# Global B - How to Add a Loss of Com DTC

This document describes how to add a new Loss of Com DTC to an AUTOSAR-compliant Global B VIP software project (i.e. MY21, MY22, MY23 Global B, and beyond). This document assumes you are starting with a baseline project with the latest ARXML database already integrated. It also assumes that you have some basic knowledge of how DTCs work in an AUTOSAR software project.

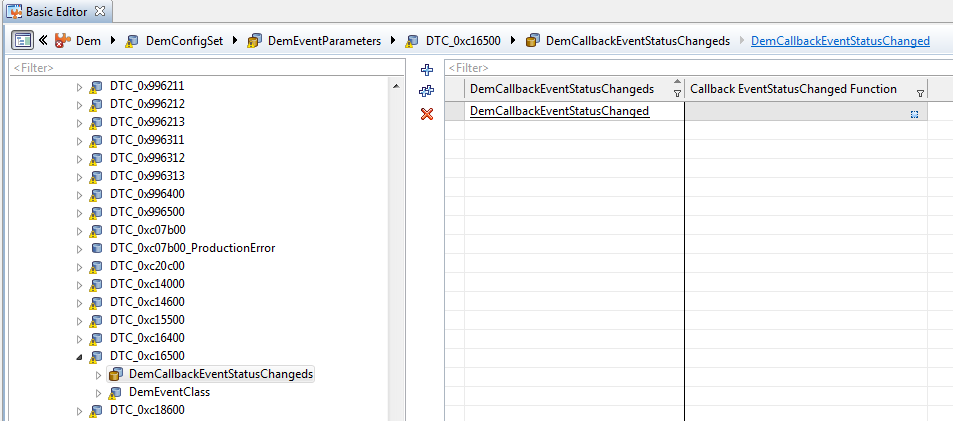
## Summary

Before delving into the details of the implementation, it can be helpful to get a high-level picture of all of the changes you will be making, since multiple components are involved.

1. DEM. Responsible for managing “diagnostic events” (essentially DTCs). Here you will configure the diagnostic event for handling the DTC, including the Status Changed callback, Enable Conditions, debouncing parameters, etc.
2. SUM\_SSM. Responsible for monitoring the status one or more signals, and associating those signals with a particular “node” (e.g. an ECU). Here you will configure the signals to be monitored and associate those signals with a monitored ECU.
3. Ea. Responsible for interacting with the EEPROM. Here you would defined

## Changes in DaVinci Configurator and Developer

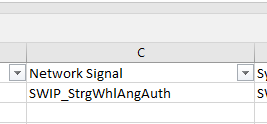
### Configurator: Configuring the DEM

1. Open your project in DaVinci Configurator.
2. Synchronize and save the project, if necessary.
3. Open the Basic Editor.
4. Expand the following node:   
     
   Dem/DemConfigSet/DemEventParameters
5. Scroll down the list of DTCs and look for the one you are trying to add. It should already be in the list. Expand the node.
6. Click on the “DemCallbackEventStatusChangeds” node. Click on the ‘plus’ button to add a callback. The default name is fine.  
     
   
7. Synchronize and save the project again.
8. Click on the “DemEventClass” node. Set:  
     
   EnableCondition Group Ref = /ActiveEcuC/Dem/DemGeneral/DTC\_LogisticTransportAndLossofCom\_Conditions  
     
   Event FailureCycle Ref = /ActiveEcuC/Dem/DemGeneral/IgnitionCycle  
     
   Event Priority = 1
9. Expand the Dem/DemConfigSet/DemEventParameters/DTC\_0xcXXXXX/DemEventClass/DemDebounceAlgorithmClass/DemDebounceCounterBased node. Click on the node.
10. Set Debounce Behavior = DEM\_DEBOUNCE\_FREEZE.
11. Close the Dem node.

### Configurator: Configuring the SUM\_SSM

During this stage, you will be mapping the supervised signals to a Monitored ECU in SUM\_SSM. Before you can proceed, you will need to know which signal(s) should be supervised for a particular ECU and DTC. This information can be found in the Signal Status DTC mapping table released by the Systems team. This table is an Excel document and can be found on GDM by searching for “Signal\_Status\_DTC map\_CSM”. Make sure you download the correct table for your program and release. For example, the MY22 table for release 149 is named “22.22.149SSM Signal\_Status\_DTC map\_CSM.xlsx”.

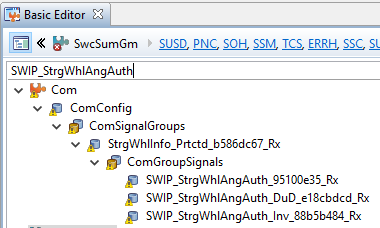
Open the Signal Status DTC mapping table and filter the “DTC Number” column by the DTC number you are working on. The “Network Signal” column should contain the list of supervised signals for the ECU and DTC. In the example below, it is “SWIP\_StrgWhlAngAuth”.



To proceed with the procedure in this section, you will first need to gather the following information about each supervised signal:

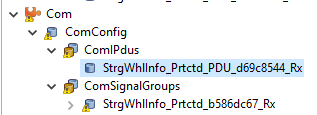
1. The signal group the signal belongs to, if any.
2. The PDU the signal belongs to.
3. The list of PNs mapped to the signal.

This information can be gathered within DaVinci Configurator. Search for your supervised signal in the Filter box in the Basic Editor.



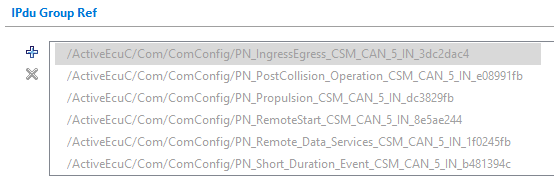
In this example, the “SWIP\_StrgWhlAngAuth” signal belongs to the “StrgWhlInfo\_Prtctd” signal group. This is item #1 above. Make a note of this.

Next search for the signal group in the filter box. You should be able to find the PDU name under the Com/ComConfig/ComIPdus node.



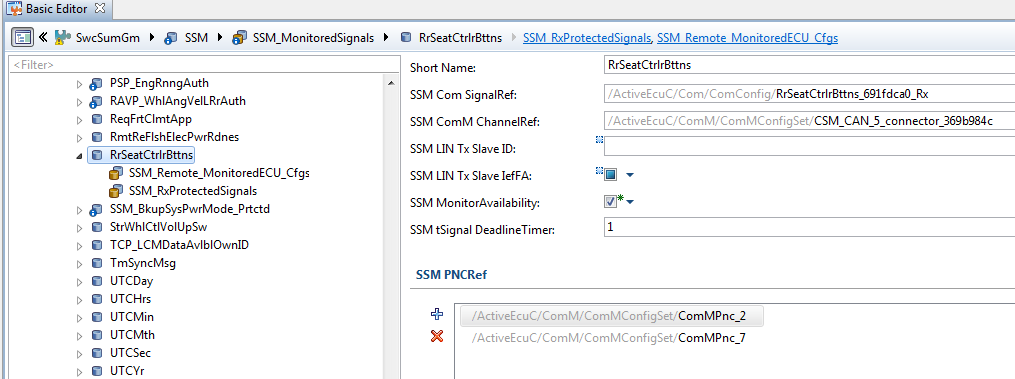
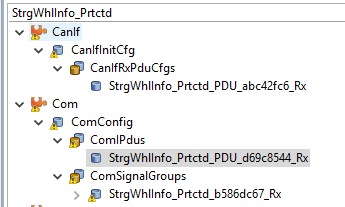
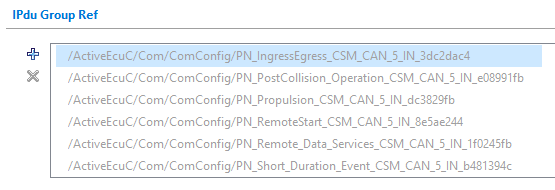
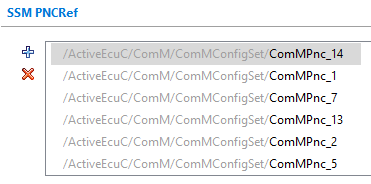
In this example, the signal group belongs to the “StrgWhlInfo\_Prtctd” PDU. This is item #2 above. Make a note of this.

Click on the PDU node. The list of mapped PNs should be shown in the “IPdu Group Ref” box. This is item #3 above.

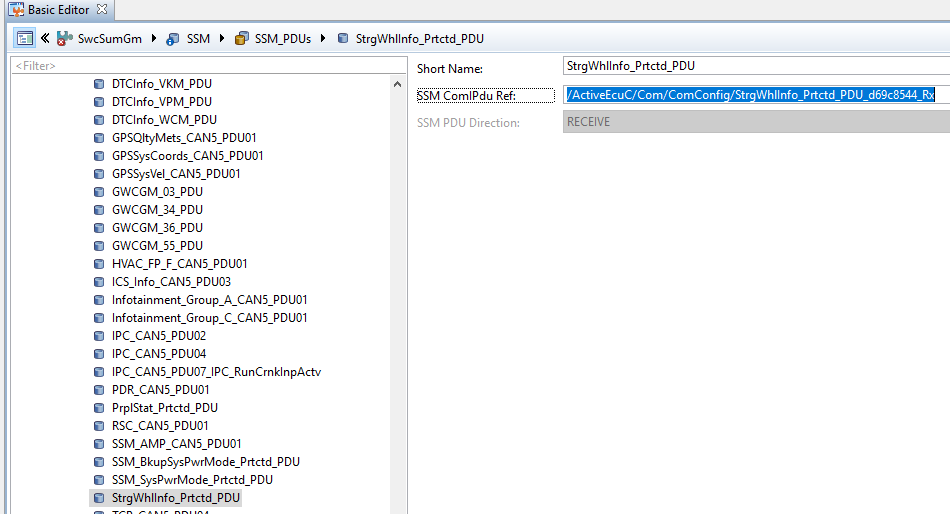
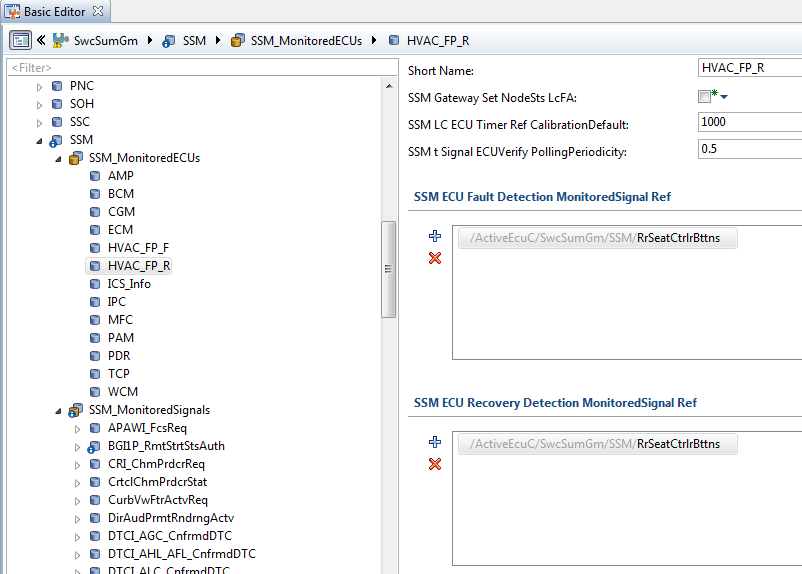


The PNs here are listed by name. These will need to be mapped to their corresponding PN numbers. There is a table at the end of this document which maps PN names to numbers.

You now have all of the required information to proceed with the below procedure to configure SUM\_SSM.

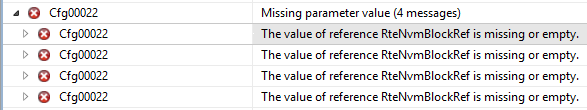
1. Expand the SwcSumGm/SSM node and click on the SSM\_MonitoredSignals node. Check to make sure that all of the monitored signals for the DTC you are implementing exist under this node. If they don’t, add them here. The parameters should be similar to the screenshot below:  
     
   
2. Make sure that the SUM PNCRef list matches up with the list of PNs for that signal as defined in Com. If you aren’t sure, search for that signal in the filter box. You should find that signal’s corresponding PDU in the Com/ComConfig/ComIPdus node.  
     
   
3. The Com PDU’s list of PNs is listed under the IPdu Group Ref box:  
     
   
4. In the SwcSumGm/SSM/SSM\_MonitoredSignals/<Signal> node, make sure that you map the same PNs as are listed in the Com configuration (information item #3 you noted above).  
     
   

Note that here the PNs may be referenced by number, whereas in Com they are listed by name. See the table at the bottom of this document for a mapping between PN names and PN numbers.

1. At this point, you may see the error “There must be 1 and only 1 Rx 'SSM\_PDU' for which the referenced 'ComIPdu' contains the monitored signal.” To fix this, open the SwcSumGm/SSM/SSM\_PDUs tab.
2. Add the corresponding PDU to the list (information item #2 you noted above).
3. Name the PDU appropriately and map it to the correct PDU object in Com.  
     
   
4. Click on the SSM\_MonitoredECUs node. Here you can find the list of ECUs which are monitored by the SSM. If your particular ECU isn’t there, click the ‘plus’ button to add it. Add the monitored signals to the SSM ECU Fault Detection MonitoredSignal Ref and SSM ECU Recovery Detection MonitoredSignal Ref sections.  
     
   

### Configurator: Configuring Ea (EEPROM Abstraction)

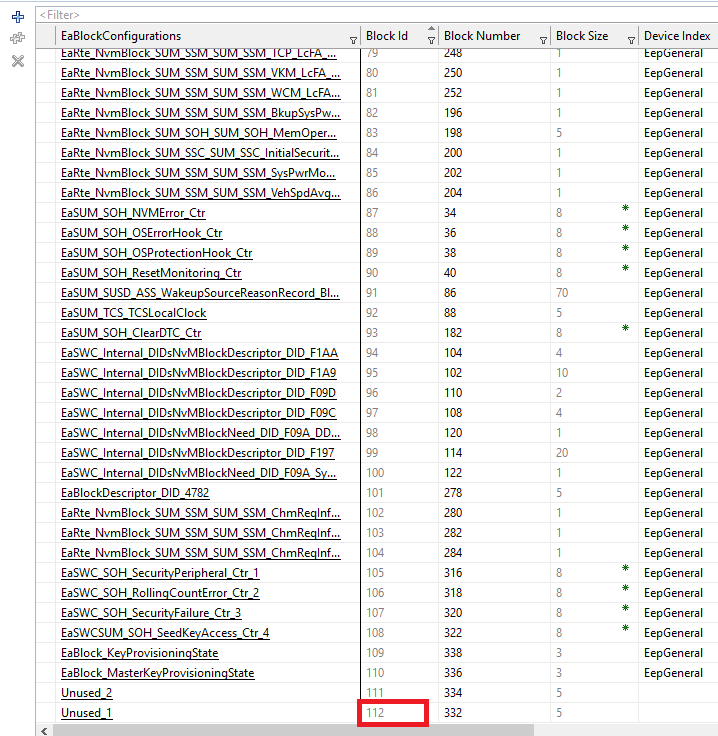
Depending on the signals you are using, it may be necessary to create FcFA, FoFA, FpFA, FsFA, and LcFA Ea blocks for each signal. You will have to do this if you see the following errors after completing the SUM\_SSM configuration above:



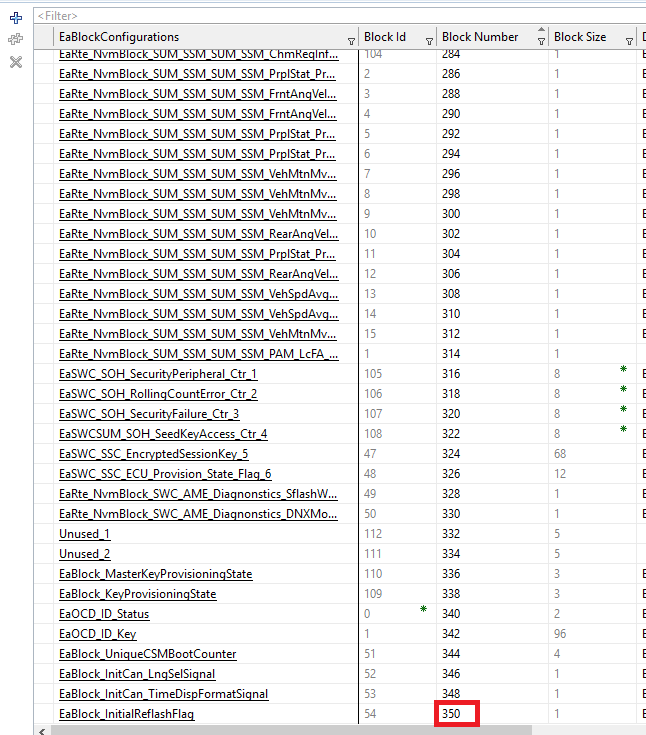
If you do not encounter these errors you can skip this section.

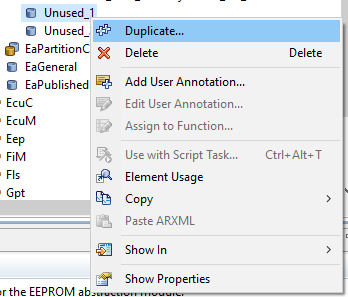
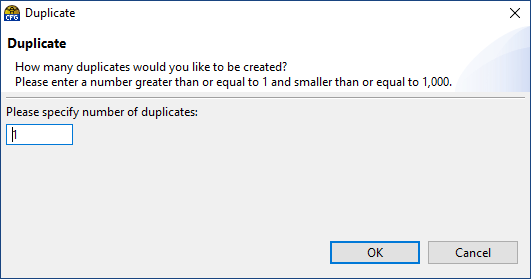
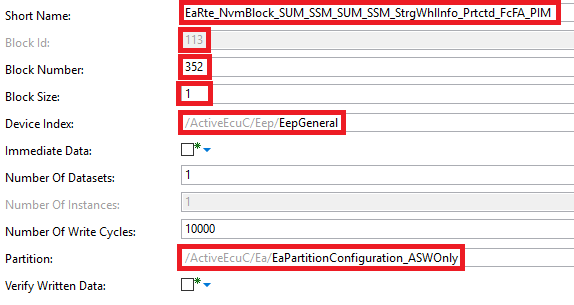
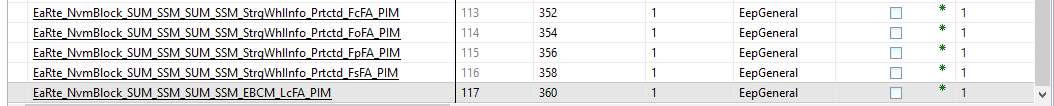
**IMPORTANT**: Before you begin, you need to determine the last used Block ID and Block Number for the existing Ea blocks. You need to make sure you use the next numbers in the sequence, otherwise you will inadvertently reorder the existing EEPROM blocks.

Click on the Ea/EaBlockConfigurations node. First sort the list by Block ID and note the latest number. In this case it is 112. You should use 113 as the next Block ID.



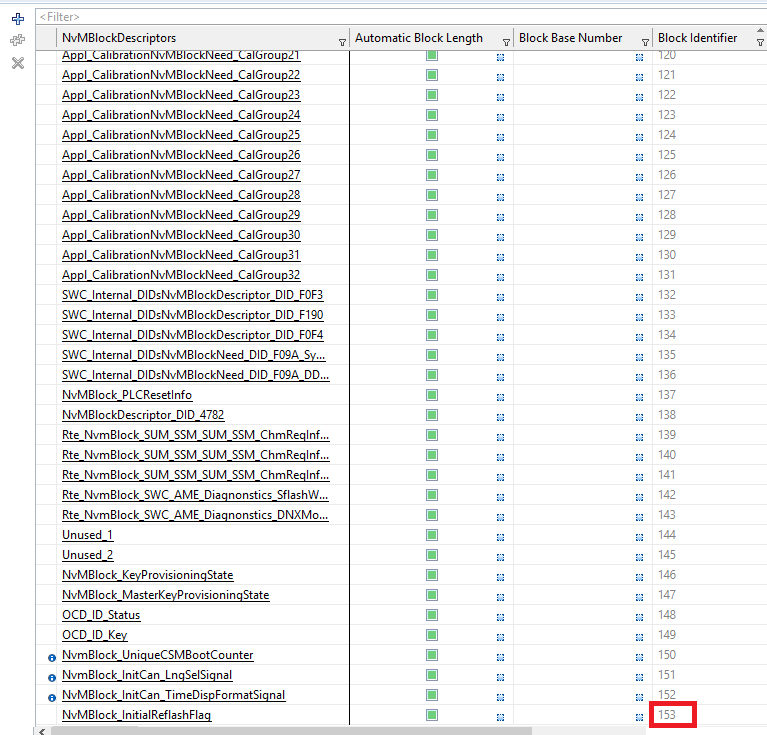
Do the same for the Block Number. In this example, the latest Block Number is 350. Your next Block Number should be 352 (they are incremented by 2).

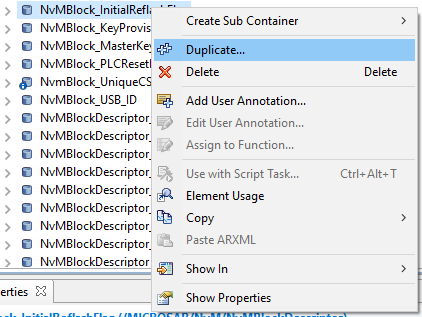
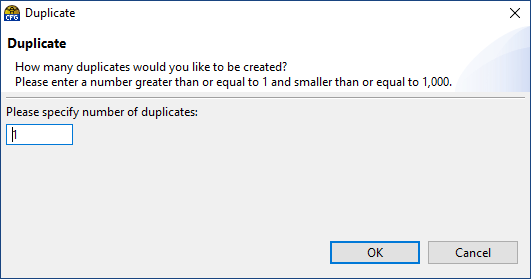
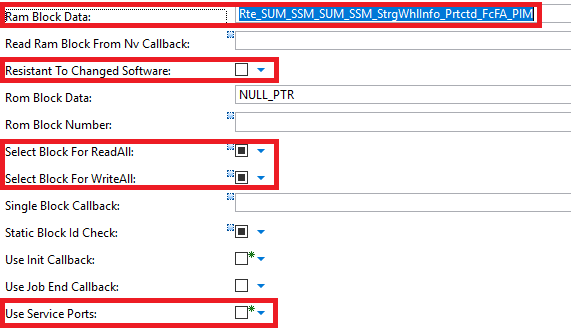
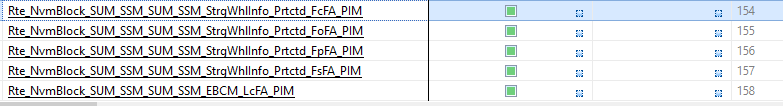


1. Open the Ea/EaBlockConfigurations node.
2. Right-click on the block with the highest Block ID. In this case it is “Unused\_1”. Select “Duplicate”.  
     
   
3. Click OK to create one duplicate.  
     
   
4. Click on the new block node.
5. **IMPORTANT**: Make sure that the Block Id column assign the next available Block ID. Also make sure that the Block Number is correct (it should be automatically incremented by 2). You need to ensure that the existing blocks have not been renumbered.
6. Set Block Size to 1, Device Index to EepGeneral, and Partition to EaPartitionConfiguration\_ASWOnly.  
     
   
7. Repeat the above steps until you have blocks for FcFA, FoFA, FpFA, FsFA, and the LcFA signal.  
     
   

### Configurator: Configuring NvM (Non-Volatile Memory Manager)

If you have changed the Ea configuration above, you will also need to add corresponding NvM blocks. If not, you can skip this section. First you must determine the last used Block Identifier so that you can use the next one in the sequence. Click on the NvM/NvMBlockDescriptors node. Sort the list by the Block Identifier column. Note the last value being used. In this example it is 153, so you should be using 154 and above for your new NvM blocks.

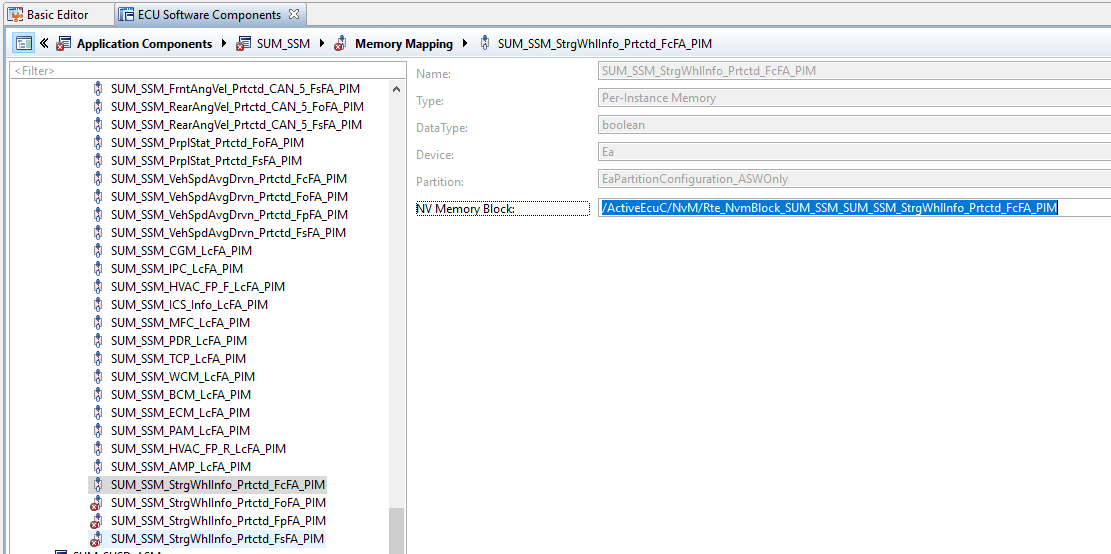


1. Open the NvM/NvMBlockDescriptions node.
2. Locate the NvM block with the highest Block Identifier. In this example, it is “NvMBlock\_InitialReflashFlag”. Right-click on the block and select “Duplicate”.  
     
   
3. Click OK to create one duplicate.  
     
   
4. Click on the new block.
5. Change the Short Name.
6. Make sure that the Block Length is 1.
7. Change the Ram Block Data to the appropriate name.
8. Uncheck the “Resistant to Changed Software” parameter if it’s checked.
9. Scroll down and right-click on the “Select Block for ReadAll” parameter. Select “Delete”.
10. Delete the “Select Block for WriteAll” parameter as well.
11. If you are configuring the FcFA, FoFA, FpFA, or FsFA, uncheck the “Use Service Ports” parameter if it’s checked. If you are configuring the LcFA signal, make sure “Use Service Ports” is checked.  
      
    
12. In the /NvMTargetBlockReference/NvMEaRef node, map the block to the corresponding block in Ea.
13. Repeat the above steps until you have blocks for FcFA, FoFA, FpFA, FsFA, and the LcFA signal. **Make sure that the new NvM blocks use the next Block Identifiers in the sequence, and that the existing NvM blocks were not renumbered.**  
    
14. Save and synchronize the project.

**Note**: Unfortunately DaVinci Configurator is a bit finicky at this point. It will try and automatically assign the Block Identifier to a new NvM block, and it may not select the next Block Identifier at the end of the range if an existing NvM block has been deleted earlier; rather it may select one in the middle of the range if that value is unused. If it does this, then it will renumber all the NvM blocks after it. **You do not want this.** Be careful as you create the new NvM blocks. While you are adding NvM blocks, it is helpful to use Git to incrementally commit your changes to the ARXML files. That way you can go back if you make a mistake. Pay close attention to the changes the tool makes to the ARXML files. If you see lots of changes to many NvM blocks, then the tool has probably renumbered the blocks and you need to go back.

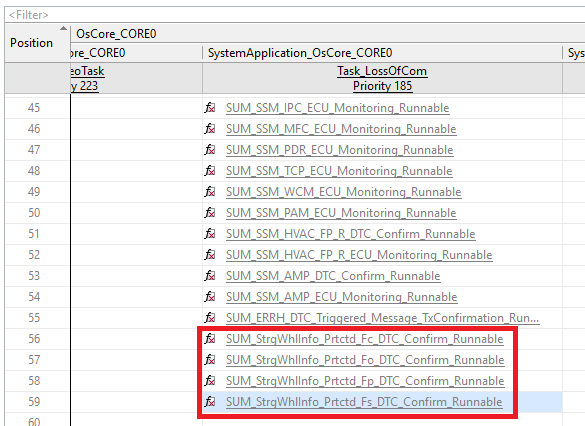
### Configurator: Completing the NvM Block Mapping for SUM\_SSM

If you created new NvM blocks in the previous section, you will need to map these to the SUM\_SSM memory map. If you skipped the previous section, you can skip this section as well.

1. In the Validation messages, double-click on one of the messages “The value of reference RteNvmBlockRef is missing or empty”.
2. Assign the appropriate NvM blocks.  
     
   

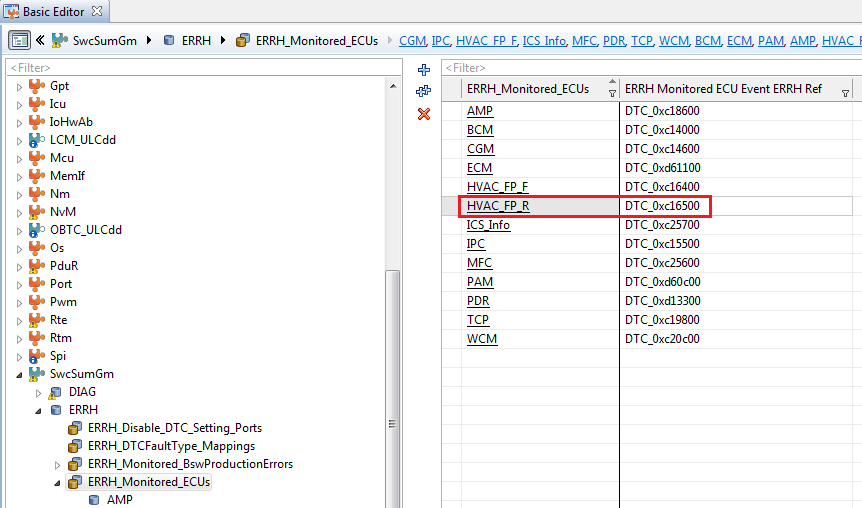
### Configurator: Mapping the Runnable Entities for SUM\_SSM

In the Validation messages you will see several errors saying “The runnable entity <…> is not mapped to a valid task.” To fix these errors, do the following:

1. In the Configuration Editors pane, open Runtime System and click Task Mapping.
2. To map a runnable function, drag and drop the functions into any free positions in the list.  
     
     
     
   Complete the task mapping according to the table below.

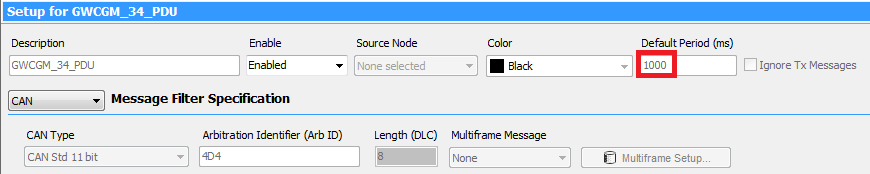
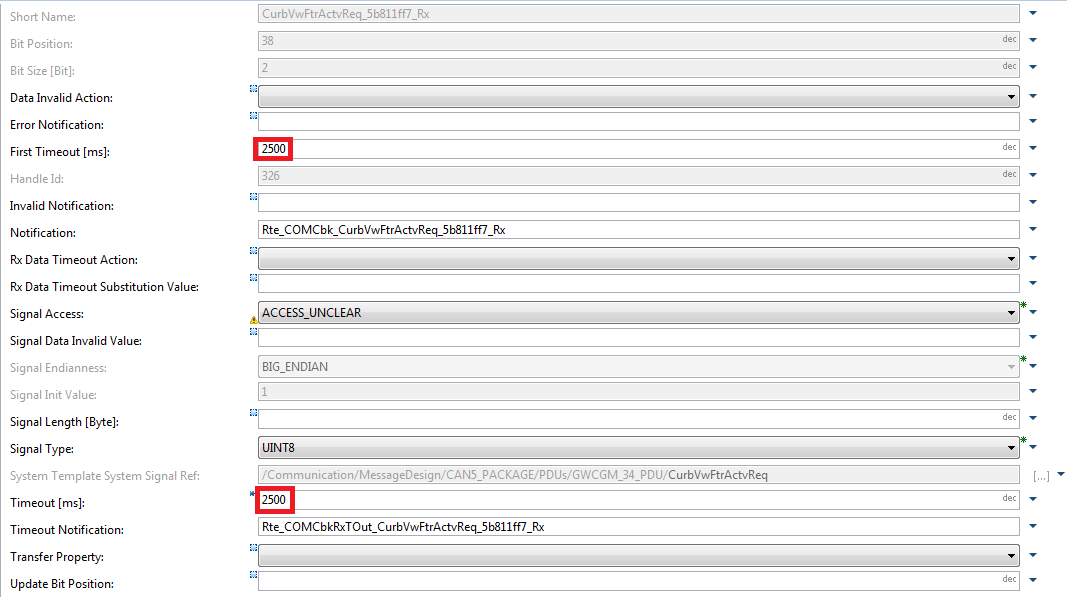
|  |  |  |
| --- | --- | --- |
| **Runnable** | **Component** | **Map to this Task** |
| SUM\_<Signal>\_Fc\_DTC\_Confirm\_Runnable | SUM\_SSM | Task\_LossOfCom |
| SUM\_<Signal>\_Fo\_DTC\_Confirm\_Runnable | SUM\_SSM | Task\_LossOfCom |
| SUM\_<Signal>\_Fp\_DTC\_Confirm\_Runnable | SUM\_SSM | Task\_LossOfCom |
| SUM\_<Signal>\_Fs\_DTC\_Confirm\_Runnable | SUM\_SSM | Task\_LossOfCom |
| SUM\_SSM\_<ECU>\_DTC\_Confirm\_Runnable | SUM\_SSM | Task\_LossOfCom |
| SUM\_SSM\_<ECU>\_DTC\_Monitoring\_Runnable | SUM\_SSM | Task\_LossOfCom |
| SUM\_SOH\_RollingCountError\_<Signal>\_Fol\_Runnable | SUM\_SOH | Task\_100MS |

### Configurator: Configuring the SUM\_ERRH

1. Expand the SwcSumGm/ERRH/ERRH\_Monitored\_ECUs node in the Basic Editor.
2. Click the ‘plus’ button and add the Monitored ECU. Map the ECU to the appropriate Monitored ECU event for the DTC.  
     
   
3. Synchronize and save the project again.

### Configurator: Changing Signal Timeouts in COM

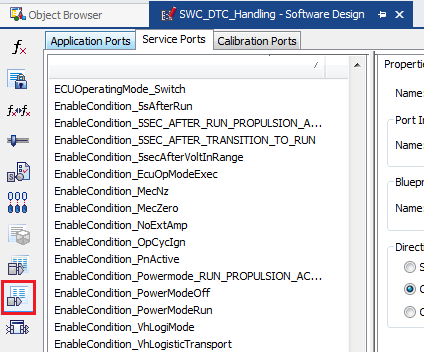
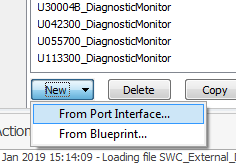
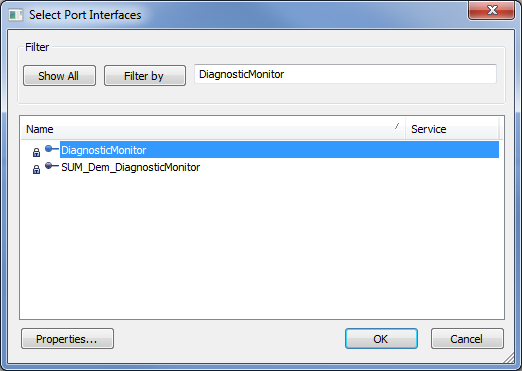
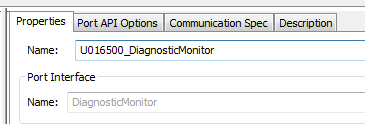
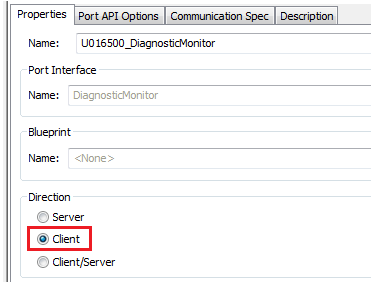
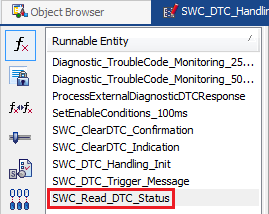
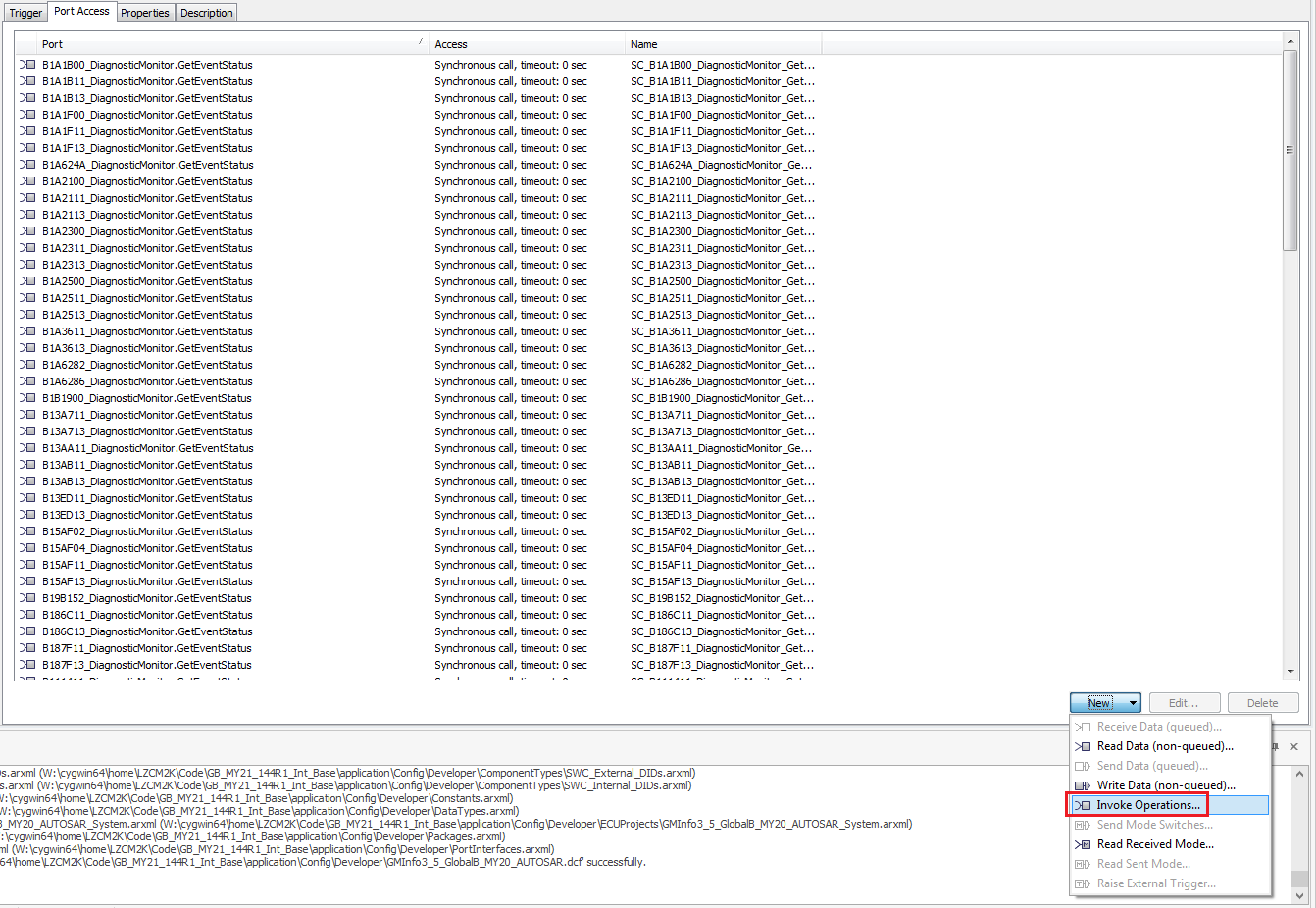
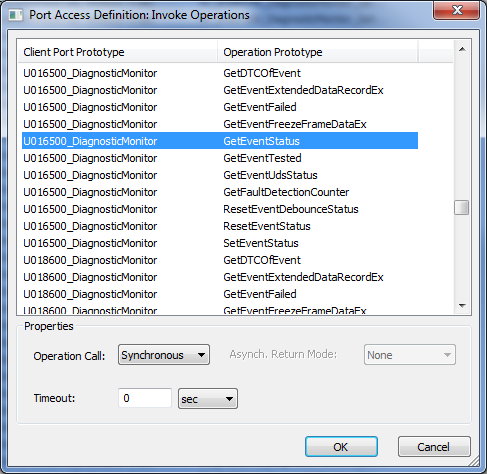
All supervised signals are periodic. You need to make sure that the timeouts specified for the supervised signals in the COM BSW component are at least 2.5 times the signal periodicity. This will prevent the possibility of falsely setting a Loss of Com DTC if the signal timing is delayed on the CAN bus. For example, if the supervised signal “CurbVwFtrActvReq” is sent by the BCM at a periodic rate of 1000 ms, the specified timeouts in COM should be 2500 ms.

1. In the Basic Editor, search for the name of the supervised signal in the Filter box.
2. Click on the signal name under Com/ComConfig/ComSignals or Com/ComConfig/ComSignalGroups if the signal is part of a group.
3. Right-click on the “Timeout [ms]” parameter and select “Set User Defined.”
4. Set the timeout to 2.5 times the periodicity of the signal. If you do not know the periodicity of the signal, you can check it in Vehicle Spy. Import the latest ARXML database, then check the “Database” section in the Messages Editor window. The Default Period should be displayed in the message setup pane.  
     
   
5. Repeat the above two steps for the “First Timeout [ms]” parameter.  
     
   

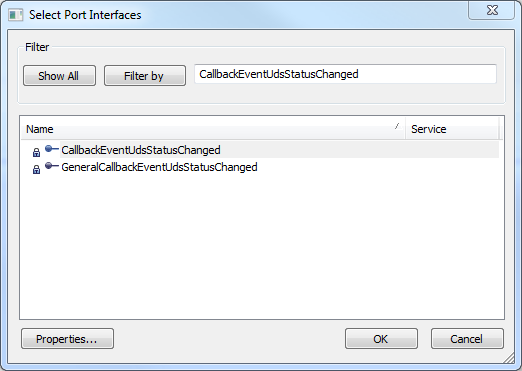
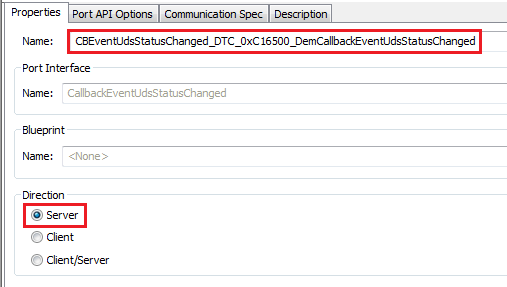
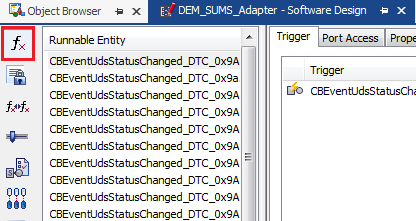
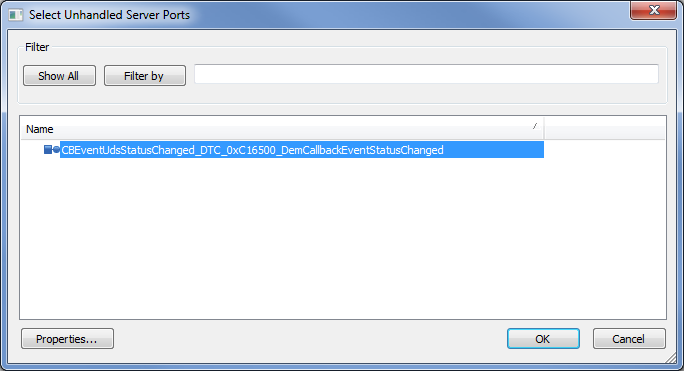
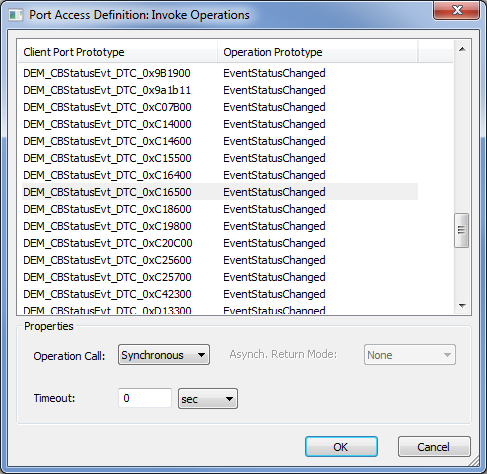
The First Timeout parameter ensures that the VIP can detect the loss of the signal even if it has never received it. If this parameter is left blank, then the VIP will not immediately set the Loss of Com DTC upon startup – you will have to start the supervised signal, then stop it again before the VIP sets the DTC.  
  
NOTE: The Notification and Timeout Notification fields will be empty at first if the signal is new. After you have generated the code for COM, these two fields will be filled in automatically. These are the names of the callbacks used for notifications when the signal is received, or when a timeout occurs. These are necessary for the Loss of Com DTC to work.

NOTE: The Notification and Timeout Notification callbacks will not be generated if the supervised signal(s) are not properly mapped down the component hierarchy to SUM\_SSM. See the section “Developer: Routing Signals to SUM\_SSM” for details.

### Developer: Making Changes to SWC\_DTC\_Handling

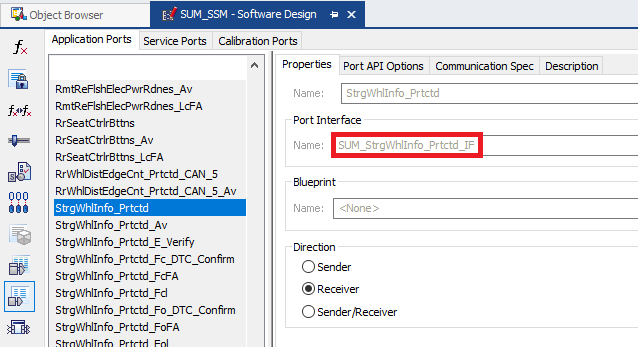
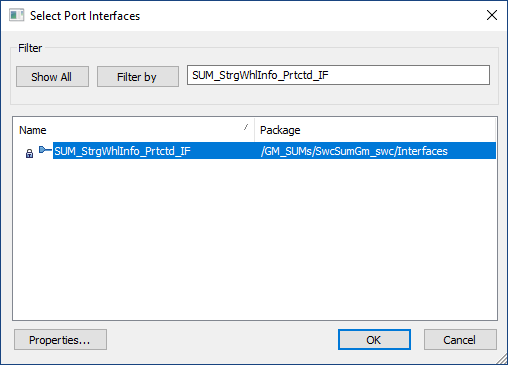
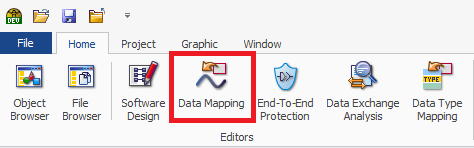
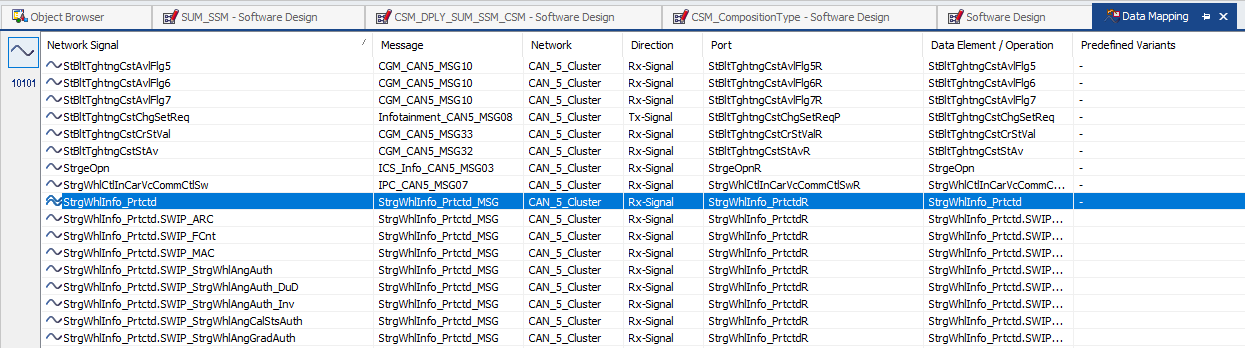
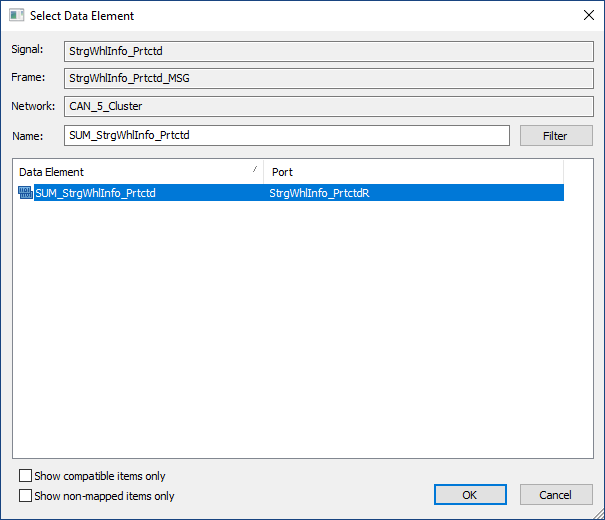
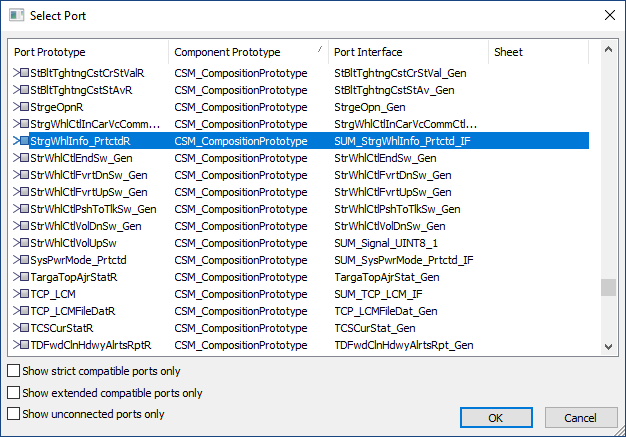
1. Open the project in DaVinci Developer.
2. In the Object Browser, open the Application Component Types node, and double-click on the SWC\_DTC\_Handling component. The Software Design window will open for this component.
3. Open the Port Prototype list and select the Service Ports tab.  
     
   
4. Click the New button and select “From Port Interface”.  
     
   
5. Select the “DiagnosticMonitor” port interface.  
     
   
6. Update the name of the port in the Properties tab.  
     
   
7. Make sure that your new port is Client. It should be by default.  
     
   
8. In the Runnable Entity List, click on SWC\_Read\_DTC\_Status.  
     
   
9. Open the Port Access tab. Click the New button and select “Invoke Operations”.  
     
   
10. Search for your new Client Port Prototype. Select the one with Operation Prototype = GetEventStatus. Set the Timeout unit to sec.  
      
    
11. Save the project and close the Software Design for SWC\_DTC\_Handling.

### Developer: Making Changes to DEM\_SUMS\_Adapter

1. In the Object Browser, open DEM\_SUMS\_Adapter.
2. Open the Port Prototype List and select Service Ports.
3. Click on the New button and select “From Port Interface”.
4. Select the “CallbackEventUdsStatusChanged” port interface.  
     
   
5. Change the name to “CBEventUdsStatusChanged\_DTC\_0xXXXXX\_DemCallbackEventStatusChanged” and the direction to Server:  
     
   
6. Click on the New button and select “From Port Interface”.
7. Select the “SUM\_Dem\_CallbackEventStatusChange” Port Interface.
8. Change the name to “DEM\_CBStatusEvt\_DTC\_0xXXXXXX” and make sure that the direction is Client.
9. Open the Runnable Entity List.  
     
   
10. Click the New button and select “Server Runnables”.
11. Select the Server Port. You can leave the prefix and postfix empty.  
      
    
12. Select the new Server Runnable from the list and open the Port Access tab.
13. Click the New button and select “Invoke Operations”.
14. Select the appropriate Port Access Definition. Change the Timeout unit to sec.  
      
    

### Developer: Routing Signals to SUM\_SSM

When implementing Loss of Com DTCs, the supervised signal(s) must be routed down the component hierarchy to SUM\_SSM. Follow the below steps for each supervised signal.

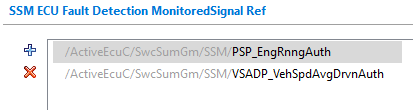
1. In DaVinci Developer Object Browser, open the SUM\_SSM component software design.
2. Open the Port Prototype List and scroll down to find the port corresponding to one of the supervised signals. Make a note of the Port Interface name.  
     
   
3. Open the software design for CSM\_DPLY\_SUM\_SSM\_CSM.
4. Open the Port Interface List.
5. Click the New button and select “From Port Interface”.
6. Select the port interface you noted above.  
     
   
7. Change the name of the port to match the signal name.
8. Change the direction to “Receiver”.
9. Open the software design for the CSM\_CompositionType component and open the Port Prototype List. If a port does not exist for the supervised signal, then repeat steps 4 to 8 above to add it. If it does exist, make sure that the port interface matches the one used in SUM\_SSM and that the direction is Receiver.
10. Repeat the above step for the CSMEcuComp component. You should now have matching ports on all four levels of the component hierarchy. This can be visualized as follows:  
    
11. The incoming signal is mapped to the port on the CSMEcuComp which is the highest level component. Since the port type has been changed, the signal needs to be remapped. To do this, Click the “Data Mapping” button on the top ribbon.  
      
    
12. Switch to Signal View Mode (the icon that looks like a sine wave on the leftmost pane). Scroll down in the list and locate the signal.  
      
    
13. Right-click on the signal and select “Unmap Data Element”. The mapping will be removed.
14. Right-click on the signal again and select “Select Data Element.
15. Select the appropriate data element.  
      
    
16. The last step is to connect the ports to each other. Start by opening the software design for CSMEcuComp.
17. Open the Connector Prototype List (P-Port View).
18. Find the port prototype corresponding to the signal. Right-click on the port prototype and select “New Connector Prototype”.
19. Select the appropriate port on CSM\_CompositionPrototype.  
      
    
20. Within CSM\_CompositionType, connect the signal port to the appropriate port on CSM\_DPLY\_SUM\_SSM\_CSM.
21. Within CSM\_DPLY\_SUM\_SSM\_CSM connect the signal port to the appropriate port on SUM\_SSM. Your components should now be configured like this:  
    

### Configurator: Multiple ECUs Monitoring a Signal

There may be a scenario in which two or more ECUs may supervise the same monitored signal. This is perfectly valid. It represents the situation in which different ECUs in different vehicle configurations may broadcast the same signal. Setting up SUM\_SSM to handle this situation requires a couple additional steps. You may skip these steps if you are not dealing with this situation.

Looking at an example will help. In the MY23 CSM, the ECP and the ECM are both set up to monitor the same set of signals:

* PSP\_EngRnngAuth. Part of signal group SrlDat22\_Prtctd.
* PSP\_RmtVehStrtEngRnngAuth. Also part of signal group SrlDat22\_Prtctd.
* VSADP\_VehSpdAvgDrvnAuth. Part of signal group VehSpdAvgDrvn\_Prtctd.

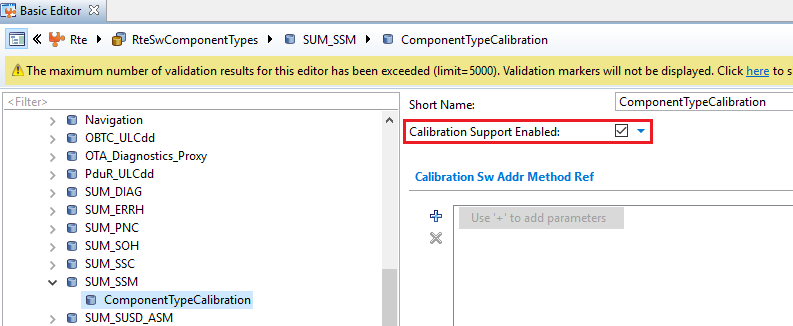


Note that PSP\_RmtVehStrtEngRnngAuth is not included in the above list because it is part of the same signal group (and PDU) as PSP\_EngRnngAuth.

To allow this kind of signal sharing, you must enable Calibration Support for SUM\_SSM if it hasn’t already been enabled previously. In the Basic Editor DaVinci Configurator, click the following node:

Rte/RteSwComponentTypes/SUM\_SSM/ComponentTypeCalibration

Check the Calibration Support Enabled checkbox if it hasn’t already been checked.



After doing this, the RTE will generate a set of calibration parameters for SUM\_SSM. Specifically, it will generate these parameters for each shared supervised signal:

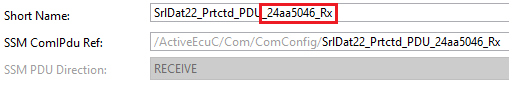
Rte\_CData\_SSM\_C\_SrlDat22\_Prtctd\_PDU\_24aa5046\_Rx\_IsReceivedFrom\_ECM  
Rte\_CData\_SSM\_C\_SrlDat22\_Prtctd\_PDU\_24aa5046\_Rx\_IsReceivedFrom\_ECP

These parameters essentially tell the SUM\_SSM from which ECU the signal originates. These can be set using the values of the DTC\_MASK calibrations. This will be shown below in the Manual Code Changes section.

Now open the following node:

SwcSumGm/SSM/SSM\_PDUs

Look for the PDU to which the shared supervised signal belongs. In this example, it is SrlDat22\_Prtctd\_PDU. Add the string “<hash>\_Rx” to the end of the Short Name, as below.



Repeat this for all shared supervised signals.

Why is this step necessary? It is because there seems to be a flaw in the current code generator for SUM\_SSM. If you were to generate all the code and build it, you may find that you get the following linker errors:

\_Rte\_CData\_SSM\_C\_SrlDat22\_Prtctd\_PDU\_IsReceivedFrom\_ECM from SUM\_SSM.o \_Rte\_CData\_SSM\_C\_SrlDat22\_Prtctd\_PDU\_IsReceivedFrom\_ECP from SUM\_SSM.o

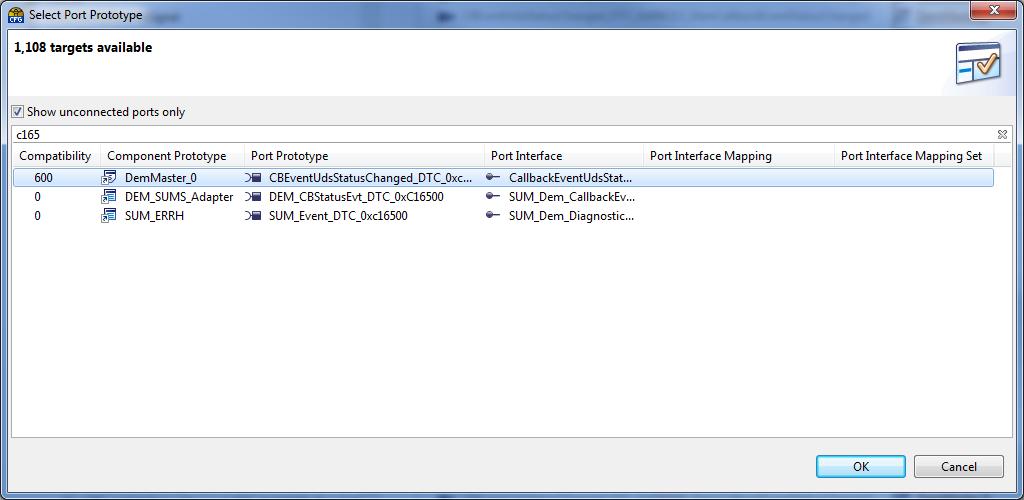
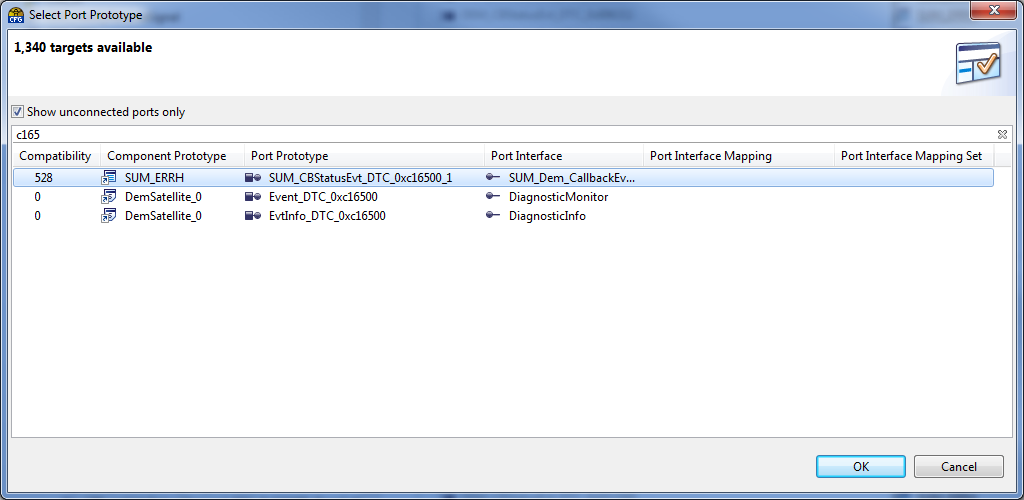
When SUM\_SSM.c is generated, the code generator inserts the incorrect symbol names in a few places in the code. In Rte\_SUM\_SSM.h, the actual symbols are defined as:

Rte\_CData\_SSM\_C\_SrlDat22\_Prtctd\_PDU**\_24aa5046\_Rx**\_IsReceivedFrom\_ECM  
Rte\_CData\_SSM\_C\_SrlDat22\_Prtctd\_PDU**\_24aa5046\_Rx**\_IsReceivedFrom\_ECP

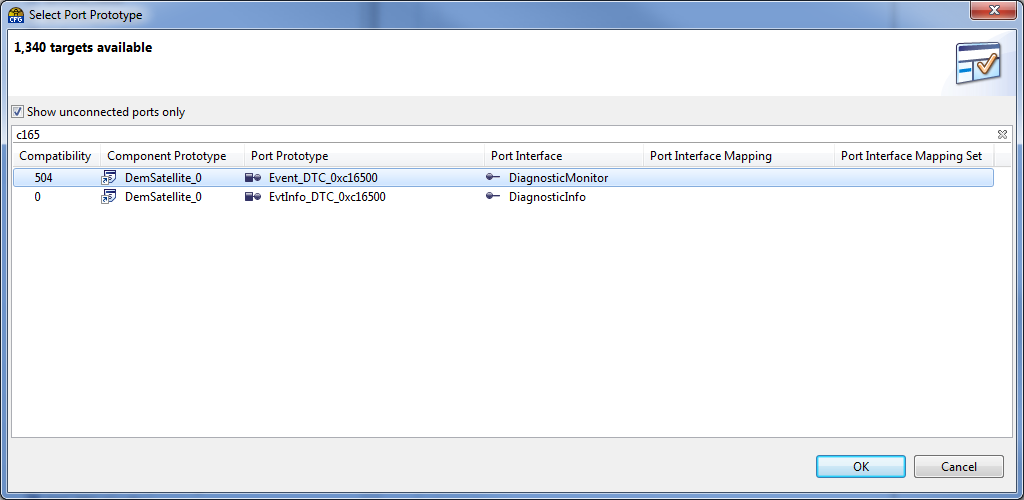
Adding the missing portion of the name in the PDU in SUM\_SSM is a way to work around this limitation.

The last step in dealing with a signal sharing scenario is to link the values of the calibration parameters to the actual values of the DTC\_MASK calibrations. This will be shown in the Manual Code Changes section below.

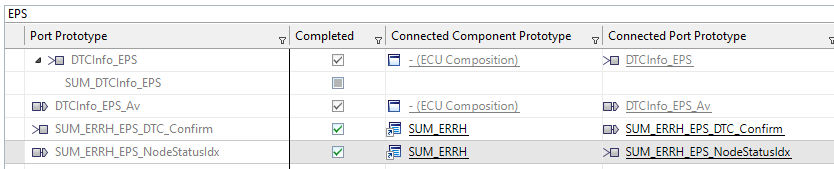
### Configurator: Connecting DEM\_SUMS\_Adapter

1. Open the project in DaVinci Configurator.
2. Synchronize the project, if necessary.
3. Open Runtime System and click ECU Software Components.
4. Open the Application Components/DEM\_SUMS\_Adapter node. Click on the Service Ports node.
5. Connect the “CBEventUdsStatusChanged\_DTC\_0xXXXXX\_DemCallbackEventStatusChanged” Port to the appropriate port on DemMaster\_0.  
     
   
6. Connect the “DEM\_CBStatusEvt\_DTC\_0xXXXXXX” port to the appropriate port on SUM\_ERRH.  
     
   

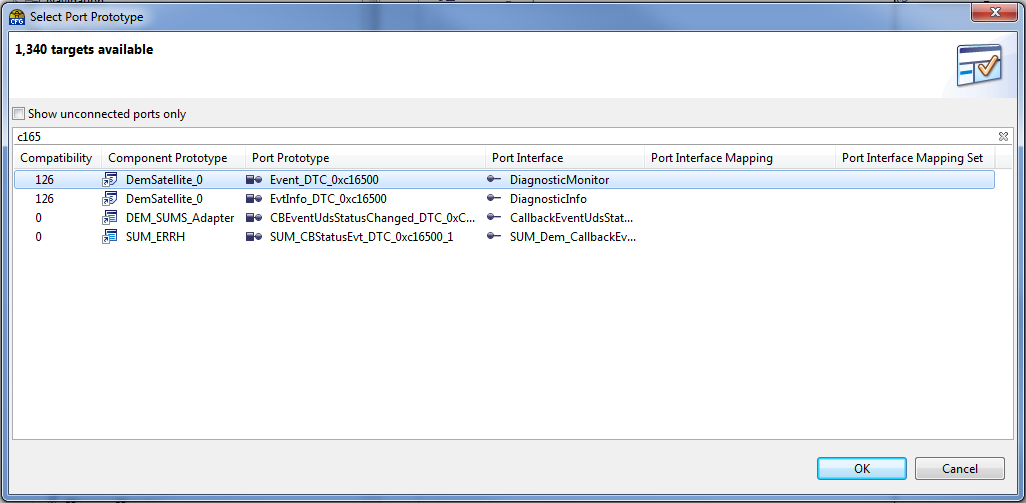
### Configurator: Connecting SUM\_ERRH

1. In the component list, open the SUM\_ERRH node and click Service Ports.
2. Connect the “SUM\_Event\_DTC\_0xXXXXXXX” port to the “Event\_DTC\_0xXXXXXX” port on DemSatellite\_0.  
     
   

### Configurator: Connecting SUM\_SSM

1. In the component list, open the SUM\_SSM node and click Application Ports.
2. Connect the SUM\_ERRH\_XXX\_DTC\_Confirm port to the appropriate port on SUM\_ERRH.
3. Connect the SUM\_ERRH\_XXX\_NodeStatusIdx port to the appropriate port on SUM\_ERRH.  
     
   

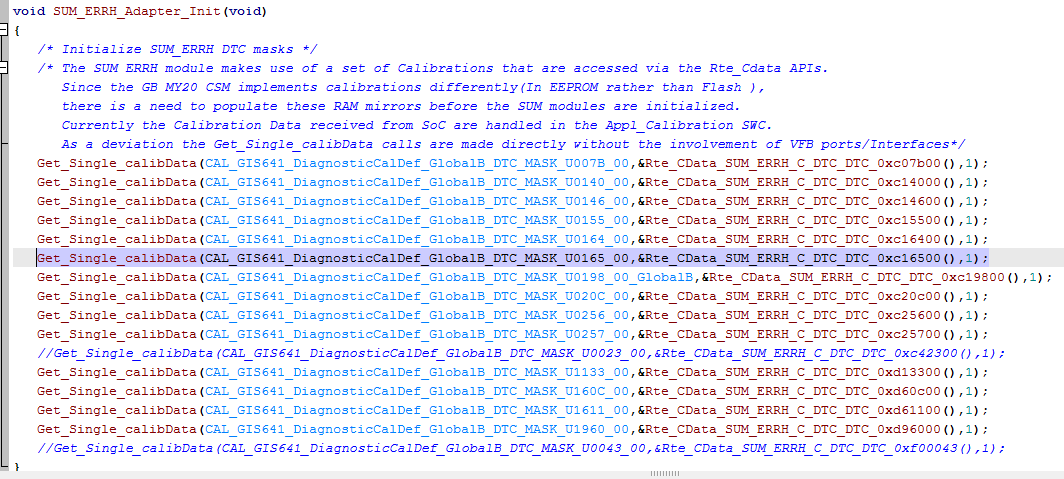
### Configurator: Connecting SWC\_DTC\_Handling

1. In the component list, open the SWC\_DTC\_Handling node and click Service Ports.
2. Connect the “XXXXXXXXX\_DiagnosticMonitor” port to the “Event\_DTC\_0xXXXXXX” port on DemSatellite\_0. You will need to uncheck the “Show connected ports only” checkbox to find it in the list.  
     
   

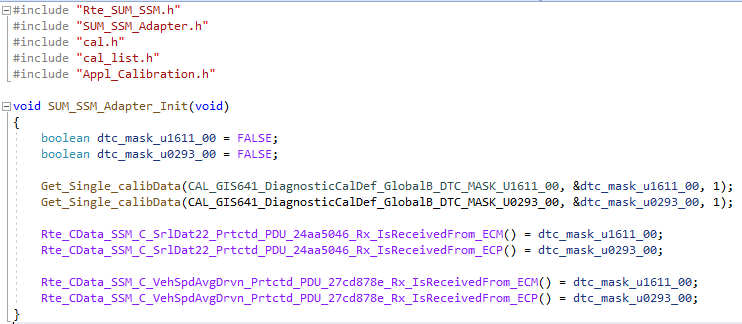
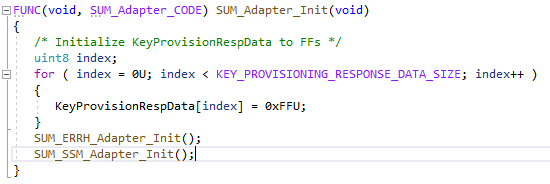
### Configurator: Generating the Code

1. Generate the code for SwcSumGm, Dem, Com, Ea, NvM, Os, and Rte.
2. Generate the Template File for DEM\_SUMS\_Adapter and SWC\_DTC\_Handling.
3. Save the project and close DaVinci Configurator.

## Manual Code Changes

1. Edit the following file: application\Appl\Source\SUM\_Adapter\SUM\_ERRH\_Adapter.c
2. Add the call to Get\_Single\_calibData to the SUM\_ERRH\_Adapter\_Init function:  
     
   
3. Edit the following: application\Appl\Source\SWC\_DTC\_Handling.c
4. Add the DTC to the switch statement in the SWC\_Read\_DTC\_Status function.
5. Edit the following: application\Appl\Source\DEM\_SUMS\_Adapter.c
6. Fill in the implementation for the CBEventUdsStatusChanged\_DTC\_0xXXXXXX\_DemCallbackEventStatusChanged\_CallbackEventUdsStatusChanged runnable.

If you have set up multiple ECUs to monitor a shared signal (according to the instructions in the “Configurator: Multiple ECUs Monitoring a Signal” section), then also do the following:

1. In the application\Appl\Source\SUM\_Adapter folder, add SUM\_SSM\_Adapter.h and SUM\_SSM\_Adapter.c (if they have not already been added).
2. The code in SUM\_SSM\_Adapter.h should look like this:  
     
   #ifndef INC\_SUM\_SSM\_ADAPTER\_H  
   #define INC\_SUM\_SSM\_ADAPTER\_H  
     
   void SUM\_SSM\_Adapter\_Init(void);  
     
   #endif
3. In SUM\_SSM\_Adapter.c, implement the SUM\_SSM\_Adapter\_Init() function. It should look something like this:  
     
     
     
   Here you are reading the values of the DTC\_MASK calibrations and setting the values to the SUM\_SSM calibration parameters.
4. Finally, open SUM\_Adapter.c. Include SUM\_SSM\_Adapter.h. Locate the SUM\_Adapter\_Init function and call the SUM\_SSM\_Adapter\_Init function.  
     
   

Once all of the above changes are complete, you may build and test your code.

## Testing Your Changes

The following is a checklist of things you should test when implementing a new Loss of Com DTC. Other tests may be needed as well, depending on the specifics of your implementation. Please use your own judgment and discretion.

1. Verify that the new DTC is included in the Supported DTCs list (using UDS Service $19).
2. Power up the VIP but do not send the supervised signal(s) initially. Verify that the Loss of Com DTC is set 5 seconds after the transition to the Ignition Run state. You should not have to start the signal(s) then stop them again. The DTC should be set immediately.
3. Disable the DTC\_MASK calibration for the new DTC. Enable all others. Make sure that the DTC is not set when the supervised signals are not being sent. Make sure that no other Loss of Com DTCs are affected.
4. Enable the DTC\_MASK only for the new Loss of Com DTC. Disable all others. Verify that the DTC can be set and cleared and that no other Loss of Com DTCs are triggered.
5. Verify DTC ‘Test Failed’ status flag is set when all supervised signal(s) are not sent.
6. Verify DTC ‘Test Failed’ status flag is cleared when all supervised signal(s) are sent.
7. Stop each supervised signal one at a time. Send all others. Verify that the DTC ‘Test Failed’ flag is set for each signal. This will ensure that none of the signals are missing from the implementation.
8. Set and confirm 8 DTCs (status $2F), other than the Loss of Com DTC you are testing. Verify the DTCConfirmed bit is not set for the Loss of Com DTC when all 8 storage slots are occupied by other DTCs.
9. Make sure less than 8 DTCs are set and confirmed (status $2F or $2E). Verify the DTCConfirmed bit is set when less than 8 storage slots are occupied by other DTCs.
10. Verify the DTCConfirmed bit is set when a storage slot becomes available after all 8 have been occupied by other DTCs.
11. Verify that the DTCInfo CAN signal is sent with the appropriate DTC Number when the DTC is set.
12. Verify that the Timestamp is updated in Extended Data Record $01 the first time the DTC is set and confirmed (after initial power up and after a Clear Diagnostic Information request).
13. Verify that the Timestamp is not updated in Extended Data Record $01 on the subsequent times the DTC is set.
14. Verify that the Occurrence Counter is updated in Extended Data Record $02 every time the DTC is set.
15. Verify that the Aging Cycle Counter is incremented in Extended Data Record $03 on every ignition cycle, if the Test Failed flag is False.
16. Verify that the Aging Cycle Counter is NOT incremented in Extended Data Record $03 on every ignition cycle, if the Test Failed flag is True.
17. Perform 40 ignition cycles. Verify that the DTC status byte is cleared to $00 when the Aging Cycle Counter reaches 40.
18. Verify that the DTC status byte is cleared to $00 when a Clear Diagnostic Information request is received, if the Test Failed flag is False.
19. Verify that the DTC status byte is NOT cleared to $00 when a Clear Diagnostic Information request is received, if the Test Failed flag is True.
20. Verify that the DTC Changed Notification IPC Message (0x06) is sent to the SoC whenever the DTC status changes.
21. Send the Read DTC Status Request IPC Message (0x04) from the SoC to the VIP. Verify that the correct DTC status is returned by the VIP in the Read DTC Status Response IPC Message (0x04).
22. Verify that each of the supervised signals is correctly monitored on each mapped PN.
23. Verify that the DTC is enabled 5 seconds after transition to Run or Propulsion.
24. Verify that the DTC is disabled when in Transport Logistics Mode.
25. Verify that the DTC is disabled during a Bus Off condition.

See RTC task 187372 for a sample Excel sheet containing test results for the above tests for DTC U160F00.

# Appendix A: Mapping Between PN Names and Numbers

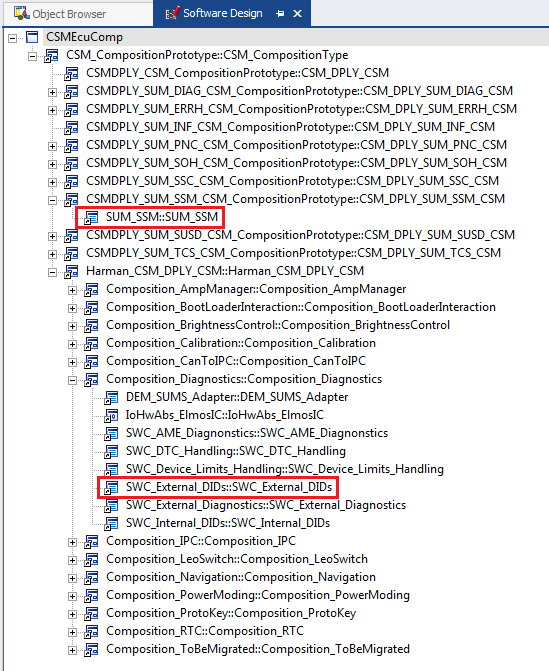
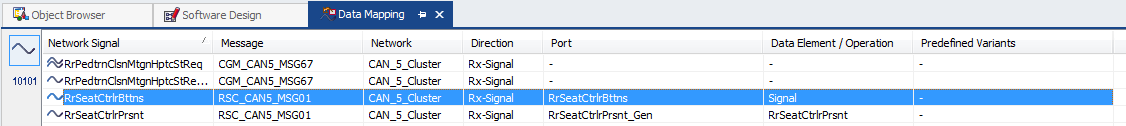
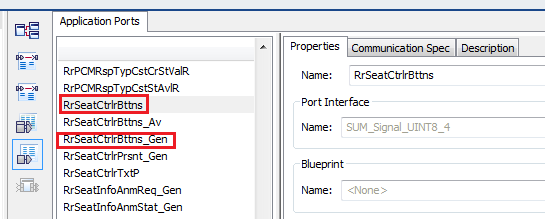
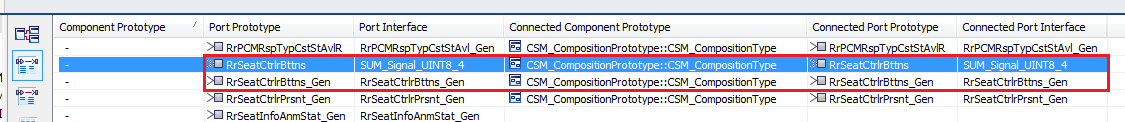
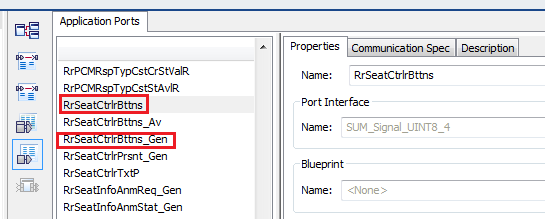
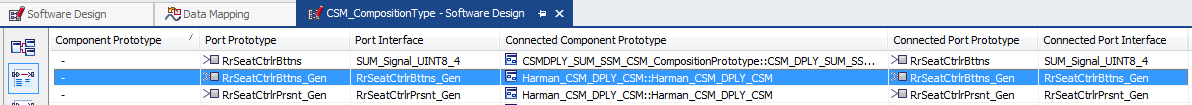
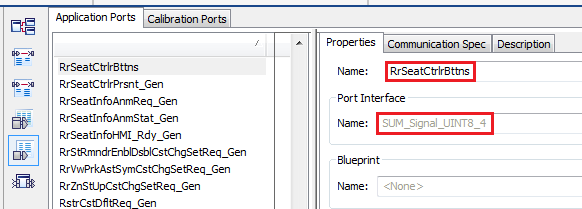
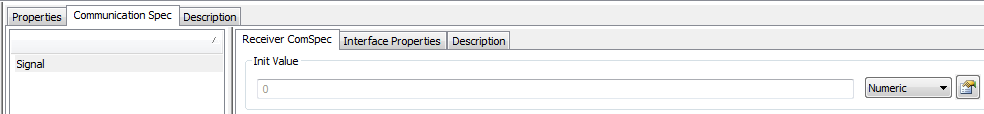
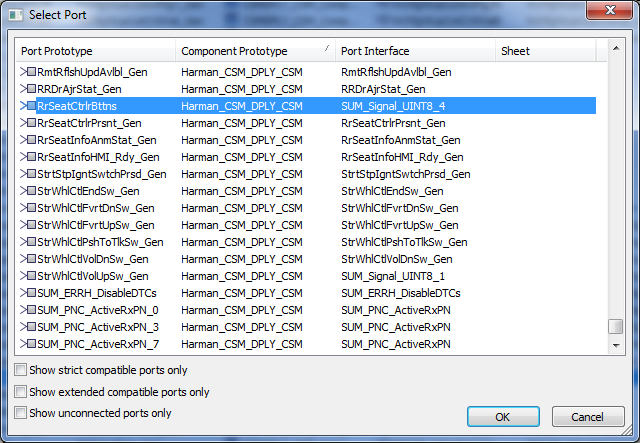
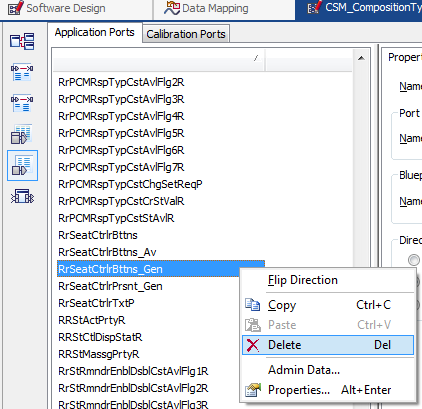
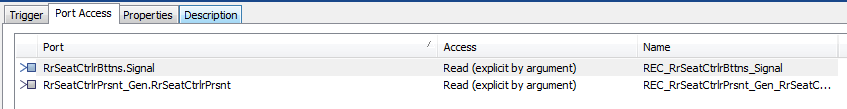
Use the following table to map between PN names and numbers. For additional details, see GB5280.

|  |  |
| --- | --- |
| **PN #** | **PN Name** |
| 0 | Driver Notification |
| 1 | Post Collision Operation |
| 2 | Remote Data Services |
| 3 | Vehicle Access |
| 4 | Exterior Lighting |
| 5 | Short Duration Event |
| 6 | Infotainment |
| 7 | Propulsion |
| 12 | Electrification |
| 13 | Remote Start |
| 14 | Ingress/Egress |
| 15 | Interior Lighting |
| 32 | Diagnose Active Safety Ethernet |
| 33 | Remote Reflash |
| 41 | Diagnose CAN1 |
| 42 | Diagnose CAN2 |
| 43 | Diagnose CAN3/CAN9 |
| 44 | Diagnose CAN4 |
| 45 | Diagnose CAN5 |
| 46 | Diagnose CAN8 |
| 47 | Diagnose Infotainment Ethernet |

# Appendix B: Routing Signals to Multiple SWCs

When implementing Loss of Com DTCs, the supervised signal(s) must be routed down the component hierarchy to SUM\_SSM. In the most basic scenario, the signal(s) in question are only needed by SUM\_SSM. However, in some cases, you may discover that the DTC relies on a CAN signal which is also used by other SWC components. The example that follows uses DTC U016500, which relies on the RrSeatCtrlBttns CAN signal. In order for the Loss of Com DTC to work, the RrSeatCtrlBttns needs to be routed to SUM\_SSM. However, the RrSeatCtrlBttns CAN signal is also used by the SWC\_External\_DIDs component for DID $41DF. So in that case, the signal needed to be shared by two different components.

If you encounter this situation, you will have to make sure that your SWCs are connected in such a way that the required signal(s) are routed correctly to all the SWCs that require it. In this section, I will work through the example of DTC U016500, DID $41DF and the RrSeatCtrlBttns signal. Hopefully this will provide you with a working example which you can generalize to your own situation.

1. Open DaVinci Developer and look at the software design for the CSMEcuComp. Expand the nodes and look for the software components which require the CAN signal. In this case, the signal is required by both SUM\_SSM and SWC\_External\_DIDs. Notice that they occupy different places in the component hierarchy.  
     
     
     
   In order to properly route the signal to each component, you will need to check the ports and connections for each of the components in the hierarchy. In this example:  
     
   SWC\_External\_DIDs -> Composition\_Diagnostics -> Harman\_CSM\_DPLY\_CSM -> CSM\_CompositionType -> CSMEcuComp  
     
   SUM\_SSM -> CSM\_DPLY\_SUM\_SSM\_CSM -> CSM\_CompositionType -> CSMEcuComp  
     
   The RrSeatCtrlBttns signal will need to be routed from the top-level component (CSMEcuComp) down through the hierarchy to each of the components that need it.
2. Open the Data Mapping window in the Signal View Mode. Scroll down and look for the required network signal. Note which port on the top-level component (CSMEcuComp) the signal is being routed to. In this example, it was originally being routed to the “RrSeatCtrlrBttns\_Gen” port. I had to remap it to the “RrSeatCtrlrBttns” port with Data Element = Signal.  
     
     
     
   Note that any given network signal can only be mapped to one port on the top-level component.
3. Go back to the Software Design for CSMEcuComp. Open the Port Prototype List and look for ports named “RrSeatCtrlrBttns”. In this case, there are two:  
     
     
     
   The RrSeatCtrlrBttns port uses the SUM\_Signal\_UINT8\_4 port interface (which maps to the RrSeatCtrlrBttns CAN signal). This is the port to which the RrSeatCtrlrBttns is mapped.  
     
   The RrSeatCtrlrBttns\_Gen port uses RrSeatCtrlrBttns\_Gen port interface. This port doesn’t have any CAN signal mapped to it.  
     
   Note that, in this case, there are two ports on CSMEcuComp which are apparently used for the same signal. You will need to see how these ports are connected to the sub-components all the way down the hierarchy to determine how they are used.
4. Open the Connector Port Prototype List (P-Port View). Scroll down and look for the RrSeatCtrlrBttns port prototype:  
     
     
     
   In this case, both the RrSeatCtrlrBttns and the RrSeatCtrlrBttns\_Gen ports are connected to corresponding ports on the CSM\_CompositionType sub-component. So we need to keep going down the hierarchy.
5. Open the software design for the CSM\_CompositionType sub-component. Open the Port Prototype List and look for RrSeatCtrlrBttns. In this case, it has ports RrSeatCtrlrBttns and RrSeatCtrlrBttns\_Gen as well:  
     
   
6. Check the connections for those ports in the Connector Port Prototype List (P-Port View).  
     
     
     
   The RrSeatCtrlrBttns port is connected to both CSM\_DPLY\_SUM\_INF\_CSM and CSM\_DPLY\_SUM\_SSM\_CSM. The RrSeatCtrlrBttns\_Gen port is connected to Harman\_CSM\_DPLY\_CSM.
7. Continue to follow connections down the hierarchy until you get to the bottom-level components. As you do, it should become clear to you how the signals are routed and whether or not there is a problem. In this example, the routing of RrSeatCtrlrBttns looks like this:  
     
     
     
   So it’s clear that the RrSeatCtrlrBttns signal is being routed down to CSM\_DPLY\_SUM\_INF\_CSM (which doesn’t currently have any sub-components) and SUM\_SSM, but it is not being routed correctly through the hierarchy to SWC\_External\_DIDs. So DID $41DF won’t work properly in this situation. From this diagram, it should be clear how to fix this. You will want something like this instead:  
     
     
     
   This fix will require five steps:  
     
   First, remove all the old connections on the dead path (on the RrSeatCtrlrBttns\_Gen ports all the way down the hierarchy).  
     
   Second, change the ports on Harman\_CSM\_DPLY\_CSM, Composition\_Diagnostics and SWC\_External\_DIDs to use the SUM\_Signal\_UINT8\_4 Port Interface, and rename them “RrSeatCtrlrBttns” (remove “\_Gen”) to match the ports on CSM\_ComponentType and CSMEcuComp. Also make sure you provide an initial value for those ports.  
     
   Third, make the new connections on the path shown in red in the above diagram.  
     
   Fourth, remove the unnecessary “RrSeatCtrlrBttns\_Gen” port on CSM\_ComponentType (DaVinci Developer will not let you delete the port on CSMEcuComp).  
     
   Fifth, remove the old port access for RrSeatCtrlrBttns\_Gen and add it for RrSeatCtrlrBttns to the ReadData runnable for DID $41DF in the SWC\_External\_DIDs component.
8. For the first step, to remove the old connections, open the software design for CSM\_CompositionType and open the Connector Prototype List (P-Prototype List). Scroll down and find the RrSeatCtrlrBttns\_Gen Port Prototype.  
     
     
     
   You will see that this port is connected to the Harman\_CSM\_DPLY\_CSM component, which is one level below. Right-click on the connection and select Delete Connector Prototypes to remove the connection.  
     
   Repeat the same process for the Harman\_CSM\_DPLY\_CSM and Composition\_Diagnostics components.
9. For the second step, open the software design for Harman\_CSM\_DPLY\_CSM and open the Port Prototype List. Scroll down and look for the RrSeatCtrlrBttns\_Gen port. Change the port name to “RrSeatCtrlrBttns” (remove “\_Gen”) and change the Port Interface to SUM\_Signal\_UINT8\_4.  
     
     
     
   Repeat the same process for the Composition\_Diagnostics and SWC\_External\_DIDs components.
10. You will also need to provide an initial value for the modified ports; if you don’t, you will get a validation error in Configurator. To do so, click on the Communication Spec tab and in the Receiver ComSpec tab, provide a valid initial value. In this case, the RrSeatCtrlrBttns port will have a numeric value of 0.  
      
      
      
    Repeat the same process for the Composition\_Diagnostics and SWC\_External\_DIDs components.
11. For the third step, open the software design for CSM\_CompositionType. Open the Connector Prototype List (P-Port View). Right-click on the RrSeatCtrlrBttns Port Prototype and select New Connector Prototype. Connect it to the RrSeatCtrlrBttns Port Prototype on Harman\_CSM\_DPLY\_CSM.  
      
      
      
    Repeat the same process for the Harman\_CSM\_DPLY\_CSM and Composition\_Diagnostics components. That should complete the new connections.
12. For the fourth step, open the software design for CSM\_CompositionType and delete the RrSeatCtrlrBttns\_Gen port in the Port Prototype List:  
      
    
13. For the fifth step, open the SWC\_External\_DIDs software design and open the Runnable Entity List. Scroll down the runnable entity list and search for the following:  
      
    DataServices\_ReadDataByIdentifier\_DID\_41DF\_RearSeatControlButtonActivation\_DataRecord\_ReadData  
      
    Open the Port Access tab. Delete the existing RrSeatCtrlrBttns port. Then click the New button and select Read Data. Add the RrSeatCtrlrBttns.Signal port.  
      
    
14. After you are done with the above steps, you can save the project.
15. Open the project in DaVinci Configurator and generate the code for the Rte and the template file for SWC\_External\_DIDs.
16. In this example, I had to also modify the code in the DataServices\_ReadDataByIdentifier\_DID\_41DF\_RearSeatControlButtonActivation\_DataRecord\_ReadData runnable in SWC\_External\_DIDs.c because after code generation the function:  
      
    Rte\_Read\_RrSeatCtrlrBttns\_Gen\_RrSeatCtrlrBttns  
      
    Changed to:  
      
    Rte\_Read\_RrSeatCtrlrBttns\_Signal