isel – iLCxx Module

Bedienungsanleitung

isel-Bibliotheken und Funktionsbausteine

Art.-Nr. 970321 CD001
Stand: 15/2008

isel-automation GmbH & Co. KG ● D-36466 Dermbach ● Untere Röde 2 ● ☎ (036964)84500 ● 🖷 036964)84510

Die in dieser Druckschrift enthaltenen Informationen, technischen Daten und Maßangaben entsprechen dem neuesten technischen Stand zum Zeitpunkt der Veröffentlichung. Etwa dennoch vorhandene Druckfehler und Irrtümer können jedoch nicht ausgeschlossen werden. Für Verbesserungsvorschläge und Hinweise auf Fehler sind wir dankbar.

Es wird darauf hingewiesen, dass die in unseren Druckschriften verwendeten Soft- und Hardwarebezeichnungen der jeweiligen Firmen im allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Hersteller: ***isel***-automation GmbH & Co. KG
Untere Röde 2
D-36466 Dermbach

Tel.: (036964) 84500
Fax: (036964) 84510
email: automation@isel.com
Web: http://www.isel.com

Inhaltsverzeichnis

1 *isel*_CAN-1.lib

Die isel-CAN-1.lib Bibliothek stellt die grundsätzlichen Komponenten zur Programmierung von IMDx Endstufen in Verbindung mit iLCxx-Controllern zur Verfügung.

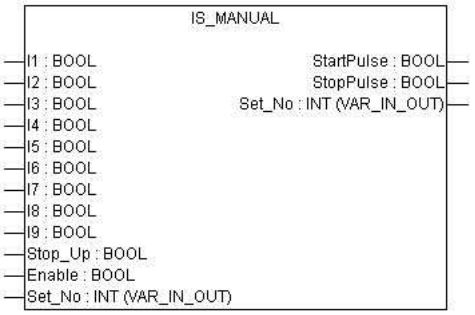
1.1 Funktionsbausteine

Funktion	**Beschreibung**
is_Manual	Baustein zum Handverfahren
is_Automatic	Baustein für den automatischen Programmablauf
is_IMDxControl_P	Baustein für die grundsätzlichen Endstufenfunktionen im Positioniermodus
is_IMDxControl_V	Baustein für die grundsätzlichen Endstufenfunktionen im Geschwindigkeitsregler-Modus
is_SPIEEProm_P	Baustein für die Verwaltung des seriellen EEPROM's in Verbindung mit Programmparametern aus der Struktur NC_Struct_P (Positioniermodus)
is_SPIEEProm_V	Baustein für die Verwaltung des seriellen EEPROM's in Verbindung mit Programmparametern aus der Struktur NC_Struct_V (Geschwindigkeitsregler-Modus)

1.1.1 is_Manual

is_Manual ist ein einfacher Baustein, um bis zu 9 Positionen in Abhängigkeit von der Eingangsbeschaltung anzusteuern.

Funktionsblock:



Definition:

```
FUNCTION_BLOCK is_Manual
VAR_INPUT
    I1, I2,I3,I4,I5,I6,I7,I8,I9:BOOL;
    Stop_Up:BOOL;
    Enable:BOOL;
END_VAR
VAR_OUTPUT
    StartPulse:BOOL:=FALSE;
    StopPulse:BOOL:=FALSE;
END_VAR
VAR_IN_OUT
    Set_No:INT;
END_VAR
```

I1..I9: Wird der angesteuerte Eingang „TRUE“, so wird die Nummer des Eingangs als „Set_No“ ausgegeben und gleichzeitig ein Startimpuls an „StartPulse“ für die Ansteuerung is_IMDxControl_P/V ausgegeben.

Stop_Up: Hat dieser Eingang den Wert „TRUE“, so wird ein Flankenwechsel HIGH→LOW an den Eingängen mit einem Impuls am Ausgang Stop_Pulse quittiert. Hierdurch wird es möglich, dass die zu bewegende Achse im Teachmodus sofort zum Stillstand kommt, wenn z.B. ein Taster losgelassen wird. In den Beispielprogrammen übernimmt diese Funktion der Schalter „Feel“.

Enable: Ist der Eingang „TRUE“, so ist der Baustein aktiv. Diese Eigenschaft kann zur Umschaltung mit einem evtl. vorhandenen Baustein is_Automatik verwendet werden.

Set_No: Nummer des angesteuerten „Ix“ Einganges bzw. der am Eingang übergebene Index des aktuellen Bewegungsdatensatzes.
Im Beispiel wird damit der Index einer vorliegenden Programmstruktur „NC_Struct_P“ oder „NC_Struct_V“ verbunden.

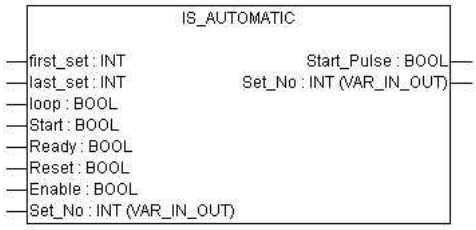
StartPulse: Dieser Ausgang liefert den Startimpuls zum Starten der Bewegung. Er wird normalerweise mit MoveStart des Bausteins is_IMDxControl_P/V verbunden. (Bei Verwendung von „is_Manual“ und „is_Automatic“ geschieht dies über eine „ODER“-Verknüpfung)

StopPulse: Dieser Ausgang wird normalerweise mit dem Eingang Abort_Segment des Bausteins „is_IMDxControl_P“ verbunden. Er liefert einen Impuls beim Flankenwechsel HIGH→Low an den Eingängen I1 – I9, wenn Stop_Up = TRUE. (s. Stop_Up)

1.1.2 is_Automatic

is_Automatik ist ein Baustein mit dem auf einfache Weise ein Programmablauf realisiert werden kann.
D.h. wenn in einer Anwendung mehrere Bewegungssegmente automatisch nacheinander abgefahren werden müssen, kann dieser Baustein verwendet werden. Dieser Baustein sollte nicht mit dem Baustein is_IMDxControl_V zusammen verwendet werden!

Funktionsblock:



Definition:

```
FUNCTION_BLOCK is_Automatic
VAR_INPUT
    first_set:INT;
    last_set:INT;
    loop:BOOL;
    Start:BOOL;
    Ready:BOOL;
    Reset:BOOL;
    Enable:BOOL:=FALSE;
END_VAR
VAR_OUTPUT
    Start_Pulse:BOOL:=FALSE;
END_VAR
VAR_IN_OUT
    Set_No:INT;
END_VAR
```

first_set:	Index für die erste Parameterzeile des Automatikprogrammes im NC_Struct_P (Datenfeld für die Bewegungsparameter).
last_set:	Index für die letzte Parameterzeile des Automatikprogrammes im NC_Struct (Datenfeld für die Bewegungsparameter)
loop:	Ist dieser Eingang „TRUE“ befindet sich das Programm in einer Endlosschleife.
Start:	Eine positive Flanke an diesem Eingang startet den automatischen Ablauf des NC-Programmes.
Ready:	Eingang, der dem Baustein anzeigt, dass die gewünschte Sollposition erreicht wurde. Dieser Eingang ist normalerweise immer mit dem Bit10 des Statuswortes zu verknüpfen.
Reset:	Eine positive Flanke an diesem Baustein setzt die Variable Set_No auf den Wert von first_set. Standardmäßig sollte dieser Eingang Eingang mit dem Reset-Eingang von is_IMDxControl_P verbunden werden.
Enable:	Ist der Eingang „TRUE“, so ist der Baustein aktiv. Dies kann zur Umschaltung mit dem Manual-Baustein verwendet werden.

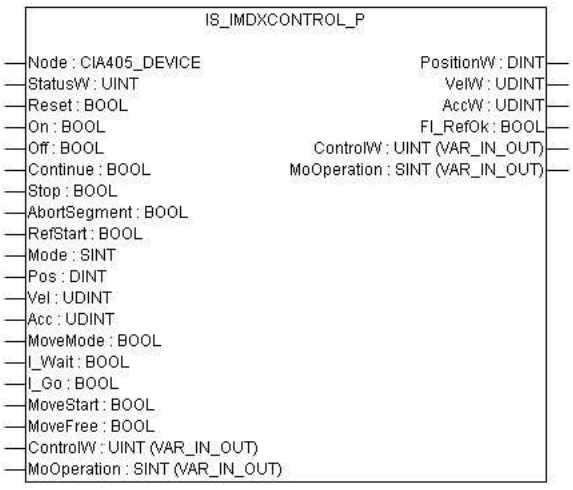
Set_No: Eingang:Index des aktuellen Bewegungsdatensatzes.
Ausgang: Index des nächsten Bewegungssatzes.
In den Beispielen wird damit der Index einer vorliegenden Programmstruktur „NC_Struct_P“ verbunden.

Start_Pulse: Dieser Ausgang gibt einen Startimpuls aus, der die Bewegung startet. Er wird normalerweise mit MoveStart des Bausteins is_IMDxControl_P verbunden. (Bei Verwendung von „is_Manual“ und „is_Automatic“ geschieht dies über eine „ODER“-Verknüpfung)

1.1.3 is_IMDxControl_P

is_IMDxControl_P ist der Baustein, der direkt die Endstufe im Positioniermodus steuert. In ihm sind Funktionen implementiert, die ein einfaches Steuern der Endstufe erlauben, ohne sich mit der Funktionsweise des CANopen Busses auseinander setzen zu müssen.

Funktionsblock:



Definition:

```
FUNCTION_BLOCK is_IMDxControl_P
VAR_INPUT
    Node: CIA405_DEVICE;
    StatusW:UINT;
    Reset: BOOL;
    On: BOOL;
    Off: BOOL;
    Continue: BOOL;
    Stop: BOOL;
    AbortSegment:BOOL;
    RefStart: BOOL;
    Mode: SINT;
    Pos: DINT;
    Vel: UDINT;
    Acc: UDINT;
    MoveMode:BOOL;
    I_Wait:BOOL;
    I_Go:BOOL;
    MoveStart:BOOL;
    MoveFree: BOOL;
END_VAR
VAR_OUTPUT
    PositionW:DINT;
    VelW:UDINT;
    AccW:UDINT;
    FI_RefOk: BOOL;
END_VAR
VAR_IN_OUT
    ControlW:UINT;
    MoOperation:SINT;
END_VAR
```

Node: Knotenadresse der Endstufe

StatusW: Die Variable des „Statusword“ aus der Steuerungskonfiguration der entsprechenden Achse sollte hiermit verknüpft werden.

Reset: Eine positive Flanke an diesem Eingang führt einen Reset an der Endstufe durch. Dieser sollte immer vor dem Einschalten der Endstufe mit „ON“ durchgeführt werden, um mögliche Fehler zurückzusetzen.

On: Ist die Hauptstromversorgung zugeschaltet und die Endstufe resetet, wird durch eine positive Planke an diesem Eingang die Endstufe in den Zustand „Operation Enable“ gesetzt. D.h. die Motoren sind nun bestromt.

Off: Eine positive Flanke an diesem Eingang „disabled“ die Endstufe. Die Motoren sind danach nicht mehr bestromt.

Continue: Eine positive Flanke an diesem Eingang führt eine durch „Stop“ angehaltene Bewegung weiter.

Stop: Eine positive Flanke an diesem Eingang hält eine begonnene Bewegung an.

AbortSegment: Eine positive Flanke an diesem Eingang bricht die Positionierung des aktuellen Segments ab. Bei Verwendung eines „is_Automatic“ Bausteins wird nach dieser Flanke mit dem nächsten Bewegungssegment fortgefahren. Man kann diese Funktion nutzen, um im Ablauf ein „Bewegung bis Signal“ zu realisieren. Der nachfolgende Befehl sollte dann ein „Absolut-Befehl sein, um keine Positionsverschiebungen zu bekommen!

RefStart: Eine positive Flanke an diesem Eingang startet eine Referenzfahrt.

Mode: 8Bit Wert, der die Betriebsart der Endstufe festlegt. Bei normalen Positionier-Anwendungen sollte dieser Wert auf „1“ stehen. Für eine Geschwindigkeitsregelung ist hier der Wert „3“ einzutragen.

Pos: 32Bit Wert. Vorgabe der Zielposition. Kann mit „Position“ von NC_Struct_P verknüpft werden. (siehe Beispielprogramm)

Vel: 16Bit Wert. Vorgabe der Sollgeschwindigkeit. Kann mit „Velocity“ von NC_Struct_P verknüpft werden. (siehe Beispielprogramm)

Acc: 16Bit Wert. Vorgabe der Sollbeschleunigung. Kann mit „Accelerartion“ von NC_Struct_P verknüpft werden. (siehe Beispielprogramm)

MoveMode: Ist dieser Eingang „TRUE“, dann wird die nächste Position relativ zur letzten Position angefahren. Bei einem „FALSE“ Wert an diesem Eingang erfolgt die Ausführung als Absolutbewegung. Kann mit „Movemode“ von NC_Struct_P verknüpft werden.

I_Wait: Ist dieser Eingang „TRUE“, dann wird die Bewegung erst gestartet, wenn der Eingang „I_Go“ eine positive Flanke erhält.
Der Eingang kann mit „Wait“ von NC_Struct_P verknüpft werden.
Es ist so möglich verschiedene Reaktionen abzuwarten. z.B.:

- warten bis eine Zeit abgelaufen ist.
- warten bis ein bestimmter Eingang gesetzt wird.

I_Go: Eine positive Flanke an diesem Eingang setzt eine durch ein „TRUE“-Signal an „I_Wait“ angehaltene Bewegung fort.
Das Signal kann durch verschiedene Ereignisse gesetzt werden. (siehe „I_Wait“)

MoveStart: Eine positive Flanke an diesem Eingang startet eine Bewegung mit den anliegenden Bewegungsparametern (z.B. durch NC_Struct_P festgelegt).

MoveFree: Eine positive Flanke an diesem Eingang bewirkt ein Freifahren aus einem Hardware-Endschalter. Hardware-Endschalter dienen zum Schutz der Achse vor Beschädigung beim Überfahren der mechanisch vorgegebenen Bewegungsbereiche. Beim Überfahren lösen sie einen Fehler aus und schalten je nach Verdrahtung des Systems die Motoren spannungsfrei. Zum Herausfahren aus dem betreffenden Endschalter muss dann die Hauptspannung wieder zugeschaltet werden.

ControlW: Die Variable des „Controlword“ aus der Steuerungskonfiguration der entsprechenden Achse sollte hiermit verknüpft werden.

MoOperation: Die Variable „Modes of Operation“ aus der Steuerungskonfiguration der entsprechenden Achse sollte hiermit verknüpft werden.

PositionW: Diese Variable verbindet den Steuerbaustein mit dem entsprechenden PDO (hier X_TargetPosition) und muss unbedingt mit angegeben werden um die Zielposition vorzugeben.

VelW: Diese Variable verbindet den Steuerbaustein mit dem entsprechenden PDO (hier X_Velocity) und muss unbedingt mit angegeben werden um die Sollgeschwindigkeit vorzugeben.

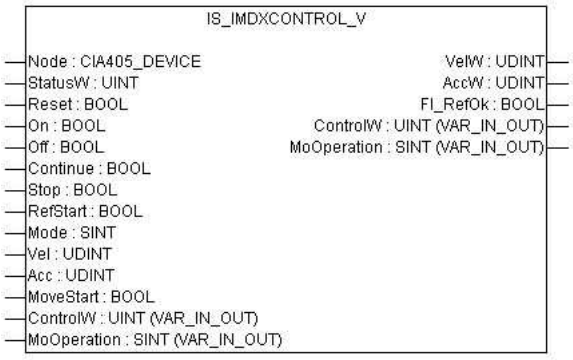
AccW: Diese Variable verbindet den Steuerbaustein mit dem entsprechenden PDO (hier X_Acceleration) und muss unbedingt mit angegeben werden um die Sollbeschleunigung vorzugeben.

Fl_RefOk: Hat dieser Ausgang den Wert „TRUE“, so ist die Referenzfahrt erfolgreich abgeschlossen.

1.1.4 is_IMDxControl_V

is_IMDxControl_V ist der Baustein, der direkt die Endstufe im Geschwindigkeitsregler-Modus steuert. In ihm sind Funktionen implementiert, die ein einfaches Steuern der Endstufe erlauben, ohne sich mit der Funktionsweise des CANopen Busses auseinander setzen zu müssen.

Funktionsblock:



Definition:

```
FUNCTION_BLOCK is_IMDxControl_V
VAR_INPUT
    Node: CIA405_DEVICE;
    StatusW:UINT;
    Reset: BOOL;
    On: BOOL;
    Off: BOOL;
    Continue: BOOL;
    Stop: BOOL;
    RefStart: BOOL;
    Mode: SINT;
    Vel: UDINT;
    Acc: UDINT;
    MoveStart:BOOL;
END_VAR
VAR_OUTPUT
    VelW:UDINT;
    AccW:UDINT;
    FI_RefOk: BOOL;
END_VAR
VAR_IN_OUT
    ControlW:UINT;
    MoOperation:SINT;
END_VAR
```

Node: Knotenadresse der Endstufe

StatusW: Die Variable des „Statusword“ aus der Steuerungskonfiguration der entsprechenden Achse sollte hiermit verknüpft werden.

Reset: Eine positive Flanke an diesem Eingang führt einen Reset an der Endstufe durch. Dieser sollte immer vor dem Einschalten der Endstufe mit „ON“ durchgeführt werden, um mögliche Fehler zurückzusetzen.

On: Ist die Hauptstromversorgung zugeschaltet und die Endstufe resetet, wird durch eine positive Planke an diesem Eingang die Endstufe in den Zustand „Operation Enable“ gesetzt. D.h. die Motoren sind nun bestromt.

Off: Eine positive Flanke an diesem Eingang „disabled“ die Endstufe. Die Motoren sind danach nicht mehr bestromt.

Continue: Eine positive Flanke an diesem Eingang führt eine durch „Stop“ angehaltene Bewegung weiter.

Stop: Eine positive Flanke an diesem Eingang hält eine begonnene Bewegung an.

RefStart: Eine positive Flanke an diesem Eingang startet eine Referenzfahrt.

Mode: 8Bit Wert, der die Betriebsart der Endstufe festlegt. Bei normalen Positionier-Anwendungen sollte dieser Wert auf „1“ stehen. Für eine Geschwindigkeitsregelung ist hier der Wert „3“ einzutragen.

Vel: 16Bit Wert. Vorgabe der Sollgeschwindigkeit. Kann mit „Velocity“ von NC_Struct_V verknüpft werden. (siehe Beispielprogramm)

Acc: 16Bit Wert. Vorgabe der Sollbeschleunigung. Kann mit „Acceleration“ von NC_Struct_V verknüpft werden. (siehe Beispielprogramm)

MoveStart: Eine positive Flanke an diesem Eingang startet eine Bewegung mit den anliegenden Bewegungsparametern (z.B. durch NC_Struct_V festgelegt).

ControlW: Die Variable des „Controlword“ aus der Steuerungskonfiguration der entsprechenden Achse sollte hiermit verknüpft werden.

MoOperation: Die Variable „Modes of Operation“ aus der Steuerungskonfiguration der entsprechenden Achse sollte hiermit verknüpft werden.

VelW: Diese Variable verbindet den Steuerbaustein mit dem entsprechenden PDO (hier X_TargetVelocity) und muss unbedingt mit angegeben werden um die Sollgeschwindigkeit vorzugeben.

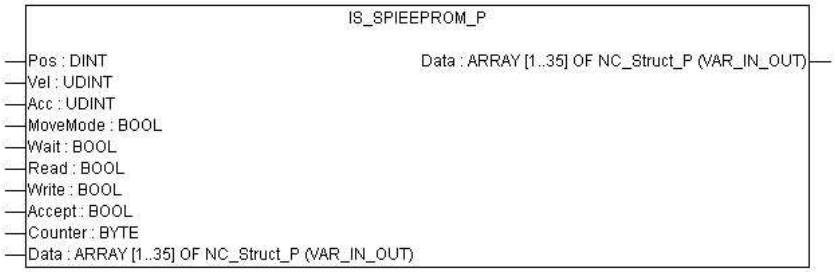
AccW: Diese Variable verbindet den Steuerbaustein mit dem entsprechenden PDO (hier X_Acceleration) und muss unbedingt mit angegeben werden um die Sollbeschleunigung vorzugeben.

Fl_RefOk: Hat dieser Ausgang den Wert „TRUE“, so ist die Referenzfahrt erfolgreich abgeschlossen.

1.1.5 is_SPIEEProm_P

is_SPIEEProm_P ist ein Funktionsbaustein um den in der iLC10/20 integrierten Flashspeicher zum Abspeichern von Bewegungsparametersätzen nach NC_Struct_P zu verwenden. Hiermit können bis zu 35 Datensätze für eine Achse im Positioniermodus abgespeichert werden. Dies ist z.B. dann sinnvoll, wenn sich während der Programmabarbeitung irgendwelche Werte ändern, die über das Abschalten der Anlage hinaus erhalten werden sollen, ohne diese wieder im Programm (CoDeSys) editieren zu müssen.

Funktionsblock:



Definition:

```
(*for saving and restoring values of NC_Struct_P at the flash memory*)
FUNCTION_BLOCK is_SPIEEProm_P
VAR_INPUT
    Pos: DINT;
    Vel: UDINT;
    Acc: UDINT;
    MoveMode:BOOL;
    Wait: BOOL;
    Read:BOOL;
    Write: BOOL;
    Accept:BOOL;
    Counter:BYTE:=1;
END_VAR


VAR_IN_OUT
    Data: ARRAY[1..35]OF NC_Struct_P;
END_VAR
```

Pos: Wert des abzuspeichernden Positionsparameters

Vel: Wert des abzuspeichernden Geschwindigkeitsparameters

Acc: Wert des abzuspeichernden Beschleunigungsparameters

MoveMode: Wert des abzuspeichernden MoveMode („TRUE“=relativ, „FALSE“=absolut)

Wait: Wert des abzuspeichernden Wait-Parameters (siehe NC_Struct_P)

Read: Eine positive Flanke an diesem Eingang liest die im Flashspeicher gespeicherten Daten in das von NC_Struct_P abgeleitete Datenfeld.

Write: Eine positive Flanke an diesem Eingang speichert die geänderten Werte in dem von NC_Struct_P abgeleiteten Datenfeld.

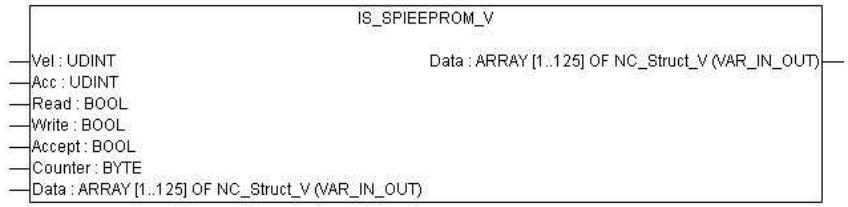
Accept: Eine positive Flanke übernimmt die am Baustein anliegenden Daten in den internen Speicher. Sie sind dadurch noch nicht im Flash-Speicher abgespeichert. Das endgültige Speicher wird erst über eine positive Flanke am Eingang „Write“ vollzogen.

Counter: Indexvorgabe für das von NC_Struct_V abgeleitete Datenfeld, um verschiedene Parametersätze anzuwählen. Dieser Parameter kann z.B. mit einer Tabelle in der Visualisierung verknüpft werden. (siehe Programmbeispiel 3)

1.1.6 is_SPIEEProm_V

is_SPIEEProm_V ist ein Funktionsbaustein um den in der iLC10/20 integrierten Flashspeicher zum Abspeichern von Bewegungsparametersätzen nach NC_Struct_V zu verwenden. Hiermit können bis zu 125 Datensätze für eine Achse im Geschwindigkeitsregler-Modus abgespeichert werden. Dies ist z.B. dann sinnvoll, wenn sich während der Programmabarbeitung irgendwelche Werte ändern, die über das Abschalten der Anlage hinaus erhalten werden sollen, ohne diese wieder im Programm (CoDeSys) editieren zu müssen.

Funktionsblock:



Definition:

```
(*for saving and restoring values of NC_Struct_V at the flash memory*)
FUNCTION_BLOCK is_SPIEEProm_V
VAR_INPUT
    Vel: UDINT;
    Acc: UDINT;
    Read:BOOL;
    Write: BOOL;
    Accept:BOOL;
    Counter:BYTE:=1;
END_VAR


VAR_IN_OUT
    Data: ARRAY[1..125]OF NC_Struct_V;
END_VAR
```

Vel:	Wert des abzuspeichernden Geschwindigkeitsparameters
Acc:	Wert des abzuspeichernden Beschleunigungsparameters
Read:	Eine positive Flanke an diesem Eingang liest die im Flashspeicher gespeicherten Daten in das von NC_Struct_V abgeleitete Datenfeld.
Write:	Eine positive Flanke an diesem Eingang speichert die geänderten Werte in dem von NC_Struct_V abgeleiteten Datenfeld.
Accept:	Eine positive Flanke übernimmt die am Baustein anliegenden Daten in den internen Speicher. Sie sind dadurch noch nicht im Flash-Speicher abgespeichert. Das endgültige Speicher wird erst über eine positive Flanke am Eingang „Write“ vollzogen.
Counter:	Indexvorgabe für das von NC_Struct_V abgeleitete Datenfeld, um verschiedene Parametersätze anzuwählen. Dieser Parameter kann z.B. mit einer Tabelle in der Visualisierung verknüpft werden.

1.2 Strukturen

Funktion	Beschreibung
NC_Struct_P	Struktur für die Datenfelder der Bewegungsparameter im Positioniermodus
NC_Struct_V	Struktur für die Datenfelder der Bewegungsparameter im Geschwindigkeitsregler-Modus

1.2.1 NC_Struct_P

Die Datenstruktur *NC_Struct_P* ist eine Struktur zur Ableitung einer Variable eines Datenfeldes zur Positioniersteuerung einer Achse. Hier können die Bewegungsparameter für die manuelle Bewegung oder eines Ablaufprogrammes hinterlegt werden. Bei der aktuellen Version können 35 Datensätze angelegt werden.

Definition:

```
TYPE NC_Struct_P :
STRUCT
    Position:DINT;
    Velocity:UDINT;
    Acceleration:UDINT;
    MoveMode:BOOL;
    Wait:BOOL;
END_STRUCT
END_TYPE
```

Position: Position, der Bewegungsausführung angefahren werden soll.

Velocity: Geschwindigkeit für die Ausführung der Bewegung

Acceleration: Beschleunigung für die Ausführung der Bewegung

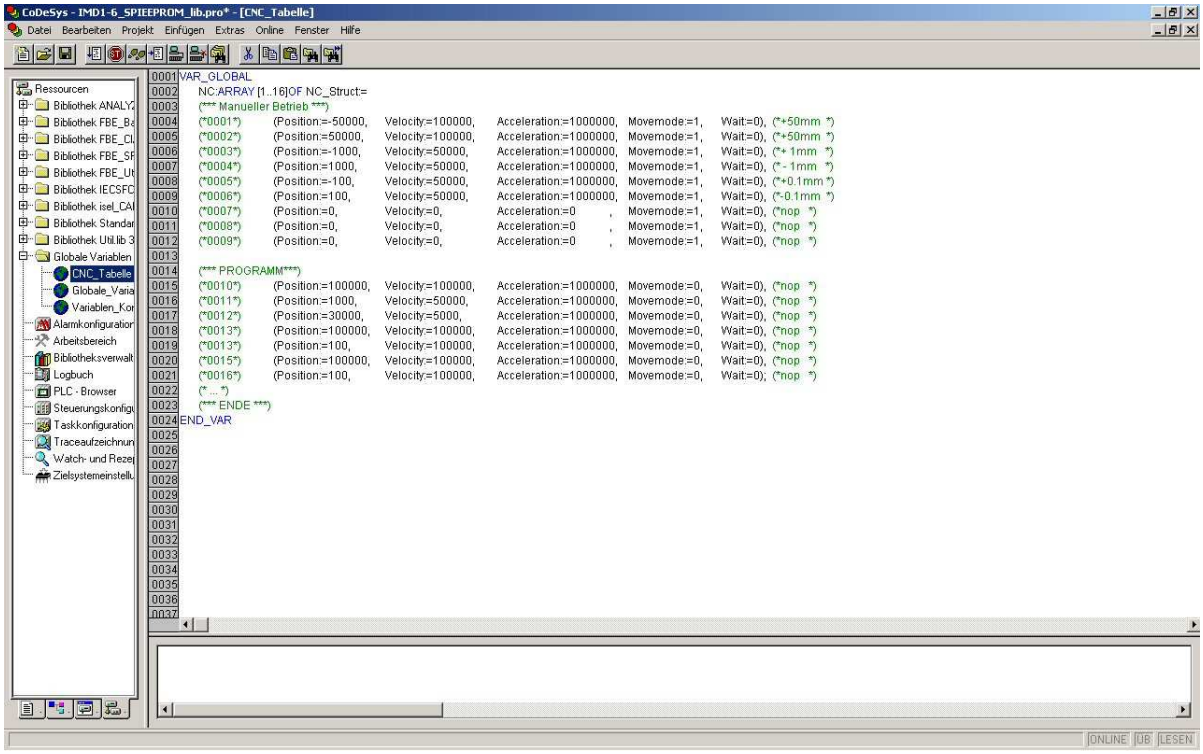
MoveMode: Wert="TRUE" → Bewegung wird als Relativbewegung ausgeführt.

Wert="FALSE" → Bewegung wird als Absolutbewegung ausgeführt.

Wait: Wert="TRUE" → Bewegung wird erst gestartet, wenn der Eingang „I_Go" des Bausteins is_IMDxControl_P „TRUE" wird.

Wert="FALSE" → Programm wird ohne Wartefunktion vom is_IMDxControl_P abgearbeitet. In diesem Fall ist der Zustand des Einganges „I_Go" von is_IMDxControl_P ohne Belang.

Beispiel für das Anlegen eines von NC_Struct_P abgeleiteten Datenfeldes



Das Datenfeld wurde hier globale Variable angelegt. Dazu wurde ein zusätzlicher Ordner in den globalen Variablen angelegt. Dies ist nicht unbedingt notwendig, erleichtert aber die Übersichtlichkeit.
Die Felder 1 – 10 sind hier für manuelle Bewegungen reserviert.
Beginnend mit der Zeile 10 fängt das Programm an. Wird eine neue Zeile hinzugefügt, so muss dies in der Deklaration des Datenfeldes auch angepasst werden. → [ARRAY1..35]
Auch muss darauf geachtet werden, dass sich dann der Wert für „first_set“, bzw. „last_set“ des Bausteins „is_Automatic“ ändert.

Die Zeilen sind durch Komma getrennt!
Hinter der letzten Zeile muss ein Semikolon stehen!

1.2.2 NC_Struct_V

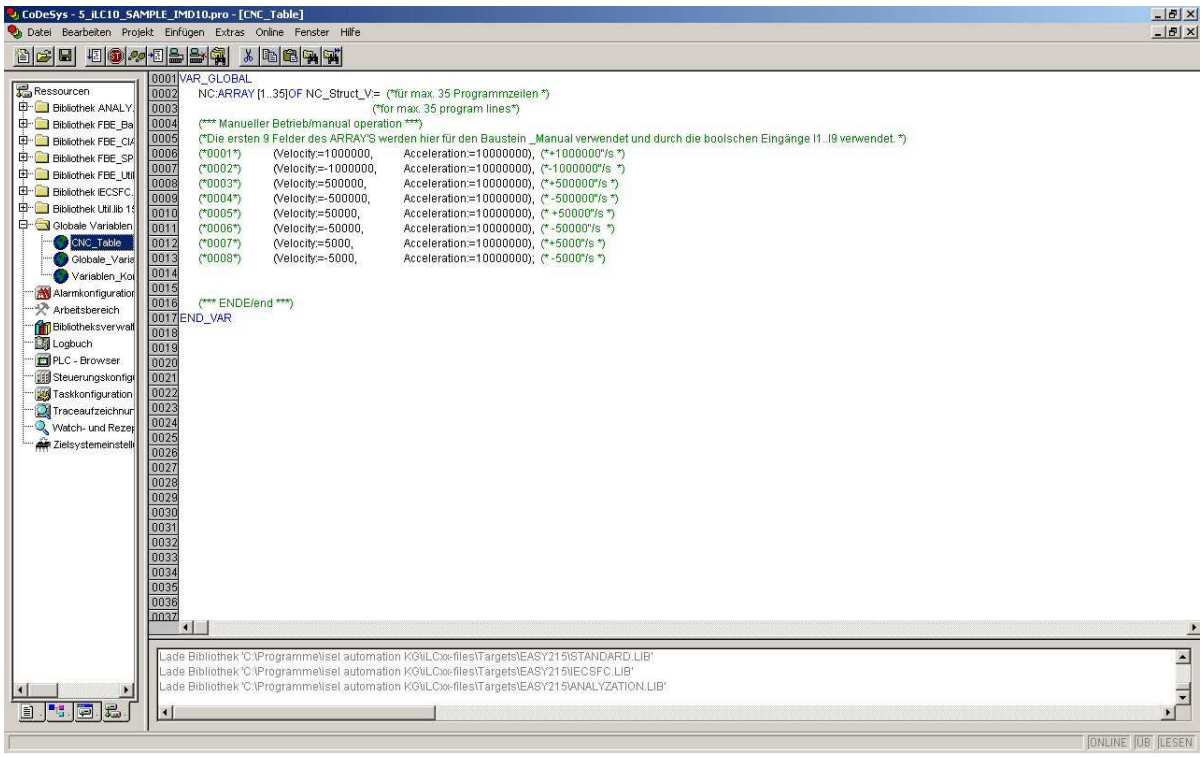
Die Datenstruktur *NC_Struct_V* ist eine Struktur zur Ableitung einer Variable eines Datenfeldes zur Geschwindigkeitsregelung einer Achse. Hier können die Bewegungsparameter für die manuelle Bewegung oder eines Ablaufprogrammes hinterlegt werden. Bei der aktuellen Version können 125 Datensätze angelegt werden.

Definition:

```
TYPE NC_Struct_V :
STRUCT
    Velocity:DINT;
    Acceleration:UDINT;
END_STRUCT
END_TYPE
```

Velocity: Zielgeschwindigkeit für die Ausführung der Bewegung

Acceleration: Beschleunigung für die Ausführung der Bewegung



Das Datenfeld wurde hier globale Variable angelegt. Dazu wurde ein zusätzlicher Ordner in den globalen Variablen angelegt. Dies ist nicht unbedingt notwendig, erleichtert aber die Übersichtlichkeit.
Die Felder 1 – 10 sind hier für manuelle Bewegungen reserviert.
Beginnend mit der Zeile 10 fängt das Programm an. Wird eine neue Zeile hinzugefügt, so muss dies in der Deklaration des Datenfeldes auch angepasst werden. → [ARRAY1..35]
Auch muss darauf geachtet werden, dass sich dann der Wert für „first_set“, bzw. „last_set“ des Bausteins „is_Automatic“ ändert.

Die Zeilen sind durch Komma getrennt!
Hinter der letzten Zeile muss ein Semikolon stehen!

1.3 Beispielprogramme

Mit der Bibliothek „isel_CAN-1.lib“ werden auch Beispielprogramme mitgeliefert, durch welche der praktische Einsatz der oben beschriebenen Bausteine näher gezeigt werden soll. Um die Beispielprogramme ausführen zu können, ist eine korrekte Installation von CoDeSys und Durchführung der entsprechenden Einstellungen notwendig. (siehe Dokumentation: „iLC_CoDeSys Einstellungen“) Die Beispiele sind schon auf die Typen iLC10 bzw. iLC20 abgestimmt. Es ist also nur das entsprechende Beispiel zu öffnen. (aus dem Dateinamen ersichtlich)

1.3.1 Überblick

Programmname	Verwendete Bausteine
1_iLC10_SAMPLE.pro 1_iLC20_SAMPLE.pro	- Positioniermodus - Baustein is_Manual - Baustein is_IMDxControl_P
2_iLC10_SAMPLE.pro 2_iLC20_SAMPLE.pro	- Positioniermodus - Baustein is_Automatic - Baustein is_IMDxControl_P
3_iLC10_SAMPLE.pro 3_iLC20_SAMPLE.pro	- Positioniermodus - Baustein is_Manual - Baustein is_Automatic - Baustein is_SPIEEPROM_P - Baustein is_IMDxControl_P
4_iLC10_SAMPLE.pro 4_iLC20_SAMPLE.pro	- Positioniermodus - Baustein is_Automatic - Baustein is_IMDxControl_P - Funktion “AbortSegment” - Funktion “iWait/iGo”
5_iLC10_SAMPLE.pro 5_iLC20_SAMPLE.pro	- Geschwindigkeitsregler-Modus - Baustein is_Manual - Baustein is_IMDxControl_V
6_iLC10_SAMPLE.pro 6_iLC20_SAMPLE.pro	- Positioniermodus - 2 Achsen - Baustein is_Automatik - 2 x Baustein is_IMDxControl_P - Funktion “AbortSegment” - Funktion “iWait/iGo”

1.3.2 Hinweise zur Verwendung

Zur Verwendung der Beispielprogramme könnten folgende Hinweise noch wichtig für die Bedienung sein:

- Alle Beispielprogramme benötigen eine .DCF-Datei mit einem bestimmten Namen. „iLC-sample.dcf“ oder iLC-sample_vel.dcf.
- Überprüfen Sie bitte nach dem Öffnen eines Beispielprogrammes die Bewegungsparameter und passen Sie evtl. die Positionen, Geschwindigkeiten, Beschleunigungen an Ihre Umgebung an. Klicken Sie dazu auf den Reiter „Ressourcen“ → „Globale_Variablen“ → „CNC_Table“.

- Nach dem Laden in die Steuerung ist das Programm in CoDeSys mit Online → Start zu starten.

- In CoDeSys können auf der Seite der Visualisierung je nach Programm verschiedene Bedienpanel aufgerufen werden.
  Diese sind durch Doppelklick auszuwählen. Der Baustein PLC_VISU ist in jedem Programm vorhanden.
  Hiermit können die verschiedenen Eingänge der Endstufe über die Software simuliert werden.

- Nach dem Starten ist die Endstufe zu reseten (RESET-Taste drücken)

- Endstufe mit „ON“ einschalten

- REFERENZ drücken um die Referenzfahrt zu starten.