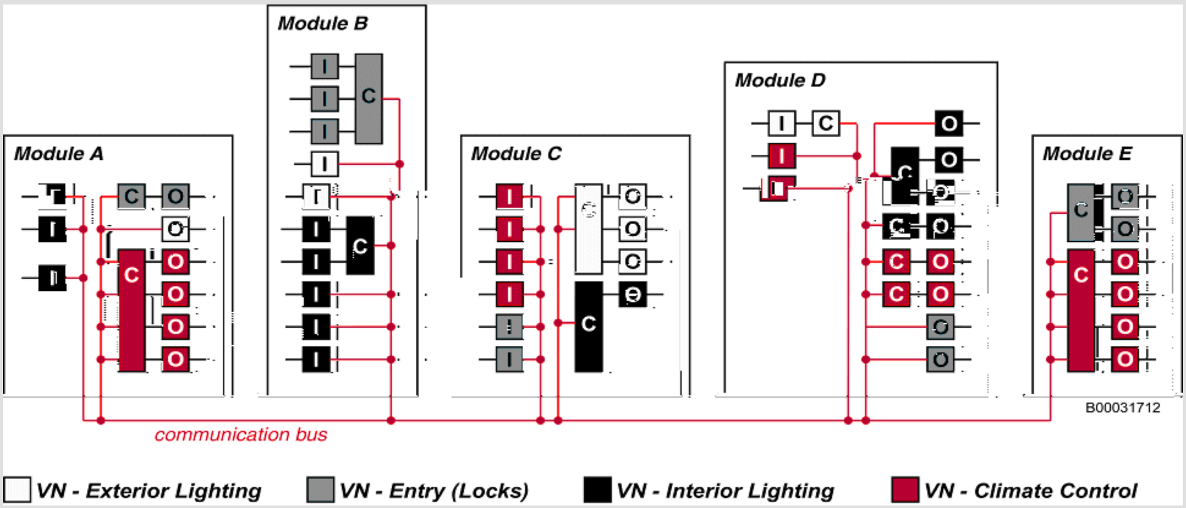
Startup with GMLAN and GENy



Authors:

Version:

Status:

History

Author	Date	Version	Remarks

Motivation For This Work

WARNING

All application code in any of the Vector User Manuals are for training purposes only. They are slightly tested and designed to understand the basic idea of using a certain component or a set of components.

Contents

Illustrations

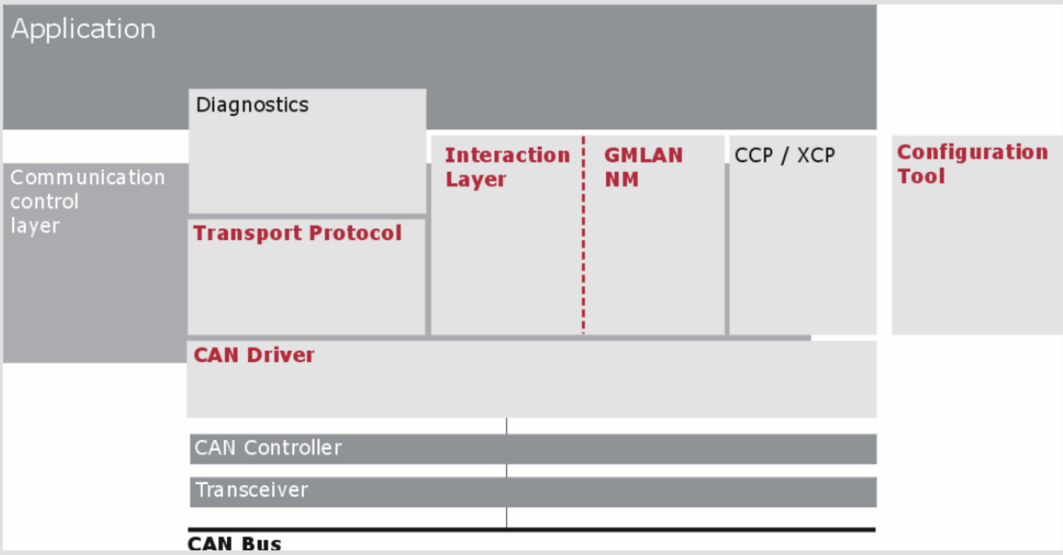
1 About this Document

1.1 Legend and Explanation of Symbols

Symbol	Meaning
▲■●	
▨	
👣	
⚠	
i	
✎	
🚫	
??	

2 What is GMLAN

GMLAN is CANbedded for GM



red bold: Standard GMLAN component other components: optional

Mandatory components of GMLAN

CAN Driver

Interaction Layer

Transport Protocol

GMLAN Network Management

Configuration Tool

Info

Additional components, optional available:

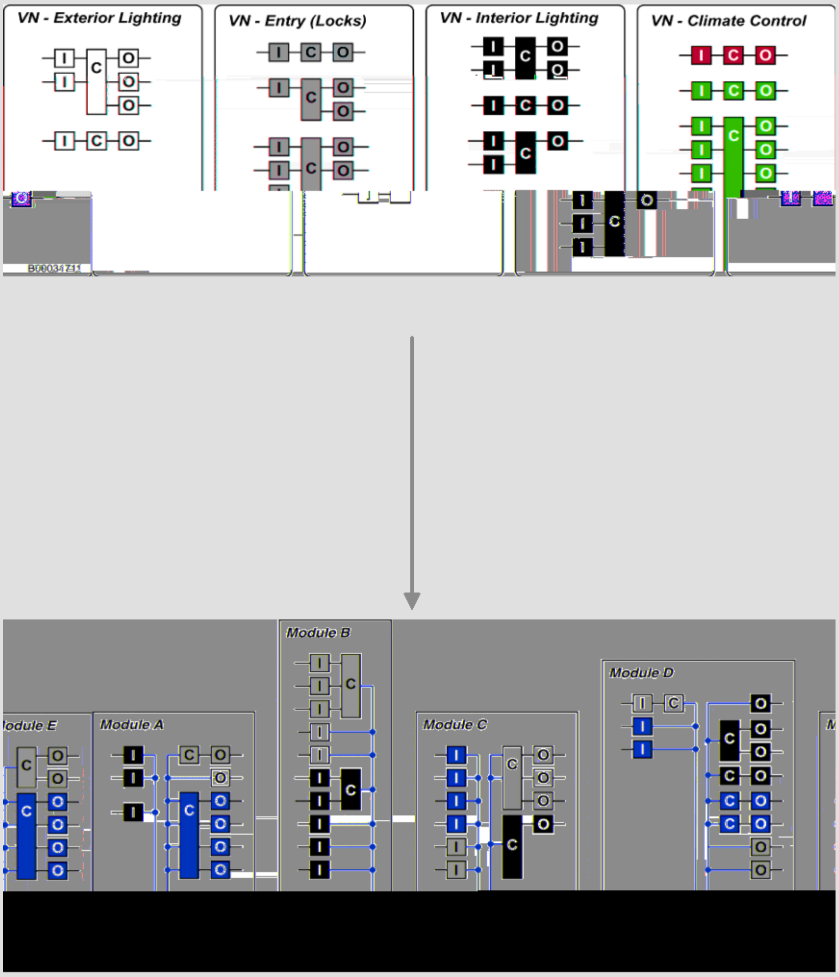
Diagnostics - CANdesc

HIGHLY RECOMMENDED by GM!

Communication Control Layer CCL

Measurement and Calibration (CCP/XCP)

2.1 Virtual Networks



What comes along with this concept?

3 Your ECU

❶ Body Bus

❷ Infotainment

❸ Powertrain

3.1 Body Bus ECU

Single Wire CAN Mixed Identifiers (11bit and 29bit)

■

■

■

■

■

■

Single Wire CAN 11 Bit Identifier

■

■

■

Multiple ECU on Single Wire CAN

■

■

■

Send message

3.2 Infotainment ECU

-
-
-
-
-

3.3 PowerTrain ECU (or Chassis expansion or Powertrain expansion)

-
-
-
-
-
-

4 GMLAN In 6 Steps

STEP 1 : PREPARE YOUR SOFTWARE PROJECT

STEP 2: CONFIGURATION TOOL AND DATA BASE FILE

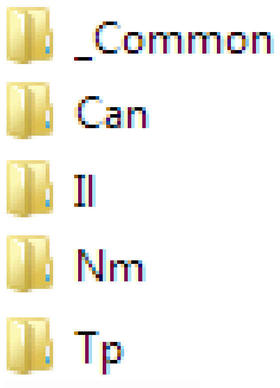
STEP 3: ADD FILES TO YOUR APPLICATION

STEP 4: ADAPTATIONS FOR YOUR APPLICATION

STEP 5: COMPILE AND LINK

STEP 6: TEST THE SOFTWARE COMPONENT

4.1 STEP 1 Prepare Your Software Project


_Common
Can
Il
Nm
Tp

Info

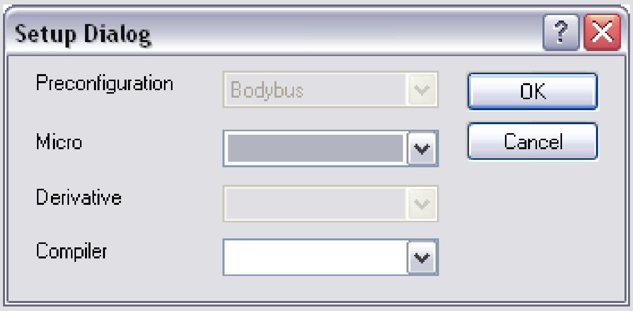
GenData

gendata

4.2 STEP 2 Configuration Tool GENy and DBC File

Info

Setup Dialog


Setup Dialog

Preconfiguration: Bodybus

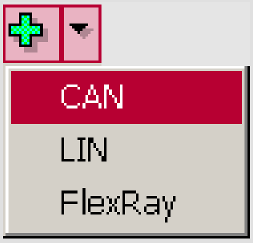
Micro:

Derivative:

Compiler:

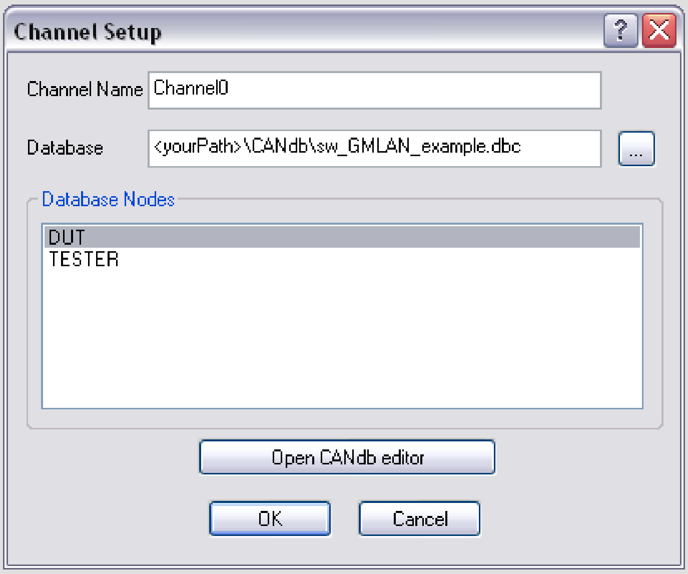
Preconfiguration

-
-
-



Info

[…]



Channel Setup

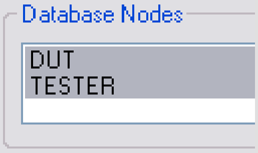
Channel Name: Channel0

Database: <yourPath>\CANdb\sw_GMLAN_example.dbc

Database Nodes:
- DUT
- TESTER

Caution for Multiple ECUs

[OK]

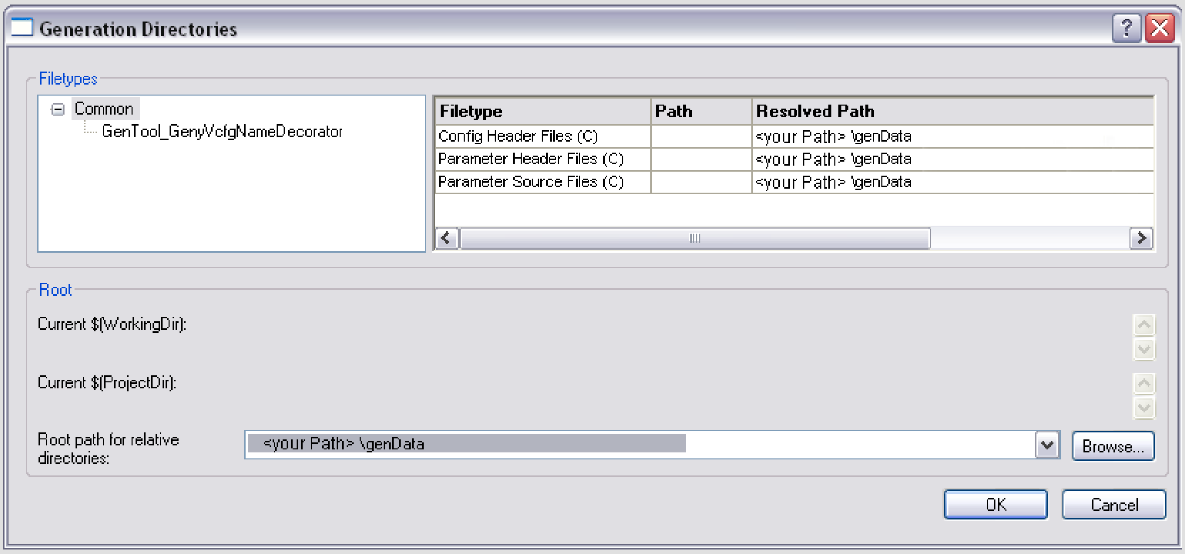


Database Nodes
- DUT
- TESTER

4.2.1 Setting of Generation Paths

Configuration|Generation Paths…

gendata



Generation Directories

Filetypes

⊟ Common
GenTool_GenyVcfgNameDecorator

Filetype	Path	Resolved Path
Config Header Files (C)		<your Path> \genData
Parameter Header Files (C)		<your Path> \genData
Parameter Source Files (C)		<your Path> \genData

Root

Current $(WorkingDir):

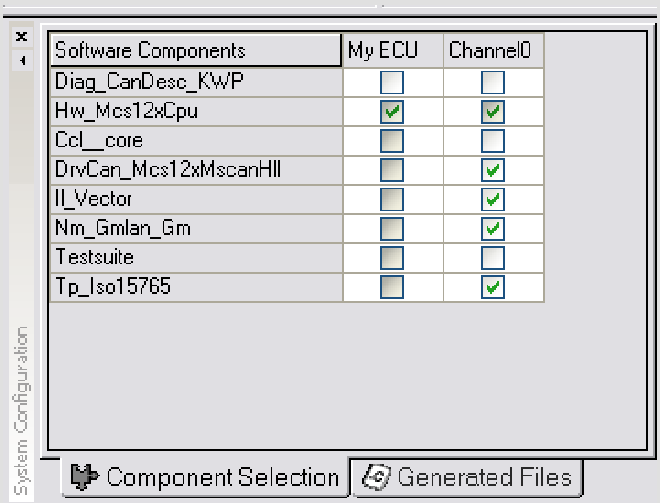
Current $(ProjectDir):

Root path for relative directories: <your Path> \genData

Root Path for relative directories

Path **Resolved Path**

4.2.2 Component Selection

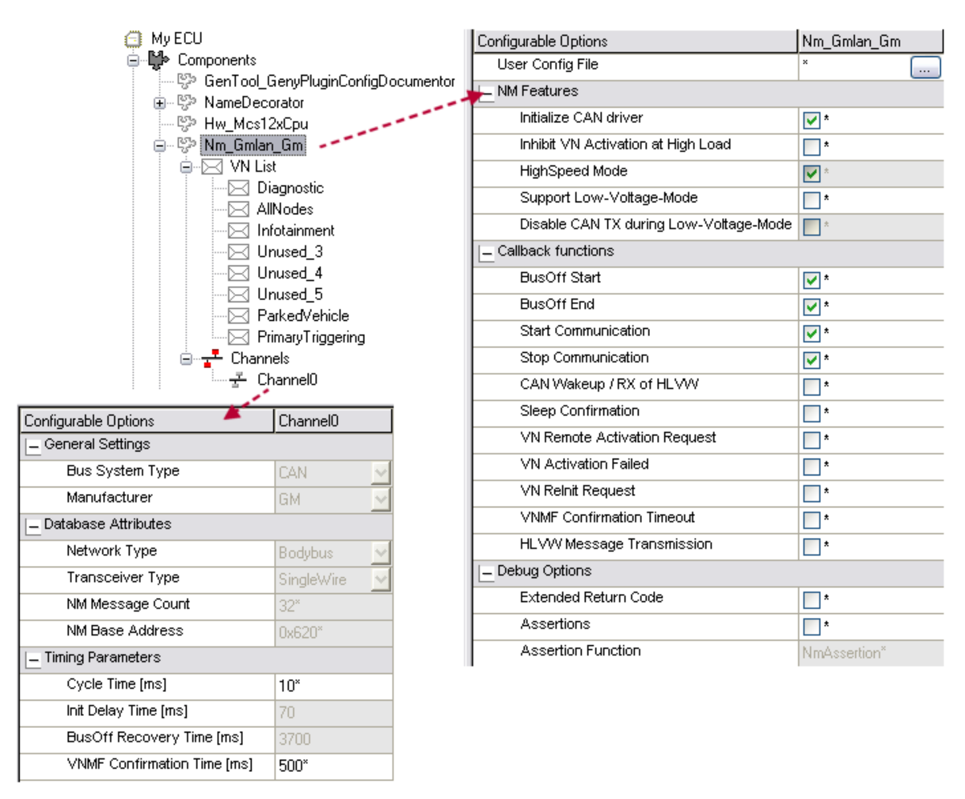


Software Components	My ECU	Channel0
Diag_CanDesc_KWP	□	□
Hw_Mcs12xCpu	☑	☑
Ccl__core	□	□
DrvCan_Mcs12xMscanHll	□	☑
Il_Vector	□	☑
Nm_Gmlan_Gm	□	☑
Testsuite	□	□
Tp_Iso15765	□	☑

4.2.3 Tree view – A List of all selected components

Tree View

MyECU|Components



Configurable Options	Channel0
− General Settings	
Bus System Type	CAN
Manufacturer	GM
− Database Attributes	
Network Type	Bodybus
Transceiver Type	SingleWire
NM Message Count	32*
NM Base Address	0x620*
− Timing Parameters	
Cycle Time [ms]	10*
Init Delay Time [ms]	70
BusOff Recovery Time [ms]	3700
VNMF Confirmation Time [ms]	500*

Configurable Options	Nm_Gmlan_Gm
User Config File	* ...
− NM Features	
Initialize CAN driver	☑*
Inhibit VN Activation at High Load	□*
HighSpeed Mode	☑*
Support Low-Voltage-Mode	□*
Disable CAN TX during Low-Voltage-Mode	□*
− Callback functions	
BusOff Start	☑*
BusOff End	☑*
Start Communication	☑*
Stop Communication	☑*
CAN Wakeup / RX of HLVW	□*
Sleep Confirmation	□*
VN Remote Activation Request	□*
VN Activation Failed	□*
VN ReInit Request	□*
VNMF Confirmation Timeout	□*
HLVW Message Transmission	□*
− Debug Options	
Extended Return Code	□*
Assertions	□*
Assertion Function	NmAssertion*

Info

NetworkType

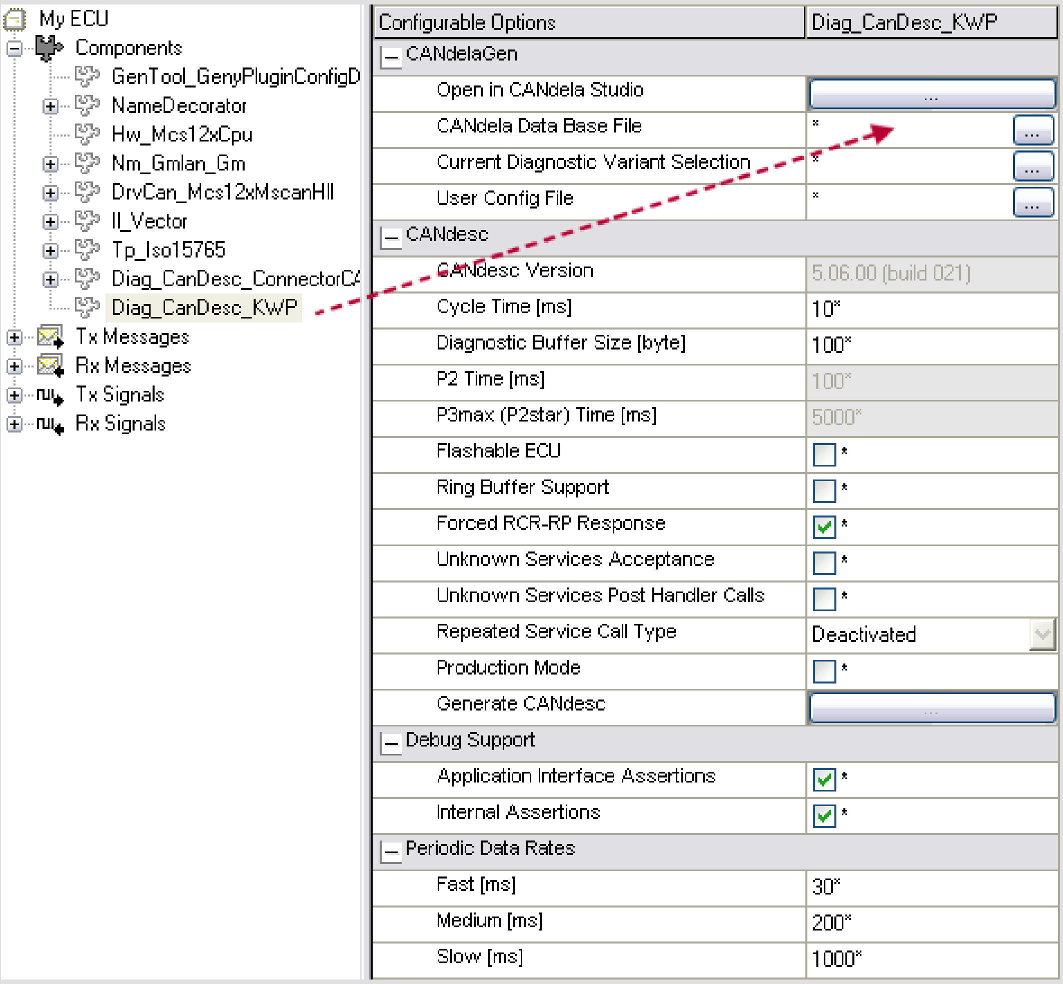
PowerTrain **BodyBus**

4.2.6 Tp_Iso15765

Info

4.2.7 Diagnostics – Diag_CanDesc_KWP

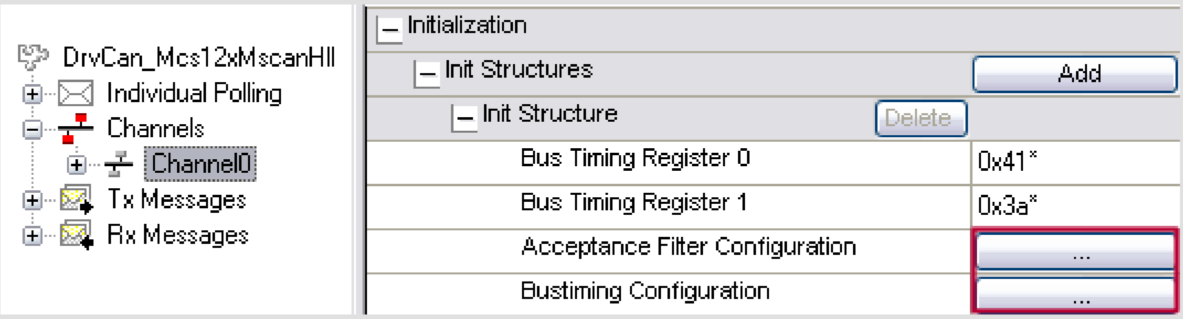
Component Selection



My ECU
- Components
  - GenTool_GenyPluginConfigD
  - NameDecorator
  - Hw_Mcs12xCpu
  - Nm_Gmlan_Gm
  - DrvCan_Mcs12xMscanHll
  - Il_Vector
  - Tp_Iso15765
  - Diag_CanDesc_ConnectorCA
  - Diag_CanDesc_KWP
- Tx Messages
- Rx Messages
- Tx Signals
- Rx Signals

Configurable Options	Diag_CanDesc_KWP
CANdelaGen	
Open in CANdela Studio	...
CANdela Data Base File	* ...
Current Diagnostic Variant Selection	* ...
User Config File	* ...
CANdesc	
CANdesc Version	5.06.00 (build 021)
Cycle Time [ms]	10*
Diagnostic Buffer Size [byte]	100*
P2 Time [ms]	100*
P3max (P2star) Time [ms]	5000*
Flashable ECU	□ *
Ring Buffer Support	□ *
Forced RCR-RP Response	☑ *
Unknown Services Acceptance	□ *
Unknown Services Post Handler Calls	□ *
Repeated Service Call Type	Deactivated
Production Mode	□ *
Generate CANdesc	...
Debug Support	
Application Interface Assertions	☑ *
Internal Assertions	☑ *
Periodic Data Rates	
Fast [ms]	30*
Medium [ms]	200*
Slow [ms]	1000*

4.2.8 DrvCan_<microcontroller>

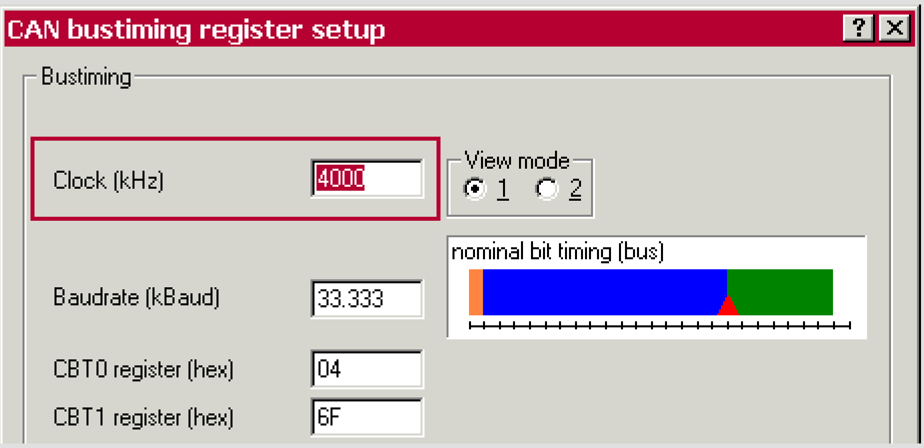


DrvCan_Mcs12xMscanHll
- Individual Polling
- Channels
  - Channel0
- Tx Messages
- Rx Messages

Initialization		
Init Structures		Add
Init Structure	Delete	
Bus Timing Register 0		0x41*
Bus Timing Register 1		0x3a*
Acceptance Filter Configuration		...
Bustiming Configuration		...

[…]

Bus timing registers



CAN bustiming register setup

Bustiming

Clock (kHz) 4000

View mode ● 1 ○ 2

nominal bit timing (bus)

Baudrate (kBaud) 33.333

CBT0 register (hex) 04

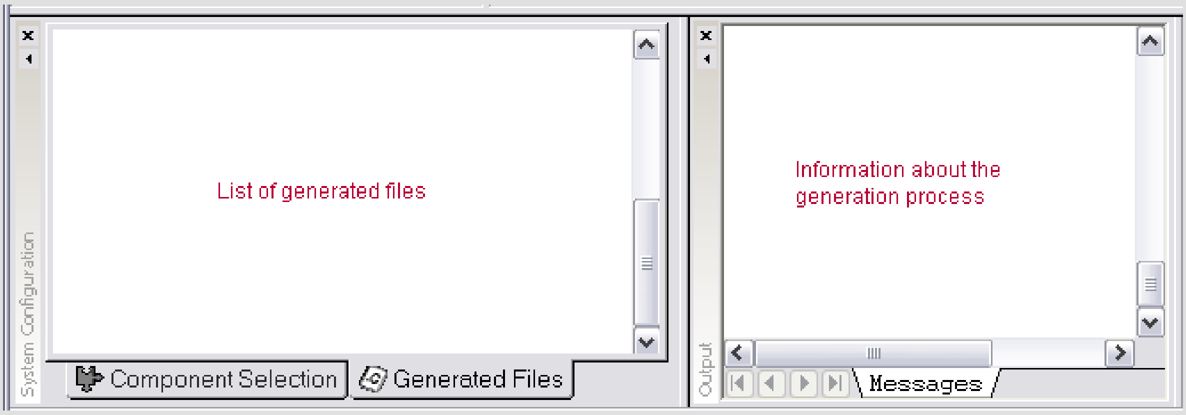
CBT1 register (hex) 6F

Acceptance registers

Optimize filters…

4.2.9 Settings are Finished

Generated Files **Messages**



Messages

Generated Files

4.3 STEP 3 Add Files to Your Application

- source code
- generated gendata

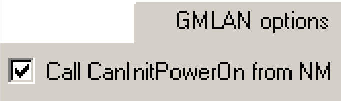
4.4 STEP 4 Adaptations For Your Application

4.4.1 Includes

il_inc.h

```
#include “il_inc.h”
```

4.4.2 Initialization


GMLAN options

☑ Call CanInitPowerOn from NM

```
/* make sure all interrupts are disabled*/
~~CanInitPowerOn();~~  /*If this initialization is
                     demanded to be done from the NM, you do not have to
                     call it.
                     The selection above can be done in the Generation
                     Tool*/
IlInitPowerOn();
TpInitPowerOn();
DescInitPowerOn();  /*optional, if CANdesc is used*/

/*Enable or Disable RX Messages based on Calibrations*/
IlSetRxMessageEnable(<Rx_Message_Name>, 0 or 1 for on or off);

/*Enable or Disable TX Messages based on Calibrations*/
IlSetTxMessageEnable(<Rx_Message_Name>, 0 or 1 for on or off);

IlSetOwnNodeAddress(SWCAN, <srcAddress>);

/*Get learnt source IDs from EEPROM (single channel system)*/
for( index=0; index<kIlNumberOfExtIdRxObjects; index++)
{
  IlSetRxMessageSourceAddress(index, GetBLearnedSourceId(index));
}
```

4.4.3 Cyclic calls to keep the components running

```
/*Network Management task*/
IlNwmTask();

/*Interaction Layer tasks, separated in Tx and Rx*/
IlRxTask();
IlTxTask();

/*Transport Protocol tasks, separated in Tx and Rx*/
TpRxTask();
TpTxTask();

/*Diagnostics task - CANdesc*/
DescTask();
```

4.4.4 Provide Callback functions

```
void ApplIlNodeCommActiveFailed( vuint8 srcAddress ){} /*for mixed IDs only*/

void ApplIlRxMsgSrcAddressLearned( IlReceiveHandle ilRxHnd, vuint8 srcAddress ){}  /*for mixed IDs only*/
void ApplIlNodeCommActiveRecovery( vuint8 srcAddress ){}  /*for mixed IDs
                                                            only*/
```

```
void ApplIlSourceAddressLearned( vuint8 srcAddress ){}    /*for mixed IDs
                                                           only*/
```

This depends on the selected diagnostic functions.

```
canuint8 ApplNwmSleepConfirmation( NM_CHANNEL_APPLTYPE_ONLY )
{
  return NmSleepOk;
}
```

```
void ApplNwmReinitRequest( NM_CHANNEL_APPLTYPE_FIRST  unsigned char VnNr,
unsigned char ReinitRequest ){}
```

```
void ApplTrcvrNormalMode( NM_CHANNEL_APPLTYPE_ONLY ){}
```

```
void ApplTrcvrSleepMode( NM_CHANNEL_APPLTYPE_ONLY ){}
```

```
void ApplTrcvrHighVoltage( NM_CHANNEL_APPLTYPE_ONLY ){}
```

```
void ApplNwmVnDeactivated( NM_CHANNEL_APPLTYPE_FIRST  unsigned char VnNr
){}
```

```
void ApplNwmVnActivated( NM_CHANNEL_APPLTYPE_FIRST  unsigned char VnNr
){}
```

4.5 STEP 5 Compile And Link

4.6 STEP 6 Test the Software Component

5 Further Information

5.1 Full CAN Message Transmission with Extended Ids

Priority: 3 bits	Parameter ID: 13 bits	Reserved: 5 bits	Source Address: 8 bits

Info

5.2 Take care when working with VNs

Caution

`IlPut`

`IlSetEvent`

`IlSetEvents`

Solution

`IlSetEvent`

`IlNwmGetStatus`

```
If( IlNwmStateNMActive( IlNwmGetStatus() ) ) == 1
```

```
{
  IlSetEvent…
}
```

StateOn flags

Send signals

```
ILGetTxDgnInfStateOn();
```

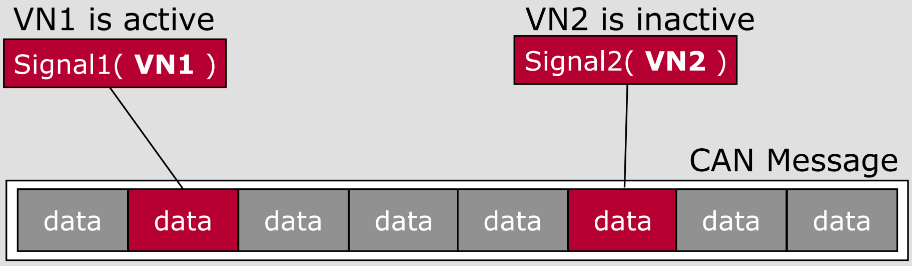
5.3 Validity of Signals

■

■

■

5.4 Signals assigned to different VNs and located in one message



`IlSetEvent`

Suggestion

`IlSetEvent`

5.5 Source Learning for Single Wire CAN with Mixed Identifiers (11 bit and 29 bit)

`ApplIlRxMsgSrcAddressLearned, ApplIlSourceAddressLearned`

5.6 Activation and deactivation of VNs

■

■

■

■

5.6.1 IlNwmActivateVN(channel, VN)

5.7 IlNwmDeactivateVN

6 Index