VX1000 DaVinci Integration

Version 1.0
2025-07-04
Application Note AN-IMC-1-504

Author Gunreben, Dominik

Restrictions Public Document

Abstract Using DaVinci to configure the VX1000 AppDriver

Table of Contents

1 Overview VX1000 DaVinci Integration

The VX1000 provides powerful measurement and calibration access to the microcontroller via debug interfaces. For maximum flexibility and optimal measurement results, the VX1000 Application Driver must be integrated into the ECU software. The VX1000 Application Driver is a Complex Device Driver in the AUTOSAR stack. AUTOSAR modules are typically configured using dedicated AUTOSAR configuration tools such as DaVinci Configurator. This application note shall provide the essential information for integrating the VX1000 Application Driver into the ECU Software with the help of DaVinci.

The VX1000 Application Driver configuration with the DaVinci Configurator is optional and allows an easy-to-use configuration of the Application Driver in an AUTOSAR project. If DaVinci is not used within the project, the C4VX1000 Application Driver can easily be configured by directly modifying its _cfg.h files. For the latter, please see the Getting Started Application Notes that are delivered with the VX1000 Application Driver.

2 Installation of the VX1000 Application Driver

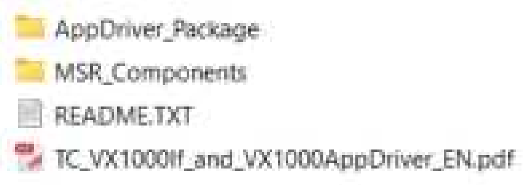
2.1 Getting the VX1000 Application Driver

The VX1000 Application Driver can be downloaded via https://www.vector.com/VX1000If-VX1000AppDriver.

Please note the Product-Specific Special Terms for VX1000If and the VX1000 AppDriver, clearly highlighted on the website or within the driver package.

3 Installation in the Software Integration Package

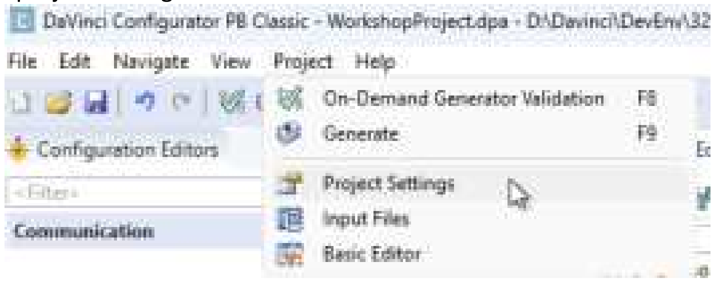
After installation, the installation folder contains two folders



AppDriver_Package
MSR_Components
README.TXT
TC_VX1000If_and_VX1000AppDriver_EN.pdf

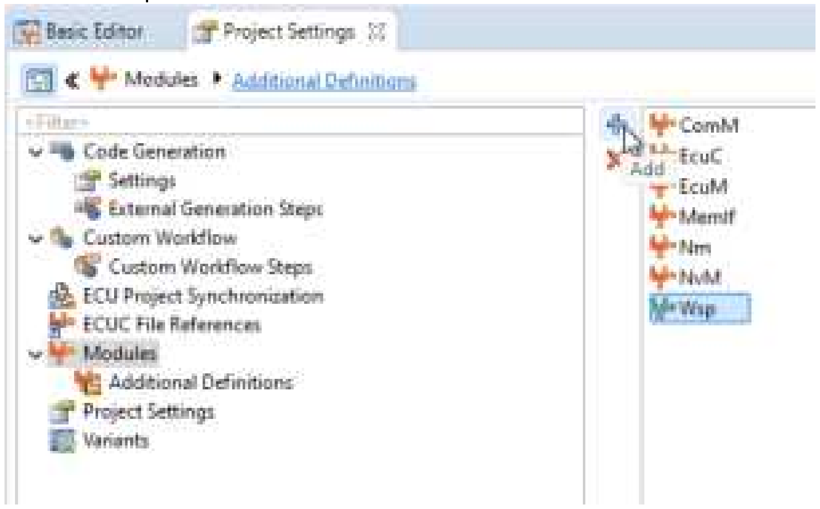
The folder “MSR_Components” is structured to allow seamless copying into the Software Integration Package (SIP) without requiring modifications.

- Ensure that your Davinci-Software is closed
- Copy $Install-Directory/MSR_Components/VX1000 to $SIPLocation/Components/VX1000
- Open DaVinci
- Open the project settings

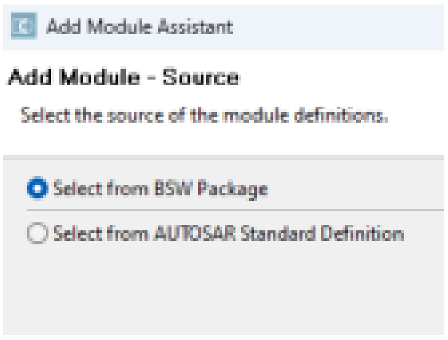

DaVinci Configurator PB Classic - WorkshopProject.dpa - D:\Davinci\DevEnv\32
File Edit Navigate View Project Help
On-Demand Generator Validation F8
Generate F9
Project Settings
Input Files
Basic Editor
Configuration Editors
<Filter>
Communication

  ○

- Select “Modules”and press Add



- Use “Select from BSW package”



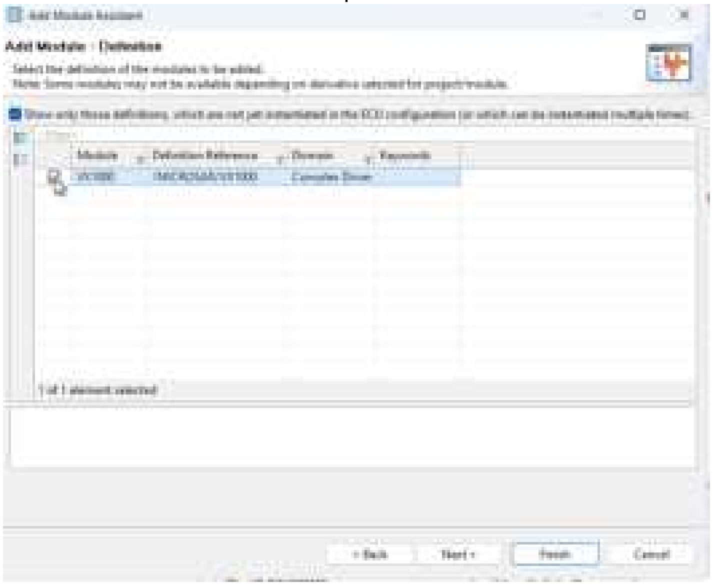
Add Module - Source

Select the source of the module definitions.

[✓ Select from BSW Package]

Select from AUTOSAR Standard Definition

- Select the VX1000 Module and press “Finish”



Add Module - Definition

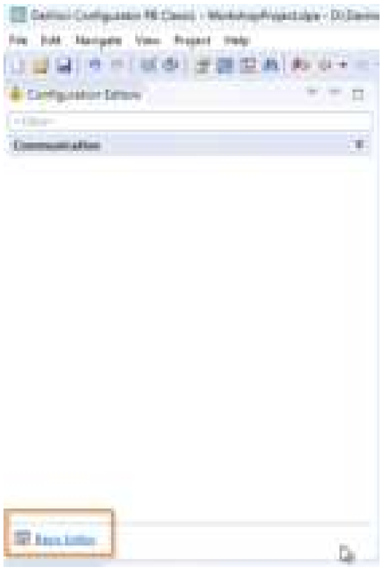
Select the definition of the modules to be added.
Note: Some modules may not be available depending on derivative selected for project/module.

[✓ Show only those definitions, which are not yet instantiated in the ECU configuration (or which can be instantiated multiple times).]

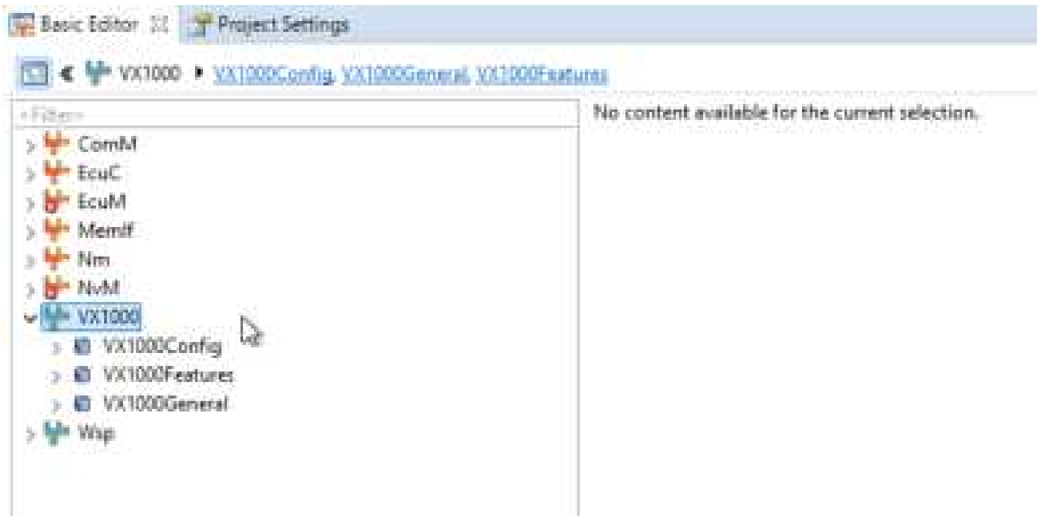
Module	Definition Reference	Domain	Keywords
VX1000	/MICROSAR/VX1000	Complex Driver	

1 of 1 element selected

- Switch to the Basic-Editor.



- Now, the VX1000 Module can be configured like other AUTOSAR modules



No content available for the current selection.

4 Modifications of the build process

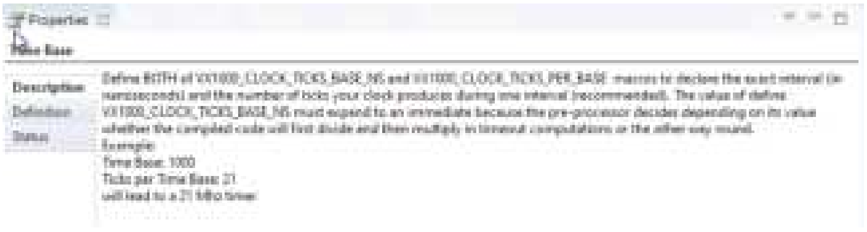
To add the VX1000 Application Driver to the build process the file $(Project)\Appl\Makefile.project.part.defines

must be modified. Add these lines at the end of the file

```
# additional includes for VX1000 driver

ADDITIONAL_INCLUDES += $(ROOT)\Components\VX1000\Implementation

ADDITIONAL_INCLUDES += $(ROOT)\Components\VX1000If\Implementation

# vx1000 driver source files

APP_SOURCE_LST += $(ROOT)\Components\VX1000\Implementation\VX1000.c

APP_SOURCE_LST += $(ROOT)\Components\VX1000If\Implementation\VX1000If.c
```

5 VX1000 Application Driver configuration

The module configuration contains several configuration options. A functional description is provided for each parameter.

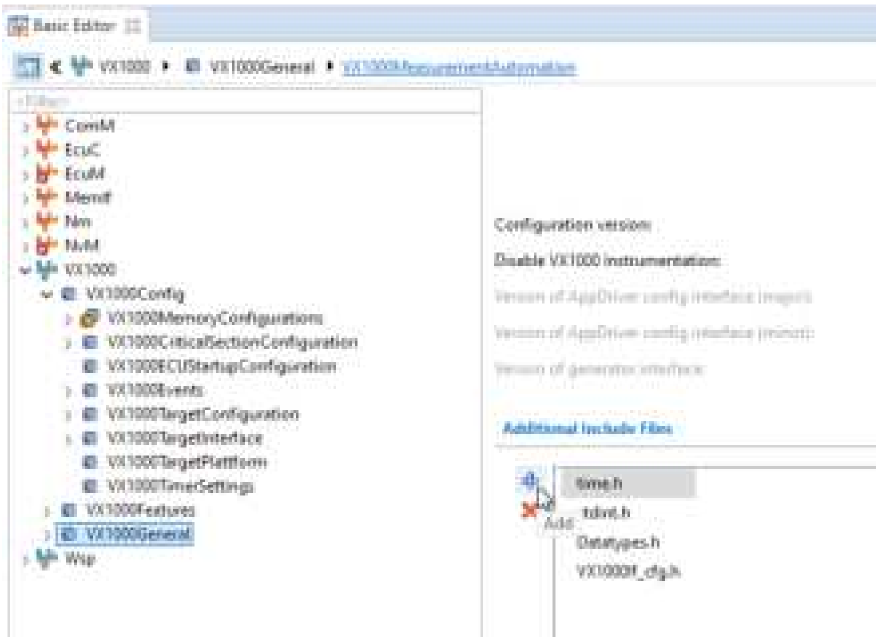


Define BOTH of VX1000_CLOCK_TICKS_BASE_NS and VX1000_CLOCK_TICKS_PER_BASE macros to declare the exact interval (in nanoseconds) and the number of ticks your clock produces during one interval (recommended). The value of define VX1000_CLOCK_TICKS_BASE_NS must expand to an immediate because the pre-processor decides depending on its value whether the compiled code will first divide and then multiply in timeout computations or the other way round.
Example:
Time Base: 1000
Ticks per Time Base: 21
will lead to a 21 MHz timer

The most important steps are described in the later sections of this Application Note. For advanced features, please refer to the corresponding Application Notes or Getting Started Manuals.

5.1 Additional Include Files.

If you require additional header files for your implementation, you can add these easily.



Basic Editor

VX1000 ▸ VX1000General ▸ VX1000MeasurementAutomation

- ComM
- EcuC
- EcuM
- MemIf
- Nm
- NvM
- VX1000
  - VX1000Config
    - VX1000MemoryConfigurations
    - VX1000CriticalSectionConfiguration
    - VX1000ECUStartupConfiguration
    - VX1000Events
    - VX1000TargetConfiguration
    - VX1000TargetInterface
    - VX1000TargetPlattform
    - VX1000TimerSettings
  - VX1000Features
  - VX1000General
- Wsp

Configuration version:

Disable VX1000 instrumentation:

Version of AppDriver config interface (major):

Version of AppDriver config interface (minor):

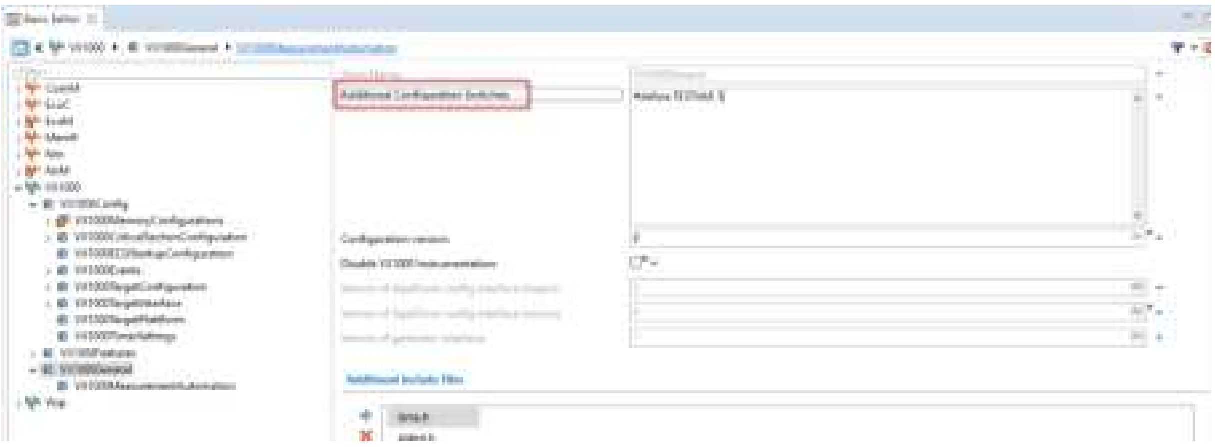
Version of generator interface:

Additional Include Files

- time.h
- tdint.h
- Datatypes.h
- VX1000If_cfg.h

5.2 Additional Code includes

If some additional code blocks are required for the configuration, these can be added within “Additional Configuration Switches”. This block will be copied without modification to the generated file.



Additional Configuration Switches

Configuration version

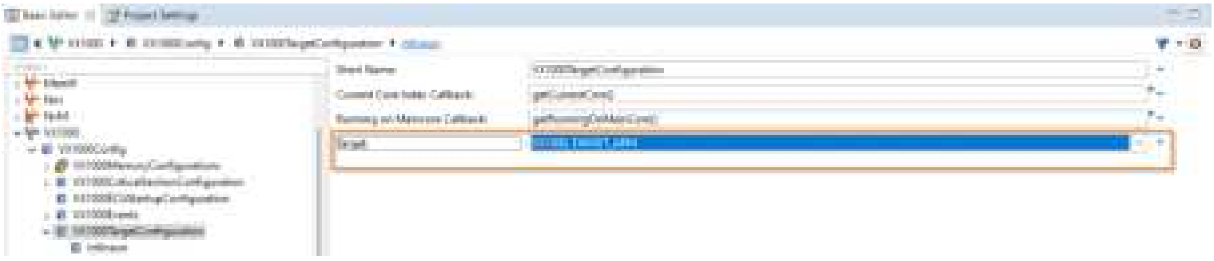
Disable VX1000 instrumentation

Additional Include Files

5.3 Target controller configuration

The VX1000 supports many different target controller families and architectures. The correct target can be configured through

VX1000/VX1000Config/VX1000TargetConfiguration[VX1000_TARGET]



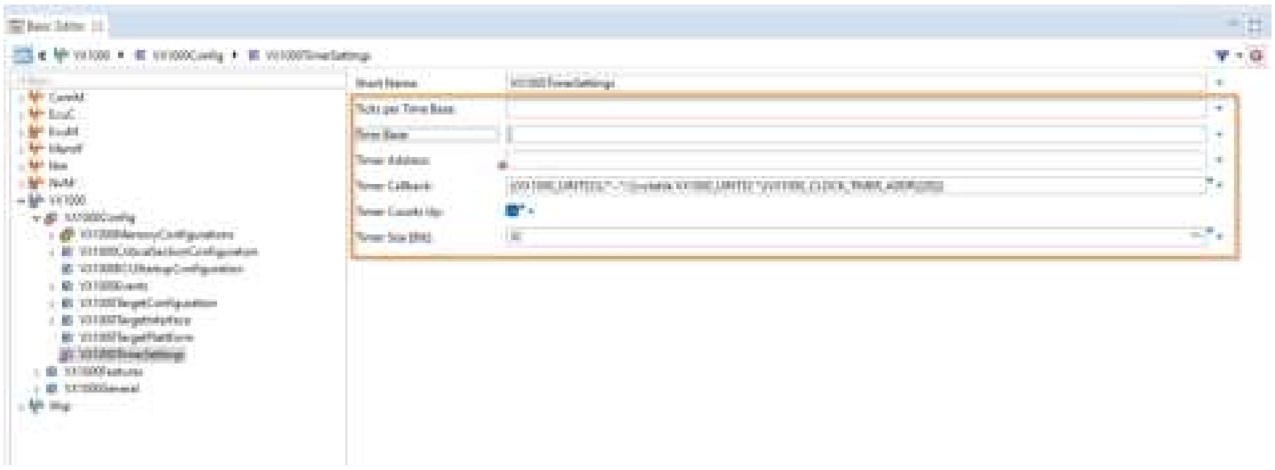
Basic Editor | Project Settings

VX1000 ▸ VX1000Config ▸ VX1000TargetConfiguration

Short Name	VX1000TargetConfiguration
Current Core Index Callback	getCurrentCore()
Running on Master Callback	getRunningOnMainCore()
Target	[illegible]

5.4 Timer configuration

Each time a measurement event is triggered, the VX1000 Application Driver captures a timestamp.
The timestamp source must be explicitly configured:



Short Name	VX1000TimerSettings
Ticks per Time Base	
Time Base	
Timer Address	
Timer Callback	[illegible]
Timer Counts Up	
Timer Size [Bit]	[illegible]

Time Base + Ticks per Time Base

For the tool to correctly interpret the timestamps, the timestamp resolution must be declared. For this, the characteristics of the timer counter must be specified as a time base in decades of nanoseconds and the ticks per time base. The latter is the counter increment within the time base period.

Example:

Time Base: 1000

Ticks per Time Base: 21

Specifies a counter incrementing at a rate of 21 MHz (21 ticks per 1000 ns)

Timer Address:

The address of the counter register of the used timer.

Timer Callback:

This callback is executed whenever the VX1000 AppDriver needs to capture a timestamp. In most cases, the default configuration is sufficient and does not require modification.

Timer Counts Up:

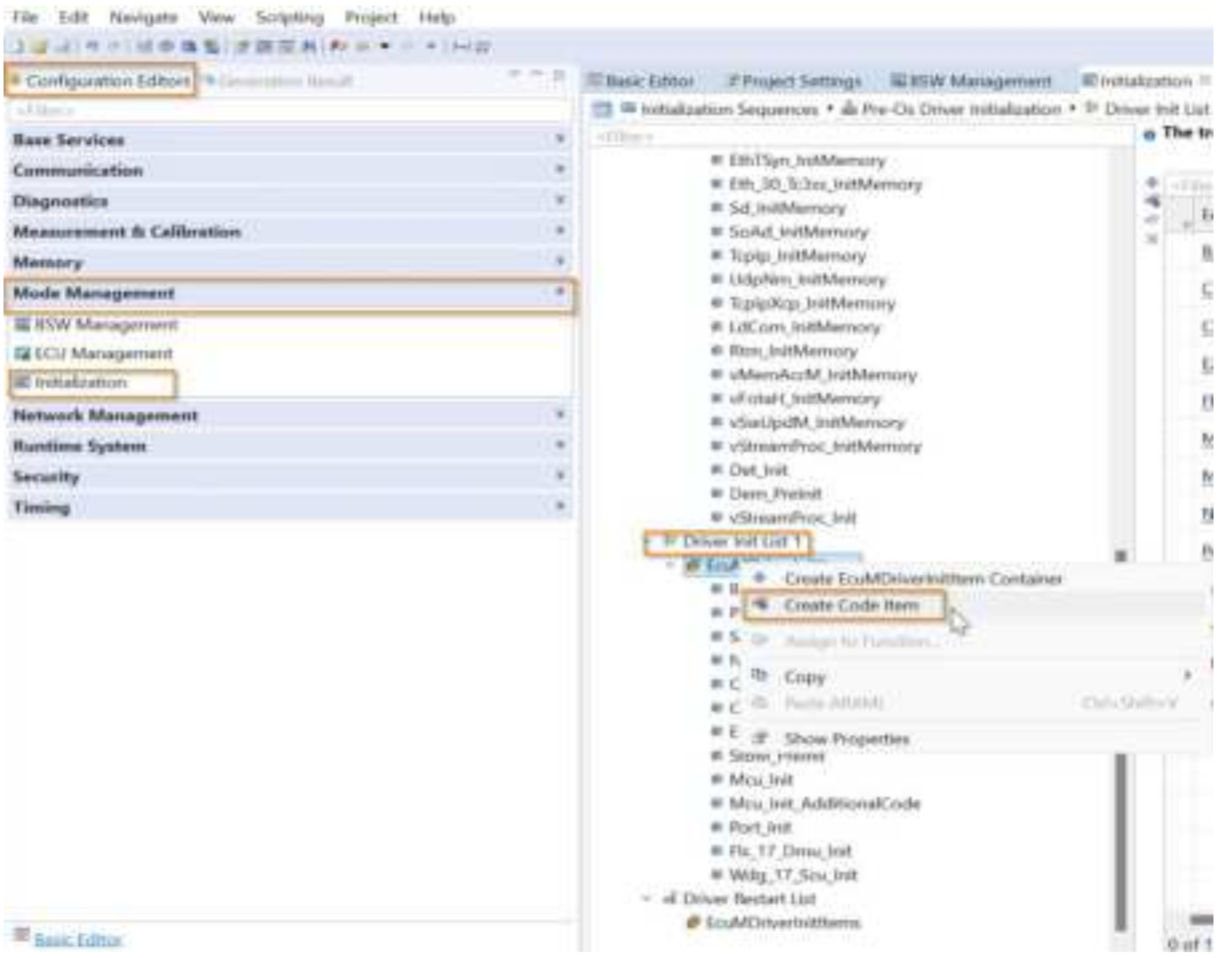
Specifies whether the counter is an up-counter or a down-counter. In most cases, up-counters are used.

Timer Size [Bit]:

Size in bytes of the timer counter which is used for timestamping. Supported timer counter widths are 16 bit, 24 bit or 32 bit. In most cases, 32 bit counters are used.

5.5 Startup configuration

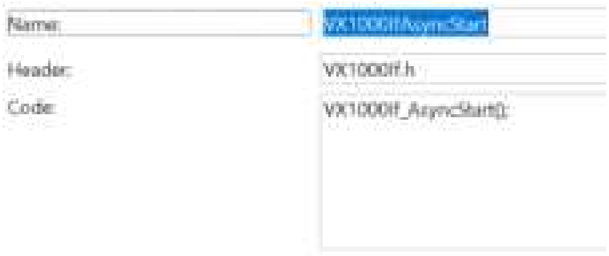
For the correct startup configuration, the Mode Management Initialization must be modified. Right click on "Driver Init List 1" and select "Create Code Item" for the initialization functions in the next sections.



5.5.1 Startup configuration VX1000If_InitAsyncStart

/Initialization Sequences/Pre-Os Driver Initialization/Driver Init List 1/EcuMDriverInitItems /VX1000If_InitAsyncStart

Select "Create Code Item" for the function VX1000If_InitAsyncStart()



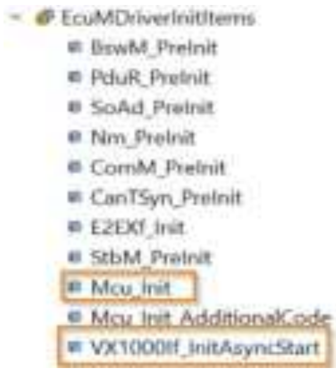
Name:	VX1000IfAsyncStart
Header:	VX1000If.h
Code:	VX1000If_AsyncStart();

Name: VX1000If_InitAsyncStart

Header: VX1000If.h

Code: VX1000If_InitAsyncStart();

Move the code after the Mcu_Init init steps

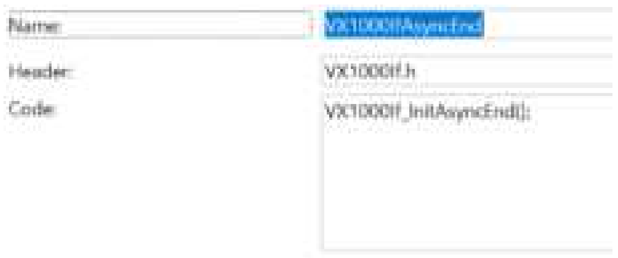


EcuMDriverInitItems
- BswM_PreInit
- PduR_PreInit
- SoAd_PreInit
- Nm_PreInit
- ComM_PreInit
- CanTSyn_PreInit
- E2EXf_Init
- StbM_PreInit
- Mcu_Init
- Mcu_Init_AdditionalCode
- VX1000If_InitAsyncStart

5.5.2 Startup configuration VX1000If_InitAsyncEnd

/Initialization Sequences/Pre-Os Driver Initialization/Driver Init List 1/EcuMDriverInitItems /VX1000If_InitAsyncEnd

Select “Create Code Item” for the function VX1000If_InitAsyncEnd()



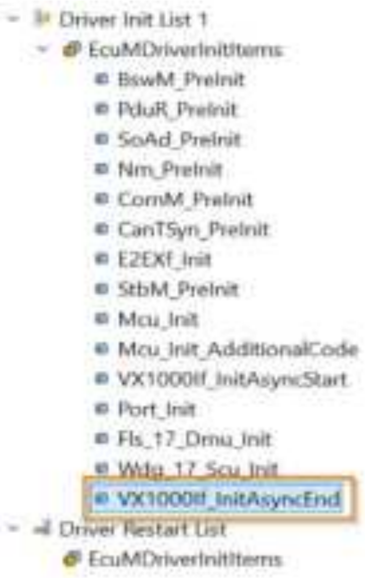
Name:	VX1000IfAsyncEnd
Header:	VX1000If.h
Code:	VX1000If_InitAsyncEnd();

Name: VX1000If_InitAsyncEnd

Header: VX1000If.h

Code: VX1000If_InitAsyncEnd();

Ensure that this function is called as late as possible, but definitely before the first measurement event is triggered.



- Driver Init List 1
  - EcuMDriverInitItems
    - BswM_PreInit
    - PduR_PreInit
    - SoAd_PreInit
    - Nm_PreInit
    - ComM_PreInit
    - CanTSyn_PreInit
    - E2EXf_Init
    - StbM_PreInit
    - Mcu_Init
    - Mcu_Init_AdditionalCode
    - VX1000If_InitAsyncStart
    - Port_Init
    - Fls_17_Dmu_Init
    - Wdg_17_Scu_Init
    - VX1000If_InitAsyncEnd
- Driver Restart List
  - EcuMDriverInitItems

5.6 Memory allocation

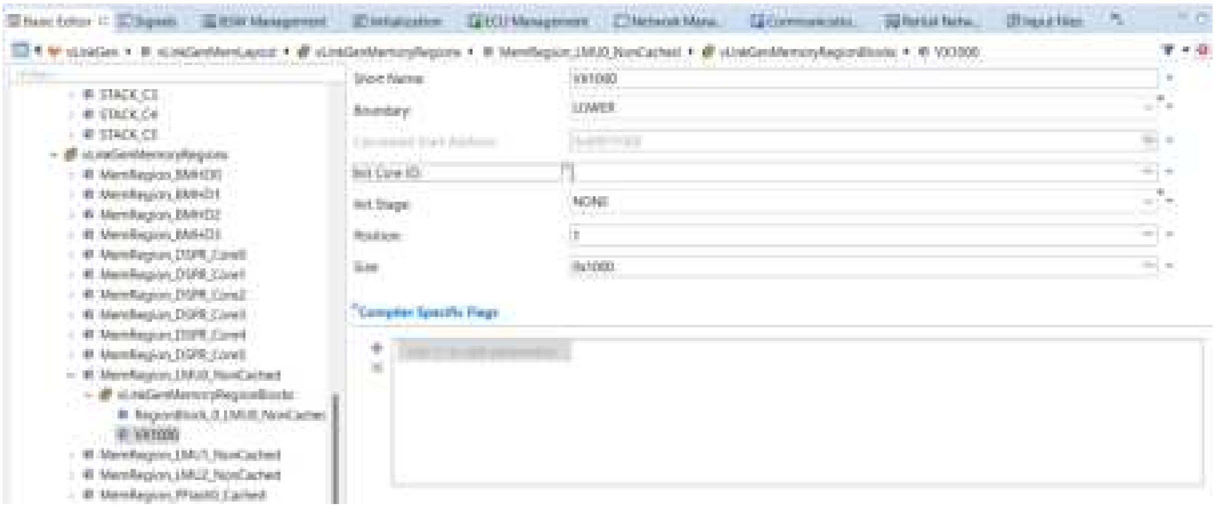
The VX1000 communicates with the VX1000 Application Driver via a shared memory structure. Adjusting the address of this communication structure in the VX1000 configuration is error-prone when switching between different ECU Software Versions. By pinning the structure to a fixed address, the workflow becomes both simpler and faster. To do so, the vLinkGen module can be used to put the variable into a linker section at a fixed address.

5.6.1 Configuration of vLinkGen

5.6.1.1 vLinkGenMemoryRegion

Create a VX1000 container in the appropriate vLinkGenMemoryRegions like the LMU0 memory. The memory must be uncached. Its required size depends on the expected extent of the DAQ configurations. The parameter /VX1000/VX1000Config/VX1000MemoryConfiguration/VX1000OldaMemory/VX1000_OLDA_MEMORY_SIZE should be taken into consideration.

Treepath: /vLinkGen/vLinkGenMemLayout/vLinkGenMemoryRegions

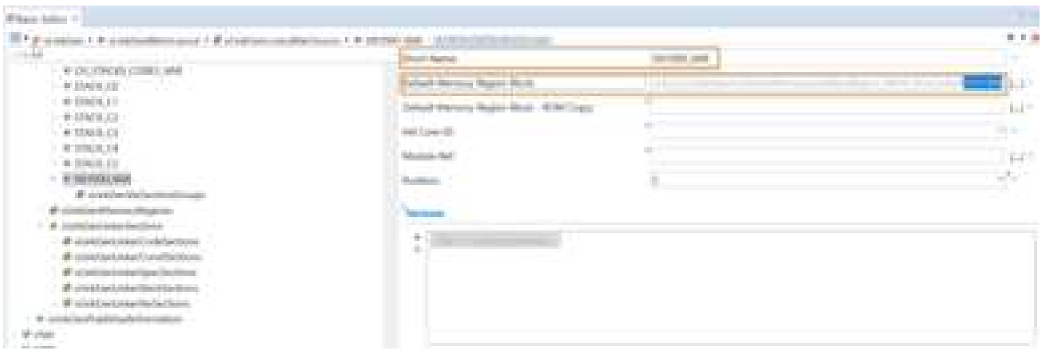


Short Name	VX1000
Boundary	LOWER
Init Core ID	
Init Stage	NONE
Position	1
Size	0x1000

5.6.1.2 Create vLinkGenLogicalVarGroups Container VX1000_VAR

Create a VX1000_VAR container in vLinkGenLogicalVarGroups container with reference “Default Memory Region Block” to the container from step 1.

Treepath /vLinkGen/vLinkGenMemLayout/vLinkGenLogicalVarGroups/VX1000_VAR

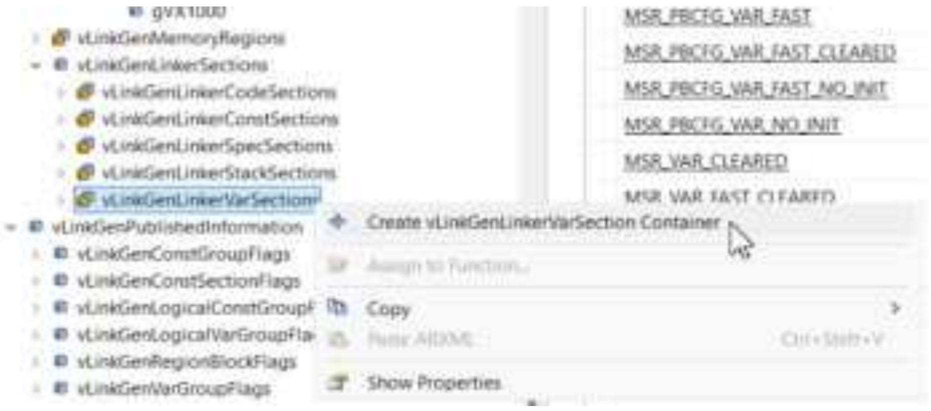


5.6.1.3 Create vLinkGenLinkerVarSection

For assigning the VX1000 Variables to a memory section, vLinkGenLinkerVarSections must be created:

/vLinkGen/vLinkGenMemLayout/vLinkGenLinkerSections/vLinkGenLinkerVarSections

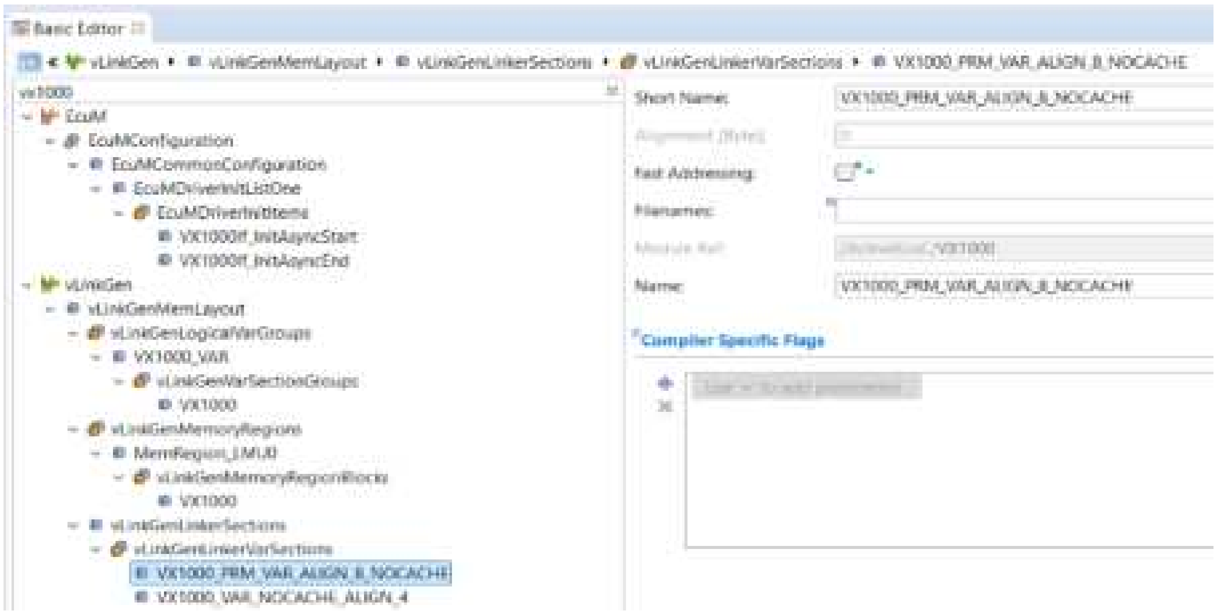
Right click on vLinkGenLinkerVarSections and press “Create vLinkGenLinkerVarSection Container” for each for the following Linker Sections


gVX1000
vLinkGenMemoryRegions
vLinkGenLinkerSections
vLinkGenLinkerCodeSections
vLinkGenLinkerConstSections
vLinkGenLinkerSpecSections
vLinkGenLinkerStackSections
vLinkGenLinkerVarSections
vLinkGenPublishedInformation
vLinkGenConstGroupFlags
vLinkGenConstSectionFlags
vLinkGenLogicalConstGroupF
vLinkGenLogicalVarGroupFla
vLinkGenRegionBlockFlags
vLinkGenVarGroupFlags
MSR_PBCFG_VAR_FAST
MSR_PBCFG_VAR_FAST_CLEARED
MSR_PBCFG_VAR_FAST_NO_INIT
MSR_PBCFG_VAR_NO_INIT
MSR_VAR_CLEARED
MSR_VAR_FAST_CLEARED
Create vLinkGenLinkerVarSection Container
Assign to Function...
Copy
Paste AR XML
Ctrl+Shift+V
Show Properties

5.6.1.3.1 Create vLinkGenLinkerVarSection VX1000_PRM_VAR_ALIGN_8_NOCACHE

Create a new vLinkGenLinkerVarSection with name VX1000_PRM_VAR_ALIGN_8_NOCACHE. Select VX1000_PRM_VAR_ALIGN_8_NOCACHE as Name and Short Name

/vLinkGen/vLinkGenMemLayout/vLinkGenLinkerSections/vLinkGenLinkerVarSections/
VX1000_PRM_VAR_ALIGN_8_NOCACHE


Basic Editor

vLinkGen ▸ vLinkGenMemLayout ▸ vLinkGenLinkerSections ▸ vLinkGenLinkerVarSections ▸ VX1000_PRM_VAR_ALIGN_8_NOCACHE

- vx1000
  - EcuM
    - EcuMConfiguration
      - EcuMCommonConfiguration
        - EcuMDriverInitListOne
          - EcuMDriverInitItems
            - VX1000If_InitAsyncStart
            - VX1000If_InitAsyncEnd
  - vLinkGen
    - vLinkGenMemLayout
      - vLinkGenLogicalVarGroups
        - VX1000_VAR
          - vLinkGenVarSectionGroups
            - VX1000
      - vLinkGenMemoryRegions
        - MemRegion_LMU0
          - vLinkGenMemoryRegionBlocks
            - VX1000
      - vLinkGenLinkerSections
        - vLinkGenLinkerVarSections
          - VX1000_PRM_VAR_ALIGN_8_NOCACHE
          - VX1000_VAR_NOCACHE_ALIGN_4

Short Name: VX1000_PRM_VAR_ALIGN_8_NOCACHE
Alignment (Byte): 8
Fast Addressing:
Filenames:
Module Ref: /ActiveEcuC/vX1000
Name: VX1000_PRM_VAR_ALIGN_8_NOCACHE

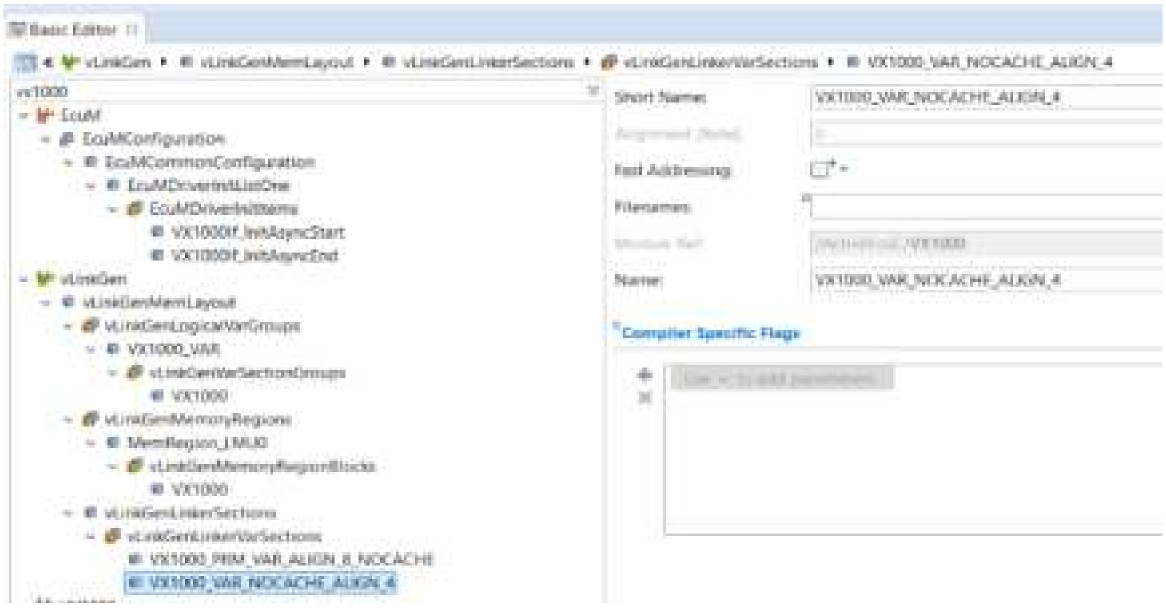
Compiler Specific Flags
Use + to add parameters

5.6.1.3.2 Create vLinkGenLinkerVarSection VX1000_VAR_NOCACHE_ALIGN_4

Create a vLinkGenLinkerVarSection VX1000_VAR_NOCACHE_ALIGN_4.

/vLinkGen/vLinkGenMemLayout/vLinkGenLinkerSections/vLinkGenLinkerVarSections/
VX1000_VAR_NOCACHE_ALIGN_4

Create a new vLinkGenLinkerVarSection with name VX1000_VAR_NOCACHE_ALIGN_4. Select VX1000_VAR_NOCACHE_ALIGN_4 as name and Short Name.


Basic Editor

vLinkGen ▸ vLinkGenMemLayout ▸ vLinkGenLinkerSections ▸ vLinkGenLinkerVarSections ▸ VX1000_VAR_NOCACHE_ALIGN_4

- vx1000
  - EcuM
    - EcuMConfiguration
      - EcuMCommonConfiguration
        - EcuMDriverInitListOne
          - EcuMDriverInitItems
            - VX1000If_InitAsyncStart
            - VX1000If_InitAsyncEnd
  - vLinkGen
    - vLinkGenMemLayout
      - vLinkGenLogicalVarGroups
        - VX1000_VAR
          - vLinkGenVarSectionGroups
            - VX1000
      - vLinkGenMemoryRegions
        - MemRegion_LMU0
          - vLinkGenMemoryRegionBlocks
            - VX1000
      - vLinkGenLinkerSections
        - vLinkGenLinkerVarSections
          - VX1000_PRM_VAR_ALIGN_8_NOCACHE
          - VX1000_VAR_NOCACHE_ALIGN_4

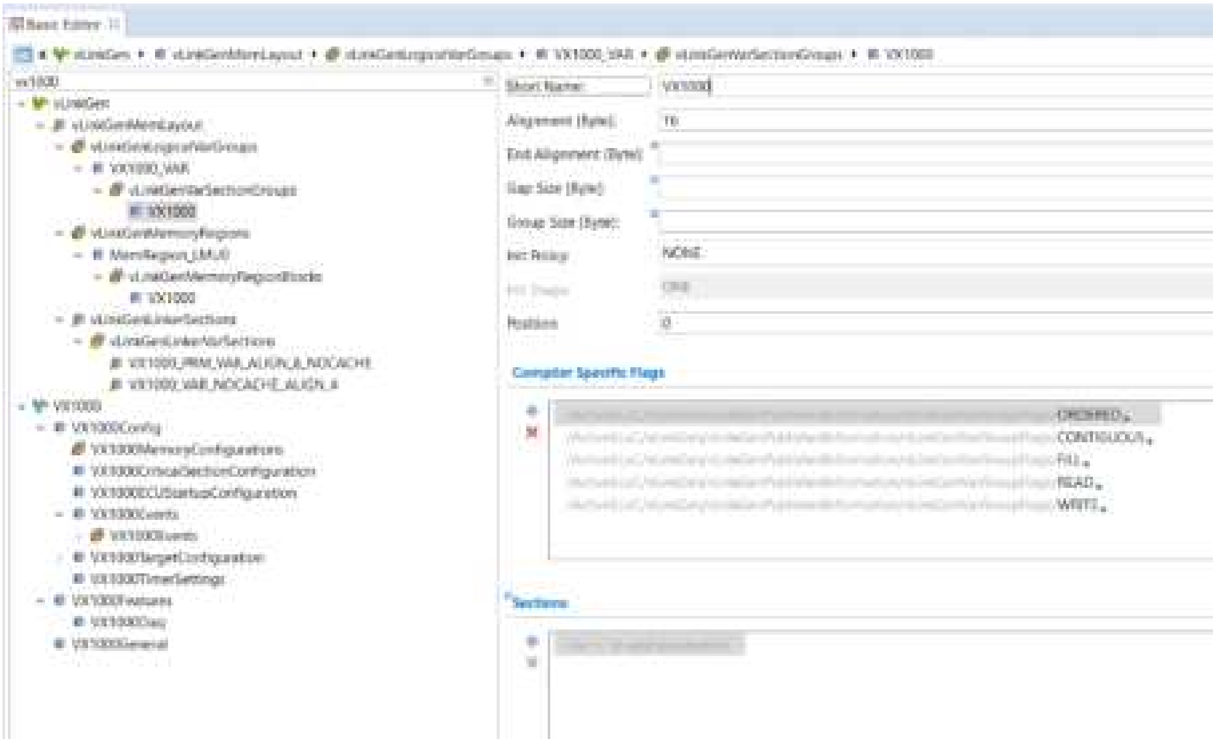
Short Name: VX1000_VAR_NOCACHE_ALIGN_4
Alignment (Byte): 0
Fast Addressing:
Filenames:
Module Ref: /ActiveEcuC/vX1000
Name: VX1000_VAR_NOCACHE_ALIGN_4

Compiler Specific Flags
Use + to add parameters

5.6.1.4 Create vLinkGenVarSectionGroup Container VX1000

Create a vLinkGenVarSectionGroup Container

/vLinkGen/vLinkGenMemLayout/vLinkGenLogicalVarGroups/VX1000_VAR/vLinkGenVarSectionGroups
/VX1000



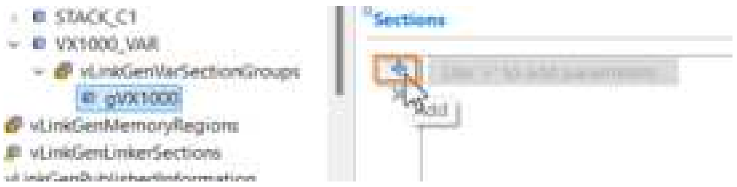
Short Name: VX1000
Alignment (Byte): 16
End Alignment (Byte):
Gap Size (Byte):
Group Size (Byte):
Init Policy: NONE
Position: 0

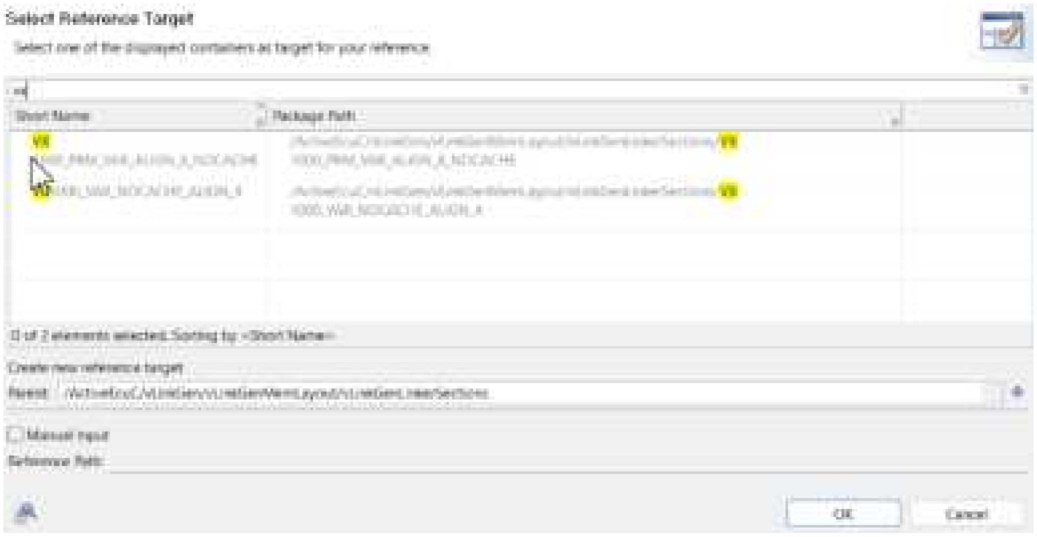
Compiler Specific Flags: ORDERED, CONTIGUOUS, FILL, READ, WRITE

Add the Linker Section Groups

VX1000_PRM_VAR_ALIGN_8_NOCACHE,

VX1000_VAR_NOCACHE_ALIGN_4,




Select Reference Target

Select one of the displayed containers as target for your reference

vx

Short Name	Package Path
VX1000_PRM_VAR_ALIGN_8_NOCACHE	/ActiveEcuC/vLinkGen/vLinkGenMemLayout/vLinkGenLinkerSections/VX1000_PRM_VAR_ALIGN_8_NOCACHE
VX1000_VAR_NOCACHE_ALIGN_4	/ActiveEcuC/vLinkGen/vLinkGenMemLayout/vLinkGenLinkerSections/VX1000_VAR_NOCACHE_ALIGN_4

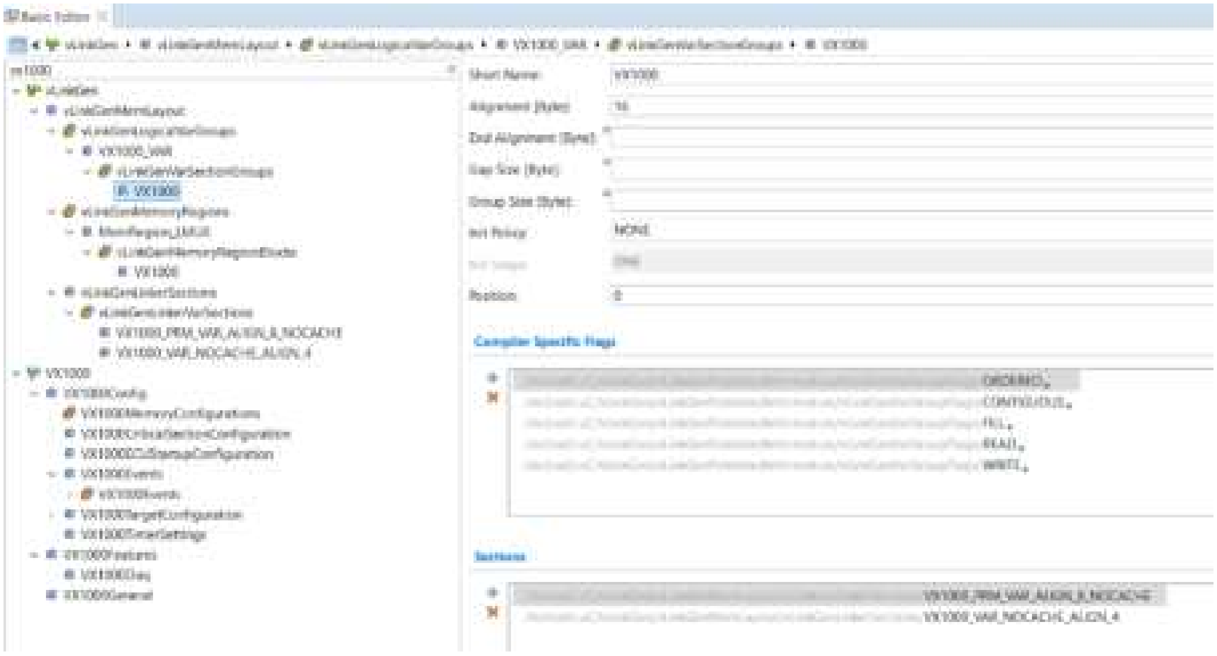
0 of 2 elements selected. Sorting by: <Short Name>

Create new reference target

Parent: /ActiveEcuC/vLinkGen/vLinkGenMemLayout/vLinkGenLinkerSections

Manual input

Reference Path:



Short Name: VX1000
Alignment (Byte): 16
Dst Alignment (Byte):
Gap Size (Byte):
Group Size (Byte):
Init Policy: NONE
Init Stage: ONE
Position: 0

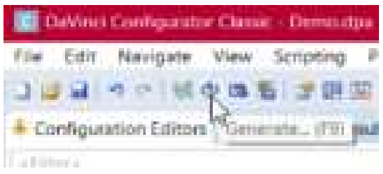
Compiler Specific Flags

ORDERED,
CONTIGUOUS,
FILL,
READ,
WRITE,

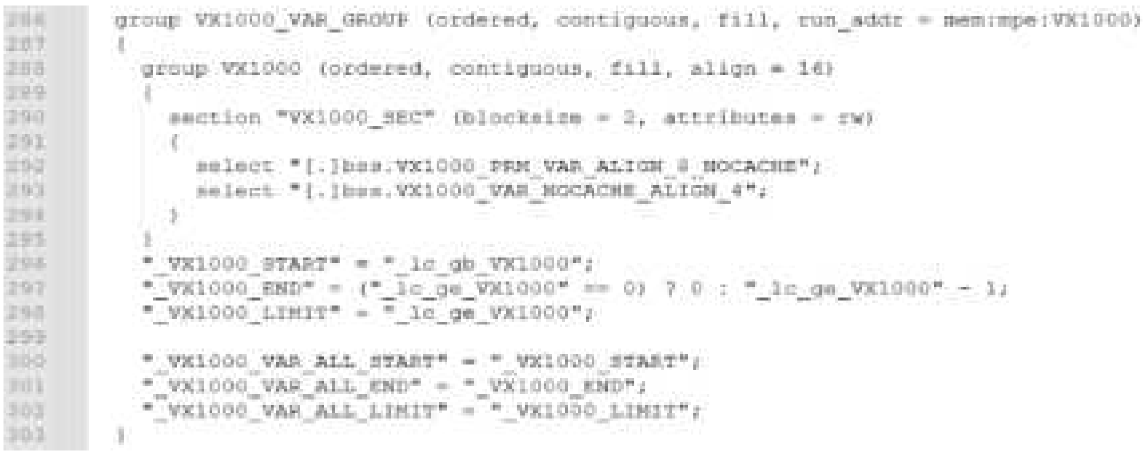
Sections

VX1000_PRM_VAR_ALIGN_8_NOCACHE
VX1000_VAR_NOCACHE_ALIGN_4

After you have generated your project …



the compiler/link should look like this (.\Appl\Source\ vLinkGen_Template.lsl)



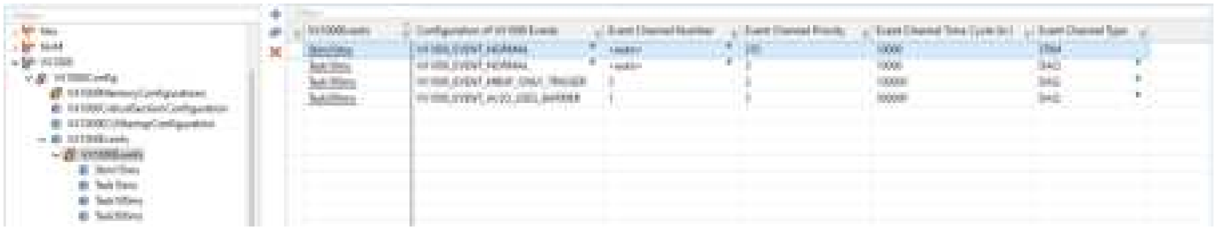
```
group VX1000_VAR_GROUP (ordered, contiguous, fill, run_addr = mem:mpe:VX1000)
{
  group VX1000 (ordered, contiguous, fill, align = 16)
  {
    section "VX1000_SEC" (blocksize = 2, attributes = rw)
    {
      select "[.]bss.VX1000_PRM_VAR_ALIGN_8_NOCACHE";
      select "[.]bss.VX1000_VAR_NOCACHE_ALIGN_4";
    }
  }
  "_VX1000_START" = "_lc_gb_VX1000";
  "_VX1000_END" = ("_lc_ge_VX1000" == 0) ? 0 : "_lc_ge_VX1000" - 1;
  "_VX1000_LIMIT" = "_lc_ge_VX1000";

  "_VX1000_VAR_ALL_START" = "_VX1000_START";
  "_VX1000_VAR_ALL_END" = "_VX1000_END";
  "_VX1000_VAR_ALL_LIMIT" = "_VX1000_LIMIT";
}
```

After compiling the application with b.bat in the .\Appl directory, the gVX1000 structure should be put in the memory region defined in 5.6.1.1 Overview VX1000 DaVinci Integration. You can double check this in the map file.

```
| 0xb0041000 | _lc_gb_VX1000_VAR_GROUP
| 0xb0041000 | gVX1000
| 0xb004150c | vx1000_DetectStartTime
```

5.7 DAQ Event configuration

The VX1000 provides the possibility to configure DAQ events and to set various attributes of these events. A detailed configuration is only needed for special features like Multibuffer OLDA or In-Place OLDA. For plain OLDA or Data Trace-based measurement, nothing must be configured and this chapter can be skipped.



VX1000Events	Configuration of VX1000 Events	Event Channel Number	Event Channel Priority	Event Channel Time Cycle [us]	Event Channel Type
[illegible]	VX1000_EVENT_NORMAL	<auto>	[illegible]	[illegible]	STIM
[illegible]	VX1000_EVENT_NORMAL	<auto>	[illegible]	[illegible]	DAQ
[illegible]	VX1000_EVENT_MBUF_ONLY_TRIGGER	[illegible]	[illegible]	[illegible]	DAQ
[illegible]	VX1000_EVENT_ALSO_USES_BARRIER	[illegible]	[illegible]	[illegible]	DAQ

Configuration of VX1000 DAQ Events:

This is the measurement type and how the event is instrumented within the ECU software.

For In-Place OLDA Events, besides triggering the event, the VX1000If_EventProcessingBarrier must be called in a defined sequence. To indicate this special type of code instrumentation, such events need to be configured as VX1000_EVENT_ALSO_USES_BARRIER.

Events that have a very short cycle time < 1ms should be marked as VX1000_EVENT_MBUF_ONLY_TRIGGER.

This annotation given by the ECU software is just a hint for the VX1000. In the VX1000 there are also configuration options to override these settings.

Event Channel Number:

This is the XCP DAQ event channel number that must be used by the measurement tool for this event channel. For <auto>, the numbers are calculated on demand and can be accessed via VX1000_EVTCH_$(Eventname) (with Eventname in uppercase letters).

Event Channel Priority:

The EVENT_CHANNEL_PRIORITY defines the processing order of this event channel by the VX1000 when multiple channels are triggered at the same time.

Event Channel Time Cycle [us]:

The EVENT_CHANNEL_TIME_CYCLE indicates the period for cyclic events, specifying how frequently this event channel is triggered. This is useful information for the VX1000 and for the measurement tool. A non-cyclic task is indicated by 0.

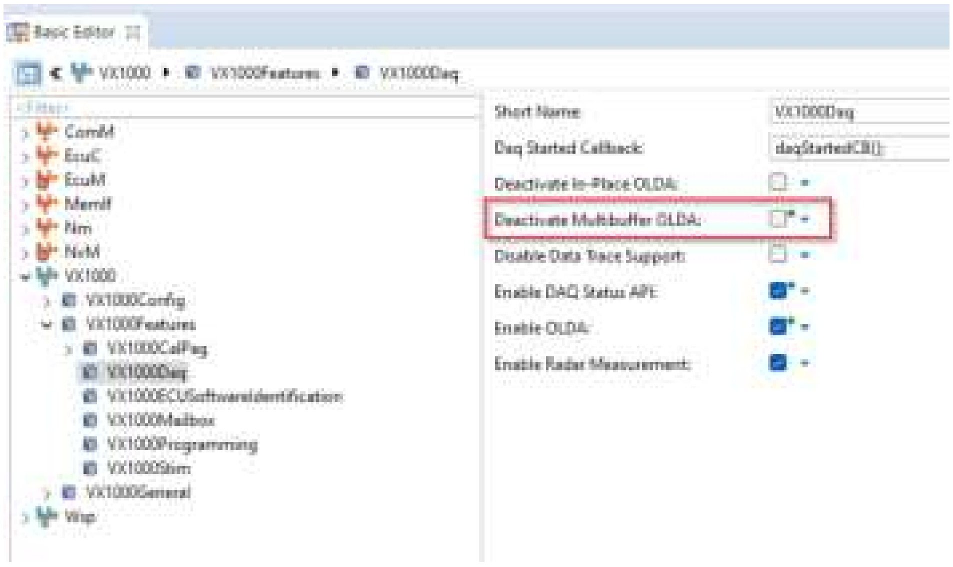
Event Channel Type:

Describes the type of the event channel. DAQ is for Synchronous Data Acquisition and is used for measuring ECU data. STIM is for Synchronous Data Stimulation and is used for sending data to the ECU like for functional bypassing use cases. DAQ_STIM events can be used for either DAQ or STIM.

5.8 Feature configuration

5.8.1 Short cycle times

Projects that must acquire data at very short cycle times of <1ms should ensure that Multibuffer OLDA is activated. Also ensure that the fast event channels are correctly configured (see 5.7 DAQ Event configuration).

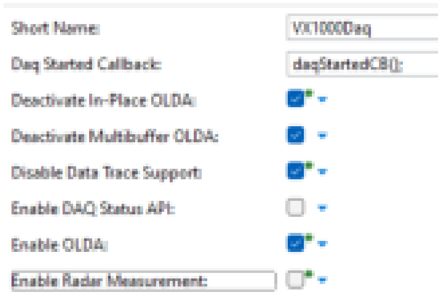


Short Name:	VX1000Daq
Daq Started Callback:	daqStartedCB();
Deactivate In-Place OLDA:	☐
Deactivate Multibuffer OLDA:	☐
Disable Data Trace Support:	☐
Enable DAQ Status API:	✓
Enable OLDA:	✓
Enable Radar Measurement:	✓

5.8.2 Minimum system

If running on a device with very limited hardware resources, the VX1000 Application Driver can be configured to have a minimal RAM and runtime footprint. Start with an empty configuration, then …

- Check: Deactivate In-Place OLDA
- Check: Deactivate Multibuffer OLDA
- Check: Disable Data Trace Support
- Uncheck: Enable DAQ Status API
- Check: Enable OLDA
- Uncheck: Enable Radar Measurement

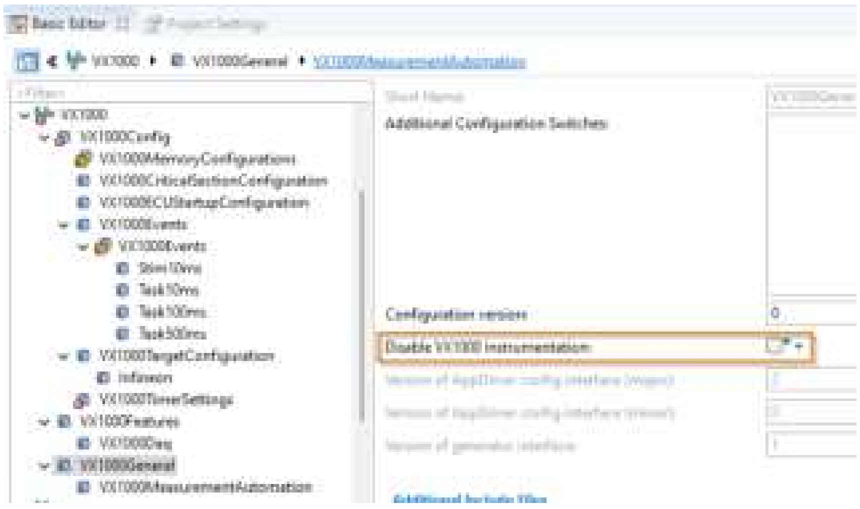


Short Name:	VX1000Daq
Daq Started Callback:	daqStartedCB();
Deactivate In-Place OLDA:	✓
Deactivate Multibuffer OLDA:	✓
Disable Data Trace Support:	✓
Enable DAQ Status API:	☐
Enable OLDA:	✓
Enable Radar Measurement:	☐

5.8.3 Disable VX1000 Application Driver

The VX1000 Application Driver provides configuration switches to remove nearly the complete driver from code. Afterwards, the VX1000 measurement features are not available anymore. The driver can be easily reactivated later with all settings preserved.

Parameter: /ActiveEcuC/VX1000/VX1000General[VX1000_DISABLE_INSTRUMENTATION]



Short Name

Additional Configuration Switches

Configuration version 0

Disable VX1000 Instrumentation

Version of generator interface 1

6 VX1000If generation

Please make sure to carefully read the VX1000If Technical Documentation and ensure that VX1000If_IsVX1000DriverAccessEnabled is correctly defined.

7 Additional Resources

TechnicalReference_VX1000.pdf: Technical Reference for the VX1000 Application Driver

TechnicalReference_VX1000If.pdf: Technical Reference for the VX1000 Interface

8 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.