BZ
Software-Version 3.01
Kurzübersicht SPS

Stand 06.05.2010

Inhalt

Einleitung

Die Empfänger-Option „Programmierbare Ablaufsteuerung / SPS“ ermöglicht es eigene Funktionen zu realisieren welche in der Standardausstattung nicht vorhanden sind.
SPS ist in der Industrie das Kürzel für „speicherprogrammierte Steuerung“. Damit lassen sich Verknüpfungen zwischen den Anschlüssen des Empfängers und einfache Abläufe per Software definieren. Im Gegensatz zu einer festverdrahteten Steuerung sind die Eigenschaften der SPS schnell und sogar während des Betriebs veränderbar.
Die SPS der Blauzahn kann sowohl digitale Schaltzustände (an/aus) als auch Proportionalwerte (-100% bis +100%) verarbeiten.
Die Programmierung erfolgt grundsätzlich über den Sender in der Programmiersprache „AWL“. Das steht für „Anweisungsliste“.
Die Option SPS steht für Empfänger ab Version 3.01 zur Verfügung.
Zum Einrichten / Programmieren ist im Sender ebenfalls eine Version ab 3.01 notwendig.

Konzept

Die SPS-Funktion ist als weiterer Modus der Funktionsbausteine integriert. Bei Empfängern, die mit der Option SPS ausgestattet sind, erscheinen die entsprechenden Modi automatisch in der Auswahlliste.
Das Verhalten eines Funktionsbausteins im Modus SPS unterscheidet sich grundlegend vom normalen Funktionsbaustein: Statt Servowegen, Totbereich, Expo usw. lassen sich für jeden Funktionsbaustein insgesamt 10 AWL-Befehlsschritte angeben.
Vor allem hat ein SPS-Funktionsbaustein keinen „Gesteuert von“ Eintrag. Die Eingangsdaten können stattdessen per AWL-Befehl abgeholt werden. Dies hat den Vorteil der höheren Flexibilität: so können nicht nur alle Senderbedienelemente und Funktionsbausteine, sondern auch die Zustände aller Empfängeranschlüsse abgefragt werden.
Die Funktion einer SPS kann man sich als endlose Programmschleife vorstellen, sie wird pro Impulsrahmen (ca. alle 20 ms) einmal durchlaufen, immer beginnend mit dem ersten Befehl.
Die Blauzahn-SPS hat zwei Arbeitsregister, Reg0 und Reg1. Deren Zustand wird am Ende des Programms in den beiden Ausgängen des Funktionsbausteins abgelegt und steht beim nächsten Durchlauf wieder zur Verfügung. Beim Einschalten des Empfängers werden die Register mit denjenigen Werten initialisiert welche sie beim letzten Speichern hatten (Analog zur Programmierung der Startstellung bei Hydraulikmodus und Memory-Schaltern).
Grundsätzlich arbeitet jeder Funktionsbaustein, der im Modus „Mini-SPS“ läuft, eigenständig. Pro Empfänger können also 5 unabhängig voneinander ablaufende SPS mit jeweils 2 Registern und 10 Programmschritten eingerichtet werden.
Werden mehr Programmschritte in einer SPS benötigt so können mehrere Funktionsbausteine Verkettet werden: der erste läuft im Modus „Mini-SPS“, der direkt darauffolgende im Modus „Mini-SPS-Erweiterung“. Dann übernimmt dieser den Zustand des vorhergehenden FB (auch die Arbeitsregister), das Programm läuft sozusagen weiter und bietet Platz für 20 Schritte.
Werden alle fünf Funktionsbausteine auf diese Weise verkettet sind maximal 50 Schritte pro Empfänger programmierbar.

A. Register Auswählen

In der Befehlsliste schaltet kurzes Drücken auf „ENTER“ zwischen den beiden Arbeitsregistern um. Das gewählte Arbeitsregister wird vor jedem Befehl angezeigt. Zum Speichern des Befehls muss lange auf „ENTER“ gedrückt werden. Lassen Sie sich bitte nicht davon Irritieren dass dabei zunächst das angezeigte Arbeitsregister wechselt, diese Änderung betrifft nur kurzzeitig das Display im Sender. Gespeichert wir der Befehl mit dem richtigen Register.

B. Befehls-Übersicht

"nop @" keine Aktion
"Ende @", Bearbeitung hier abbrechen

"= Reg0 a", Reg = Reg0
"= Reg1 a", Reg = Reg1

"= Ansch. p", Anschluss/Steuerwert übernehmen
"= # x", expliziten Wert setzen

" + Ansch. p", Anschluss/Steuerwert addieren
" + # x", expliziten Wert addieren
" - Ansch. p" Anschluss/Steuerwert subtrahieren
"=|Reg| a", bildet Betrag von Reg

" * Ansch. p", Multiplizieren mit Anschluss/Steuerwert (100% entspricht 1,0 => 100%*100%=100%)
" * # x", Multiplizieren mit explizitem Wert (-128..+128 => -100%..+100%)
" Scal # y", scalieren mit explizitem Wert (0..255 => 0%.. 200%)

"=Delta R a", gibt die Änderung (des Reg) seit letzem Durchlauf aus
"=|Delta R|a", gibt Betrag der Änderung (des Reg) seit letzem Durchlauf aus
"=Delta |R|a", gibt die Änderung des Betrag(des Reg) seit letzem Durchlauf aus

"= |Trigg.|a", Triggerfunktion positiv (Änderung von Reg führt zu vollausschlag +/+, konstanter Reg liefert -100
"= Trigg. a", Triggerfunktion. Macht kurze vollausschläge wenn sich Reg-Zustand ändert (ON/OFF)

" Invert. a", Invertieren (</>0 => ON/OFF)
" AND An. p", UND-Verknüpfen mit Anschluss/Steuerwert (ON/OFF)
" OR An. p", ODER-Verknüpfen mit Anschluss/Steuerwert (ON/OFF)
" XOR An. p", XODER-Verknüpfen mit Anschluss/Steuerwert (ON/OFF)

"= -Reg a", Vorzeichenumkehr analog

"Blink aus @", Blinker abstellen
"Failsafe @", Falisafe aktivieren

" else @", nächsten Befehl ausführen wenn vorangegangene Bedingung nicht erfüllt

"Teste pos P", Teste Anschluss p, ausführen wenn in pos. Richtung gesetzt
"Teste neg P", Teste Anschluss p, ausführen wenn in neg. Richtung gesetzt

" ==Ansch. p", mit Anschluss/Steuerwert vergleichen = (analog), wenn gesetzt nächsten Befehl ausführen

```
" == #   x",      SPS_EQAD,    //
" !=Ansch. p"    mit Anschluss/Steuerwert vergleichen != (analog), wenn nicht gesetzt nächsten Befehl ausführen
" != #   x",      SPS_NEAD,    //
"==bin An. p",   mit Anschluss/Steuerwert vergleichen =(ON/OFF) , wenn ungleich nächsten Befehl überspringen
"==bin  # x",    mit expliziten Wert vergleichen = (ON/OFF), wenn ungleich nächsten Befehl überspringen
"!=bin An. p",   mit Anschluss/Steuerwert vergleichen != (ON/OFF), wenn gleich nächsten Befehl überspringen
"!=bin  # x",    mit expliziten Wert vergleichen != (ON/OFF), wenn gleich nächsten Befehl überspringen

">= Ansch. p",   mit Anschluss/Steuerwert vergleichen >= (analog), wenn kleiner nächsten Befehl überspringen
" >= #   x",      mit expliziten Wert vergleichen >= (analog), wenn kleiner nächsten Befehl überspringen
"<= Ansch. p",   mit Anschluss/Steuerwert vergleichen <= (analog), wenn größer nächsten Befehl überspringen
" <= #   x",      mit expliziten Wert vergleichen <= (analog), wenn größer nächsten Befehl überspringen
">  Ansch. p",   mit Anschluss/Steuerwert vergleichen > (analog), wenn kleiner/gleich nächsten überspringen
" >  #   x",      mit expliziten Wert vergleichen > (analog), wenn kleiner/gleich nächsten Befehl überspringen
"<  Ansch. ",    mit Anschluss/Steuerwert vergleichen < (analog), wenn größer/gleich nächsten überspringen
" <  #   ",      mit expliziten Wert vergleichen < (analog), wenn größer/gleich nächsten Befehl überspringen

"Ebene/Ende",    bricht ab wenn layer nicht in Maske
" Ebene/0  ",    setzt ergebnis auf Neutral wenn layer nicht in Maske
```

"Ebene == ", nächsten Befehl ausführen wenn layer mit Maske übereinstimmt
"Ebene <> ", nächsten Befehl ausführen wenn layer nicht in Maske

"Nachlauf Ap", Nachlauf um vom Anschluss/Steuerwert vorgegebenen sekunden (0..100)
"Nachlauf ss", Nachlauf um i sekunden (0..255)
"Verzöger Ap", Verzögerung um vom Anschluss/Steuerwert vorgegebenen sekunden (0..100)
"Verzöger ss", Verzögerung um i sekunden (0..255)

"Zeit An.p", Zeitglied flankengetriggert p sekunden (0..255)
"Zeit ss", Zeitglied flankengetriggert i sekunden (0..255)

"Zeit_n.An.p", Zeitglied flankengetriggert nachtriggerbar p sekunden (0..255)

"Zeit_n. ss", Zeitglied flankengetriggert nachtriggerbar i sekunden (0..255)

"Zähler An.p", Zählt steigende Flanken im Reg (<10 -> >10), Zählerstand im Reg, bei überschreiten von parameter auf 0
"Zähler #u", Zählt , aber direkter Parameter

"Zähle An. p", Zählt steigende Flanken am Port p, (<10 -> >10), Zählerstand im Reg
"Zähl Set #U", setzt Zählerstand auf direkten wert
"Zähl Max #u", begrenzt Zählerstand, beim überschreiten zurück auf 0, Reg triggert einmal mit -127

"=Takt ss", Taktgeber p sekunden (0..255)
"=Takt An.p", Taktgeber i sekunden (0..255)
"=Interv ss", Intervallgeber i sekunden (0..255)
"=Interv A.p", Intervallgeber p sekunden (0..255)

"FF flank #i", Flipflop flankengetriggert von Reg (Schwellwert angeben)
"FF flank p", Flipflop flankengetriggert von Anschluss

"-> Ansch. p", setzt Anschluss auf wert von Reg (wirkt nur bei Memory)
"1-> Ansch.P", setzt Anschluss auf "ein" (Oder-Verknüpft mit Steuerfunktion)
"0-> Ansch.P", setzt Anschluss auf "aus" (Oder-Verknüpft mit Steuerfunktion)
"Invert.An.P", Invertiert wert von Anschluss (Oder-Verknüpft mit Steuerfunktion)

"->M_Ansch.p",setzt Anschluss auf wert von Reg bei großen Ausschlägen (wirkt nur bei Memory)
"1-> M_An. P", setzt Anschluss-Memory auf "ein" (wirkt nur bei Memory)
"0-> M_An. P", setzt Anschluss-Memory auf "aus" (wirkt nur bei Memory)
"Inv. M_An.P", Invertiert wert von Anschluss-Memory (wirkt nur bei Memory)

"-> Servo p", setzt Anschluss-Pulslänge auf Wert von Reg (wirkt nur bei Hydraulik)
"+ Servo p", addiert Register zu Anschluss-Pulslänge (wirkt nur bei Hydraulik)