DOKUMENTATION ISG-kernel

Funktionsbeschreibung
Export V.E.-Variablen in SPS-Struktur

Kurzbezeichnung:
FCT-C22

www.isg-stuttgart.de
support@isg-stuttgart.de

Dokumentation Version: 1.24
12.12.2023

Allgemeine- und Sicherheitshinweise

Verwendete Symbole und ihre Bedeutung

In der vorliegenden Dokumentation werden die folgenden Symbole mit nebenstehendem Sicherheitshinweis und Text verwendet. Die (Sicherheits-) Hinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

Symbole im Erklärtext

➢ Gibt eine Aktion an.

⇨ Gibt eine Handlungsanweisung an.

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen und Maschinen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen und Maschinen geschädigt werden!

Achtung

Einschränkung oder Fehler

Dieses Symbol beschreibt Einschränkungen oder warnt vor Fehlern.

Hinweis

Tipps und weitere Hinweise

Dieses Symbol kennzeichnet Informationen, die zum grundsätzlichen Verständnis beitragen oder zusätzliche Hinweise geben.

Beispiel

Allgemeines Beispiel

Beispiel zu einem erklärten Sachverhalt.

Programmierbeispiel

NC-Programmierbeispiel

Programmierbeispiel (komplettes NC-Programm oder Programmsequenz) der beschriebenen Funktionalität bzw. des entsprechenden NC-Befehls.

Versionshinweis

Spezifischer Versionshinweis

Optionale, ggf. auch eingeschränkte Funktionalität. Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Inhaltsverzeichnis

Allgemeine- und Sicherheitshinweise **2**

1 Übersicht **4**

2 Beschreibung **5**

2.1 Erzeugung der Ausgabedatei (#EXPORT VE) 5

2.2 Fehler beim Export 6

3 Beispiele **7**

3.1 Beispiel 1- Verwenden kurzer Textstrings 7

3.1.1 V.E.-Variablenliste 7

3.1.2 Exemplarische Ausgabedatei für CODESYS 8

3.1.3 SPS- Beispiel 9

3.1.4 Exemplarische Ausgabedatei für MULTIPROG 10

3.2 Beispiel 2- Verwenden langer Textstrings 11

3.2.1 V.E.-Variablenliste 11

3.2.2 Exemplarische Ausgabedatei für CODESYS 12

3.2.3 SPS- Beispiel 13

3.3 Beispiel Abstandsregelung 14

3.3.1 V.E.-Variablenliste 14

3.3.2 Export der V.E.-Variablenliste 14

3.3.3 Import des Codes in SPS 14

3.3.4 Integration in SPS-Programm 15

4 Einschränkungen bei mehrkanaligem Steuerungsaufbau **16**

5 Kurzanleitung zur Nutzung der Exportfunktionalität bei V.E.- Variablen. **17**

6 Parameter **18**

7 Anhang **19**

Stichwortverzeichnis **20**

1 Übersicht

Aufgabe

Die Exportfunktionalität erzeugt aus einer vorhandenen "Liste der Externen Variablen" (folgend V.E.-Liste oder -Variable) einer Maschinenkonfiguration eine kanalspezifische Datenstruktur, die alle Variablen enthält.

Eigenschaften

Diese generierte Datenstruktur kann in eine SPS-Umgebung importiert werden und ermöglicht so den Zugriff der SPS auf die V.E.-Variablen. Somit kann schnell und sicher eine Schnittstelle zwischen NC-Steuerung und SPS für den Datenaustausch erstellt werden.

Parametrierung

Der Parameter P-EXTV-00022 legt fest, welche vorgegebene Anzahl von Zeichen bei Stringvariablen verwendet wird.

Programmierung

Der Export der Datenstruktur erfolgt über den Befehl #EXPORT VE[...] [▶ 5] in einem kleinen NC-Programm. Da sich die Konfiguration der V.E.-Variablen nach dem Hochlauf nicht mehr ändert, wird dieser Vorgang üblicherweise nur einmalig während der Inbetriebnahme der Maschine durchgeführt.

Obligatorischer Hinweis zu Verweisen auf andere Dokumente

Zwecks Übersichtlichkeit wird eine verkürzte Darstellung der Verweise (Links) auf andere Dokumente bzw. Parameter gewählt, z.B. [PROG] für Programmieranleitung oder P-AXIS-00001 für einen Achsparameter.

Technisch bedingt funktionieren diese Verweise nur in der Online-Hilfe (HTML5, CHM), allerdings nicht in PDF-Dateien, da PDF keine dokumentenübergreifenden Verlinkungen unterstützt.

2 Beschreibung

Datenaustausch zwischen SPS und CNC über V.E.-Variablen

V.E-Variablen ermöglichen den Austausch von Daten zwischen einem NC-Programm und der SPS in beliebiger Richtung.

Die Nachbildung der V.E.-Variablen als Datenstruktur in der SPS ermöglicht den Zugriff auf diese V.E.-Variablen aus der SPS.

Ausgangslage

Es ist eine Variablenliste der Konfiguration angelegt.

2.1 Erzeugung der Ausgabedatei (#EXPORT VE)

Der NC-Befehl **#EXPORT VE [..]** erzeugt für den Kanal die entsprechende Datenstruktur für die V.E.-Variablen, in dem der Befehl verwendet wird.

Bei mehrkanaligen Systemen muss der NC-Befehl in jedem Kanal eingesetzt werden um die Datenstruktur für den jeweiligen Kanal zu erzeugen.

Programmiersyntax

Syntax:

#EXPORT VE [3S | TWINCAT | KW] non-modal

3S / TWINCAT — Für TwinCAT sowie die originäre 3S CODESYS SPS-Umgebung:

Ausgabedatei: plc_3s_ve_types_ch_**<i>**.exp

Ausgabeverzeichnis:

- Unter TwinCAT SystemManager : CNC-Konfiguration - CNC-Task GEO - Reiter HLI - Eingabefeld: NC Datei Pfad
- Ohne TwinCAT, nur 3S: Verzeichnisangabe über P-STUP-00020 oder ab V3.1.3052.05 über P-CHAN-00403)

KW — Für MULTIPROG SPS-Umgebung:

Ausgabedatei: plc_kw_ve_types_ch_**<i>**.exp

Ausgabeverzeichnis: applikationsspezifisch (P-STUP-00020 oder ab V3.1.3052.05 mit P-CHAN-00403)

Ist bei einer TwinCAT-Konfiguration kein Ausgabeverzeichnis festgelegt, so wird die Ausgabedatei in Abhängigkeit von der TwinCAT-Version in folgendes Verzeichnis platziert:

- TwinCAT 2 32-bit: Hauptverzeichnis C:\
- TwinCAT 3 64-bit: C:\Windows\SysWOW64

Voraussetzung dafür sind die entsprechenden Schreibrechte im jeweiligen Verzeichnis.

Hinweis

Der Bezeichner <i> im Dateinamen der Ausgabedatei ist ein Platzhalter für die CNC-Kanalnummer.

Hinweis

Der Aufruf des CNC-Befehls #EXPORT VE benötigt als Parameter zwingend die Angabe des SPS-Zielsystems. Das Ergebnis wird entsprechend benannt.

Bei fehlendem Parameter wird die Fehlermeldung 20509 ausgegeben.

Programmierbeispiel

Erzeugung der Ausgabedatei

```
#EXPORT VE [TWINCAT] ;Erzeugung V.E.-SPS-Struktur für TwinCAT
#EXPORT VE [3S] ;Erzeugung V.E.-SPS-Struktur für 3S CODESYS
#EXPORT VE [KW] ; Erzeugung V.E.-SPS-Struktur für MULTIPROG von KW
```

Der Befehl kann in einem NC-Programm stehen oder als Handsatz ausgeführt werden. Durch den Befehl wird eine Datei erzeugt, die für alle im NC-Kanal angelegten V.E-Variablen in einer Datenstruktur, gemäß IEC 61131-3, deklariert werden.

Die erzeugte Datei entspricht dem Im-/Exportformat für die Entwicklungsumgebungen CODESYS bzw. MULTIPROG und kann dort direkt importiert werden.

Hinweis

In der Ausgabedatei sind zusätzliche Strukturdeklarationen erforderlich.

2.2 Fehler beim Export

Bevor die SPS-Datenstruktur durch die Funktion erzeugt wird, erfolgt eine Überprüfung der Deklaration der V.E-Variablen.

Aufgetretene Fehlermeldungen werden in der (EXPORT-) Ausgabedatei protokolliert!

3 Beispiele

Die beiden nachfolgenden Beispiele zeigen wie die exportierte SPS-Struktur ausgehend von einer V.E.-Variablenlisten in ein SPS-Projekt integriert wird.

Beide Beispiele unterscheiden sich nur in der Parametrierung von P-EXTV-00022. Dieser Parameter ist ausschlaggebend wie lang Variablen vom Typ String sein dürfen. Die Länge des Typs hat Auswirkungen auf das erzeugt Speicherlayout.

3.1 Beispiel 1- Verwenden kurzer Textstrings

3.1.1 V.E.-Variablenliste

Belegung in V.E. Variablenliste:

```
#
use_extended_string_var      0    # P-EXTV-00022
#
anzahl_belegt                4
#
var[0].name                  FARBE
var[0].type                  UNS16
var[0].scope                 CHANNEL
var[0].synchronisation       FALSE
var[0].access_rights         READ_WRITE
var[0].array_elements        3
#
var[1].name                  TEXT
var[1].type                  STRING
var[1].scope                 CHANNEL
var[1].synchronisation       FALSE
var[1].access_rights         READ_WRITE
var[1].array_elements        2
#
var[2].name                  INFO_IN
var[2].type                  OFFSET
var[2].scope                 GLOBAL
var[2].synchronisation       FALSE
var[2].access_rights         READ_WRITE
var[2].array_elements        2
#
var[3].name                  INFO_OUT
var[3].type                  OFFSET
var[3].scope                 GLOBAL
var[3].synchronisation       FALSE
var[3].access_rights         READ_WRITE
var[3].array_elements        2
var[3].create_hmi_interface  0
#
struct[0].name               OFFSET
struct[0].element[0].name    X
struct[0].element[0].type    UNS16
struct[0].element[1].name    Y
struct[0].element[1].type    UNS16
#
```

3.1.2 Exemplarische Ausgabedatei für CODESYS

Darstellung in der exportierten Datei:

```
TYPE STRING_20:
STRUCT
    token:STRING(20);
    f1_st: ARRAY[0..2] OF BYTE;
END_STRUCT
END_TYPE

TYPE STRING_20_2:
STRUCT
    token:STRING(20);
    f1_st: ARRAY[0..106] OF BYTE;
END_STRUCT
END_TYPE

TYPE OFFSET:
STRUCT
    X: UINT;
    Y: UINT;
END_STRUCT
END_TYPE

TYPE VE_CHANNEL_DATA_CH_1:
STRUCT
    FARBE: ARRAY[0..2] OF UINT;
    f1: ARRAY[0..17] OF BYTE;
    TEXT: ARRAY[0..1] OF STRING_20;
END_STRUCT
END_TYPE

TYPE VE_GLOBAL_DATA_FROM_CH_1:
STRUCT
    INFO_IN: ARRAY[0..1] OF OFFSET;
    INFO_OUT: ARRAY[0..1] OF OFFSET;
END_STRUCT
END_TYPE
```

3.1.3 SPS- Beispiel

Integration der Struktur in 3S SPS-Programm:

```
VAR
  (* Nutzen der erzeugten Strukturbeschreibungen *)
  p_ve_chan_1 : POINTER TO VE_CHANNEL_DATA_CH_1;
  p_ve_glob   : POINTER TO VE_GLOBAL_DATA_FROM_CH_1;
  text        : STRING(20);
  init_ve_ptr : BOOL := TRUE;

END_VAR

(* Sicherstellung, dass die internen Verwaltungsdaten initialisiert sind
*)

Hli(Start := TRUE);


IF Hli.Initialized = TRUE AND Hli.Error = FALSE THEN

  IF init_ve_ptr = TRUE THEN

    (* Zeiger auf Struktur(en) bereitstellen *)
    p_ve_chan_1 := ADR( gpVECH[0]^.ext_var32[0]);
    p_ve_glob   := ADR(gpVEGlobal^.ext_var32[0]);

  END_IF;

  (* Mit den Variablen arbeiten (lesen, schreiben) *)
  text := p_ve_chan_1^.TEXT[0].token;
  p_ve_chan_1^.FARBE[1] := 2;
END_IF
```

3.1.4 Exemplarische Ausgabedatei für MULTIPROG

Der nachfolgende Export für MULTIPROG basiert auf der identischen V.E.-Variablenliste [▸ 7] wie der Export für CODESYS [▸ 8].

```
TYPE
  TYPE_STRING_20 : ARRAY[0..20] OF BYTE;
END_TYPE

TYPE
  ALIGN_STRING_20_1 : ARRAY[0..2] OF BYTE;
END_TYPE

TYPE
  ALIGN_STRING_20_2 : ARRAY[0..106] OF BYTE;
END_TYPE

TYPE STRING_20_1:
STRUCT
  Token     : TYPE_STRING_20;
  alignment : ALIGN_STRING_20_1;
END_STRUCT;
END_TYPE

TYPE STRING_20_2:
STRUCT
  Token     : TYPE_STRING_20;
  alignment : ALIGN_STRING_20_2;
END_STRUCT;
END_TYPE

TYPE OFFSET:
STRUCT
  X: UINT;
  Y: UINT;
END_STRUCT;
END_TYPE

TYPE
  T2_FARBE : ARRAY[0..2] OF UINT;
END_TYPE

TYPE
  F1_2:ARRAY[0..17] OF BYTE;
END_TYPE

TYPE
  T2_TEXT : ARRAY[0..1] OF STRING_20_1;
END_TYPE

TYPE VE_CHANNEL_DATA_CH_1:
STRUCT
    FARBE: T2_FARBE;   (* index = 0 *)
    f1 : F1_2;
    TEXT: T2_TEXT;   (* index = 1 *)
END_STRUCT;
END_TYPE

TYPE
  T3_OFFSET : ARRAY[0..1] OF OFFSET;
END_TYPE

TYPE
```

```
  T3_OFFSET : ARRAY[0..1] OF OFFSET;
END_TYPE

TYPE VE_GLOBAL_DATA_FROM_CH_1:
STRUCT
    INFO_IN: T3_OFFSET;
    INFO_OUT: T3_OFFSET;
END_STRUCT;
END_TYPE
```

3.2 Beispiel 2- Verwenden langer Textstrings

3.2.1 V.E.-Variablenliste

Belegung in V.E. Variablenliste:

```
#
use_extended_string_var      1    # P-EXTV-00022
#
anzahl_belegt                4
#
var[0].name                  FARBE
var[0].type                  UNS16
var[0].scope                 CHANNEL
var[0].synchronisation       FALSE
var[0].access_rights         READ_WRITE
var[0].array_elements        3
#
var[1].name                  TEXT
var[1].type                  STRING
var[1].scope                 CHANNEL
var[1].synchronisation       FALSE
var[1].access_rights         READ_WRITE
var[1].array_elements        2
#
var[2].name                  INFO_IN
var[2].type                  OFFSET
var[2].scope                 GLOBAL
var[2].synchronisation       FALSE
var[2].access_rights         READ_WRITE
var[2].array_elements        2
#
var[3].name                  INFO_OUT
var[3].type                  OFFSET
var[3].scope                 GLOBAL
var[3].synchronisation       FALSE
var[3].access_rights         READ_WRITE
var[3].array_elements        2
var[3].create_hmi_interface  0
#
struct[0].name               OFFSET
struct[0].element[0].name    X
struct[0].element[0].type    UNS16
struct[0].element[1].name    Y
struct[0].element[1].type    UNS16
#
```

3.2.2 Exemplarische Ausgabedatei für CODESYS

Darstellung in der exportierten Datei:

```
TYPE OFFSET:
STRUCT
    X: UINT;
    Y: UINT;
END_STRUCT
END_TYPE

TYPE VE_CHANNEL_DATA_CH_1:
STRUCT
    FARBE: ARRAY[0..2] OF UINT;
    TEXT: ARRAY[0..1] OF STRING(127);
END_STRUCT
END_TYPE

TYPE VE_GLOBAL_DATA_FROM_CH_1:
STRUCT
    INFO_IN: ARRAY[0..1] OF OFFSET;
    INFO_OUT: ARRAY[0..1] OF OFFSET;
END_STRUCT
END_TYPE
```

3.2.3 SPS- Beispiel

Integration der Struktur in 3S SPS-Programm:

```
VAR
  (* Nutzen der erzeugten Strukturbeschreibungen *)
  p_ve_chan_1 : POINTER TO VE_CHANNEL_DATA_CH_1;
  p_ve_glob   : POINTER TO VE_GLOBAL_DATA_FROM_CH_1;
  text        : STRING(128);
  init_ve_ptr : BOOL := TRUE;

END_VAR

(* Sicherstellung, dass die internen Verwaltungsdaten initialisiert sind
*)

Hli(Start := TRUE);


IF Hli.Initialized = TRUE AND Hli.Error = FALSE THEN

  IF init_ve_ptr = TRUE THEN
    (* Zeiger auf Struktur(en) bereitstellen *)
    p_ve_chan_1 := ADR( gpVECH[0]^.ext_var32[0]);
    p_ve_glob   := ADR(gpVEGlobal^.ext_var32[0]);



  END_IF;

  (* Mit den Variablen arbeiten (lesen, schreiben) *)
  text := p_ve_chan_1^.TEXT[0].token;
  p_ve_chan_1^.FARBE[1] := 2;
END_IF
```

3.3 Beispiel Abstandsregelung

3.3.1 V.E.-Variablenliste

```
#
use_extended_string_var      0
#
number_used_variables        2
#
var[0].name                  sensor
var[0].type                  SGN32
var[0].scope                 GLOBAL
var[0].synchronisation       FALSE
var[0].access_rights         READ_WRITE
var[0].array_size            0
#
var[1].name                  sensor_ch1
var[1].type                  REAL64
var[1].scope                 CHANNEL
var[1].synchronisation       TRUE
var[1].access_rights         READ_ONLY
var[1].array_size            2
```

3.3.2 Export der V.E.-Variablenliste

Der NC-Befehl für den Export der V.E.-Variablenliste zur Nutzung mit CoDeSys ist wie folgt:

```
#EXPORT VE[3S]
```

Die Einstellung für den Standardpfad für NC-Programme ist wie folgt:

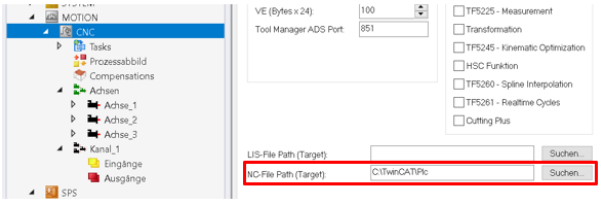

Abb. 1: Beispiel3- Einstellung Standardprogrammpfad

Die exportierte Datei hat den Namen „plc_3s_ve_types_ch_1.xml“ und ist in obigem Standard-pfad zu finden.

3.3.3 Import des Codes in SPS

Das Exportergebnis kann dann über „Rechtsklick“ importiert werden.

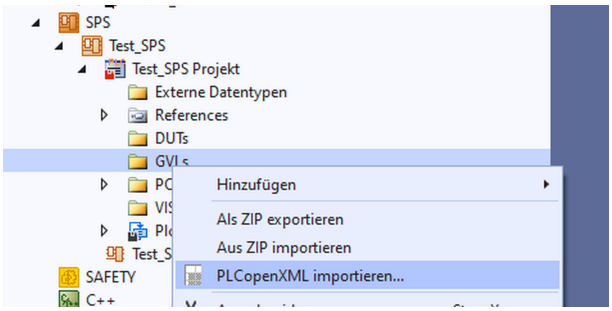

Abb. 2: Import im Entwicklungssystem

Nach Auswahl der zuvor exportierten XML-Datei folgt nachfolgendes Fenster.

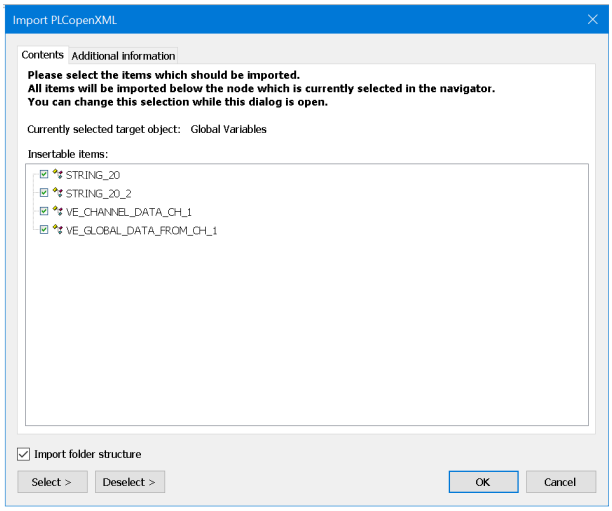

Abb. 3: Importfenster

```
TYPE VE_CHANNEL_DATA_CH_1 :
STRUCT
  sensor_ch1: ARRAY[0..1] OF LREAL;
END_STRUCT
END_TYPE

TYPE VE_GLOBAL_DATA_FROM_CH_1 :
STRUCT
 sensor: DINT;
END_STRUCT
END_TYPE
```

3.3.4 Integration in SPS-Programm

```
VAR_GLOBAL
p_ve_global : POINTER TO VE_GLOBAL_DATA_FROM_CH_1;
p_ve_channel: POINTER TO VE_CHANNEL_DATA_CH_1;
END_VAR

(* Sicherstellung, dass die internen Verwaltungsdaten initialisiert sind
*)
Hli(Start := TRUE);

IF Hli.Initialized = TRUE AND Hli.Error = FALSE THEN

  IF init_ve_ptr = TRUE THEN
    (* Zeiger auf Struktur(en) bereitstellen *)
    p_ve_chan := ADR(gpVECH[0]^.ext_var32[0]);
    p_ve_glob := ADR(gpVEGlobal^.ext_var32[0]);
  END_IF;

(* Sensorwerte der Variablen übergeben *)
p_ve_global^.sensor := LREAL_TO_DINT(vz_sensor * SENSOR_ENCODER_OUT);

END_IF
```

4 Einschränkungen bei mehrkanaligem Steuerungsaufbau

Beim Hochlauf der CNC werden für jeden NC-Kanal die “GLOBAL“ deklarierten Variablen zu bereits evtl. vorhandenen Variablen inkrementell hinzugefügt. Erst nach Abschluss des Hochlaufs steht das Layout des Speichers in der Gesamtheit fest. Die Anfangsadresse auf den gemeinsamen Speicher wird dann der SPS zur Verfügung gestellt.

- Die #EXPORT-Funktion kann nur jeweils in einem Kanal gestartet werden.
- Die Funktion kennt daher nur die in diesem Kanal mit deklarierten “GLOBAL“ Variablen. Variablen aus anderen Kanälen, die z.B. mit anderen Index-Werten belegt sind, sind unsichtbar. Sie werden daher auch nicht in der Struktur VE_GLOBAL_DATA_FROM_CH_<i> eingetragen.
- Je kanalspezifischer V.E-Liste entsteht eine eigenständige VE_GLOBAL_DATA_FROM_CH_<i> Struktur wobei

Empfehlung

Globale Variablen in mehreren Kanälen in den jeweiligen Kanälen identisch anlegen.

5 Kurzanleitung zur Nutzung der Exportfunktionalität bei V.E.- Variablen.

Vorgehensweise anhand von TwinCAT:

1. Exportieren der V.E.Variablen aus der CNC mit dem Export-Befehl #EXPORT VE[TWINCAT]
2. Öffnen der Exportdatei mit einem Editor und auf eventuelle Warnungen oder Fehler prüfen. Diese werden textuell in der Datei dargestellt.
3. Importieren der Exportdatei in das bestehende SPS Projekt
4. Zeiger auf Strukturen anlegen
   (* Nutzen der erzeugten Strukturbeschreibungen *)
   p_ve_chan_1 : POINTER TO VE_CHANNEL_DATA_CH_1;
   p_ve_glob : POINTER TO VE_GLOBAL_DATA_FROM_CH_1;
5. Einmalig die Adressen der V.E.-spezifischen HLI-Bereiche wie im Beispiel den definierten Zeigervariablen [▸ 9] zuweisen
6. Lese- und Schreibzugriffe auf Strukturen integrieren
   p_ve_glob^.VARIABLE_1 := 22; (*Schreibzugriff*)
   gl_ar_var_3 := p_ve_glob^.VARIABLE_1; (*Lesezugriff*)

6 Parameter

P-EXTV-00022	**Zeichenanzahl von Stringvariablen**
Beschreibung	Mit dem Parameter kann die zulässige Zeichenanzahl von Stringvariablen von 21 auf 128 Zeichen (jeweils inklusive Endmarke) erhöht werden. Falls die Adressen der V.E. Variablen in 24-Byte Blöcken (siehe Speicherlayout) vorgegeben sind, ist bei den 128 Byte großen Variablen vom Typ STRING zu beachten, dass sie im Speicherlayout mehrere 24-Byte Blöcke belegen und der Index entsprechend hochgezählt (vergl. Variablenarrays) werden muss.
Parameter	use_extended_string_var
Datentyp	BOOLEAN
Datenbereich	TRUE, FALSE
Dimension	----
Standardwert	FALSE
Anmerkungen	

7 Anhang

Stichwortverzeichnis

P

P-EXTV-00022 18

www.isg-stuttgart.de
support@isg-stuttgart.de