# Beispieldokumentation

## Deutsche Beschreibung

NUTZUNGSBEDINGUNGEN

Die Verwendung der Beispielprogramme erfolgt ausschließlich unter Anerkennung folgender Bedingungen durch den Benutzer:

INSEVIS bietet kostenlose Beispielprogramme für die optimale Nutzung der S7-Programmierung und zur Zeitersparnis bei der Programmerstellung. Für direkte, indirekte oder Folgeschäden des Gebrauchs dieser Software schließt INSEVIS jegliche Gewährleistung genauso aus, wie die Haftung für alle Schäden, die aus die aus der Weitergabe der die Beispielinformationen beinhaltenden Software resultieren.

**BEISPIELBESCHREIBUNG EMESS_UI**

**Prozessdaten und Remanenz**

Die Prozessdaten der Baugruppe EMESS werden, als onboard–Peripherie, wie üblich ins Prozessabbild abgebildet. Für Dezentrale Peripherie ist alternativ ein Abbild in Datenbausteine möglich.
Da die Baugruppe EMESS-UI keinen remanenten Datenspeicher besitzt, müssen die remanenten Daten der SPS benutzt werden. D.h. die Energiezähler müssen zyklisch in remanente S7-Operanden kopiert und im Anlauf von dort initialisiert werden.

**Beispiel - Prozessdaten**

In diesem Beispiel werden die ersten 64 Byte der Prozessdaten von onboard-EMESS-Baugruppen über Blocktransfer in einen Datenbaustein kopiert. Die Daten der dezentralen EMESS-Baugruppen folgen per ConfigStage-Konfiguration.
Damit ergibt sich eine einheitliche Struktur aller EMESS-Baugruppen in EINEM Datenbaustein und die Remanenz ist sichergestellt. Bevor im ersten Zyklus Daten empfangen werden, müssen die Energiezähler aus diesem Datenbaustein beschrieben werden.

Es wird weiterhin angenommen, dass die Größe des Prozessabbildes in der Hardwarekonfiguration (ConfigStage) auf den vom EMESS belegten Adressbereich angepasst wird. Anderenfalls müssten Direkte Peripheriezugriffe benutzt werden, was einen Blocktransfer ausschließt.

**Visualisierung**

Um die Prozessdaten alle darstellen zu können, ist ein Multiplexer (FC1) programmiert. Dieser kopiert den ausgewählten Datensatz in ein Abbild (DB1), welches in der Visualisierung dargestellt wird.

**SetupDaten**

Die in der ConfigStage definierten Daten können bei EMESS (ausnahmsweise) per S7 (onboard mit L PEW) bzw. per SDO (DP) gelesen und beschrieben werden. Dieses Beispiel implementiert nur das Rücklesen um die Konfiguration aus der ConfigStage zu verifizieren.

**Diagnose-Flags**

Die vom Modul EMESS bereitgestellten Diagnose-Flags (Unter-, Überspannung, Überstrom) können onboard über L PEW (Offset 224) gelesen werden, per CAN über SDO (Objekt 3110hex + Slotindex - FB12) oder es werden die Emergency-Telegrammdaten ausgewertet (FB106). Um Kollisionen zu vermeiden sollte nur eine Variante im Code aktiviert werden.

**Programmstruktur**

Im OB100 werden die Energiezähler im Hochlauf mit den letzten Messwerten (vor dem Ausschalten) initialisiert.
Im OB1 werden die Prozessdaten zyklisch im DB2 kopiert und zur Visualisierung in DB1.
Die Setupdaten werden nur bei Änderung des aktuellen Verbrauchers zurück gelesen.
Die Diagnosedaten werden bei onboard-Modulen zyklisch gelesen, bei DP-Modulen werden die Daten der Emergency-Nachrichten (über FB106 ausgewertet) benutzt. Diese enthalten nur die Fehler-Flags, keine Konfigurationsbits, diese stehen in den zurück gelesenen Setupdaten.

**„Übrige Bausteine“**

Um aus S7 Setupdaten (sinnvoll) zu modifizieren, müssen diese remanent gehalten werden und im Hochlauf

die von der ConfigStage vorgegebenen Werte überschreiben. Dies kollidiert mit diesem Beispiel. Deshalb sind FC114 und FB1111 unbenutzt. Sie sollten aber dem interessierten Anwender nicht vorenthalten sein.

**Anpassung an den realen Hardwareausbau**
Die Datenstrukturen sind für folgende Maximalkonfiguration angelegt: onboard 11 Slots, DP Node 1 11 Slots, DP NodeID 2 11 Slots.
Wird nur eine Teilmenge bestückt, funktioniert das Programm nicht (Peripherie-Quittungsverzug – mit OB122 abgefangen) bzw. die CAN-Funktionen verursachen enorme Reaktionszeiten durch kumulierte Timeouts.
Eine grobe Anpassung des S7-Programms an den realen Hardwareausbau sollte im OB100 durch Aus- bzw. Ein-Kommentierung erfolgen.
Das Projekt „emeas_demo_full“ beinhaltet den Vollausbau, die Beispiele „emeas_demo_Slot1“ und „emeas_demo_DP1“ sind die Minimalversionen mit 1 Modul onboard bzw. dezentral.

**Hinweis:**
- Besonderheit im Anlauf:
Da der OB100 nur einmalig aufgerufen wird und die Laufzeit nicht überwacht wird, ergibt sich für die Kommunikationsbausteine, diese in kleinen Warteschleifen aufzurufen. Diese Programmierung ist für einen Aufruf im OB1 ungeeignet.
- Die SDO-Kommunikation basiert auf dem INSEVIS-Beispiel zur CAN SDO-Kommunikation.

RÜCKMELDUNGEN
Möchten Sie Erweiterungswünsche oder Fehler zu diesen Beispielen melden oder wollen Sie anderen eigene Bespielprogramme kostenlos zur Verfügung stellen? ***Bitte informieren Sie uns unter info@insevis.de***
Gern werden Ihre Programme -auf Wunsch mit Benennung des Autors- allen INSEVIS- Kunden zur Verfügung gestellt.

# English description

TERMS OF USE

The use of this sample programs is allowed only under acceptance of following conditions by the user:
The present software which is for guidance only aims at providing customers with sampling information regarding their S7-programs in order to save time. As a result, INSEVIS shall not be held liable for any direct, indirect or consequential damages respect to any claims arising from the content of such software and/or the use made by customers of this sampling information contained herin in connection with their own programs.

**SAMPLE DESCRIPTION EMESS_UI**

**General process data and nonvolatile (retentive) memory**
Process data of module EMESS – used as on-board–module - are mapped into process image as usual. Using EMESS as a decentralized peripheral module (via CAN-DP) alternatively mapping into data blocks is possible. Due to EMESS-UI has no nonvolatile memory, retentive data of PLC can be used: i. e. the energy counter must be copied cyclical into retentive S7-operands and initialized from there at start-up.

**Example- process data**
In this example the first 64 Byte of process data of each on-board EMESS module is copied by block move into a data block. The data of decentralized EMESS-modules follow subsequently, configured thus by ConfigStage. The outcome of this is a unified structure of all EMESS-modules in ONE data block and the retentivity is ensured. Before the first data are transferred cyclically (in OB1), the energy counter are written with data from these data block (in OB100).
Furthermore it is assumed that the configured process input image area size (ConfigStage) contains the area occupied by EMESS. Else Direct Periphery Access must be used, and block move is not possible.

**Visualization**
To display all the process data (especially in case of many EMESS), a multiplexer is progammed (FC1). These copies the current data record into a mirror (DB1), which is displayed in visualization.

**Setup Data**
Exceptionally at EMESS you can read and write the setup data, configured by ConfigStage, via S7:
on-board with L/T PEW respectively with SDO as DP. These example implements only read back to verify the configuration set up in ConfigStage.

**Diagnostic flags**
The module EMESS supports diagnostic-Flags for under- and overvoltage and overcurrent. Read these flags at modules onboard via L PEW (Offset 224), or via CAN at SDO (object 3110hex + slotindex) – see FB12 or use data of Emergency-messages (FB106). To avoid collisions use only one of both code blocks.

**Program structure**
In OB100 the energy counter are initialized with the values stored last before power off.
In OB1 the process data are copied cyclically into DB2 and into DB1 for visualization.
The setup data of the current consumer are read back only if the current consumer will be changed.
The diagnostic data are read cyclically for on-board-moduls, diagnostic data of DP-moduls are fetched out of the data received by Emergency – messages.( read by FB106). These diag data contains only error flags, setup flags are in read-back setup data.

**„Rest Blocks“**
To modify setup data inside S7 (wise), these data must be kept retentive and overwrite the configuration stored by ConfigStage at start up. This conflicts with this example. FC114 and FB1111 are unused, they are reserved for advanced user .

**Adjustments to real hardware configuration**
The data structures are defined for maximum configuration: on-board 11 Slots, DP Node 1 11 Slots, DP NodeID 2 11 Slots.
If only a part of this is assembled, the program won't run (periphery-timeout – to treat with OB122) respectively. the CAN-bootup and SDO fuctions will cause huge reaction times due to a lot of timeouts.
A coarse adaption of the S7-program can be done in OB100 by insert or remove of commented code.
The project „emeas_demo_full“ contains the completion, the projects „emeas_demo_Slot1“ and

„emeas_demo_DP1“ are most simplified versions with 1 module onboard resp. decentral.

**Hints:**
- startup code feature :
Due to OB100 runs only once and run-time will not be supervised, it is obviously to realize the communication functions in local loops. The parts of the program are inappropriate to be used cyclically in OB1
- The SDO - communication routines are based on the INSEVIS - example CAN SDO-communication.

FEEDBACK
Do you want to inform us about necessary increments or errors or do you want to provide us with your sample programs to offer it for free to all customers? ***Please inform us at info@insevis.de***
Gladly we would provide your program -if you wish with the authors name- to all other customers of INSEVIS.