Calling the GM-Bootloader from CANdesc-Applications

Version 1.00
04/05/04

Application Note AN-ISC-2-1011_CANfblGM_Call_from_CANdesc

Author(s)	Armin Happel

Restrictions	Draft Document
Abstract	How to call the Vector bootloader from a diagnostic layer built with CANdesc-GM

Table of Contents

Author(s) Armin Happel
Restrictions Draft Document
Abstract How to call the Vector bootloader from a diagnostic layer built with CANdesc-GM

Table of Contents

1.0 Overview

This application note describes how to configure the GM's CDD-File to prepare a CANdesc-GM to call the Vector Flash Bootloader for GM.

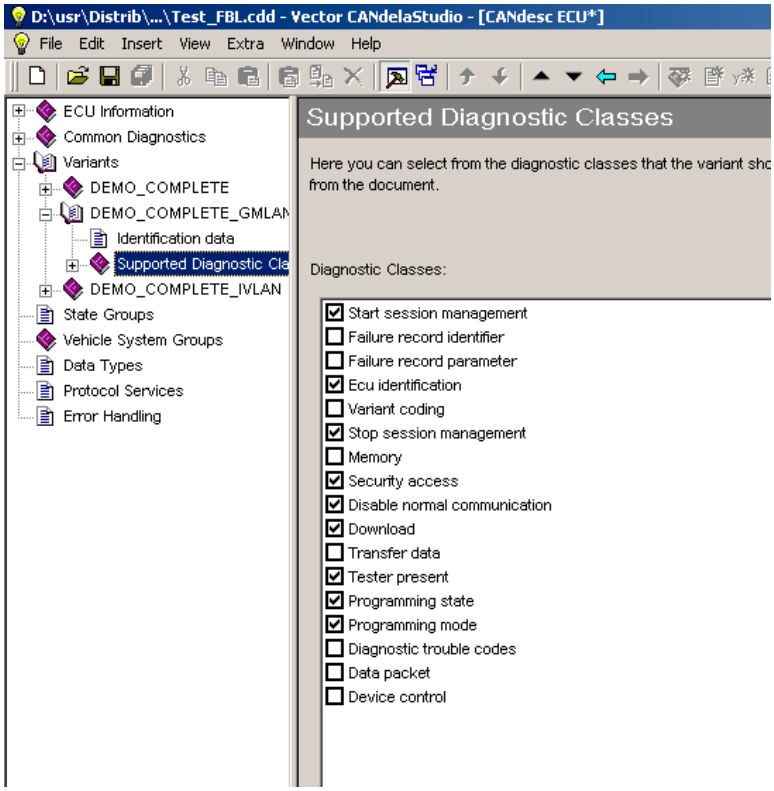
1.1 Target Group

The target group of this application note are ECU software developers and quality engineers involved in developing and testing CAN-based vehicle software.

2.0 Preparing the CDD-File with CANdelaStudio

The following snapshots describe how to set-up the CDD-file with CANdelaStudio.

As a first step, the following services need to be defined:



The picture above shows which services have to be supported to cooperate with a flash download. More checkboxes can appear in your configuration.

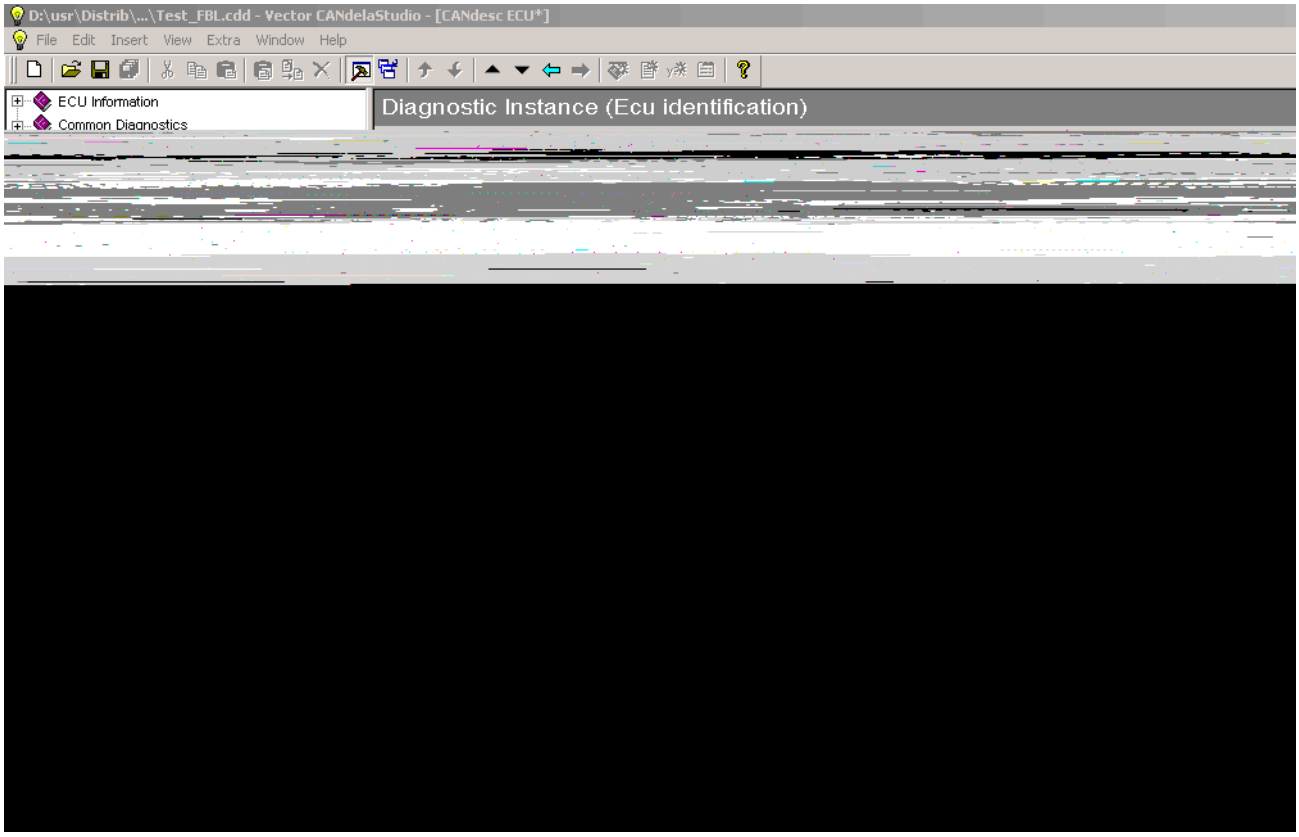
The jump to the bootloader must be configured within the Diagnostic Class “Download”. All other services are required to manage the prologue.

The prologue looks as follows:

FLASH REPROGRAMMING PROLOGUE

Mandatory requests	Optional requests

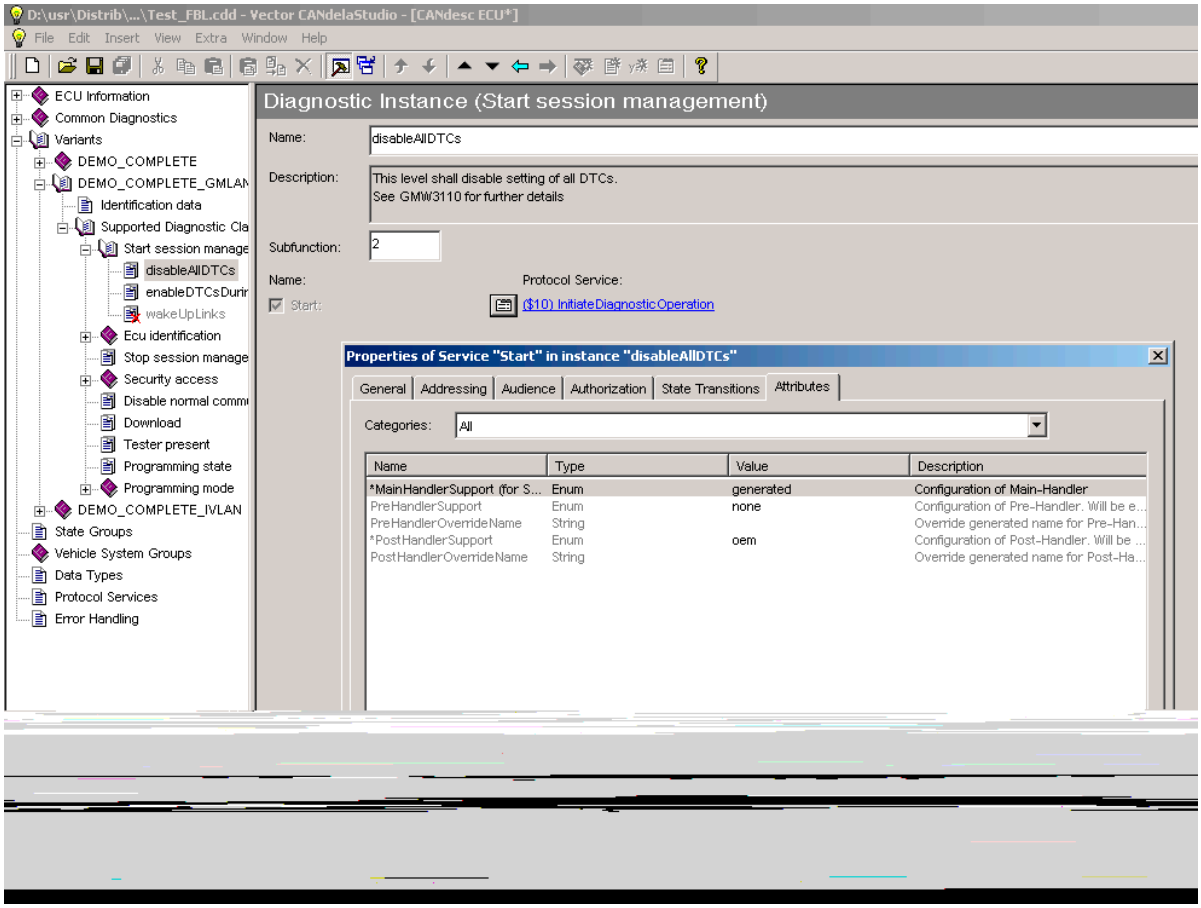
The application needs to support the services listed above.



The picture above shows the configuration for the service “ReadECUIdentification” ($1A B0). The return value is a constant, namely the ECU-Diagnostic Address. Therefore, a generated MainHandler can be configured within the property of ReadDataByIdentifier.

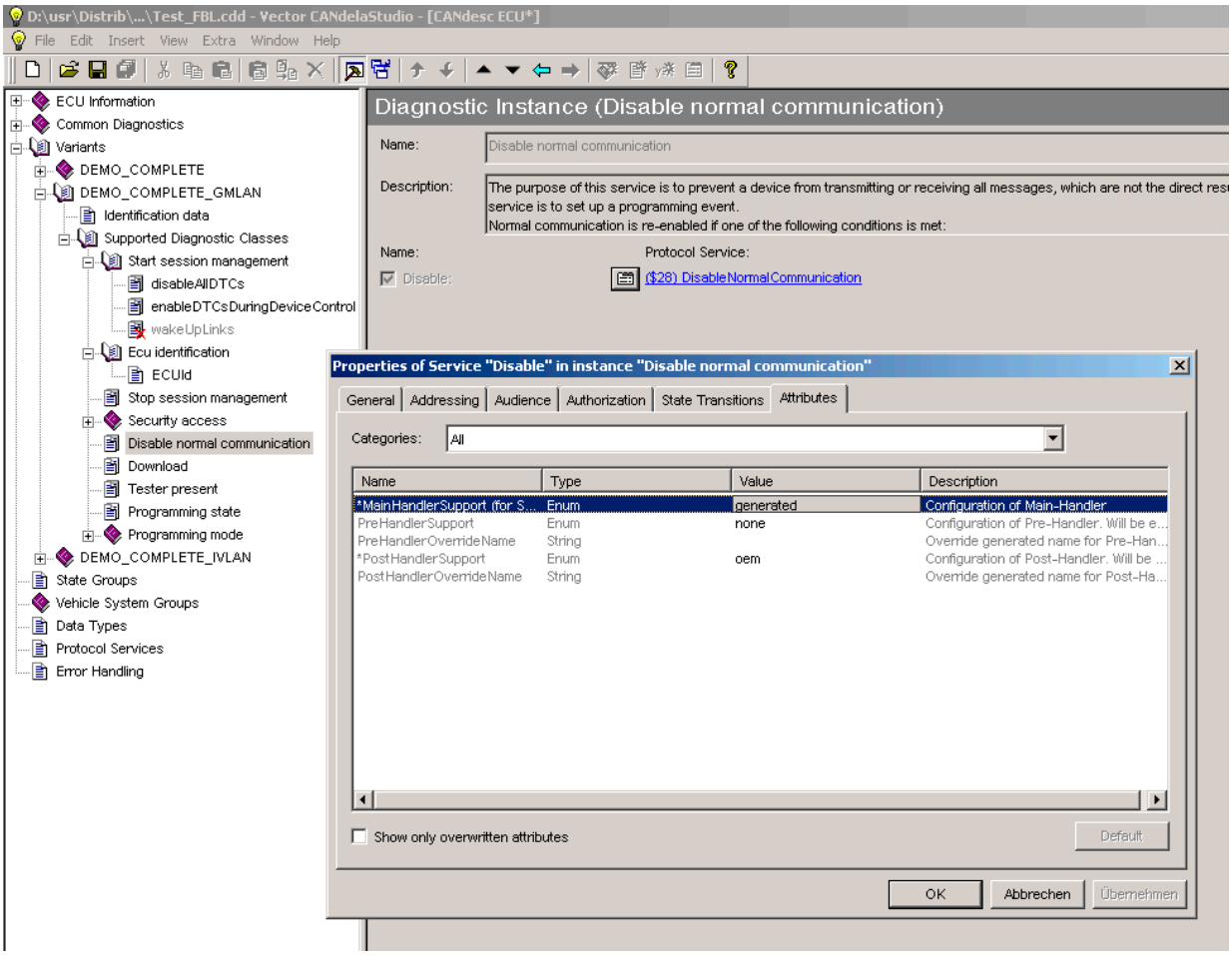
For a full support of the prologue, additional service instances for “ReadDataByIdentifier” need to be implemented. These instances must have the DIDs $C1 to $C9 if one or more calibration files need to be supported. This service instance reads the software module identifier from the GM-header[1] of the application or calibration files. Additional service instances that need to be supported are the DIDs $D1 to $D9 to retrieve the alphacode.

[1] The GM-header must be the section of the downloaded data. It contains the Module checksum, the SWMI and DLS-code and some additional information. Refer to GMW3110, section 11 for more information on GM-header.

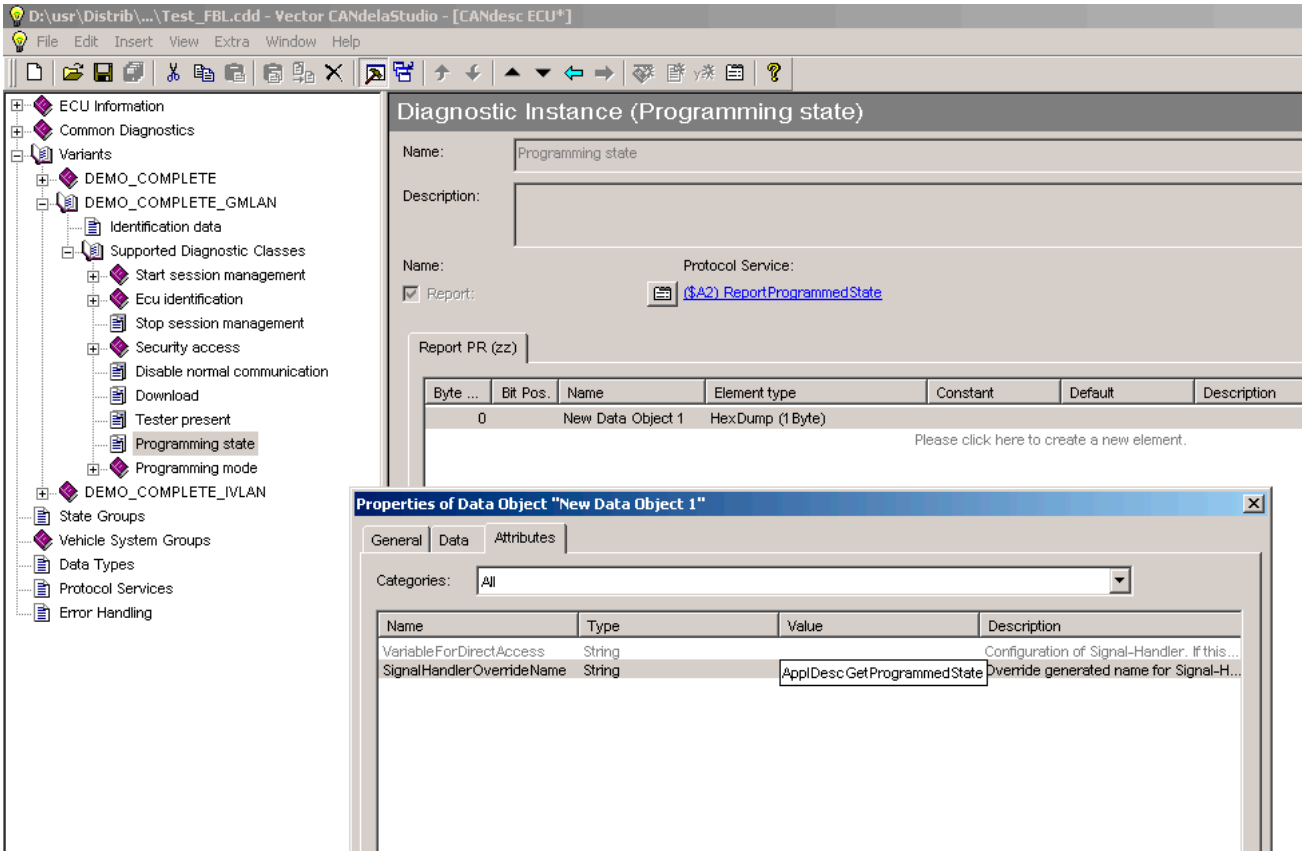


This next picture shows how to set-up the the optional service “Initiate Diagnostic Operation”.

The generated MainHandler-support of CANdesc can be used for this service.

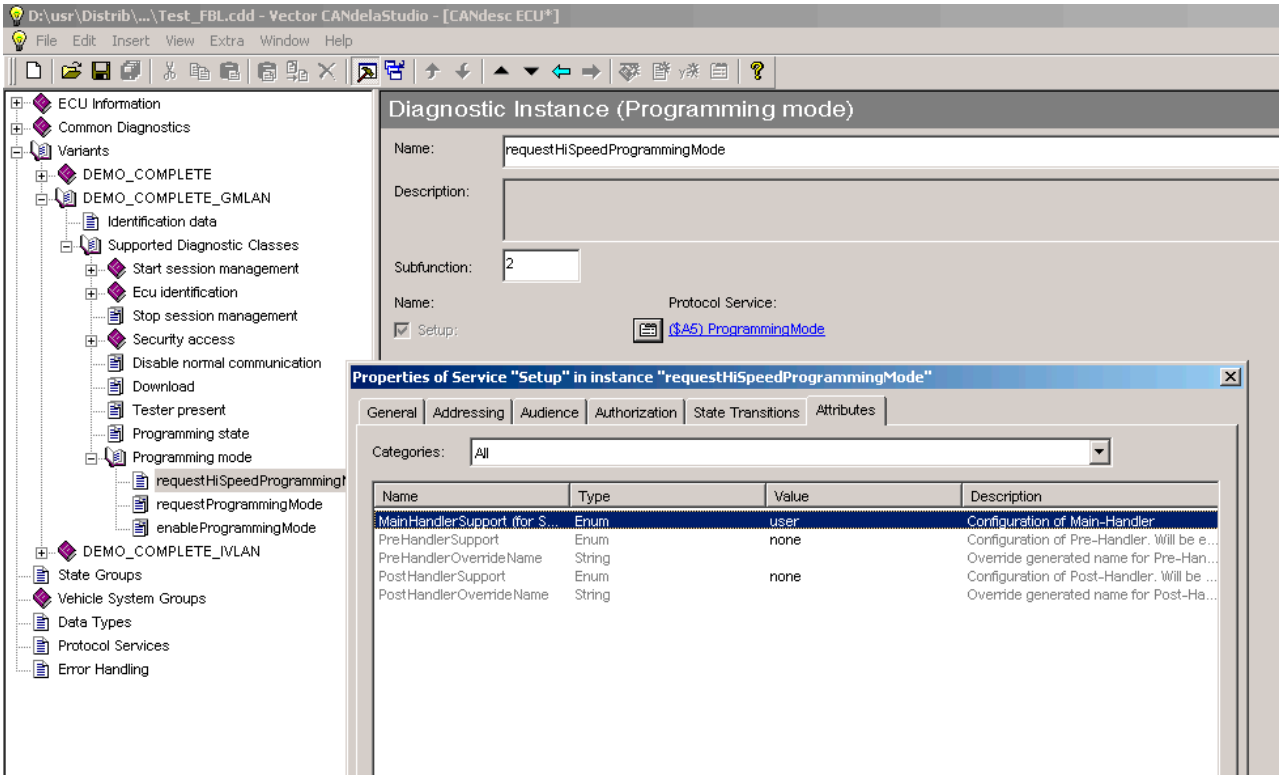


The picture above shows the configuration of the service DisableNormalComm ($28). CANdesc will automatically call the NM-Function llNwmDisableNormalComm(). This function of the GMLAN-NM will automatically shutdown all VNs (except VN #0).



This picture shows the configuration of the service ReportProgrammedState. The service returns a single byte that reflects the status of the application. This status is usually of type Fully programmed (value 0) and can therefore be a constant. Thus, it could be a fully generated service, too.

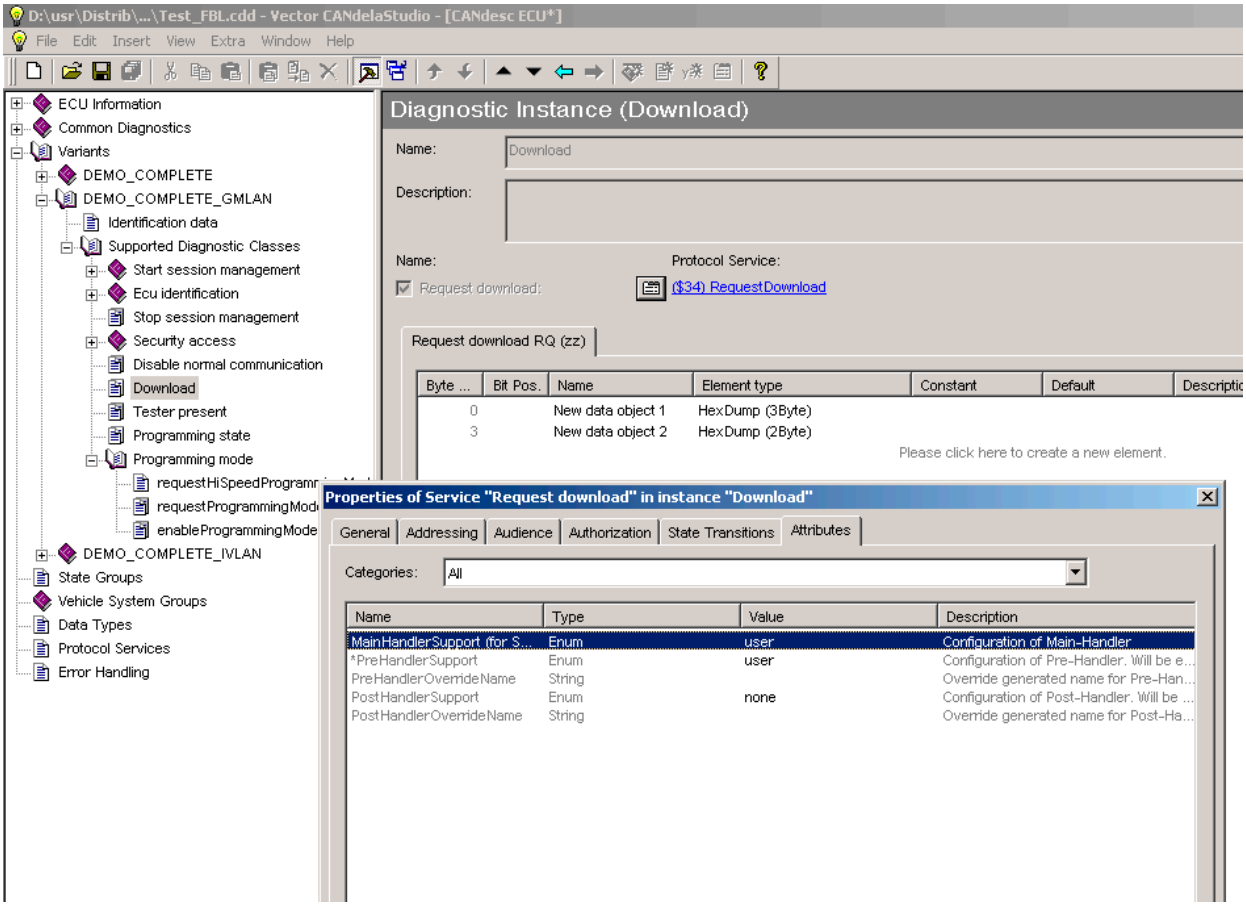
If you want to provide different values here, a signal handler can be set-up to adapt it, as it is described above.



The picture above shows now the configuration of the ProgrammingMode.

The service instance “requestHiSpeedProgrammingMode” is only necessary on the Single-Wire CAN. This must be enabled even if your ECU is not programmed in this mode. It is necessary to allow other ECUs to be programmed in this mode. On Powertrain Networks (aka. “hsCAN”) this service instance is not required.

Mandatory service instances are “RequestProgrammingMode” and “EnableProgrammingMode”. Note that only the service instances “RequestProgrammingMode” and “RequestHiSpeedProgrammingMode” can be denied. Once accepted, the following “EnableProgrammingMode” must be accepted as well. This service instance does not send any response. A generated Post-Handler of CANdesc will automatically call the Network Management function “**`IlNwmSetHispeedMode`**()” to switch to the Hi-speed on the Single-Wire CAN, when “RequestHiSpeedProgrammingMode” was requested before. Remember that you have to configure the generation tool to prepare the GMLAN-Network Management to switch to a higher baud rate (“supported for Hi-speed mode for single wire CAN”.).



The last service in the prologue is the request download. Note that this service is requested using the physical requests CAN-identifier.

A user-main handler must be configured for this Diagnostic Instance.

The application callback-function must include the definition-header of the bootloader. The application must not respond to this service, because the bootloader does this. Some parameters need to be passed to the bootloader, e.g. the bit-timing of the CAN-chip, the SPS_TYPE_B CAN-identifier the error status and the cause of the jump to the bootloader (usually the reception of the RequestDownload).

The following code sequence shows the implementation of a jump to the bootloader. It must be noted, that the example below is hardware dependent (here: the HC08). The parameters may be grabbed in different ways on other hardware platforms.

```
#include "fbl_def.h"

tCanInitTable  CanInitTable;

void ApplDescRequestDownloadDownload(DescMsgContext *pMsgContext)
{
   extern canuint8 CanInitCBTR0[],CanInitCBTR1[], CanInitCMCR0[], CanInitCMCR1[];

   /* ---------------------------------------- */
   /* --  Controller specific section      -- */
   /* -- Initialize CanInitTable structure -- */
   /* ---------------------------------------- */

   /* -- Initialize the pre-programmed CAN-identifier -- */
# if defined( C_COMP_COSMIC_08 )
# if defined(C_ENABLE_EXTENDED_ID)
   extern canuint8 CanTxIdMidHi[];   /* Middlehigh byte of identifier */
   extern canuint8 CanRxIdMidHi[];  /* Middlehigh byte of identifier */
# endif

   CanInitTable.TpRxIdHigh    = (canuint8)(CanRxIdHi[kTpRxHandle]);
   CanInitTable.TpTxIdHigh    = CanTxIdHi[kTpTxHandle];
#  if defined ( C_ENABLE_EXTENDED_ID )
   CanInitTable.TpRxIdLow     = (canuint8)((CanRxIdMidHi[kTpRxHandle]));
   CanInitTable.TpTxIdLow     = CanTxIdMidHi[kTpTxHandle];
#  else
   CanInitTable.TpRxIdLow     = (canuint8)((CanRxIdLo[kTpRxHandle]));
   CanInitTable.TpTxIdLow     = CanTxIdLo[kTpTxHandle];
#  endif
# endif

#if defined (DESC_ENABLE_REQ_HISPEED_PROG)
  if (g_descOemStateCtrl.hiSpeedMode == kDescHiSpeedModeActive)
  {
      CanInitTable.CanInitCMCR0  = CanInitCMCR0[kNmCanInitObjHispeed];
      CanInitTable.CanInitCMCR1  = CanInitCMCR1[kNmCanInitObjHispeed];
      CanInitTable.CanInitCBTR0  = CanInitCBTR0[kNmCanInitObjHispeed];
      CanInitTable.CanInitCBTR1  = CanInitCBTR1[kNmCanInitObjHispeed];

      CanInitTable.RequestProgrammingMode = REQUEST_DOWNLOAD_MODE_HIGHSPEED;
  }
  else
#endif
  {
      CanInitTable.CanInitCMCR0  = CanInitCMCR0[kNmCanInitObjStd];
      CanInitTable.CanInitCMCR1  = CanInitCMCR1[kNmCanInitObjStd];
      CanInitTable.CanInitCBTR0  = CanInitCBTR0[kNmCanInitObjStd];
      CanInitTable.CanInitCBTR1  = CanInitCBTR1[kNmCanInitObjStd];

      CanInitTable.RequestProgrammingMode = REQUEST_DOWNLOAD_MODE_LOWSPEED;
  }
   /* ProgrammedState: fully programmed */
   CanInitTable.ProgrammedState = 0x00;
```

```
    /* This is not an error-condition. Set error to '0'. */
    CanInitTable.ErrorCode = 0x00;

    /* Start flash boot loader. Never comes back.     */
    /* Trigger (window-)watchdog before if necessary. */
    /* But take care of Service timeout               */
    (FblHeaderTable->FblStartFct)(&CanInitTable);

    /* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! */
    /* !! NOTE: Code should never return here !!! */
    /* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! */
}
```

3.0 Contacts

Vector Informatik GmbH
Ingersheimer Straße 24
70499 Stuttgart
Germany
Tel.: +49 711-80670-0
Fax: +49 711-80670-111
Email: FblSupport@vector-informatik.de

Vector CANtech, Inc.
39500 Orchard Hill Pl., Ste 550
Novi, MI 48375
Tel: (248) 449-9290
Fax: (248) 449-9704
Email: info@vector-cantech.com

VecScan AB
Fabriksgatan 7
412 50 Göteborg
Sweden
Tel: +46 (0)31 764 76 00
Fax: +46 (0)31 764 76 19
Email: info@vecscan.com

Vector France SAS
168 Boulevard Camélinat
92240 Malakoff
France
Tel: +33 (0)1 42 31 40 00
Fax: +33 (0)1 42 31 40 09
Email: information@vector-france.fr

Vector Japan Co. Ltd.
Nishikawa Bld. 2F, 3-3-9 Nihonbashi
Chuo-ku Tokyo 103-0027
Japan
Tel: +81(0)3-3516-7850
Fax: +81-(0)3-3516-7855
Email: info@vector-japan.co.jp