ISOBUS VT Client

Example project

Contents

1. ISOBUS project requirements
2. Configuring ISOBUS in MultiTool
3. CODESYS code template
4. ISO-Designer
5. Downloading application and object pool

ISOBUS Project Requirements

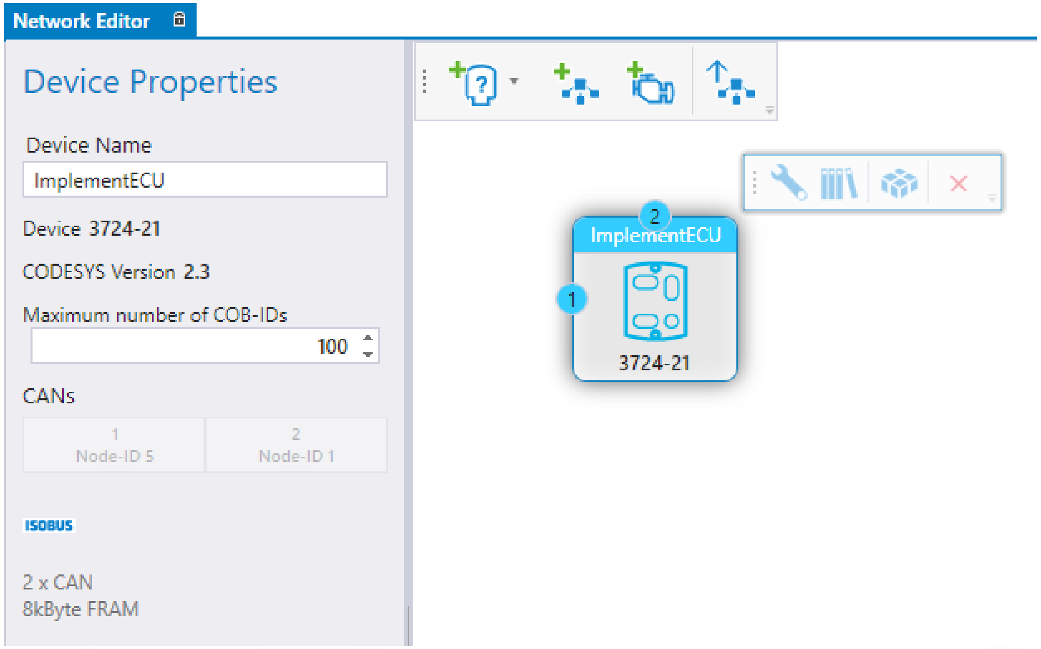
- For this example, Epec device needs to support ISOBUS
  - Fourth digit in the product code is ”E”, for example, E30E3606-23
  - Firmware version 1.168 or newer
- The following installations are needed
  - CODESYS 2.3
  - Jetter ISO-Designer 4.0.6 or newer
  - Epec SDK 2.3 or newer
  - Epec CANmoon

MultiTool

Configure ISOBUS features

Creating a MultiTool Project

1. Open MultiTool and create a new project
2. Add 3606/3724 ISOBUS device using
3. Select the device and rename it (e.g., ISOBUSimplement)



Device Properties

Device Name: ImplementECU

Device 3724-21

CODESYS Version 2.3

Maximum number of COB-IDs: 100

CANs

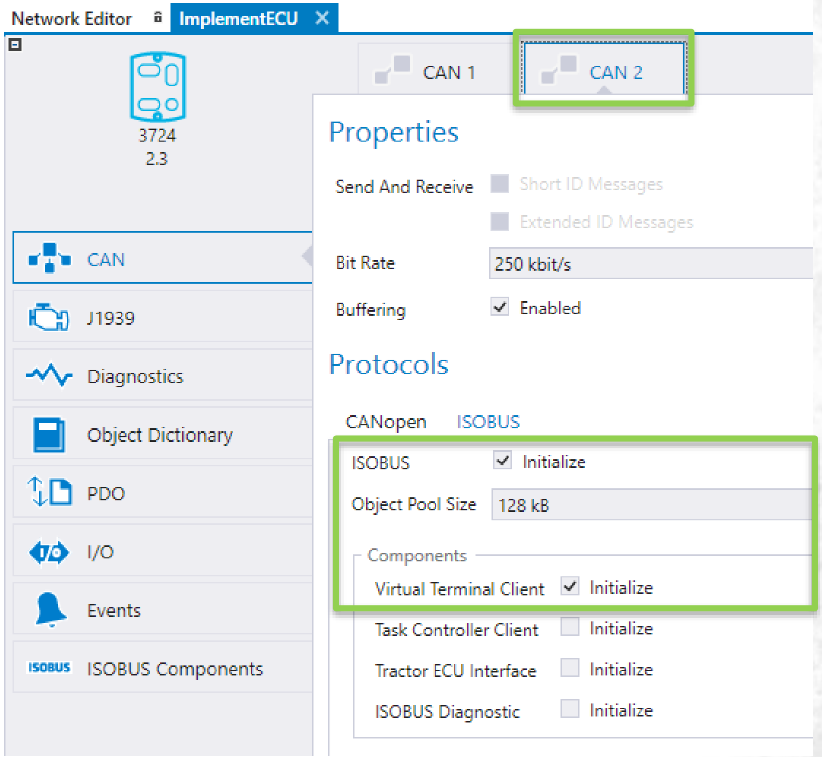
1	2
Node-ID 5	Node-ID 1

ISOBUS

2 x CAN
8kByte FRAM

Configuring ISOBUS

4. Double-click the device to open the configuration view
   - The needed ISOBUS functionalities are selected in **CAN** tab> **Protocols** > **ISOBUS**
5. Select **CAN 2** > **Protocols** > **ISOBUS**
6. **Initialize** ISOBUS and Virtual Terminal Client
7. Select **Object Pool Size** (default 64 kB)



Properties

Send And Receive □ Short ID Messages

□ Extended ID Messages

Bit Rate 250 kbit/s

Buffering ☑ Enabled

Protocols

CANopen ISOBUS

ISOBUS ☑ Initialize

Object Pool Size 128 kB

Components

Virtual Terminal Client ☑ Initialize

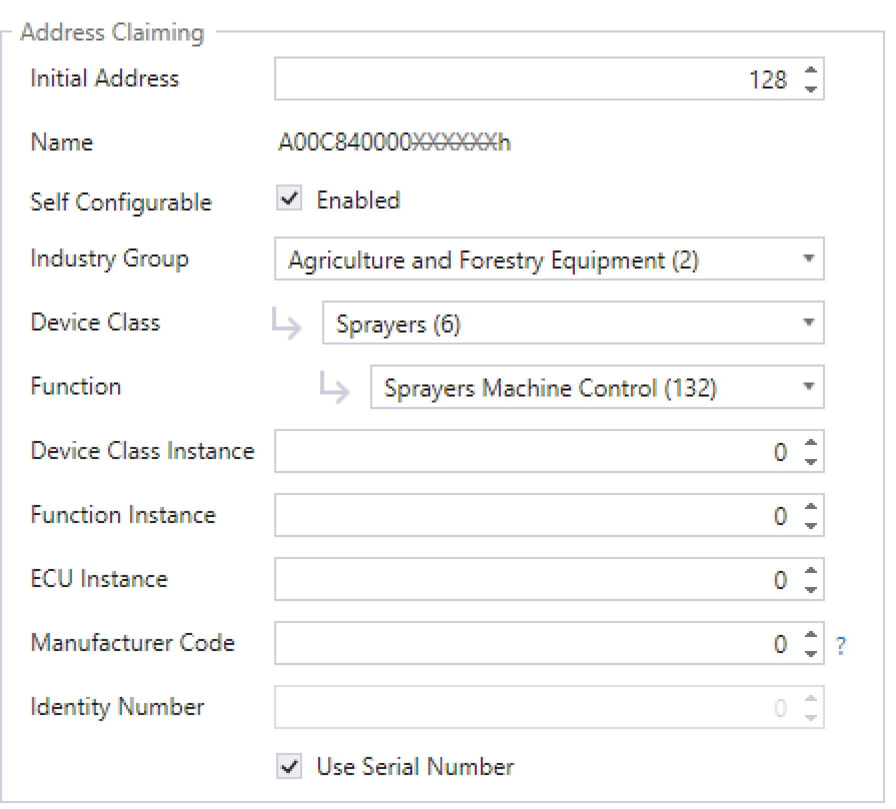
Task Controller Client □ Initialize

Tractor ECU Interface □ Initialize

ISOBUS Diagnostic □ Initialize

Configure Address Claiming

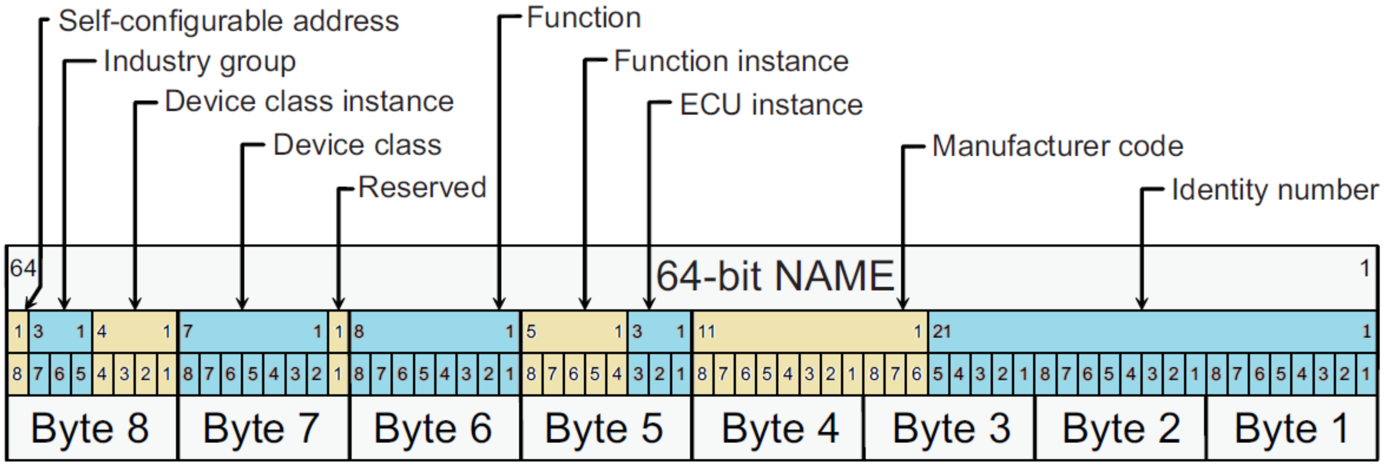
7. Configure the **Address Claiming** protocol
   - **Name** is combined from the given data in the Address Claiming section
   - **Name** is needed for every device in the ISOBUS network
   - **Initial Address**: it is recommended to use values 128 – 237 for the implement application



Address Claiming

Initial Address	128
Name	A00C840000~~XXXXXX~~h
Self Configurable	☑ Enabled
Industry Group	Agriculture and Forestry Equipment (2)
Device Class	↳ Sprayers (6)
Function	↳ Sprayers Machine Control (132)
Device Class Instance	0
Function Instance	0
ECU Instance	0
Manufacturer Code	0 ?
Identity Number	0
	☑ Use Serial Number

Address Claiming



NOTE The 64-bit value is sent with byte 1 first and byte 8 last when transmitted on the network.

Figure 1 — NAME bit fields in controller area network (CAN) message data bytes

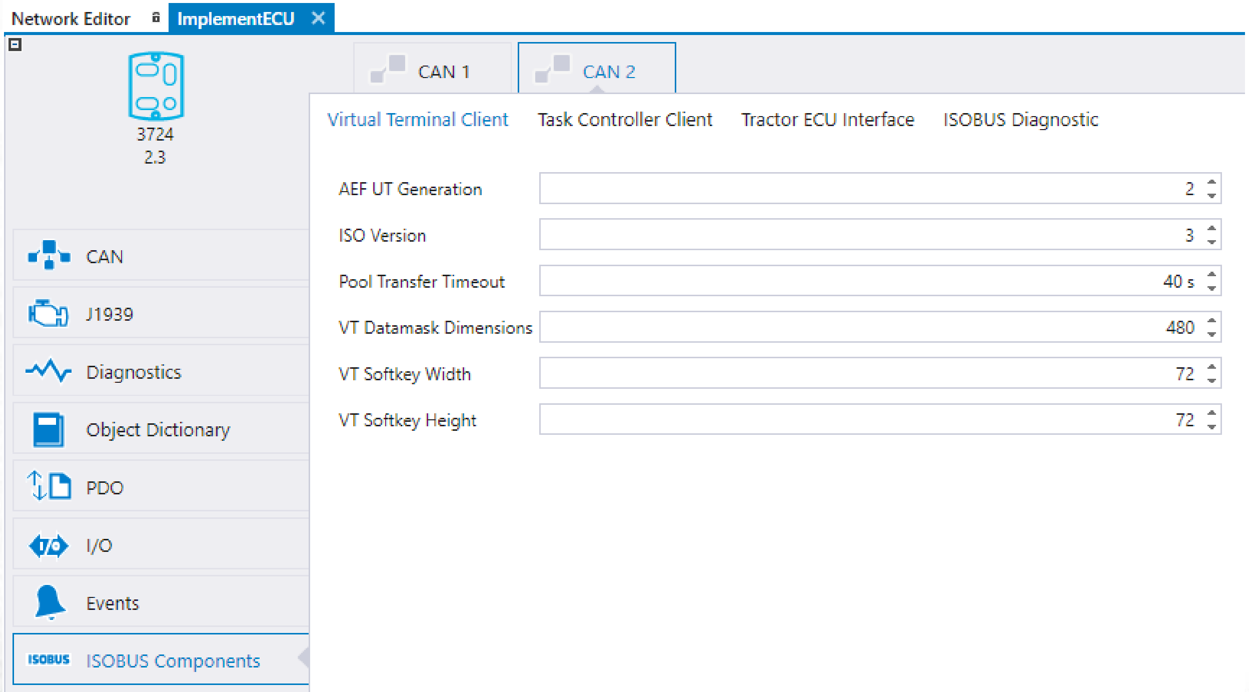
Image source ISO 11783-5:2011(E), chapter **4.3.2 NAME**

Address Claiming

Field	Description
Self-configurable address	Self-configurable device is able to select a new address on an address conflict. Self-configurable (1) / not (0)
Industry group	Industry group, for example, 2 = Agriculture and forestry equipment
Device class	Provides name for group of functions which are combined under same device class, for example, 4 = Planter and seeders
Function	Function for control function, for example, 132 = Planters/Seeders Machine Control
Device class instance	Value is used to make difference for identical device classes in same network. Instance number 0 recommended.
Function instance	Value is used to make difference with several identical function instances. Instance number 0 recommended.
ECU instance	Value is used to make difference if there is several ECUs which together form a single function. Instance number 0 recommended (= function is managed by one ECU).
Manufacturer code	Indicates the machine manufacturer (see www.sae.org for SAE Manufacturer Code Request).
Identity number	Assigned by application code, recommended to use serial number for this field (select Use Serial Number box).

ISOBUS Components

- ISOBUS functionalities have additional settings/definitions that are configured in **ISOBUS Components** tab
- The configurations are imported to CODESYS code template

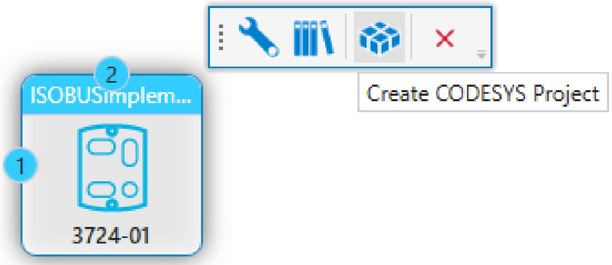


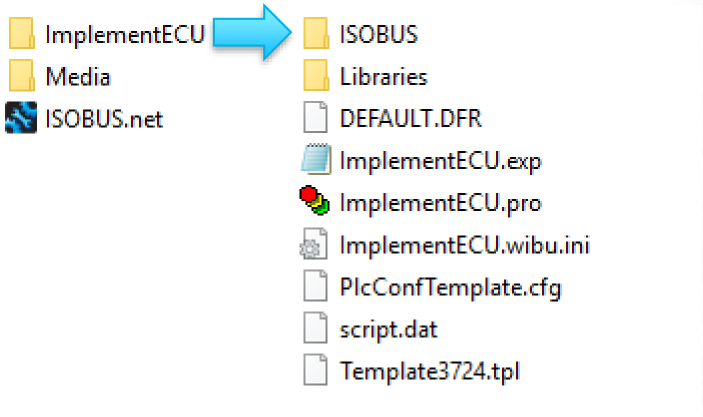
AEF UT Generation	2
ISO Version	3
Pool Transfer Timeout	40 s
VT Datamask Dimensions	480
VT Softkey Width	72
VT Softkey Height	72

VT Datamask Dimensions or **VT Softkey Width/Height** update requires updates to ISO-Designer template project.

Creating a MultiTool Project

8. To create the CODESYS project, select the device and then **Create CODESYS Project**
9. MultiTool creates a project structure including **ISOBUS** folder




ImplementECU
Media
ISOBUS.net

ISOBUS
Libraries
DEFAULT.DFR
ImplementECU.exp
ImplementECU.pro
ImplementECU.wibu.ini
PlcConfTemplate.cfg
script.dat
Template3724.tpl

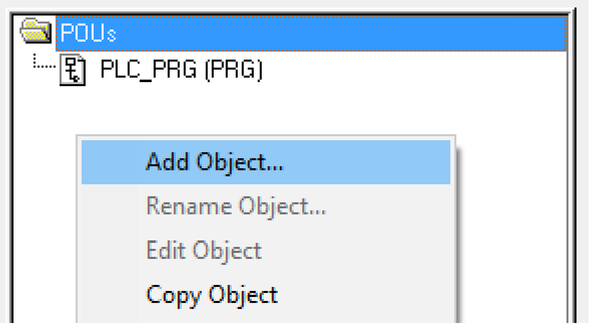
ISOBUS Folder Structure

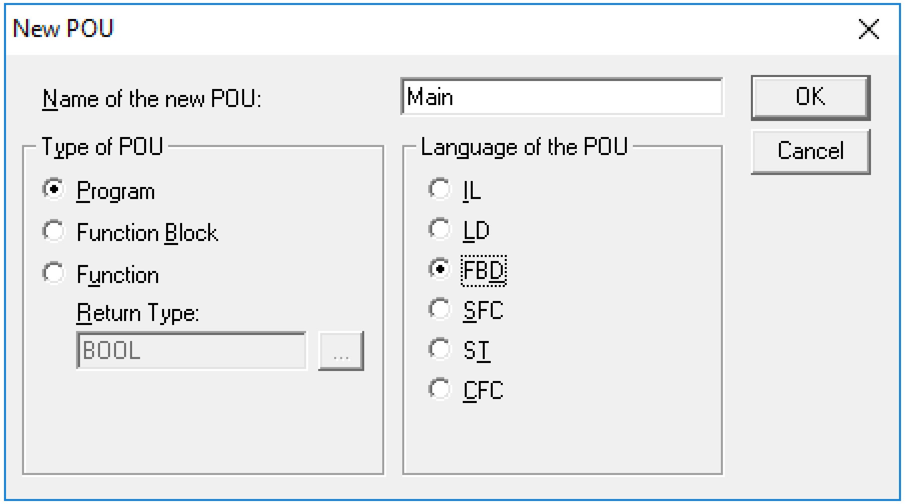
Folder name	Description
★ Jetter	Includes a template project for ISO-Designer (IsobusVtObjectPool.jvw)
Python	Python scripts handles the communication between CODESYS and ISO-Designer/XML definitions
★ BinaryMaker	Combines data from IsobusTc, IsobusVt and Languages folders to one object pool file that is downloaded to the control unit (*downloaded.bin*)
Exp	Import ISOBUS macro related files
IsobusTc	Code template update files. Example XML file for TC client (tcClientPool.xml). The XML file, for example, describes the used measurement units in the machine. See also, http://dictionary.isobus.net/isobus/dDEntity
IsobusVt	Code template update files (*IsobusExportVtInfo.exp*)
Languages	Includes *languages.xml* that can be used for localization (languages, text IDs and corresponding texts)

CODESYS PROJECT

Adding Main Program

10. Add *Main* program
    - Right-click > **Add Object**
    - This will be the main program for the user application





Name of the new POU: Main

Type of POU

- ● Program
- ○ Function Block
- ○ Function

Return Type: BOOL

Language of the POU

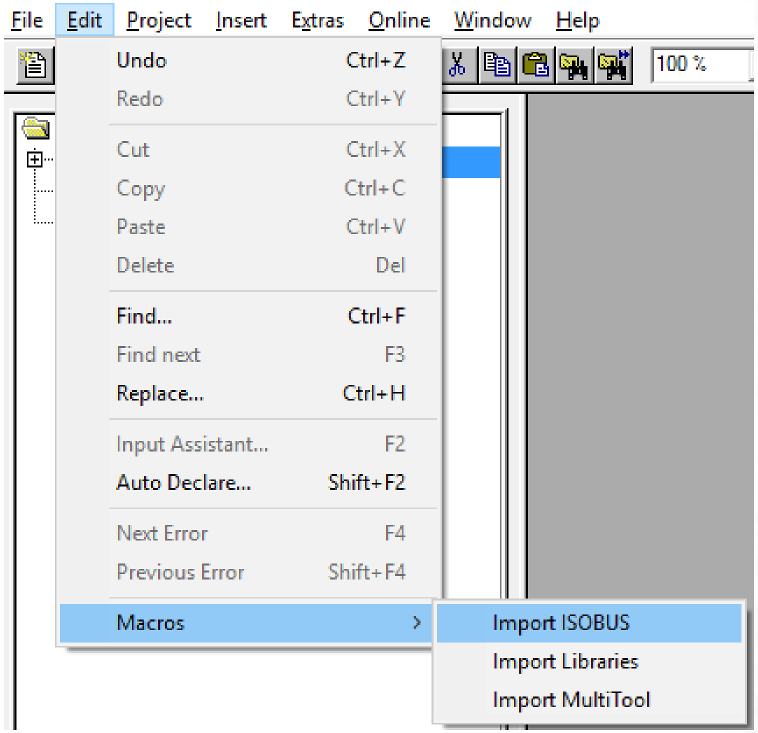
- ○ IL
- ○ LD
- ● FBD
- ○ SFC
- ○ ST
- ○ CFC

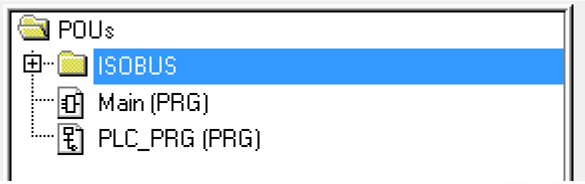
Import ISOBUS

11. Run **Edit > Macros > Import ISOBUS** macro

Import ISOBUS macro
- Adds ISOBUS programs, object handlers and a template program for data/alarm masks
- Updates object pool (downloaded binary)
- Updates code template's list of object pool's objects (global variables)

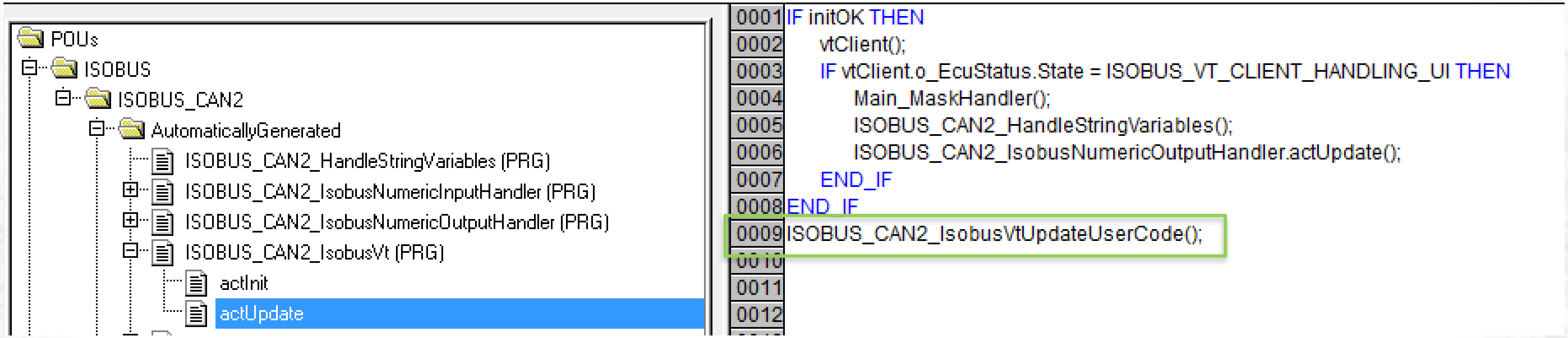
Import MultiTool updates MultiTool changes to CODESYS code template





User Code Init and Update

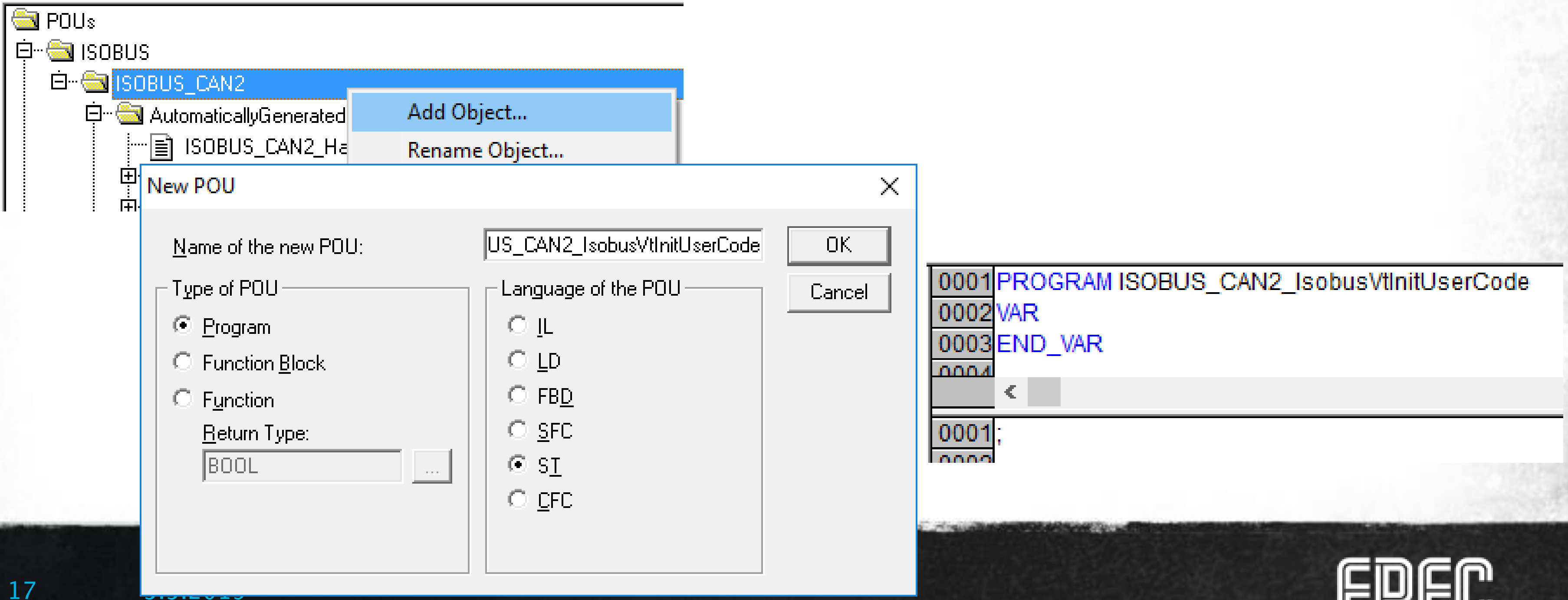
12. The VT client program *ISOBUS_CANx_IsobusVt* requires own programs for user init and update code
    - These programs are called in *ISOBUS_CANx_IsobusVt* program’s *actInit* and *actUpdate*


POUs
ISOBUS
ISOBUS_CAN2
AutomaticallyGenerated
ISOBUS_CAN2_HandleStringVariables (PRG)
ISOBUS_CAN2_IsobusNumericInputHandler (PRG)
ISOBUS_CAN2_IsobusNumericOutputHandler (PRG)
ISOBUS_CAN2_IsobusVt (PRG)
actInit
actUpdate

```
IF initOK THEN
    vtClient();
    IF vtClient.o_EcuStatus.State = ISOBUS_VT_CLIENT_HANDLING_UI THEN
        Main_MaskHandler();
        ISOBUS_CAN2_HandleStringVariables();
        ISOBUS_CAN2_IsobusNumericOutputHandler.actUpdate();
    END_IF
END_IF
ISOBUS_CAN2_IsobusVtUpdateUserCode();
```

User Code Init and Update

13. Add user code init and update programs
    - *ISOBUS_CANx_IsobusVtInitUserCode*
    - *ISOBUS_CANx_IsobusVtUpdateUserCode*
    - To build without errors, add a semicolon **;** to both programs



Name of the new POU: US_CAN2_IsobusVtInitUserCode

Type of POU
● Program
○ Function Block
○ Function
Return Type: BOOL

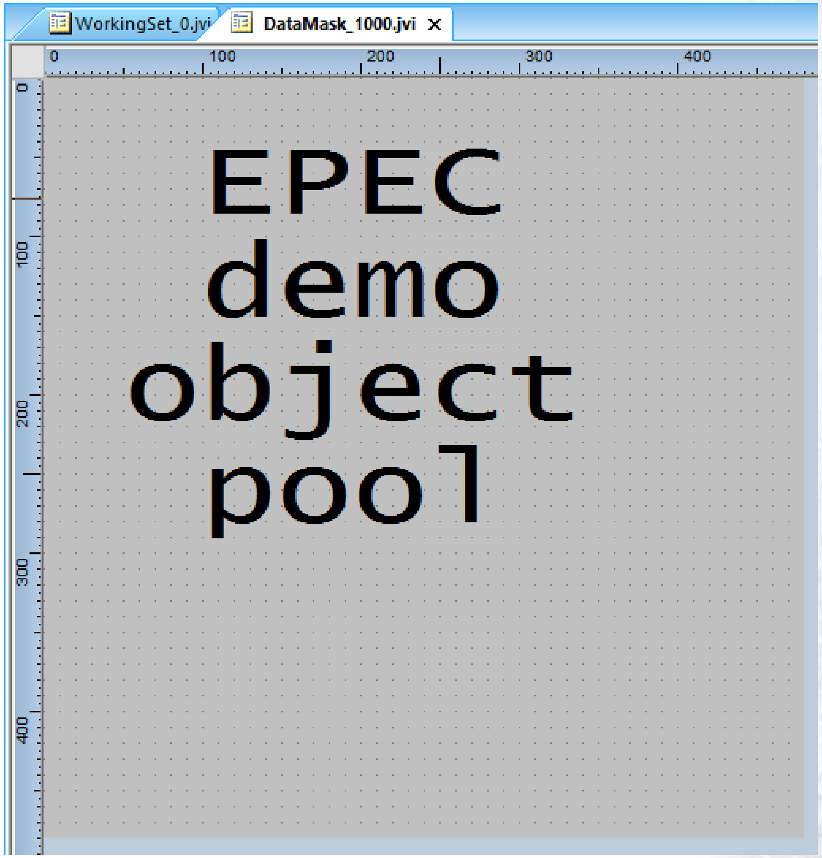
Language of the POU
○ IL
○ LD
○ FBD
○ SFC
● ST
○ CFC

```
PROGRAM ISOBUS_CAN2_IsobusVtInitUserCode
VAR
END_VAR
```

```
;
```

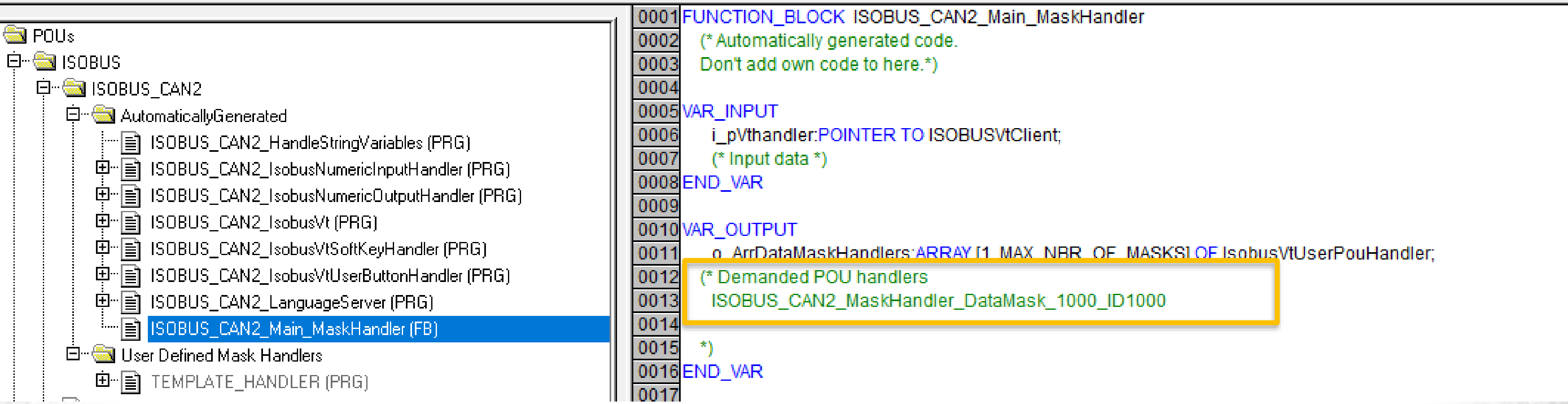
ISO-Designer Template Project

- The ISO-Designer template project includes one
  - Working Set
  - Data Mask with a text
- The data mask needs to have a handler program in CODESYS application



Creating First Mask Handler

- Import ISOBUS macro generates names for each mask to *ISOBUS_CANx_Main_MaskHandler*
- The program name can be copied from *ISOBUS_CANx_Main_MaskHandler* comment:
  - *ISOBUS_CANx_MaskHandler_DataMask_1000_ID1000*

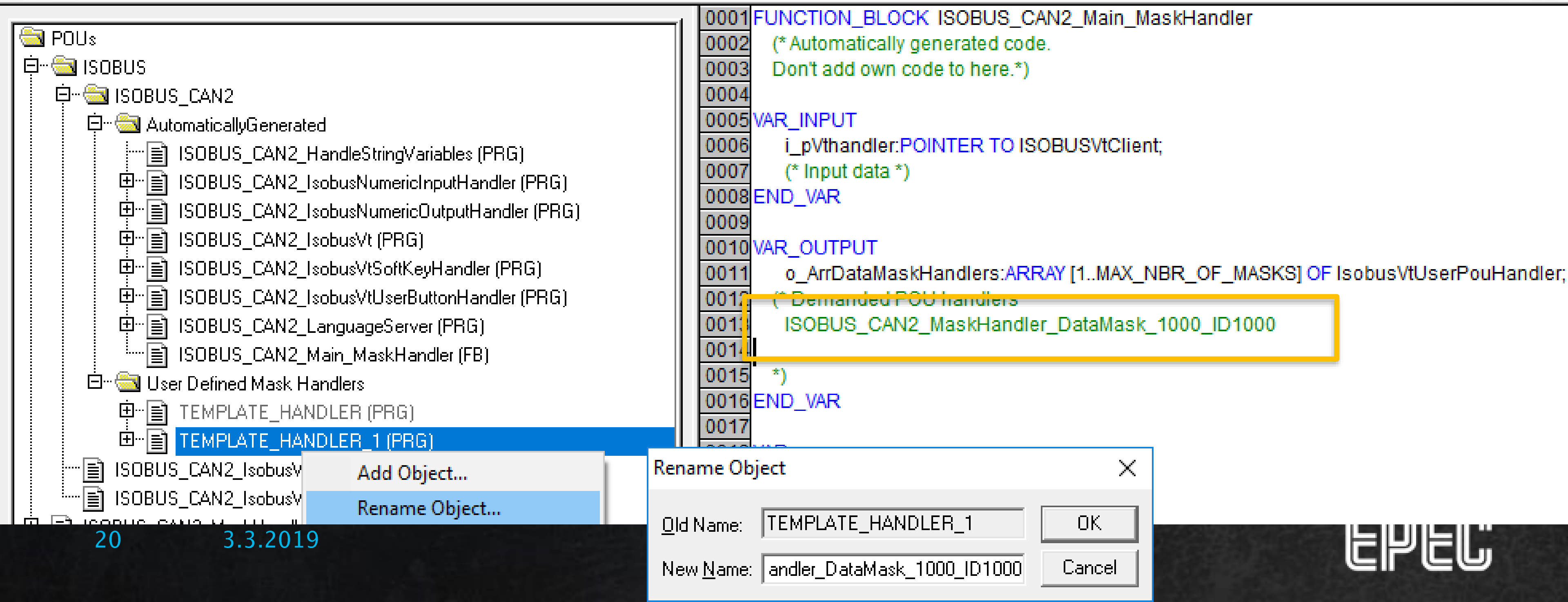


```
POUs
⊟ ISOBUS
  ⊟ ISOBUS_CAN2
    ⊟ AutomaticallyGenerated
        ISOBUS_CAN2_HandleStringVariables (PRG)
      ⊞ ISOBUS_CAN2_IsobusNumericInputHandler (PRG)
      ⊞ ISOBUS_CAN2_IsobusNumericOutputHandler (PRG)
      ⊞ ISOBUS_CAN2_IsobusVt (PRG)
      ⊞ ISOBUS_CAN2_IsobusVtSoftKeyHandler (PRG)
      ⊞ ISOBUS_CAN2_IsobusVtUserButtonHandler (PRG)
      ⊞ ISOBUS_CAN2_LanguageServer (PRG)
        ISOBUS_CAN2_Main_MaskHandler (FB)
    ⊟ User Defined Mask Handlers
      ⊞ TEMPLATE_HANDLER (PRG)
```

```
FUNCTION_BLOCK ISOBUS_CAN2_Main_MaskHandler
  (* Automatically generated code.
  Don't add own code to here.*)

VAR_INPUT
    i_pVthandler:POINTER TO ISOBUSVtClient;
    (* Input data *)
END_VAR

VAR_OUTPUT
    o_ArrDataMaskHandlers:ARRAY [1..MAX_NBR_OF_MASKS] OF IsobusVtUserPouHandler;
  (* Demanded POU handlers
    ISOBUS_CAN2_MaskHandler_DataMask_1000_ID1000

  *)
END_VAR
```

Creating First Mask Handler

- Steps to be done
  1. Copy-paste *TEMPLATE_HANDLER (PRG)*
  2. Copy the name of the mask handler (from *ISOBUS_CANx_Main_MaskHandler*)
  3. Rename *TEMPLATE_HANDLER_1 (PRG)*


POUs
ISOBUS
ISOBUS_CAN2
AutomaticallyGenerated
ISOBUS_CAN2_HandleStringVariables (PRG)
ISOBUS_CAN2_IsobusNumericInputHandler (PRG)
ISOBUS_CAN2_IsobusNumericOutputHandler (PRG)
ISOBUS_CAN2_IsobusVt (PRG)
ISOBUS_CAN2_IsobusVtSoftKeyHandler (PRG)
ISOBUS_CAN2_IsobusVtUserButtonHandler (PRG)
ISOBUS_CAN2_LanguageServer (PRG)
ISOBUS_CAN2_Main_MaskHandler (FB)
User Defined Mask Handlers
TEMPLATE_HANDLER (PRG)
TEMPLATE_HANDLER_1 (PRG)
ISOBUS_CAN2_IsobusV
ISOBUS_CAN2_IsobusV
Add Object...
Rename Object...
Rename Object
OK
Cancel

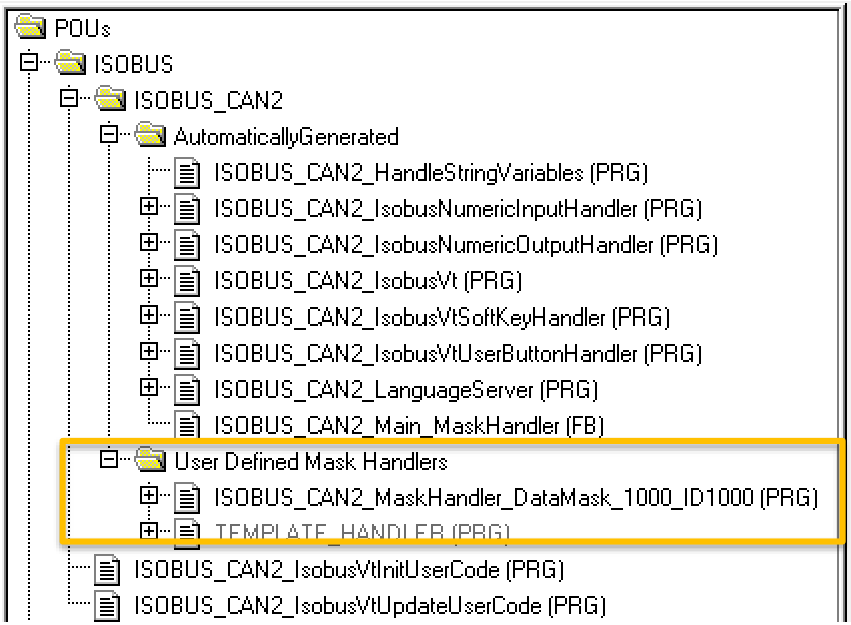
```
FUNCTION_BLOCK ISOBUS_CAN2_Main_MaskHandler
  (* Automatically generated code.
  Don't add own code to here.*)

VAR_INPUT
    i_pVthandler:POINTER TO ISOBUSVtClient;
    (* Input data *)
END_VAR

VAR_OUTPUT
    o_ArrDataMaskHandlers:ARRAY [1..MAX_NBR_OF_MASKS] OF IsobusVtUserPouHandler;
  (* Demanded POU handlers
    ISOBUS_CAN2_MaskHandler_DataMask_1000_ID1000

  *)
END_VAR
```

Old Name: TEMPLATE_HANDLER_1

New Name: andler_DataMask_1000_ID1000

Creating First Mask Handler

- *ISOBUS_CANx_Main_MaskHandler* makes a list of required data/alarm mask handler programs



```
POUs
  ISOBUS
    ISOBUS_CAN2
      AutomaticallyGenerated
        ISOBUS_CAN2_HandleStringVariables (PRG)
        ISOBUS_CAN2_IsobusNumericInputHandler (PRG)
        ISOBUS_CAN2_IsobusNumericOutputHandler (PRG)
        ISOBUS_CAN2_IsobusVt (PRG)
        ISOBUS_CAN2_IsobusVtSoftKeyHandler (PRG)
        ISOBUS_CAN2_IsobusVtUserButtonHandler (PRG)
        ISOBUS_CAN2_LanguageServer (PRG)
        ISOBUS_CAN2_Main_MaskHandler (FB)
      User Defined Mask Handlers
        ISOBUS_CAN2_MaskHandler_DataMask_1000_ID1000 (PRG)
        TEMPLATE_HANDLER (PRG)
      ISOBUS_CAN2_IsobusVtInitUserCode (PRG)
      ISOBUS_CAN2_IsobusVtUpdateUserCode (PRG)
```

Build and Download

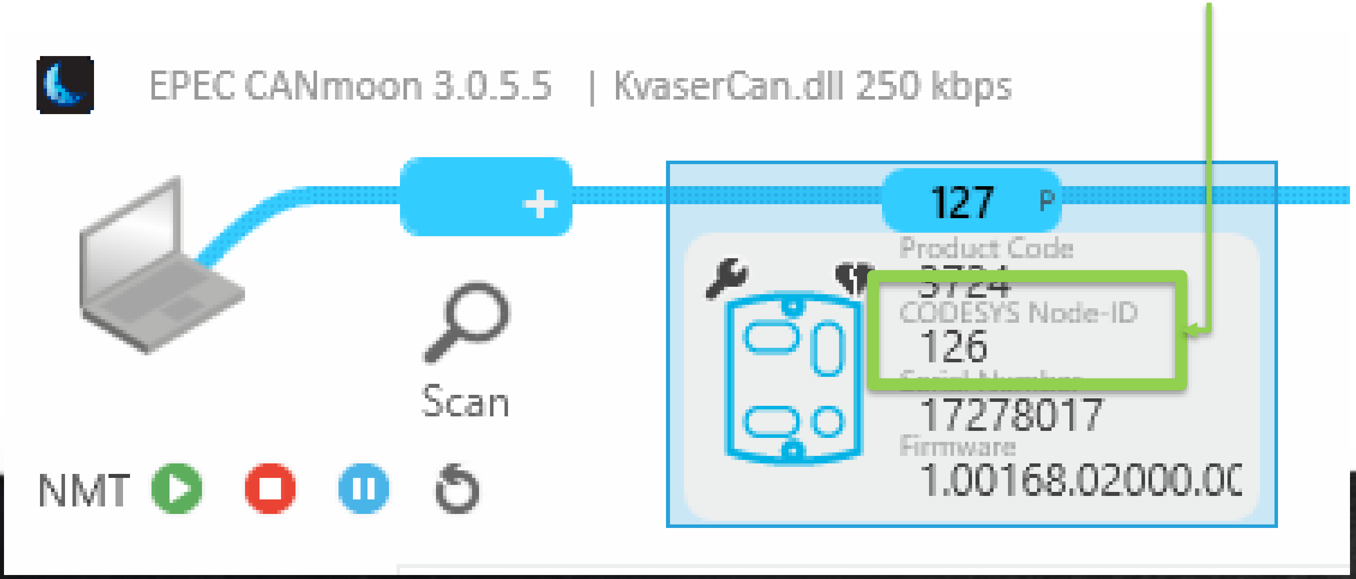
Using CODESYS for the application – CANmoon for the object pool

Build and Download

- Build project **Project > Rebuild all**
  - Check possible errors from build messages [F4]
- To download the application
  1. Check the control unit communication parameters
     - By default, the units have
       - CODESYS communication node-ID 126 (download node-ID)
       - Application node-ID 127
  2. Define the communication parameters to CODESYS
  3. Login and download
- The following slides show these steps in more detail

Check the Communication Parameters

- At this point,
  - the unit should be connected to supply voltage
  - the CAN card should be attached (PC <> CAN bus)
  - terminating resistor(s) should be attached
- Open Epec CANmoon, select used CAN card
- Scan the CAN bus to find out the CODESYS node-ID



EPEC CANmoon 3.0.5.5 | KvaserCan.dll 250 kbps

127 P

Product Code
3724

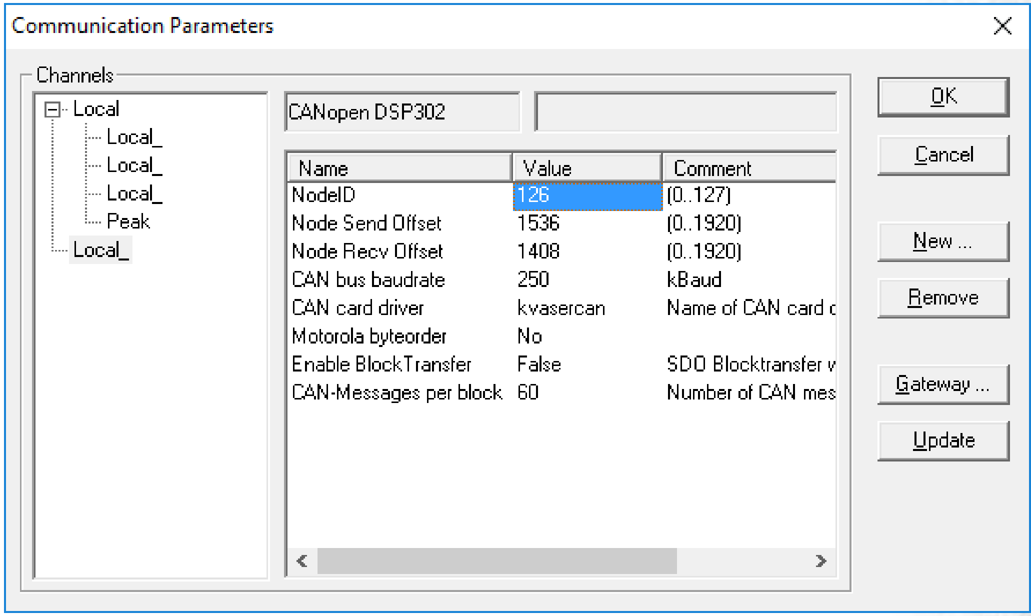
CODESYS Node-ID
126

Serial Number
17278017

Firmware
1.00168.02000.0C

Define the Communication Parameters

- Open CODESYS project
- Go to **Online > Communication Parameters**
  - Set *NodeID* to be the CODESYS node-ID
  - Check *CAN bus baudrate* (250 kbit/s)
  - Set *CAN card driver*
    - Kvaser CAN cards → kvasercan
    - Peak → peakcan
    - IXXAT → ixxatvci
    - Vector → vectorcan_chx

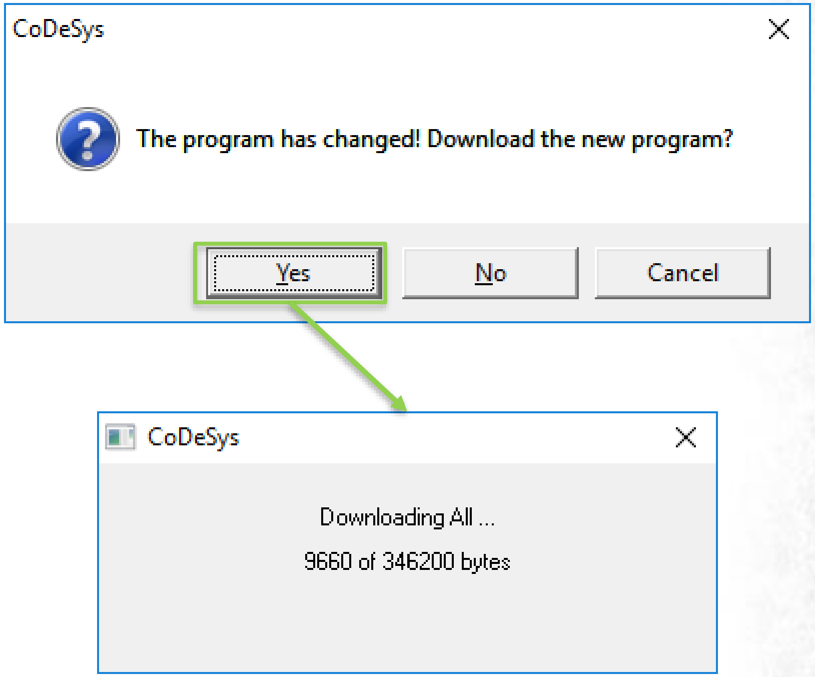


CANopen DSP302

Name	Value	Comment
NodeID	126	(0..127)
Node Send Offset	1536	(0..1920)
Node Recv Offset	1408	(0..1920)
CAN bus baudrate	250	kBaud
CAN card driver	kvasercan	Name of CAN card d
Motorola byteorder	No	
Enable BlockTransfer	False	SDO Blocktransfer v
CAN-Messages per block	60	Number of CAN mes

Downloading the Application

- Select **Online** > **Login**
- Select **Yes** when CODESYS asks if download should be done
- After download, select **Online** > **Run (F5)**



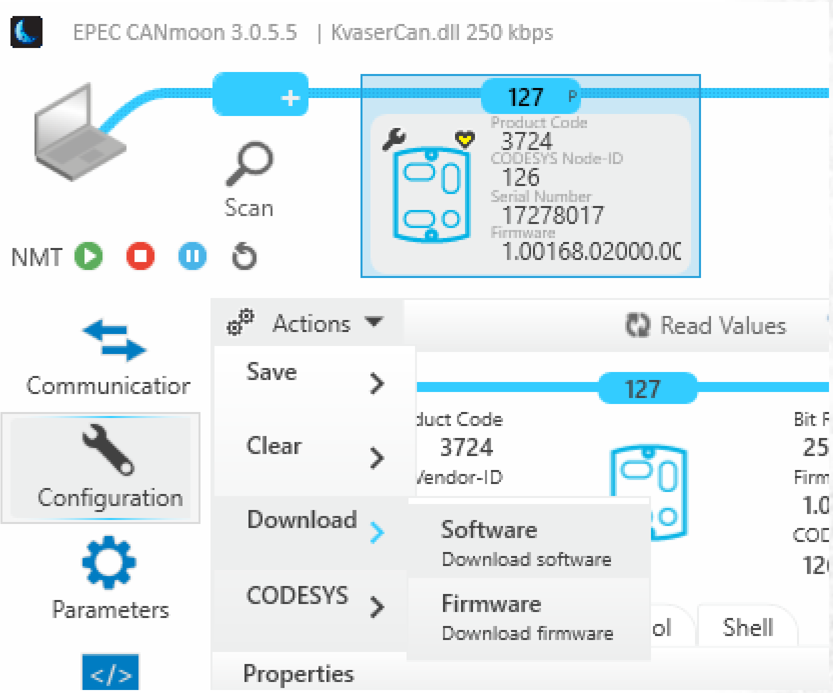
The program has changed! Download the new program?

Downloading All ...

9660 of 346200 bytes

Downloading the Object Pool

1. Open Epec CANmoon
2. Scan the CAN bus
3. Double-click the found ISOBUS unit
4. Select **Actions** > **Download** > **Download Software**
5. Select **...** To browse to the project folder > **ISOBUS** > **Python** > **BinaryMaker**
6. Select the *downloaded.bin* and **OK**
7. Select **Run** from CANmoon
8. After download, reboot the control unit



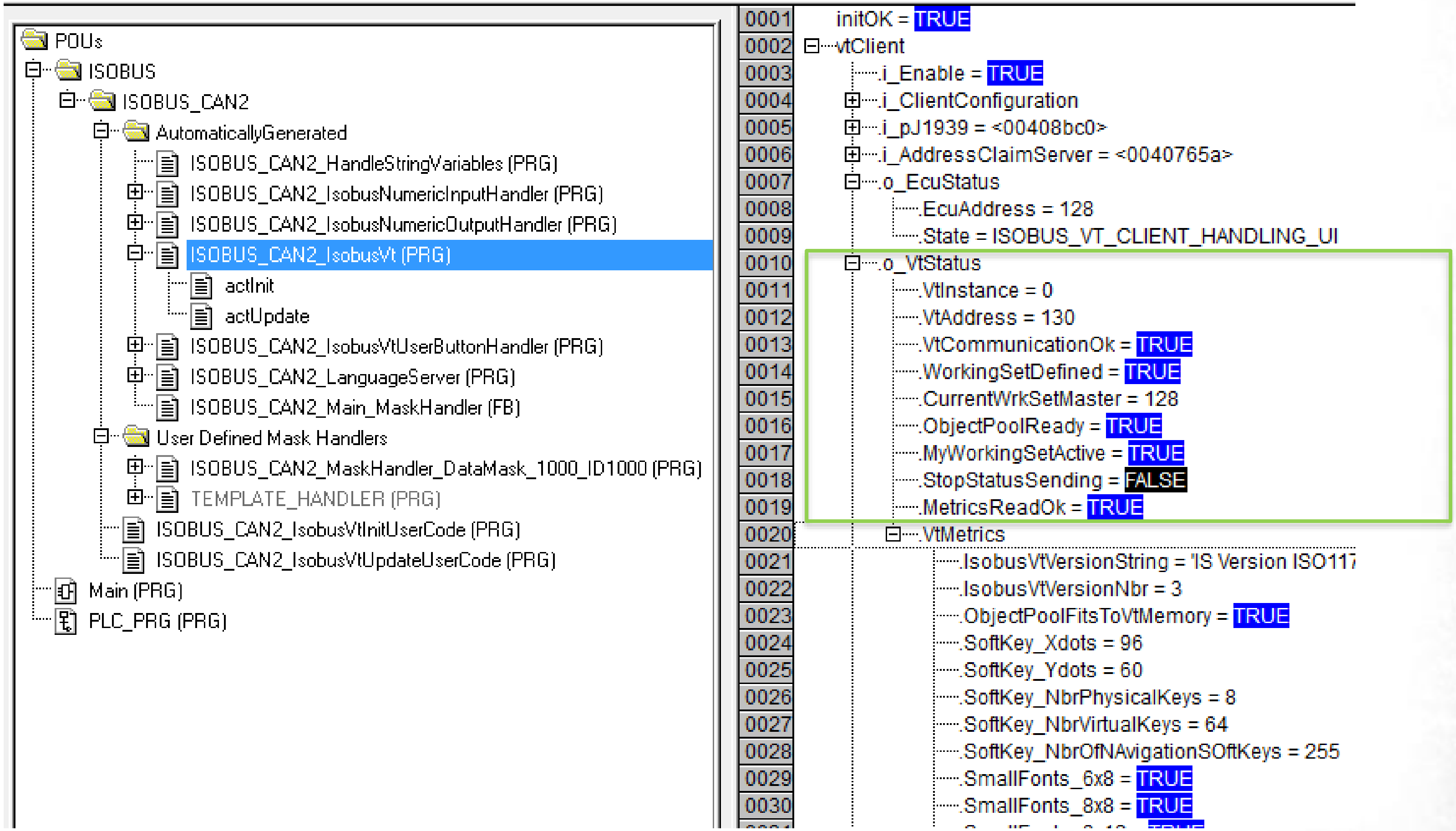
EPEC CANmoon 3.0.5.5 | KvaserCan.dll 250 kbps

Product Code 3724
CODESYS Node-ID 126
Serial Number 17278017
Firmware 1.00168.02000.0C

Testing the Connection with VT

- After download and reboot the unit, the object pool is automatically downloaded to the VT
- To check the VT client status, go online with CODESYS (**Online** > **Login**)
- Double-click *ISOBUS_CANx_IsobusVt* program open
  - Output *o_VtStatus* provides information about VT communication information
    - Is the communication working? (*VtCommunicationOk*)
    - Is the object pool working? (*ObjectPoolReady*)
    - Has the VT metrics read succesfully? (*VtMetricsOk*)

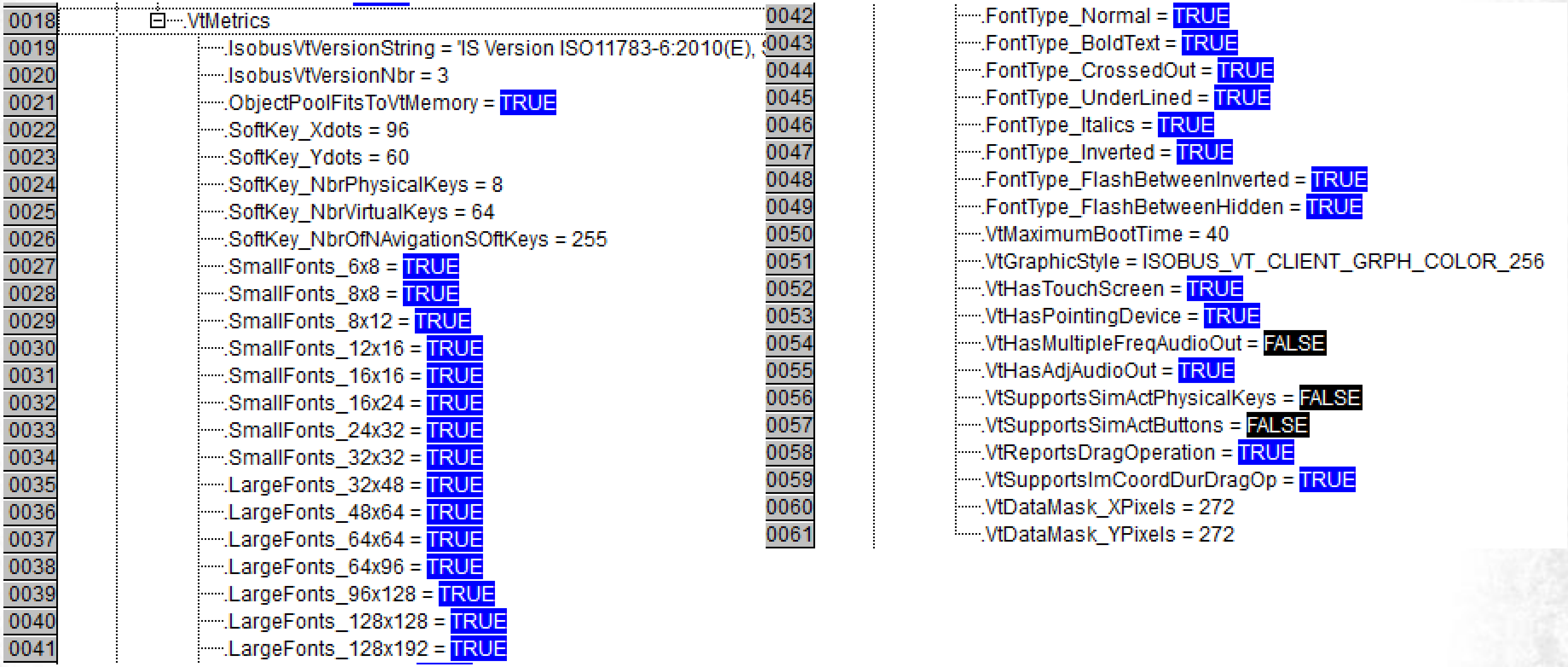
ISOBUS_CAN2_IsobusVt



```
POUs
  ISOBUS
    ISOBUS_CAN2
      AutomaticallyGenerated
        ISOBUS_CAN2_HandleStringVariables (PRG)
        ISOBUS_CAN2_IsobusNumericInputHandler (PRG)
        ISOBUS_CAN2_IsobusNumericOutputHandler (PRG)
        ISOBUS_CAN2_IsobusVt (PRG)
          actInit
          actUpdate
        ISOBUS_CAN2_IsobusVtUserButtonHandler (PRG)
        ISOBUS_CAN2_LanguageServer (PRG)
        ISOBUS_CAN2_Main_MaskHandler (FB)
      User Defined Mask Handlers
        ISOBUS_CAN2_MaskHandler_DataMask_1000_ID1000 (PRG)
        TEMPLATE_HANDLER (PRG)
      ISOBUS_CAN2_IsobusVtInitUserCode (PRG)
      ISOBUS_CAN2_IsobusVtUpdateUserCode (PRG)
  Main (PRG)
  PLC_PRG (PRG)
```

```
initOK = TRUE
vtClient
  .i_Enable = TRUE
  .i_ClientConfiguration
  .i_pJ1939 = <00408bc0>
  .i_AddressClaimServer = <0040765a>
  .o_EcuStatus
    .EcuAddress = 128
    .State = ISOBUS_VT_CLIENT_HANDLING_UI
  .o_VtStatus
    .VtInstance = 0
    .VtAddress = 130
    .VtCommunicationOk = TRUE
    .WorkingSetDefined = TRUE
    .CurrentWrkSetMaster = 128
    .ObjectPoolReady = TRUE
    .MyWorkingSetActive = TRUE
    .StopStatusSending = FALSE
    .MetricsReadOk = TRUE
    .VtMetrics
      .IsobusVtVersionString = 'IS Version ISO117
      .IsobusVtVersionNbr = 3
      .ObjectPoolFitsToVtMemory = TRUE
      .SoftKey_Xdots = 96
      .SoftKey_Ydots = 60
      .SoftKey_NbrPhysicalKeys = 8
      .SoftKey_NbrVirtualKeys = 64
      .SoftKey_NbrOfNAvigationSOftKeys = 255
      .SmallFonts_6x8 = TRUE
      .SmallFonts_8x8 = TRUE
```

VT Server Metrics



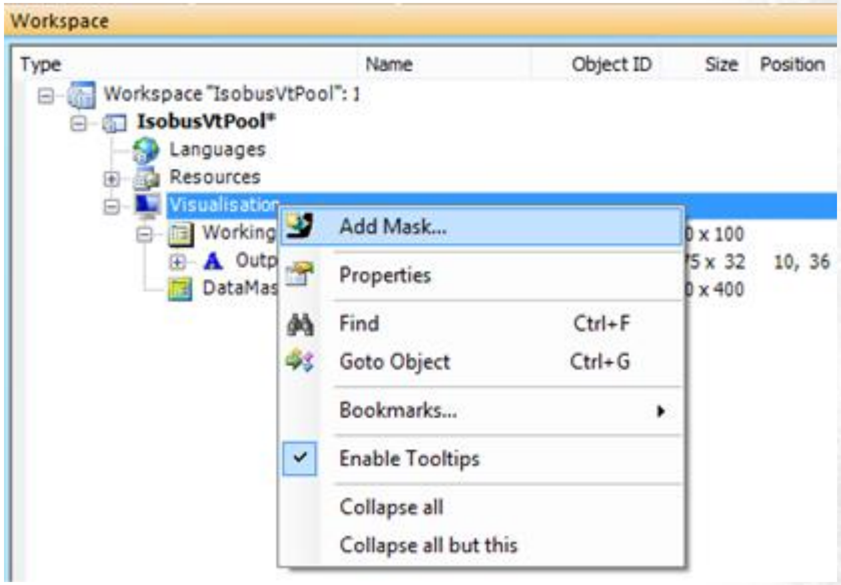
```
.VtMetrics
    .IsobusVtVersionString = 'IS Version ISO11783-6:2010(E), 
    .IsobusVtVersionNbr = 3
    .ObjectPoolFitsToVtMemory = TRUE
    .SoftKey_Xdots = 96
    .SoftKey_Ydots = 60
    .SoftKey_NbrPhysicalKeys = 8
    .SoftKey_NbrVirtualKeys = 64
    .SoftKey_NbrOfNAvigationSOftKeys = 255
    .SmallFonts_6x8 = TRUE
    .SmallFonts_8x8 = TRUE
    .SmallFonts_8x12 = TRUE
    .SmallFonts_12x16 = TRUE
    .SmallFonts_16x16 = TRUE
    .SmallFonts_16x24 = TRUE
    .SmallFonts_24x32 = TRUE
    .SmallFonts_32x32 = TRUE
    .LargeFonts_32x48 = TRUE
    .LargeFonts_48x64 = TRUE
    .LargeFonts_64x64 = TRUE
    .LargeFonts_64x96 = TRUE
    .LargeFonts_96x128 = TRUE
    .LargeFonts_128x128 = TRUE
    .LargeFonts_128x192 = TRUE
    .FontType_Normal = TRUE
    .FontType_BoldText = TRUE
    .FontType_CrossedOut = TRUE
    .FontType_UnderLined = TRUE
    .FontType_Italics = TRUE
    .FontType_Inverted = TRUE
    .FontType_FlashBetweenInverted = TRUE
    .FontType_FlashBetweenHidden = TRUE
    .VtMaximumBootTime = 40
    .VtGraphicStyle = ISOBUS_VT_CLIENT_GRPH_COLOR_256
    .VtHasTouchScreen = TRUE
    .VtHasPointingDevice = TRUE
    .VtHasMultipleFreqAudioOut = FALSE
    .VtHasAdjAudioOut = TRUE
    .VtSupportsSimActPhysicalKeys = FALSE
    .VtSupportsSimActButtons = FALSE
    .VtReportsDragOperation = TRUE
    .VtSupportsImCoordDurDragOp = TRUE
    .VtDataMask_XPixels = 272
    .VtDataMask_YPixels = 272
```

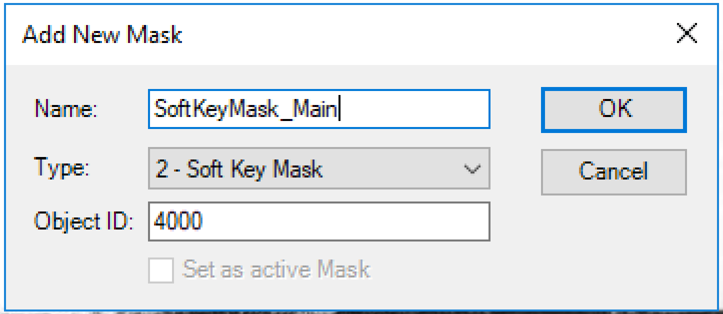
ISO-Designer

Graphical editor for ISOBUS compliant files

Adding Softkey Mask

1. Open the ISO-Designer template project from path *..\ISOBUS\Jetter\IsobusVtPool.jvw*
2. Open *Workspace* view, right-click *Visualisation* and select **Add Mask**
3. Add a softkey mask
   - Type → **2 – SoftKeyMask**
   - Name → **SoftKeyMask_Main**





Name: SoftKeyMask_Main

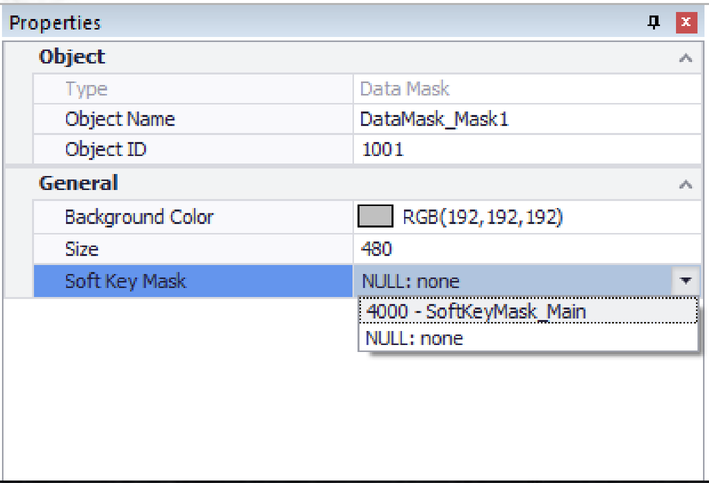
Type: 2 - Soft Key Mask

Object ID: 4000

□ Set as active Mask

Adding Data Masks

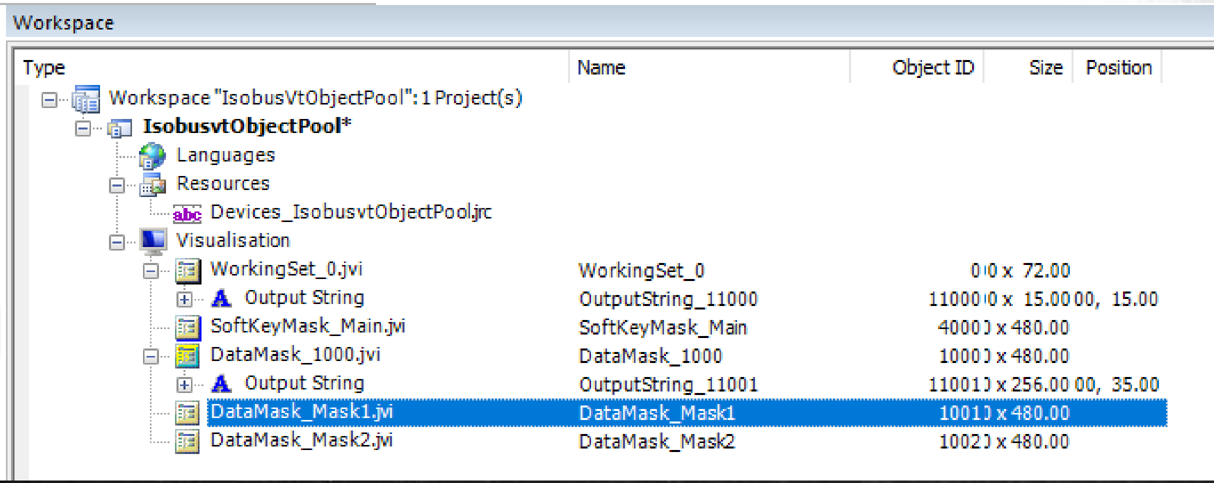
4. Add two data masks to the project:
   - *DataMask_Mask1*
   - *DataMask_Mask2*
5. Select *DataMask_Mask1* to see its **Properties**
6. Define the used **Soft Key Mask** to *SoftKeyMask_Main*
7. Repeat steps with *DataMask_Mask2*



Properties

Object	
Type	Data Mask
Object Name	DataMask_Mask1
Object ID	1001
General	
Background Color	RGB(192,192,192)
Size	480
Soft Key Mask	NULL: none

4000 - SoftKeyMask_Main
NULL: none

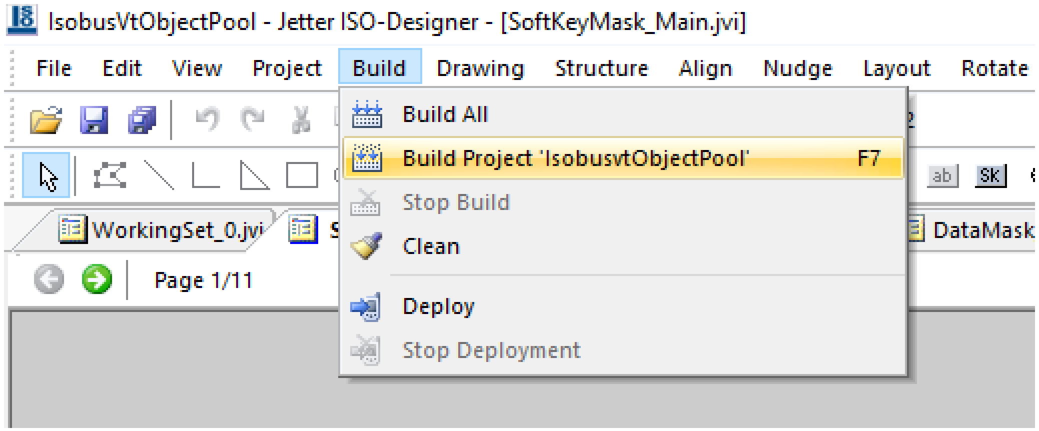


Workspace

Type	Name	Object ID	Size	Position
Workspace "IsobusVtObjectPool": 1 Project(s)				
IsobusvtObjectPool*				
Languages				
Resources				
Devices_IsobusvtObjectPool.jrc				
Visualisation				
WorkingSet_0.jvi	WorkingSet_0	0	0 x 72.00	
Output String	OutputString_11000	11000	0 x 15.00	00, 15.00
SoftKeyMask_Main.jvi	SoftKeyMask_Main	4000	0 x 480.00	
DataMask_1000.jvi	DataMask_1000	1000	0 x 480.00	
Output String	OutputString_11001	11001	0 x 256.00	00, 35.00
DataMask_Mask1.jvi	DataMask_Mask1	1001	0 x 480.00	
DataMask_Mask2.jvi	DataMask_Mask2	1002	0 x 480.00	

Build ISO-Designer Project

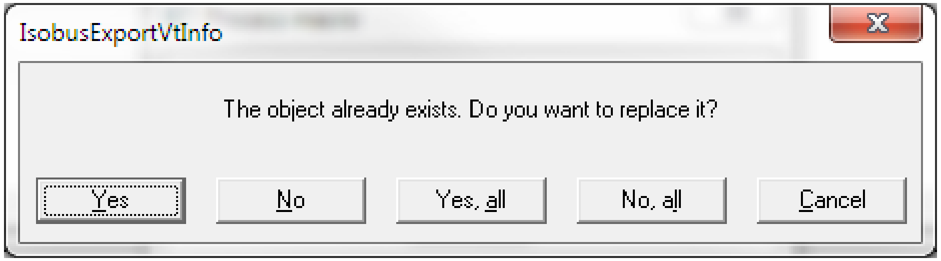
- To build the ISO-Designer project, select **Build > Build Project ...**



Update changes to the CODESYS project

To update ISO-Designer changes to the CODESYS project:

1. Open the CODESYS project
2. Select **Edit** > **Macros** > **Import ISOBUS**
   - Select **Yes, all** to CODESYS popup

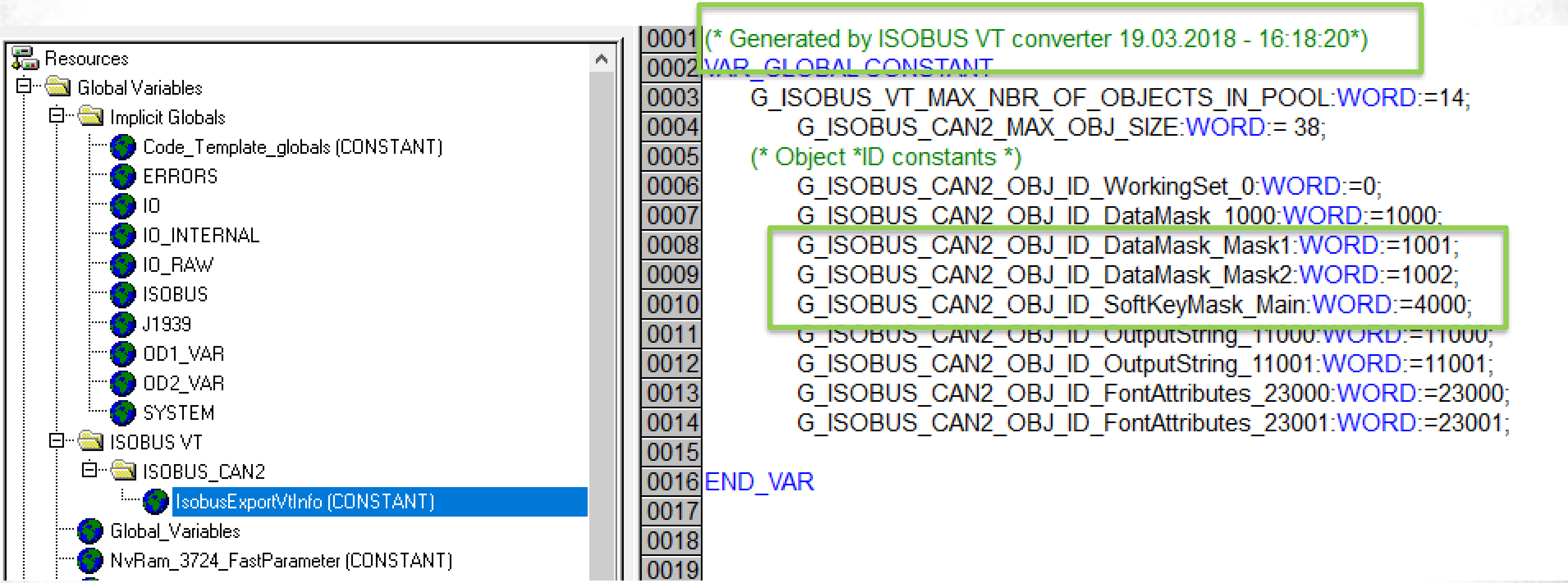


IsobusExportVtInfo

The object already exists. Do you want to replace it?

Update changes to the CODESYS project

- All object pool elements are listed in a global constant variable list *IsobusExportVtInfo*



- Resources
  - Global Variables
    - Implicit Globals
      - Code_Template_globals (CONSTANT)
      - ERRORS
      - IO
      - IO_INTERNAL
      - IO_RAW
      - ISOBUS
      - J1939
      - OD1_VAR
      - OD2_VAR
      - SYSTEM
    - ISOBUS VT
      - ISOBUS_CAN2
        - IsobusExportVtInfo (CONSTANT)
    - Global_Variables
    - NvRam_3724_FastParameter (CONSTANT)

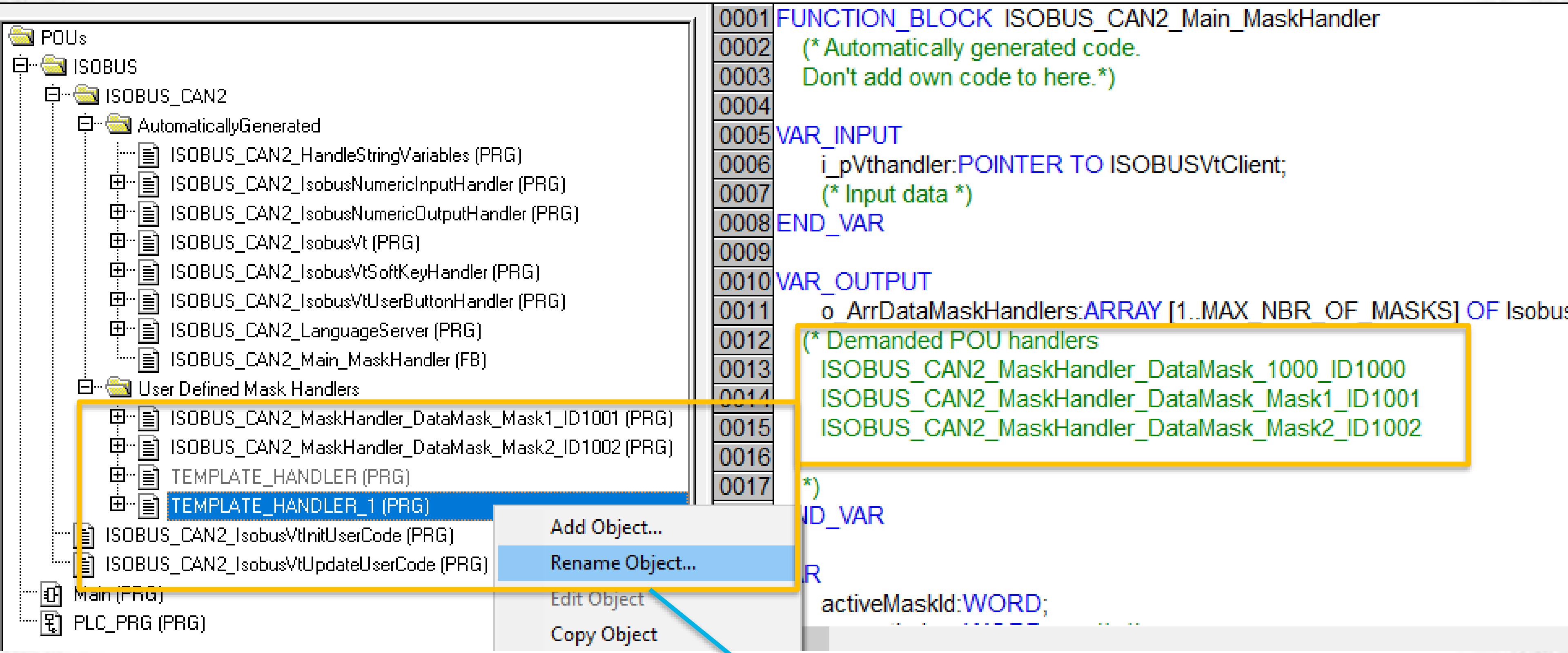
```
(* Generated by ISOBUS VT converter 19.03.2018 - 16:18:20*)
VAR_GLOBAL CONSTANT
	G_ISOBUS_VT_MAX_NBR_OF_OBJECTS_IN_POOL:WORD:=14;
		G_ISOBUS_CAN2_MAX_OBJ_SIZE:WORD:= 38;
	(* Object *ID constants *)
		G_ISOBUS_CAN2_OBJ_ID_WorkingSet_0:WORD:=0;
		G_ISOBUS_CAN2_OBJ_ID_DataMask_1000:WORD:=1000;
		G_ISOBUS_CAN2_OBJ_ID_DataMask_Mask1:WORD:=1001;
		G_ISOBUS_CAN2_OBJ_ID_DataMask_Mask2:WORD:=1002;
		G_ISOBUS_CAN2_OBJ_ID_SoftKeyMask_Main:WORD:=4000;
		G_ISOBUS_CAN2_OBJ_ID_OutputString_11000:WORD:=11000;
		G_ISOBUS_CAN2_OBJ_ID_OutputString_11001:WORD:=11001;
		G_ISOBUS_CAN2_OBJ_ID_FontAttributes_23000:WORD:=23000;
		G_ISOBUS_CAN2_OBJ_ID_FontAttributes_23001:WORD:=23001;

END_VAR
```

Update changes to the CODESYS project

3. To add handler programs for *DataMask_1000*, *DataMask_Mask1* and *DataMask_Mask2*, copy-paste *TEMPLATE_HANDLER* program three times
4. Rename the copied programs with POU names that are already given in *ISOBUS_CAN1_Main_MaskHandler*
   - *ISOBUS_CAN2_MaskHandler_DataMask_1000_ID1000*
   - *ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001*
   - *ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002*

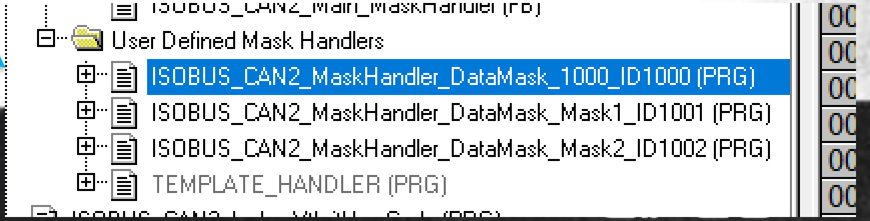
- This needs to be done every time when new masks are added

Update changes to the CODESYS project



```
POUs
ISOBUS
  ISOBUS_CAN2
    AutomaticallyGenerated
      ISOBUS_CAN2_HandleStringVariables (PRG)
      ISOBUS_CAN2_IsobusNumericInputHandler (PRG)
      ISOBUS_CAN2_IsobusNumericOutputHandler (PRG)
      ISOBUS_CAN2_IsobusVt (PRG)
      ISOBUS_CAN2_IsobusVtSoftKeyHandler (PRG)
      ISOBUS_CAN2_IsobusVtUserButtonHandler (PRG)
      ISOBUS_CAN2_LanguageServer (PRG)
      ISOBUS_CAN2_Main_MaskHandler (FB)
    User Defined Mask Handlers
      ISOBUS_CAN2_MaskHandler_DataMask_Mask1_ID1001 (PRG)
      ISOBUS_CAN2_MaskHandler_DataMask_Mask2_ID1002 (PRG)
      TEMPLATE_HANDLER (PRG)
      TEMPLATE_HANDLER_1 (PRG)
    ISOBUS_CAN2_IsobusVtInitUserCode (PRG)
    ISOBUS_CAN2_IsobusVtUpdateUserCode (PRG)
Main (PRG)
PLC_PRG (PRG)
```

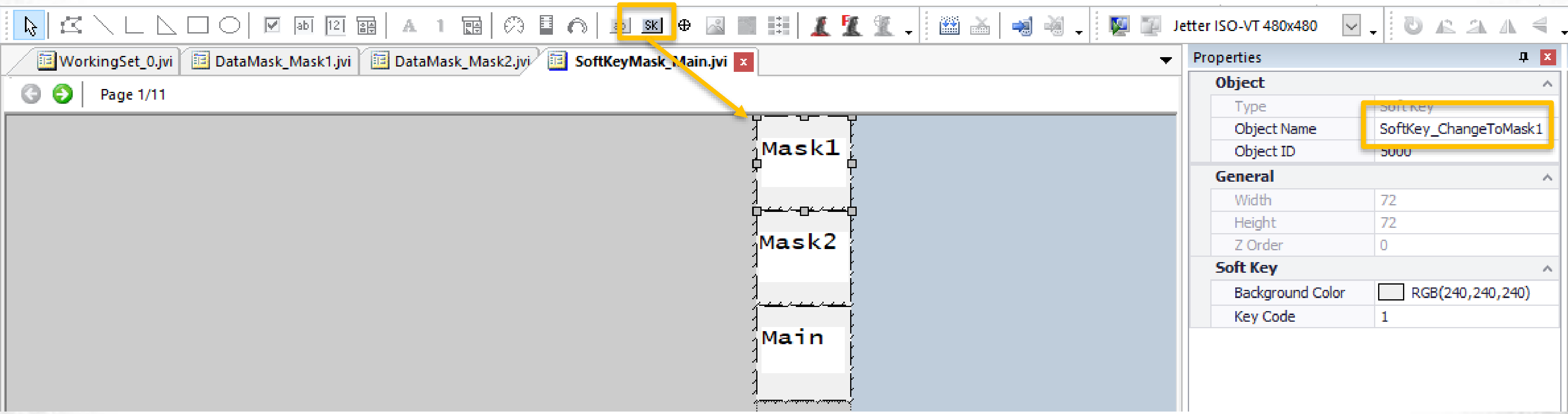
```
FUNCTION_BLOCK  ISOBUS_CAN2_Main_MaskHandler
   (* Automatically generated code.
   Don't add own code to here.*)

VAR_INPUT
     i_pVthandler:POINTER TO ISOBUSVtClient;
     (* Input data *)
END_VAR

VAR_OUTPUT
     o_ArrDataMaskHandlers:ARRAY [1..MAX_NBR_OF_MASKS] OF Isobus
   (* Demanded POU handlers
     ISOBUS_CAN2_MaskHandler_DataMask_1000_ID1000
     ISOBUS_CAN2_MaskHandler_DataMask_Mask1_ID1001
     ISOBUS_CAN2_MaskHandler_DataMask_Mask2_ID1002

   *)
ND_VAR

R
     activeMaskId:WORD;
```



```
ISOBUS_CAN2_Main_MaskHandler (FB)
User Defined Mask Handlers
  ISOBUS_CAN2_MaskHandler_DataMask_1000_ID1000 (PRG)
  ISOBUS_CAN2_MaskHandler_DataMask_Mask1_ID1001 (PRG)
  ISOBUS_CAN2_MaskHandler_DataMask_Mask2_ID1002 (PRG)
  TEMPLATE_HANDLER (PRG)
```

Adding Softkeys

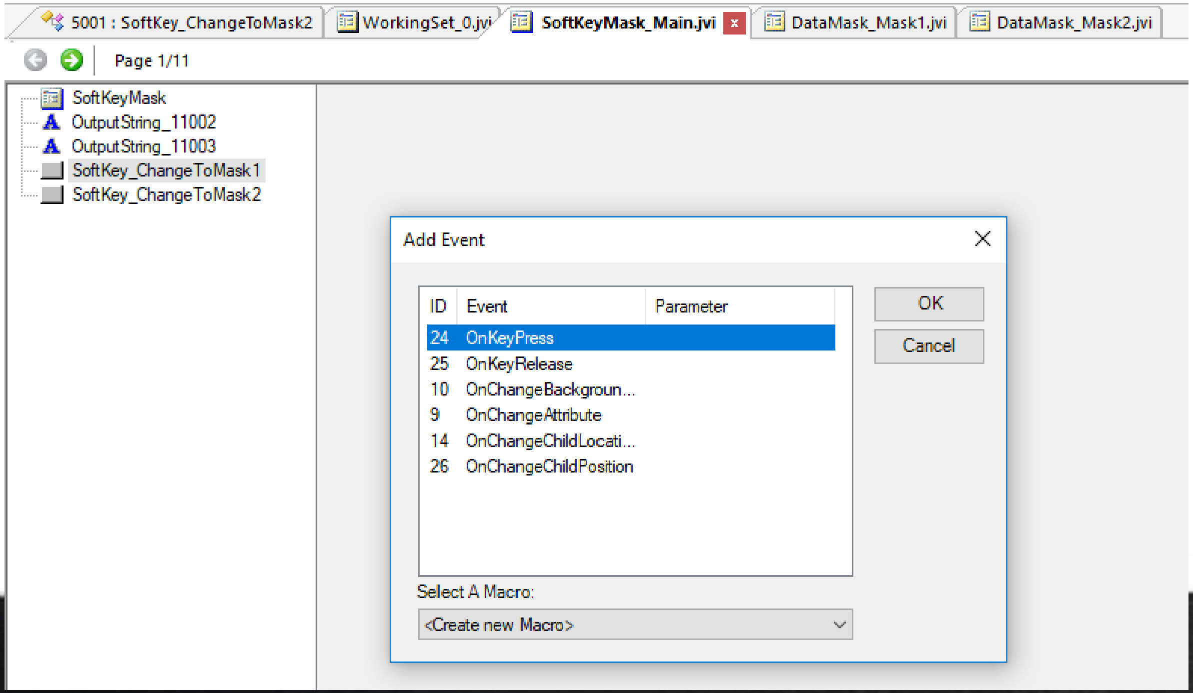
- Open *SoftKeyMask_Main* and add three softkeys
- Add output strings to softkeys with Values *Mask1*, *Mask2*, *Main*
- Name softkeys in Workspace view as *SoftKey_ChangeToMask1* and *SoftKey_ChangeToMask2*



Object	
Type	Soft Key
Object Name	SoftKey_ChangeToMask1
Object ID	5000
General	
Width	72
Height	72
Z Order	0
Soft Key	
Background Color	RGB(240,240,240)
Key Code	1

Transitions Between DataMasks

- Select *SoftKeyMask_Main* > **Event Handler** tab
- Right-click on *SoftKey_ChangeToMask1* and select **Add Event > OnKeyPress**
- Open **Command** list and select **ChangeActiveMask**



SoftKeyMask
OutputString_11002
OutputString_11003
SoftKey_ChangeToMask1
SoftKey_ChangeToMask2

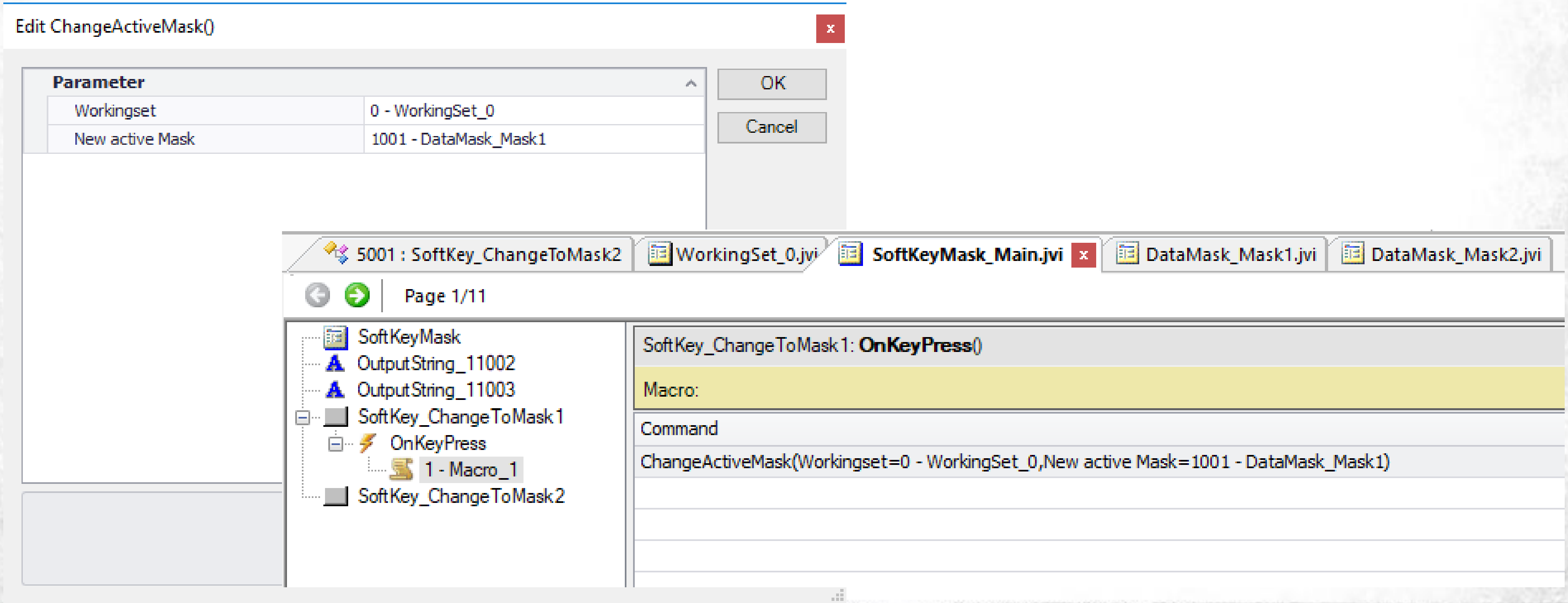
Add Event

ID	Event	Parameter
24	OnKeyPress	
25	OnKeyRelease	
10	OnChangeBackgroun...	
9	OnChangeAttribute	
14	OnChangeChildLocati...	
26	OnChangeChildPosition	

Select A Macro: <Create new Macro>

Transitions Between DataMasks

- Open Command list and select ChangeActiveMask
- Give needed parameters



Edit ChangeActiveMask()

Parameter	
Workingset	0 - WorkingSet_0
New active Mask	1001 - DataMask_Mask1

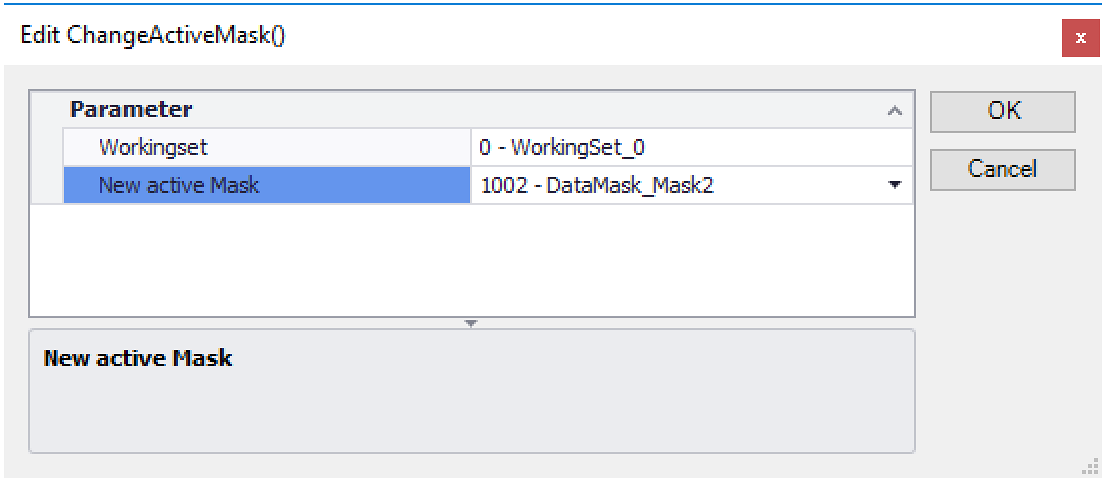
SoftKey_ChangeToMask1: **OnKeyPress**()

Macro:

Command
ChangeActiveMask(Workingset=0 - WorkingSet_0,New active Mask=1001 - DataMask_Mask1)

Transitions Between DataMasks

- Set the **Workingset** and **New active Mask** parameters
- Repeat for *SoftKey_ChangeToMask2*



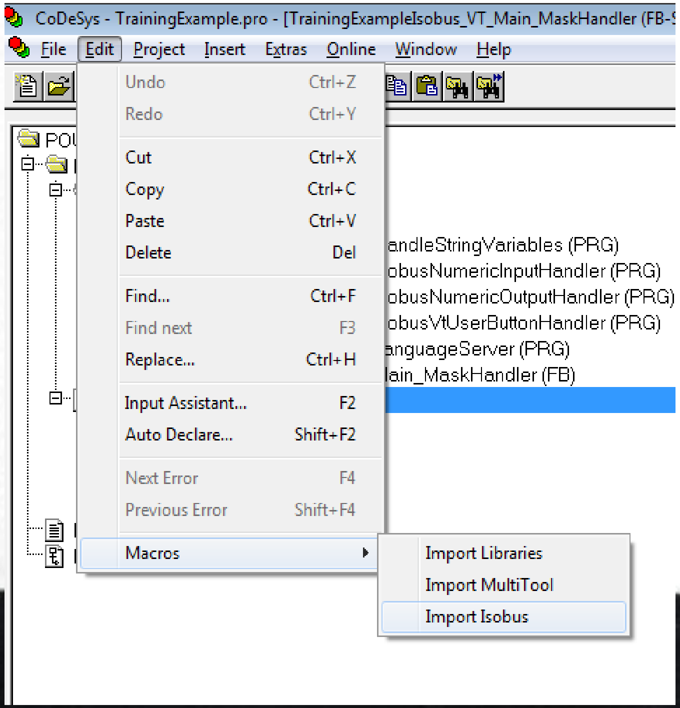
Edit ChangeActiveMask()

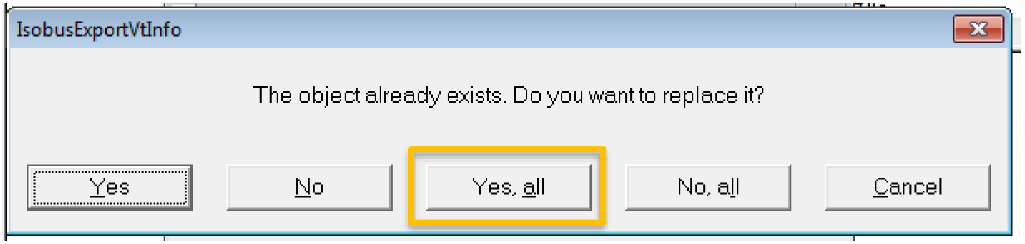
Parameter	
Workingset	0 - WorkingSet_0
New active Mask	1002 - DataMask_Mask2

New active Mask

Update CODESYS Project

1. Build the ISO-Designer project (**Build → Build project ”IsobusVtObjectPool”**)
2. Open **CODESYS** and select **Edit > Macros > Import Isobus**
3. Click **Yes, all** when CODESYS wants to overwrite objects

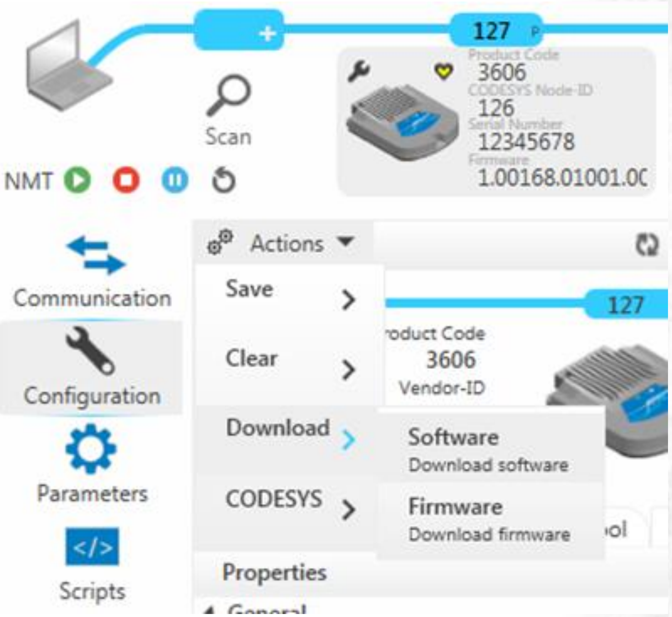

CoDeSys - TrainingExample.pro - [TrainingExampleIsobus_VT_Main_MaskHandler (FB-S


IsobusExportVtInfo

The object already exists. Do you want to replace it?

Downloading the Application

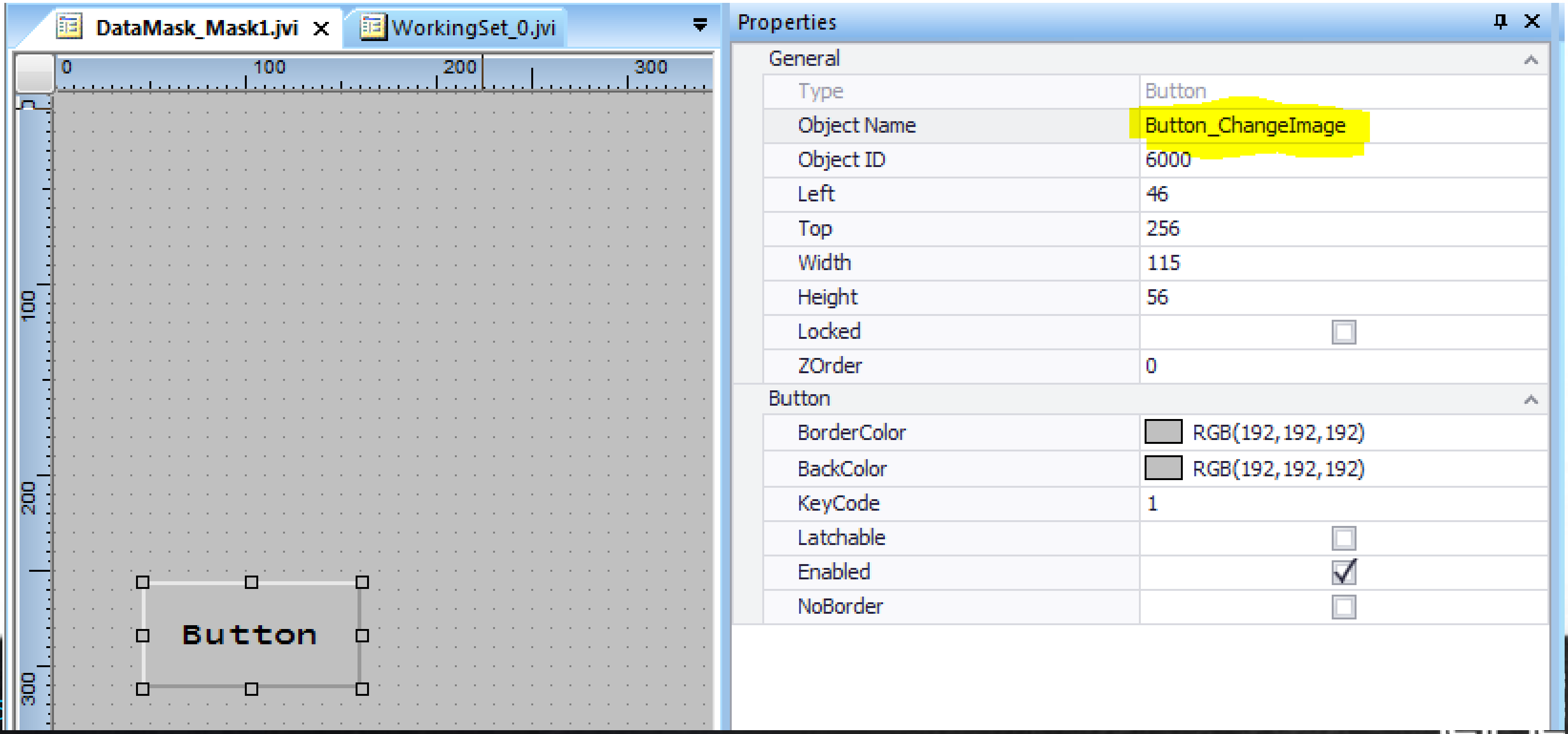
- Download the CODESYS application normally via CODESYS or CANmoon
- ISOBUS pool binary is downloaded with CANmoon:
  - Scan CAN bus and double-click the unit and select **Actions>Download>Software**
  - Select object pool binary *downloaded.bin* from {DeviceFolder}*ISOBUS\\Python\\Binary Maker*
  - Click **Run** → CANmoon downloads the binary to the unit
  - Reboot the unit



Product Code 3606
CODESYS Node-ID 126
Serial Number 12345678
Firmware 1.00168.01001.0C

Adding a Button

- Add a button to *DataMask_Mask1*
- Rename the button's **Object Name** to *Button_ChangeImage*
- Build the project and import changes to CODESYS

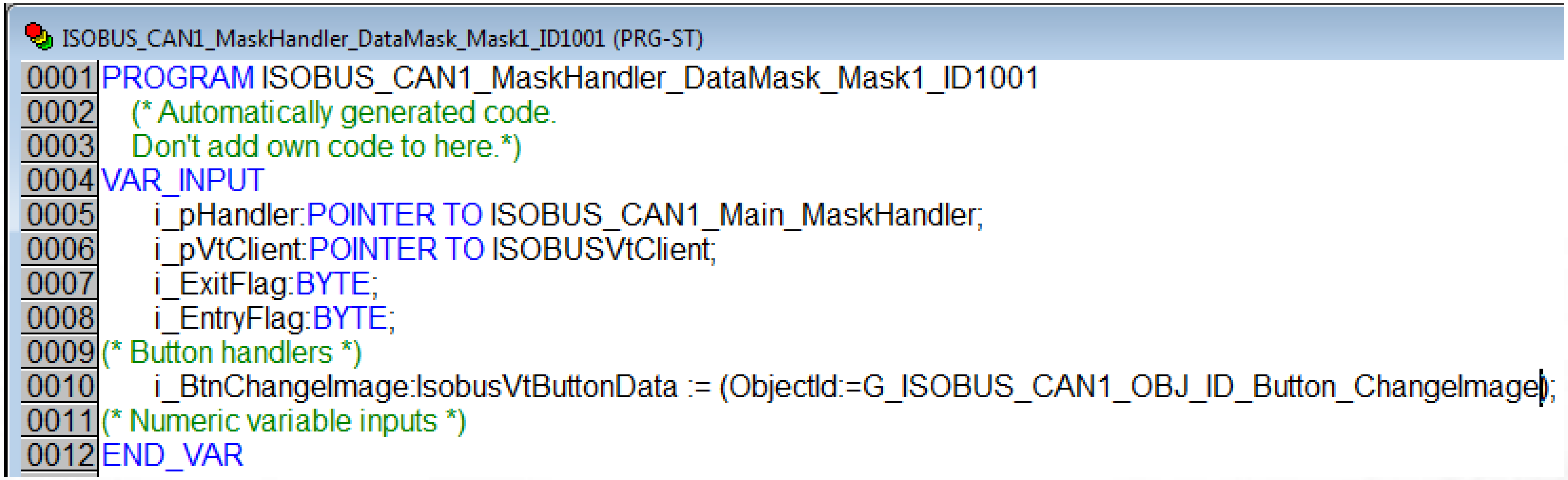


Properties

General	
Type	Button
Object Name	Button_ChangeImage
Object ID	6000
Left	46
Top	256
Width	115
Height	56
Locked	□
ZOrder	0
Button	
BorderColor	RGB(192,192,192)
BackColor	RGB(192,192,192)
KeyCode	1
Latchable	□
Enabled	☑
NoBorder	□

Adding a Button Handler

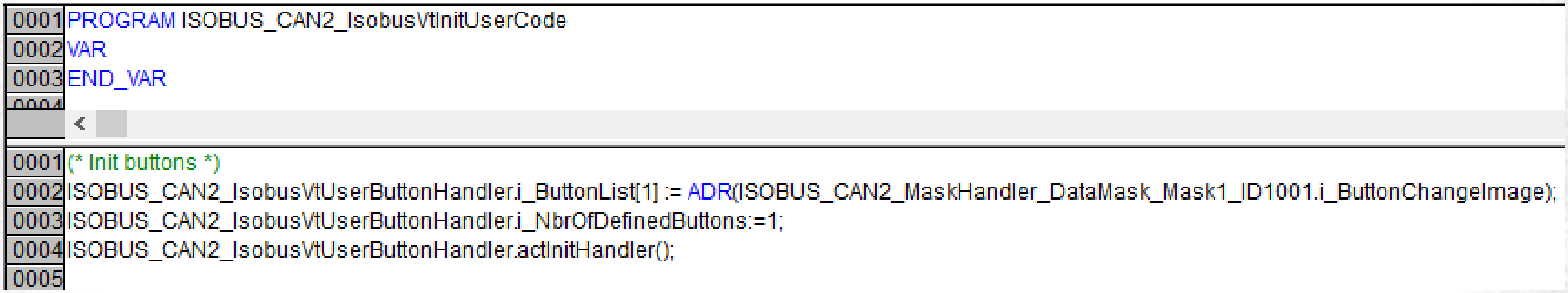
- Buttons have their own structure *IsobusVtButtonData*
- The structure can be found from CODESYS Data types tab
- Add a button input of type *IsobusVtButtonData* to the data mask handler *ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001*



```
PROGRAM ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001
   (* Automatically generated code.
   Don't add own code to here.*)
VAR_INPUT
     i_pHandler:POINTER TO ISOBUS_CAN1_Main_MaskHandler;
     i_pVtClient:POINTER TO ISOBUSVtClient;
     i_ExitFlag:BYTE;
     i_EntryFlag:BYTE;
(* Button handlers *)
     i_BtnChangeImage:IsobusVtButtonData := (ObjectId:=G_ISOBUS_CAN1_OBJ_ID_Button_ChangeImage);
(* Numeric variable inputs *)
END_VAR
```

Adding Button Handler

- Initialize button handler in *ISOBUS_CAN1_IsobusVtInitUserCode* (PRG)
- Download the CODESYS application, update the pool binary to the unit with CANmoon and reboot the unit
    - After reboot, the new object pool is downloaded to the VT

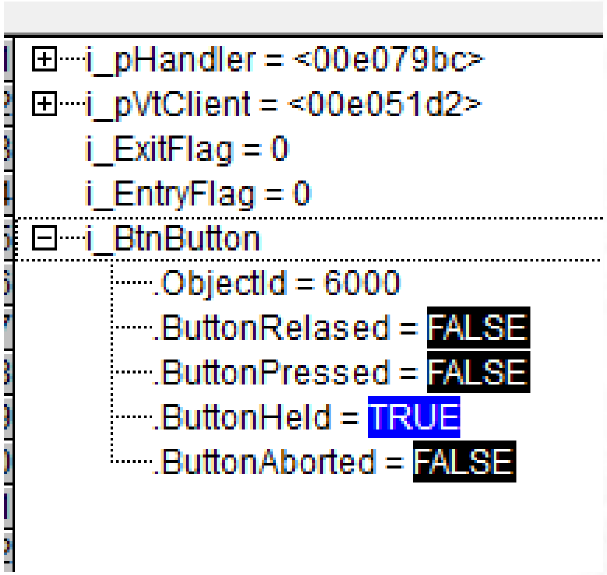


```
PROGRAM ISOBUS_CAN2_IsobusVtInitUserCode
VAR
END_VAR
```

```
(* Init buttons *)
ISOBUS_CAN2_IsobusVtUserButtonHandler.i_ButtonList[1] := ADR(ISOBUS_CAN2_MaskHandler_DataMask_Mask1_ID1001.i_ButtonChangeImage);
ISOBUS_CAN2_IsobusVtUserButtonHandler.i_NbrOfDefinedButtons:=1;
ISOBUS_CAN2_IsobusVtUserButtonHandler.actInitHandler();
```

Checking Button Action

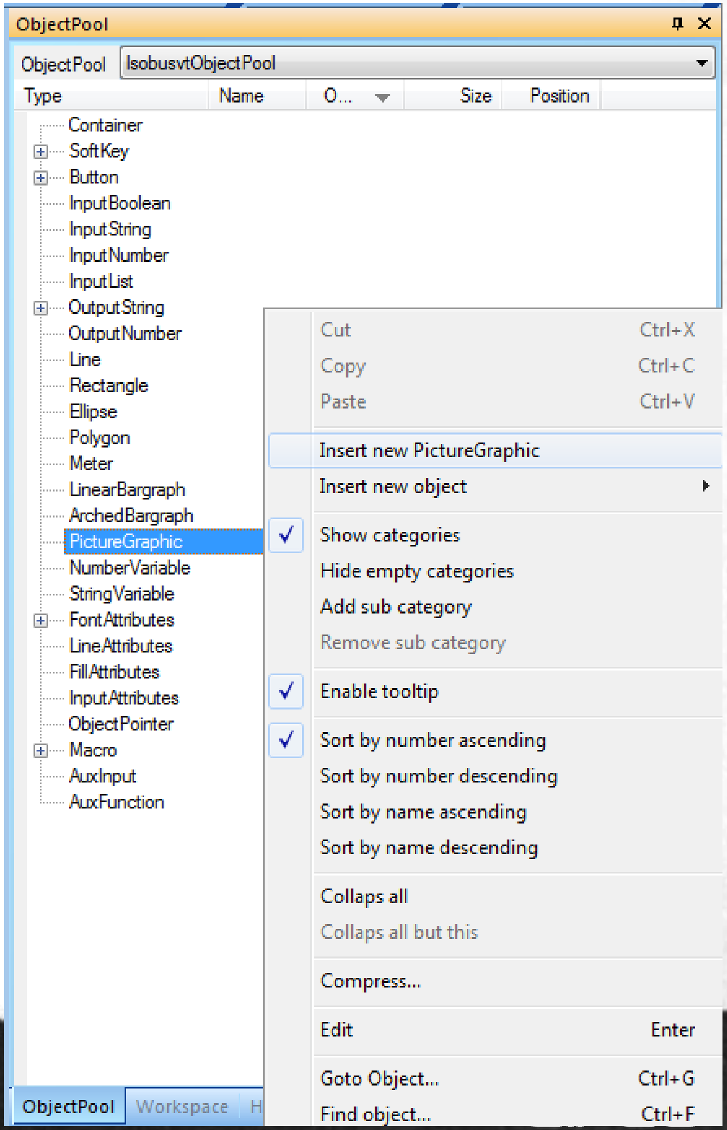
- Press the button in the terminal
- Button state change should now be seen in CODESYS variable *i_BtnChangeImage*
- Button variable keeps the latest state in CODESYS
  - **The input structure is updated when a new message is received from the VT**



```
⊞····i_pHandler = <00e079bc>
⊞····i_pVtClient = <00e051d2>
     i_ExitFlag = 0
     i_EntryFlag = 0
⊟····i_BtnButton
         ·······.ObjectId = 6000
         ·······.ButtonRelased = FALSE
         ·······.ButtonPressed = FALSE
         ·······.ButtonHeld = TRUE
         ·······.ButtonAborted = FALSE
```

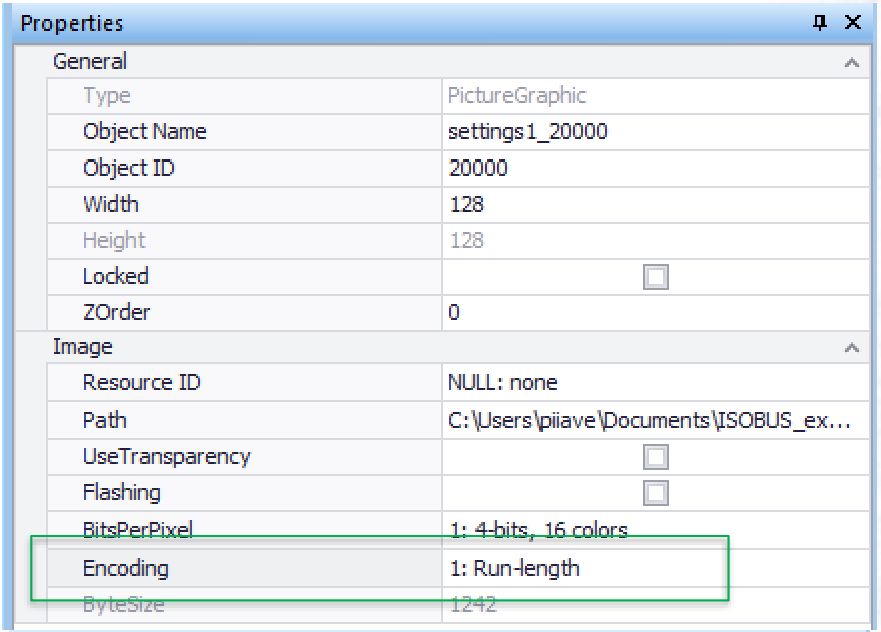
Using Object Pointer and PictureGraphic

- Add two images to the object pool and use the object pointer to change the shown image
- Choose two bitmaps
  - 16 color bitmaps recommended → works best in different VTs
- Select **ObjectPool** tab in the ISO-Designer
- Right-click **PictureGraphic**
- Select **Insert New PictureGraphic** and add images



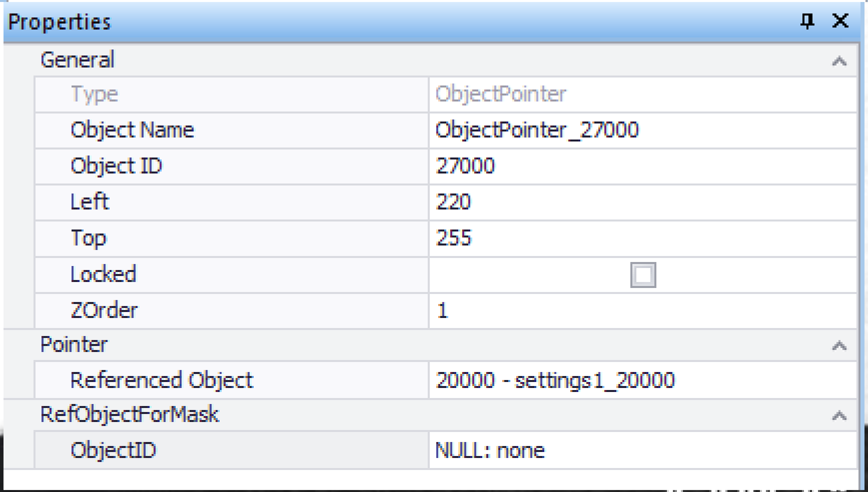
Using Object Pointer and PictureGraphic

- Select images from the ObjectPool list and select **Properties** > **Encoding** > **Run-length**
- Add an object pointer to *DataMask_Mask1*
- Go to object pointer **Properties** and add a link to one image in **Referenced Object**



Properties

General	
Type	PictureGraphic
Object Name	settings1_20000
Object ID	20000
Width	128
Height	128
Locked	□
ZOrder	0
Image	
Resource ID	NULL: none
Path	C:\Users\piiave\Documents\ISOBUS_ex...
UseTransparency	□
Flashing	□
BitsPerPixel	1: 4-bits, 16 colors
Encoding	1: Run-length
ByteSize	1242

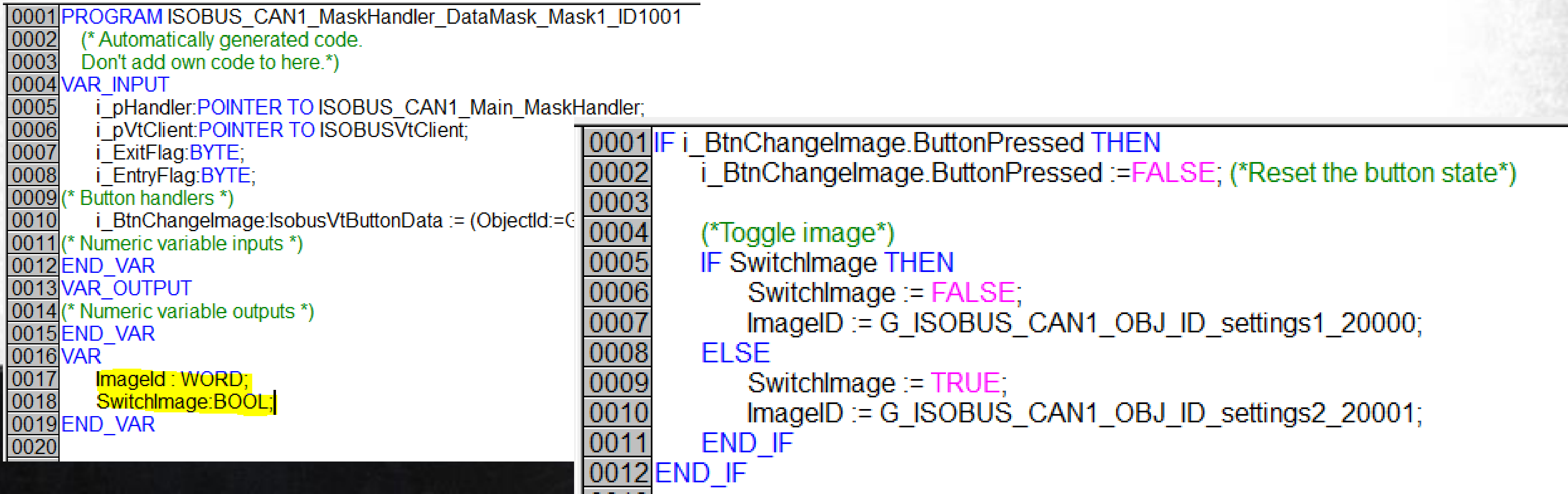


Properties

General	
Type	ObjectPointer
Object Name	ObjectPointer_27000
Object ID	27000
Left	220
Top	255
Locked	□
ZOrder	1
Pointer	
Referenced Object	20000 - settings1_20000
RefObjectForMask	
ObjectID	NULL: none

Using Object Pointer and PictureGraphic

- Open the CODESYS project and select **Edit > Macros > Import Isobus**
- Add code for the object pointer that
  - checks the button state (*i_BtnChangeImage*)
  - changes the image when the button is pressed
- *ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001 > actUserCodeMain*

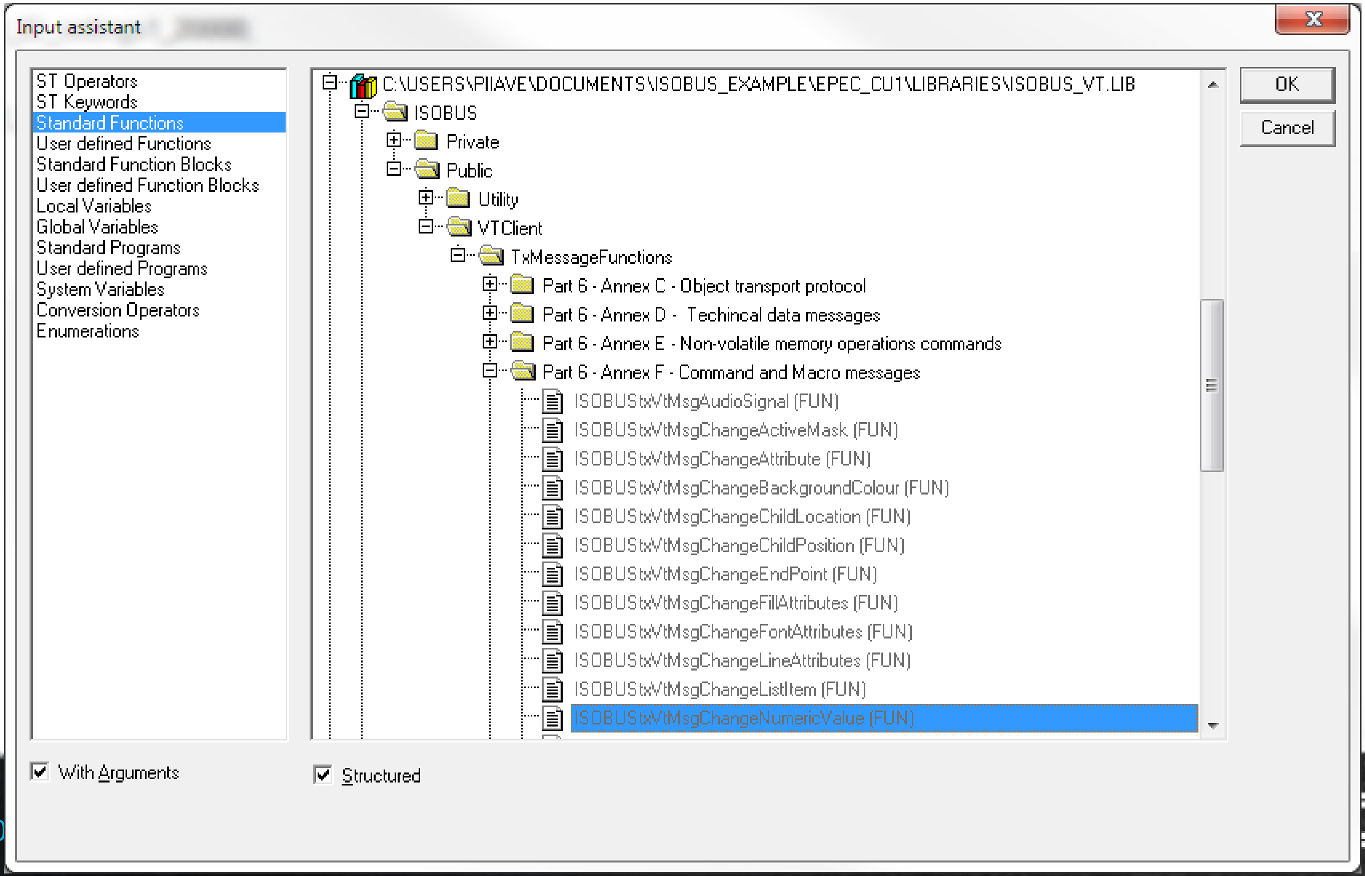


```
PROGRAM ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001
    (* Automatically generated code.
    Don't add own code to here.*)
VAR_INPUT
      i_pHandler:POINTER TO ISOBUS_CAN1_Main_MaskHandler;
      i_pVtClient:POINTER TO ISOBUSVtClient;
      i_ExitFlag:BYTE;
      i_EntryFlag:BYTE;
(* Button handlers *)
      i_BtnChangeImage:IsobusVtButtonData := (ObjectId:=G
(* Numeric variable inputs *)
END_VAR
VAR_OUTPUT
(* Numeric variable outputs *)
END_VAR
VAR
      ImageId : WORD;
      SwitchImage:BOOL;
END_VAR
```

```
IF i_BtnChangeImage.ButtonPressed THEN
    i_BtnChangeImage.ButtonPressed :=FALSE; (*Reset the button state*)

    (*Toggle image*)
    IF SwitchImage THEN
        SwitchImage := FALSE;
        ImageID := G_ISOBUS_CAN1_OBJ_ID_settings1_20000;
    ELSE
        SwitchImage := TRUE;
        ImageID := G_ISOBUS_CAN1_OBJ_ID_settings2_20001;
    END_IF
END_IF
```

Using Object Pointer and PictureGraphic

- The changed object pointer value (new image ID) is done with function *ISOBUSStxVtMsgChangeNumericValue* from ISOBUS_VT.lib



Input assistant [redacted]

ST Operators
ST Keywords
Standard Functions
User defined Functions
Standard Function Blocks
User defined Function Blocks
Local Variables
Global Variables
Standard Programs
User defined Programs
System Variables
Conversion Operators
Enumerations

```
⊟ C:\USERS\PIIAVE\DOCUMENTS\ISOBUS_EXAMPLE\EPEC_CU1\LIBRARIES\ISOBUS_VT.LIB
   ⊟ ISOBUS
      ⊞ Private
      ⊟ Public
         ⊞ Utility
         ⊟ VTClient
            ⊟ TxMessageFunctions
               ⊞ Part 6 - Annex C - Object transport protocol
               ⊞ Part 6 - Annex D -  Techincal data messages
               ⊞ Part 6 - Annex E - Non-volatile memory operations commands
               ⊟ Part 6 - Annex F - Command and Macro messages
                  ISOBUStxVtMsgAudioSignal (FUN)
                  ISOBUStxVtMsgChangeActiveMask (FUN)
                  ISOBUStxVtMsgChangeAttribute (FUN)
                  ISOBUStxVtMsgChangeBackgroundColour (FUN)
                  ISOBUStxVtMsgChangeChildLocation (FUN)
                  ISOBUStxVtMsgChangeChildPosition (FUN)
                  ISOBUStxVtMsgChangeEndPoint (FUN)
                  ISOBUStxVtMsgChangeFillAttributes (FUN)
                  ISOBUStxVtMsgChangeFontAttributes (FUN)
                  ISOBUStxVtMsgChangeLineAttributes (FUN)
                  ISOBUStxVtMsgChangeListItem (FUN)
                  ISOBUStxVtMsgChangeNumericValue (FUN)
```

☑ With Arguments

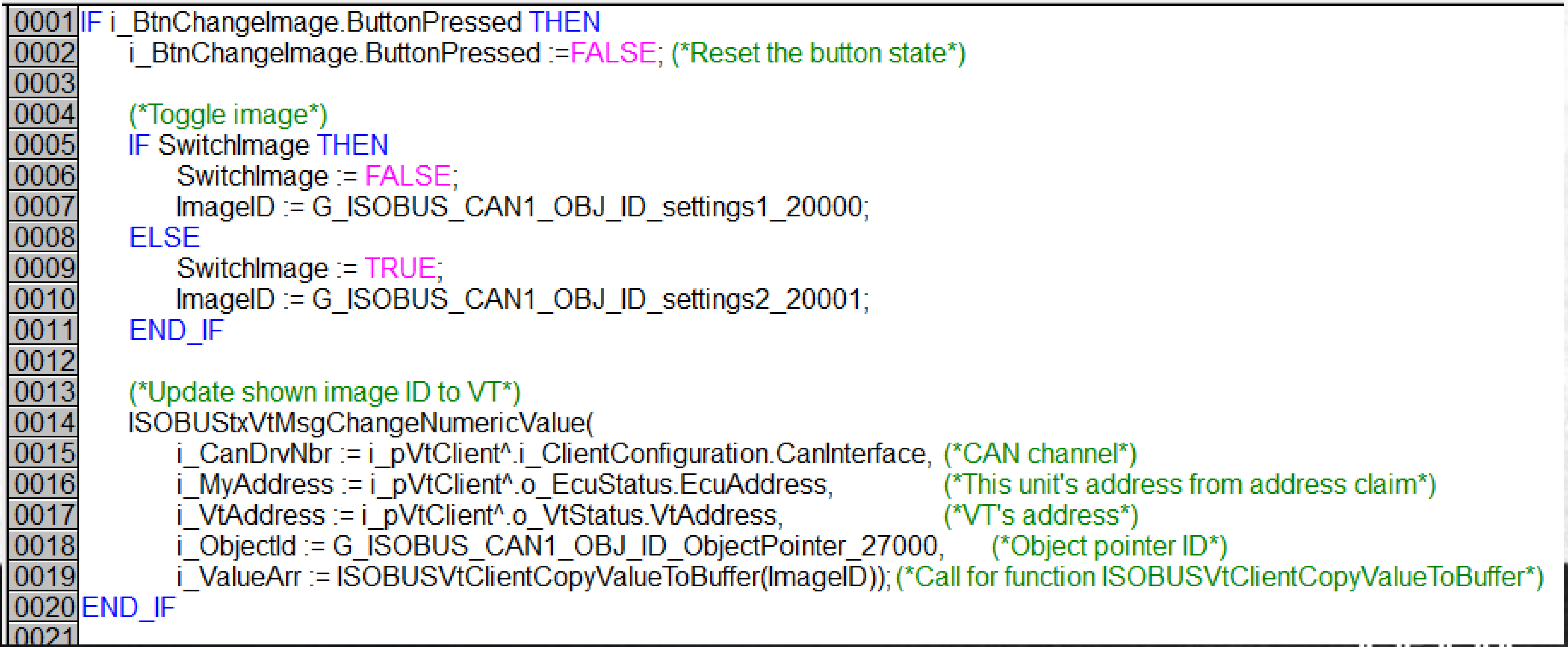
☑ Structured

Using Object Pointer and PictureGraphic

- *ISOBUStxVtMsgChangeNumericValue* input *i_ValueArr* requires the value as an array

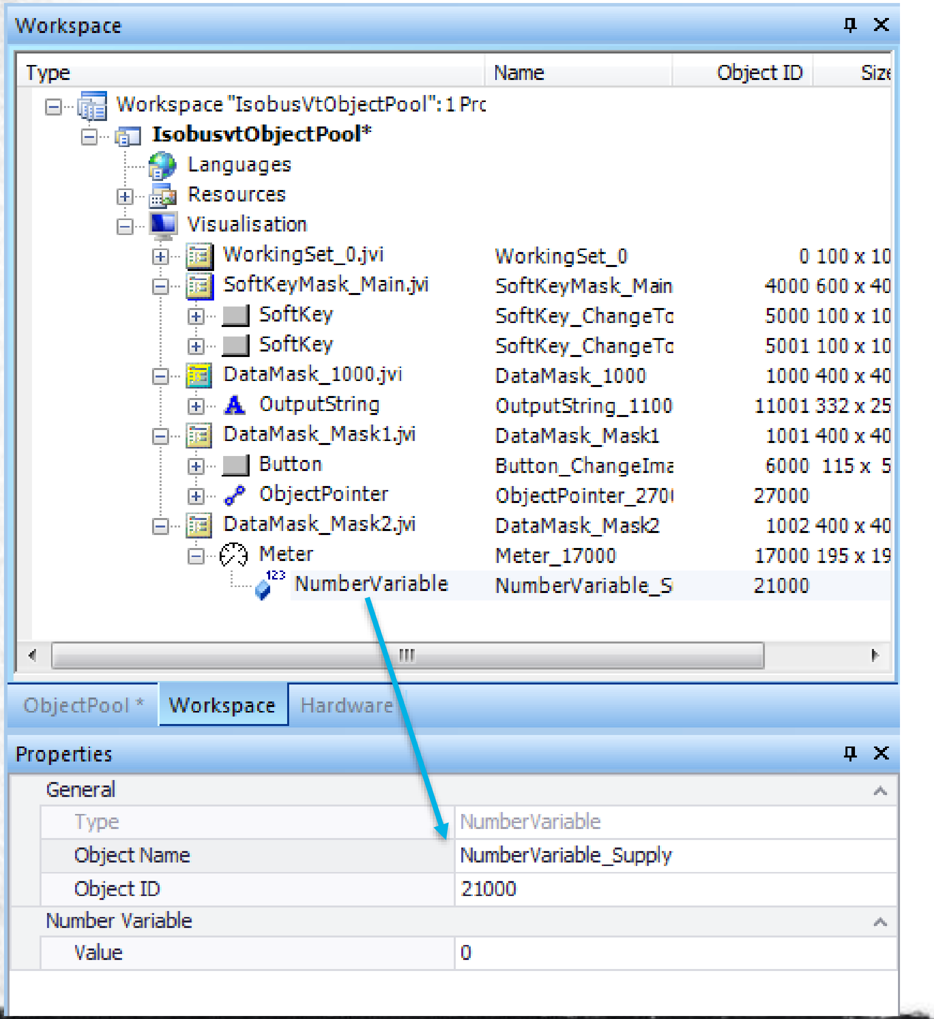
  → the value (ImageID) needs to be copied to a buffer

  → use function *ISOBUSVtClientCopyValueToBuffer*



```
IF i_BtnChangeImage.ButtonPressed THEN
    i_BtnChangeImage.ButtonPressed :=FALSE; (*Reset the button state*)

    (*Toggle image*)
    IF SwitchImage THEN
        SwitchImage := FALSE;
        ImageID := G_ISOBUS_CAN1_OBJ_ID_settings1_20000;
    ELSE
        SwitchImage := TRUE;
        ImageID := G_ISOBUS_CAN1_OBJ_ID_settings2_20001;
    END_IF

    (*Update shown image ID to VT*)
    ISOBUStxVtMsgChangeNumericValue(
        i_CanDrvNbr := i_pVtClient^.i_ClientConfiguration.CanInterface,  (*CAN channel*)
        i_MyAddress := i_pVtClient^.o_EcuStatus.EcuAddress,              (*This unit's address from address claim*)
        i_VtAddress := i_pVtClient^.o_VtStatus.VtAddress,                (*VT's address*)
        i_ObjectId := G_ISOBUS_CAN1_OBJ_ID_ObjectPointer_27000,              (*Object pointer ID*)
        i_ValueArr := ISOBUSVtClientCopyValueToBuffer(ImageID)); (*Call for function ISOBUSVtClientCopyValueToBuffer*)
END_IF
```

Using Meter



Workspace

Type	Name	Object ID	Size
Workspace "IsobusVtObjectPool": 1 Prc			
IsobusvtObjectPool*			
Languages			
Resources			
Visualisation			
WorkingSet_0.jvi	WorkingSet_0	0	100 x 10
SoftKeyMask_Main.jvi	SoftKeyMask_Main	4000	600 x 40
SoftKey	SoftKey_ChangeTc	5000	100 x 10
SoftKey	SoftKey_ChangeTc	5001	100 x 10
DataMask_1000.jvi	DataMask_1000	1000	400 x 40
OutputString	OutputString_1100	11001	332 x 25
DataMask_Mask1.jvi	DataMask_Mask1	1001	400 x 40
Button	Button_ChangeIma	6000	115 x 5
ObjectPointer	ObjectPointer_270	27000	
DataMask_Mask2.jvi	DataMask_Mask2	1002	400 x 40
Meter	Meter_17000	17000	195 x 19
NumberVariable	NumberVariable_S	21000	

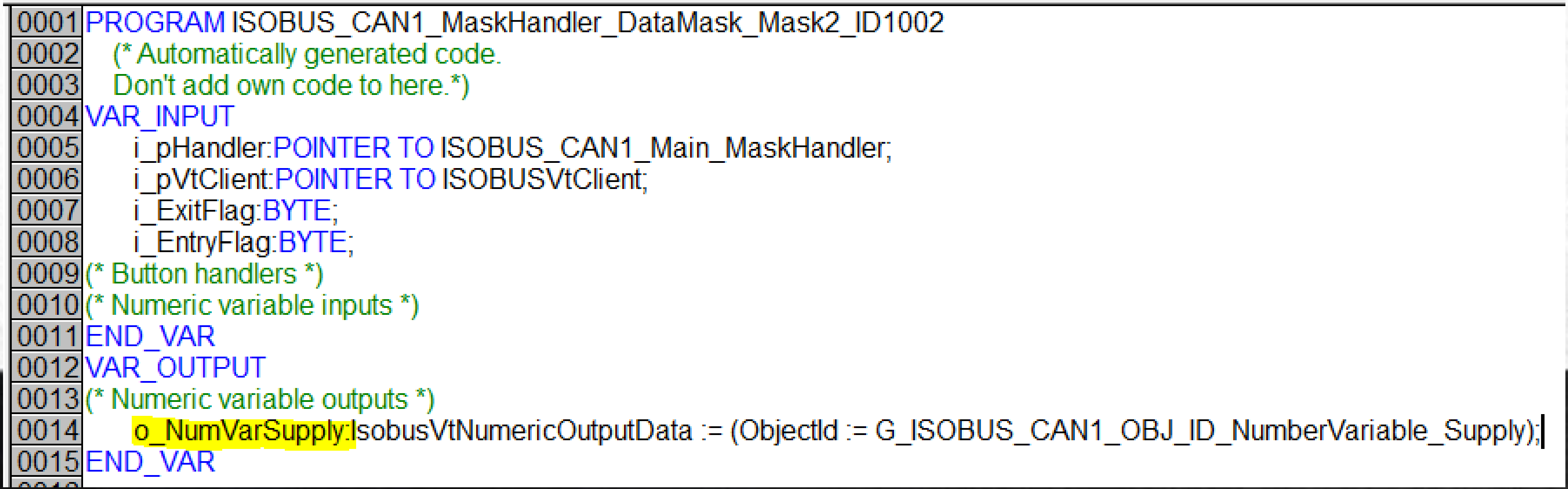
Properties

General	
Type	NumberVariable
Object Name	NumberVariable_Supply
Object ID	21000
Number Variable	
Value	0

- Add a meter object to *DataMask_Mask2*
- Define min/max value from 0 to 2400
- Add a new **Number Variable** from meter **Properties**
  - Go to the Workspace tab, select the variable and rename it as *NumberVariable_Supply*
- Build the object pool and import updates to the CODESYS project

Adding Handling for Meter

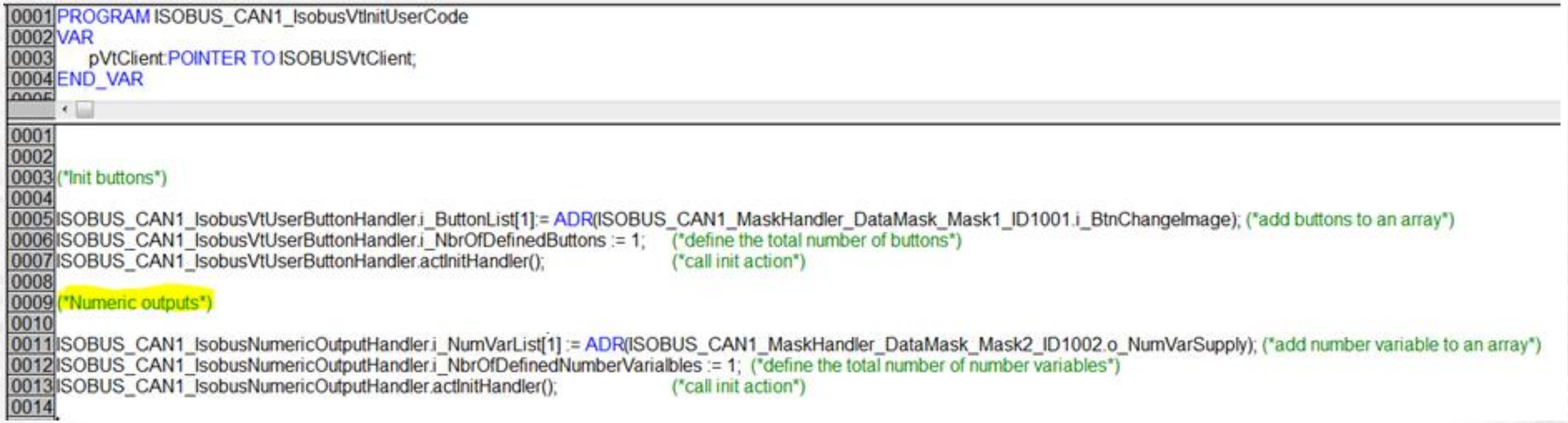
- Numeric outputs have their own structure *IsobusVtNumericOutputData* (CODESYS Data types tab)
- Add a numeric output of type *IsobusVtNumericOutputData* to the data mask handler *ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002*
    - *ObjectId* can be found from *IsobusExportVtInfo* global variables list



```
PROGRAM ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002
   (* Automatically generated code.
   Don't add own code to here.*)
VAR_INPUT
     i_pHandler:POINTER TO ISOBUS_CAN1_Main_MaskHandler;
     i_pVtClient:POINTER TO ISOBUSVtClient;
     i_ExitFlag:BYTE;
     i_EntryFlag:BYTE;
(* Button handlers *)
(* Numeric variable inputs *)
END_VAR
VAR_OUTPUT
(* Numeric variable outputs *)
     o_NumVarSupply:IsobusVtNumericOutputData := (ObjectId := G_ISOBUS_CAN1_OBJ_ID_NumberVariable_Supply);
END_VAR
```

Adding Handling for Meter

- Initialize the numeric variable in *ISOBUS_CAN1_IsobusVtInitUserCode*
  - Use *ISOBUS_CAN1_IsobusNumericOutputHandler* program

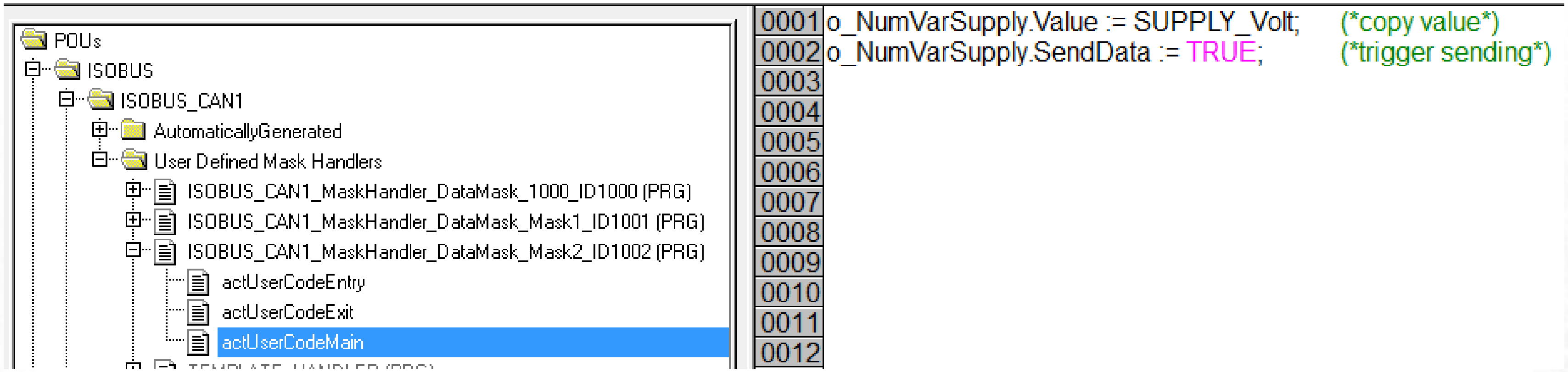


```
PROGRAM ISOBUS_CAN1_IsobusVtInitUserCode
VAR
    pVtClient:POINTER TO ISOBUSVtClient;
END_VAR
```

```


(*Init buttons*)

ISOBUS_CAN1_IsobusVtUserButtonHandler.i_ButtonList[1]:= ADR(ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001.i_BtnChangeImage); (*add buttons to an array*)
ISOBUS_CAN1_IsobusVtUserButtonHandler.i_NbrOfDefinedButtons := 1;     (*define the total number of buttons*)
ISOBUS_CAN1_IsobusVtUserButtonHandler.actInitHandler();               (*call init action*)

(*Numeric outputs*)

ISOBUS_CAN1_IsobusNumericOutputHandler.i_NumVarList[1] := ADR(ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002.o_NumVarSupply); (*add number variable to an array*)
ISOBUS_CAN1_IsobusNumericOutputHandler.i_NbrOfDefinedNumberVarialbles := 1;  (*define the total number of number variables*)
ISOBUS_CAN1_IsobusNumericOutputHandler.actInitHandler();              (*call init action*)

```

Adding Handling for Meter

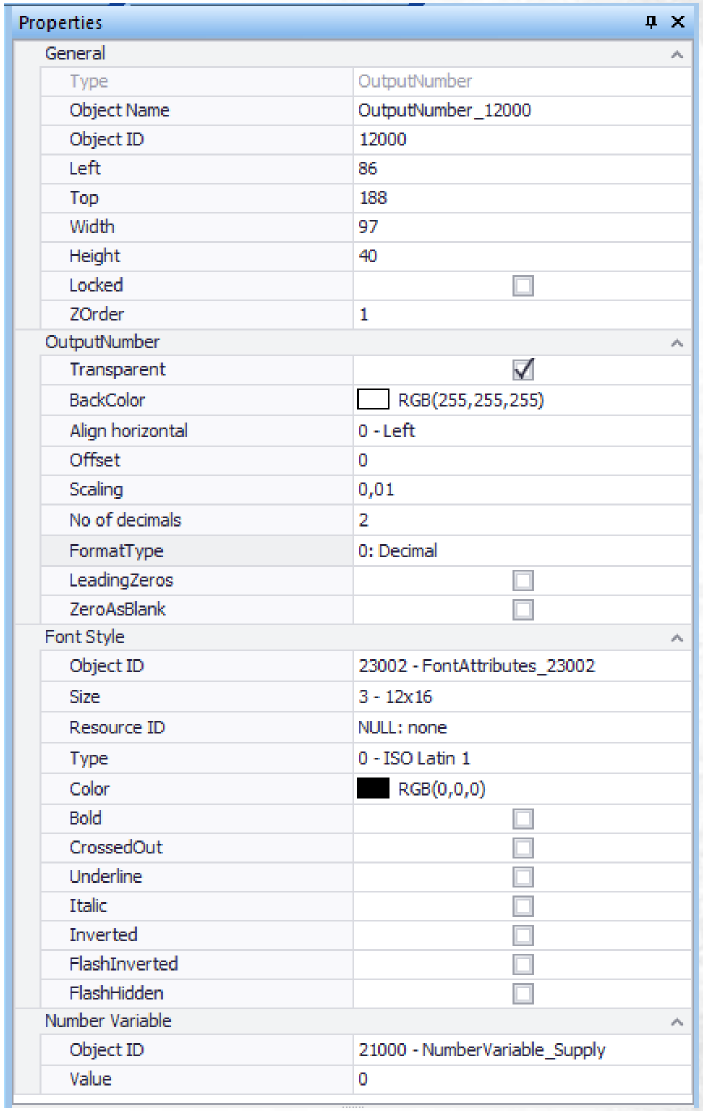
- Open *ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002*
- Copy the *SUPPLY_Volt (IO_INTERNAL)* value to the output variable *o_NumVarSupply*
- Download the CODESYS application, update the object pool binary to the unit with CANmoon and reboot the unit


POUs
ISOBUS
ISOBUS_CAN1
AutomaticallyGenerated
User Defined Mask Handlers
ISOBUS_CAN1_MaskHandler_DataMask_1000_ID1000 (PRG)
ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001 (PRG)
ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002 (PRG)
actUserCodeEntry
actUserCodeExit
actUserCodeMain
TEMPLATE_HANDLER (PRG)

```
o_NumVarSupply.Value := SUPPLY_Volt;         (*copy value*)
o_NumVarSupply.SendData := TRUE;             (*trigger sending*)
```

Adding Output Number

- Add an **OutputNumber** to *DataMask_Mask2*
- Show the supply voltage value sent from Epec unit, the value 2400 = 24V. Set
  - **No of decimals → 2**
  - **Scaling** → 0,01
- Use *NumberVariable_Supply* as **Object ID**
- Run **Import ISOBUS** macro in CODESYS, update the object pool binary to the unit with CANmoon and reboot the unit

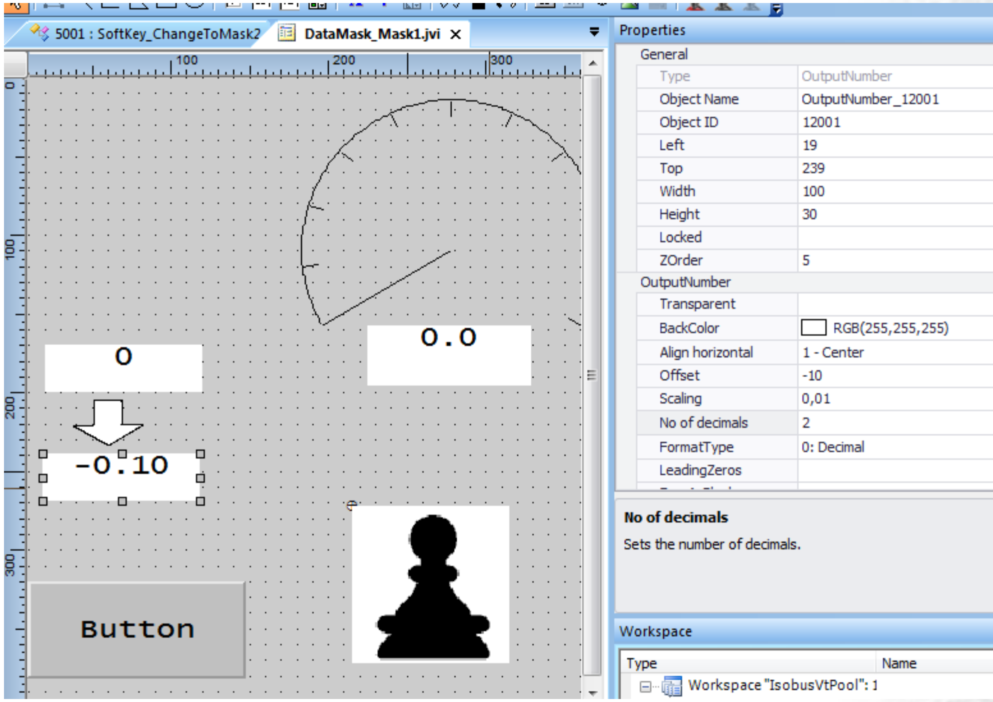


Properties

General	
Type	OutputNumber
Object Name	OutputNumber_12000
Object ID	12000
Left	86
Top	188
Width	97
Height	40
Locked	□
ZOrder	1
OutputNumber	
Transparent	☑
BackColor	□ RGB(255,255,255)
Align horizontal	0 - Left
Offset	0
Scaling	0,01
No of decimals	2
FormatType	0: Decimal
LeadingZeros	□
ZeroAsBlank	□
Font Style	
Object ID	23002 - FontAttributes_23002
Size	3 - 12x16
Resource ID	NULL: none
Type	0 - ISO Latin 1
Color	■ RGB(0,0,0)
Bold	□
CrossedOut	□
Underline	□
Italic	□
Inverted	□
FlashInverted	□
FlashHidden	□
Number Variable	
Object ID	21000 - NumberVariable_Supply
Value	0

Using Input and Output Numbers

- Add **InputNumber** and add a **New Number Variable** *NumberVariable_Input* to it.
- Add an **OutputNumber** and link the numeric variable *NumberVariable_Output* to it. In **Properties**, set
  - **Scaling** 0,01
  - **Offset** -1
  - **No of decimals** 2
- Draw an arrow from input to output with polygon



General	
Type	OutputNumber
Object Name	OutputNumber_12001
Object ID	12001
Left	19
Top	239
Width	100
Height	30
Locked	
ZOrder	5
OutputNumber	
Transparent	
BackColor	□ RGB(255,255,255)
Align horizontal	1 - Center
Offset	-10
Scaling	0,01
No of decimals	2
FormatType	0: Decimal
LeadingZeros	

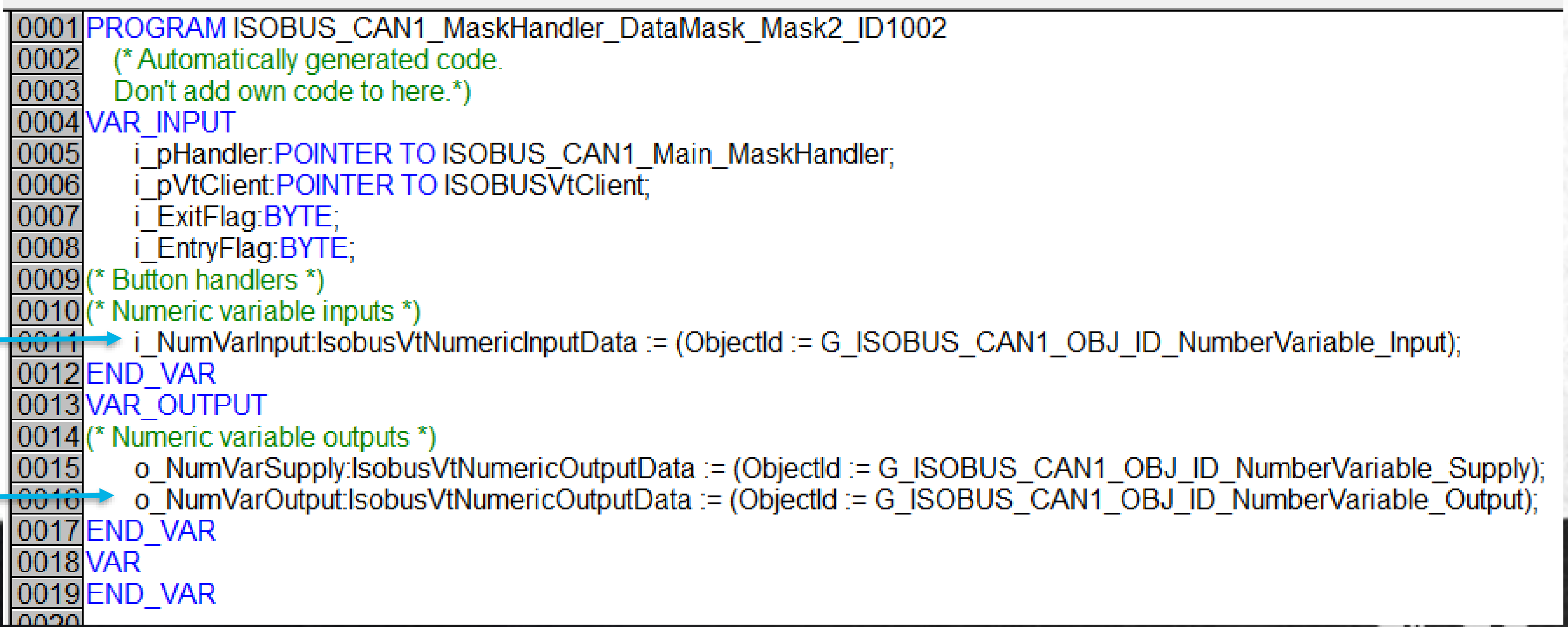
No of decimals

Sets the number of decimals.

Type	Name
Workspace "IsobusVtPool": 1	

Using Input and Output Numbers

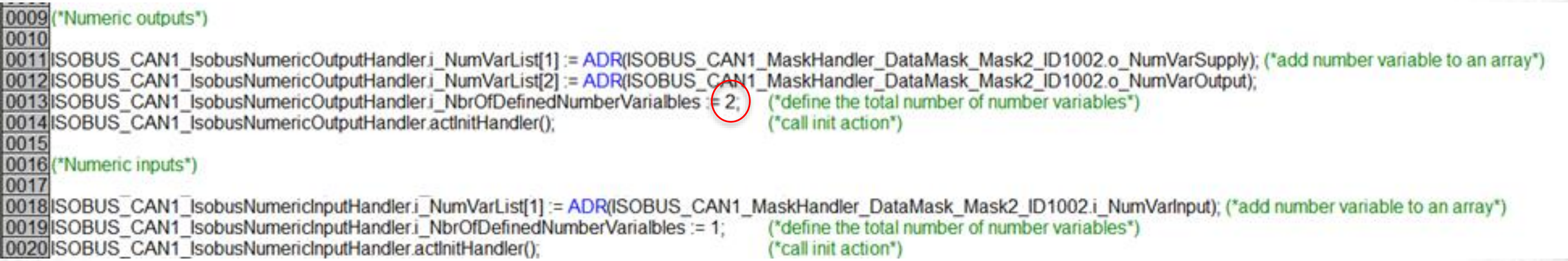
- Build the ISO-Designer project and import changes to the CODESYS project (**Edit > Macros > Import ISOBUS**)
- Open *ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002*
- Add declarations for input *i_NumvarInput* and output *o_NumvarOutput*



```
PROGRAM ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002
   (* Automatically generated code.
   Don't add own code to here.*)
VAR_INPUT
      i_pHandler:POINTER TO ISOBUS_CAN1_Main_MaskHandler;
      i_pVtClient:POINTER TO ISOBUSVtClient;
      i_ExitFlag:BYTE;
      i_EntryFlag:BYTE;
(* Button handlers *)
(* Numeric variable inputs *)
      i_NumVarInput:IsobusVtNumericInputData := (ObjectId := G_ISOBUS_CAN1_OBJ_ID_NumberVariable_Input);
END_VAR
VAR_OUTPUT
(* Numeric variable outputs *)
      o_NumVarSupply:IsobusVtNumericOutputData := (ObjectId := G_ISOBUS_CAN1_OBJ_ID_NumberVariable_Supply);
      o_NumVarOutput:IsobusVtNumericOutputData := (ObjectId := G_ISOBUS_CAN1_OBJ_ID_NumberVariable_Output);
END_VAR
VAR
END_VAR
```

Using Input and Output Numbers

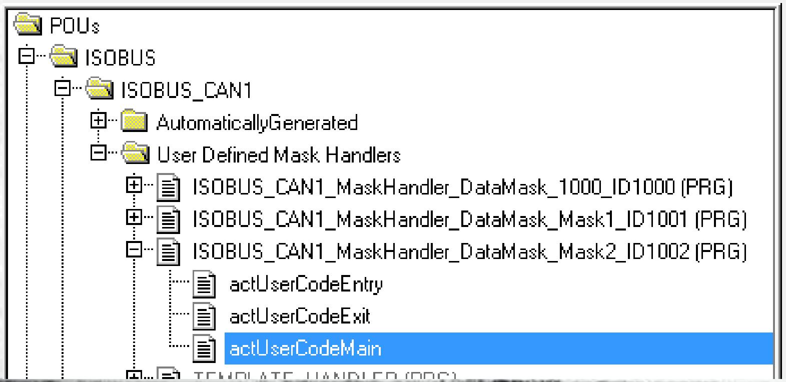
- Open *ISOBUS_CAN1_IsobusVtInitUserCode* program and initialize the added input and output to handler programs



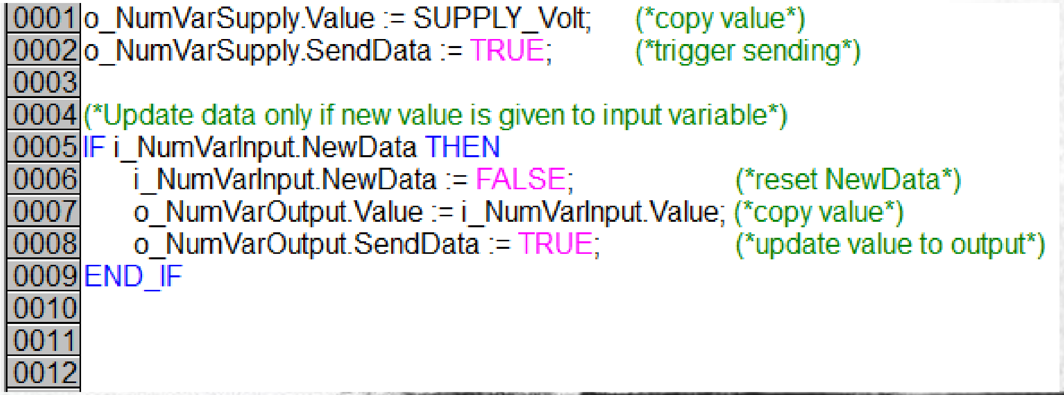
```
(*Numeric outputs*)

ISOBUS_CAN1_IsobusNumericOutputHandler.i_NumVarList[1] := ADR(ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002.o_NumVarSupply); (*add number variable to an array*)
ISOBUS_CAN1_IsobusNumericOutputHandler.i_NumVarList[2] := ADR(ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002.o_NumVarOutput);
ISOBUS_CAN1_IsobusNumericOutputHandler.i_NbrOfDefinedNumberVarialbles := 2;    (*define the total number of number variables*)
ISOBUS_CAN1_IsobusNumericOutputHandler.actInitHandler();                        (*call init action*)

(*Numeric inputs*)

ISOBUS_CAN1_IsobusNumericInputHandler.i_NumVarList[1] := ADR(ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002.i_NumVarInput); (*add number variable to an array*)
ISOBUS_CAN1_IsobusNumericInputHandler.i_NbrOfDefinedNumberVarialbles := 1;      (*define the total number of number variables*)
ISOBUS_CAN1_IsobusNumericInputHandler.actInitHandler();                         (*call init action*)
```

Using Input and Output Numbers

- Open *ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002*
- Add code that
  - checks if a new input value is received from the VT
  - copies the new input value to the output value
  - updates the output value to the VT
- Download the CODESYS application, update the object pool binary to the unit with CANmoon and reboot the unit



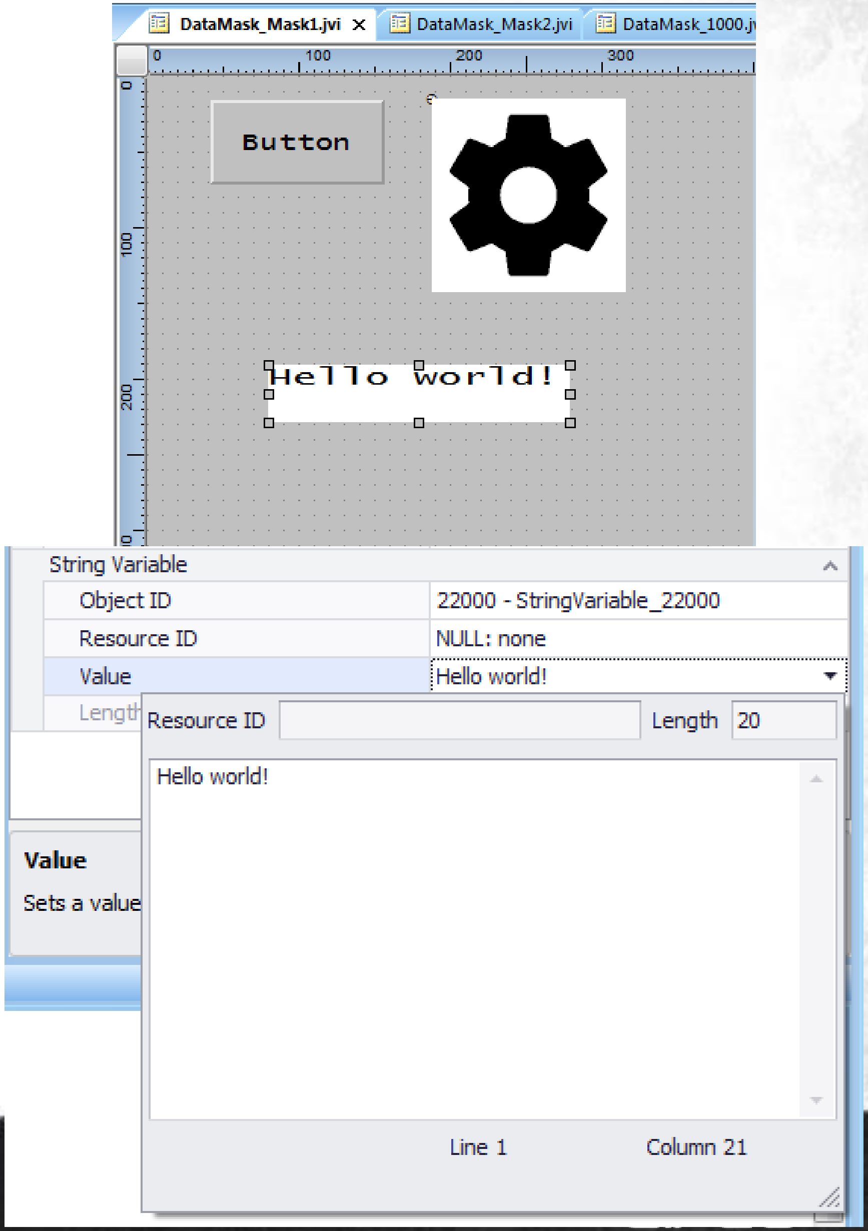
```
POUs
⊟ ISOBUS
  ⊟ ISOBUS_CAN1
    ⊞ AutomaticallyGenerated
    ⊟ User Defined Mask Handlers
      ⊞ ISOBUS_CAN1_MaskHandler_DataMask_1000_ID1000 (PRG)
      ⊞ ISOBUS_CAN1_MaskHandler_DataMask_Mask1_ID1001 (PRG)
      ⊟ ISOBUS_CAN1_MaskHandler_DataMask_Mask2_ID1002 (PRG)
          actUserCodeEntry
          actUserCodeExit
          actUserCodeMain
      ⊞ TEMPLATE_HANDLER (PRG)
```



```
o_NumVarSupply.Value := SUPPLY_Volt;       (*copy value*)
o_NumVarSupply.SendData := TRUE;           (*trigger sending*)

(*Update data only if new value is given to input variable*)
IF i_NumVarInput.NewData THEN
    i_NumVarInput.NewData := FALSE;                (*reset NewData*)
    o_NumVarOutput.Value := i_NumVarInput.Value;   (*copy value*)
    o_NumVarOutput.SendData := TRUE;               (*update value to output*)
END_IF
```

Using OutputString

- Add an **OutputString** object to *DataMask_Mask1*
- **Add New** string variable in OutputString Properties and name it as *StringVariable_HelloWord*
- Set **String Variable > Value** to ”Hello world” and add spaces until the length is 20 characters



String Variable

Object ID	22000 - StringVariable_22000
Resource ID	NULL: none
Value	Hello world!
Length	

Value

Sets a value

Resource ID
Length 20

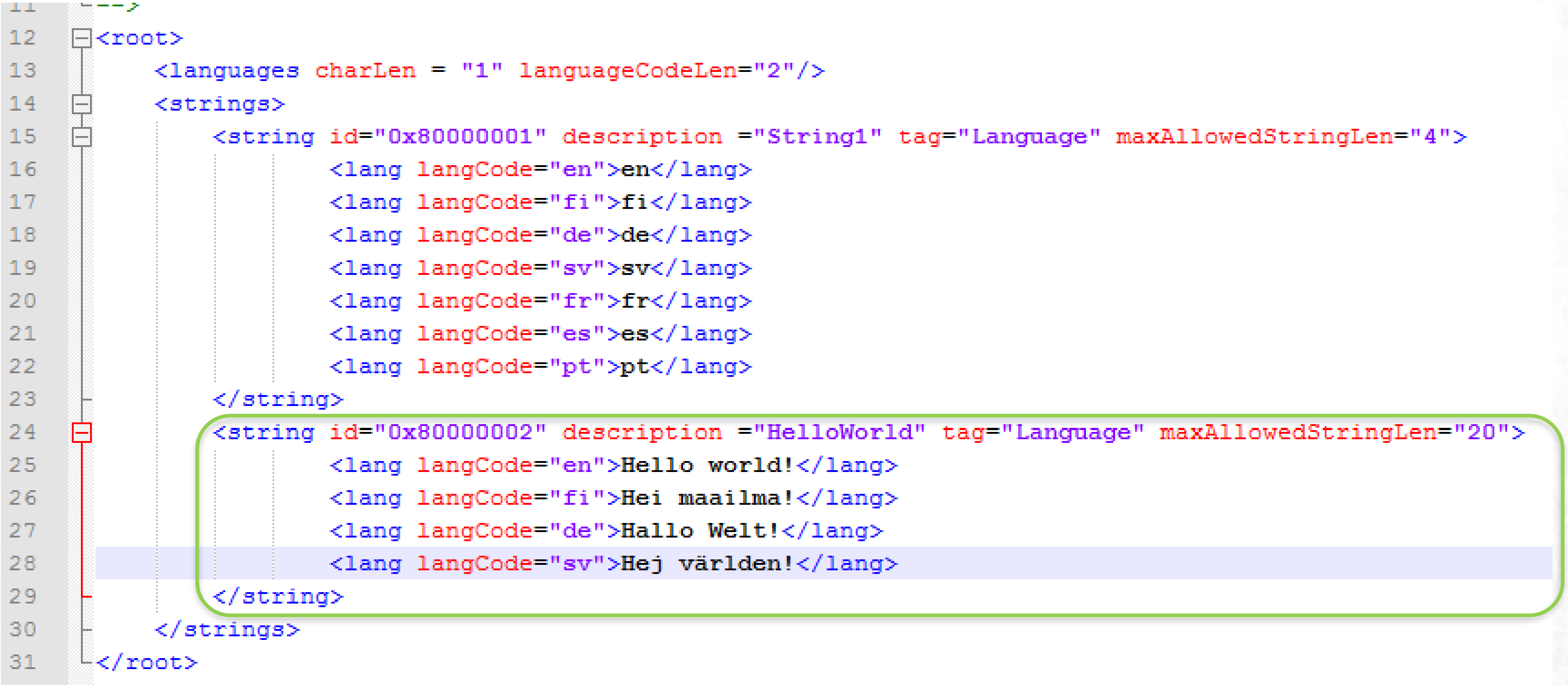
Hello world!

Line 1
Column 21

Editing the Language File

- Open file *{Device}\ISOBUS\Python\Languages\languages.xml*
- Adding new text with translations is done by adding
  - a new string ID
  - element lang tags and language texts
- The language is referenced in application by a character code ('en','de',...)
- This example uses ISO639-2 language codes
- Build the ISO-Designer project and import changes to the CODESYS project

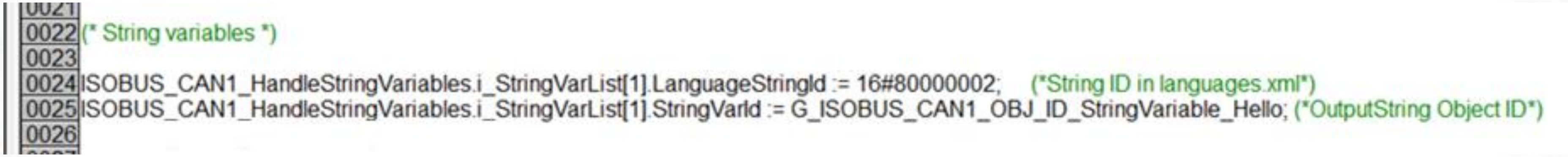
Editing Language File



```
-->
<root>
    <languages charLen = "1" languageCodeLen="2"/>
    <strings>
        <string id="0x80000001" description ="String1" tag="Language" maxAllowedStringLen="4">
                <lang langCode="en">en</lang>
                <lang langCode="fi">fi</lang>
                <lang langCode="de">de</lang>
                <lang langCode="sv">sv</lang>
                <lang langCode="fr">fr</lang>
                <lang langCode="es">es</lang>
                <lang langCode="pt">pt</lang>
        </string>
        <string id="0x80000002" description ="HelloWorld" tag="Language" maxAllowedStringLen="20">
                <lang langCode="en">Hello world!</lang>
                <lang langCode="fi">Hei maailma!</lang>
                <lang langCode="de">Hallo Welt!</lang>
                <lang langCode="sv">Hej världen!</lang>
        </string>
    </strings>
</root>
```

Init Language Handler

- Add init code to *ISOBUS_CAN1_IsobusVtInitUserCode*
- *HandleStringVariables* does not need the amount of string variables as an input



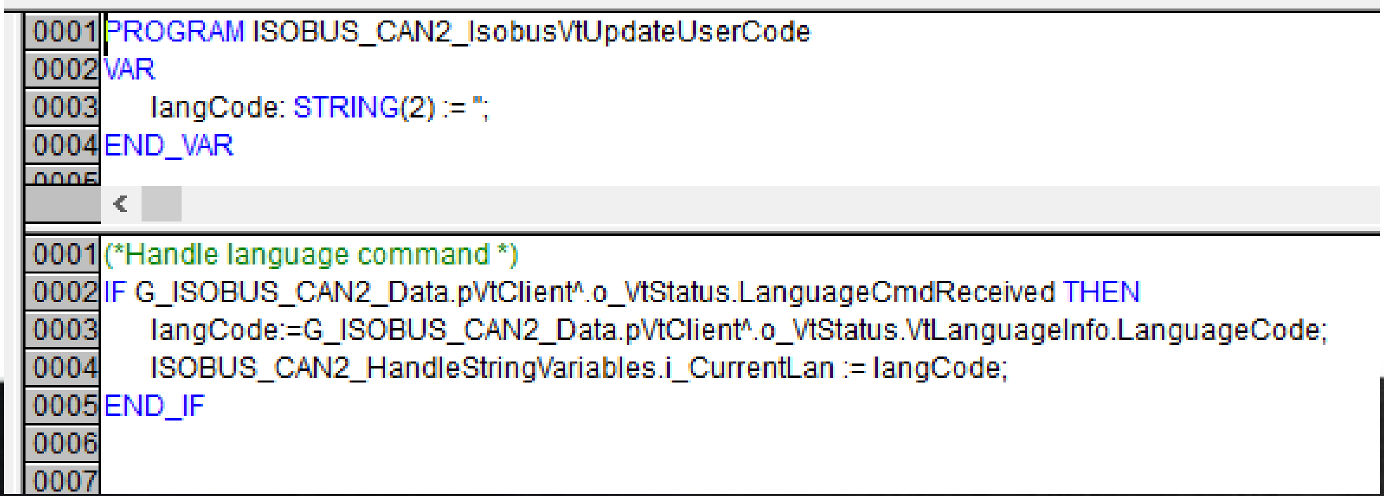
```
(* String variables *)

ISOBUS_CAN1_HandleStringVariables.i_StringVarList[1].LanguageStringId := 16#80000002;    (*String ID in languages.xml*)
ISOBUS_CAN1_HandleStringVariables.i_StringVarList[1].StringVarId := G_ISOBUS_CAN1_OBJ_ID_StringVariable_Hello; (*OutputString Object ID*)

```

Language Commad

- Language command is sent in system initialization and on change
- After the system has completed its power-on and address claims, the VT (virtual terminal) sends a language command message which includes information about selected language, formats and measurement units

Changing Language

- Open *ISOBUS_CAN1_IsobusVtUpdateUserCode*
- To handle language command message
  - add a variable *langCode*
  - get VT client’s language code
  - assign it to string handler’s input *i_CurrentLan*
- Download the CODESYS application, update the object pool binary to the unit with CANmoon and reboot the unit



```
PROGRAM ISOBUS_CAN2_IsobusVtUpdateUserCode
VAR
    langCode: STRING(2) := '';
END_VAR
```

```
(*Handle language command *)
IF G_ISOBUS_CAN2_Data.pVtClient^.o_VtStatus.LanguageCmdReceived THEN
    langCode:=G_ISOBUS_CAN2_Data.pVtClient^.o_VtStatus.VtLanguageInfo.LanguageCode;
    ISOBUS_CAN2_HandleStringVariables.i_CurrentLan := langCode;
END_IF
```

Images via Object Pointer in Softkeys

- When an object pointer is used with soft key, ISO-Designer only gives information about *one referenced object* in object pointer properties (in this case SoftKey_START picture graphic)
- This one object will be correctly scaled, but for the other used objects the following definition needs to be added to ISOBUS main program > action actInitVt so that it is scaled correctly:

```
FOR i:= 1 TO vtReadBinaryData.o_NbrOfObjects DO
    IF vtClient.i_ClientConfiguration.ObjectPool.pObjectPoolList^[i].ObjectID =
    G_TrainingExampleIsobus_VT_OBJ_ID_SoftKey_STOP_20006 THEN
       vtClient.i_ClientConfiguration.ObjectPool.pObjectPoolList^[i].TopLevelObjectType :=
       ISOBUS_VT_POOL_OBJ_TYPE_SOFTKEY_MASK;
      END_IF
END_FOR
```

Giving a Default Value for Numeric Inputs

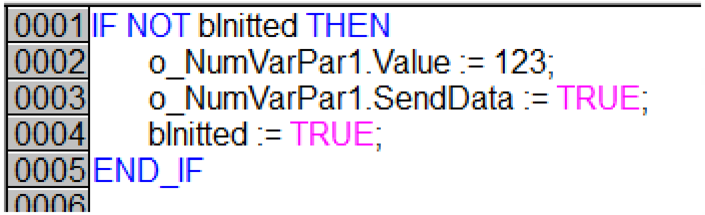
1. Define numeric input variable normally for the data mask handler

```
i_NumVarPar1:IsobusVTNumericInputData        :=
(ObjectId:=G_TrainingExampleIsobus_VT_OBJ_ID_NumberVariable_Par1);
```

2. Define a corresponding output variable too (this gives the initial value for the VT)

```
o_NumVarPar1:IsobusVtNumericOutputData :=
(ObjectId:=G_TrainingExampleIsobus_VT_OBJ_ID_NumberVariable_Par1);
```

3. Add a new action (for example, actSetDefaults) for the data mask handler
4. Add initialization to the new action



```
IF NOT bInitted THEN
    o_NumVarPar1.Value := 123;
    o_NumVarPar1.SendData := TRUE;
    bInitted := TRUE;
END_IF
```

5. Call actSetDefaults in ISOBUS_CAN2_IsobusVtUpdateUserCode

MADE IN FINLAND
Extreme conditions? Bring it on!

Thank you!

Customer Support

techsupport@epec.fi