

Praxisbeispiel / Application Note

**Kuhnke PLCs- Persistente Daten
mit dem CODESYS PersistenceManager**

E 854 DE

28.03.2019

Inhaltsverzeichnis

1 Vorwort	3
1.1 Impressum	3
1.1.1 Kontaktdaten	3
1.1.2 Versionshistorie	3
2 Systembeschreibung	Fehler! Textmarke nicht definiert.
3 Produktbeschreibung	Fehler! Textmarke nicht definiert.
4 Anhang	12
4.1 Sales & Service	12
4.1.1 Stammwerk Malente	Fehler! Textmarke nicht definiert.

1 Vorwort

1.1 Impressum

1.1.1 Kontaktdaten

Kendrion Kuhnke Automation GmbH
Industrial Control Systems
Lütjenburger Straße 101
D-23714 Malente, Deutschland

Tel. +49 4523 402-0
Fax +49 4523 402-201

E-Mail sales-ics@kendrion.com
Internet www.KUHNKE.de

1.1.2 Versionshistorie

Versionshistorie

Datum	Art	Kommentare / Änderungen
		Ursprungsversion

2 Allgemein

2.1 Gültigkeitsbereich

Steuerungssystem mit CODESYS SPS

2.2 Systemvoraussetzungen

Mindestens CODESYS Version 3.5 SP8

2.3 Beschreibung

Einige Steuerungen unterstützen nicht die Deklaration von remanenten Variablen über die Schlüsselworte „RETAIN“, „PERSISTENT“ bzw. Kombinationen aus diesen.

Zur Speicherung remanenter Variablen kann der Persistence Manager aus dem CODESYS Applikation Composer verwendet werden.

Allgemeine Informationen zum Persistence Manager finden Sie in der CODESYS Hilfe unter:

https://help.codesys.com/webapp/f_application_composer_persistence_manager;product=core_Application_Composer;version=3.5.14.0

Im folgende sind die Unterschiede zum Mechanismus der Deklaration über die Schlüsselworte (PERSISTENT) aufgeführt:

- Die persistenten Daten werden in einer externen Datei gespeichert.
- Die persistenten Daten können zwischen Projekten ausgetauscht werden.
- Persistente Variablen können aus der Applikation gelöscht, oder neue hinzugefügt werden, ohne dass die persistenten Werte verloren gehen.
- Daten, welche durch den Persistence Manager erzeugt wurden, können mit externen Editoren verändert werden. (beispielsweise mit Notepad).

3 Quick Start Guide

3.1 Arbeiten mit dem CODESYS Persistence Manager

3.1.1 Ansicht „Module“ aktivieren

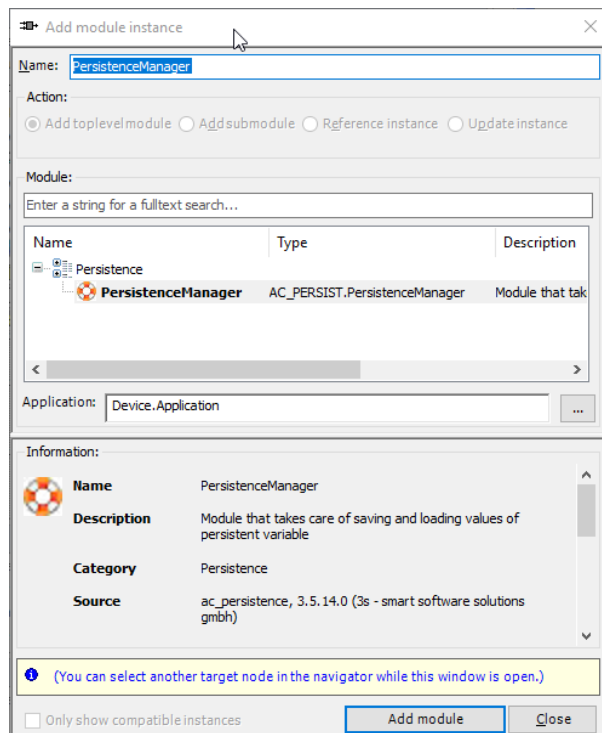
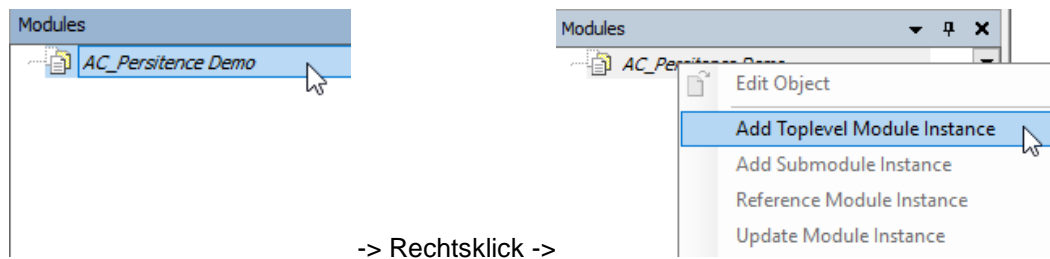
Zur Verwendung des Persistence Manager muss in CODESYS die Ansicht „Module“ aktiviert sein. Wählen Sie dazu im Menü „Ansicht“ -> „Module“ aus

3.1.2 Modulbibliothek hinzufügen

Wählen Sie dazu im Menü „Composer“ -> „Modulbibliothek“ zum Projekt hinzufügen“

3.1.3 Persistence Manager hinzufügen

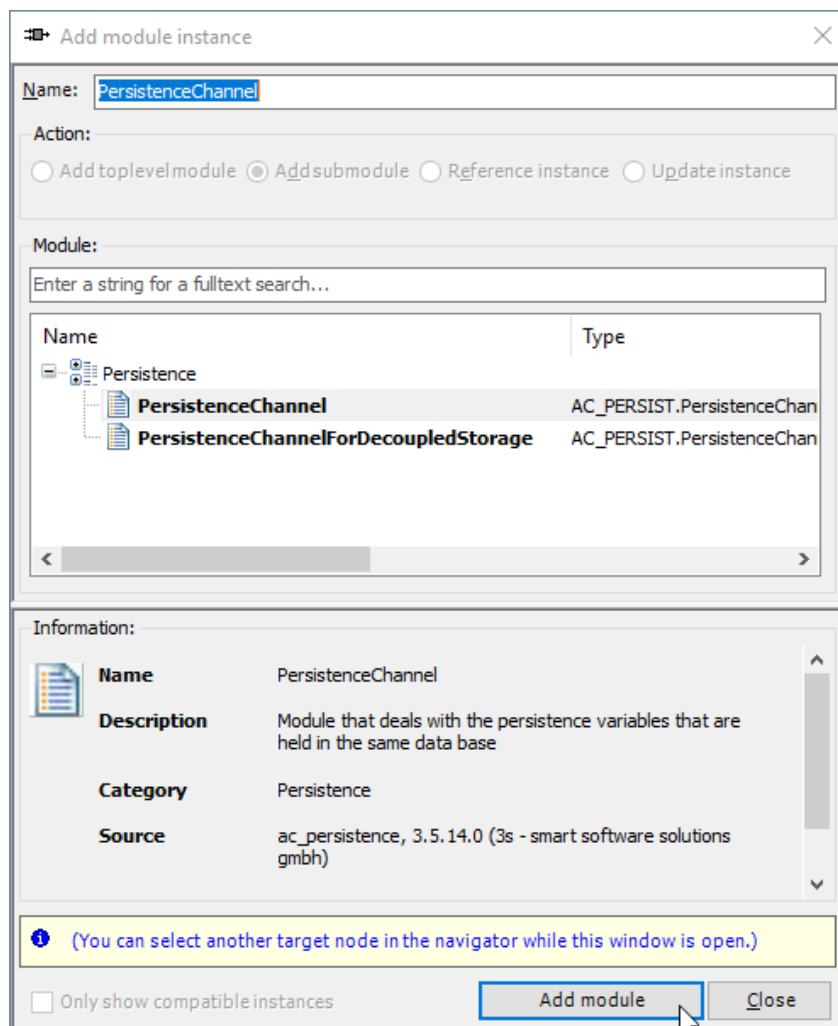
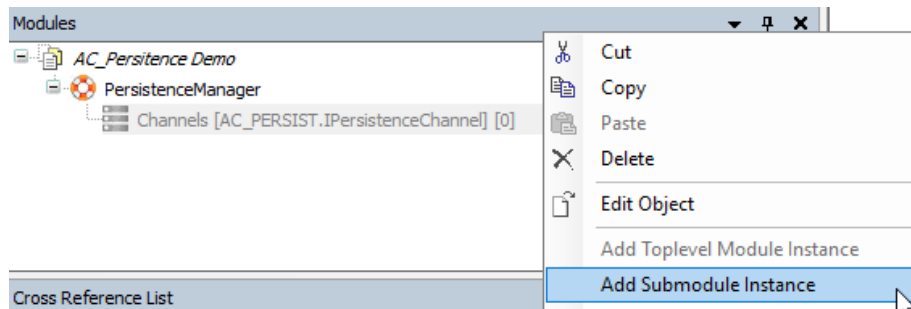
Fügen Sie unter Module den „PersistenceManager“ als Toplevelinstanz hinzu.



Der Name kann bei Bedarf angepasst werden. Klicken Sie auf „Modul hinzufügen“.

3.1.4 Persistenzkanal definieren

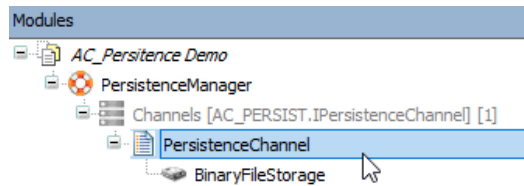
Definieren Sie nun unter dem PersistenceManager einer Persistenzkanal, in dem Sie mit einem Rechtsklick auf „Channels“ klicken und im Kontextmenü „Submodulinstantz hinzufügen“ wählen



Der Name kann bei Bedarf angepasst werden. Wenn Sie verschieden Persistenz Kanäle verwenden, empfehlen wir einen sprechenden Namen zu verwenden. Klicken Sie auf „Modul hinzufügen“.

3.1.5 Parametrierung des Persistenzkanals

Um den Persistenzkanal zu Parametrieren, klicken Sie doppelt auf den gewünschten Eintrag in der Modulansicht:



Das Modul wird im Bearbeitungsbereich geöffnet. Im Tab „Parameter“ können Sie die Einstellung an Ihre Erfordernisse anpassen.

Default Parameter

Group / Parameter	Type	Value	Description
tPeriodicSaving	TIME	TIME#60m0s0ms	time after which the variables are stored (0: periodic saving off)
xSaveOnChange	BOOL	FALSE	TRUE: permanently compare old and actual values and save when different
xReadVarsDuringInit	BOOL	FALSE	TRUE: read the persistent variables during initialization of application; FALSE: read variable values during first cycle
xCompressTags	BOOL	TRUE	TRUE: compress variable tags
xConsistentCopyInHighPrioTask	BOOL	FALSE	TRUE: persistent variables are copied in high priority task
xConvertVarsWithDifferentType	BOOL	TRUE	TRUE: if types of stored and actual variable are different, try to convert stored value
xIntegrityCheckBeforeReading	BOOL	TRUE	TRUE: do an integrity check of data base
xSeparateArchivePerToplevelInstance	BOOL	FALSE	TRUE: generate a separate archive for each toplevel instance
xMakeDataCRCConsistencyCheck	BOOL	FALSE	TRUE: a CRC is calculated before and after the saving process, whereas both CRC have to match for a successful saving
uiSavingRetriesIfCRCConsistencyCheckFails	UINT	0	If xMakeDataCRCConsistencyCheck is TRUE, this value indicates how often saving is retried if a CRC mismatch was detected

Empfohlene Änderungen:

Group / Parameter	Type	Value	Description
tPeriodicSaving	TIME	TIME#60m0s0ms	time after which the variables are stored (0: periodic saving off)
xSaveOnChange	BOOL	TRUE	TRUE: permanently compare old and actual values and save when different
xReadVarsDuringInit	BOOL	TRUE	TRUE: read the persistent variables during initialization of application; FALSE: read variable values during first cycle
xCompressTags	BOOL	TRUE	TRUE: compress variable tags
xConsistentCopyInHighPrioTask	BOOL	FALSE	TRUE: persistent variables are copied in high priority task
xConvertVarsWithDifferentType	BOOL	TRUE	TRUE: if types of stored and actual variable are different, try to convert stored value
xIntegrityCheckBeforeReading	BOOL	TRUE	TRUE: do an integrity check of data base
xSeparateArchivePerToplevelInstance	BOOL	FALSE	TRUE: generate a separate archive for each toplevel instance
xMakeDataCRCConsistencyCheck	BOOL	FALSE	TRUE: a CRC is calculated before and after the saving process, whereas both CRC have to match for a successful saving
uiSavingRetriesIfCRCConsistencyCheckFails	UINT	0	If xMakeDataCRCConsistencyCheck is TRUE, this value indicates how often saving is retried if a CRC mismatch was detected

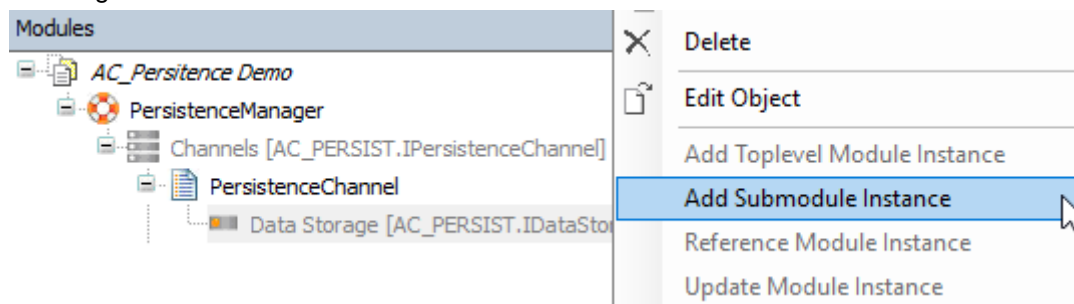


Information

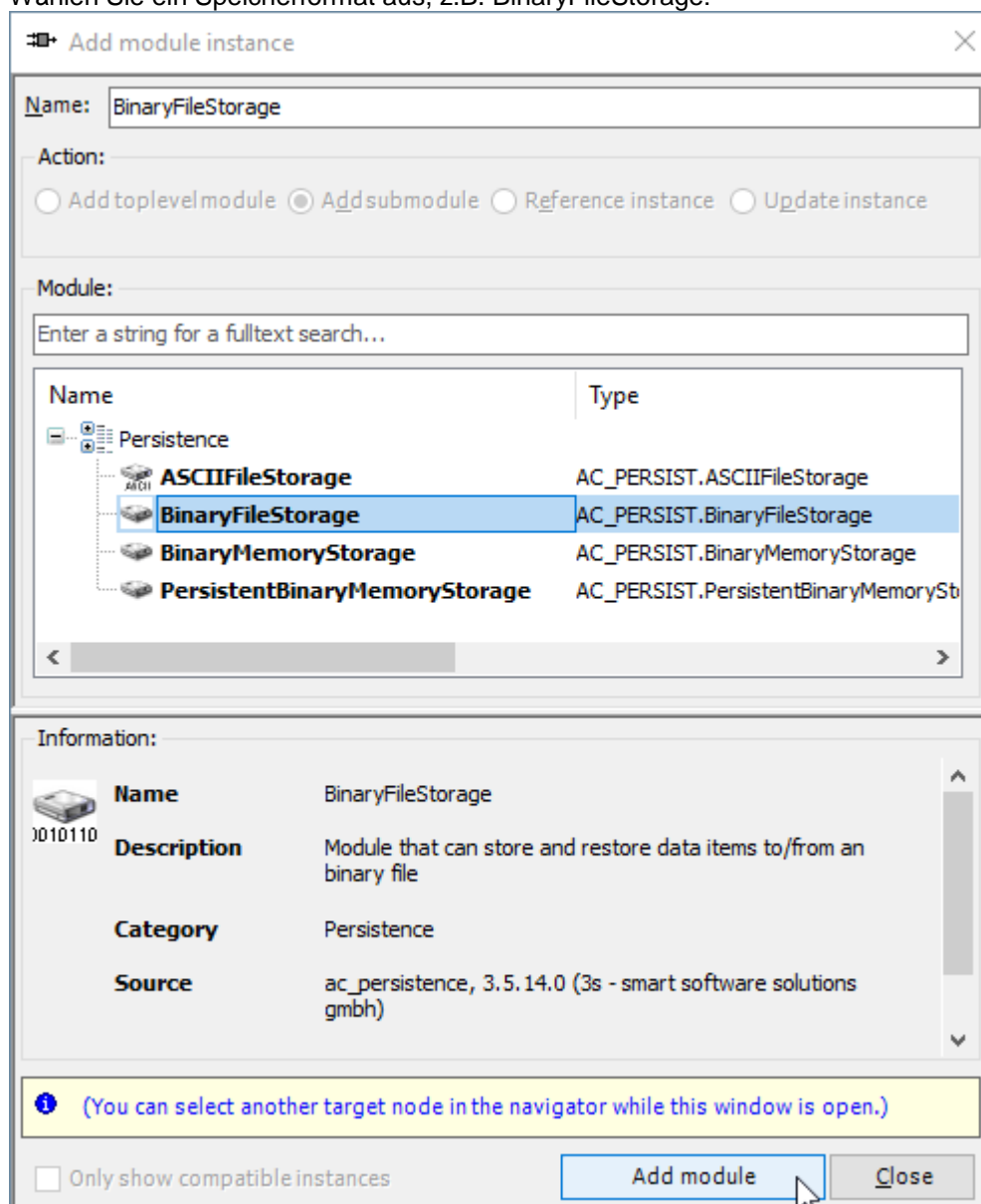
xSaveOnChange: Verwenden Sie diese Einstellung mit Bedacht. Bei jeder Änderung erfolgt ein Schreibzugriff auf den Flash-Speicher.

3.1.6 Datenspeicher konfigurieren

Definieren Sie im Submodul Data Storage das gewünschte Speicherformat der Persistenten Variablen, indem Sie mit einem Rechtsklick auf „Data Storage“ klicken und im Kontextmenü „Submodulinanz hinzufügen“ wählen.



Wählen Sie ein Speicherformat aus, z.B. BinaryFileStorage:



3.1.7 Variablendeklaration

Setzen Sie vor jeder Variablen, die Remanent sein sollen folgendes Attribut:

```
{attribute 'ac_persist' := 'PersistenceChannel'}
```

Der Name, hier 'PersistenceChannel' muss mit dem unter 3. Definierten Persistenzkanal übereinstimmen.

Beispieldeklaration einer Variablen:

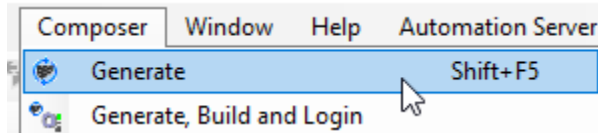
```
VAR_GLOBAL
  {attribute 'ac_persist' := 'PersistenceChannel'}
  uiStartupCounter: UINT; // Counts the machine startups
END_VAR
```

Damit die Variable dem Persistenzkanal hinzugefügt wird, muss diese auch im Projekt verwendet werden.

Die Startups können z.B. im PLC_PRG wie folgt erfasst werden:

```
IF NOT xInit THEN
  uiStartupCounter := uiStartupCounter + 1;
  xInit := TRUE;
END_IF
```

Nun muss der Code erzeugt werden, Menü Composer -> Erzeugen



Information

Generell ist auf Systemen mit Massenspeicher zu hinterfragen, wie sinnvoll die Verwendung großer Mengen von Retain-Daten ist.

3.1.8 Persistente Variablen als Datenstruktur

Wir empfehlen bei der Verwendung des PersistenceManager für die persistenten Daten eine Datenstruktur anzulegen. Sie müssen dann das Attribut {attribute 'ac_persist' := 'PersistenceChannel'} nur einmal bei der Variablendeklaration setzen, damit alle Daten aus der Datenstruktur remanent sind. Ein Beispielprojekt stellen wir Ihnen auf Anfrage gerne zur Verfügung.

3.1.8.1 Definition einer Strukturvariable

Fügen Sie mit einem Rechtsklick auf die Applikation -> Objekte hinzufügen -> DUT eine Struktur hinzu.

Nachfolgend finden Sie ein Beispiel zum möglichen Inhalt der Datenstruktur:

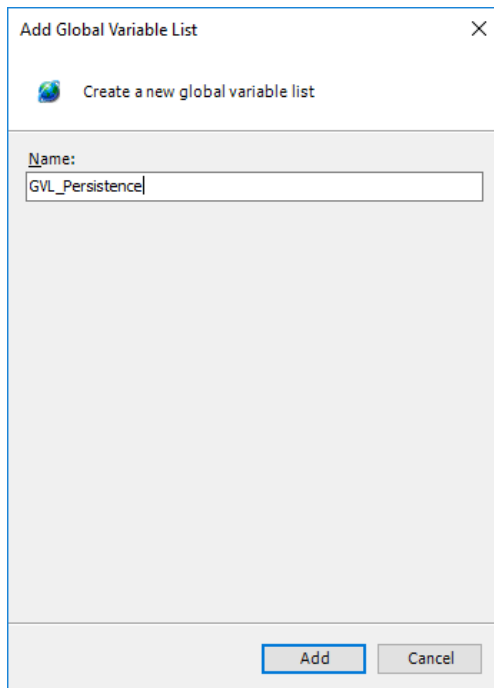
```

TYPE tPersistence :
STRUCT
    uiCounter: UINT; // Machine startup counter
    uiActState: UINT; // Actual machine state
    xModeAutomatic: BOOL; // Automatic mode
    xModeManual: BOOL; // Manual mode
END_STRUCT
END_TYPE

```

3.1.8.2 Erstellen einer globalen Variablenliste

Fügen Sie mit einem Rechtsklick auf die Applikation -> Objekte hinzufügen -> Globale Variablenliste eine Variablenliste für die remanenten Variablen hinzu.



3.1.8.3 Deklaration der remanenten Datenstruktur

Erstellen Sie eine Deklaration für die Datenstruktur.

```
{attribute 'qualified_only'}  
VAR_GLOBAL  
    {attribute 'ac_persist' := 'PersistenceChannel'}  
    Persistence: tPersistence;  
END_VAR
```

3.1.8.4 Verwendung der remanenten Variablen

Die remanenten Variablen können wie folgt in Programmbausteinen verwendet werden:

```
IF NOT xInit THEN  
    // Counts the machine startups  
    GVL_Persistence.Persistence.uiCounter := GVL_Persistence.Persistence.uiCounter + 1;  
    xInit := TRUE;  
END_IF  
  
IF NOT GVL_Persistence.Persistence.xModeAutomatic AND NOT GVL_Persistence.Persistence.xModeManual  
THEN  
    // If any mode is active, set the machine in manual mode  
    GVL_Persistence.Persistence.xModeManual := TRUE;  
END_IF
```

4 Anhang

4.1 Sales & Service

Informationen über unser Verkaufs- und Servicenetz mit den zugehörigen Adressen finden Sie problemlos im Internet. Selbstverständlich stehen Ihnen auch die Mitarbeiter im Stammwerk Malente gerne zur Verfügung:

Kendrion Kuhnke Automation GmbH
Industrial Control Systems

Lütjenburger Str. 101
23714 Malente

Tel.: +49 4523 402 0
Fax: +49 4523 402 201

sales-ics@kendrion.com
www.kendrion.com