FAQ COMBIVIS studio 6

Software Engineering Guideline for IEC 61131-3 libraries and application

FAQ No.

Part	Version	Revision	Date	Status
de	6.4.0.1	001	2022-08-04	

History:

Date	Version	Comment	Author
2011_10_18	2.0.	Created for generation 6	Fischer
2011_11_09	2.1	Review and Release	Kaiser, Grabbe, Pieper, Fischer
2013_03_19	2.2	Review	CCT EUROPE
2017_11_20	6.4.0.0	- KEB Automation KG standards introduced - Library docformat restructured text introduced - Library categories chapter updated	Fischer
2022_08_04	6.4.0.1	- Changed library history style	Schneider

KEB Automation KG
Südstraße 38
32683 Barntrup, Germany
E-mail: info@keb.de
Tel. +49 5263 401-0
Fax +49 5263 401-116
www.keb.de

District Court Lemgo HRA 5649
DUNS-No. 314108728
VAT-No. DE309087075
Bank Details: Sparkasse Paderborn-Detmold
IBAN DE 19 4765 0130 0000 0060 07
BIC WELADE3L

General Partner:
Vittorio Tavella
KEB Verwaltungs-GmbH, Barntrup
District Court: Lemgo HRB 8965
Directors: Curt Bauer CMO,
Ralf Lutter COO, Vittorio Tavella CFO,
Wolfgang Wiele CTO

FAQ COMBIVIS studio 6

Inhalt

Einführung

Dieses Dokument beschreibt die Richtlinien für KEB IEC 61131-3 Bibliotheken und Applikationen, die auf der IDE KEB COMBIVIS studio 6 basieren.
Diese Richtlinien sollen ein weltweit einheitliches Erscheinungsbild von IEC 61131-3 Bibliotheken, Projekten und Applikationen realisieren.
Eine Programmierung, die sich an diesen Richtlinien orientiert, bedeutet:

- Erstellen von wiederverwendbarem IEC 61131-3-Code.
- Reduzieren Sie die Zeit für die Applikationsentwicklung und -wartung.
- Erweitern Sie die Sammlung der einsatzbereiten Funktionsbausteine für Ihren Anwendungsbereich.
- Profitieren Sie vom globalen KEB-Anwendungs-Know-how.

Allgemeine Richtlinien

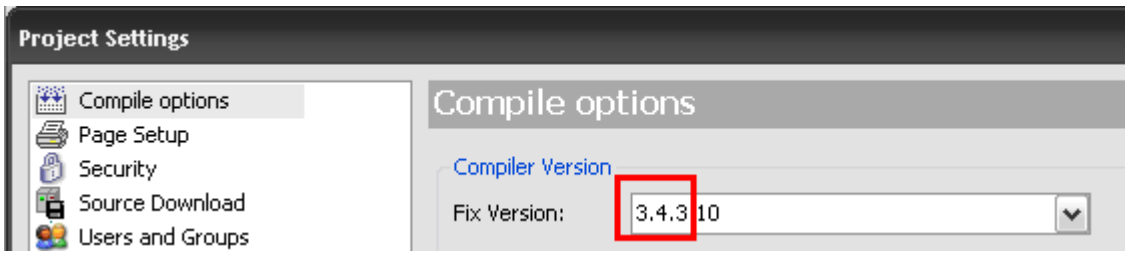
Projekt Name

Die Projektnamen werden mindestens durch den Applikationsnamen und einen Versionscode gebildet. Zusätzlich können der Firmenname und die Art der Steuerung angehängt werden.

- Der Name darf nur die Zeichen 0...9, die Buchstaben A...Z, a…z und Unterstriche enthalten
- Punkte sind nur für das Version-Tag erlaubt.

<Firma>_<Applikation>_<Zielsteuerung(optional)>_<Version>
z.B: KEB_Winder_C6C_3.4.3.15

<Version>x.x.x.x = <Compiler Version>x.x.x.x + <Implementation Version>
Die Compiler-Version kann als Versionscode angehängt werden, zusätzlich wird bei jeder Änderung die Implementierungsversion hochgezählt. Dieser Versionscode wird auch für die internen Projektinformationen und für Einträge in der Textdatei der Versionshistorie verwendet.
z.B.: 3.4.3.15 heißt: Compiler V3.4.3.10, 5th implementation.



<Zielsteuerung>
Enthält den Namen des Zielsteuerungssystems.
C6x: KEB COMBICONTROL C6 device
C6Cx: COMPACT class version x
C6Sx: SMART class version x
C6Ex: ECONOMY class version x
C6Px: PERFORMANCE class version x
P6x: P6 control unit version x
H6x: H6 control unit version x
WIN: Windows simulation device

IEC Programmiersprachen

Die folgende Tabelle zeigt, in welchen Fällen welche Sprache verwendet werden darf.

KEB Object	Programming language
State machines (in project)	ST
Function blocks, Libraries	ST
Samples	ST, FBD, CFC, SFC

Namenskonventionen

Zur Verwendung in Applikationen und neuen Bibliotheken.

- **Namen müssen immer auf Englisch geschrieben werden!**
- Erlaubte Zeichen im Allgemeinen: A…Z, a…z, 0...9, "_".
- Verwenden Sie keine unbekannten/ irreführenden Abkürzungen, siehe Abkürzungsliste. Längere, aussagekräftige Namen werden bevorzugt.

„Camel Case“ Schreibweise

Der erste **Buchstabe** von jedem **Wort** eines Basisnamens sollte ein Großbuchstabe sein, die anderen sollten klein sein (So wie “CamelCase”). Das erste Zeichen muss immer ein Großbuchstabe (A-Z) sein, mit Ausnahme von speziellen Präfixen, wie p für Zeiger. Unterstriche sind zu vermeiden.
z.B.: MyCamelCaseVariable

Die folgende Tabelle zeigt eine Übersicht von speziellen Fällen:

POU	Beispiel	Prefix
PROGRAM	**FileHandling()**; **FileHandling**.Done **HmiCom(Enable:=TRUE)**;	
FUNCTION	**CalcDiameter(Radius)**;	
FUNCTION BLOCK **(Class name)**	*Prefix KEB_ for KEB libraries (Bei internen/hilfefähigen FBs kann das Firmenpräfix weggelassen werden).* **KEB_HomingOnBlock** **MC_ReadActualPosition** **MC_MoveVelocity**	

FUNCTION BLOCK
 	
(Instance name)
	Hsp5Master: KEB_Hsp5Master;
	DataLoggerInverter: DataLogger;
	DataLoggerPlc: DataLogger;
	
	Bei bekannten FBs ist der Klassenname als Instanzpräfix möglich.
	TonWait: TON; //wait timer
	TonInit: TON; //init timer
	RtrigReset: R_trig; // Trigger positive Flanke des Rücksetzeingangs	

	Variablen, Eigenschaften, Methoden und Aktionen werden auf die gleiche Weise behandelt. ***Es wird kein Präfix verwendet. Keine Unterstriche. Beginnen Sie mit Großbuchstaben.***	
ACTION	MyPid.**ResetIntegral()**; //reset Integral part of PID controller	
PROPERTY	PropValueOld:= LoggerPitch**.LastIndex**; LoggerPitch**.LastIndex** := PropValueNew;	
METHOD	CleanUpDone := LoggerPitch**.CleanBuffer()**;	

VAR **VAR_IN_OUT** **VAR INPUT** **VAR OUTPUT**	*Correct:* *Wrong:* **MaxFileSize** max_file_size **SetVelocity** setVelo **ActPositionInc** CurPosIncre //Actual position [increments] **ActDcVoltage** actDC_Voltage, act_DC_Voltage **LogData: ARRAY [0..10] OF REAL;**	
VAR GLOBAL (Access)	*erwenden Sie immer den Namen der globalen Variablenliste, um auf globale Variablen zuzugreifen.* GVL_MOTOR.**ActCurrent**	
Loop counter	**i, j, k, l, m, n** *(one letter)*	
Pointers	**pBuffer**: POINTER TO BYTE;	**p<>**
INTERFACE	**IKebComSlave** *(Das Präfix ist groß i" ohne Unterstrich)*	**I<>**
Visualization object	**VISU_Master, VISU_Home, VISU_ErrorHandling**	**VISU_<>**
Visualization library template	**KEB_VISU_ChannelHandler** **KEB_VISU_DriveControl**	**KEB_VISU <>**

Großbuchstaben Schreibweise

Statische Definitionen wie Dateneinheitentypen und Konstanten werden immer in Großbuchstaben geschrieben. Teile von Namen werden durch einen Unterstrich "_" getrennt.
z.B.: TO_BE_CARVED_IN_STONE

POU	Beispiel	Prefix
VAR CONSTANT	**LAST_INDEX**: WORD := 20; **PI**: REAL := 3.14159; **HSP5_MAX_BUFF_CNT**: WORD := 64;	
ENUM	*Die Typdefinition besteht aus Präfix + Großbuchstaben. Mitglieder einer Enumeration werden wie Konstanten behandelt.* TYPE **T_KEB_SAMPLE_ERROR** : (NO_ERROR:= 0, //no active error TIMEOUT:=1, //timeout error INTERNAL:= 2, //internal FB error INVALID_DATA:= 3 //invalid data error); END_TYPE	**T_<>**
STRUCT	*Die Typdefinition besteht aus Präfix + Großbuchstaben. Mitglieder einer Struktur werden wie Variablen behandelt (CamelCase), da ihre Werte nicht konstant sind.* TYPE **T_LOG_ENTRY** :	**T_<>**

POU	Beispiel	Prefix

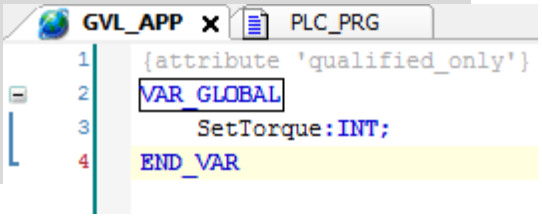
	STRUCT
	ErrorID: T_KEB_SAMPLE_ERROR; //error identification
	LogTime: DWORD; //error time stamp
	END_STRUCT
	END_TYPE
	
	Instanzen von Strukturen werden wie Variablen behandelt (CamelCase).
	
	LogData: ARRAY [0..99] OF **T_LOG_ENTRY;**	

Global variable list	**GVL_GATEWAY**	**GVL_**

Hinweis: Für ENUMs und GVLs ist das Pragma {attribute 'qualified_only'} oben in der Erklärung zu verwenden, um den Zugriff über den GVL-Namen zu erzwingen.

```
{attribute 'qualified_only'}
VAR_GLOBAL
        SetTorque :REAL;
END_VAR

Sample instruction: GVL_APP.SetTorque := 100;
```



Empfohlene numerische Datentypen

Um den Bedarf an Typkonvertierungen zu minimieren, verwenden Sie die empfohlenen Basisdatentypen.

z.B.:
Nutze BYTE anstelle von UNSIGNED SHORT INT (USINT).
Nutze WORD anstelle von UNSIGNED INT (UINT).
Nutze DWORD anstelle von UNSIGNED DINT (UDINT).

Data type	Lower limit	Upper limit	Size
BOOL	**FALSE**	**TRUE**	**8 Bit**
BYTE	**0**	**255**	**8 Bit**
WORD	**0**	**65 535**	**16 Bit**
DWORD	**0**	**4 294 967 295**	**32 Bit**
LWORD	**0**	**2^{64}-1**	**64 Bit**
SINT	**-128**	**127**	**8 Bit**
INT	**-32 768**	**32 767**	**16 Bit**
DINT	**-2 147 483 648**	**2 147 483 647**	**32 Bit**
LINT	**-2^{63}**	**2^{63}-1**	**64 Bit**
REAL	**1.175494351e-38 to 3.402823466e+38**		**32 Bit**
LREAL	**2.2250738585072014e-308 to 1.7976931348623158e+308**		**64 Bit**

Die Benennung von POUs & Variablen Namen

Es sollte für jede variable ein aussagekräftiger Name genutzt werden.

Type	Regel	Beispiel
Method **Action**	*Methoden starten mit einen Verb*	GetPosition() ResetError()
BOOL	*Boolsche Namen beschreiben die Aktion/den Status des TRUE-Zustandes.*	EnableVoltage //switch StartHoming VoltageEnabled //status TorqueControlActive InitDone

Standard Bezeichnung

Im Allgemeinen sind aussagekräftige Namen zu verwenden.
Für allgemeine Begriffe ist eine einheitliche Abkürzung oder ein einheitlicher Begriff zu verwenden.

Begriff	Abkürzung
Applikation	
Actual Value	Act
Setpoint	Set
Value from last cycle	Old
Minimum	Min
Maximum	Max
Position	Pos
Increments	Inc
Velocity	Vel
Frequency	Frq
Acceleration	Acc
Deceleration	Dec
Absolute	Abs
Relative	Rel

Begriff	Abkürzung
Kommunikation	
Process data	PD
Address	Addr
Request	Req
Response	Rsp
Index	Idx
Buffer	Buff
Count	Cnt
Memory	Mem
Length	Len
Identifier	Id
Source	Src
Destination	Dest

Formatierung

Pro Ebene der Funktionstiefe wird ein Einzug von einem Tabulator verwendet.

Beispiel einer IF-Anweisung

```
IF (x>0) THEN
  BuffCnt := BuffCnt + BuffOffset;  //Count up
ELSIF (x<0) THEN
  BuffCnt := BuffCnt - BuffOffset; //Count down
ELSE
  Error:=TRUE;       //Set error
END_IF
```

Beispiel einer FOR-Anweisung

```
FOR i:= 1 TO AXIS_CNT DO
  DriveCtrl[i](Start:=TRUE);  //Start all drives
END_FOR
```

Beispiel einer CASE-Anweisung

```
//main state machine
CASE State OF

  STATE_NOT_ENABLED: //FB not active, trigger Execute input to start action
  (*=================================================================*)
    IF Execute THEN
      State:=STATE_BUSY; //switch to busy state
      Busy:=TRUE;
      IsBusy(); //work on the FB task
    END_IF

  STATE_BUSY://FB is busy
  (*=================================================================*)
    IsBusy(); //work on the FB task

  STATE_DONE://action sucessfully done, FB is idle, reset Execute input to reset outputs
  (*=================================================================*)
    IF NOT Execute THEN
      ResetOutputs();
    END_IF

  STATE_ERROR: // error occured, FB is idle, reset Execute input to reset outputs
  (*=================================================================*)
    IF NOT Execute THEN
      ResetOutputs();
    END_IF

  ELSE //unknown state, should not happen
  (*=================================================================*)
    SetError(ErrID:=999); //set unknown error

END_CASE
```

Dokumentation/ Kommentare

Allgemeine Regeln

- **Kommentare müssen immer auf Englisch verfasst sein!**
- Jede öffentliche Variablendeklaration benötigt einen Kommentar!
- Kommentare beschreiben die Funktion eines Bausteins / einer Codezeile, nicht die Anweisung!

Gutes Beispiel:	IF (x > 100) THEN
	  x := 0;
	END_IF
		// If max. buffer index reached
		// reset buffer index

Schlechtes Beispiel:	IF (x > 100) THEN
	  x := 0;
	END_IF
		// If x > 100
		// reset x to 0

- Kommentare über mehr als eine Zeile sind erlaubt (`(*  *)`).
- Löschen Sie nicht verwendeten, nicht kommentierten Code für mehr Lesbarkeit.
- Geben Sie detaillierte Informationen, wenn der Quellcode keine Standardlösung ist.
- z.B: //Achtung, dies ist ein Workaround für Fall x; //Dies ist ein Testfall.
- Geben Sie detaillierte Informationen, wenn etwas fehlt, z.B: //To Do: Überprüfung auf Grenzwerte

- **Tipp: Es können Message-Pragmas erstellt werden, um den Benutzer während der Kompilierung über kritische Dinge zu informieren (siehe CODESYS Online-Dokumentation).**

Pragma	Message type
{text 'textstring'}	**Text**: The specified **textstring** will be displayed.
{info 'textstring'}	**Information** The specified **textstring** will be displayed.
{warning digit 'textstring'}	**Warning** The specified **textstring** will be displayed.
{error 'textstring'}	**Error** The specified textstring will be displayed.

Funktionsblockregeln

- Über dem ersten Schlüsselwort muss eine englische POU-Beschreibung hinzugefügt werden (z.B. FUNCTION BLOCK, siehe Beispielbeschreibung unten). Diese Baustein-Dokumentation steht den Endanwendern in der Online-Hilfe und in der Ansicht Bibliotheksverwalter zur Verfügung.
- Sie können für jeden Baustein nach dem ersten Schlüsselwort eine Änderungshistorie einfügen. Dieser Text wird NICHT in der Online-Hilfe und in der Ansicht des Bibliotheksmanagers veröffentlicht. (Siehe Beispiel-Beschreibung unten)
- Autor: Vollständiger Name muss verwendet werden (z.B.: Hugo Mueller).
- Das Datumsformat ist: Jahr_Monat_Tag (z.B.: 2011_02_03).

Spezielle Dokumentationskommandos

Die Beschreibungssprache reStructuredText kann verwendet werden, um eine gut formatierte Dokumentation innerhalb des Quellcodes zu erstellen. Die Dokumentation selbst wird automatisch generiert, wenn die Bibliothek in die Ide importiert wird.
Zeilenumbrüche werden automatisch erkannt. Historische Befehle wie ~~
, <p/>~~ sind zu vermeiden.
Die Funktion muss in den Projekteigenschaften für jede Bibliothek aktiviert sein, siehe 3.2.

Beispiel Funktionsblockbeschreibung

```
(*
This function block (FB) controls a KEB F5 Drive (A-board) in different
drive modes without softmotion.

Drivemodes: VELOCITY, POSITIONINGABSOLUTE,
 POSITIONINGRELATIVE, SETPOSITION, HOMING
*)

FUNCTION_BLOCK KEB_DriveCtrl
(*
LibVersion      Date                Author              Topic
3.4.0.0  2010_06_09          Hugo Mueller        generated
3.4.0.1  2011_02_03          Hugo Mueller        New input x
*)

VAR_INPUT
        Execute:BOOL; //Positive edge enables FB
END_VAR

VAR_OUTPUT
        Done:BOOL; //TRUE: Successfully done
        Busy:BOOL;     //TRUE: FB is working
        Error:BOOL;    //TRUE: Error occured
        ErrorID:INT;   //Shows error ID
END_VAR
```

Musterdokumentation

Start des FB

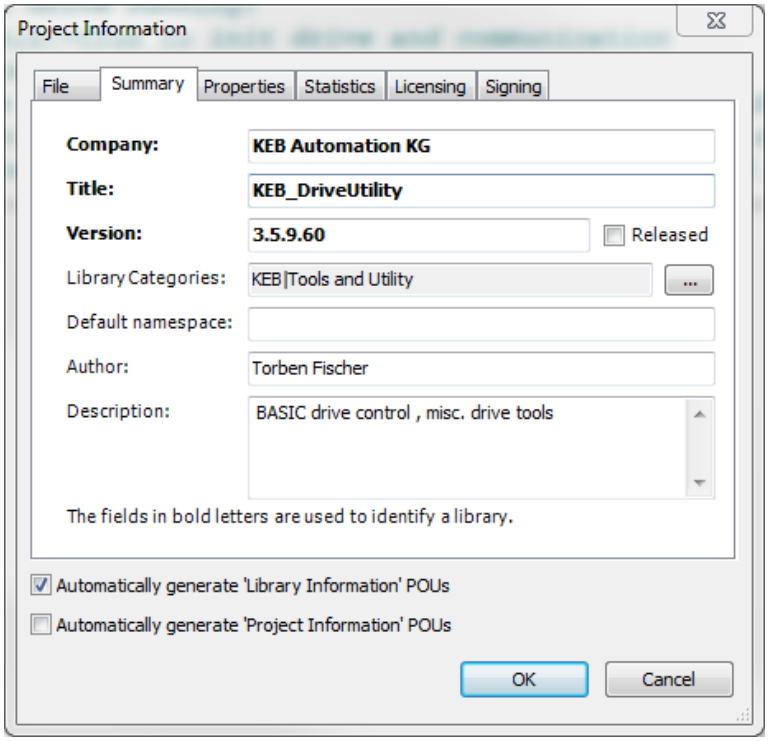
Versteckte Versionshistorie

Beginn des Deklarationsteils

Richtlinien für Bibliotheken

Projekt Informationen

Für Bibliotheken sind einige strenge Regeln für die Projektinformationen zu beachten.



Firma: Verwenden Sie den richtigen Begriff des Unternehmens, z.B. „KEB Automation KG”

Titel: Der Bibliotheksname beginnt mit "KEB_" + CamelCase Notation.

Version: Verwenden Sie den Versionscode der aktuellen Compiler-Version. Die letzte Zahl wird für jede neu kompilierte Bibliotheksversion (sog. Implementierungsversion) hochgezählt.
Vergessen Sie nicht, die Version jedes Mal hochzuzählen, wenn eine neue kompilierte Bibliothek erstellt wird!

Bibliotheks Kategorien: Die Kategorien sind in der Datei "KebLibraryCategory.libcat.xml" festgelegt (siehe IDE-Installationspfad); z.B....: "KEB|Werkzeuge und Dienstprogramm"

Bibliothekskategorien sind bei der KEB Automation KG, Barntrup, anzufordern.
Die Verwendung der Kopfkategorie "KEB|" ist verboten, eine Bibliothek muss immer in eine Unterkategorie gestellt werden.

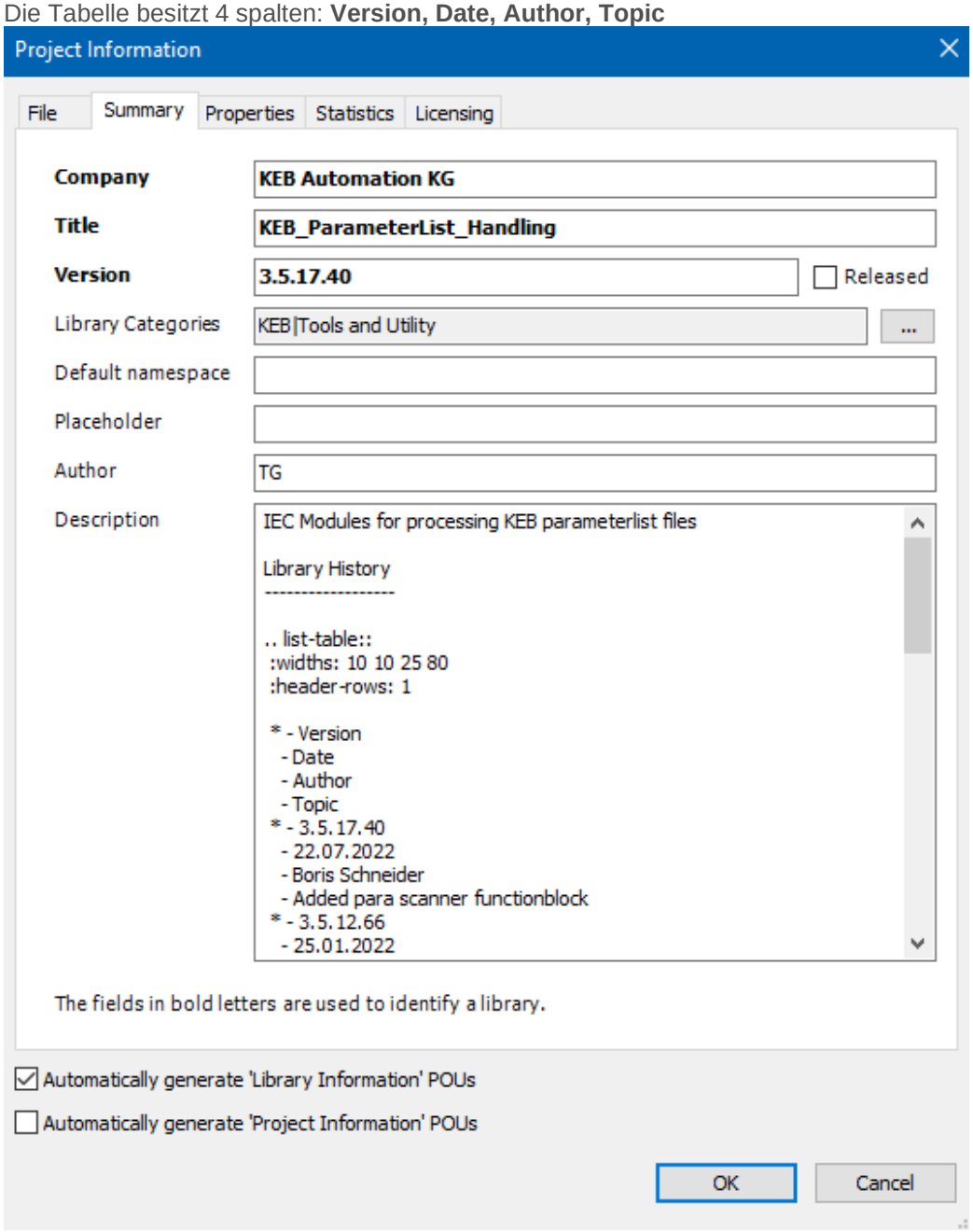
Autor: Autor der Bibliothek, voller Name.

Beschreibung: Kurze Bibliotheksbeschreibung, wie in der Textdatei der Versionshistorie.

Automatische Generierung von POUs für Bibliotheksinformationen: Option muss aktiviert sein.

Library Historie

Um alle Bibliotheks-/Projektänderungen zu dokumentieren, wird eine Tabelle mit der Überschrift „Library History“ innerhalb der Projekt Informationen/Description abgelegt. Dazu wird stehts das Restruct Objekt „list-table“ verwendet.
Die Tabelle besitzt 4 spalten: **Version, Date, Author, Topic**



Beispiel der Darstellung im Library manager

Library History

Version	Date	Author	Topic
3.5.17.40	22.07.2022	Boris Schneider	Added para scanner functionblock
3.5.12.66	25.01.2022	Boris Schneider	Fixed a bug that caused the first parameter not to be downloaded correctly
3.5.12.65	26.11.2021	Boris Schneider	Fixed a limitation for 64-Bit compatibility
3.5.12.64	04.08.2020	Boris Schneider	#38052 Fixed a bug regarding reinit of the lists
3.5.12.63	29.05.2020	Boris Schneider	#37706 Changed download rules for WA parameter to follow the dw5 guideline #37286 Fixed reinit issue #37019 Fixed issue with not resetable errors #37018 Added repeat/ignore counter to output #36350 encrypted library source files

Beispiel für Library History text

```
This is the sample library with no function at all


Library History
----------------

.. list-table::
 :widths: 10 10 25 80
 :header-rows: 1

 * - Version
   - Date
   - Author
   - Topic
 * - 3.5.17.40
   - 22.07.2022
   - Boris Schneider
   - Added para scanner functionblock
* - 3.5.12.64
   - 04.08.2020
   - Boris Schneider
   - #38052 Fixed a bug regarding reinit of the lists
 * - 3.5.12.63
   - 29.05.2020
   - Boris Schneider
   - | #37706 Changed download rules for WA parameter to follow the dw5 guidel
     | #37286 Fixed reinit issue
     | #37019 Fixed issue with not resetable errors
     | #37018 Added repeat/ignore counter to output
     | #36350 encrypted library source files
```

Library Bescheidungstext

Überschrift für die Historie

Definition des ReStruct text objects

Header der Spalten

Tabellen inhalt (standard)

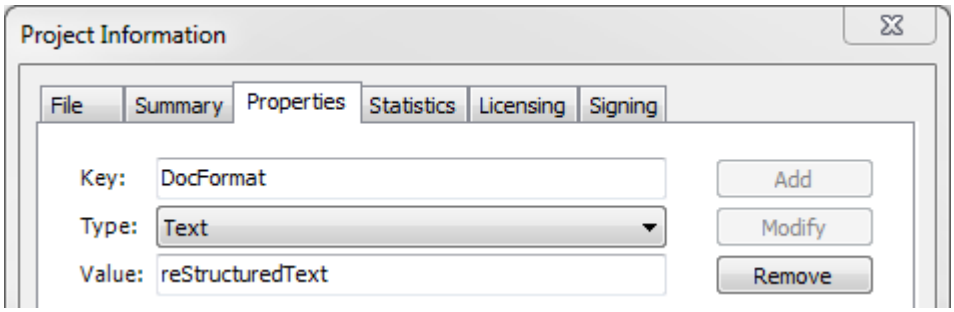
Tabellen inhalt (ticket referenz)

Tabellen inhalt (multible lines)

Eigenschaften von Projektinformationen

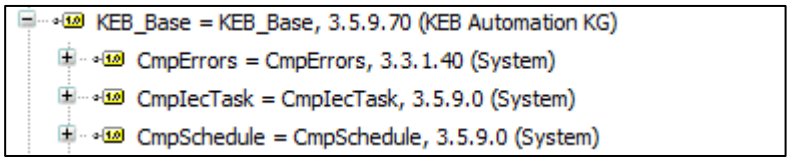
Die Bibliotheksdokumentation muss auf " restructuredText " gesetzt werden, indem der folgende Schlüssel hinzugefügt wird:

Key: DocFormat
Type: Text
Value: reStructuredText

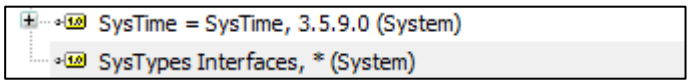


Bibliotheksmanager

Es ist zwingend erforderlich, alle Bibliotheken mit **Platzhaltern** hinzuzufügen!
Diese werden einen gültigen Satz von Bibliotheken anhängen, abhängig von der Compiler-Version und dem Gerät.



Die Verwendung des asterix- Platzhalters (immer die neueste installierte Bibliothek.) ist nicht erlaubt, außer bei Interface-Bibliotheken.

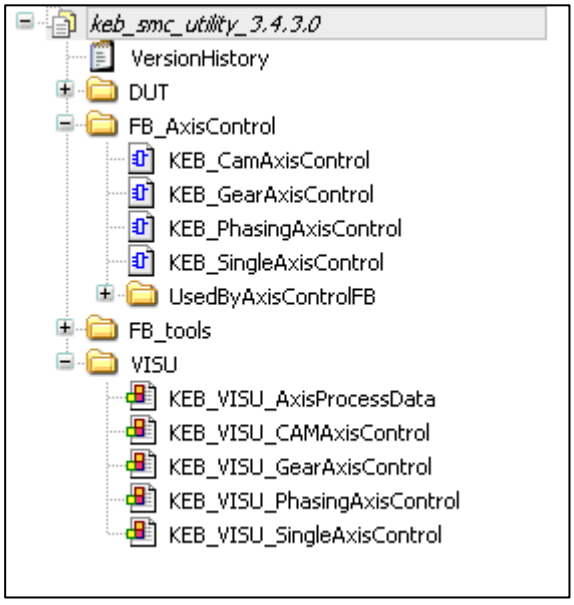


POU Organisation

Verwenden Sie die **POU-Ansicht** (Alt+0) (anstelle der "Navigator-Ansicht"), um alle POUs übersichtlich zu organisieren.

Verwenden Sie Unterordner, um zusammengehörige Elemente zu summieren.
Unterordner verwenden, wenn es mehr als einen Baustein eines beliebigen Typs gibt.
Dies ist wichtig für die automatische Online-Dokumentation der Bibliothek!

Beispielunterordner:
DUT : Data Unit Types
FB: Function Blocks
VISU: Visualization Templates
Help/ Intern/ BaseHilfe-POUs, die von anderen POUs verwendet werden, sind für den Endbenutzer nicht wichtig.



Visualisierungsvorlagen

Namen von Visualisierungsvorlagen müssen beginnen mit:

<KEB_VISU_>
z.B.: KEB_VISU_MotorMonitoring

Bibliothekswarnungen

Alle Warnungen müssen beseitigt werden. Wenn eine Warnung nicht durch Quelltextkorrektur behoben werden kann, kann stattdessen ein Pragma verwendet werden.
z.B.: {warning disable C0371} … {warning restore C0371

Anhang

Praxistipps

Achten Sie auf Indexfehler in Schleifen und Arrays: Der Index sollte >=0 und <= limit sein.
Achten Sie auf die Division durch Null: Ein solcher Fall erfordert eine Fehlerbehandlung oder Einschränkung.
Überprüfen Sie jede Variable, ob ein Überlauf/ Unterlauf möglich ist.
Überprüfen Sie immer die Rückmeldung der Bausteine auf Fehler/Zeitüberschreitungen.
Zeitkritische Prozesse: Eigentlich sollte jede Zeile Code so klein wie möglich gehalten werden.
Kontrollstrukturen wie IF/ THEN/ ELSE oder CASE OF sollten einen Zweig haben, wenn eine Situation eintritt, die nicht geplant war: Z.B. CASE OF Anweisungen sollten einen ELSE-Zweig haben, der eine Fehlercodemeldung enthält.
Keine Mehrzweckfunktionen/-prozeduren, lieber kleinere, separate Unterprogramme.
Aufgabenverwaltung: • Halten Sie es einfach, so wenig Aufgaben wie möglich. • Überprüfen Sie, ob die Zykluszeit nach einem ganzen Anwendungszyklus überläuft. • Wählen Sie eine geeignete Zykluszeit für den technischen Prozess, stellen Sie sicher, dass die CPU genügend Leerlaufzeit hat, um ein deterministisches Verhalten zu gewährleisten.
Benennung: • Verwenden Sie einen Objektnamen nicht zweimal, wenn das Zugriffsrecht global ist. • Verwenden Sie keine Objektnamen, die falsch sein können, vermeiden Sie Kombinationen von ähnlichen Buchstaben und Zahlen, z.B. IO, 1O, 1O, i0. • Verwenden Sie keine unaussprechlichen Namen. Ein längerer sinnvoller Name ist immer die bessere Idee. Z.B. ist ResetInverter ein besserer Name als RstInv.
Programmierung: Vermeiden Sie "Magische Zahlen": Deklarieren Sie eine konstante Variable, anstatt eine statische Zahl irgendwo im Quellcode zu verwenden. Dies erleichtert die Wartung erheblich, z.B., wenn sich die Anzahl in der Zukunft ändert. **Schlecht:** FOR i := 1 TO 5 DO (drive[i].SetSpeed := 100); END_IF **Gut:** LAST_DRIVE: INT := 5; SET_SPEED: INT:= 100; FOR i := 1 TO LAST_DRIVE DO (drive[i].SetSpeed := SET_SPEED); END_IF

Nützliche Pragmas

Eine vollständige Liste finden Sie in der Online-Hilfe.

Pragma/ Syntax	Beschreibung
{text 'textstring'}	Text: The specified textstring will be displayed.
{info 'textstring'}	Information The specified textstring will be displayed.
{warning digit 'textstring'}	Warning The specified textstring will be displayed.
{error 'textstring'}	Error The specified textstring will be displayed.
{attribute 'hide'}	Verhindern Sie, dass Variablen, Methoden usw. oder sogar ganze Signaturen innerhalb der Funktionalität "Listenkomponenten", des Eingabeassistenten oder des Deklarationsteils im Online-Modus angezeigt werden. Nur die Variable nach dem Pragma wird ausgeblendet. Verwenden Sie dieses Attribut z.B. für private Methoden.
{attribute 'hide_all_locals'}	Verhindern Sie, dass alle lokalen Variablen, Methoden usw. oder auch ganze Signaturen innerhalb der Funktionalität "Listenkomponenten", des Eingabeassistenten oder des Deklarationsteils im Online-Modus angezeigt werden.
{attribute 'qualified_only'}	Weisen Sie einer globalen Variablenliste oder Aufzählungsdeklaration eine Zuweisung vorangestellt zu. Auf die Variablen dieser Liste kann nur über den globalen Variablennamen/Aufzählungsnamen zugegriffen werden, z.B. GVL_MOTOR.ActCurrent. Dies ist nützlich, um Namensfehlanpassungen mit lokalen Variablen zu vermeiden.
{warning disable C0XYZ} … {warning restore C0XYZ	Der Code zwischen diesen Pragmas löst die Warnung XYZ nicht aus.

Disclaimer

KEB Automation KG reserves the right to change/adapt specifications and technical data without prior notification. The safety and warning reference specified in this manual is not exhaustive. Although the manual and the information contained in it is made with care, KEB does not accept responsibility for misprint or other errors or resulting damages. The marks and product names are trademarks or registered trademarks of the respective title owners.

The information contained in the technical documentation, as well as any user-specific advice in verbal or in written form are made to the best of our knowledge and information about the application. However, they are considered for information only without responsibility. This also applies to any violation of industrial property rights of a third-party.

Inspection of our units in view of their suitability for the intended use must be done generally by the user. Inspections are particular necessary, if changes are executed, which serve for the further development or adaption of our products to the applications (hardware, software or download lists). Inspections must be repeated completely, even if only parts of hardware, software or download lists are modified.

Application and use of our units in the target products is outside of our control and therefore lies exclusively in the area of responsibility of the user.

KEB Automation KG
Südstraße 38 • D-32683 Barntrup
fon: +49 5263 401-0 • fax: +49 5263 401-116
net: www.keb.de • mail: info@keb.de