %bn x
Parameter	%bn = Achsnummer
Funktion	Aktuelle Position als neuen Realnullpunkt definieren.
APOSS-Verweis	DEF ORIGIN
Beispiel	Achse 1 -> 5000 BE, als neuen Realnullpunkt definieren A1x5000 U1x G

V – Positionier-Geschwindigkeit setzen

Befehl	V %bn x %wv
Parameter	%bn = Achsnummer %wv = Geschwindigkeit (–Geschwindigkeitsteiler … +Geschwindigkeitsteiler)
Funktion	Setzen des Geschwindigkeitswertes beim Positionierbetrieb.
APOSS-Verweis	VEL
Beispiel	Geschwindigkeit setzen, Achse 1 -> 50000 V1x10 A1x50000 G

W - Wartezeit

Befehl	W %lt
Parameter	%lt = Wartezeit in Millisekunden
Funktion	Definierte Programmverzögerung in Millisekunden.
APOSS-Verweis	WAITT
Beispiel	Achse 1 -> 50000, 10 Sekunden warten A1x50000 W10000 G

X - Warten auf Eingang

Befehl	X %bi %bs
Parameter	%bi = Eingangsnummer (1…8) %bs = f => Warten auf Low-Pegel = t => Warten auf High-Pegel
Funktion	Warten bis am definierten Eingang gewünschter Signalpegel anliegt.
APOSS-Verweis	WAITI
Beispiel	Warten auf High an Eingang 1, dann Ausgang 7 setzen X1t O7t G

Y - Eingänge byteweise lesen

Befehl	Y %bi
Parameter	%bi = Nummer des Eingangsbytes (0… max. 255, abhängig von der eingesetzten Hardware)
Funktion	Simultanes Lesen von acht digitalen Eingängen, wobei eine byteweise Gruppierung der Eingänge stattfindet.
Rückmeldung	M#23_%bi_%bw %bi = Nummer des Eingangsbytes %bw = Bytewert der acht Eingänge
APOSS-Verweis	INB
Beispiel	Zustand der ersten 8 Eingänge lesen Y0x G Rückmeldung: M#23 0 64 (=> nur Eingang 7 gesetzt)

Z - Systemstatus holen	**Befehl**	Z %bn x
	Parameter	%bn = Achsnummer
	Funktion	Auslesen des systeminternen Statusworts.
	Rückmeldung	M#24_%lv %lv = Systemstatus (32 Bit)
	APOSS-Verweis	STAT
	Beispiel	Systemstatus Achse 1 lesen Z1x G Rückmeldung: M#24 4981280
a - Achsstatus holen	**Befehl**	a %bn x
	Parameter	%bn = Achsnummer
	Funktion	Status einer Achse einlesen.
	Rückmeldung	M#24_%bu %bu = Achsstatus (8 Bit)
	APOSS-Verweis	AXEND
	Beispiel	Status Achse 1 lesen a1x G Rückmeldung: M#24 1 (=> Achse ist im Stillstand)
b - Kreisbogen definiere	**Befehl**	b %lp x
	Parameter	%lp1 = Position 1 %lp2 = Position 2 %lα = Winkel
	Funktion	Definieren eines Kreisbogens, beginnend am aktuellen Standort.
	APOSS-Verweis	ARC
	Beispiel	b 1000 1000 10 G
c - Kommando-Position abfragen	**Befehl**	c %bn x
	Parameter	%bn = Achsnummer
	Funktion	Aktuelle Kommando-Position in Quadcounts abfragen.
	Rückmeldung	M#21_%bn_%bc %bn = Achsnummer %bc = Kommando-Position (in qc)
	APOSS-Verweis	CPOS
	Beispiel	Kommando-Position Achse 1 abfragen c1x G Rückmeldung: M#21 1 0 (=> Achse 1, Pos.: 0)

d - Analogeingang lesen	**Befehl**	d %bA
	Parameter	%bA = Analogeingang
	Funktion	Lesen des Analogeinganges. Das Ergebnis wird über die RS232-Schnittstelle zurückgemeldet
	APOSS-Verweis	INAD
	Beispiel	d2 G
f - Lageregelung abschalten	**Befehl**	f %bn x
	Parameter	%bn = Achsnummer
	Funktion	Lageregelung abschalten.
	APOSS-Verweis	MOTOR OFF
	Beispiel	Lageregelung Achse 1 abschalten f1x G
i - Indexposition suchen	**Befehl**	i %bn x
	Parameter	%bn = Achsnummer
	Funktion	Indexposition des Encoders suchen.
	APOSS-Verweis	INDEX
	Beispiel	Indexposition Achse 1 anfahren i1x G
l - Geradenstück definieren	**Befehl**	l %lp
	Parameter	%lp1 = Position 1 %lp2 = Position 2
	Funktion	Definieren eines Geradenstücks
	APOSS-Verweis	VEC
	Beispiel	l 10000 10000
m - RAM sichern	**Befehl**	m
	Funktion	Sichert RAM ins EEPROM
	APOSS-Verweis	SAVEPROM
	Beispiel	m G

n - Lageregelung aktivieren

Befehl	n %bn x
Parameter	%bn = Achsnummer
Funktion	Lageregelung aktivieren und Antrieb auf aktueller Position halten.
APOSS-Verweis	MOTOR ON
Beispiel	Lageregelung Achse 1 aktivieren n1x G

o - Temporären Nullpunkt setzen

Befehl	o %bn x %lp
Parameter	%bn = Achsnummer %lp = Absolutposition (in BE)
Funktion	Absolutposition als temporären Nullpunkt definieren.
APOSS-Verweis	SET ORIGIN
Beispiel	Absolutposition 50000 als Temporär-NP o1x50000 G

p - Temporären Achsparameter lesen

Befehl	p %bn x %bm p
Parameter	%bn = Achsnummer %bm = Parameternummer
Funktion	Wert eines temporären Achsparameters lesen.
Rückmeldung	P%bm_M#26$%bm_%lv %bm = Parameternummer %lv = Parameterwert
Anmerkung	Als temporäre Achsparameter werden die Parameter bezeichnet, die lediglich während des ASCII-Modus ihre Gültigkeit behalten und nach dem Verlassen des ASCII-Modus wieder die Werte der permanenten Benutzerparameter annehmen.
APOSS-Verweis	GET
Querverweis	Parameter-Referenz in Kapitel II.
Beispiel	ORIGIN-Geschwindigkeit lesen p1x16p G Rückmeldung: P16 M#26$16 5

q - Permanenten Achsparameter lesen

Befehl	q %bn x %bm p
Parameter	%bn = Achsnummer %bm = Parameternummer
Funktion	Wert eines permanenten Achsparameters lesen.
Rückmeldung	P%bm_M#26$%bm_%lv %bm = Parameternummer %lv = Parameterwert

	Anmerkung	Man unterscheidet zwischen temporären (so genannten Programm-) Parametern und permanenten (so genannten Benutzer-) Parametern. Benutzerparameter sind im Speicher abgelegt und bleiben auch nach dem Verlassen des ASCII-Modus oder dem Abschalten der Positioniersteuerung gültig.
	Querverweis	ASCII-Kommando: S, p, w Parameter-Referenz in Kapitel II.
	Beispiel	ORIGIN-Geschwindigkeit lesen q1x16p G Rückmeldung: P16 M#26$16 5
r - Temporären Nullpunkt löschen	**Befehl**	r %bn x
	Parameter	%bn = Achsnummer
	Funktion	Definition des temporären Nullpunktes löschen.
	APOSS-Verweis	RST ORIGIN
	Beispiel	Temporärnullpunkt setzen Positionieren Temporärnullpunkt löschen o1x50000 a1x0 r1x G
s - Bewegungsablauf unterbrechen	**Befehl**	s %bn x
	Parameter	%bn = Achsnummer
	Funktion	Bewegungsablauf mit definierter Bremsrampe unterbrechen.
	Anmerkung	Dieser Befehl stoppt die entsprechende Achse abrupt (Version H) bzw. mit der eingestellten Beschleunigung (Version N)
	APOSS-Verweis	MOTOR STOP
	Beispiel	Positionieren, 0.5 Sekunde warten, Positionierung unterbrechen A1x100000 W500 s1x G
t - Interne Systemzeit abfragen	**Befehl**	t
	Funktion	Abfragen der internen Systemzeit seit Einschalten der Positioniersteuerung.
	Rückmeldung	M#57_0_%lv %lv = Systemzeit (in ms)
	APOSS-Verweis	TIME
	Beispiel	Systemzeit abfragen t G Rückmeldung: M#57 0 532355 (=> 8min, 52s, 355ms)

v - Analogausgang setzen

Befehl	v %bn x %bv
Parameter	%bn = Achsnummer %bv = Bytewert(-127 bis 128)
Funktion	Setzen des Analogausganges von der entsprechenden Achse auf einen entsprechenden Wert.
Anmerkung	Vorher muss der Befehl MOTOR OFF ausgeführt werden (nur mit der HCTL-Version möglich)
APOSS-Verweis	OUTAN

w - Permanenten Achsparameter setzen

Befehl	w %bn x %bm p %lv
Parameter	%bn = Achsnummer %bm = Parameternummer %lv = Parameterwert
Funktion	Wert eines permanenten Achsparameters neu setzen.
Anmerkung	Die geänderten Parameterwerte werden im Speicher abgelegt und bleiben auch nach dem Abschalten der Positioniersteuerung erhalten. Man spricht in diesem Fall von Benutzerparameter.
Querverweis	ASCII-Kommando: S, p, q Parameter-Referenz in Kapitel II.
Beispiel	Origin-Geschwindigkeit permanent neu setzen w1x16p5 G

x - Sollposition auf Istposition setzen

Befehl	x %bn
Parameter	%bn = Achsnummer
Funktion	Sollposition auf Istposition setzen
Beispiel	x1x G

y - Abgebrochenes Programm fortsetzen

Befehl	y
Funktion	Setzt ein abgebrochenes Programm fort
APOSS-Verweis	CONTINUE
Beispiel	y G

[- Achsfestlegung für Bahnbewegungen

Befehl	[
Funktion	Legt fest, welche beiden Achsen an der Bahnbewegung teilnehmen.
APOSS-Verweis	MOVE
Beispiel	& [1x[2x G

\ - Bahngeschwindigkeit festlegen

Befehl	\
Funktion	Legt die Bahngeschwindigkeit fest in Benutzereinheiten pro Sekunde.
APOSS-Verweis	MVEL
Beispiel	\10000 G

] - Vektorlänge in Benutzereinheiten

Befehl	]
Funktion	Festlegung der Länge der folgenden Vektoren; sie müssen alle die gleiche Länge haben.
Anmerkung	Derzeit muss das Verhältnis Vektorlänge/Geschwindigkeit < 3 sein.
Beispiel	] 1000

^ - Anfang der Vektorfolge

Befehl	^
Funktion	Signalisiert den Anfang der Vektorfolge, gefolgt von der Anzahl (nn) der Vektoren. Der zweite Parameter bestimmt, welche Achse im Drehzahlmodus betrieben wird. Die Vektoren werden aber während der Programmausführung über die serielle Schnittstelle eingelesen. Jeder Vektor wird nach der Überprüfung der Kontrollsumme mit V0 bestätigt. Im Falle eines Fehlers wird V255 zurückgeschickt. Die Daten werden binär geschickt. Die Kontrollsumme wird über Bytes gebildet und als negative Zahl über RS232 geschickt
Beispiel	\20000]100 & [1x[2x ^50 3 G V0 14 99 20 –133 V0 … X0 G \20000]100 & [1x[2x ^50 0 G V0 14 99 –133 V0 … X0 G

_ - Anfang der Vektorfolge

Befehl	_
Funktion	Signalisiert den Anfang der Vektorfolge, gefolgt von der Anzahl (nn) der Vektoren. Danach folgen die Vektoren, wobei die Vektoren immer relative Werte enthalten, zuerst für die erste Achse und dann für die zweite, jeweils durch ein Leerzeichen getrennt. Der Befehl wird durch einen Nullvektor (0 0) beendet, also beide Werte 0.
Anmerkung	Die einzelnen Koordinaten dürfen nicht größer sein als MAXINT (–32768 bis 32767)
Beispiel	_nn 14 99 20 98 … 00 G

` - Nullpunkt versetzen

Befehl	`
Funktion	Versetzt den Nullpunkt um den angegebenen Wert.
Anmerkung	Befehl ist nur bei der HCTL-Version möglich.
Beispiel	`1x 30000

Kapitel IV ASCII-Echomodus

ASCII-Echomodus

Der Echomodus erhöht die Übertragungssicherheit durch den Austausch von Quittungssignalen zwischen Sender und Empfänger.

Funktionsweise des Echomodus

Alle empfangenen Zeichen werden von der Positioniersteuerung sofort wieder an den Sender zurückgesandt.

Der Sender kann die ursprünglich gesandte mit der empfangenen Information vergleichen und nach jedem vollständigen Befehl die Richtigkeit der Übertragung durch ein Quittungssignal bestätigen oder einen Übertragungsfehler signalisieren.

Wird ein Übertragungsfehler signalisiert, so werden die seit dem letzten Quittungssignal bzw. seit dem letzten G-Befehl empfangenen Zeichen gelöscht und die betreffenden Befehle nicht ausgeführt.

Werden keine Quittungssignale gesendet, so werden nach einem G-Befehl die zuvor übertragenen Befehle ebenfalls als quittiert gewertet und ausgeführt.

Die Aktivierung des Echomodus, die Quittierung bzw. Nicht-Quittung geschieht dabei über spezielle Zeichencodes, die jeweils nach einem kompletten Befehl gesendet werden können.

Die speziellen Zeichencodes sind:

Bezeichnung	ASCII-Wert dezimal	PC-Tastatur
SYN	22	**Strg** + **V**
ACK	06	**Strg** + **F**
NAK	21	**Strg** + **U**

Funktion und Bedeutung der Quittungssignale

SYN Innerhalb des ASCII-Kommandomodus wird der Echomodus alternierend ein- und ausgeschaltet.

ACK Die seit dem letzten NAK. ACK oder G-Befehl gesandten Befehle werden als korrekt quittiert. Die Befehle werden in diesem Fall als gültig abgelegt, bis sie durch einen G-Befehl ausgeführt werden.
Ein ACK-Signal innerhalb eines Befehls wird als Fehler gewertet!

NAK

1. NAK wird an die Positioniersteuerung gesendet:
   Alle seit dem letzten ACK-Signal oder G-Befehl empfangenen Zeichen werden aus dem Speicher der Positioniersteuerung gelöscht und die betreffenden Befehle werden nicht ausgeführt.
2. NAK wird von der Positioniersteuerung gesendet:
   Die Positioniersteuerung hat eines der empfangenen Zeichen nicht richtig verstanden. Alle weiteren Befehle werden ebenfalls so lange mit einem NAK quittiert, bis der Sender ebenfalls ein NAK sendet, um die Blockade zu beheben.

Kapitel IV Fehlermeldungen im ASCII-Kommandomodus

Fehlermeldungen im ASCII-Kommandomodus

Folgende Fehlermeldungen können, ergänzend zu den im vorigen Abschnitt behandelten Fehlern während der Befehlsausführung im ASCII-Kommandomodus auftreten.

P1_no_free_memor	Ursache	Es sind zu viele Programme gespeichert und im temporären Speicher steht nicht mehr genügend Platz zur Verfügung um weitere Befehle aufzunehmen.
	Hinweis	Speicher löschen
P2_number_expected	Ursache	Der Befehlsaufbau ist nicht korrekt. Bei einem Befehlsteil, an dem eine Zahl erwartet wurde, steht ein anderes Zeichen.
	Hinweis	Befehlsaufbau überprüfen
P3_axes_order _expected	Ursache	Der Befehlsaufbau ist nicht korrekt. Es wurde keine gültige Achskennung wie 1x, 1X oder 2x, etc. eingegeben.
	Hinweis	Befehlsaufbau überprüfen
P4_internal_error, _illeg._param	Ursache	Interner Fehler
	Hinweis	Tritt dieser Fehler auf, so sollte die Positioniersteuerung abgeschaltet und der technische Service benachrichtigt werden.
P5_illegal_ Parameter, _command_ignored	Ursache	Der Befehlsaufbau ist nicht korrekt. Der angegebene Parameter entsprach nicht dem erwarteten Typ (Zahl, Achsenkennung etc.). Der entsprechende Befehl wurde daher ignoriert.
	Hinweis	Befehlsaufbau überprüfen
P6_illegal_command _*%c*	Variable	*%c* = unbekannte Befehlskennung
	Ursache	Die übergebene Befehlskennung (erster Buchstabe) konnte keinem zulässigen ASCII-Kommando zugeordnet werden.
P7_number_too_big	Ursache	Die eingegebene Zahl weist mehr Ziffern auf als in diesem Befehlsteil zulässig sind.
P8_%nnn%c_ expected	Variable	*%c* = Befehlskennung *%nnn* = erwarteter Befehlsteil
	Bedeutung	Diese Meldung gibt an, welcher Befehlsteil eigentlich erwartet wurde.