gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel
Tel.: 06101/523100 • Fax: 06101/523101
eMail: info@graebner-elektronik.de • **Internet** http://www.graebner-elektronik.eu

Handbuch SLAP2

Handbuch Version 1.01
Software Version 2205
Hardware Version 2209

gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel

Tel.: 06101/523100 • Fax: 06101/523101

eMail: info@graebner-elektronik.de • **Internet** http://www.graebner-elektronik.eu

gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel
Tel.: 06101/523100 • Fax: 06101/523101
eMail: info@graebner-elektronik.de • **Internet** http://www.graebner-elektronik.eu

Allgemeines

Merkmale SLAP2

- Betrieb von bürstenbehafteten DC-Motoren
- Eine Versorgungsspannung im Bereich von 12..30V=
- Digitaler PID-Regler
- Positionierbereich +-16777216 (+-2^24)
- Maximaler Motorstrom ca. 2000mA (interne einstellbare Strombegrenzung)
- Eingänge für nicht-differenzielle Winkelencoder mit und ohne Null-Spur
- Zwei optogekoppelte Endschaltereingänge
- Platinenformat: ca. 36x48x19mm³

Die Baugruppe dient zur exakten Positionierung und Drehzahlregelung kleiner DC-Motoren bis zu einer Leistung von ca. 50W. Integriert ist ein digitaler PID-Regler dessen Parameter jederzeit anpassbar sind. Die Sollwert- und Parametervorgaben erfolgen über eine serielle RS232-Schnittstelle. Darüber erfolgen alle Vorgaben für den Motorbetrieb:
Positionier- oder Drehzahlregelbetrieb
Beschleunigung/Verzögerung
Anzufahrende Position/Drehzahl
Aktivierung/Deaktivierung der Endschalter
Achskalibrierung (Auf Endschalter und/oder Nullspur)
Abfrage der Ist-Position des Motors usw.

Anforderungen an den Motor

Anschließbar sind nur bürstenbehaftete Motoren mit inkrementalem Winkelencoder. Die **minimale Anschlussinduktivität** des Motors beträgt **1mH**. Sollte der eine geringere Induktivität haben, sind in die Motorzuleitungen Drosseln mit entsprechender Strombelastbarkeit einzufügen. Der Grund für diese Maßnahme ist, die Stromwelligkeit herabzusetzen um die Endstufen nicht zu zerstören. Beherzigen Sie diese Vorschrift nicht, kann das zu abgebrannten Leiterbahnen und defekten Endstufen führen.

Inbetriebnahme

Um mit der Baugruppe arbeiten zu können muss sie zunächst korrekt angeschlossen und konfiguriert werden.
Folgen Sie dabei den Anweisungen die in unserem Dokument “InbetriebSLAP0100.rtf” gegeben werden um die Baugruppe korrekt zu konfigurieren.
Fahren Sie nach der Konfiguration fort:
Schalten Sie die Stromversorgung ab und schließen Sie den Motor an.
Sorgen Sie dafür, dass der Motor frei drehen kann. Koppeln Sie notfalls die Mechanik ab. Das ist sehr wichtig ! Sie könnten sonst die Mechanik zerstören !
Schalten Sie die Spannungsversorgung wieder ein. Wieder müssen Sie die Einschaltmeldung lesen können und die grüne LED leuchtet.
Geben Sie zunächst den Befehl “*pm*” ein.
Jetzt kann es passieren, dass der Motor mit hoher Drehzahl plötzlich losläuft nachdem Sie den Rotor mit der Hand verdreht haben. Das ist nicht weiter schlimm (wenn Sie die Mechanik abgekoppelt haben). Schalten Sie die Spannungsversorgung ab und vertauschen Sie die Encoder-Anschlüsse für Spur A und Spur B. Schalten Sie wieder ein und versuchen Sie nochmals “pm” und das Verdrehen des Rotors. Nun müsste der Motor ruhig stehen und sich gegen die Kraft wehren mit der Sie den Rotor verdrehen, jedenfalls darf er nicht mit hoher Drehzahl drehen. Steht der Motor war’s das. Sehen Sie sich die verfügbaren Befehle an und probieren Sie sie aus.
Dreht der Motor immer noch mit hoher Drehzahl, überprüfen Sie alle Motoranschlüsse und wiederholen Sie die gesamte Prozedur ab Motoranschluss.
Hintergrundinformation zur Fehleranalyse:

Nach “pm” wird der Regler eingeschaltet und der Motor bestromt wenn der Rotor verdreht wird. Wenn der Drehsinn des Motors nicht mit dem Drehsinn des Encoders übereinstimmt kommt es dazu, dass der Motor mit hoher Drehzahl dreht.
Ein anderes Phänomen tritt auf, wenn eines der Encodersignale (A oder B) fehlt. Dann kann die Elektronik nicht erkennen, dass sich der Rotor bzw. der Encoder gedreht hat. Damit bleibt der Motor stromlos und man kann den Rotor verdrehen obwohl der Regler mit “pm” eingeschaltet wurde.
Sie können jederzeit überprüfen, ob der Encoder angeschlossen ist. Das geht einfach mit dem Befehl “rp”. Er gibt die aktuelle Encoderposition zurück. Der interne Positionszähler läuft, solange die Spannungsversorgung eingeschaltet ist, vollkommen unabhängig vom Betriebsmodus.

LED-Betriebsanzeige

Die auf dem Board befindliche LED zeigt unmittelbar nach Power-On den Betriebsbereitzustand an. In den Modi Positioniermodus (“pm”), Drehzahlregelmodus (“vm”) und PWM-Modus (“spwm”) signalisiert sie durch ihr Verlöschen (oder Flackern), dass der Motorstrom von der internen Elektronik begrenzt wird. Die Motorstrombegrenzung ist im Auslieferungszustand auf 2000mA eingestellt. Zur Einstellung und Abfrage der Motorstrombegrenzung dienen die Befehle “scl n” und “rcl”.

Befehlstabelle

Zu einigen Befehlen gehört ein entsprechender Parameter. Er ist zusammen mit dem Befehl einzugeben und wird in der folgenden Liste durch *n* repräsentiert.

**Alle Befehle der „alten“ Module SLAP gibt es auch in den neuen Modulen SLAP2.
Wir empfehlen jedoch allen Neuanwendern nur noch den neuen Befehlssatz zu benutzen. In der folgenden Befehlstabelle sind alle Befehle des neuen Befehlssatzes** *fett kursiv* **gedruckt. Danach sind die Befehle aufgelistet, die es in ihrer z.T. speziellen Form bei den SLAP/SLAP-Modulen gab.**

Befehl	
se n	SElect. Modul mit der Adresse n selektieren.
pm	Position Mode On. Positioniermodus einschalten.
vm	Velocity Mode On. Drehzahlregelmodus einschalten
spwm n	Set PWM. Ungeregelten Motorstrom erzeugen. n=-254..+254
st	STop. Eingestellten Modus (pm, vm, spwm) abschalten.
cal n	CALibrate. n=0..5. Siehe weiter unten.
ma n	Move Absolute to position n. Auf Position n verfahren.
mr n	Move Relative to position n. Um n Positionen verfahren.
rp	Read Position. Auslesen der augenblicklichen Motorposition.
sp n	Set Position to n. Setzen des Positionszählers auf n.
sv n	Set Velocity to n. Setzen der Verfahrgeschw. bzw. Drehzahl auf n.
rv	Read Velocity. Auslesen der eingestellten Geschw./Drehzahl.
sa n	Set Acceleration to n. Setzen der Beschleunigung auf n.
ra	Read Acceleration. Auslesen der eingestellten Beschleunigung.
scv n	Set Calibration Velocity to n. Setzen der Kalibriergeschw. (nur CAL)
rcv	Read Calibration Velocity. Auslesen der Kalibriergeschwindigkeit.
sca n	Set Calibration Acceleration. Setzen der Kalibrierbeschleunigung (nur CAL)
rca	Read Calibration Acceleration. Auslesen der Kalibrierbeschleunigung.
kp n	Setzen des P-Koeffizienten des PID-Reglers.
ki n	Setzen des I-oeffizienten des PID-Reglers.
kd n	Setzen des DKoeffizienten des PID-Reglers.
qp	Auslesen des P-Koeffizienten.
qi	Auslesen des I-oeffizienten.
qd	Auslesen des DKoeffizienten.

gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel

Tel.: 06101/523100 • Fax: 06101/523101

eMail: info@graebner-elektronik.de **• Internet** http://www.graebner-elektronik.eu

pe	Position Error. Auslesen der Soll-/Ist-Positionsdifferenz.
rss	Read StatuS. Ähnlich ss aber mit anderen, neuen Bits
rrss	Read Read StatuS. Alle Flags aus rss in Klarschrift auflisten (nicht in Programmen benutzen, Ausgabe ist mehrzeilig).
sipw n	Set In Position Window. Identisch mit ws.
ripw	Read In Position Window. Identisch mit rw.
sipt n	Set In Posítion Time. Siehe weiter unten.
ript	Read In Positin Time. Auslesen des mit sipt gesetzten Wertes.
id	Identification. Auslesen des Modultyps, Softwareversion und Seriennummer.
pg	ProGram. Einstellungen im EEPROM speichern(nicht während pm oder vm!).
de n	Delete eeprom. n=Serialnumber, siehe id. Löschen des EEPROMS. Das EEPROM wird nach Power On mit Default-Werten beschrieben.
ssyscon n	Set SYStemCONfiguration. Setzen der Systemkonfiguration.
rsyscon	Read SYStemCONfiguration. Auslesen der Systemkonfiguration.
rrsyscon	Read Read SYStemCONfiguration. Auslesen der Systemkonfiguration. Anzeige in Klartext (nicht in Programmen benutzen, Ausgabe ist mehrzeilig).
rep	REPort. Alle wichtigen Einstellungen werden aufgelistet (nicht in Programmen benutzen, Ausgabe ist mehrzeilig).
ssb n	Set Syscon Bit. Setze Bit in der Systemkonfiguration auf 1. n=Bitnummer.
rsb n	Reset Syscon Bit. Setze Bit in der Systemkonfiguration auf 0(Null). n=Bitnummer.
scl n	Set Current Limit to n. Setzen des max. zulässigen Motorstroms in mA.
rcl	Read Current Limit. Auslesen des mit scl gesetzen max. Motorstroms. Anzeige in mA.
rve	Read Velocity. Gemessene Drehzahl/Verfahrgeschwindigkeit des Motors.
rep	REPort. Auflistung aller benutzerdefinierbaren Einstellungen (siehe unten) (nicht in Programmen benutzen, Ausgabe ist mehrzeilig).
li n	Limit1+2 n=0=Off / n=1=On
ca n	CAlibrate. n=0..5. Siehe weiter unten.
zp	Zero Position. Setzen des Positionszählers auf 0(Null).
ss	StatuS. Auslesen des Statusbytes. Siehe weiter unten.
il n	Invert Limits. n=0..3. Invertiere Pegel der Endschaltereingänge.
ql	Query Limits. Auslesen der Einstellungen von il.
ws n	Window Size. Setzen der Fehlerfenstergröße (siehe sipw).
rw	Read Window. Auslesen der Fehlerfenstergröße (siehe ripw).
sc n	Set Current limit to n. Strombegrenzung auf 0..15 setzen. 15 entspricht ca. 2000mA.
rc	Read Current limit. Auslesen der mittels sc gesetzten Strombegrenzung.

Hinweis:
Die Ausführung eines Befehls kann verhindert werden wenn SLAP2 VOR dem Return-Zeichen ein Strg-X (dezimal 24, hex $18) empfängt. Mit diesem Kontrollzeichen wird der interne Empfangspuffer geleert.

pm

Position Mode
Damit wird der Regler eingeschaltet und versuchen die augenblickliche Motorposition zu halten.

vm

Velocity Mode
Der Regler wird eingeschaltet und der Motor sofort mit der eingestellten Beschleunigung (siehe „sv“) auf die eingestellte Drehzahl beschleunigt. Währenddessen kann jederzeit die Drehzahl verändert werden.

st

STop
Der Regler wird abgeschaltet, der Motor wird stromlos. Der eingestellte Modus („pm“ oder „vm“) wird verlassen/beendet.

ma n

Move Absolute
Dar Motor wird auf die mitübergebene Position verfahren. Beispiel:
ma -10000
wird den Motor auf die Position –10000 verfahren.
Das Ende der Bewegung kann über die Flag's “move” und/oder “inpos” des Status ermittelt werden.
Voraussetzung: Der Positioniermodus „*pm“* muss eingeschaltet sein.

mr n

Move Relative
Dar Motor wird um die angegebenen Positionen verfahren. Beispiel:
mr 2000
wird den Motor um 2000 Positionen verfahren.
Das Ende der Bewegung kann über die Flag's “move” und/oder “inpos” des Status ermittelt werden.
Voraussetzung: Der Positioniermodus „*pm“* muss eingeschaltet sein.

sv n

Set Velocity
Setzen der (maximalen) Verfahrgeschwindigkeit/Motordrehzahl. Beispiel:
sv 500
Berechnung der Drehzahl:
ve = der Wert der mit sv übergeben wird
UpM = die gewünschte Drehzahl in Umdrehungen pro Minute
Linien = Linien des Encoders
Drehzahl[UpM] = SV[Counts] * 140,417 / Linien
SV[Counts] = Drehzahl[UpM] * Linien / 140,417

Beispiel:
Der benutzte Encoder ist mit 512 Linien angegeben. Um den Motor nun mit einer Drehzahl von 2500UpM drehen zu lassen ist der Wert 9116 zu übergeben:
(2500 * 512)/140,417=9115.7

rv

Read Velocity
Auslesen der eingestellten (maximalen) Verfahrgeschwindigkeit bzw. Drehzahl.

Am Römerbrunnen 11a • 61118 Bad Vilbel
Tel.: 06101/523100 • Fax: 06101/523101
eMail: info@graebner-elektronik.de • **Internet** http://www.graebner-elektronik.eu

sa n

Set Acceleration
Setzen der Beschleunigung. Beispiel:
sa 50
Berechnung der Beschleunigung:
ac = der Wert der mit sa übergeben wird
UpM/M = die gewünschte Beschleunigung in Umdrehungen pro Minute in der Minute
Linien = Linien des Encoders
Beschleunigung[UpM²] = SA[Counts] * 35946.7 / Linien
SA[Counts] = Beschleunigung[UpM²] * Linien / 35946.7

Beispiel:
Der benutzte Encoder ist mit 512 Linien angegeben. Um den Motor nun mit 5000UpM/M zu beschleunigen ist der Wert 71 zu übergeben:
(5000 * 512) / 35946.7=71.21

ra

Read Acceleration
Auslesen der eingestellten Beschleunigung.

kp n / ki n / kd n

Einstellen der Reglerparameter P, I und D. Beispiel:
kp 40
ki 40
kd 80

qp / qi / qd

Auslesen der eingestellten Reglerparameter P, I und D

sp n

Set Position
Mit dem Befehl kann der interne Motorpositionszähler auf jeden Wert zwischen –16777216 und +16777216 (+-2^24) gesetzt werden. Beispiel:
sp 0
sp 5000
Der Befehl kann immer ausgeführt werden, selbst während eine Bewegung stattfindet!

rp

Read Position
Die augenblickliche Motorposition wird ausgelesen. Der Befehl funktioniert immer, egal, in welchem Modus (“pm” oder “vm” oder auch “st”) sich der Regler befindet. Zahlenbereich +-2^24.

cal n

CALibrate
Nach dem Einschalten der Spannungsversorgung hat der interne Positionszähler den Wert 0 (Null). Das hat jedoch nichts mit der mechanischen, tatsächlichen Position des Motors zu tun. Daher ist es auch hier, wie bei allen inkrementalen Systemen, notwendig, die tatsächliche mechanische Position zu ermitteln. Das geschieht, indem der Motor solange in eine bestimmte Richtung verfahren wird bis ein Schalter durch den Motor betätigt wird. Dieser Schalter ist natürlich in geeigneter Weise mit dem Regler verbunden und liefert daher die erforderliche Rückmeldung.
An die SLAP2 lassen sich maximal zwei Endschalter und ggf. der Index-Puls des Winkelencoders anschließen.
Hinweise:

1) Nach erfolgreicher Kalibrierung wird ein Flag im Statusregister gesetzt. Siehe „ss“, Flag **cal**.

2) Ein mit „cal x“ eingeleiteter Kalibriervorgang wird abgebrochen wenn das Zeichen Strg-K (dezimal 11, Hex $0b) empfangen wurde. In diesem Fall bleibt das Flag **cal** im Statusregister auf 0 (Null).

Der mit “cal” übergebene Parameter *n* bestimmt nun, welcher der drei Eingänge als Referenzschalter dienen soll:

cal 0	Endschalter 1 dient als Referenzschalter (negative Drehrichtung)
cal 1	Endschalter 2 dient als Referenzschalter (positive Drehrichtung)
cal 2	Endschalter 1 und der Index-Puls dienen als Referenzschalter (negative Drehrichtung)
cal 3	Endschalter 2 und der Index-Puls dienen als Referenzschalter (positive Drehrichtung)
cal 4	Nur der Index-Puls dient als Referenzschalter (negative Drehrichtung)
cal 5	Nur der Index-Puls dient als Referenzschalter (positive Drehrichtung)

Negative Drehrichtung: Der Motor dreht in Richtung negative Positionen
Positive Drehrichtung: Der Motor dreht in Richtung positive Positionen

cal 0
Der Motor wird mit der mit *scv* eingestellten Drehzahl und der durch *sca* vorgegebenen Beschleunigung in Richtung Endschalter 1 verfahren. Wird der Endschalter betätigt, wird der Motor sofort gestoppt. Danach wird der Motor mit 1/16 der mit *scv* eingestellten Drehzahl und 1/16 der mit *sca* eingestellten Beschleunigung in Richtung Endschalter 2 verfahren bis der Endschalter 1 wieder gelöst ist. Der Motor stoppt dann sofort und die Kalibrierfahrt ist beendet. Die Motorposition kann nun mittels “rp” ausgelesen oder mit “sp..” auf den gewünschten Wert gesetzt werden.
Die Vorgaben von *scv* und *sca* sind nur während des Kalibrierens wirksam. Danach sind automatisch wieder die Werte von *sv* und *sa* wirksam.
cal 1
Der Motor wird mit der mit *scv* eingestellten Drehzahl und der durch *sca* vorgegebenen Beschleunigung in Richtung Endschalter 2 verfahren. Wird der Endschalter betätigt, wird der Motor sofort gestoppt. Danach wird der Motor mit 1/16 der mit *scv* eingestellten Drehzahl und 1/16 der mit *sca* eingestellten Beschleunigung in Richtung Endschalter 1 verfahren bis der Endschalter 2 wieder gelöst ist.
Der Motor stoppt dann sofort und die Kalibrierfahrt ist beendet. Die Motorposition kann nun mittels “rp” ausgelesen oder mit “sp..” auf den gewünschten Wert gesetzt werden.
Die Vorgaben von *scv* und *sca* sind nur während des Kalibrierens wirksam. Danach sind automatisch wieder die Werte von *sv* und *sa* wirksam.
cal 2
Der Motor wird mit der mit *scv* eingestellten Drehzahl und der durch *sca* vorgegebenen Beschleunigung in Richtung Endschalter 1 verfahren. Wird der Endschalter betätigt, wird der Motor sofort gestoppt. Danach wird der Motor mit 1/16 der mit *scv* eingestellten Drehzahl und 1/16 der mit *sca* eingestellten Beschleunigung in Richtung Endschalter 2 verfahren bis der Endschalter 1 wieder gelöst ist.
Der Motor dreht nun solange weiter bis der Index-Impuls des Winkelencoders erkannt wird.
Der Motor stoppt dann sofort und die Kalibrierfahrt ist beendet. Die Motorposition kann nun mittels “rp” ausgelesen oder mit “sp..” auf den gewünschten Wert gesetzt werden.
Die Vorgaben von *scv* und *sca* sind nur während des Kalibrierens wirksam. Danach sind automatisch wieder die Werte von *sv* und *sa* wirksam.
cal 3
Der Motor wird mit der mit *scv* eingestellten Drehzahl und der durch *sca* vorgegebenen Beschleunigung in Richtung Endschalter 2 verfahren. Wird der Endschalter betätigt, wird der Motor sofort gestoppt. Danach wird der Motor mit 1/16 der mit *scv* eingestellten Drehzahl und 1/16 der mit *sca* eingestellten Beschleunigung in Richtung Endschalter 1 verfahren bis der Endschalter 2 wieder gelöst ist.
Der Motor dreht nun solange weiter bis der Index-Impuls des Winkelencoders erkannt wird.
Der Motor stoppt dann sofort und die Kalibrierfahrt ist beendet. Die Motorposition kann nun mittels “rp” ausgelesen oder mit “sp..” auf den gewünschten Wert gesetzt werden.
Die Vorgaben von *scv* und *sca* sind nur während des Kalibrierens wirksam. Danach sind automatisch wieder die Werte von *sv* und *sa* wirksam.

cal 4
Der Motor wird mit der Drehzahl von *scv* und Beschleunigung von *sca* in Richtung Endschalter 1 bzw. in negative Richtung verfahren bis der Index-Impuls des Winkelencoder erkannt wird.
Der Motor stoppt dann sofort und die Kalibrierfahrt ist beendet. Die Motorposition kann nun mittels "rp" ausgelesen oder mit "sp.." auf den gewünschten Wert gesetzt werden.
Die Vorgaben von *scv* und *sca* sind nur während des Kalibrierens wirksam. Danach sind automatisch wieder die Werte von *sv* und *sa* wirksam.
cal 5
Der Motor wird mit der Drehzahl von *scv* und Beschleunigung von *sca* in Richtung Endschalter 2 bzw. in positive Richtung verfahren bis der Index-Impuls des Winkelencoder erkannt wird.
Der Motor stoppt dann sofort und die Kalibrierfahrt ist beendet. Die Motorposition kann nun mittels "rp" ausgelesen oder mit "sp.." auf den gewünschten Wert gesetzt werden.
Die Vorgaben von *scv* und *sca* sind nur während des Kalibrierens wirksam. Danach sind automatisch wieder die Werte von *sv* und *sa* wirksam.

ca n

Der Unterschied zwischen den Befehlen „ca" und „cal" besteht darin, dass bei „ca" nicht die Kalibriergeschwindigkeit „scv" benutzt wird sondern die Geschwindigkeit die mit „sv" eingestellt wurde. Ebenso wird zur Beschleunigung nicht „sca" sondern „sa" benutzt. Der Grund dafür ist historisch bedingt: Die Module SLAP kannten den Befehl „scv" bzw. „sca" nicht. *ca n* ist also aus Kompatibilitätsgründen implementiert.

scv n

Set Calibration Velocity
Setzen der (maximalen) Verfahrgeschwindigkeit/Motordrehzahl während des Kalibrierens. Beispiel:
scv 500
Zur Berechnungen siehe *sv*.

rcv

Read Calibration Velocity
Auslesen der eingestellten Kalibrier-Drehzahl.

sca n

Set Calibration Acceleration
Setzen der Beschleunigung während des Kalibrierens. Beispiel:
sca 50

rca

Read Calibration Acceleration
Auslesen der eingestellten Kalibrier-Beschleunigung.

pe

Position Error
Dieser Befehl liefert die Differenz zwischen der vom Bewegungsprofil-Generator (Rampen-Generator) erzeugten Motorsollposition und der tatsächlichen Motor-Istposition. Diese Differenz wird übrigens oft auch als Schleppfehler bezeichnet.
Der Schleppfehler gibt Auskunft über das gesamte mechanische und elektrische System. Daran kann erkannt werden ob eine Störung des Gesamtsystems vorliegt.
Wenn beispielsweise die Mechanik klemmt wird sich das sehr schnell als großer Schleppfehler bemerkbar machen. Es ist eine gute Idee, den Schleppfehler während einer Bewegung zyklisch abzufragen. Wie groß der Schleppfehler normalerweise ist, hängt ausschließlich vom Motor, den eingestellten Reglerparametern und der angeschlossenen Mechanik ab und kann hier nicht quantifiziert werden. Ermitteln Sie doch einfach den Schleppfehler in Ihrem einwandfrei arbeitenden

System und legen Sie damit einen Grenzwert fest. Wird der Grenzwert überschritten können Sie dann geeignete Maßnahmen einleiten.

ssyscon n

Set SYStem CONfiguration
Mit diesem Befehl (und dem zugehörigen Parameter) wird die Baugruppe konfiguriert.

Bit	Flag	
0	l1on	High= Endschalter 1 aktiv
1	l2on	High= Endschalter 2 aktiv
2	inv1	High= Pegel an Endschalter 1 wird logisch invertiert
3	inv2	High= Pegel an Endschalter 2 wird logisch invertiert
4	hex	High= Alle Werteausgaben erfolgen in hexadezimaler Form. z.B 0x1A00
5	ucon	High= Wird ein Kommando nicht verstanden erfolgt unmittelbar die Ausgabe des Strings „-1UC“

Mit den Flag’s „l1on“ und „l2on“ kann einer oder beide Endschalter abgeschaltet werden.
Mit den Flag’s „inv1“ und „inv2“ können die Logikpegel invertiert werden. Nützlich wenn Schließer statt Öffner betrieben werden.
Das Flag „hex“ bewirkt, dass alle Werteausgaben hexadezimal erfolgen. Beispiel: 0x3E8 (dezimal 1000).
Das Flag „ucon“ bewirkt, dass die Fehlermeldung „-1UC“ ausgegeben wird, falls das gegebene Kommando unbekannt oder nicht ausführbar war. Beispiel: Der folgende Befehl ist unbekannt
abc
-1UC
Die Fehlermeldung erfolgt im Antwortstring des (unbekannten) Befehls wenn das Flag „ucon“ auf High gesetzt wurde.

rsyscon

Read SYStem CONfiguration
Die eingestellte Konfiguration wird ausgelesen und angezeigt. Die Bedeutung der verschiedenen Flags, siehe „ssyscon“.

rrsyscon

Read SYStem CONfiguration
Achtung: Dieser Befehl darf nur benutzt werden, wenn die Kommunikation mit der SLAP2 über ein Terminalprogramm (z. B. GrTermW) erfolgt !
Der Befehl listet die Konfiguration der SLAP2 in Klartext auf. Die Ausgabe ist mehrzeilig !!!!

sipw n

Set InPos Window
Fenstergröße für das Flag “inpos” setzen. Beispiel
sipw 5
Nähere Erläuterungen finden Sie beim Befehl “rss” und dessen Flag “inpos” sowie im Anhang.

ripw

Read inpos Window
Gesetzte Fenstergröße auslesen. Es wird der Wert geliefert, der mittels “sipw n” gesetzt wurde.

sipt n

Set InPos TImer
Zeit für den Inpos-Timer in Schritten von 841.5µs vorgeben. Beispiel
sipt 50
Setzt die inpos-Verweildauer auf rund 42ms. Nähere Erläuterungen finden Sie beim Befehl “rss” und dessen Flag “inpos” sowie im Anhang.

ript

Read InPos TImer
Gesetzten Timer-Wert auslesen. Es wird der Wert geliefert der mittels “sipt n” gesetzt wurde.

rss

Read System Status

Bit	Flag	
0	limit1	Pegel am Endschaltereingang 1
1	limit2	Pegel am Endschaltereingang 2
2	vm	High= vm aktiv
3	pm	High= pm aktiv
4	move	High= Bewegung läuft
5	inpos	High= Motor ist in Position. Siehe sipw und sipt.
6	cal	High= System wurde mittels ca oder cal erfolgreich kalibriert.
7	oc	High= Motorstrom wird auf den Wert von sc bzw. scl begrenzt.
8	uc	High= Das vorherige Kommando war falsch/wurde nicht verstanden.

limit1 / limit 2
Durch Auswertung der beiden Flag‘s kann festgestellt werden, ob einer der Schalter durch die Mechanik betätigt ist oder nicht.
vm
Das Flag ist High, wenn der Drehzahlregelmodus (”vm”) eingeschaltet wurde. Es wird nach ”st” Low.
pm
Das Flag ist High, wenn der Positioniermodus (”pm”) eingeschaltet wurde. Es wird nach ”st” Low.
move
Das Flag wird High nachdem im Positioniermodus (”pm”) ein Befehl zum Verfahren des Motors (”ma” oder ”mr”) empfangen wurde. Mit dem Empfang von ”ma” oder ”mr” startet der Rampengenerator mit seinen Berechnungen und das Flag ”move” wird gesetzt. Das Flag bleibt High solange der Rampengenerator arbeitet, d.h. bis die vorgegebene Sollposition erreicht ist. Die Sollposition entspricht aber nicht unbedingt der Motor-Istposition (siehe ”pe” – Schleppfehler) zeigt aber, ob die nächste Bewegung gestartet werden kann.
inpos
Das Flag wird mit dem Starten einer Bewegung auf alle Fälle auf Low gesetzt. Nachdem der Rampengenerator fertig ist (Move-Flag=Low) wird die augenblickliche Motorposition mit der gewünschten Sollposition verglichen. Ist die Differenz zwischen Soll- und Istposition kleiner als der eingestellte Wert von "sipw”, wird ein Zeitgeber gestartet nach dessen Ablauf das Flag ”inpos” auf High gesetzt wird.
Die Länge des Timers wird von ”sipt” gesteuert. Gerät der Motor während der ablaufenden Zeit aus dem Positionsfenster, wird der Timer auf 0(Null) zurückgesetzt und beginnt automatisch von vorne, ”inpos” bleibt dadurch Low bis die gewünschten Bedingungen erfüllt sind.
cal
Die mittels „ca“ oder „cal“ aufgerufene Kalibrierung war erfolgreich verlaufen. Dieses Bit wird nach einem Reset oder Power Off / Power On automatisch auf Low gesetzt.
oc
Die interne Strombegrenzung ist wirksam. Der Motor will also offensichtlich mehr Strom als durch „sc“ oder „scl“ festgelegt wurde. Normalerweise ist das ein wichtiger Indikator, dass im Gesamtsystem (Motor und Mechanik) etwas nicht in Ordnung ist.
uc
Unknown Command. Der vorher empfangene Befehl war unbekannt/fehlerhaft oder konnte nicht ausgeführt werden.

rrss

Read System Status
Achtung: Dieser Befehl darf nur benutzt werden, wenn die Kommunikation mit der SLAP2 über ein Terminalprogramm (z. B. GrTermW) erfolgt !

Der Befehl listet die Status-Flag's der SLAP2 in Klartext auf. Die Ausgabe ist mehrzeilig !!!!

ss

Status
Aus Kompatibilitätsgründen gibt es auch nach wie vor „ss" mit den folgenden Flag's.

Bit	Flag	
0	limit1	Pegel am Endschaltereingang 1
1	limit2	Pegel am Endschaltereingang 2
2	vmode	High=vm aktiv
3	pmode	High=pm aktiv
4	move	High=Bewegung läuft
5	deccel	High=Verzögerung aktiv
6	esactiv	High=Beide Endschalter werden berücksichtigt/sind aktiv.
7	uc	High=Das vorherige Kommando war falsch/wurde nicht verstanden.

limit1 / limit 2
Durch Auswertung der beiden Flag' kann festgestellt werden, ob einer der Schalter durch die Mechanik betätigt ist oder nicht.
pmode
Das Flag ist High, wenn der Positioniermodus ("pm") eingeschaltet wurde. Es wird nach "st" Low.
vmode
Das Flag ist High, wenn der Drehzahlregelmodus ("vm") eingeschaltet wurde. Es wird nach "st" Low.
move
Das Flag wird High nachdem im Positioniermodus ("pm") ein Befehl zum Verfahren des Motors ("ma" oder "mr") empfangen wurde. Mit dem Empfang von "ma" oder "mr" startet der Rampengenerator mit seinen Berechnungen und das Flag "move" wird gesetzt. Das Flag bleibt High solange der Rampengenerator arbeitet, d.h. bis die vorgegebene Sollposition erreicht ist. Die Sollposition entspricht aber nicht unbedingt der Motor-Istposition (siehe "pe" – Schleppfehler) zeigt aber, ob die nächste Bewegung gestartet werden kann.
deccel
Das Flag ist High wenn der Motor ist in der Bremsphase/Verzögerungsrampe ist.
esactiv
Das Flag ist High wenn die Endschalter aktiviert sind.
uc
Unknown Command
Dieses Flag ist High wenn ein unbekanntes Kommando empfangen wurde.

Auswertung der Flags
In jeder Programmiersprache können die einzelnen Flag's recht leicht ausgewertet werden. Im folgenden Beispiel wird "C" verwendet:
Angenommen, der mit "ss" abgefragte Wert steht numerischer Wert in der Integer-Variabelen status, dann kann mit

```
if( (status&8)==0   printf ("Positioniermodus nicht eingestellt");
else                printf ("Positioniermodus eingeschaltet");
```

die entsprechende Meldung erzeugt werden.
Mit

```
if ( (status&3)!=0) printf("Ein Endschalter ist betätigt!");
```

wird die Meldung ausgegeben wenn einer der beiden (oder beide) Endschalter betätigt ist.

id

Versionsmeldung der Baugruppe mit Angabe der Geräteseriennummer.

pg

Die augenblicklich eingestellten Parameter werden dauerhaft im EEPROM gespeichert. Nach dem Abschalten und dem späteren Einschalten werden die gespeicherten Werte gelesen und automatisch wieder eingestellt.
Welche Werte gespeichert werden, entnehmen Sie bitte dem Anhang.

scv

Set Calibration Velocity
Während der Kalibrierung ("cal n") wird diese besondere Kalibriergeschwindigkeit benutzt. Sie sollte im Interesse der Kalibriergenauigkeit deutlich unter der sonst benutzten Verfahrgeschwindigkeit liegen – auch wenn das natürlich lange dauert. Aber dagegen ist auch ein Kraut gewachsen: Angenommen Sie möchten auf Endschalter 1 kalibrieren, dann können Sie zunächst mit einer mittleren Geschwindigkeit gegen den Endschalter fahren und dann erst den Befehl "cal.." benutzen. Das ergibt eine gewisse Zeitersparnis. Programmzeilen dazu:
sa 400
sv mittlere Geschwindigkeit
pm
ma –8000000
Nun warten bis das move-Flag des Statuswortes=Low ist und dann
cal 0
senden.
Hintergrundinfo: Sobald ein Endschalter betätigt wird, wird der Motor abrupt angehalten, die Bewegung abgebrochen und das move-Flag des Statuswortes auf Low gesetzt. Der Befehl „cal" erkennt, dass der Motor im Endschalter steht und wird den Motor aus dem Endschalter herausdrehen. Sobald der Endschalter geschlossen ist wird der Motor gestoppt.

rcv

Read Calibration Velocity
Zum Überprüfen des mit "scv" eingestellten Wertes.

sca n

Set Calibration Acceleration
Um ein sanftes Beschleunigen auch bei dem Befehl "cal n" zu gewährleisten. Allerdings wird der Motor gnadenlos gestoppt, wenn der betreffende Endschalter betätigt wird.

rca

Read Calibration Acceleration
Zum Überprüfen des mit "sca" eingestellten Wertes.

scl n

Set Current Limit
Das Modul SLAP2 verfügt über eine elektronische Motorstrombegrenzung. Der mit „scl" zu übergebende Wert *n* darf zwischen 0 und 2000 liegen und gibt an, bei welchem Motorstrom (in Milliampere) die Begrenzung einsetzen soll. Auch dieser Wert wird im EEPROM gespeichert und nach Power Off/Power On wieder hergestellt.
Beispiel für eine Motorstrombegrenzung von 500mA:
scl 500
Hinweis: Die (grüne) LED auf der Leiterplatte signalisiert durch ihr Verlöschen oder Flackern, dass die Strombegrenzung wirksam ist. Zudem wird im Statusregister (siehe „ss") das Flag „oc" (Over Current) gesetzt.
Anmerkung: Der Einsatz der Strombegrenzung ist stark vom angeschlossenen Motor abhängig. Daher ist die Einstellung in Milliampere relativ ungenau aber gut reproduzierbar. Ist es in Ihrer Applikation wichtig, dass der Motorstrom einen bestimmten Wert nicht überschreitet, nehmen Sie die Einstellung

Am Römerbrunnen 11a • 61118 Bad Vilbel
Tel.: 06101/523100 • Fax: 06101/523101
eMail: info@graebner-elektronik.de • **Internet** http://www.graebner-elektronik.eu

des max. Motorstroms mit Hilfe eines üblichen Multimeters vor das Sie in die Motorzuleitungen einschleifen.

rcl

Read Current Limit
Zum Überprüfen des mit “scl” eingestellten Wertes.

spwm

Set PWM
Mit diesem Befehl (Wertebereich –255 bis +255) kann ein ungeregelter Motorstrom erzeugt werden. Bei bürstenbehafteten Motoren genügt der Anschluss der Motorzuleitungen. Der Anschluss der Winkelencoders ist nicht zwingend notwendig.
Der Befehl kann dazu dienen Motoren zu überprüfen, oder in speziellen Fällen ein (ungeregeltes) Drehmoment zu erzeugen – nützlich um zu verhindern, dass sich eine Spule abwickelt.

se n

Dieser Befehl selektiert die Baugruppe mit der Nummer “n”. Z.B.:
se 2
Nach dem Senden des Befehls kommuniziert Ihr Rechner mit der Baugruppe 2 solange bis eine andere Baugruppe mittels “se n” angewählt wird.
Es ist also nicht nötig und nicht sinnvoll vor jedem Befehl eine Adressierung vorzunehmen. Eine einmal selektierte Baugruppe bleibt solange selektiert bis eine andere Baugruppe gezielt gewählt wird.
Ein Hinweis noch:
Eine Baugruppe im Verbund sollte die Adresse 0 haben, denn nur diese Baugruppe meldet sich automatisch mit ihrer Kennung nach “Power-On”.
Baugruppen mit anderen Adressen melden sich nicht automatisch.
Welche Adresse ein Modul hat hängt von der Beschaltung der Eingänge AD0 bis AD3 ab. Die Eingänge haben einen Pull Up-Widerstand. Sind alle Eingänge AD0..AD3 unbeschaltet hat das Modul die Adresse 15. Sind alle Eingänge auf GND gelegt hat das Modul die Adresse 0(Null).
Sie legen also die Adresse eines Moduls mit der Verdrahtung auf der Mutterplatine fest. Auf diese Art und Weise müssen neue Module nicht vor Einsatz in ihrer Adresse konfiguriert werden.

rve

Solange die Stromversorgung eingeschaltet ist wird die Drehzahl des Motors gemessen. Dieser Wert kann mit „rve“ ausgelesen werden.

rep

Dieser Befehl listet in leicht zu lesender Form alle benutzerdefinierbaren Parameter und Einstellungen auf und kann dazu dienen Konfigurierungsfehlern auf die Spur zu kommen. Dazu gibt es in unserer Terminalsoftware GrTermW114 eine Funktion, die es dem Anwender leicht macht diesen Report via eMail zu versenden. Drücken Sie bei angeschlossenem und eingeschaltetem Modul die F4-Taste so wird von der Terminalsoftware der Befehl „rep“ an das Modul gesendet und alles was das Modul zurücksendet in das File „report.txt“ geschrieben. Das File befindet sich im gleichen Verzeichnis wie GrTermW114 und kann nun leicht als Attachment via eMail an uns verschickt werden.

gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel

Tel.: 06101/523100 • Fax: 06101/523101

eMail: info@graebner-elektronik.de **• Internet** http://www.graebner-elektronik.eu

Lage der Anschlüsse

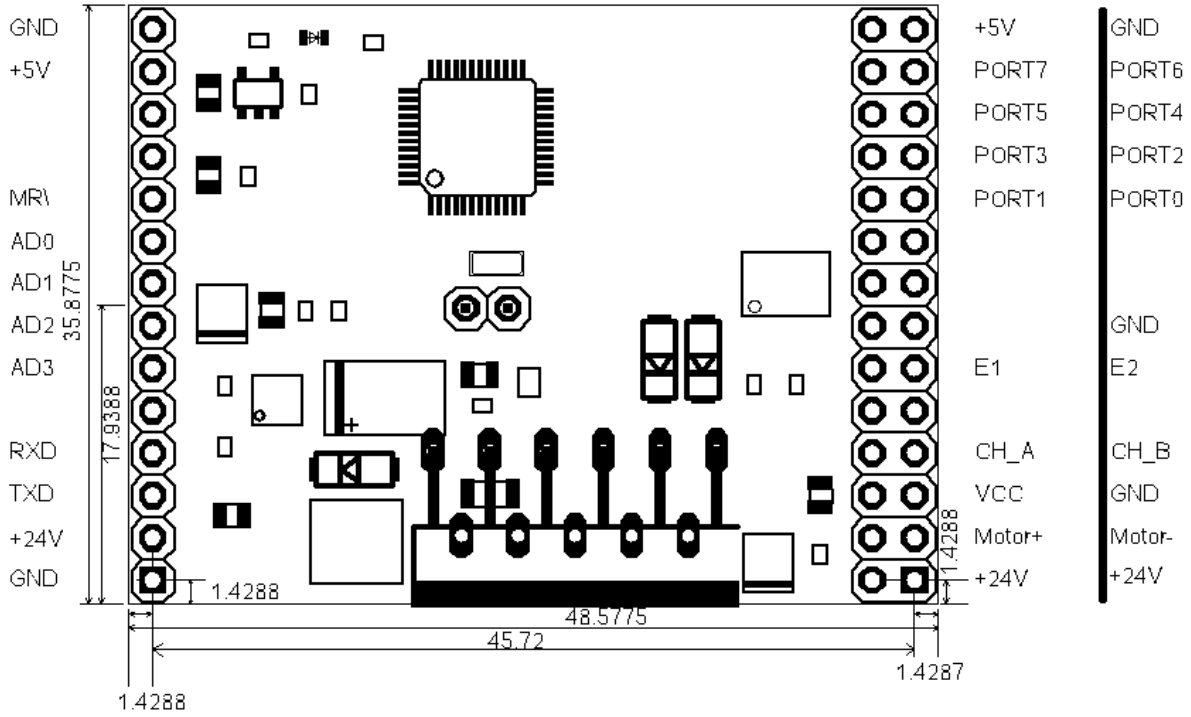

Bild: Draufsicht

Achtung:

Bei den Pin's +5V und VCC handelt es sich um Ausgänge !

Die einzige Versorgungsspannung wird an die +24V-Pins gelegt.

Linke einreihige Stiftleiste (A)

Pin	Signal	Bezeichnung
1	GND	Masse (Im Bild der rechteckige Pin)
2	+24V	Spannungsversorgung der Baugruppe
3	TxD	Sendeleitung des Moduls
4	RxD	Empfangsleitung des Moduls
5		Nicht Kontaktieren!
6	AD3	Adresseingang 3 (TTL-Pegel)
7	AD2	Adresseingang 2 (TTL-Pegel)
8	AD1	Adresseingang 1 (TTL-Pegel)
9	AD0	Adresseingang 0 (TTL-Pegel)
10	MR\	Externer Reset-Eingang (Low aktiv)
11		Nicht Kontaktieren!
12		Nicht Kontaktieren!
13	+5V	Ausgangsspannung +5V max. 50mA
14	GND	Masse

gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel
Tel.: 06101/523100 • Fax: 06101/523101
eMail: info@graebner-elektronik.de • **Internet** http://www.graebner-elektronik.eu

Rechte zweireihige Stiftleiste (B)

1	+24V	Spannungsversorgung der Baugruppe
2	+24V	Spannungsversorgung der Baugruppe
3	Motor-	Motoranschluß ”-”
4	Motor+	Motoranschluß ”+”
5	GND	Masse (Versorgung für Winkelenkoder)
6	+5V	+5V (Versorgung für Winkelenkoder (Ausgang))
7	CH B	Kanal B des Winkelenkoders
8	CH A	Kanal A des Winkelenkoders
9		Nicht Kontaktieren!
10		Nicht Kontaktieren!
11	E2	Endschaltereingang 2
12	E1	Endschaltereingang 1
13	GND	Masse
14		Nicht Kontaktieren!
15		Nicht Kontaktieren!
16		Nicht Kontaktieren!
17		Nicht Kontaktieren!
18		Nicht Kontaktieren!
19	Null	TTL-Eingang für Null-Spur des Winkelenkoders(High-Impuls)
20		Nicht Kontaktieren!
21		Nicht Kontaktieren!
22		Nicht Kontaktieren!
23		Nicht Kontaktieren!
24		Nicht Kontaktieren!
25		Nicht Kontaktieren!
26		Nicht Kontaktieren!
27	GND	Masse
28	+5V	+5V / max. 50mA (Ausgang)

Bürstenbehaftete Motoren (Motor+ und Motor-)
werden an Pin B3 und B4 angeschlossen.

Winkelencoder (CH A und CH B)
werden an die Pins B7 und B8 angeschlossen. Falls der Encoder über eine Nullspur verfügt kann diese an Pin B19 (Null) angeschlossen werden.

Die Endschalter (E1 und E2)
führen auf die Kathoden der LED’s in den Optokopplern. Deren Anoden liegen über 2,2kOhm an den internen +5V. Die Anschlüsse sind B11 und B12.
Wenn diesen Eingängen eine externe Spannung von >5V zugeführt wird, werden die Optokoppler unweigerlich zerstört !

Die Spannungsversorgung
von 12..24V für die Baugruppe wird nur über die Pins A2, B1 und B2 hergestellt.
Masse der Spannungsversorgung möglichst an alle GND-Pins, hauptsächlich aber alle der B-Leiste.
*Das ist die einzige zuzuführende Spannung und sie ist aus technischen Gründen **nicht verpolsicher**!*
Aus dieser Versorgung wird die Motorendstufe gespeist und die intern benötigten +5V mittels Schaltregler erzeugt.

RS232 (TxD und RxD)
über die Pins A3 und A4. Es werden RS232-konforme Pegel erwartet bzw. erzeugt.

Reset (MR\)
Der Pin A10 (MR\) dient zum externen resetten des Micrcontrollers. Um einen Reset auszulösen muss dieser Pin auf GND gezogen werden. Eine Einspeisung von gleichgültig welcher Spannung ist nicht

erlaubt. Der Pin ist mit dem Ausgang eines Reset-Controllers verbunden der auch die 5V-Versorgung überwacht. Kommt es zu einem Brown-Out (kurzzeitigem Spannungseinbruch) wird der Reset-Controller den Microcontroller resetten. Wird eine externe Spannung eingespeist ist diese Funktion nicht mehr gewährleistet. U.U. kann das zur Zerstörung des Reset-Kreises führen. Der Pin darf also nur mit einem Open-Collector oder Relaiskontakt beschaltet werden (Pin A10 an Anode einer Diode und deren Kathode an TTL-Ausgang wird auch funktionieren).
Wird der Pin nicht benutzt, muss er unbeschaltet bleiben.

ANHANG

Allgemeines, Zusammenhänge und Hintergründe

Hier werden einige Zusammenhänge innerhalb der Baugruppe beschrieben die zum besseren Verständnis der internen Abläufe dienen soll.

Endschalteranschluss

Bei rotatorischen Anwendungen gibt es i. A. keine Endschalter. Dann bleiben die Anschlüsse unbeschaltet jedoch muss der SLAM2 mitgeteilt werden, dass keine Endschalter vorhanden sind. Hierfür gibt es die Bit's „limit1" und „linit2" in der Systemkonfiguration (siehe „ssyscon").
Die Endschaltereingänge führen auf die Kathoden der LED's in den Optokopplern.
Die Anoden dieser LED's sind über einen Vorwiderstand mit +5V verbunden.
Extrem wichtig ist der korrekte Anschluss der Endschalter im Bezug auf räumliche Anordnung und Drehsinn des Motors. Endschalter 1 wird der Schalter, der betätigt werden soll wenn der Motor in negative Richtung verfahren wird. Endschalter 2 wird der Schalter, der betätigt wird, wenn der Motor in positive Richtung verfahren wird.
Stimmt diese Zuordnung nicht, lässt sich der Motor nicht verfahren oder fährt auf den mechanischen Anschlag. Ferner werden die internen Kalibrierroutinen („cal") nicht ordnungsgemäß funktionieren.

Spannungsversorgung

Die SLAP2 benötigt nur eine Spannungsversorgung im Bereich 12..24V.
Aus dieser Versorgungsspannung werden alle anderen Hilfsspannungen gewonnen, die auf dem Board benötigt werden. Auch die Motoren werden aus dieser Spannung gespeist. Es ist also darauf zu achten, dass genügend Strom zur Verfügung steht, das Netzteil also entsprechend leistungsfähig ist.
Die SLAP2 nimmt im Leerlauf weniger als 40mA auf. Zu diesem Leerlaufstrom addieren Sie den zulässigen maximalen Strom des Motors und geben eine Reserve hinzu. Diese Reserve richtet sich danach, ob der Motor in Ihrer Applikation mit hohen Beschleunigungswerten verfahren werden muss. Erfahrungsgemäss treten beim Beschleunigen und Verzögern die höchsten Motorströme auf. Wenn Sie sicher gehen wollen, wählen Sie die Reserve so groß wie der maximal zulässige Motorstrom.
Übrigens wird der Motorstrom permanent überwacht und dafür gesorgt, dass er nicht grösser als ca. 5A werden kann. Dabei handelt es sich um eine rein elektronische Schutzmassnahme die immer nur dann greift, wenn die Bottom-Transistoren (Transistoren nach GND) durchgeschaltet werden. Fliesst zu diesem Zeitpunkt ein höherer Strom, werden diese Transistoren sofort gesperrt. Dieser Vorgang wiederholt sich beim nächsten PWM-Zyklus. Das alles geschieht automatisch durch die Hardware der SLAP2 und ist nach außen hin nicht sichtbar, äußert sich jedoch mit einem merkwürdigen Kratzen des Motors.
Die mit 5V bezeichneten Klemmen stellen Ausgänge dar die z.B. zur Versorgung des Winkelencoders dienen. Wird an diese Klemmen eine externe Spannung angelegt, wird die Baugruppe sofort zerstört und ist nicht mehr reparierbar!

Winkelencoder

An die SLAP2 können fast alle handelsüblichen Encoder angeschlossen werden. Einzige Einschränkung: Deren Ausgangssignale dürfen 5V nicht überschreiten.

Nichtdifferenzielle Encoder gibt es in vielen verschiedenen Ausführungen. Wenn sie mit 5V gespeist werden müssen, können auch sie direkt angeschlossen werden. Ist deren Spannungsversorgung jedoch höher, müssen Sie vor dem Anschluss prüfen, ob sie Signale mit 5V-Pegeln abgeben. Ggf. müssen Sie eine Anpass-Schaltung vorschalten.
Speziell HP liefert einige Encoder mit Open-Collector-Ausgängen. Auch diese Encoder können direkt angeschlossen werden. Auf dem Board befinden sich Pull-Up-Widerstände.
Oftmals werden Encoder geliefert, die mit einer Null-Spur ausgestattet sind. Die Nullspur muss nur dann angeschlossen werden wenn die Funktionen *cal 2, cal 3, cal 4* oder *cal 5* (siehe dort) verwendet werden sollen. Die Nullspur (auch Index genannt) wird sonst nirgendwo benutzt. Die Positionsinformation wird ausschließlich aus den Spuren A und B gewonnen und niemals mit der Nullspur synchronisiert. Der Anschluss der Nullspur kann also unter den genannten Umständen entfallen.

Welche Einstellungen werden dauerhaft gespeichert ?

Wie aus dem Vorangegangenen ersichtlich, werden verschiedene Einstellungen dauerhaft im internen EEPROM gespeichert und nach einem Power-On wiederhergestellt.
Hier nun eine Liste dieser Einstellungen:

Bezeichnung	Befehl zum Setzen/Auslesen
PID-Parameter P	kp/qp
PID-Parameter I	ki/qi
PID-Parameter D	kd/qd
Positionierfehler-Fenster Grösse	sipw/ripw
Positionierfehler-Fenster Zeit	sipt/ript
Kalibrierdrehzahl	scv/rcv
Kalibrierbeschleunigung	sca/rca
Motorstrombegrenzung	scl/rcl
Systemkonfiguration	ssyscon/rsyscon
Drehzahl bzw. Verfahrgeschwindigkeit	sv/rv
Beschleunigung	sa/ra

Anmerkungen zum Positionierfehler-Fenster

Sobald der Positioniermodus(*pm*) eingeschaltet wird, wird die Motor-Istposition überwacht. Das Bit *inpos* im Status (rss) spiegelt diesen Vorgang wieder.
Zunächst werden intern zwei Grenzwertpositionen gebildet:
1) Sollposition – Wert von *sipw* und
2) Sollposition + Wert von *sipw*.

Nun wird die Motor-Istposition mit diesen beiden Grenzwerten verglichen. Liegt die Motor-Istposition unter Grenzwert 1 oder über Grenzwert 2 bleibt das Flag *inpos* auf Null.
Liegt die Motor-Istposition irgendwo zwischen Grenzwert 1 und Grenzwert 2 dann wird ein Zeitzähler gestartet. Dieser Vergleich findet nun in Intervallen von 841.5µs fortlaufend statt.
Ist der Vergleich positiv (Istpos > Grenzwert 1 < Grenzwert2) dann wird der Zeitzähler um 1 erhöht.
Ist der Vergleich negativ, wird der Zeitzähler auf Null gesetzt.
Erreicht der Zeitzähler den Wert, der mittels *sipt* vorgegeben wurde, wird das Flag *inpos* auf High gesetzt.
Die Gründe dieser Vorgehensweise sind schnell beschrieben: Am Ende einer Bewegung kann es durch schlecht eingestellte PID-Parameter zu einem Schwingen des Motors kommen. In diesem Fall würde der Motor über die gewünschte Sollposition hinausfahren, dann die Drehrichtung umkehren und unter die Sollposition drehen, usw. Damit der Anwender “sicher” sein kann, dass der Motor die gewünschte Sollposition erreicht hat, muss nach dem oben beschriebenen Verfahren der Motor für den Zeitraum von *sipt* innerhalb des Fensters sein das mit *sipw* vorgegeben wurde. Nehmen wir an, mit *sipw* wurde 5 übergeben und mit *sipt* der Wert 100, so muss sich der Motor für mindestens 84ms innerhalb des Fensters von +-5 Positionen um die Sollposition herum befunden haben wenn das Flag *inpos* gesetzt ist.

gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel

Tel.: 06101/523100 • Fax: 06101/523101

eMail: info@graebner-elektronik.de **• Internet** http://www.graebner-elektronik.eu

Kommunikationsprotokoll

Die Kommunikation mit einem übergeordneten Rechner findet über die RS232-Schnittstelle statt. Dazu sind lediglich die Signale TxD (Sendeleitung), RxD (Empfangsleitung) und natürlich GND (Masse) notwendig.
Da keinerlei Handshake-Leitungen benutzt werden, muss das im folgenden beschriebene Kommunikations-Protokoll unter allen Umständen eingehalten werden, soll es nicht zu Daten-/Zeichenverlusten kommen.
Zunächst: Alle Zeichen die über die Schnittstelle laufen sind ASCII-Zeichen. Numerische Werte die übermittelt werden sollen, müssen als dezimale Werte im ASCII-Format übermittelt werden. Der Wert 100 wird also als eine Serie von 3 Zeichen übermittelt: Zuerst eine “1” (ASCII 49) und dann zweimal “0” (ASCII 48).
Die beiden grundlegenden Routinen zur Kommunikation bestehen aus dem Senden eines einzelnen ASCII-Zeichens und dem Empfang eines einzelnen Zeichens über die von Ihnen gewählte RS232-Schnittstelle Ihres Rechners.
Die Routine zum Empfang eines Zeichens sollte in jedem Fall über einen Timeout verfügen. Wird innerhalb einer bestimmten Zeit (<200ms) kein Zeichen empfangen, sollte diese Routine mit einer Fehlermeldung abbrechen. Sinnvoll ist ein Rückgabewert von –1, denn alle regulär empfangenen ASCII-Zeichen liegen im Bereich von 0 bis 127.
Testen Sie diese beiden Routinen ausgiebig, sie stellen die Basis für die Kommunikation dar und sind damit elementar.
Damit bei den nachfolgenden Erläuterungen klar ist, welche Routine gemeint ist, vergebe ich für die Zeichen-Sende-Routine den Namen “schar” und für die Zeichen-Empfangs-Routine den Namen “rchar”.
Als nächstes ist eine Routine zu schreiben, die die empfangenen Zeichen analysiert und ggf. in einem String sammelt: Diese Routine sollte alle Zeichen > ASCII 31 in den String schreiben und sich beenden, wenn ein ASCII 13 (CR) empfangen wurde. Diese Routine erhält von mir den Namen “rstring”.
Die nun zu schreibende vierte Routine erhält den Namen “sstring” und soll einen Übergabestring auf besondere Art über die RS232 senden. Prinzipiell muss diese Routine die Zeichen des Strings einzeln senden und, nachdem ein Zeichen gesendet wurde, dessen Echo abwarten das die SLAP2 erzeugt. Also: In einer Schleife wird zunächst das erste Zeichen isoliert und “schar” übergeben. Danach wird “rchar” aufgerufen und das Echo des Zeichens eingesammelt (und wenn gewünscht, mit dem gesendeten Zeichen verglichen). Danach wird das zweite Zeichen mit “schar” gesendet und dessen Echo geholt usw. Ist das letzte Zeichen gesendet und dessen Echo abgeholt, muss ein ASCII 13 (CR) gesendet werden. Auch dessen Echo muss mit “rchar” abgeholt werden.

Die fünfte Routine bekommt den Namen “sbefehl”. An sie wird ein beliebiger Befehlsstring übergeben und liefert die von der SLAP2 erzeugte Antwort zurück. “sbefehl” übergibt ihrerseits also den Befehlsstring an “sstring” und ruft anschließend sofort “rstring” auf. Der von “rstring” zurückgegebene String wird ohne Bearbeitung von “sbefehl” an das aufrufende Programm zurückgegeben.
Das war’s. Wenn Sie sich akribisch an diese Beschreibung halten, wird die Kommunikation mit der Baugruppe praktisch auf Anhieb funktionieren und zwar unter allen praktischen Umständen.

Zum weiteren Verständnis nun Folgendes: Jeder an die SLAP2 übergebene Befehl erzeugt einen Antwortstring. Der kann aus einer Zeichenfolge plus CR bestehen oder nur aus CR (ASCII 13 (CR)).
1. Beispiel:
Es soll die aktuelle Motorposition ausgelesen werden. Der Befehl lautet *rp* ohne weiteren Parameter. Also wird “sbefehl” aufgerufen, im Übergabestring steht “rp”.
“sbefehl” übergibt den String mit dem Inhalt “rp” an “sstring”. “sstring” sendet das erste Zeichen, holt dessen Echo, sendet das zweite Zeichen und holt dessen Echo. Nun wird ein ASCII 13 (CR) gesendet und dessen Echo geholt. “sstring” ruft nun “rstring” auf. “rstring” holt alle Zeichen von der seriellen Schnittstelle und packt sie in den Rückgabestring sofern sie > ASCII 31 sind. Empfängt “rstring” ein ASCII 13 (CR) übergibt “rstring” die empfangenen Zeichen an das aufrufende “sbefehl”. “sbefehl”

liefert diesen String an Ihr Hauptprogramm in dem Sie die empfangenen Zeichen in einen numerischen Wert wandeln und auswerten können.

2. Beispiel:
Es soll der Positioniermodus eingeschaltet werden. Der Befehl lautet *pm* ohne weitere Parameter. "sbefehl" übergibt den String mit dem Inhalt "pm" an "sstring". "sstring" sendet die Zeichen einzeln, holt jeweils deren Echo, sendet nach dem zweiten, letzten Zeichen ein ASCII 13 (CR) und holt dessen Echo. "sstring" ruft nun "rstring" auf. "rstring" holt das erste Zeichen von der seriellen Schnittstelle und würde es in den Rückgabestring packen wenn es grösser als ASCII 31 wäre. Das ist es aber nicht denn dieses Zeichen ist ein ASCII 13 (CR). "rchar" wird ein ASCII 13 (CR) zurückliefern. Damit ist die Endebedingung für "rstring" erreicht. An "sbefehl" wird der Antwortstring auf den Befehl *pm* zurückgeliefert der eine Länge von 0 Zeichen hat – also ein Leerstring. Dieser Leerstring wird natürlich auch an Ihr Hauptprogramm geliefert. Da keine Antwort von *pm* zu erwarten war, kann damit die Auswertung des Antwortstrings unterbleiben.

3. Beispiel:
Es soll der Motor im eingeschalteten Positioniermodus auf Position 1234 verfahren werden. Der Befehl lautet *ma1234*.
"sbefehl" übergibt den String mit dem Inhalt "ma1234" an "sstring". "sstring" sendet die Zeichen einzeln, holt jeweils deren Echo, sendet nach dem letzten Zeichen ein ASCII 13 (CR) und holt dessen Echo. "sstring" ruft nun "rstring" auf. "rstring" holt das erste Zeichen von der seriellen Schnittstelle und würde es in den Rückgabestring packen wenn es grösser als ASCII 31 wäre. Das ist es aber nicht, denn dieses Zeichen ist ein ASCII 13 (CR). "rchar" wird ein ASCII 13 (CR) zurückliefern. Damit ist die Endebedingung für "rstring" erreicht. An "sbefehl" wird der Antwortstring auf den Befehl *ma1234* zurückgeliefert der eine Länge von 0 Zeichen hat – also ein Leerstring. Dieser Leerstring wird natürlich auch an Ihr Hauptprogramm geliefert. Da keine Antwort von *ma1234* zu erwarten war, kann damit die Auswertung des Antwortstrings unterbleiben.

Dieses Protokoll bleibt auch gültig wenn mehrere SLAP2 an einer seriellen Schnittstelle betrieben werden:
Angenommen die SLAP2 mit der Adresse 2 ist augenblicklich selektiert. Nun soll die Karte mit der Adresse 1 selektiert werden. Der Befehl dazu lautet *se1*. Folgendes geschieht intern in den SLAP2-Modulen: Die drei Zeichen werden zusammen mit dem abschließenden ASCII 13 (CR) empfangen (und geechot). SLAP2 mit Adresse 2 analysiert den String und ermittelt den übergebenen Wert (die Adresse) und stellt fest, dass dieser Wert nicht mit ihrer eigenen, internen Adresse übereinstimmt. Daraufhin wird die SLAP2, Adresse 2 sofort ihren Sendekanal abschalten.
SLAP1, Adresse 1 hat diesen String auch empfangen und festgestellt, dass der Wert mit ihrer internen Adresse übereinstimmt. Sie wird nun sofort ihren Sendekanal einschalten und anschließend ein ASCII 13 (CR). Somit ist auch in diesem Fall dem Protokoll genüge getan.

Wenn Sie das Ganze einmal überdenken, werden Sie erkennen, dass das Protokoll ein hohes Maß an Sicherheit bietet und etwaige Fehler leicht erkannt und behandelt werden können. Wenn mittels *se* eine Karte adressiert wird, die es gar nicht gibt, wird der Antwortstring aus CR ausbleiben. Wenn Sie das Timeout in "rchar" richtig implementiert haben und die darüber liegenden Routinen den Fehlercode (mein Vorschlag war –1) korrekt weitergeben, kann Ihr Hauptprogramm entsprechend reagieren. Gleiches gilt logischerweise auch, wenn eine der RS232-Leitungen gebrochen ist – dann bleibt das Echo auf das erste gesendete Zeichen eines Befehls vollständig aus.

In diesem Zusammenhang noch eine wichtige Anmerkung: Die Baugruppe mit der Adresse 0 meldet sich automatisch nach Power-On ! Damit stehen Zeichen in dem Puffer Ihres Rechners, die nicht mittels eines Befehls angefordert wurden. Das bringt das Ganze natürlich durcheinander. Abhilfe zu schaffen ist aber recht einfach: Rufen Sie "rchar" sooft auf, bis "rchar" den Timeout-Fehlercode zurückliefert. Dann ist der interne UART-Puffer Ihres Rechners garantiert leer.

Nebenbei sei noch angemerkt, das Leerzeichen (ASCII 32) in den SLAP2 keinerlei Bedeutung haben und beim Empfang bereits ausgefiltert werden. Es spielt also keine Rolle, ob Sie z.B.

Am Römerbrunnen 11a • 61118 Bad Vilbel
Tel.: 06101/523100 • Fax: 06101/523101
eMail: info@graebner-elektronik.de • **Internet** http://www.graebner-elektronik.eu

"sv 1000" oder "sv1000" senden.

PID-Parameter

Die Parameter P, I und D haben den gleichen Wertumfang: 0..32767. Nur positive Zahlen sind erlaubt.
Die Grundeinstellung ist:
P=40, I=40, D=80
Das ist richtig für einen unbelasteten Motor Typ Faulhaber 2224 mit 512er Encoder.
Die Steifigkeit des Motors wird mit P verändert.
Die Schwingneigung wird mit D unterdrückt.
I verschafft Positioniergenauigkeit am Zielort.
Parameter vorsichtig ändern, all zu große Wertänderungen sind meist nicht sinnvoll, besser ist es, sich langsam an das "Optimum" heranzutasten. Folgende Vorgehensweise kann empfohlen werden: I-Anteil abschalten bzw. stark verringern ("KI0" oder "KI1"). Nun ist die Positioniergenauigkeit sehr schlecht, aber der D- und P-Anteil wird nicht mehr vom I-Anteil beeinflusst. D-Anteil und P-Anteil erhöhen, bis die gewünschte oder machbare Steifigkeit erreicht ist. Um dem Schwingen vorzubeugen ggf. D-Anteil variieren. Ist man mit P und D zufrieden, wird der I-Anteil solange erhöht bis auch die Positioniergenauigkeit stimmt.
Um Veränderungen des Regelverhaltens feststellen zu können muss zwischen den einzelnen Schritten der Parameterermittlung der Motor immer wieder verfahren werden, nur dann bemerkt man die vorgenommenen Veränderungen.
Dieser Vorgang ist nicht sonderlich schwierig, ich empfehle aber, sich einen sehr ruhigen Raum zu suchen (damit man den Motor hört), sich 30 Minuten Zeit zu nehmen. Danach hat man ein Gefühl für den Motor und die Regelung und kennt die Grenzen von Beiden. Weitere Nachbesserungen gehen dann leicht von der Hand. Vorteilhaft ist auch, den Motor in die Hand zu nehmen. Dabei spürt man geringste Schwingungen, denn, egal was man will, der Motor muss ruhig und gleichmäßig laufen, das ist oberste Maxime. Übrigens gibt es Applikationen bei denen es besser ist, die Steifigkeit des Systems zu verringern. Das wirkt sich vorteilhaft auf die Geschwindigkeitsregelung aus, wenn mehr Gleichmäßigkeit als Drehzahlgenauigkeit gefragt ist. Im Positioniermodus bringt das nur dann etwas, wenn sich das System ein wenig federnd verhalten soll. Man sollte jetzt aber nicht auf die Idee kommen mit den Motoren und der Regelung eine Feder nachzubilden. Der Regelung ist das egal, aber der Motor macht es nicht lange, wenn die Bürsten immer über ein und dieselbe Stelle kratzen.

Verdrahtungshinweise

Der angeschlossene Motor wird über eine PWM-Endstufe versorgt. Dabei werden z.T. erhebliche Ströme geschaltet die erhebliche Störungen verursachen können.
Aus diesem Grund müssen die Motorzuleitungen über ein von allen anderen Leitungen getrenntes und abgeschirmtes Kabel mit der SLAP2 verbunden werden.
Alle anderen Anschlüsse wie Winkelencoder und Endschalter können gemeinsam in einem Kabel geführt werden das ebenfalls geschirmt sein sollte.
Erfolgt die Verkabelung nicht nach diesen Empfehlungen ist ein ordnungsgemäßer Betrieb des Motors nicht zu erwarten.

Technische Daten SLAP2

Ab Hardwareversion V2211 / Software V2205

Datenübertragungsrate
19200Bd, 8Bit, 1 Stoppbit, Keine Parität

Pegel auf den Datenübertragungsleitungen
Es werden Pegel nach RS232 (ca. -+12V) erwartet bzw. generiert

Spannungsversorgung
12..30V Gleichspannung

gräbner-elektronik gmbh

Am Römerbrunnen 11a • 61118 Bad Vilbel
Tel.: 06101/523100 • Fax: 06101/523101
eMail: info@graebner-elektronik.de **• Internet** http://www.graebner-elektronik.eu

Stromaufnahme ohne Motor
<40mA bei 24V Versorgungsspannung

Strombegrenzung
Auslieferungszustand: ca. 2000mA

Eingänge Winkelenkoder
TTL-Pegel, Innenwiderstand ca. 3000 Ohm
Es werden zwei, um 90° phasenverschobene Rechtecke erwartet (Quadraturenkoder)

Maximal zulässige Ausgangsfrequenz des Winkelenkoders
0,5MHz

Flankensteilheit der Enkodersignale
Kleiner 500ns