

APOSS® WIN ACHSEN-POSITIONIER- UND SYNCHRONISIER SPRACHE

Hochsprachen-Programmiersprache für die moderne Antriebstechnik

APOSS ist die erste Wahl, wenn es um eine integrierte Entwicklungsoberfläche für die Antriebsprogrammierung und CAN-Vernetzung geht. APOSS verbindet Hochsprachenprogrammierung mit leistungsstarken Werkzeugen für die Optimierung von Antrieb und Anwendung.

Intuitiv und leistungsstark

- Optimiert für Antriebssteuerungen plus SPS Funktionalität und CAN-Netzwerk Unterstützung.
- Hochsprachenprogrammierung ähnlich strukturiertem Text, erweitert um integrierte Befehle für Antriebspositionierung und -synchronisation sowie CAN Kommunikation.
- Aufzeichnung und Visualisierung von Prozessdaten & Bewegungsprofilen.
- Umfangreiche Online-Hilfe.
- Industriepilot in Hunderten von Anwendungen und Tausenden von Anlagen.

Anwendungen

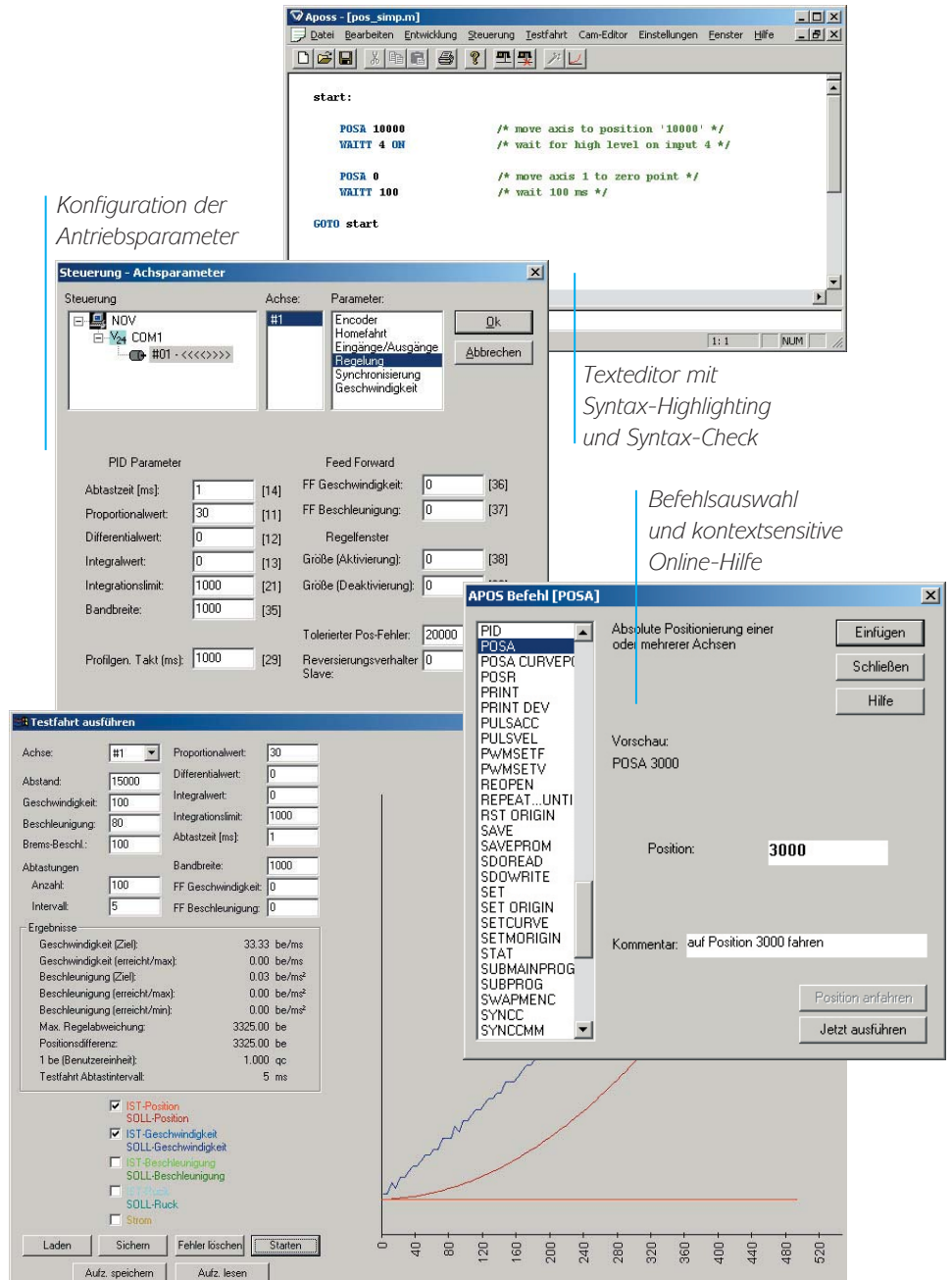
Steuerungen mit der APOSS Software werden zum Beispiel eingesetzt beim ...

- Positionieren von Komponenten
- Ausrichten von Werkzeugen
- Etikettieren von Waren
- Wickeln von Draht und Bändern
- Aufbringen von Kleber
- Beschicken mit Komponenten
- Heben von Waren
- Regeln des Pumpendurchfluss

Ihr Einsatzgebiet ist nicht enthalten?

Die zub machine control AG zeigt Ihnen gerne persönlich Ihren Nutzen durch den Einsatz von APOSS auf.

Konfiguration der Antriebsparameter



The screenshot displays the APOSS software interface with several windows open:

- Apos - [pos_simp.m]**: A text editor window showing G-code-like commands such as `POSA 10000`, `WAIT 4 0H`, `POSA 0`, and `WAIT 100`. It features syntax highlighting and a syntax check.
- Steuerung - Achsparameter**: A configuration window for axis parameters, including PID parameters (Abtastzeit, Proportionalwert, etc.) and Feed Forward settings.
- APOS Befehl [POSA]**: A command selection dialog box with a list of commands like `POSA CURVE`, `POSA`, `POSR`, etc., and a preview section showing the selected command's effect (e.g., moving to position 3000).
- Testfahrt ausführen**: A window for running a test run, showing parameters like Abstand, Geschwindigkeit, and Beschleunigung, along with a graph of the resulting motion profile.

Texteditor mit Syntax-Highlighting und Syntax-Check

Befehlsauswahl und kontextsensitive Online-Hilfe

Antriebs- & Prozess-Optimierung

Die Aufzeichnung und grafische Darstellung der Bewegungsdaten liefert die benötigten Grundlagen zur Antriebs- und Regleroptimierung.

Beliebige Prozessdaten können zudem zur Programmlaufzeit ausgelesen und aufgezeichnet werden.

Programmauswahl & -start

Die integrierte Programmverwaltung erlaubt es mehrere Programme in der Steuerung abzulegen. Ein Programm kann für den automatischen Start beim Einschalten konfiguriert werden. Alternativ ist auch die Programmanwahl über eine SPS oder über Schalter an den digitalen Eingängen möglich.

Antriebsynchronisation

Ein oder mehrere Antriebe können auf eine Leitachse synchronisiert werden, wie dies zum Beispiel bei elektronischen Königswellen notwendig ist.

Die APOSS Programmiersprache unterstützt verschiedene Synchronisationsvarianten, die auf den folgenden Leitsignalen beruhen:

- ◆ Position / Winkel
- ◆ Geschwindigkeit
- ◆ Master und Slave Marker

Die Kombination von Synchronisation und Markerkorrektur erlaubt dabei den Schlupf von Antrieben zu kompensieren.

Die Geschwindigkeitssynchronisation ermöglicht den Anlauf von Bändern (nach einer Störung) ohne dass versucht wird, die Position aufzuholen.

Konfigurierbare Geschwindigkeits- und Beschleunigungsbegrenzungen verhindern mechanische Beschädigungen oder eine Überbeanspruchung.

CAN-Kommunikation

Lokale oder dezentrale CANopen Ein- und Ausgänge können mit denselben IN- und OUT-Befehlen angesprochen werden. Die eventuell notwendige CAN-Kommunikation erfolgt automatisch im Hintergrund.

Die Steuerung von CANopen-Verstärkern und Frequenzumrichtern kann einfach über Parameter konfiguriert werden.

Die Istposition wird dann über den CAN gelesen und der Sollwert ebenfalls über den CAN statt über die ± 10 V Schnittstelle vorgegeben.

Mit jedem CANopen-Gerät können über SDOs und PDOs Daten ausgetauscht werden. Dabei sind weder Kenntnisse zur CAN-Kommunikation noch ist ein Protokoll notwendig. Der Aufruf der SDO READ- und SDO WRITE-Befehle und das Lesen und Schreiben der PDO Systemvariablen beinhaltet im Hintergrund automatisch den notwendigen Datenaustausch und die Fehlerbehandlung.

Kurvenscheiben-Steuerung

Die APOSS Entwicklungsoberfläche beinhaltet einen der modernsten Kurvenscheiben-Editoren für die Erstellung von Kurvenscheiben mit und ohne Markerkorrektur.

Die Kurvenpunkte können sowohl über die Werteingabe in eine Tabelle definiert werden als auch im grafischen Editor mit der Maus platziert und verschoben werden. Positions-, Geschwindigkeits-, Beschleunigungs- und Ruckkurven werden automatisch unter Berücksichtigung der definierten Antriebsparameter aktualisiert.

Zwischen den Kurvenpunkten wird die Kurve als Spline-Interpolation berechnet. Tangentenpunkte erlauben die problemlose Definition von Bereichen, in denen die Geschwindigkeit konstant und die Beschleunigung 0 sein muss.

Ergänzend besteht die Möglichkeit parallel zu allen Synchronisations- und Positionierbefehlen Interrupts zu definieren, die an die Slave-, Master- oder Kurvenscheibenposition gebunden sind und so die Funktion eines Nockenschaltwerks ermöglichen.

```

APOSS - [sync_simp.m *]
Datei Bearbeiten Entwicklung Steuerung Testfahrt Cam-Editor Einstellungen Fenster Hilfe

SYNCP // start synchronization

start:

IF (IN 1) THEN // switch on constant?

CVEL 50 // constant velocity 50 %
CSTART // start constant run
WHILE (IN 1) DO
ENDWHILE
SYNCP

ENDIF

GOTO start

Syntaxprüfung ... ok (71 bytes)
Für Hilfe F1 drücken
    
```

```

APOSS - [can_simp.m]
Datei Bearbeiten Entwicklung Steuerung Testfahrt Cam-Editor Einstellungen Fenster Hilfe

/* Initialisieren der CAN-Module mit voreingestellter Teil

CANINI 1,2,3,4 // Modul 1 bis 4 werden angemeldet

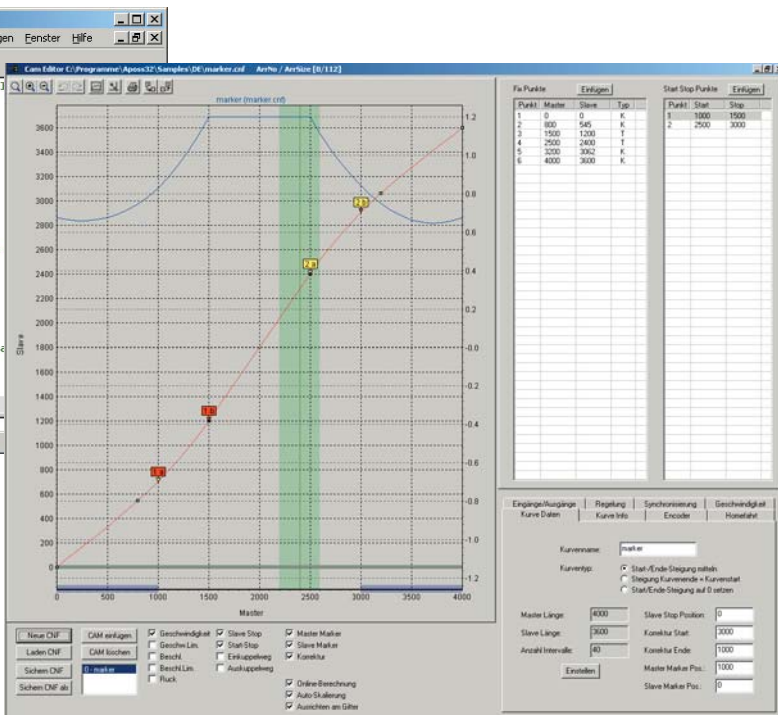
OUT 257 1 // Setzen von Ausgang 1 auf
// Modulnr * 256 + Ausgangsnummer

value = IN 515 // Lesen des Eingangs 3 auf
// Modulnr * 256 + Eingangsnummer

/* Sogar Interruptfunktionen können auf CAN-Module gelegt

ON INT 516 GOSUB userprog
// bei positiver Flanke auf Eingang

/* CAN-Module werden genauso angesprochen
nur mit höheren Ein-/Ausgangsnummern: Diese entstehen in
gleiche Weise,
Modulnr * 256 + ID number */
    
```



APOSS® WIN

ALLE APOSS-BEFEHLE IN DER ÜBERSICHT

Von ACC bis #INCLUDE

Initialisierung der Steuerung

DEFCORIGIN	Sollposition als Nullpunkt setzen
DEFMORIGIN	Aktuelle Master-Position als Nullpunkt für Master setzen
DEF ORIGIN	Aktuelle Position als Realnullpunkt setzen
DELETE ARRAYS	Alle Arrays im RAM löschen.
ERRCLR	Fehlermeldung löschen
HOME	Maschinennullpunkt anfahren
INDEX	Indexposition des Drehgebers anfahren
MOTOR OFF / ON	Motorregelung ausschalten / einschalten / Antrieb stoppen
SAVE ARRAYS / AXPARS / GLBPARS	Arrays / aktuelle Achsenparameter / aktuelle globale Parameter im EEPROM sichern
SAVEPROM	Speicher in EEPROM sichern
SET ORIGIN / RST ORIGIN	Temporärnullpunkt setzen / löschen
SETMORIGIN	Beliebige Position als Nullpunkt für den Master setzen
SWAPMENC	Master- und Slave-Drehgeber intern tauschen

Kontrollbefehle (CON)

CONTINUE	Abgebrochene Positionier- u Drehzahlbefehle fortsetzen
DELAY	Zeitverzögerung
DIM	Definition eines Arrays
EXIT	Vorzeitiger Programmabbruch
GOSUB	Unterprogramm aufrufen
GOTO	Sprung zu Programmlabel
IF THEN / .. ELSE .. IF .. THEN / .. ELSE	Bedingte einfache / bedingte mehrfache / alternative Programmverzweigung

ENDIF	Ende der Programmverzweigung
LOOP	Def. Schleifenwiederholung
MOTOR STOP	Antrieb stoppen
NOWAIT ON / OFF	Wartemodus ein-/ausschalten
REPEAT.. UNTIL	Bedingte Schleife Anfang .. Bedingte Schleife Ende
SUBMAINPROG .. ENDPROG	Beginn .. Ende der Unterprogrammdefinition
SUBPROG .. RETURN	Beginn .. Ende eines Unterprogramms
SYSVAR	Systemvariable (Pseudo-Array) liest Systemwerte
WAITAX	Warten bis Zielposition erreicht ist
WAITI	Warten auf bestimmten Eingangszustand
WAITNDX	Warten auf Index
WAITP	Warten bis Position erreicht
WAITT	Zeitverzögerung
WHILE ... DO .. ENDWHILE	While-Schleife Anfang .. Ende
#INCLUDE	Inhalt einer Datei einfügen

Positionierbefehle (ABS, REL)

ACC	Beschleunigung setzen
DEC	Bremsrampe setzen
LINA / LINR	Positioniere Achsen zeit-synchron absolut / relativ
POSA	Positioniere Achsen absolut
POSRR	Positioniere Achsen relativ
VEL	Geschwindigkeit für relative und absolute Bewegungen und max. zulässige Geschwindigkeit für Synchronisationen setzen

Drehzahlregelung (DRE)

CSTART	Drehzahlmodus starten
CSTOP	Drehzahlmodus stoppen
CVEL	Geschwindigkeit Drehzahlregelung setzen

Ein-/Ausgabe-Befehle (I/O)

APOS	Aktuelle Position lesen
AXEND	Status einer Achse abfragen
CPOS	Sollposition lesen
ERRAX	Nummer der Achse, die den Fehler auslöst
ERRNO	Fehlernummer lesen
IN / INB	Eingänge bitweise (einzeln) / bytewise (8) lesen
INAD	Analogeingänge lesen
INKEY	ASCII-Zeichen RS232 einlesen
MAPOS	Aktuelle Istposition des Masters abfragen
MAVEL / AVEL	Aktuelle Geschwindigkeit des Masters / der Achse abfragen
MIPOS / IPOS	Letzte Index- bzw. Markerposition des Masters / des Slaves abfragen
OUT	Ausgänge bitweise setzen
OUTAN	Analogausgang setzen (einer Achse zugeordnet)
OUTB	Ausgänge bytewise (8) setzen
OUTDA	(freien) Analogausgang setzen
PID	PID-Berechnung durchführen
PRINT	Informationen ausgeben
PRINTDEV	Stoppt die Ausgabe von Informationen
PWMSETF / PWMSETV	Pulsweitenmoduliertes Ausgangssignal setzen (Frequenz / Wert)
STAT	Status der Achse lesen
SYNCERR	Aktuellen Synchronisationsfehler einer Achse abfragen
TESTSETP / TESTSTART	Aufzeichnungsdaten für Testfahrt festlegen / Testfahrtaufzeichnung starten
TIME	Systemzeit auslesen
TRACKERR	Aktuellen Synchronisationsfehler einer Achse abfragen
_GETVEL	Abtastzeit für AVEL und MAVEL ändern

Interrupt-Funktionen (INT)

- DISABLE .. / DISABLE ALL**
Interrupt / alle Interrupts (ohne ON ERROR) ausschalten
- ENABLE .. / ENABLE ALL**
Interrupt / alle Int. einschalten
- ON DELETE .. GOSUB**
Löscht einen Pos.-Interrupt
- GOSUB** Unterprogramm bei folgenden Bedingungen aufrufen ...
- ON APOS .. GOSUB**
... wenn die Slave-Position xxx passiert wurde
- ON CANINPUT id GOSUB**
... wenn ein CAN-Telegramm vom Typ 'id' ankommt
- ON CANMSG .. GOSUB**
... bei Eintreffen einer gepufferten CAN-Nachricht
- ON COMBIT .. GOSUB**
... wenn Bit n des Kommunikationspuffers gesetzt ist
- ON ERROR GOSUB**
... bei Fehler
- ON INT .. GOSUB**
... bei Flanke eines Eingangs
- ON KEYPRESSED .. GOSUB**
... wenn ein Zeichen über serielle Schnittstelle kommt
- ON MAPOS / MCPOS .. GOSUB**
... wenn die Master-Position xxx [qc] / [MU] passiert wurde
- ON PARAM .. GOSUB**
... wenn sich ein Parameter ändert
- ON PERIOD .. GOSUB**
... in regelmässigen Zeitabständen
- ON STATBIT .. GOSUB**
... wenn Bit n des Statuswortes gesetzt ist
- ON TIME .. GOSUB**
... nach 1 x Ablauf der Zeit

Bahnsteuerung (Option)

- ARC** Kreisbogen in einer Bahnsequenz
- MOVE / ENDMOVE**
Start / Ende Bahnsequenz
- MVEL** Geschw. für Bahnfahrt setzen
- VEC** Gerade in einer Bahnsequenz

Parameter (PAR)

- GET** Achs- und globalen Parameterwert lesen
- LINKPDO**
Systemvariable mit PDO verknüpfen und in die internen Parameter kopieren
- LINKSDO**
Internes Objekt in PDO kopieren
- SET** Achs- und globalen Parameterwert temporär setzen

CAN-Bus (Option)

- Nur in Verbindung mit Version H.
- CANDEL** Löscht alle Objekte
- CANIN** Liest ein Objekt
- CANINI** Initialisieren des CAN-Moduls mit der eingestellten Teilnehmernummer
- CANNR** Variable mit CAN-Nummer
- CANOPENSLAVE**
Initialisiert CAN-Open-Funktionalität mit der eingestellten CAN-Nummer (id)
- CANOUT** Sendet ein Objekt
- DEFCANIN / DEFCANOUT**
Definiert ein Empfangsobjekt / ein Sendeobjekt
- INGLB** Liest eine Glb-CAN-Nachricht
- INMSG** Liest eine gepufferte CAN-Nachricht
- MSGVAL** Liefert den Long-Wert der letzten Nachricht
- OUTMSG**
Sendet eine gepufferte CAN-Nachricht
- PDO** Pseudo-Array für den direkten Zugriff auf die CANopen PDOs
- REOPEN**
CAN-Handshake für Pufferung ein- oder ausschalten
- SDOREAD / SDOWRITE**
Liest / setzt SDO eines angeschlossenen CANopen-Gerätes
- USRSTAT**
Setzt Benutzerstatus

Synchronisation (SYN)

- DEF SYNCORIGIN**
Definiert Master:Slave-Verhältnis für den nächsten SYNCOP oder SYNCM Befehl
- MOVE SYNCORIGIN**
Synchronisationsursprung relativ verschieben
- PULSACC / PULSVEL**
Beschleunigung / Geschwindigkeit für eine Master-Simulation setzen
- SYNCP** Winkel-/Positionssynchronisation
- SYNCM** Winkel-/Positionssynchronisation mit Markerkorrektur
- SYNCSTAT**
Flag für Synchronisationsstatus abfragen
- SYNCSTATCLR**
Zurücksetzen der Flags MERR und MHIT
- SYNCV** Geschwindigkeits-synchronisation

CAM-Modus (CAM)

- CURVEPOS**
Slave-Position, die der aktuellen Master-Position der Kurve entspricht, abfragen.
- POSA CURVEPOS**
Slave auf CURVEPOS fahren
- DEFCMPOS**
Anfangsposition des Masters definieren
- SETCURVE**
CAM-Kurve setzen
- SYNCC** Synchronisation im CAM-Mode
- SYNCCMM**
Synchronisation mit Markerkorrektur des Masters
- SYNCCMS**
Synchronisation des Slaves
- SYNCCSTART**
Slave zur Synchronisation im CAM-Mode starten
- SYNCCSTOP**
Slave nach der CAM-Synchronisation anhalten