



Device OD extension with user defined FAQ No.0017 subobjects and a callback method

Part	Version	Revision	Date	Status
en	6.4.0.0	001	2019-01-01	Released

Content

Introduction	2
OD access to objects and subobjects	2
Device OD extension with a user defined subobject	3
OD object values: validity and access	3
Example: User OD subobject redirection to the device OD.....	4
User OD object function index modification	4
OD callback method environment	5
OD callback method implementation	6
Redirection results.....	7
Restrictions	7
Disclaimer	8

Introduction

This document gives a general overview how an already existing object in an internal CANopen device OD (Object Dictionary) can be extended by a user defined subobject. For this, an OD callback method has to be introduced in the CVs6 (COMBIVIS studio 6) application. This is useful for working with any subobjects, which are included just in selected KEB devices (e.g. subobject 1800:3 inhibit time).

The further illustrated solution gives information for some background parts and a sample implementation. It has been developed on the base of a CANopen slave device and has been introduced in the library KEB_CanSlave 3.5.9.72.

OD access to objects and subobjects

Every CANopen slave device has a device OD which contains all needed device specific objects. It can be extended with a user defined OD by the user's requirements for any additional parameters and values. This user OD has to be instanced directly in the CVs6 application and needs to be registered at the UserOD VAR_INPUT of the CANopen slave device.



KEB_CanSlave CanOpenSlaveDevice: VAR_INPUT + VAR_OUTPUT

The user OD instance provides an easy and flexible access to its content by the application variables. The device OD is fixed and limited compared to a user OD, because it's only directly accessible by the device itself. The access from the CAN bus to get or set CANopen values of any OD is provided by the library KEB_CanSlave and will be handled automatically by using the KEB Channelhandler.

In detail, the CanOpenSlaveDevice methods GetCanValue() and SetCanValue() are called for commands from the CAN bus and they will try to access the requested object in the user OD first, if it is registered.

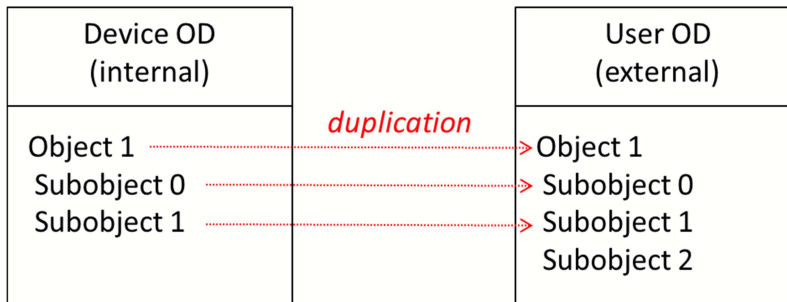
The device OD will only be accessed, if the user OD has not been registered, or else if the requested object is not included or not accessible. This order is for the reason, that an object, included in both ODs, could be modified to reject the access in the user OD and redirect it to the device OD if needed. In this case, there is also no other possibility to work with the internal object directly.

This mixed access to an object is useful for the main topic of this document and will be explained in the next sections. The way how to create a user defined OD is not content of this document.

For some more information regarding the KEB_ObjectDictionary library please watch the section [OD callback method implementation](#).

Device OD extension with a user defined subobject

To extend an existing object in the device OD with an additional subobject, the whole object has to be duplicated in the user OD. This results in an access problem, which also has to be resolved by the user.



Object Dictionary: duplication of Object 1, extension with Subobject 2

The access problem causes two questions:

1. Which subobject is valid in which OD?
2. How can the access to the concerned OD for this subobject be granted?

OD object values: validity and access

For any duplicated or added subobject in the user OD, the next list shows the validity and the access properties of the values. The user OD has a higher priority and is accessible without any further adaptations.

1. Subobject 0: valid in the user OD (number of further subobjects).

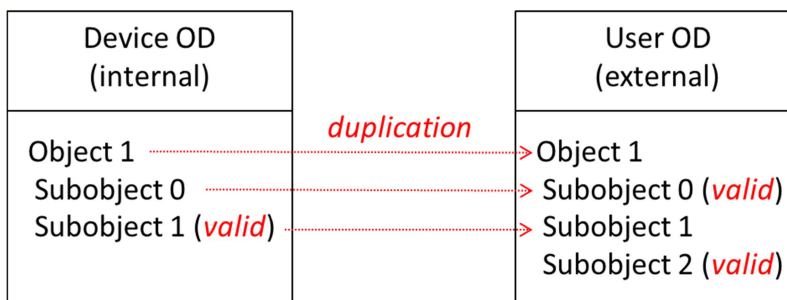
The subobject 0 contains the number of further subobjects (array). Because all objects will be accessed in the user OD first and the total number of subobjects has been increased, it is needed to use the updated value in the user OD. The device's OD subobject 0 stays at the previous value for the internal access of the device.

2. Subobject 2: valid in the user OD (additional subobjects).

Additional subobjects exist only in the user OD and are accessible without any restrictions.

3. Subobject 1: valid in the device OD (duplicated subobjects).

Further duplicated subobjects contained in both ODs, have to be redirected to the device OD for the reason that the device is only working with its internal objects.



Object Dictionary: validity of duplicated object

For the 3rd option (duplicated subobjects), it is necessary to organize the shared access to the device and user parts of the whole object.

Example: User OD subobject redirection to the device OD

In the next CVs6 application example it will be assumed, that the device object 1800 with two subobjects has been duplicated in the user OD of a CANopen device. The entry “inhibit time” (included in selected KEB devices only) has been added in the subobject 3.

```
//Pdo1 Tx Communication, subobject 0
(
  Index      :=16#1800,           //Index
  SubIdxMod  :=16#0000,         //SubIdxMod
  Itype      :=OD_TBL_ITEM_ENTRIES_3 OR
  //further properties, collapsed [10 lines]
  ReadFuncIdx :=0,             //ReadFuncIdx
  WriteFuncIdx :=0,           //WriteFuncIdx
  LowerLim    :=0,             //LowerLim
  UpperLim    :=0,             //UpperLim
  DefValue    :=3,             //DefValue
  pData      :=ADR(Para1800_00) //pData
),
```

Example: object 1800, subobject 0, addressing 3 further subobjects, valid in the user OD

To redirect the two device subobjects to the device OD again, the object function index values unequal to zero will be used. The subobject 3 doesn't need any modification.

User OD object function index modification

The read and write function index value enables an OD callback method separately for a reading or writing access. The user defined value can be interpreted in the method to differ between individual meanings for any number. The function index' default value “0” disables the callback method.

In this example it will be assumed, that the function index value of subobject 1+2 has been set to “1” for both access directions. The variables Para1800_dummy are needed for placeholder reasons only.

```
//subobject 1
(
  Index      :=16#1800,           //Index
  SubIdxMod  :=16#0100,         //SubIdxMod
  Itype      :=OD_TBL_ITEM_ENTRIES_0 OR
  //further properties, collapsed [9 lines]
  ReadFuncIdx :=1,             //ReadFuncIdx
  WriteFuncIdx :=1,           //WriteFuncIdx
  LowerLim    :=0,             //LowerLim
  UpperLim    :=65535,         //UpperLim
  DefValue    :=0,             //DefValue
  pData      :=ADR(Para1800_dummy) //pData
),
//subobject 2
(
  Index      :=16#1800,           //Index
  SubIdxMod  :=16#0200,         //SubIdxMod
  Itype      :=OD_TBL_ITEM_ENTRIES_0 OR
  //further properties, collapsed [9 lines]
  ReadFuncIdx :=1,             //ReadFuncIdx
  WriteFuncIdx :=1,           //WriteFuncIdx
  LowerLim    :=0,             //LowerLim
  UpperLim    :=65535,         //UpperLim
  DefValue    :=0,             //DefValue
  pData      :=ADR(Para1800_dummy) //pData
),
```

Example: object 1800, subobject 1+2 with function index 1, valid in the device OD only

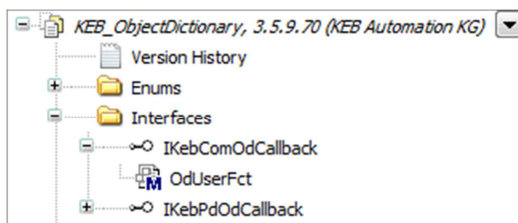
OD callback method environment

The next table shows the relation between the different POUs (Program Organisation Units) that are needed for an OD callback implementation in CVs6:

Library / POU	POU used	POU instances
KEB_CanSlave	CanOpenSlaveDevice	KEB_CanOpenSlave
KEB_ObjectDictionary	ObjectDictionary	UserOdCallback
KEB_ObjectDictionary	IKebComOdCallback (interface) with IKebComOdCallback.OdUserFct() (interface method)	
new Function Block	UserOdCallback (FB) impl. IKebComOdCallback with UserOdCallback.OdUserFct()	MyCallback MyCallback. OdUserFct()

The CANopen slave device is listed here for the connection of a user OD in the initialisation steps of the CVs6 application. An overview to the CanOpenSlaveDevice POU can be found at the section ["OD access to objects and subobjects"](#).

The OD callback method OdUserFct() is included in the interface IKebComOdCallback of the KEB_ObjectDictionary library. This interface has to be implemented in a new function block. So, the interface method can immediately be modified by the user's needs.



KEB_ObjectDictionary with interface IKebComOdCallback and method OdUserFct()

First, the instance of the user OD has to be registered in the CANopen slave device and the instance of the new function block has to be registered at the user OD's .callbackInst VAR_INPUT.

```
//created and initialized a user OD before
KEB_CanOpenSlave.UserOD := ADR(UserOD);
UserOd.callbackInst := MyCallBack;
```

Registration of the user OD and the user callback instance method

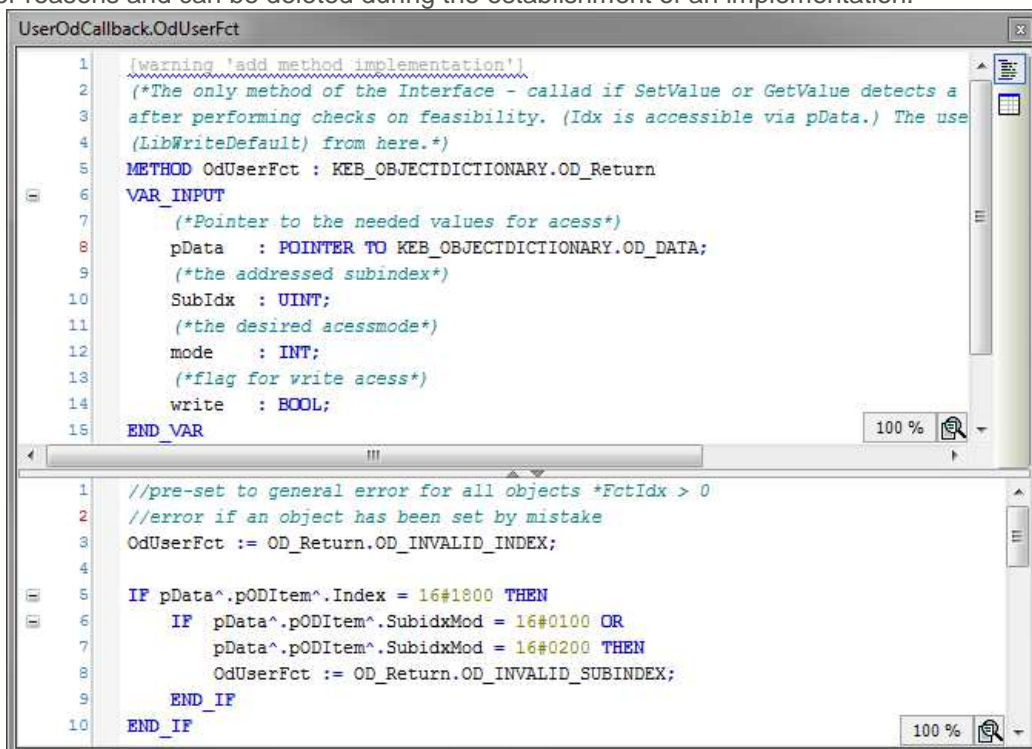
The way how to create a user defined OD is not content of this document.

OD callback method implementation

Now, the content of the interface method has to be modified in the function block instance by the requirements of the example introduced in the last sections.

The main target is, to override the behaviour of that OD, where the method has been registered. By calling the GetValue() and SetValue() methods of the OD, the function index values of the actual subobject will be checked and if they have been modified, the callback method will be triggered. Its OD_RETURN value will override the return value of the internal OD methods and gets back to the caller of the OD (e.g. the KEB Channelhandler).

First remark to be focused on is the warning statement on top of the variable declaration. It's for reminder reasons and can be deleted during the establishment of an implementation.



```

UserOdCallback.OdUserFct
1  [warning 'add method implementation!']
2  (*The only method of the Interface - called if SetValue or GetValue detects a
3  after performing checks on feasibility. (Idx is accessible via pData.) The use
4  (LibWriteDefault) from here.*)
5  METHOD OdUserFct : KEB_OBJECTIDictionary.OD_Return
6  VAR_INPUT
7  (*Pointer to the needed values for access*)
8  pData : POINTER TO KEB_OBJECTIDictionary.OD_DATA;
9  (*the addressed subindex*)
10 SubIdx : UINT;
11 (*the desired accessmode*)
12 mode : INT;
13 (*flag for write access*)
14 write : BOOL;
15 END_VAR

1 //pre-set to general error for all objects *FctIdx > 0
2 //error if an object has been set by mistake
3 OdUserFct := OD_Return.OD_INVALID_INDEX;
4
5 IF pData^.pODItem^.Index = 16#1800 THEN
6     IF pData^.pODItem^.SubIdxMod = 16#0100 OR
7     pData^.pODItem^.SubIdxMod = 16#0200 THEN
8         OdUserFct := OD_Return.OD_INVALID_SUBINDEX;
9     END_IF
10 END_IF
    
```

Example implementation of the callback method OdUserFct() for two subobjects

The VAR_INPUT pData contains the actual object properties. This structure has to be accessed for executing any user defined operation and setting any return values depending on the object identity with the index, the subindex and further values. Be aware that the subindex value is the MSB (most significant byte) of the OD_DATA item SubIdxMod.

In case of the 3rd subobject option of the list in section [OD object values: validity and access](#), the return value "OD_INVALID_SUBINDEX" can be used to reject the access to the user OD for the duplicated subobjects in this specific way.

It's good to initiate the method's return value to any general item of the enumeration OD_Return for the reason, that if any function index value has been set by mistake, it could be detected by checking this value.

For any directory specific procedure the write VAR_INPUT needs to be interpreted.

Redirection results

Requests for any (sub-) object, included in just one of the ODs, will be directed to the fitting OD. Requests for a missing object will be rejected.

The callback method will be triggered if the OD's GetValue or SetValue method detects a read or write function index value > 0 in the actual object. Then, the OD's returning value can be overwritten by the method with an error value that rejects the access to the actual OD for the actual (sub-) object.

If both function index values equals to 0, only the OD standard methods are responsible for the returning value. If any function index value has been set unequal to 0 by mistake, it can be detected by an unexpected OD_RETURN value (if this has been pre-set), because the callback methods shall never be triggered in the case the user wants to read an object only by the OD standard methods.

Restrictions

As mentioned in the introduction, this solution is only applicable for the developed CANopen slave device. The behaviour of any other device might be different in important details and has to be checked for any use.

A duplication of objects is only possible for a continuous number of subobjects. This has to be created and organised if needed.

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 particularly necessary, if changes are executed, which serve for the further development or adaptation 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