

2002:E014



UNIVERSITY
TROLLHÄTTAN · UDDEVALLA
DEPARTMENT OF TECHNOLOGY

DISSERTATION

CAN Bus Diagnostic Tool for PocketPC

Marcus Pettersson

July 3, 2002

University of Trollhättan/Uddevalla
Department of Technology
Box 957, S-461 29 Trollhättan, SWEDEN
Phone: +46 520 47 50 00 Fax: +46 520 47 50 99
E-mail: teknik@htu.se

DISSERTATION

CAN Bus Diagnostic Tool for PocketPC

Summary

Today's development, where almost everything related to engine control is electronic, there is an urgent need for development of new fault searching tools. The old mechanical engines are phased out being substituted by fully electronic engines e.g. with CAN bus systems. This it is a very rapid process that put demands on human resource development as well as tool and systems used. Volvo Penta has chosen to develop those tools on a PDA (Personal Digital Assistant) platform, instead of using a traditional computer or laptop. The objective is to make life easier for service engineers by providing small and easy-to-use hand-held devices. For this purpose Volvo Penta wanted to develop a prototype software that read, analyze and display the information from the CAN data link SAE J1939. The software was developed with eMbedded Visual C++, adapted for PocketPC, or better-said Microsoft Windows CE. For the first phase this software was developed for Volvo Penta engines with EDC and EMS because they all rely on CAN SAE J1939. We decided to name the application "CAN Bus diagnostic tool for PocketPC", same as the title of this dissertation.

Keywords: CAN, SAE J1939, PDA

Publisher: University of Trollhättan/Uddevalla, Department of Technology
Box 957, S-461 29 Trollhättan, SWEDEN
Phone: + 46 520 47 50 00 Fax: + 46 520 47 50 99 E-mail: teknik@htu.se

Author: Marcus Pettersson

Examiner: Bo Svensson

Advisor: Esbjörn Sjöberg, AB Volvo Penta

Subject: Electrical Engineering, Information Systems

Language: English **Number:** 2002:E014 **Date:** July 3, 2002

Foreword

This dissertation “Can Bus Diagnostic Tool for PocketPC”, have been performed at AB Volvo Penta in Sweden, Gothenburg. The proposal for the dissertation comes from Anders Näzell and via Esbjörn Sjöberg it was given to me. Esbjörn Sjöberg has been my supervisor, he has helped and supported me during this dissertation.

Examiner and supervisor at University in Trollhättan/Uddevalla, has been Bo Svensson at Department of Technology.

Above all I want to thank Tobias Rasmusson at Newmad Technologies AB who helped me get going with embedded Visual C++ when we decided to change programming language from Visual Basic to Visual C++. He also gave a helping hand sorting out some programming problems and answering questions. I also want to thank Henrik Fagrell, at Newmad Technologies, that had me connected with skilled programmers at Newmad Technologies - Tobias Rasmussen and Jonas Larsson.

Many thanks also to Product Development that gave me access to their test laboratory and by that, there test equipment so I could test the program that was developed; Anders Johansson, Anders Holmström, Jonas Welinder and Pål Loodberg.

Last, I want to thank all involved at AB Volvo Penta, everyone at unit Customer Support, and especially my supervisor Esbjörn Sjöberg.

Table of Contents

1	Introduction	1
1.1	<i>Background</i>	<i>1</i>
1.2	<i>Purpose and target</i>	<i>1</i>
1.3	<i>Delimitations</i>	<i>1</i>
1.4	<i>Course of action</i>	<i>1</i>
1.5	<i>Equipment</i>	<i>2</i>
1.5.1	Hardware	2
1.5.2	Software	2
1.6	<i>Changes in condition</i>	<i>2</i>
2	CAN data link	3
2.1	<i>CAN J1939</i>	<i>3</i>
2.2	<i>Data Messages at CAN J1939</i>	<i>6</i>
2.2.1	VP Status	7
2.2.2	VP Engine Industry	7
2.3	<i>CAN controller</i>	<i>7</i>
2.3.1	LAPcan II controller card	8
2.3.2	DRVcan 251 cable	8
3	Volvo Penta engines	8
3.1	<i>EMS – Engine Management System</i>	<i>8</i>
3.2	<i>D12 Interface</i>	<i>10</i>
3.2.1	CAN link J1939	10
3.2.2	CIU	10
3.2.3	Stand alone connections	10
4	Realization	10
4.1	<i>Programming</i>	<i>10</i>
4.1.1	Visual Basic	11
4.1.2	Visual C++	11
4.2	<i>PocketPC programming</i>	<i>11</i>
4.2.1	Database for Signals	11
4.2.2	Reading CAN J1939	12
4.2.3	The Program (Can Bus Diagnostic Tool for PocketPC)	12
4.2.4	GUI (Graphic User Interface)	13
4.2.5	First version	14
4.2.6	Second version	14
4.2.7	Third version	14
4.3	<i>Connection to Volvo Pentas Engine cable harness</i>	<i>14</i>
5	Results	15
6	Conclusions	15
6.1	<i>Analyses of the result</i>	<i>15</i>
6.2	<i>Recommendations for continues work</i>	<i>16</i>
7	Reference	17
7.1	<i>Books and datasheet</i>	<i>17</i>
7.2	<i>Websites</i>	<i>17</i>
	Appendix	18

List of Symbols

API stands for *Application Programming Interface*

CPU stands for *Central Processing Unit*, the key component of a computer system, which contains the circuitry necessary to execute program instructions.

EDC stands for *Electronic Diesel Control*

EMS stands for *Engine Management System*

GUI stands for *Graphical User Interface*, is the software interface designed to standardize and simplify the use of computer programs.

PCMCIA stands for *Personal Computer Memory Card International Association* and is an external accessible expansion slot that accepts compatible cards for enhancing the computers functions.

PDA stands for *Personal Digital Assistant* and is a hand-held computer, often pen based that provides especially organization software as appointment calendar and communication tools.

SAE stands for *Society of Automotive Engineering* and is a world wide standard that almost everyone of engine-, truck-, bus-, genset-builders and so on are using today as an interface between e.g. vehicle-transmission-engine-hydraulic-systems.

1 Introduction

This dissertation was performed at AB Volvo Penta in Sweden, Gothenburg where I designed an analyzer tool for their engines using CAN SAE J1939. This tool was designed for PocketPC, or more exact for a Compaq iPAQ with CAN hardware interface in terms of a PCMCIA card LAPcan II from Kvaser AB.

1.1 Background

The background to this dissertation is that AB Volvo Penta want to have an analyze tool for the CAN-link J1939 on a handheld computer called PDA (Personal Digital Assistant).

Today the engines from Volvo Penta are for the most part electronic and equipped with EDC, EMS or any other electronic control system. At installations of Volvo Pentas components in other manufactures products a requirement came up; the possibility to analyze what is happening in the complete installation. Concrete it meant the possibility to read what signals are inactive or active on the protocol SAE J1939, which is used to/from the control unit at Volvo Pentas products and the Customers own panels. This way you can identify errors that occur in new installations or in existing installations in the aftermarket.

1.2 Purpose and target

Develop an interface (tool) for installations with EMS, existing of a PDA, software and necessary cables for the connection to the application via Volvo Pentas engine cable harness at SAE J1939. The tool shall be able to capture the traffic at CAN J1939 and present the information in plain language, Swedish or English. A typical user is an educated product specialist from Volvo Penta, alternatively a well-educated service engineer that serves in the aftermarket at a service dealership.

The prototype shall be an English version with preparations to handle more languages.

1.3 Delimitations

The tool shall only handle engines with EMS and the prototype will only be in English.

The Hardware should be Compaq's iPAQ and necessary software will be developed for PocketPC in Visual Basic. The development platform will be Microsoft eMbedded Visual Tools with several smaller components.

1.4 Course of action

Learn how the CAN SAE J1939 works and how messages appear and look like. Then also learn to program Visual Basic, which was a new program language for me. During

the dissertations we decided to change the programming language to Visual C++, why I was forced to learn how to program Visual C++ too. Supply Volvo Penta with cable harness and connectors for SAE J1939.

Identify the messages on the CAN SAE J1939. Program a prototype that will translate the messages at the CAN to English. Documentation was written when necessary, during the dissertation.

1.5 Equipment

The equipment I used in this dissertation is presented in sections 1.5.1 and 1.5.2, hardware and software.

1.5.1 Hardware

Compaq iPAQ H3850, Compaq Expansion Jacket PC CARD For IPAQ H3XXX, Xircom CompactCard Ethernet 10 Type II

Kvaser LAPcan II – two channel CAN-controller card, two DRVcan 251 cables

Computer, Dell OptiPlex GX, P2 350MHz, 192 Mb ram.

Test equipment for simulation of signals for real engine.

1.5.2 Software

Microsoft Windows 2000, Microsoft Office 2000, Microsoft eMbedded Visual Tools, The PocketPC 2002 SDK. Kvaser SDK and drivers for PocketPC.

1.6 Changes in condition

From the beginning AB Volvo Penta decided that the program should be written in Visual Basic. It was also the intention to use another hardware or CAN interface. The CAN interface that should have been used is manufactured by a company named Setek, but they were not ready when we needed it. Then we looked into alternative solutions, and found a CAN interface from Kvaser, a PCMCIA card named LAPcan II. When we got that card we found out that the programming libraries for Visual Basic wasn't compatible with eMbedded Visual Basic for PocketPC

We then discussed the possibility to develop a dll that could be called from within Visual Basic. This never became a real alternative, since it is hard to develop a dll with no previous experience of it.

The decision was then to leave eMbedded Visual Basic for eMbedded Visual C++. I thought that was the best alternative, because eMbedded Visual C++ is much more adapted to programming on PocketPC.

2 CAN data link

Here facts are being described regarding the CAN data link.. About the CAN data link I have described some facts from standard, about the data messages and how you read the messages from the data link.

In order to develop an analysis tool for the CAN J1939, you need to know the basic principles of J1939. For that reason and to learn how it works, I had to study the standard document for CAN J1939, SAE J1939 and some other literature.

2.1 CAN J1939

The J1939 is the most used standard in the heavy vehicles industry. The first specification of J1939 came 1994. J1939 is based on the CAN protocol and it handle completely or partly, services from the application and transport layer and lower, according to Figure 1.5.2-1.

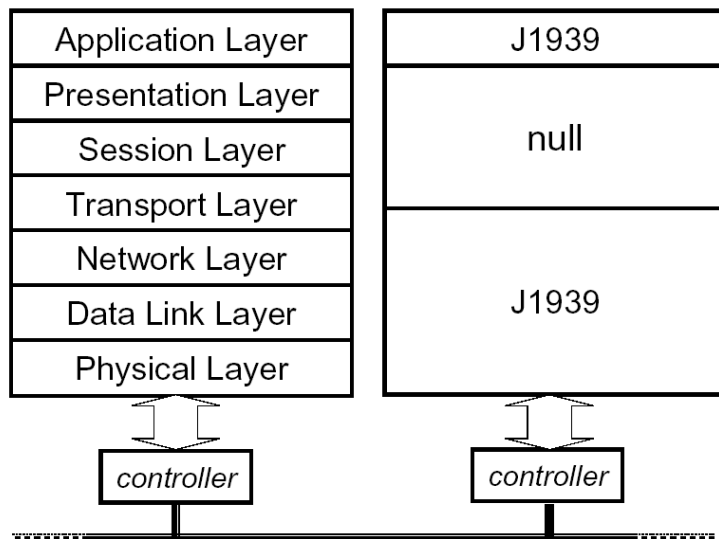


Figure 1.5.2-1 - The OSI model for SAE J1939

The functionality of each layer is:

1. Physical — Concerns the transmission of structured bit stream over physical media; deals with the mechanical, electrical, functional, and procedural characteristics to access the physical media
2. Data Link — Provides the reliable transfer of information across the physical layer; sends blocks of data (frames) with the necessary synchronization, error control, sequence control, and flow control;
3. Network — Provides upper layers with independence from the data transmission and switching technologies used to connect systems; responsible for establishing, maintaining, and terminating connections;
4. Transport — Provides reliable, transparent transfer of data between end points; provides end-to-end error recovery and flow control; provides segmentation and reassembly of very large messages;
5. Session — Provides the control structure for communication between applications; establishes, manages, and terminates connections (sessions) between cooperating applications;
6. Presentation — Provide independence to the application process from differences in data representation (syntax); and
7. Application — Provides access to the OSI environment for users and also provides distributed information services.

Bus access by message priority

- CSMA (Carrier Sense Multiple Access) / CR (Collision Resolution)

Bus access conflict is resolved by arbitration

- Bit-wise
- Non-destructive
- Guaranteed latency time

CAN has no node addresses, every node receive every message and decides itself whether to use it or not. It also has an extensive error checking, with five different checks. Every connected node participates. The data consistency is secured in the way that all nodes or none accept a message.

Different Bus Management Methods can be applied for CAN systems, for example;

- Bit-wise arbitration
- Master/Slave

- Daisy Chain
- TDMA

CAN is only a low level specification, that means that a higher layer protocol always is required. The capability of CAN is restricted by the higher layer protocol chosen;

- Market segment
- Real-time requirements
- Product Administration requirements, etc.

For Priority and/or Identification the CAN J1939 can use 11-bit as you can see in Figure 1.5.2-2 or 29-bit as shown in Figure 1.5.2-3.

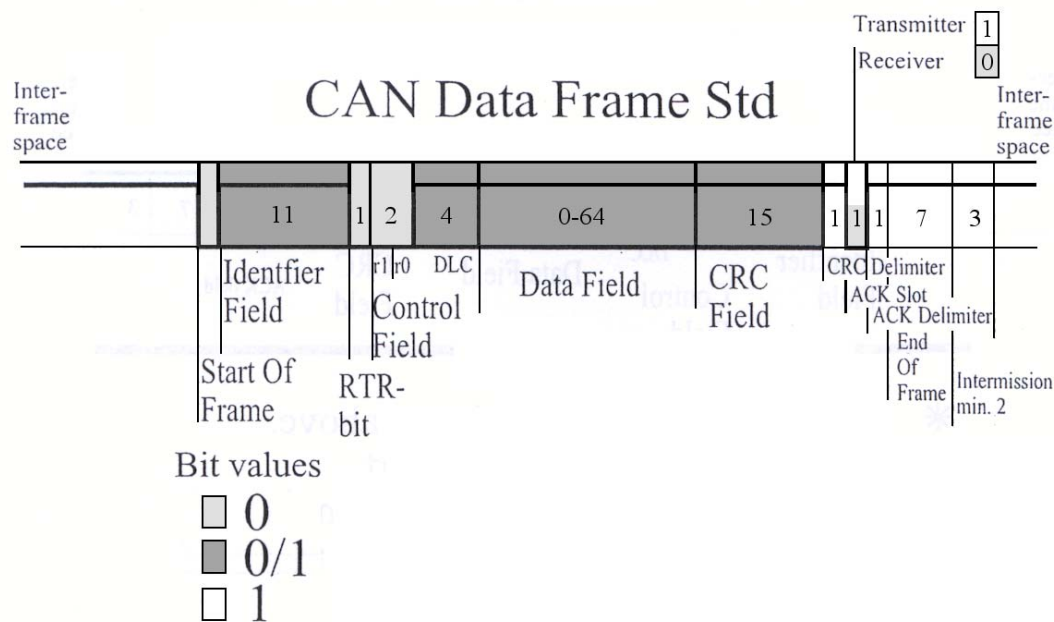


Figure 1.5.2-2 - CAN 11bit identifier

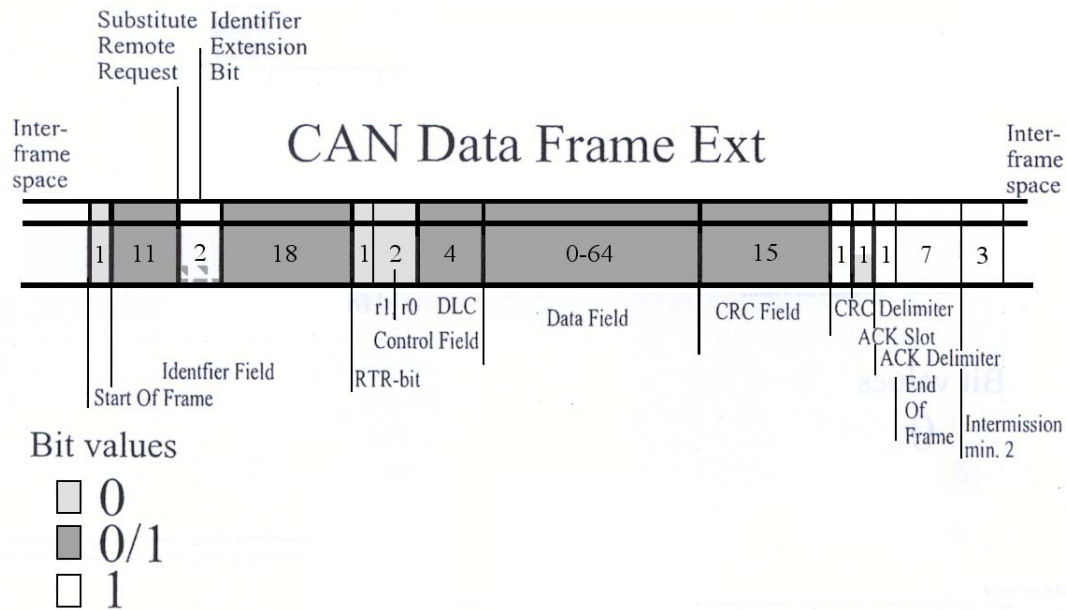


Figure 1.5.2-3 - CAN 29bit identifier

The 11-bit identifier is standard and the 29-bit is the extended version. Volvo uses the extended frames.

The arbitrations are done or performed in the identifier part and it also sets the priority of the message. The Remote request bit (RTR) is a part of this field. CAN-controllers often support this identification of the message by filtration.

In Control and Data length fields resides the main function Data Length Code (DLC). DLC can have values from 0 to 8. Values from 9 to 15 are not allowed. Two bits of the control field are used to indicate extended frames. When you use standard frames these bits are reserved fixed dominant.

The Data field or Message field can be from zero to eight byte, and it is always a full byte with 8-bits. These bytes can have any value. Some CAN controllers can extend ID filtration into the data field.

The CRC field is a checksum of the bits in the message and it is also optimized for this type of message. The CRC check is only one of the error checks in the CAN communication.

Acknowledgment field is to secure at least one receiver, the transmitter sets it to 1 and the receiver sets it to 0. Then you have fixed value bits, which have a fixed value for every message frame. [1]

2.2 Data Messages at CAN J1939

Data messages sent at CAN J1939, is surrounded with various identifier and control fields, according to previous caption CAN J1939. This message contains the data field with the signals and this message is 8 bytes. The signals are both transmitted and

received by the EDC. It is ten messages that have to be identified and then worked up on bit level and then viewed at the display. Just two of them are Volvo Penta specific and are called VP Status and VP Engine Industry. You can read about them in chapter 2.2.1 and 2.2.2. The other eight is according to the SAE J1939 standard document. All those messages including VP Status and VP Engine Industry are technically described in Appendix A and Appendix B.

2.2.1 VP Status

VP Status has 7-8 signals, which depends of the engine type we want to analyze. This type contains of the following signals; Start request that indicates if you want to start the engine, Stop request is the opposite and therefore a stop signal. Governor mode request (depends on model) indicates if the engine shall operate in engine speed mode or droop mode. This message also indicates if you selected the engine to run at idle speed. You can also see what frequency you have selected (either be 1500 rpm or 1800 rpm), in other words what engine speed in rpm that is chosen for engine operation. You can also see the status of the diagnosis switch and preheat request (not used in genset specification). The last signal is the throttle position, which is viewed as a value between 0-100 (%). [7, 8]

2.2.2 VP Engine Industry

This message consists of 12-13 signals depending on which engine we are analyzing. It contains of a preheat indication (depends on model) and an indication whether the engine is running or not. The running indication shows if the engine is running and has a work interval that starts at 5% below nominal engine speed, consequently 5% below 1500 rpm or 5% below 1800 rpm. Then there is also some alarms: Overspeed alarm, oil pressure alarm, oil temperature alarm, coolant temperature alarm, coolant level alarm and finally the charge alarm. VP Engine also contains indications as buzzer, general lamptest, buzzertest / lamptest, EMS Diagnose Yellow lamp and EMS Diagnose Red lamp. [7, 8]

2.3 CAN controller

The CAN controller interface that is used with the Compaq iPAQ H3850 is one LAPcan II PCMCIA card from Kvaser AB, and two DRVcan 251 cables because the card has two channels. From the beginning, the intention was to use a the CAN controller interface from SETEK Elektronik AB, but they did not have their interface ready in due time for this project, and even so, there would not have been so many copies of it. For those reasons we chose the Kvaser products..

2.3.1 LAPcan II controller card

The CAN controller LAPcan II is a PC card (PCMCIA) with two channels. It has dual Philips SJA1000 CAN controllers, and it has a high performance microprocessor, memory architecture and has according to the manufacturer enhanced ESD robustness. [11]

2.3.2 DRVcan 251 cable

The DRVcan 251 is a combined cable, connector and CAN driver for the LAPcan PCcard CAN interface. The DRVcan 251 contains an industry standard, high-speed CAN driver circuit. It uses a high-speed, up to 1 Megabit/second. [11]

3 Volvo Penta engines

Here I give an overview about the most elementary facts for the engines. This software is made for the engines that are for the most part electronic and equipped with EDC, EMS or any other electronic control system.

3.1 EMS – Engine Management System

This means that an electronic unit controls the engine. Engine starter and injectors are controlled by the engine management system. The following data is measured for calibrating optimal amount of fuel and used for control strategy:

- Engine speed, measured both on cam wheel and flywheel
- Boost pressure
- Boost temperature
- Coolant Temperature
- Oil Pressure
- Oil Temperature
- Fuel alarm; summer alarm indicating water in fuel or low fuel pressure
- Coolant level

The engine management system uses two different data links for communication.

- SAE J1939 (CAN) for engine control & engine monitoring.
- SAE J1587 for diagnosis, software download, parameter setting, back-up start, stop, speed control (All-speed engines).

The system also has some control functions: Injection control, Engine protection and Cylinder balancing.

- *Injection control*
Precise calculations of amount of fuel and timing of injection. This gives powerful response with virtually no smoke and low emissions.
- *Engine protection*
When needed, the engine will automatically shut down or decrease delivered torque in order to not get damaged.
- *Cylinder balancing*
The control system compensates for production differences of injectors during idle speed.

Coolant temperature:	Shut down (97-103 °C).Coolant level: Shut down.
Boost temperature:	Shut down (85 °C).Oil temperature: Shut down (122-132 °C).
Oil pressure:	Shut down (2,2 / 0,4 Bar at 1500/Idle)Water in fuel: Error code, no decrease
Low fuel pressure:	Error code, no decrease

Engine protection All-speed

Coolant temperature:	Shut down (3-4sek) (97-103 °C).Coolant level: Shut down.
Boost temperature:	Decreasing to 50% power increasing to full power when temp is decreasing (102 °C). (Powerpac: Shut down at 102 °C).
Oil temperature:	Decreasing to 50% power, increasing to full power when temp is decreasing 122-132 °C). (Powerpac: Shut down at 122-132 °C).
Oil pressure:	Shut down (2,2 / 0,4 Bar at 1500/Idle).
Water in fuel:	Error code, no decrease
Low fuel pressure:	Error code, no decrease

Those engines have some built-in system functionality. Engine starter control is a way of achieving safe starting procedure. Further, the engines are prepared for electronic transmissions. [9]

3.2 D12 Interface

There are 3 different interfaces on the D12-industry engine, both genset and all-speed. They are as follows.

- SAE J1939 CAN-link
- CIU
- Stand alone connections

3.2.1 CAN link J1939

When using this link the Customer buys the engine as it is. No additional cables or senders are needed. What the Customer needs from VP is the Technical Regulation document 874423 for genset and 874424 for all-speed that describes what the protocol looks like (see an extract in Appendix A). With this document the Customer can design his own CAN-system to his control panel. Everything is included on the CAN-link, e.g. start, speed adjustment, alarm, gauges and diagnostic codes (flashing lamps). You can also see chapter CIU-interface 3.2.2. [5, 6]

3.2.2 CIU

If the Customer doesn't want to be connected directly on the CAN-link he can choose to have the CIU (Control Interface Unit). This device converts the CAN-protocol to analogue/digital signals and analogue/digital signals to the CAN-protocol (see picture below). The CIU's options are identical as on the CAN-link. [5, 6]

3.2.3 Stand alone connections

The stand alone connections is hard wires only, i.e. analogue and digital signals, no CAN-link available. This interface doesn't support any signals to instruments, and thus this must be installed by the Customers themselves. [5, 6]

4 Realization

To carry out this dissertation I had study and partly learn Visual Basic and Visual C++, and also how they work as programming languages. I have also done some studying learning how the CAN J1939 works as I have explained earlier.

4.1 Programming

Here I will describe the programming language I have used in this dissertation.

4.1.1 Visual Basic

Because I haven't programmed Visual Basic earlier, I had to put a lot of work in learning how that programming language works. However it wasn't foreign to learn a new programming language, since I have some knowledge in other programming languages like C/C++ and Fortran. The programming language is of course separated in syntax but they are often built on the same principals and thinking.

I started with studying how Visual Basic works generally. After learning Basics and realizing most imported issues, I converted the code into Visual Basic for PocketPC or really eMbedden Visual Basic that is a better explanation. At the beginning it seemed to be equal, with some small differences. But very soon I discovered that eMbedded Visual Basic was a slim version of Visual Basic with many functions missing (which I realized I had to use).

After three weeks time we decided to change programming language to eMbedded Visual C++ instead, because there were no drivers or modules from the hardware manufacture supporting Visual Basic.

4.1.2 Visual C++

Embedded Visual C++ is also included in Microsoft eMbedded Visual Tools software package. I have programmed C/C++ before but I discovered that Visual C++ was a bit different. So for that reason I had to spend some time learning this programming language too. After a quick look at Visual C++ I can say that it is very different towards programming Visual Basic. Visual C++ is much more developed for eMbedded systems than Visual Basic. Tobias Rasmusson from Newmad Technologies AB helped me get going with Visual C++ when we decided to use that instead. He gave me some tip and advice that I could avoid common mistakes.

4.2 PocketPC programming

From the beginning it was intended that the program should be designed in Visual Basic, but when the hardware manufacture of the CAN card interface didn't support that, we decided to program in Visual C++ instead. Another idea was to write a dll I C++ that should be called from inside Visual Basic, but that idea was discarded because of recommendations from experienced programmers. They considered writing such a dll is a dissertation in itself and therefore it will take too much time because it is so much new to learn.

4.2.1 Database for Signals

To make this analyze tool fairly prepared for different language, it is suitable to make a database for signal names and signal identifiers. In that database you can put the language alternatives in a table. This makes it simpler for future upgrades, as you don't

need to edit more than one file. The database can for example consist of two linked tables, one table for message identifier and one for language identifier, both tables contains numbers identifying message and language. Every signal message is allocates one number, which becomes primary identifier. Then you can have a number code for every language, Swedish is for example 01 and English 03 according to previous used language codes used at Volvo Penta. Unfortunately this hasn't been implemented in the software as planned, but hopefully the final software will have that function. At least the application is prepared for it.

4.2.2 Reading CAN J1939

To read the CAN bus the PCMCIA card from Kvaser is used as described earlier in this report. There software or implementation library is called CANLIB and is used for reading from the CAN card.

Kvasers CANLIB API consists exclusively of functions. Most functions return a status code, which is a negative number if the call failed and a positive number if call succeeded. The drivers for Windows CE have the same API as the "ordinary" Windows versions except 7 functions.

The first call that must be performed initialize the function library. After that is done a channel must be opened. Next that needs to be done is to set parameters bus speed. When all this is done with positive status, you can start reading information on the bus.

After writing or reading bus you need to go of the bus and close the channel for clean up the circuit. [2]

4.2.3 The Program (Can Bus Diagnostic Tool for PocketPC)

The program is built with several modules. To start with there is a main dialog that doesn't contain any direct functions, just calls to open other dialogs. The "About" dialog doesn't contain any functions either. However, if you open the "Settings" dialog the function `getChannel` is called, and then the default setting for the channel at the CAN card will be set to channel zero. You can choose whether you want to use channel one or zero and your selection will be saved with help of function `setChannel`. If function is cancelled `setChannel` will not save the value.

The dialog "canRead" calls the CANLIB library for the CAN card, initialize it and opens the channel you selected for reading. Next you have to decide whether you want to start reading CAN bus through selected channel or open the "viewSettings" dialog where you choose what messages you want to display. The "viewSettings" dialog is a so called modeless dialog which means that the user can supply information, in this case what messages to viewed, and return to the previous task without closing the dialog box. It is possible to start the reading without choosing a message to view, but then it doesn't write something either.

The viewing of the messages is put through a filter, which decides whether the message will be displayed or not. Not selected messages are discarded, which saves CPU usage (the importance of this became obvious to me during the development). When you chose to close the “canRead” dialog it also closes the CAN card channel and cleans up the buffers it may have used.

4.2.4 GUI (Graphic User Interface)

The Graphic User Interface is done as simple as possible. The program has one main dialog, where you can choose between “CanBusRead”, “Settings” or “About”. The “About” dialog just shows information about the program and then it will return to the main dialog. If you choose “Settings” you get to a dialog where you can choose what channel you want to use on the CAN card. There is also an option to choose what language you want to use (not in use). If you choose “CanBusRead” you will get to a dialog where the signals will be shown. You must choose what message you want to display before starting reading. As default there are no messages displayed and you can only show one message at the time at this moment. When one message is chosen to be displayed, you go back to the “CanBusRead” dialog and there the signals will be displayed. As said, this is a very simple design. This is just a simple description and illustrations (screenshots) can be found in Appendix C.

When you build a Graphic User Interface for a PDA there are some limitations. The screen is not “unlimited”, it is a very small space compared with an ordinary computer. The space you can use is 240x320 pixels, as you can se in an example in Figure 4.2.4-1.

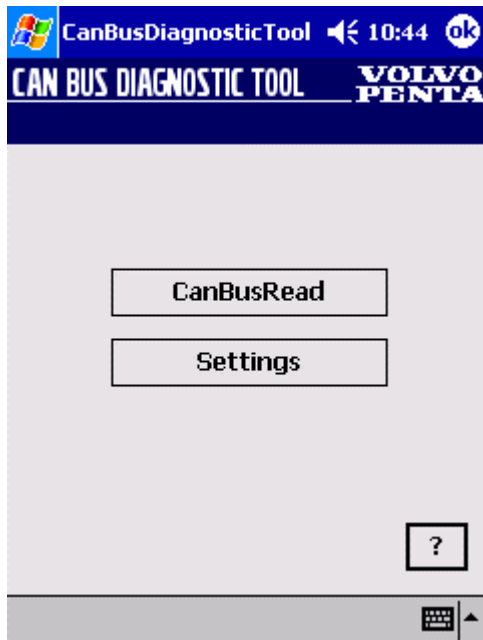


Figure 4.2.4-1 – Example of screen size

4.2.5 First version

The first version of the program that I wrote worked but it was really very slow. That was because all data that was read was treated at bit level. This meant that every message that was sent out had a short delay with approximately 4 seconds. This resulted in a search for possible solutions to this problem. First I thought all messages were buffered before being displayed, why I tried to empty the buffer. Soon I realized that this wasn't a solution to the problem, instead the idea came up to filter messages before they were processed at bit level. This is how the second version is constructed.

4.2.6 Second version

The second version has precisely the same bit filtering for viewing signals, because that worked without any problem. The difference is, here I have a filter that doesn't care about the messages I don't want to show, why they are deleted immediately. This version also has another thread then the first version, and hopefully this made the messages appear faster. This should have the consequence that the whole program would work faster. Unfortunately I haven't been able to test more then two messages properly, and that was the EEC1 with the engine speed and the VP Status with for example Throttle position. These messages were now updated much faster then before, approximately with a delay less than one second. But there are still some bugs. Updates to screen works for about 30 seconds or up to a minute and then it stops.

4.2.7 Third version

After contact with Martin Henriksson at Kvaser AB, that is the hardware manufacture, I got some tips on how to get rid of the update problem and the delay. He told me, that I could try to take the "thread" away and replace it with the built-in canNotify function that is available in their drivers and CANlib library. This function sends a notification every time a message is received from the CAN data link and in this way the software don't crash the application on the handheld computer. Because of this the delay disappeared and the update to the display worked without faults. By that the software is working very good and accordingly to targets.

4.3 Connection to Volvo Pentas Engine cable harness

Mission objective was to supply with cables for connecting to the engine's cable harness. There already exists one cable for that at Volvo Penta, but that cable doesn't have the right connector. The existing cable also needs a new connector for the tool or a new cable. Volvo Penta will later decide how to solve this problem.

5 Results

As described in the objectives the result should be a prototype software that reads the CAN bus at Volvo Pentas engines with EDC and EMS. This software is called “Can Bus Diagnostic Tool for PocketPC” and is a software that reads CAN bus and displays the result of the signals on the screen at the PDA. How the GUI (Graphic User Interface) is constructed you can see if you study the snapshots that are made from the PDA. You find them together with the block diagram in Appendix C. According to me results are very positive. It is really possible to design a tool for this purpose, for a PDA and I am really happy to have got so far in a short period of time.

6 Conclusions

As conclusion, you can say this result agrees with objectives and what was proposed in the beginning of my dissertation. That is a prototype software to read the CAN SAE J1939 data link on a PDA. The J1939 data link is used for control and for monitoring data.

6.1 Analyses of the result

As conclusion you can say that I have got three different types of software prototypes, which all work but with small differences in performance. The first one reads all the defined messages and manages them all even if only the ones you have chosen are displayed. That version works with a delay of approximately 4 seconds. I think that delay depends on the program managing all the messages and at the same time. It is a proper working version, but maybe not the final solution since the delay cannot really be accepted.

The second version was thought to be faster, and the delays would be eliminated. However it looks like it has become a bit faster, there are some bugs instead. The update of the screen doesn't work properly at all times. When testing for example the engine speed it works for approximately half a minute and then it stops updating the screen. The software however reads CAN data for the whole time and sends it further for screen update, but there is no update to be seen on the display. This software is a bit faster than the first one regarding the EEC1 and VP Status messages that was the best message to study in this case.

The third version works very good and all the delays and update problems are solved. However there is still some small bugs but now the program works very good.

A result or conclusion of this dissertation is that I can say it's possible to design a software like this, that works quite excellent and with more functions.

6.2 Recommendations for continues work

If Volvo Penta chooses to continue developing the software I have developed during this dissertation, there are some things to elucidate. For one thing, there must be much more testing to get rid of the small bugs there is left. Second thing would be to implement more functions, for example a log function to log the messages to a file.

There are some questions, referring to Anders Näzell, about how a Compaq iPAQ manage damp, shock or a situation where you drop it. This question should be answered by Compaq, but with normal handling the selected PDA will suite its propose. A solution to this can be to put the PDA into one of many existing casings on the market. Perhaps it is also possible to have an agreement with Compaq that covers damage and other things related to using the PDA in the aftermarket as an engineer's tool.

7 Reference

7.1 Books and datasheet

- 1 Kvaser AB Sweden, 1998, CAN
- 2 Kvaser AB Sweden, 2001, CANLIB SDK User's Guide Version 3.0
- 3 Deitel, H. M., Deitel, P. J., 1997, C++ How to Program
- 4 SAE International, 1996-05, SAE J1939/71
- 5 AB Volvo Penta, 2001-02-01, Electrical Interface, TWD1240VE
- 6 AB Volvo Penta, 2001-01-30, Electrical Interface, TAD1240, 1241, 1242GE
- 7 AB Volvo Penta, 2000-12-14, Technical Regulation; no. 874423, Interface specification for TAD124xGE
- 8 AB Volvo Penta, 2000-12-20, Technical Regulation; no 8774424, Interface specification for TWD1240VE
- 9 AB Volvo Penta, Document made by Anders Näzell

7.2 Websites

- 11 Kvaser AB, <http://www.kvaser.com>, 2002-05-23

Appendix

A Volvo Penta Specific messages	13 pages
B Messages from standard SAE J1939	18 pages
C Overview of the program.....	2 pages

Extract from Technical Regulation, Interface specification for TAD124xGE and TWD1240VE.

Volvo Penta specific messages

TAD124xGE

J1939 Proprietary frames

VP Engine Industry

Data Length	8
Transmission update period	50 ms
Data page	0
PDU format	255
PDU specific	71
Priority	3
Source address	0
PGN Number (calculated)	65351
Identifier (calculated)	218056448
Description	Engine information

Byte	Bit	Signal
1	1-2	Not used
1	3-4	Running indication
1	5-6	Overspeed alarm
1	7-8	Oil pressure alarm
2	1-2	Oil temperature alarm
2	3-4	Coolant temperature alarm
2	5-6	Coolant level alarm
2	7-8	Charge alarm
3	1-2	Buzzer
3	3-8	Not used
4	1-2	General lamptest
4	3-4	Buzzertest / Lamptest
4	5-8	Not used
5	1-2	EMS Diagnose Yellow lamp
5	3-4	EMS Diagnose Red lamp
5	5-8	Not used
6	1-24	Not used

Running indication

The running status of the engine

Value	Description
0	Stopped
1	Running
2	Reserved
3	Not available

Type: Measured
Format: Numeric

Overspeed alarm

The status of the (virtual) overspeed alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured
Format: Numeric

Oil pressure alarm

The status of the (virtual) oil pressure alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured
Format: Numeric

Oil temperature alarm

The status of the (virtual) oil temperature alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured
Format: Numeric

Coolant temperature alarm

The status of the (virtual) coolant temperature alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured

Format: Numeric

Coolant level alarm

The status of the coolant level alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured

Format: Numeric

Charge alarm

The status of the (virtual) charge alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured

Format: Numeric

Buzzer

Controls the buzzer

Value	Description
0	Inactive
1	Active
2	Reserved
3	Not available

Type: Status

Format: Numeric

General lamptest

Controls the general lamptest

Value	Description
0	Inactive
1	Active
2	Reserved
3	Not available

Type: Status

Format: Numeric

Buzzertest / Lamptest

Controls the buzzertest / lamptest

Value	Description
0	Inactive
1	Active
2	Reserved
3	Not available

Type: Status

Format: Numeric

EMS Diagnose Yellow lamp

The status of the yellow diagnose lamp of the EMS (Mirror of PID 44, J1587)

Value	Description
0	Inactive
1	Active
2	Error condition
3	Not available

Type: Status

Format: Numeric

EMS Diagnose Red lamp

The status of the red diagnose lamp of the EMS (Mirror of PID 44, J1587)

Value	Description
0	Inactive
1	Active
2	Error condition
3	Not available

Type: Status

Format: Numeric

VP Status	
Data Length	8
Transmission update period	20 ms
Data page	0
PDU format	255
PDU specific	70
Priority	3
Source address	17
PGN Number (calculated)	65350
Identifier (calculated)	218056209
Description	CIU status information

Byte	Bit	Signal
1	1-2	Start request
1	3-4	Stop request
1	5-6	Not used
1	7-8	Idle speed select
2	1-2	Frequency select
2	3-4	Diagnosis switch
2	5-6	Preheat request, not used in genset spec.
2	7-8	Not used
3	1-16	Accelerator pedal position
5	1-32	Not used

Start request

Start request

Value	Description
0	Inactive
1	Active
2	Error indication
3	Not available

Type: Status

Format: Numeric

Stop request

Stop request

Value	Description
0	Inactive
1	Active
2	Error indication
3	Not available

Type: Status

Format: Numeric

Idle speed select

Indicates if the engine shall operate at idle speed or at running speed.

Value	Description
0	Normal running speed request
1	Idle speed request
2	Error indication
3	Not available

Type: Status

Format: Numeric

Frequency select

Indicates if the engine shall operate at primary engine speed or secondary engine speed rpm.

Value	Description
0	Primary engine speed request
1	Secondary engine speed request
2	Error indication
3	Not available

Type: Status

Format: Numeric

Diagnosis switch

Status of the Diagnosis switch

Value	Description
0	Inactive
1	Active
2	Error indication
3	Not available

Type: Status

Format: Numeric

Accelerator pedal position

The accelerator pedal position

Resolution: 0.097752 %/bit (1/1023)

Offset: 0

Size: 2 bytes, bit 0-9

Data range: 0 to 100

Unit: %

Error indication: FEFF

Type: Measured

Format: Numeric

TWD1240VE

J1939 Proprietary frames

VP Engine Industry

Data Length	8
Transmission update period	50 ms
Data page	0
PDU format	255
PDU specific	71
Priority	3
Source address	0
PGN Number (calculated)	65351
Identifier (calculated)	218056448
Description	Engine information

Byte	Bit	Signal
1	1-2	Preheat indication
1	3-4	Running indication
1	5-6	Overspeed alarm
1	7-8	Oil pressure alarm
2	1-2	Oil temperature alarm
2	3-4	Coolant temperature alarm
2	5-6	Coolant level alarm
2	7-8	Charge alarm
3	1-2	Buzzer
3	3-8	Not used
4	1-2	General lamptest
4	3-4	Buzzertest / Lamptest
4	5-8	Not used
5	1-2	EMS Diagnose Yellow lamp
5	3-4	EMS Diagnose Red lamp
5	5-8	Not used
6	1-24	Not used

Preheat indication

The status of the preheat relay

Value	Description
0	Inactive
1	Active
2	Error state
3	Not available

Type: Measured

Format: Numeric

Running indication

The running status of the engine

Value	Description
0	Stopped
1	Running
2	Reserved
3	Not available

Type: Measured
Format: Numeric

Overspeed alarm

The status of the (virtual) overspeed alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured
Format: Numeric

Oil pressure alarm

The status of the (virtual) oil pressure alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured
Format: Numeric

Oil temperature alarm

The status of the (virtual) oil temperature alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured
Format: Numeric

Coolant temperature alarm

The status of the (virtual) coolant temperature alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured

Format: Numeric

Coolant level alarm

The status of the coolant level alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured

Format: Numeric

Charge alarm

The status of the (virtual) charge alarm switch

Value	Description
0	Inactive
1	Active
2	Switch error
3	Not available

Type: Measured

Format: Numeric

Buzzer

Controls the buzzer

Value	Description
0	Inactive
1	Active
2	Reserved
3	Not available

Type: Status

Format: Numeric

General lamptest

Controls the general lamptest

Value	Description
0	Inactive
1	Active
2	Reserved
3	Not available

Type: Status

Format: Numeric

Buzzertest / Lamptest

Controls the buzzertest / lamptest

Value	Description
0	Inactive
1	Active
2	Reserved
3	Not available

Type: Status

Format: Numeric

EMS Diagnose Yellow lamp

The status of the yellow diagnose lamp of the EMS (Mirror of PID 44, J1587)

Value	Description
0	Inactive
1	Active
2	Error condition
3	Not available

Type: Status

Format: Numeric

EMS Diagnose Red lamp

The status of the red diagnose lamp of the EMS (Mirror of PID 44, J1587)

Value	Description
0	Inactive
1	Active
2	Error condition
3	Not available

Type: Status

Format: Numeric

VP Status	
Data Length	8
Transmission update period	20 ms
Data page	0
PDU format	255
PDU specific	70
Priority	3
Source address	17
PGN Number (calculated)	65350
Identifier (calculated)	218056209
Description	CIU status information

Byte	Bit	Signal
1	1-2	Start request
1	3-4	Stop request
1	5-6	Governor mode request
1	7-8	Idle speed select
2	1-2	Frequency select
2	3-4	Diagnosis switch
2	5-6	Preheat request
2	7-8	Not used
3	1-16	Accelerator pedal position
5	1-32	Not used

Start request

Start request

Value	Description
0	Inactive
1	Active
2	Error indication
3	Not available

Type: Status

Format: Numeric

Stop request

Stop request

Value	Description
0	Inactive
1	Active
2	Error indication
3	Not available

Type: Status

Format: Numeric

Governor mode request

Indicates if the engine shall operate in Engine speed mode or Droop mode.

Value	Description
0	Engine speed mode request
1	Droop mode request
2	Error indication
3	Not available

Type: Status

Format: Numeric

Idle speed select

Indicates if the engine shall operate at idel speed or at running speed

Value	Description
0	Normal running speed request
1	Idle speed request
2	Error indication
3	Not available

Type: Status

Format: Numeric

Frequency select

Indicates if the engine shall operate at primary engine speed or secondary engine speed rpm.

Value	Description
0	Primary engine speed request
1	Secondary engine speed request
2	Error indication
3	Not available

Type: Status

Format: Numeric

Diagnosis switch

Status of the Diagnosis switch

Value	Description
0	Inactive
1	Active
2	Error indication
3	Not available

Type: Status

Format: Numeric

Preheat request

Status of the Preheat request.

Value	Description
0	Inactive
1	Active
2	Error indication
3	Not available

Type: Status

Format: Numeric

Accelerator pedal position

The accelerator pedal position

Resolution: 0.097752 %/bit (1/1023)

Offset: 0

Size: 2 bytes, bit 0-9

Data range: 0 to 100

Unit: %

Error indication: FEFF

Type: Measured

Format: Numeric

Extract from standard document SAE J1939/71

SAE International SAE J1939/71, Surface Vehicle Standard – Vehicle Application Layer, Issued 1994-08, Revised 1996-05

Messages from standard SAE J1939

1 ELECTRONIC ENGINE CONTROLLER #2: EEC2

Transmission repetition rate: 50 ms
 Data length: 8 bytes
 Data page: 0
 PDU format: 240
 PDU specific: 3
 Default priority: 3
 Parameter group number: 61,443 (00F00316)

Byte: 1 Status_EEC2 Bit: 8-5 Not defined
 4-3 AP kickdown switch 1.1
 2-1 AP low idle switch 1.2
 2 Accelerator pedal (AP) position 1.3
 3 Percent load at current speed 1.4
 4-8 Not defined

1.1 *Accelerator Pedal Kickdown Switch*—Switch signal which indicates whether the accelerator pedal kickdown switch is opened or closed. The kickdown switch is defined in SAE J1843.

00 - Kickdown passive
 01 - Kickdown active
 Type: Measured
 Suspect Parameter Number: 559

1.2 *Accelerator Pedal Low Idle Switch*—Switch signal which indicates whether the accelerator pedal low idle switch is opened or closed. The low idle switch is defined in SAE J1843.

00 - Accelerator pedal not in low idle condition
 01 - Accelerator pedal in low idle condition
 Type: Measured
 Suspect Parameter Number: 558

1.3 *Accelerator Pedal Position*—The ratio of actual accelerator pedal position to maximum pedal position. Although it is used as an input to determine powertrain demand, it also provides anticipatory information to transmission and ASR algorithms about driver actions.

Data Length: 1 byte
 Resolution: 0.4%/bit gain, 0% offset
 Data Range: 0 to 100%
 Type: Measured
 Suspect Parameter Number: 91

- 1.4 *Percent Load at Current Speed*—The ratio of actual engine percent torque (indicated) to maximum indicated torque available at the current engine speed, clipped to zero torque during engine braking.
 Data Length: 1 byte
 Resolution: 1%/bit gain, 0% offset
 Range: 0 to 125%
 Type: Status
 Suspect Parameter Number: 92

2 ELECTRONIC ENGINE CONTROLLER #1: EEC1

Transmission repetition rate: engine speed dependent (see 3.1.6.2)
 Data length: 8 bytes
 Data page: 0
 PDU format: 240
 PDU specific: 4
 Default priority: 3
 Parameter group number: 61,444 (00F00416)

Byte: 1 Status_EEC1 Bit: 8-5 Not defined
 4-1 Engine/retarder torque mode 2.1
 2 Driver's demand engine - percent torque 2.2
 3 Actual engine - percent torque 2.3
 4,5 Engine speed 2.4
 6-8 Not defined

- 2.1 *Engine and Retarder Torque Mode (4 bits)*—State signal which indicates which engine or retarder torque mode is currently generating, limiting or controlling the torque. Note that the modes are not in prioritized order. Not all modes may be relevant for a given device. Some devices may not implement all functions. For typical priorities refer to Figures 1 and 2 for engine control and Tables 2 to 4 for retarder control. The data type of this parameter is measured.

Mode 00002 means “No request”: engine torque may range from 0 to full load only due to low idle governor output; retarder torque = 0 (no braking).

Modes 00012 to 11102 indicate that there is either a torque request or the identified function is currently controlling the engine/retarder: engine/retarder torque may range from 0 (no fueling/no braking) to the upper limit.

Suspect Parameter Number: Engine Mode: 899
 Retarder Mode: 900

Table 1 - Engine/Retarder Torque Modes

Bit States	Engine/Retarder Torque Mode
0000	Low idle governor/no request (default mode)
0001	Accelerator pedal/operator selection
0010	Cruise control
0011	PTO governor
0100	Road speed governor
0101	ASR control
0110	Transmission control
0111	ABS control
1000	Torque limiting
1001	High speed governor
1010	Braking system
1011	not defined
1100	not defined
1101	not defined
1110	Other
1111	Not available

- 2.1.1 Low Idle Governor/No request (Default mode)—This mode is active if the accelerator pedal (not necessarily the torque output of the driver input, see Figure 1 and Figure 2) is zero. This is the default mode. At low speed the low idle governor may be active while at higher speed it is zero.
- 2.1.2 Accelerator Pedal—This mode is active if the accelerator pedal position is active (being followed). This mode is active for the retarder if it is turned on by the operator. Note that it may be disabled by the accelerator pedal or clutch switches (operator selection).
- 2.1.3 Cruise Control—This mode is active if cruise control is active and greater than the accelerator pedal request.
- 2.1.4 PTO Governor—This mode is active if the PTO governor is active.
- 2.1.5 Road Speed Governing—Indicates that road speed governing is active and limiting torque.
- 2.1.6 ASR Control—Indicates that the ASR command is active (Speed, Torque, or Speed/Torque Limit Control).
- 2.1.7 Transmission Control—Indicates that the transmission command is active (Speed, Torque, or Speed/Torque Limit Control).
- 2.1.8 ABS Control—Indicates that the ABS is controlling torque.
- 2.1.9 Torque Limiting—This mode is active if the demanded or commanded engine torque is limited by internal logic due to full load, smoke and/or emissions control, engine protection and/or other factors. A reduced torque limit may be

necessary for engine protection if the engine temperature is too high or a sensor fails (speed, timing, or boost pressure), as examples.

- 2.1.10 High Speed Governor—his mode is active if the engine is controlled by the high speed governor due to normal operation.
- 2.1.11 Brake System (Electronic)—This indicates that the brake pedal is controlling the torque. Note that this may include enabling of the retarder when the brake pedal is depressed (touched).
Note that if there is a request to the retarder but operating conditions do not allow braking, this situation will be reflected by the Percent Retarder Torque = 0 when broadcast.
- 2.1.12 Other—Torque control by a type of device which is different than those defined in 2.1.1 through 2.1.11.
- 2.2 *Driver's Demand Engine - Percent Torque*—The requested torque output of the engine by the driver. It is based on input from the following requestors external to the powertrain: operator (via the accelerator pedal), cruise control and/or road speed limit governor. Dynamic commands from internal powertrain functions such as smoke control, low- and high-speed engine governing; ASR and shift control are excluded from this calculation. The data is transmitted in indicated torque as a percent of the indicated peak engine torque. See 4.3.17 in standard document for the engine configuration message. Several status bits are defined separately to indicate the request which is currently being honored. This parameter may be used for shift scheduling.
Data Length: 1 byte
Resolution: 1%/bit gain, -125% offset (00 = -125%, 125 = 0%, 250 = +125%)
Data Range: -125 to 125%
Operating Range: 0 to 125%
Type: Measured
Suspect Parameter Number: 512

Figure 1 and Figure 2 show two typical torque calculations in an engine controller. On the left side of the figures there are single engine controller functions. The output torque signals of these functions are connected in the manner shown. The result is the actual engine percent torque which is realized by the engine.

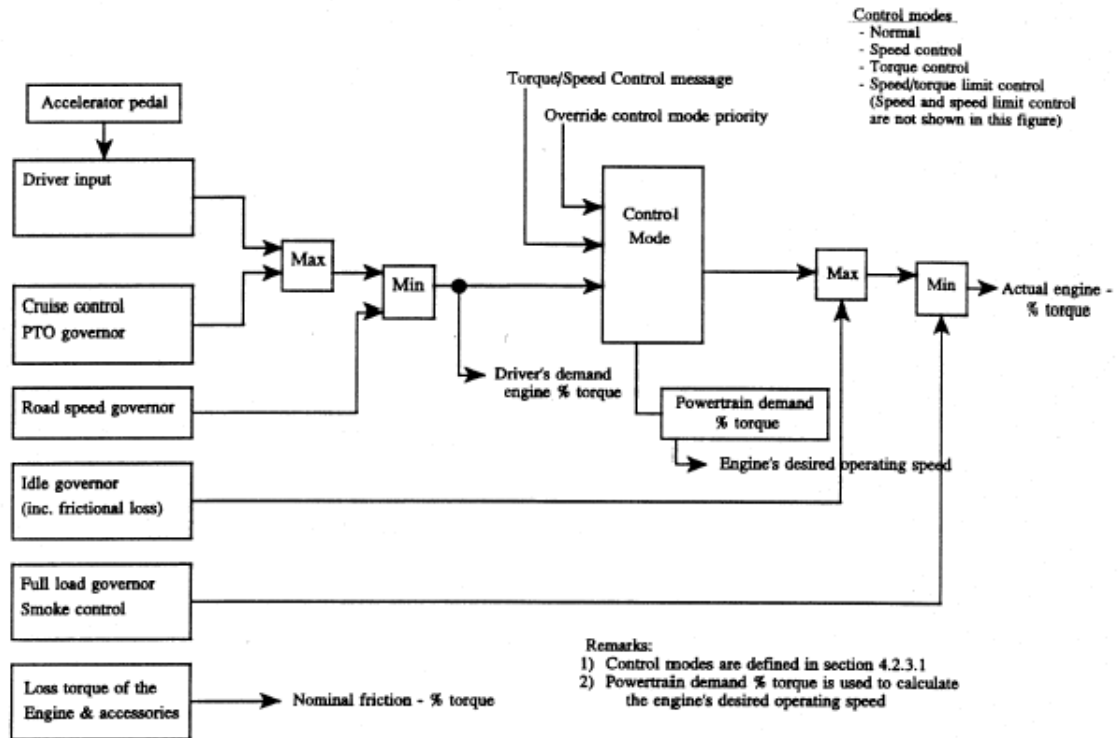


Figure 1 - Torque commands and calculations when a "maximum selection for low idle" technique is used

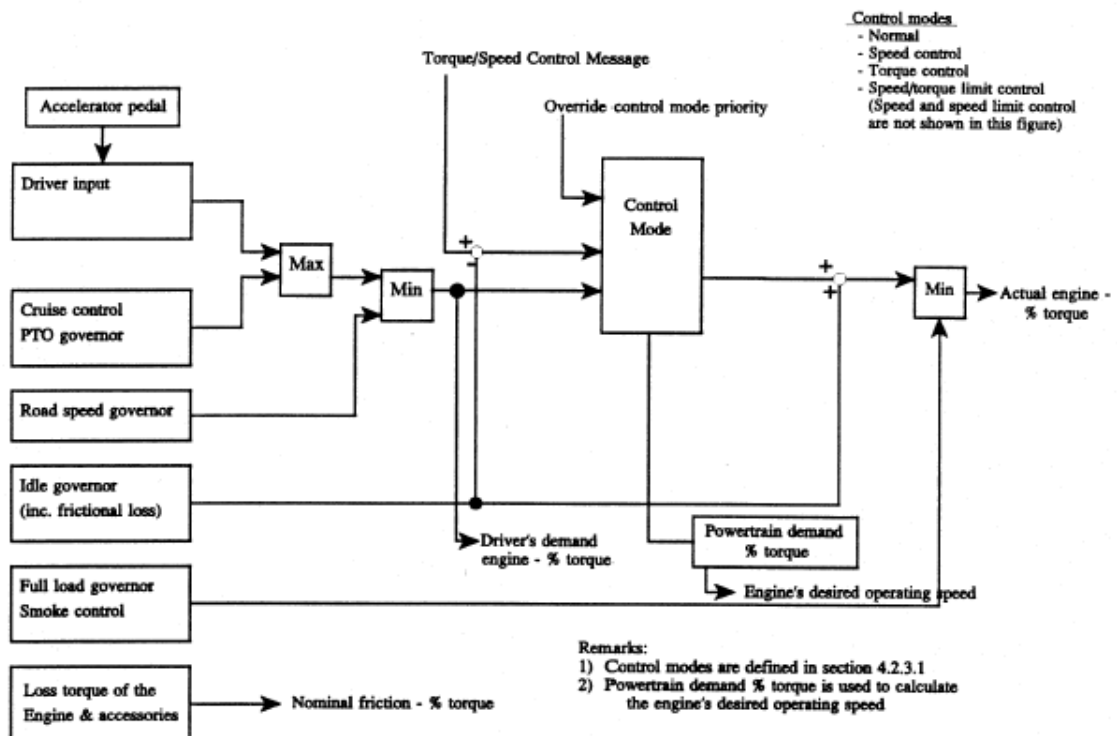


Figure 2 - Torque commands and calculations when a "summation with low idle" technique is used

On top of the figures, external torque commands (e.g., traction and transmission control) can control the engine. These commands can influence the engine torque by four control modes. Four engine internal signals are transmitted to the network:

- a. Driver's demand engine - percent torque
- b. Actual engine - percent torque
- c. Nominal friction - percent torque
- d. Engine's desired operating speed

The difference between Figure 1 and Figure 2 is the connection of the idle governor output to the torque calculation. In Figure 1 there is a maximum selection, while in Figure 2 a summation is used. The summation method needs a subtraction point for each external command input because the starting point of an ASR or a shift operation should be the present actual engine - percent torque value. As the actual engine - percent torque signal contains the idle governor output and the external commands are compared with the driver's demand engine - percent torque or the powertrain demand which don't contain the idle governor output, the external commands must be subtracted by the idle governor output to get the correct signals for comparison. The advantage of the maximum selection (Figure 1) is that no other speed controller can work parallel to the idle governor. This allows for a better optimization of the different speed control loops. The advantage of the summation method (Figure 2) is that changes of the idle governor output influence the engine directly (no dead zones exist).

- 2.3 *Actual Engine - Percent Torque*—The calculated output torque of the engine. The data is transmitted in indicated torque as a percent of peak engine torque. The engine percent torque value will not be less than zero and it includes the torque developed in the cylinders required to overcome friction as described in 4.2.1.3 standard document.
Data Length: 1 byte
Resolution: 1%/bit gain, -125% offset
Data Range: -125 to 125%
Operating Range: 0 to 125%
Type: Measured
Suspect Parameter Number: 513
- 2.4 *Engine Speed*—Actual engine speed which is calculated over a minimum crankshaft angle of 720 degrees divided by the number of cylinders.
Data Length: 2 bytes
Resolution: 0.125 rpm/bit gain, 0 rpm offset (upper byte resolution = 32 rpm/bit)
Data Range: 0 to 8031.875 rpm
Type: Measured
Suspect Parameter Number: 190

3 ENGINE TEMPERATURE: ET

Transmission repetition rate: 1 s
 Data length: 8 bytes
 Data page: 0
 PDU format: 254
 PDU specific: 238
 Default priority: 6
 Parameter group number: 65,262 (00FEEE16)

Byte: 1 Engine coolant temperature 3.1
 2 Fuel temperature 3.2
 3,4 Engine oil temperature 3.3
 5,6 Turbo oil temperature 3.4
 7 Engine intercooler temperature 3.5
 8 Not defined

3.1 *Engine Coolant Temperature*—Temperature of liquid found in engine cooling system.

Data Length: 1 byte
 Resolution: 1 °C/bit gain, -40 °C offset
 Data Range: -40 to +210 °C (-40 to 410 °F)
 Type: Measured
 Suspect Parameter Number: 110

3.2 *Fuel Temperature*—Temperature of fuel entering injectors.

Data Length: 1 byte
 Resolution: 1 °C/bit gain, -40 °C offset
 Data Range: -40 to +210 °C (-40 to 410 °F)
 Type: Measured
 Suspect Parameter Number: 174

3.3 *Engine Oil Temperature*—Temperature of the engine lubricant.

Data Length: 2 bytes
 Resolution: 0.03125 °C/bit gain, -273 °C offset
 Data Range: -273 to +1735.0 °C (-459.4 to 3155.0 °F)
 Type: Measured
 Suspect Parameter Number: 175

3.4 *Turbo Oil Temperature*—Temperature of the turbocharger lubricant.

Data Length: 2 bytes
 Resolution: 0.03125 °C/bit gain, -273 °C offset
 Data Range: -273 to +1735.0 °C (-459.4 to 3155.0 °F)
 Type: Measured
 Suspect Parameter Number: 176

3.5 *Engine Intercooler Temperature*—Temperature of liquid found in the intercooler located after the turbocharger.

Data Length: 1 byte
 Resolution: 1 °C/bit gain, -40 °C offset
 Data Range: -40 to +210 °C (-40 to 410 °F)

Type: Measured
Suspect Parameter Number: 52

4 ENGINE FLUID LEVEL/PRESSURE: EFLP

Transmission repetition rate: 0.5 s
Data length: 8 bytes
Data page: 0
PDU format: 254
PDU specific: 239
Default priority: 6
Parameter group number: 65,263 (00FEEF16)

Byte:	1 Fuel delivery pressure	4.1
	2 Not defined	
	3 Engine oil level	4.2
	4 Engine oil pressure	4.3
	5,6 Crankcase pressure	4.4
	7 Coolant pressure	4.5
	8 Coolant level	4.6

4.1 *Fuel Delivery Pressure*—Gage pressure of fuel in system as delivered from supply pump to the injection pump.

Data Length: 1 byte
Resolution: 4 kPa/bit gain, 0 kPa offset
Data Range: 0 to +1000 kPa (0 to 145 psi)
Type: Measured
Suspect Parameter Number: 94

4.2 *Engine Oil Level*—Ratio of current volume of engine sump oil to maximum required volume.

Data Length: 1 byte
Resolution: 0.4 %/bit gain, 0 % offset
Data Range: 0 to +100 %
Type: Measured
Suspect Parameter Number: 98

4.3 *Engine Oil Pressure*—Gage pressure of oil in engine lubrication system as provided by oil pump.

Data Length: 1 byte
Resolution: 4 kPa/bit gain, 0 kPa offset
Data Range: 0 to +1000 kPa (0 to 145 psi)
Type: Measured
Suspect Parameter Number: 100

4.4 *Crankcase Pressure*—Gage pressure inside engine crankcase.

Data Length: 2 bytes
Resolution: 7.8125 x 10⁻³ kPa/bit gain (1/128 kPa/bit), -250 kPa offset
Data Range: -250 to +251.99 kPa (-36.259 to +36.548 lbf/in²)

Type: Measured
Suspect Parameter Number: 101

- 4.5 *Coolant Pressure*—Gage pressure of liquid found in engine cooling system.
Data Length: 1 byte
Resolution: 2 kPa/bit gain, 0 kPa offset
Data Range: 0 to +500 kPa (0 to 72.5 psi)
Type: Measured
Suspect Parameter Number: 109

- 4.6 *Coolant Level*—Ratio of volume of liquid found in engine cooling system to total cooling system volume.
Data Length: 1 byte
Resolution: 0.4 %/bit gain, 0 % offset
Data Range: 0 to +100 %
Type: Measured
Suspect Parameter Number: 111

5 INLET/EXHAUST CONDITIONS: IEC

Transmission repetition rate: 0.5 s
Data length: 8 bytes
Data page: 0
PDU format: 254
PDU specific: 246
Default priority: 6
Parameter group number: 65,270 (00FEF616)

Byte:	1 Particulate trap inlet pressure	5.1
	2 Boost pressure	5.2
	3 Intake manifold temperature	5.3
	4 Air inlet pressure	5.4
	5 Air filter differential pressure	5.5
	6,7 Exhaust gas temperature	5.6
	8 Coolant filter differential pressure	5.7

- 5.1 *Particulate Trap Inlet Pressure*—Exhaust back pressure as a result of particle accumulation on filter media placed in the exhaust stream.
Data Length: 1 byte
Resolution: 0.5 kPa/bit gain, 0 kPa offset
Data Range: 0 to +125 kPa (0 to +18.1 psi)
Type: Measured
Suspect Parameter Number: 81
- 5.2 *Boost Pressure*—Gage pressure of air measured downstream on the compressor discharge side of the turbocharger.
Data Length: 1 byte
Resolution: 2 kPa/bit gain, 0 kPa offset
Data Range: 0 to +500 kPa (0 to 72.5 psi)

Type: Measured
Suspect Parameter Number: 102

- 5.3 *Intake Manifold Temperature*—Temperature of precombustion air found in intake manifold of engine air supply system.
Data Length: 1 byte
Resolution: 1 °C/bit gain, -40 °C offset
Data Range: -40 to +210 °C (-40 to 410 °F)
Type: Measured
Suspect Parameter Number: 105
- 5.4 *Air Inlet Pressure*—Absolute air pressure at inlet to intake manifold or air box.
Data Length: 1 byte
Resolution: 2 kPa/bit gain, 0 kPa offset
Data Range: 0 to +500 kPa (0 to 72.5 psi)
Type: Measured
Suspect Parameter Number: 106
- 5.5 *Air Filter Differential Pressure*—Change in engine air system pressure, measured across the filter, due to the filter and any accumulation of solid foreign matter on or in the filter.
Data Length: 1 byte
Resolution: 0.05 kPa/bit gain, 0 kPa offset
Data Range: 0 to +12.5 kPa (0 to +1.8 psi)
Type: Measured
Suspect Parameter Number: 107
- 5.6 *Exhaust Gas Temperature*—Temperature of combustion byproducts leaving the engine.
Data Length: 2 bytes
Resolution: 0.03125 °C/bit gain, -273 °C offset
Data Range: -273 to +1735.0 °C (-459.4 to 3155.0 °F)
Type: Measured
Suspect Parameter Number: 173
- 5.7 *Coolant Filter Differential Pressure*—Change in coolant pressure, measured across the filter, due to the filter and any accumulation of solid or semisolid matter on or in the filter.
Data Length: 1 byte
Resolution: 0.5 kPa/bit gain, 0 kPa offset
Data Range: 0 to +125 kPa (0 to +18.1 psi)
Type: Measured
Suspect Parameter Number: 112

6 WATER IN FUEL INDICATOR: WIFL

Transmission repetition rate: 10 s
Data length: 8 bytes
Data page: 0

PDU format: 254
PDU specific: 255
Default priority: 6
Parameter group number: 65,279 (00FEFF16)

Byte: 1 Water in fuel indicator 6.1
2-8 Not defined

6.1 *Water In Fuel Indicator*—Signal which indicates the presence of water in the fuel.
00- No
01- Yes
Type: Measured
Suspect Parameter Number: 97

7 TORQUE/SPEED CONTROL #1: TSC1

Transmission repetition rate: when active; 10 ms to the engine - 50 ms to the retarder
Data length: 8 bytes
Data page: 0
PDU format: 0
PDU specific: Destination address
Default priority: 3
Parameter group number: 0 (00000016)

Byte: 1 Control bits Bit: 8-7 Not defined
6,5 Override control mode priority 7.1
4,3 Requested speed control conditions 7.2
2,1 Override control modes 7.3
2,3 Requested speed/Speed limit 7.4
4 Requested torque/Torque limit 7.5
5-8 Not defined

NOTE—Retarder may be disabled by commanding a torque limit of 0%. For example, this permits the brake switch to enable the retarder, up to an amount selected by another device or the operator. Note that the brake switch can be treated as an operator input (mode 00012) or as a brake system input (mode 10102) relative to the active torque mode.

7.1 *Override Control Mode Priority (2 bits)*—This field is used as an input to the engine or retarder to determine the priority of the Override Control Mode received in the Torque/Speed Control message. The default is 11 (Low priority). It is not required to use the same priority during the entire override function. For example, the transmission can use priority 01 (High priority) during a shift, but can set the priority to 11 (Low priority) at the end of the shift to allow traction control to also interact with the torque limit of the engine.
The four priority levels defined are:
00- Highest priority

01- High priority
10- Medium priority
11- Low priority
Type: Status
Suspect Parameter Number: 897

- 7.1.1 Highest Priority 00—Used for situations that require immediate action by the receiving device in order to provide safe vehicle operation (i.e., braking systems). This level of priority should only be used in safety critical conditions.
 - 7.1.2 High Priority 01—Used for control situations that require prompt action in order to provide safe vehicle operation. An example is when the transmission is performing a shift and requires control of the engine in order to control driveline reengagement.
 - 7.1.3 Medium Priority 10—Used for powertrain control operations which are related to assuring that the vehicle is in a stable operating condition. An example is when the traction control system is commanding the engine in order to achieve traction stability.
 - 7.1.4 Low Priority 11—Used to indicate that the associated command desires powertrain control but is needed for function which improves the driver comfort which may be overridden by other devices. An example is cruise control or the non-critical part of a transmission shift to a new gear.
- 7.2 *Requested Speed Control Conditions (2 bits)*—This mode tells the engine control system the governor characteristics that are desired during speed control. The four characteristics defined are:
- 00 - Transient Optimized for driveline disengaged and non-lockup conditions
 - 01 - Stability Optimized for driveline disengaged and non-lockup conditions
 - 10 - Stability Optimized for driveline engaged and/or in lockup condition 1 (e.g., vehicle driveline)
 - 11 - Stability Optimized for driveline engaged and/or in lockup condition 2 (e.g., PTO driveline)
- Type: Status
Suspect Parameter Number: 696

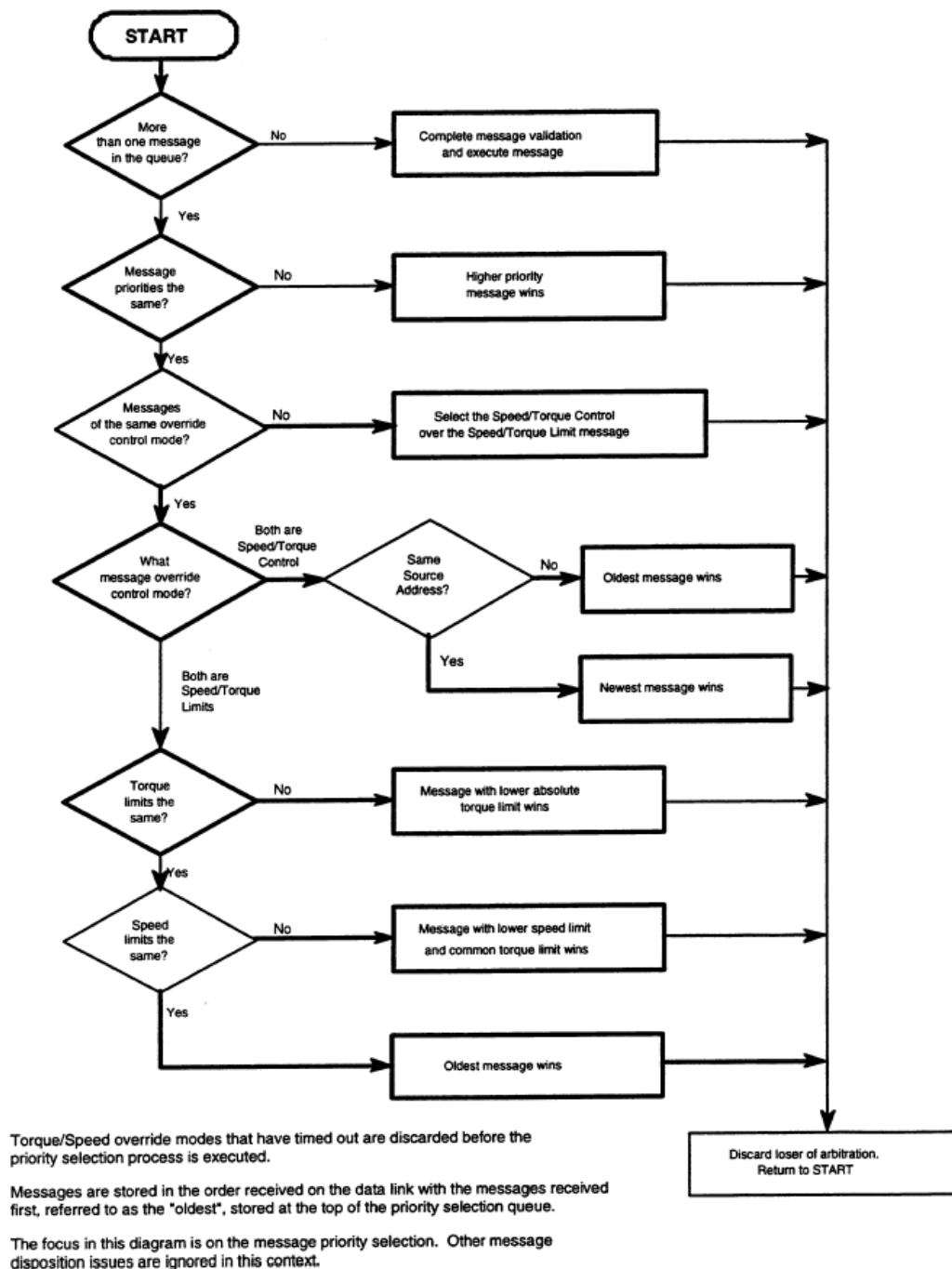


Figure 3 - Torque/Speed control priority selection logic

- 7.2.1 Speed Control Characteristic 00—This speed governor gain selection is adjusted to provide rapid transition between speed setpoints. RPM overshoot and undershoot may be greater than what is seen when the “speed control characteristic” is set to be stability optimized.
- 7.2.2 Speed Control Characteristic 01—This control condition has been optimized to minimize rpm overshoot and undershoot given an expected plant consisting of the engine and its accessory loads. This gain adjustment is not intended to compensate for driveline characteristics. This characteristic is most appropriate when no driveline is connected.

- 7.2.3 Speed Control Characteristic 10—This control condition has been optimized to minimize rpm overshoot and undershoot given a more complex plant. For instance, the more complex plant would contain the engine, its accessory loads and the driveline characteristics. As an example the driveline characteristics might include the effective spring mass relationship of pumps, tires, clutches, axles, driveshafts, and multiple gear ratios. This characteristic is most appropriate when a driveline is engaged.
- 7.2.4 Speed Control Characteristic 11—This speed control characteristic is available for applications requiring compensation for more than one driveline characteristic. It has been optimized to minimize rpm overshoot and undershoot given a more complex plant of the second variety. This more complex plant would again contain the engine, its accessory loads and a second driveline characteristic unique from the one described in speed control characteristic 10.
- 7.3 *Override Control Mode (2 bits)*—The override control mode defines which sort of command is used:
- 00 - Override disabled - Disable any existing control commanded by the source of this command.
 - 01 - Speed control - Govern speed to the included “desired speed” value.
 - 10 - Torque control - Control torque to the included “desired torque” value.
 - 11 - Speed/torque limit control - Limit speed and/or torque based on the included limit values.
- The speed limit governor is a droop governor where the speed limit value defines the speed at the maximum torque available during this operation.
- Type: Status
- Suspect Parameter Number:695

If a device wants to know whether it has access to the engine, there are several possibilities:

- a. Comparing its command with the actual engine broadcasts.
- b. Looking at command modes from other devices.
- c. Looking to the engine and retarder torque mode.

Remarks:

- a. The realization of a torque limit (minimum selection) is possible by setting the speed limit to a high value (FAFF16).
- b. The realization of a speed limit (minimum selection) is possible by setting the torque limit to a high value (FA16).
- c. Limiting the retarder torque means to limit the magnitude of the torque request. As the brake torque is represented by negative torque values, the limitation must be done by a maximum selection of the requested torque and the retarder internal torque signals.
- d. For torque increasing functions, time limits for the torque or speed value (command) and the direct modes are desirable.
- e. The selection of which device has control of the engine's speed or torque depends on the override mode priority (see 7.1) with the highest priority device gaining control. In the case of two devices with identical priority, the engine responds to speed/torque control commands over speed/torque limit commands and will act on the speed or torque commands on a first

come, first served basis. The torque limit will be a “lowest wins” selection (e.g, if one device commands 60% limit and another 80% limit, then the engine will limit torque to 60%). Figure 3 provides a flowchart of the torque/speed control priority selection logic.

- 7.4 *Requested Speed*—Parameter provided to the engine from external sources in the torque/speed control message. This is the engine speed which the engine is expected to operate at if the speed control mode is active or the engine speed which the engine is not expected to exceed if the speed limit mode is active.
Data Length: 2 bytes
Resolution: 0.125 rpm/bit gain, 0 rpm offset (upper byte resolution = 32 rpm/bit)
Data Range: 0 to 8031.875 rpm
Type: Status
Suspect Parameter Number: 898
- 7.5 *Requested Torque*—Parameter provided to the engine or retarder in the torque/speed control message for controlling or limiting the output torque. Requested torque to the engine is measured in indicated torque as a percentage of peak engine torque. This is the engine torque at which the engine is expected to operate if the torque control mode is active or the engine torque which the engine is not expected to exceed in the torque limit mode is active. Zero torque can be requested which implies zero fuel and, according to Figures 1 and 2, the engine will not be allowed to stall. The actual engine percent torque (2.3) should be zero and the engine should decelerate until the low idle governor kicks in, at which time the actual engine percent torque will be calculated as shown in Figures 1 and 2 and the engine torque mode bits (2.1) should be equal to 00002 - Low Idle Governor. Requested torque to the retarder is measured in indicated torque as a percentage of peak retarder torque. The logic used in enabling or disabling the retarder is based on the override control mode priority bits (7.1).
Data Length: 1 byte
Resolution: 1%/bit gain, -125% offset
Data Range: -125 to 125%
Operating range: 0 to 125% for engine torque requests
-125 to 0% for retarder torque requests
Type: Status
Suspect Parameter Number: 518

The driver input in Table 2 refers to the enable retarder - brake assist switch (4.2.2.11 in standard document). The driver input switch should not power the retarder control directly, otherwise it is not possible for other devices on the network to request retardation and override the driver. Engine retarders cannot be activated while the engine is being fuelled. During simultaneous network commands for both retarder enable and engine speed or torque control, the retarder commands will be restricted if the engine is not zero brake torque, and allowed once the engine achieves zero brake torque. During Electronically Controlled Braking, where the brake pedal position (or pressure) directly relates to the amount of retardation, the driver preselected level is not used to

determine the amount of retardation provided by the engine brake. A separate enable retarder - shift assist switch (4.2.2.12 in standard document) is used by the transmission to decide if network control should be used to activate the engine retarder for shift assist. When this switch is enabled, the transmission may use network commands to activate the engine retarder in order to enhance engine decel rates and improve shift control. The retarder is available for cruise control only if the retarder is first enabled by the driver using the brake assist switch.

The “previous state” when all devices become “don't care” may depend on an earlier set of conditions (i.e., a device may request retardation temporarily and then “don't care.” The retarder should then return to its “previous state.” This is typically zero retardation, but not in all cases). Multiple retarder control messages from the network are handled through the use of the override control mode priority bits described in 7.1.

Table 2 - Primary retarder - Before transmission (Engine Retarder)

Input Pedal	Input Cruise	Input Driver	Input Network	Output Retarder	
00 (ap)	--	01	11	0	ap pressed
11	00	01	11	0	cc fueling
--	--	00	11	0	driver disable (brake assist)
--	--	--	00	0	network control
01 (bp)	--	01	11	1	bp pressed ⁽¹⁾
11	01	01	11	1	cc initiated
11	11	01	11	x	previous state
--	--	--	01	1	network control

INPUTS:
00 = Requesting zero retardation = disable
01 = Requesting retardation
11 = Don't care (Control mode set to 00 = normal or no request sent)
-- = Any state

OUTPUTS:
0 = Disable retarder
1 = Enable retarder

1. Not all implementations of engine retarders use the brake pedal as input to activate the retarder.

The driver input in Table 3 refers to the switch setting (i.e., off, percent on). The driver select switch should not power the retarder control directly, otherwise it is not possible for other devices on the network to request retardation and override the driver.

The retarder is available for cruise control only if the retarder is first enabled by the driver.

The “previous state” when all devices become “don't care” may depend on an earlier set of conditions (i.e., a device may request retardation temporarily and then “don't care.” The retarder should then return to its “previous state.” This is typically zero retardation, but not in all cases).

Multiple retarder control messages from the network are handled through the use of the override control mode priority bits described in 7.1.

Table 3 - Primary retarder - Before transmission (Exhaust Brake)

Input Cruise	Input Driver	Input Network	Output Retarder	
00	01	11	0	cc fueling
--	00	11	0	driver disable
--	--	00	0	network control
--	01	11	1	driver enable
01	01	11	1	cc initiated
11	01	11	X	previous state
--	--	01	1	network control

INPUTS:
00 = Requesting zero retardation = disable
01 = Requesting retardation
11 = Don't care (Control mode set to 00 = normal or no request sent)
-- = Any state

OUTPUTS:
0 = Disable retarder
1 = Enable retarder

The driver input in Table 4 refers to the selector switch setting of the retarder (e.g., off, 20%, 40%, 50%, 80%, 100%). During Electronically Controlled Braking, where the brake pedal position (or pressure) directly relates to the amount or retardation, the driver preselected level is not used to determine the amount of retardation. Also note that cruise control or the driver can independently disable the retarder, but that the cruise control can only override the driver select switch to use the retarder if the driver selector switch is not off.

Multiple retarder control messages from the network are handled through the use of the override control mode priority bits described in 7.1

Table 4 - Secondary retarder - After transmission (Transmission/Driveline Retarders)

Input Cruise	Input Driver	Input Network	Output Retarder	
00	01	11	0	cc fueling
--	00	11	0	driver disable
--	--	00	0	network control
--	01	11	1	driver activate
01	01	11	1	cc initiated
11	01	11	x	previous state
--	--	01	1	network control
--	11	--	NA	

INPUTS:
00 = Requesting zero retardation = disable
01 = Requesting retardation
11 = Don't care (Control mode set to 00 = normal or no request sent)
-- = Any state

OUTPUTS:
0 = Disable retarder
1 = Enable retarder
NA = Not allowed condition

Extract from standard document SAE J1939/21

SAE International SAE J1939/21, Surface Vehicle Recommended Practice – Data Link Layer, Issued 1994-07, Revised 1998-07

Messages from standard SAE J1939

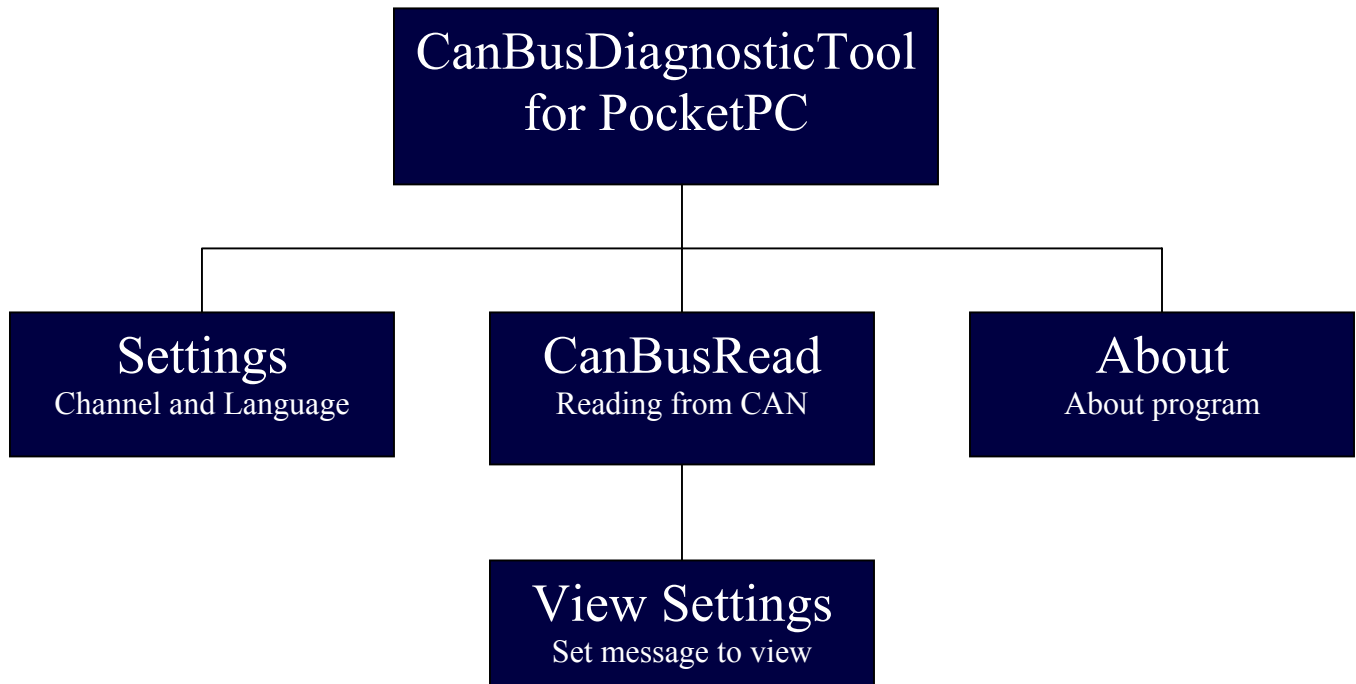
8 Broadcast Announce Message (BAM)

The TP.CM_BAM is used to inform all the nodes of the network that a large message is about to be broadcast. It defines the parameter group and the number of bytes to be sent. After TP.CM_BAM is sent, the Data Transfer Messages are sent and they contain the packetized broadcast data.

TP.CM_BAM is only transmitted by the originator.

Overview of the software

Blockdiagram of analyse program



Blockdiagram with GUI screenshots

