LinuxCNC

Spindel synchronisieren + orientieren

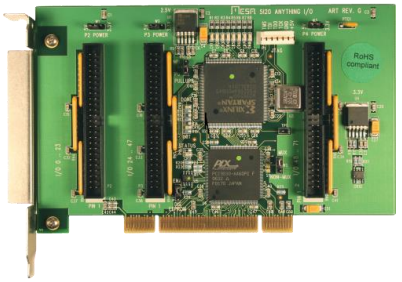
Meine verwendeten Komponenten.

Servodrive

Estun EDB-10A + Estun Servomotor 1KW 1000 U/min

Mesa Karten

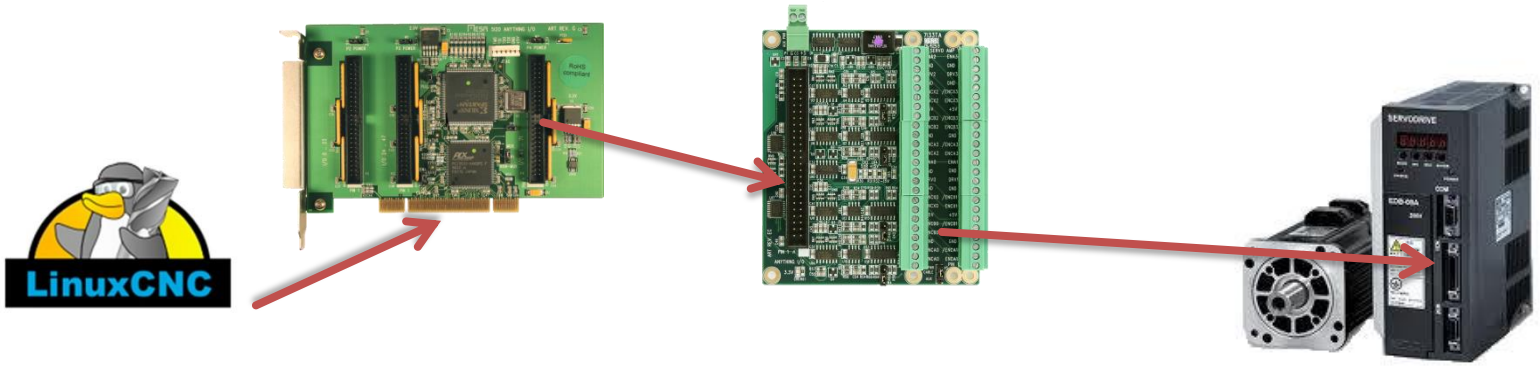
5i20 + 7i33



Theorie - Konfiguration der Spindel

- Drehzahlvorgabe (per Analogwert)
- Drehzahlrampen (Beschleunigung und Bremsen der Spindel einstellbar)
- Drehzahlvergleich (Achsbewegung beginnt erst wenn die eingestellte Drehzahl erreicht ist)
- Synchronisation (damit diversen G-Befehle (Gewinden) funktionieren)
- Positionierung (über M19 die Spindelposition einstellen)

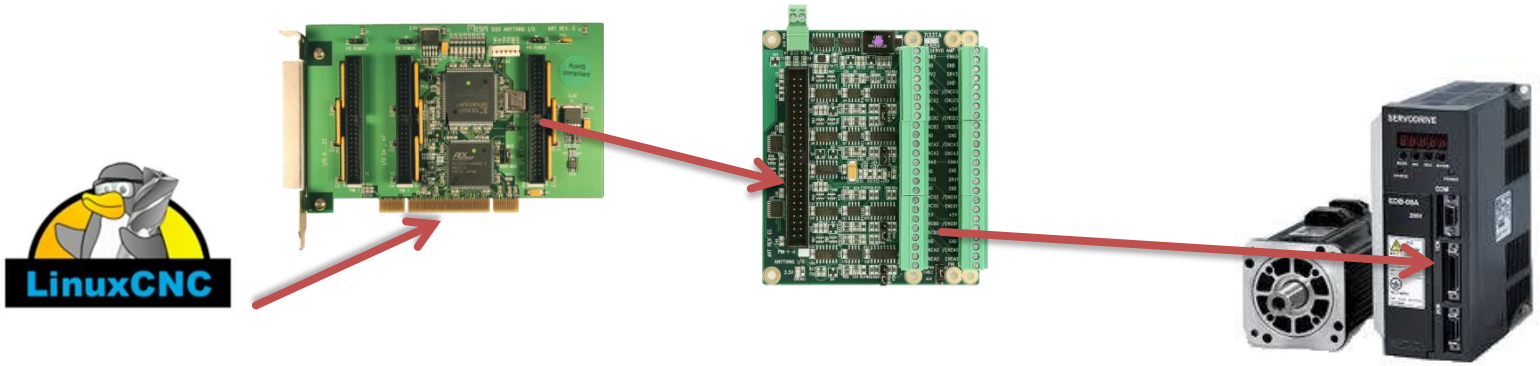
Aber wie



Kommunikation

Die 5i20 steckt im PCI Steckplatz vom PC und kommuniziert über Hostmot2 mit LinuxCNC.
Beim Start von LinuxCNC wird über eine Zeile in der HAL das jeweilige Bitfile in die Karte geladen.
Das Bitfile entscheidet darüber welche Funktionen an den einzelnen Pins zur Verfügung stehen.

Hier verwende ich folgende Zeile zur Initialisierung der Karte.
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT num_encoders=4 num_pwmgens=4 num_stepgens=0„



Kommunikation

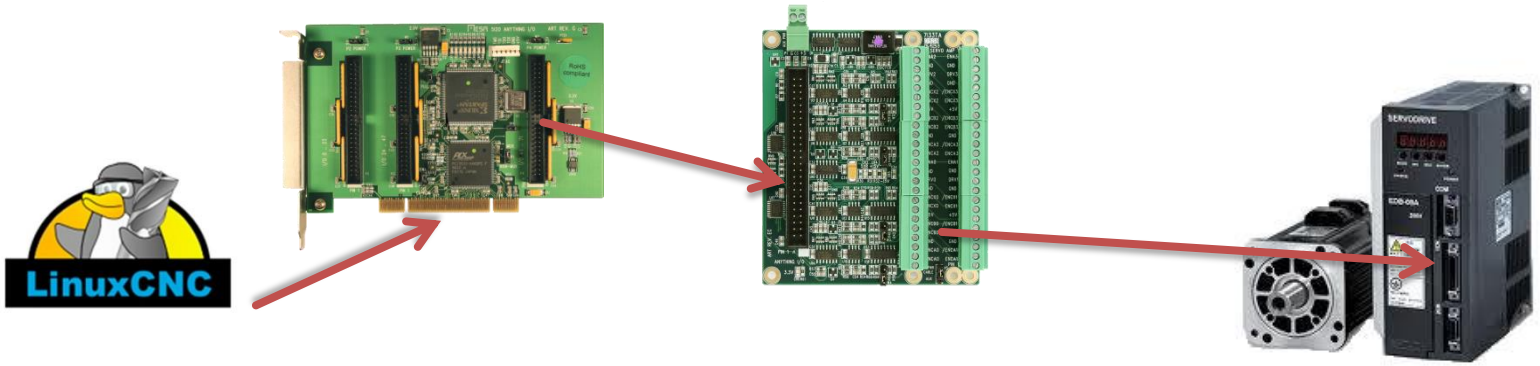
Die 5i20 steckt im PCI Steckplatz vom PC und kommuniziert über Hostmot2 mit LinuxCNC.
Beim Start von LinuxCNC wird über eine Zeile in der HAL das jeweilige Bitfile in die Karte geladen.
Das Bitfile entscheidet darüber welche Funktionen an den einzelnen Pins zur Verfügung stehen.

Hier verwende ich folgende Zeile zur Initialisierung der Karte.
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT num_encoders=4 num_pwmgens=4 num_stepgens=0„

PIN	IO#	Module	Chan	Func
P2-1	0	Encoder	1	Phase B (in)
P2-3	1	Encoder	1	Phase A (in)
P2-5	2	Encoder	0	Phase B (in)
P2-7	3	Encoder	0	Phase A (in)
P2-9	4	Encoder	1	Index (in)
P2-11	5	Encoder	0	Index (in)
P2-13	6	PWMGen	1	PWM/Up (out)
P2-15	7	PWMGen	0	PWM/Up (out)
P2-17	8	PWMGen	1	Dir/Down (out)
P2-19	9	PWMGen	0	Dir/Down (out)
P2-21	10	PWMGen	1	Enable (out)
P2-23	11	PWMGen	0	Enable (out)
P2-25	12	Encoder	3	Phase B (in)
P2-27	13	Encoder	3	Phase A (in)
P2-29	14	Encoder	2	Phase B (in)
P2-31	15	Encoder	2	Phase A (in)
P2-33	16	Encoder	3	Index (in)
P2-35	17	Encoder	2	Index (in)
P2-37	18	PWMGen	3	PWM/Up (out)
P2-39	19	PWMGen	2	PWM/Up (out)
P2-41	20	PWMGen	3	Dir/Down (out)
P2-43	21	PWMGen	2	Dir/Down (out)
P2-45	22	PWMGen	3	Enable (out)
P2-47	23	PWMGen	2	Enable (out)

PIN	IO#	Module	Chan	Func
P3-1	24	Encoder	5	Phase B (in)
P3-3	25	Encoder	5	Phase A (in)
P3-5	26	Encoder	4	Phase B (in)
P3-7	27	Encoder	4	Phase A (in)
P3-9	28	Encoder	5	Index (in)
P3-11	29	Encoder	4	Index (in)
P3-13	30	PWMGen	5	PWM/Up (out)
P3-15	31	PWMGen	4	PWM/Up (out)
P3-17	32	PWMGen	5	Dir/Down (out)
P3-19	33	PWMGen	4	Dir/Down (out)
P3-21	34	PWMGen	5	Enable (out)
P3-23	35	PWMGen	4	Enable (out)
P3-25	36	Encoder	7	Phase B (in)
P3-27	37	Encoder	7	Phase A (in)
P3-29	38	Encoder	6	Phase B (in)
P3-31	39	Encoder	6	Phase A (in)
P3-33	40	Encoder	7	Index (in)
P3-35	41	Encoder	6	Index (in)
P3-37	42	PWMGen	7	PWM/Up (out)
P3-39	43	PWMGen	6	PWM/Up (out)
P3-41	44	PWMGen	7	Dir/Down (out)
P3-43	45	PWMGen	6	Dir/Down (out)
P3-45	46	PWMGen	7	Enable (out)
P3-47	47	PWMGen	6	Enable (out)

PIN	IO#	Module	Chan	Func
P4-1	48	StepGen	0	Step (out)
P4-3	49	StepGen	0	Dir (out)
P4-5	50	StepGen	0	StepTable 2 (out)
P4-7	51	StepGen	0	StepTable 3 (out)
P4-9	52	StepGen	0	StepTable 4 (out)
P4-11	53	StepGen	0	StepTable 5 (out)
P4-13	54	StepGen	1	Step (out)
P4-15	55	StepGen	1	Dir (out)
P4-17	56	StepGen	1	StepTable 2 (out)
P4-19	57	StepGen	1	StepTable 3 (out)
P4-21	58	StepGen	1	StepTable 4 (out)
P4-23	59	StepGen	1	StepTable 5 (out)
P4-25	60	StepGen	2	Step (out)
P4-27	61	StepGen	2	Dir (out)
P4-29	62	StepGen	2	StepTable 2 (out)
P4-31	63	StepGen	2	StepTable 3 (out)
P4-33	64	StepGen	2	StepTable 4 (out)
P4-35	65	StepGen	2	StepTable 5 (out)
P4-37	66	StepGen	3	Step (out)
P4-39	67	StepGen	3	Dir (out)
P4-41	68	StepGen	3	StepTable 2 (out)
P4-43	69	StepGen	3	StepTable 3 (out)
P4-45	70	StepGen	3	StepTable 4 (out)
P4-47	71	StepGen	3	StepTable 5 (out)



Kommunikation

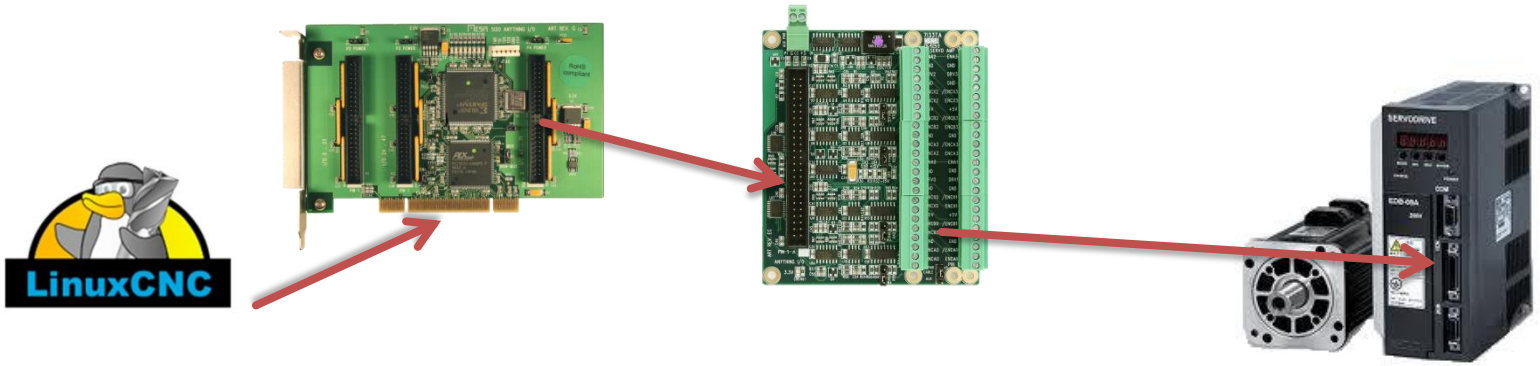
Die 5i20 steckt im PCI Steckplatz vom PC und kommuniziert über Hostmot2 mit LinuxCNC.
Beim Start von LinuxCNC wird über eine Zeile in der HAL das jeweilige Bitfile in die Karte geladen.
Das Bitfile entscheidet darüber welche Funktionen an den einzelnen Pins zur Verfügung stehen.

Hier verwende ich folgende Zeile zur Initialisierung der Karte.
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT num_encoders=4 num_pwmgens=4 num_stepgens=0„

loadrt trivkins
loadrt [EMCMOT]EMCMOT servo_period_nsec=[EMCMOT]SERVO_PERIOD num_joints=[TRAJ]AXES
loadrt hostmot2

loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT num_encoders=4 num_pwmgens=4 num_stepgens=0"

setp hm2_5i20.0.pwmgen.pwm_frequency 20000
setp hm2_5i20.0.pwmgen.pdm_frequency 6000000
setp hm2_5i20.0.watchdog.timeout_ns 5000000
addf hm2_5i20.0.read servo-thread
addf hm2_5i20.0.write servo-thread
addf motion-command-handler servo-thread
addf motion-controller servo-thread



Kommunikation

Die 5i20 steckt im PCI Steckplatz vom PC und kommuniziert über Hostmot2 mit LinuxCNC.
Beim Start von LinuxCNC wird über eine Zeile in der HAL das jeweilige Bitfile in die Karte geladen.
Das Bitfile entscheidet darüber welche Funktionen an den einzelnen Pins zur Verfügung stehen.

Hier verwende ich folgende Zeile zur Initialisierung der Karte.
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT num_encoders=4 num_pwmgens=4 num_stepgens=0„

Somit kann ich die 7i33 mit einem 50 pol Flachbandkabel an den P2 anschließen. P3 und P4 nutze ich hier erstmal nicht.

Am P2 der 5i20 stehen wie rechts abgebildet nun Folgende Funktionen zur Verfügung. 4 x PWM Ausgang / 4 x Dir Ausgang 4 x Enable Ausgang
4 x Encoder Eingang

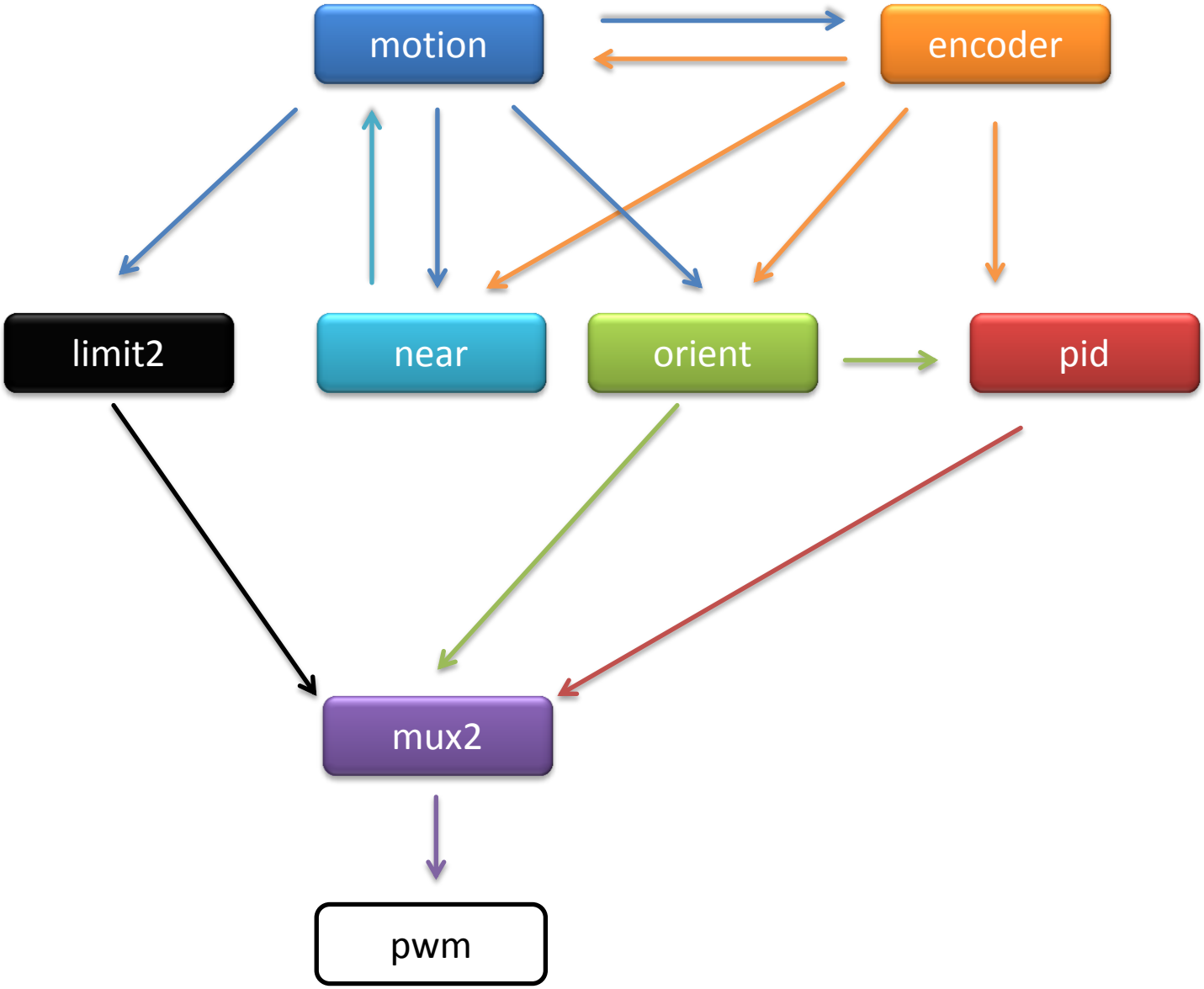
Die Pin Belegung ist kompatibel mit der 7i33 es muss also nichts im Bitfile angepasst werden.

Das PWM und DIR wird von der 7i33 in ein Analogsignal
±10V umgewandelt. Somit lässt sich mein Regler über Speed ansteuern. Ich habe DRV0 von der 7i33 mit dem Analogeingang (1CN) des Reglers verbunden.

Damit LinuxCNC nun auch den Encoder vom Motor sieht habe ich die Encoder Signale A/B/Z vom Regler Anschluss 1CN an die 7i33 (ENC0A / ENC0B und ENC0X)angeschlossen.

In der HAL muss ich dann die PINS vom hm2_5i20.0.pwmgen.00. und hm2_5i20.0.encoder.00. verwenden.

Übersicht



HAL - Konfiguration der Spindel

Als erstes erstelle ich in der INI eine Sektor [SPINDLE_9] und einige Variablen die dann von der HAL gelesen werden.
Damit kann man später die Spindel einfacher parametrieren.

```
#********************
# Spindle
#********************
[SPINDLE_9]

P = 5000
I = 0
D = 0
FF0 = 0
FF1 = 0
FF2 = 0
BIAS = 0
DEADBAND = 0
MAX_OUTPUT = 100

ENCODER_SCALE = 10000        (Mein Encoder liefert 10000 Impulse pro Umdrehung. Volle Auflösung hier eintragen)
OUTPUT_SCALE  = 1500         (bei ±10V die maximale Drehzahl)
ACCELERATION  = 1000         (mit welcher Beschleunigung und Bremswirkung soll gefahren werden)
MAX_ERROR     = 0.2          (akzeptierter Fehler zwischen Sollwert und Istwert angegeben in U/Sekunde)
OFF_DELAY     = 1.5          (Abschaltverzögerung in Sekunden für die Reglerfreigabe)
```

HAL - Konfiguration der Spindel

```
#  SPINDLE

setp    hm2_5i20.0.encoder.00.counter-mode          0
setp    hm2_5i20.0.encoder.00.filter                1
setp    hm2_5i20.0.encoder.00.index-invert          0
setp    hm2_5i20.0.encoder.00.index-mask            0
setp    hm2_5i20.0.encoder.00.index-mask-invert     0
setp    hm2_5i20.0.encoder.00.scale                 [SPINDLE_9]ENCODER_SCALE        wert aus INI übernehmen
setp    hm2_5i20.0.pwmgen.00.output-type            1
setp    hm2_5i20.0.pwmgen.00.scale                  [SPINDLE_9]OUTPUT_SCALE         wert aus INI übernehmen

loadrt limit2       names=spindle-ramp                          einmal limit2 mit Namen weiter verwenden
loadrt near         names=spindle-at-speed,spindle-at-pos       zweimal near mit Namen weiter verwenden
loadrt timedelay    names=spindle-active-delay                  einmal limit2 mit Namen weiter verwenden
loadrt orient       names=spindle-orient                        einmal limit2 mit Namen weiter verwenden
loadrt pid          names=spindle-pid                           einmal limit2 mit Namen weiter verwenden
loadrt mux2         names=spindle-pwm-switch                    einmal limit2 mit Namen weiter verwenden
loadrt or2          count=0                                     einmal or2 numerisch weiter verwenden
loadrt not          count=0                                     einmal not numerisch weiter verwenden
loadrt and2         count=0                                     einmal and1 numerisch weiter verwenden
loadrt offset       count=0                                     einmal offset numerisch weiter verwenden

addf spindle-ramp                   servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf spindle-at-speed               servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf spindle-at-pos                 servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf spindle-active-delay           servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf spindle-orient                 servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf spindle-pid.do-pid-calcs       servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf spindle-pwm-switch             servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf or2.0                          servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf not.0                          servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf and2.0                         servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
addf offset.0.update-output         servo-thread        Komponente in den servo-thread laden damit diese berechnet werden.
```

HAL - Konfiguration der Spindel

```
setp   spindle-pid.Pgain                    [SPINDLE_9]P
setp   spindle-pid.Igain                    [SPINDLE_9]I
setp   spindle-pid.Dgain                    [SPINDLE_9]D
setp   spindle-pid.bias                     [SPINDLE_9]BIAS
setp   spindle-pid.FF0                      [SPINDLE_9]FF0
setp   spindle-pid.FF1                      [SPINDLE_9]FF1
setp   spindle-pid.FF2                      [SPINDLE_9]FF2
setp   spindle-pid.deadband                 [SPINDLE_9]DEADBAND
setp   spindle-pid.maxoutput                [SPINDLE_9]MAX_OUTPUT
setp   spindle-pid.error-previous-target true

setp   spindle-ramp.maxv                    [SPINDLE_9]ACCELERATION
setp   spindle-at-speed.difference          [SPINDLE_9]MAX_ERROR
setp   spindle-at-pos.difference            0.01
setp   spindle-at-pos.in1                   0
setp   spindle-active-delay.on-delay        0
setp   spindle-active-delay.off-delay       [SPINDLE_9]OFF_DELAY
```

HAL - Konfiguration der Spindel

```
# orient mit motion verknuepfen
net orient-angle        motion.spindle-orient-angle    => spindle-orient.angle
net orient-mode         motion.spindle-orient-mode   => spindle-orient.mode
net orient-enable       motion.spindle-orient           => and2.0.in1

# Position vom Encoder in den pid / orient und motion schieben
net spindle-pos          => spindle-pid.feedback      => spindle-orient.position
net spindle-pos          <= motion.spindle-revs       <= hm2_5i20.0.encoder.00.position

# Encodergeschwindigkeit U/sek in den near und motion schieben
net spindle-fb-rps       motion.spindle-speed-in     <= hm2_5i20.0.encoder.00.velocity   => spindle-at-speed.in2

# Positionsvorgabe vom orient in den pid schieben
setp offset.0.offset 1
net spindle.otient-cmd   spindle-orient.command      => offset.0.in
net spindle-orient-cmd1  offset.0.out                => spindle-pid.command

# Drehzahlvorgabe U/min aus motion in das limit2 schieben
net spindle-speed-rpm   motion.spindle-speed-out    => spindle-ramp.in

# Drehzahlvorgabe U/sek aus motion in das near schieben
net spindle-speed-rps   motion.spindle-speed-out-rps => spindle-at-speed.in1

# Wenn Solldrehzahl gleich Istdrehzahl dann aus near das bit in motion schieben
net spindle-at-speed    spindle-at-speed.out        => motion.spindle-at-speed

# Wenn Sollpositon gleich Istpositon dann aus near das bit in motion schieben
net spindle-pos-err     spindle-at-pos.in2          <= spindle-pid.error
net spindle-at-pos      spindle-at-pos.out          =>                                  Signal ist true wenn in Position zum Beispiel interessant beim WZW

net spindle-on0         motion.spindle-on           => spindle-active-delay.in
net spindel-on1         spindle-active-delay.out    => or2.0.in0   => not.0.in
net spindel-on2         and2.0.in0                  <= not.0.out

# orient und pid aktivieren
net orient-active       and2.0.out => or2.0.in1 => spindle-pwm-switch.sel
net orient-active       => spindle-pid.enable    => spindle-orient.enable
net spindle-enable      or2.0.out                => hm2_5i20.0.pwmgen.00.enable

# motion gibt Signal an Encoder das beim naechsten Z Signal auf 0 gestellt werden soll
net  spindle-sync       motion.spindle-index-enable  => hm2_5i20.0.encoder.00.index-enable
sets spindle-sync       1

# pwm Signal entweder vom motion oder orient
net pwm-switch-in0   spindle-pwm-switch.in0   <= spindle-ramp.out
net pwm-switch-in1   spindle-pwm-switch.in1   <= spindle-pid.output
net pwm-switch-out   spindle-pwm-switch.out   => hm2_5i20.0.pwmgen.00.value
```

LinuxCNC

Spindel Konfiguration

http://talla83.de/linuxcnc/config.htm
http://linuxcnc.org/docs/2.7/html/man/man9/hostmot2.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/near.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/limit2.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/timedelay.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/orient.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/pid.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/offset.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/or2.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/and2.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/mux2.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/not.9.html
http://linuxcnc.org/docs/2.7/html/man/man9/motion.9.html