Diagnostic Message Example

J1939 DM Application

Application Note

crosscontrol

Making machines safer, smarter and more productive

Contents

Revision history

Rev	Date	Author	Comments
1.0	2023-06-08	Daniel N.G	Implementation
1.0	2023-06-16	Foad Ghafourian	Documentation

Glossary

Word/Abrevation	Explanation
DM	Diagnostic Messages
DTC	Diagnostic Trouble Codes
SPN	Suspect Parameter Number
Qt	Development framework Qt | Tools for Software Development
FMI	Failure Mode Indicator
API	Application Programming Interface
CM	Conversion Method
VM	Virtual Machine
OCC	Occurrence Count

1. Introduction

This is an example project using CrossControl Data Engine and Fieldbus Access Management tools (qt creator plugins) for receiving and displaying J1939 DM1 messages in a Qt Quick2 (QML) application. Incoming messages are received as Data Engine BLOB data messages. The first two bytes of the ByteArray hold the CAN message source address. The rest of the message contains the Diagnostic Trouble Codes **DTC**.

The following information is contained within a DTC.

- **Suspect Parameter Number (SPN)**
- **Failure Mode Indicator (FMI)**
- **Occurrence Count (OCC)**
- **Conversion Method (CM)**

An active Diagnostic message contains the following parameters.

Parameter	Description
Protect Lamp Status	Lamp to indicate a problem with a vehicle system that is most likely not electronic subsystem related. e.g. Coolant Temperature has exceeded its defined range.
Amber Warning Lamp Status	Lamp to indicate a problem with the vehicle system but the vehicle does not need to be stopped immediately.
Red Stop Lamp Status	Lamp to indicate a problem that is severe enough to warrant stopping the vehicle.
Malfunction Indicator Lamp Status	Lamp to indicate when there is an emission related trouble code active.
DTC[0], DTC[1], DTC[2] ...DTC[n]	The DTC list contains 1 or more DTC's. Each entry in the list contains an SPN, FMI, occurrence Count, and Conversion Method.

In case you are interested to read and learn more about diagnostic messages, please head to these following suggestions below:

- J1939 Software Stack
- Monitor J1939 Diagnostic Trouble Codes

2. Installation

The example application builds in CrossControl UX-Designer environment using Qt Creator SDK and qmake builds system. A basic **LinXManager_FA_configfile.json** is included in the project, may be viewed by LinX-Manager Fieldbus Access (QtCreator plugin in UX-Designer). This includes only a basic J1939 config with one CAN-interface (CAN-bus 0) and two diagnostic messages checked (DM1/DM2).

The application can be executed on a CCpilot target display or in the virtual machine (UX-Designer)

Requirements

UX-Designer: UX Designer | CrossControl
Download Link: UX-Designer Development Environment Downloads | CrossControl
For more readings head to the online documentation: UX Designer V5.0. Online Documentation

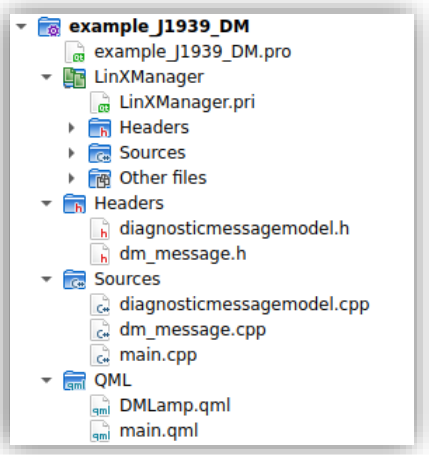
CrossControls **UX-Designer** has all the neccessery development environment tools such as: **Qt Creator**, **Fieldbus Access**, **Data Engine** and **SDK** packages that can be installed in order to allow developers to start development on CCpilot display computers.

CCpilot Display Computers: Display computer | CrossControl

3. Deploy in Virtual Machine (UX-Desginer)

After downloading the "*example_J1939_DM*" file, move it to your VM and unzip it, then open the "*example_J1939_DM.pro*" file in Qt Creator.

Make sure that all the necessary files are included, also the *Fieldbus Access* configuration file!



First you need to start *Data Engine* runtime from a terminal:

```
$ /opt/bin/dataengineserverApp --debug &
```

After that, start the *Fieldbus Access* runtime with path to the deployed configuration file:

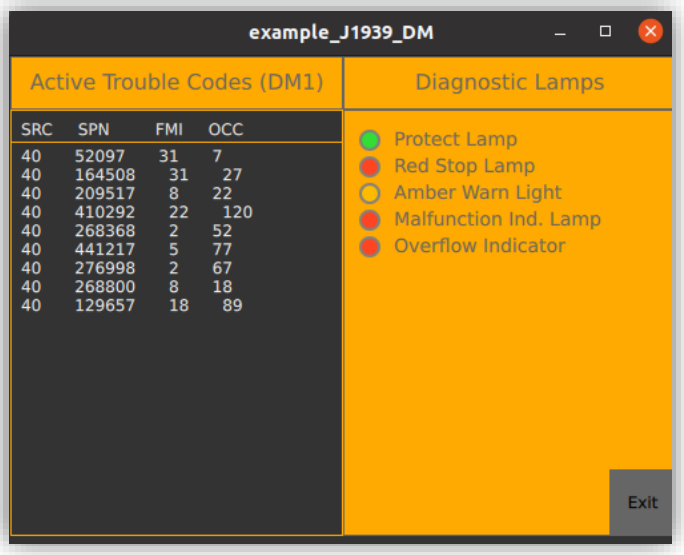
```
$ /opt/bin/fieldbusaccess /home/ccs/qt/example_J939_DM/LinXManager_FA_configfile.json &
```

Note that the path to your Qt project may be different and you need to make sure that you use the right path!

Next build and run the application from Qt Creator. If any DM1 messages are received on CAN-interface 0 they will show up on the display.

For example, you can simulate and send DM1 (FECA) messages by using the following CAN-utils command:

```
$ cangen can0 -e -I 18FECA28 -L 8 -g 1000 -v
```



The simulated DM1 messages on VM

4. Deploy on CCpilot Display Computer (V1200)

This procedure is very similar to the previous chapter (the on in VM), but you need to deploy the application on a CCpilot Display Computer. This is demonstrated in the UX-Designer Online Documentation. You will also need a physical CAN connection (a dongle or bus-tool) that can send and receive CAN messages between your host computer and the CCpilot display!

Before you start with the deployment you need to connect to your CCpilot display.

Log into the target by the IP-Address using SSH: Username: *ccs* Password: *default*

```
$ ssh ccs@<IP-Address>
```

Once you are logged in, start by running the *Data Engine* runtime first:

```
$ /opt/bin/dataengineserverApp --debug &
```

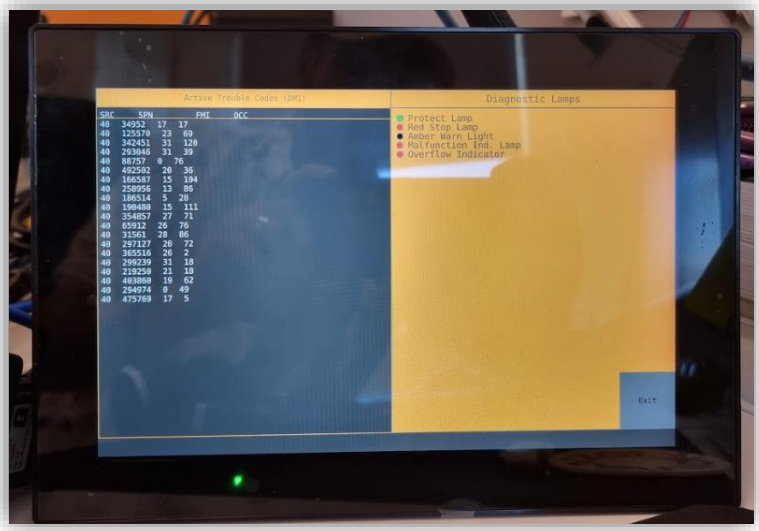
After that, start the Fieldbus Access runtime with path to the deployed configuration file:

```
$ /opt/bin/fieldbusaccess /opt/bin/LinXManager_FA_configfile.json
```

Next start your application. If any DM1 messages are received on can-interface 0 they will show up on the display.

You can use the same CAN-utils command for simulating and sending DM1 (FECA) messages:

```
$ cangen can0 -e -I 18FECA28 -L 8 -g 1000 -v
```



The simulated DM1 messages deployed on CCpilot V1200

For more information head to our website: [CrossControl | Making Machines Smart, Safe and Productive]
If you have questions or issues, please contact our support team: [Support | CrossControl]