Investigation of a Real-Time Communication Interface Between C200 and a Computer

Eric Eklund

Luleå University of Technology
BSc Programmes in Engineering
Automotive Engineering
Department of Applied Physics and Mechanical Engineering
Division of Computer Aided Design

2009:005 HIP - ISSN: 1404-5494 - ISRN: LTU-HIP-EX--09/005--SE
Preface

This bachelor thesis is my completion of the Automotive Engineering program at Luleå University of Technology. The thesis has been performed at Scania Technical Centre, department of Fleet Management in Södertälje during winter 2009.

I would like to thank all the people who have helped me in this thesis. Peter Madsen who gave me the opportunity and confidence to work with the project, my supervisors Anders Björkman and Jan Lindman and everyone else who have been involved here at Scania.
Not to forget my supervisor at the university Ove Isaksson who gave me valuable tips about project work and report writing.
Thanks everyone!

It has been a pleasure working at Scania. For me one of my dreams has come true. Scania has a special place in my heart and that feeling has not changed rather reinforced.

Södertälje 27 Mars 2009

[Signature]

Eric Eklund
Abstract

Scania is developing a telematic unit for remote communication called Communicator 200 (C200). The purpose of the unit is to collect vehicle data and send it back to Scania. C200 is a black box hidden in the top drawer with no interaction towards the driver. Field tests are performed to verify functions in C200. Data from the tests are stored in databases in the office where it is analyzed. The loss of a driver interface makes the tests uncertain and inefficient. To simplify and make the test process more efficient an interface between C200 and the driver needs to be developed. A driver can then analyze and verify data on a computer in the vehicle during a test.

This thesis is an investigation about how such an interface shall look like. Three communication interfaces and two communication protocols have been reviewed. Interviews and discussions with engineers complemented with literature studies have been the source of knowledge. The conclusion of the thesis is to use C200 extern CAN bus together with the Can Calibration Protocol (CCP) for the communication. An interface for real-time logging of parameters would make the test process much more efficient. This will reduce resources in form of saved time and human efforts. Future work contains an implementation of CCP in C200 and an investigation about suitable PC-program to show parameters in.

Keywords: Vehicle testing, KWP2000, CCP, CAN, RS-232
Figures

Figure A: Project assignment................................................................. 8
Figure B: Communicator 200............................................................... 12
Figure C: Layer structure................................................................. 13
Figure D: C200 architecture.............................................................. 13
Figure E: Notifications and subscriptions......................................... 14
Figure F: Scania Fleet Management.................................................. 14
Figure G: Extended CAN data frame.................................................. 16
Figure H: CAN Network in a Scania vehicle...................................... 17
Figure I: VCI2 connector................................................................. 17
Figure J: Test method......................................................................... 19
Figure K: Sniffer in PuTTY................................................................. 21
Figure L: KWP2000 Message flow...................................................... 23
Figure M: KWP Manager in C200..................................................... 24
Figure N: CCP communication type.................................................. 25
Figure O: Communication flow in CCP............................................. 26
Figure P: Organization of Data Acquisition....................................... 27
Figure Q: KWP2000 communication system..................................... 29
Figure R: CCP implementation in C200........................................... 31
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASAP</td>
<td>Arbeitskreis zur Standardisierung von Applikationssystemen</td>
</tr>
<tr>
<td>BSP</td>
<td>Board Support Package</td>
</tr>
<tr>
<td>C200</td>
<td>Communicator 200</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CCP</td>
<td>Can Calibration Protocol</td>
</tr>
<tr>
<td>CRM</td>
<td>Command Return Message</td>
</tr>
<tr>
<td>CRO</td>
<td>Command Receive Object</td>
</tr>
<tr>
<td>DAQ</td>
<td>Data Acquisition</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>DTCO</td>
<td>Digital Tachograph</td>
</tr>
<tr>
<td>DTO</td>
<td>Data Transmission Object</td>
</tr>
<tr>
<td>ECU</td>
<td>Electronic Control Unit</td>
</tr>
<tr>
<td>FMP</td>
<td>Fleet Management Portal</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>KWP2000</td>
<td>Keyword Protocol 2000</td>
</tr>
<tr>
<td>OBD</td>
<td>On-board Diagnostics</td>
</tr>
<tr>
<td>ODT</td>
<td>Object Descriptor Table</td>
</tr>
<tr>
<td>OSI</td>
<td>Open System Interconnection</td>
</tr>
<tr>
<td>RS-232</td>
<td>Recommended Standard 232</td>
</tr>
<tr>
<td>SAE</td>
<td>Society of Automotive Engineers</td>
</tr>
<tr>
<td>SCOMM</td>
<td>Scania Communication Module</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Scania

Scania is one of the world leading manufacturers of trucks for heavy transports, buses as well as industrial- and marine engines. A growing part of the enterprise consists of products and services within finance and service, which will guarantee customers cost efficient transport solutions and high availability. Scania have enterprises in hundreds of countries with 35000 employees. Research and development is concentrated to Sweden and Södertälje. Manufacturing takes place in Europe and Southern America. Scanias turnover 2007 for vehicles totaled 84486 MSEK.

1.2 Background

Scania is developing a telematic unit for remote communication between a vehicle and office called Communicator 200 (C200). The purpose of the unit is to collect vehicle data and send it back to Scania. C200 is a black box hidden in the top drawer with no interaction towards the driver. Field tests are performed to verify functions in C200. The loss of a driver interface makes the tests uncertain and inefficient. Instead of getting test results directly in the vehicle the driver gets feedback from engineers in the office where data from the tests is stored. To make the test process more efficient an interface between C200 and a driver needs to be developed.

By using an interface to show test data in real-time from C200 Scania can save both money and resources. Some of the benefits with a real-time interface:

- Reduces office resources during test
- On board test verification
- On board evaluation

Figure A: Project assignment
1.3 Project assignment

The assignment of this project is to investigate how a communication between C200 and a computer in a vehicle for real-time logging shall look like. The problem is divided into four parts.

1. Prioritise parameters to show on a computer.
2. Find the best communication interface towards C200.
3. Find a suitable communication protocols.
4. Investigate PC-programs to show parameters in.

The focus of this thesis is point one to three.

1.4 Project limits

The timeline of this project have been ten weeks. The main objective of the project has been to do a solid investigation. The time was too short to build a demonstration system. Three possible communication interfaces and two communication protocols are investigated. At last an overview of conceivable PC-programs is presented.
2 Method

2.1 Original assignment

The assignment of this thesis has changed during the project process. The original assignment told to investigate suitable PC-programs to show parameters in. Some of the characteristics of the computer programs to investigate were: price, performance, maintenance, design and user-friendliness. The investigation should have end up with creating a protocol and implement the system in C200. Already after two weeks of interviews and researches I and my supervisor realized that this was not the best way of attacking the problem. The fundamental communication problem needed to be solved before start looking for a suitable PC-program. It felt like building a house, starting with the roof without a having the foundation of the house. Finding out how the communication between C200 and a computer should look like became the new working assignment as seen in headline 1.3.

2.2 Project realization

The project started with a literature study of relevant parts of the project. Information regarding Communicator 200 was the central to get knowledge about. How is the unit edified? What is the purpose of the unit? How do Scania use it today? These were questions I wanted to get an answer to.

The work began with an investigation about PC-programs to show parameters in as the original assignment told. Four divisions at Scania were selected: Break controls, Cab development, Power Train control system and Vehicle Management controls. The investigation started with a brainstorming exercise to generate interesting and intelligent questions. It followed by interviews and discussions with engineers. Simultaneous started a literature study of the PC-programs. At this point the project changed direction and work according to the new assignment began.

The first thing to investigate was parameters to supervise during tests. It was important to know the amount of parameters to be able to estimate the data transfer. Communication interfaces and protocols have different capacity, without knowing the approximate size of the data transfer incorrect conclusions can be taken about them. To find out information about the parameters discussions with engineers who been involved in the test process of C200 and a literature study of documents regarding available parameters in the unit were made. Which parameters are used today when functionality in C200 is tested? And which parameters would be interesting to supervise when a real-time interface is developed? These were questions to find an answer of.

The next step was to find a communication interface. Which way should the data take in and out from C200? That was the question for the interface problem. The parameter research had given an answer of the amount of data to be transferred. Now the task was to find a suitable interface that could deliver a safe and reliable transfer in real-time.
To realize that a study of communication interfaces in C200 were carried out. Three interfaces where identified as possible to use. The research contained a literature study and discussions with engineers. The literature study gave a theoretical background like: architecture, transfer rates and reliability. On top of that discussions with engineers gave enough information to take a decision of which interface to use.

At this time I knew which parameters to supervise and which communication interface to use. Next step was to find a suitable communication protocol. When two computers communicate with each other they need to speak the same language. That problem is solved with a protocol. The interface and protocol are closed connected to each other. To find a suitable protocol the communication interface connection needs to be solved. A protocol can be dependent on a special interface like CAN. After some research two protocols remained for a final investigation. Many questions needed an answer. The three most important of these were: Can the protocol manage a real-time interface? How many parameters can the protocol handle? Implementation possibilities?

As always the research started with a literature study. Internet and internal documents were used. Performance, architecture, maintenance and reliability were investigated. Visits at other divisions at Scania complemented the literature study to get their opinion about the benefits and disadvantages with the protocols. To solve the implementation problem a study of the source code of the protocols were made to see how that should be done. Taken together all the information resulted in a recommendation of a suitable protocol.

For last a compilation of PC-programs is presented. That compilation is based on information from the original assignment research.

2.3 Resources

Since this project's focus has been on very Scania specific hardware and software most studied literature have been Scania internal reports and specifications. Internet has helped me to improve my skills in Linux and textbooks [1] and [2] for C and C++.

To look for working methods and report writing structures old final thesis has been studied. To see how other students have structured their works I have learned a lot of how my own report shall look like.
3 Theory

3.1 Communicator 200

Communicator 200 is a vehicle telematic unit developed by Scania for use in trucks and buses. C200 is the communication link between the vehicle and the Scania Fleet Management Portal. Communicator 200 is designed to meet the increasing demands of customer services through the Fleet Management Portal and future functionality [10].

Figure B: Communicator 200

The system has the following main objectives [10]:
- Support advanced Vehicle Management Reports
- Enabler for DTCO data remote upload
- Driver/Vehicle performance
- The communication link between the vehicle ECU's and Scania.

3.1.1 System Overview

C200 architecture is seen in figure D. Some of C200 interfaces for communication are: GPS-receiver, GPRS/GSM-modem and CAN interface. The GPRS-modem is the connection to Scania Fleet Management Portal (FMP) and the GPS-receiver sends vehicle position to the portal. C200 has two CAN-buses, blue and yellow. The yellow bus is the interface towards the vehicle and the blue is an extern bus connected to the digital tachograph. C200 has Linux as operation system and a Boarder Support Package (BSP) with drivers for the hardware [11]. Applications are written in C++ and communicate with each other through an internal protocol.
Figure D: C200 architecture

The reason why C200 has Linux as OS is following [11]:
- The use of an reliable IP stack requires an OS
- C200 has no hard real-time applications (until this project)
- The different C200 applications must be able to work independent of each other
- Upgradeability
- It's a future safe platform
- Portability
- HW platform independent
- The most commonly used embedded OS

3.1.2 Software Architecture

The C200 software architecture is based on a layered structure as seen in figure C. The layer in the bottom handles all the hardware, while the applications are kept independent of the hardware. Linux provides support for IP-communication (IP Stack) by GPRS and Ethernet and offers IP-routing capabilities. Between the OS and the hardware a Board Support Package (BSP) is used. The BSP provides support code (drivers) to conform to the operating system. The server layer provides services to the application layer and is responsible for multi client connections [10].

At the application level a number of applications have been developed by using C++ and are designed as a framework. All applications have their own functionality e.g. vehicle monitoring and vehicle positioning.

Communication between the applications is structured as notifications and subscriptions. See figure E. Data sent between applications has a parameter attached to it. Each parameter represents a specific kind of data. A subscription is only intended to work from the bottom and up. Servers can subscribe for parameters in the application layer but it should be avoided.

To send data to an application, a notification is used. A notification is a message containing data with a parameter associated with the data.
3.2 Scania Fleet Management

Figure F shows an overview for Scania Fleet Management and how C200 is integrated with the vehicle, tachograph and Fleet Management Portal [17].
3.3 C200 communication interfaces

3.3.1 RS-232

RS-232 or recommended standard 232 is a standard for serial communication. It was introduced in 1960, and was then one the most widely used communication protocols. RS-232 describes a standard where data is send bit by bit on a physical channel. It is simple and relatively slow. RS-232 is a single-ended data transmission system, which means it uses a single wire for data transmission. Signals are processed by determining whether they are positive or negative when compared with a ground. Because signals traveling this single wire are vulnerable to degradation [18], RS-232 systems are recommended for communication over short distances, up to 15 meters and at relatively slow data rates, up to 20 kbps. However, in practice these limits can be exceeded [3]. The serial port controller in C200 is in accordance to the TIA-232-F standard.

3.3.2 CAN Controller Area Network

CAN is a standard for serial communication for data exchange between electronic control units developed by Robert Bosch in the 1980’s. It was developed for the automotive industry to increase the safety and to create an effective data communication. CAN is an international standard defined in ISO 11898. ISO 11898 defines the physical layer and the data link layer. The layers above the data link layer require additional software protocols such as SAE J1939 [20]. CAN is a two-wire, (CAN high and CAN low), half duplex, high-speed network with a maximum speed of 1 Mbit/s. The speed is related to the length of the bus and disturbances in the environment.

A CAN network contains several nodes which communicate with each other through messages. A node in the network can both send and receive messages. The bus master is the node transmitting a data during the transmission [20]. The bus access is managed through non-destructive bit-wise arbitration, which in turn provides message collision avoidance in case that multiple nodes attempt to access the bus at the same time. This makes the CAN network safe and reliable therefore it is used in the automotive industry.

In the CAN standard all messages are referred to as frames. Following four frames are available:

- **Data frame** Contains data sent to nodes
- **Remote frame** Requests data from another source and is followed by a data frame containing the requested data.
- **Error frame** Participant nodes can send this frame to report error.
- **Overload frame** Overloaded nodes can send a delay request between data and a remote frame transmission.
Every message has a unique identifier which defines the message substance. There are two different identification formats, standard with 11 bits (2.0A) and extended with 29 bits (2.0B). Scania uses the extended address register with 29 bits identifier. Figure G shows an extended 29 bits data frame.

The CAN network has a message priority system based on the identification number. Messages with low identification number have higher priority then messages with high identification number. High priority messages will have bus access within shortest time even when the bus load is high due to the number of lower priority messages. If the bus load is too high the nodes will send the messages slower or may not be able to send them at all [20].

### 3.3.3 J1939

As many of the European truck manufacturers Scania uses the SAE J1939-protocol as standard in the physical layer with a 29 bits address register.

J1939 is a high speed network supporting real-time closed loop control functions between electronic control units within a vehicle. It is developed by Society of Automotive Engineers (SAE). Its documentation covers all layers in the ISO/OSI model therefore its scope is broader than CAN [19].

The speed of J1939 is 250 kbit/s. The physical medium is intended to be shielded twisted pair.

### 3.3.4 CAN network in a Scania vehicle

A Scania truck has several ECU’s connected to three CAN-buses. The buses have different colors depending on the importance of the ECU’s function. The red bus is the most critical and controls ECU’s like engine, gearbox, brakes and suspension. The lowest critical rate has the green bus. ECU’s connected to this bus is systems that the vehicle can manage without risking the safety. All three buses are connected with each other through a coordinator system (COO) [12]. The coordinator’s task is to
route the CAN traffic to avoid unnecessary conversation between the buses. ECU-messages sending from and to the red bus have a low message ID and will therefore always have priority over messages from the other buses. C200 is located on the yellow bus. The transfer speed on the CAN-buses is 250kbit/s, which render possibility to send 3900 messages per second on each bus. Figure H shows the CAN network.

Figure H: CAN Network in a Scania vehicle

3.3.5 VCI2

VCI2 or Vehicle Connection Interface 2 is the interface between a vehicle and computer. It converts USB serial data to CAN serial data and can thus be used to connect a computer to the CAN-bus using the USB port. A computer connected to the on-board diagnostic socket and VCI2 is shown in figure I.

Figure I: VCI2 connector
3.4 Communication protocol implementation

3.4.1 SCOMM

SCOMM is an Application Programming Interface (API). The API appears in a form of a dynamic link library (DLL). The SCOMM software performs following [8] [9]:
- Takes care of ECU specific knowledge.
- Creates a higher abstraction level of the KWP2000 protocol.
- Presents data from/to ECU’s in understandable units, like V or rpm.
- Offers a generic low level interface to the KWP2000 services.
- Handles security levels in the KWP2000 protocol.
- Secures "authority" to protected ECU data by protecting security access keys.
- Enables communication with many ECU’s at the same time.
- Enables synchronously execution of functions.
- Logs communication.

3.4.2 Wrapper

Wrapper is a code which is combined with another piece of code to determine how that is executed. The wrapper acts as an interface between its caller and the wrapped code. This may be done for compatibility, e.g. if the wrapped code is in a different programming language or uses different calling conventions, or for security, e.g. to prevent

3.4.3 SMAPi

Smapi is a wrapper which makes the communication possible between the PC-application and SCOMM.
4 Test and verification

4.1 C200 test method

Test of functions in C200 is performed as following:
A C200 box is mounted in a vehicle, connected to the CAN-network and antenna. A test plan is created where the test is specified. Example of contents: Accomplish three harsh accelerations, three harsh brakes, over speed in 30 seconds etc.
A driver runs the test and notifies the time of every event. Simultaneously one or more engineers supervise the test from the office. After the test is done it is analyzed in the office. With an on-board computer test evaluation could be done in the vehicle during or after a test.

![Image](image.png)

**Figure J: Test method**

4.2 Parameters

C200 receives and refines several parameters from the vehicle CAN-network. Messages from the CAN network passes the hardware layer and operation system and reaches the CAN server. From CAN server data are distributed in the application layer depending on subscriptions from applications. Parameters are not stored anywhere in C200.

For this project two applications are interesting to know more about: CAN server and Truck Monitor. CAN server collects CAN data from the vehicle while Truck Monitor refines data from CAN-server. For example: CAN server receives a CAN message from the red bus with information about the vehicle speed and creates from that data parameter: CAN_SPEED_WHEELBASED. Truck Monitor subscribes for that
parameter and recieves the message. Truck Monitor then uses the parameter to calculate and generate new parameters like:

```
TM_TOT_NR_HARSH_ACCELERATIONS
TM_TOT_TIME_OVERSPEEDING
TM stands for Truck Monitor, TOT for total and NR number.
```

The harsh acceleration parameter is triggered when a powerful acceleration has been made. The parameter Overspeeding is triggered when the speed has been above a set speed limit for 10 seconds [14].

Below is a list of selected parameters from CAN server and Truck Monitor.

- CAN_ENGINE_SPEED
- TM_ACCELERATION
- TM_IGNITION_CHANGE
- TM_TOT_NR_HARSH_BRAKES
- TM_MAX_SPEED
- TM_TOT_TIME_OVERREVVING
- TM_TOT_DIST_OUT_OF_GEAR
- TM_DRIVER_ID

4.3 Sniffer

Scania has developed a Sniffer program for listening to internal variables in C200. It collects data from C200 to databases or a computer screen. Sniffer can listen to parameters in all C200 servers and applications, either through an instantaneous request or continuous listening. Data is transferred through the serial port or GPRS. Serial port is used for showing parameters on a computer screen and GPRS to send data and store it in servers. One problem with sending data through GPRS is that the connection is too slow to show parameters in real-time. For that purpose it is better to connect C200 to a computer via the serial port [7].

One disadvantage with the Sniffer program is to parse data. It types out data in the same speed as it comes from the applications or servers. The program does not have a function to translate incomprehensive data into understandable units. This makes it difficult to analyze data, especially when a continuous listening is performed. Today the solution is to cut interesting data and paste it in to another program where it can be translated.
Figure K shows an example of an instantaneous reading of two parameters from two servers showed in PuTTY. The id is the parameter identification number. State shows the status of the parameter, 0=ok, 1=error and 2=not available. Timestamp is the internal time in C200 and data is the specific parameter information [7].
5 Communication Protocols

5.1 General about protocols

In order for computers to communicate with each other, standard methods of information transfer and processing have been developed. These are called protocols. Computers need to have everything defined and structured to establish a reliable communication. If a computer wishes to communicate with another, it has to know in advance exactly how information is to be exchanged and what the format will be [21]. Therefore standard methods of transmitting and processing various kinds of information are used. Protocols are established by international agreement. There are many protocols for different kinds of information and functions.

5.2 KWP2000 – Keyword Protocol 2000

5.2.1 Introduction

KWP2000 is a high level signal protocol frequently used in the automotive industry. It is defined in ISO 14230. The standard defines the application, data link and physical layer. The ISO standard does not define all details, which makes it possible for manufacturers to specify their own implementation requirements. KWP2000 is often used by the after sales to diagnose ECU’s. Services that are defined by the protocol: read trouble codes, erase trouble codes, end of line programming, control I/O, flash units and read predefined signals.

5.2.2 KWP2000 message flow

Communication with an ECU comes in the form of request and response. The client uses the application layer services to request functions to be executed in one or more ECU’s. The server, usually a function that is part of an ECU, uses the application layer services to send response data. The data provided by the requested diagnostic service is sent back to the client [5].

A KWP2000 request or response consists of a diagnostic service identifier (SID) and parameters that belong to that service. The service identifier describes what actions to be taken by the ECU if it is a request. If it is a response the service identifier will either signal a positive or negative response. A negative response will be sent if the operation was unsuccessful or not completed in time. The parameters sent along with the service identifier describe what the service identifier should do. For example, SID number 0x10 tells the ECU to change its diagnostic session. The parameter sent along with the SID would then describe which diagnostic session to enter.

Each manufacturer defines application layer timing parameters. All ECU’s shall in the normal case respond to a request message within T1=100 ms. This means that each ECU shall start sending its response message within T1 after the request message has been correctly received.

When sending a request the response that matches that request has to arrive before sending the next request. This is referred to as synchronous communication. The KWP2000 message flow is seen in figure L.
5.2.3 Diagnostic Services

To create a communication between a client and an ECU a diagnostic service needs implemented in the session layer. Two services are identified as conceivable to use in this project: ReadDataByLocalIdentifier and ReadDataByCommonIdentifier. Both services work essentially the same, the differences are: CommonIdentifier can store more parameters than LocalIdentifier and is supported by multiple servers [5]. Only one diagnostic service can be active at the time in an ECU.

The parameter RecordLocalIdentifier in the ReadDataByLocalIdentifier request message identifies a server specific parameter. This parameter shall be available in the server memory. The RecordLocalIdentifier value shall either exist in fixed memory or temporarily stored in RAM [5]. For the ReadDataByLocalIdentifier service 239 parameters can be stored. Some of these are already occupied by other parameters in C200 therefore will number of parameters to reserve decrease. All parameters are allotted a specific hex value. The service ReadDataByCommonIdentifier and the parameter recordDataByCommonIdentifier have a 16 bits identifier and have the ability to store more parameters then ReadDataByLocalIdentifier with its 8. CommonIdentifier is the recommended service to use.

To request data record values (parameters), ReadDataByCommonIdentifier request message from the server identified by a RecordCommonIdentifier. The server sends parameters via the ReadDataByCommonIdentifier positive response message [5].
5.2.4 KWP2000 in C200

The Keyword Protocol server is implemented in C200. The server is found in the application KWP Manager.

KWP manager listens for parameters from other applications like CAN server and Truck Monitor. All parameters have to be implemented and reserved in the KWP2000 specification. When a parameter changes its value in another application or server KWP manager sense it and receive the new value.

5.3 CCP – Can Calibration Protocol

5.3.1 Introduction

CCP is a protocol for communication between a target processor and a computer via the CAN network. CCP is a part of the standard in ASAP. It was developed by a company called Helmut Kleinknecht in the 1990’s [6]. It is used in development process and may be used for logging signals in real-time with a sample rate of 100 Hz or less. CCP defines the communication of controllers with a master device using the CAN 2.0B (29-bit identifier), which includes 2.0A (11-bit identifier) for data acquisition from the controllers, memory transfers to and from the controllers and control functions in the controllers for calibration.

CCP can be used in the following areas [6]:
- Development of ECU.
- Systems for functional and environmental tests of an ECU.
- Test systems and test stands for the controlled devices.
- On-board test and measurement systems of pre-series vehicles.
- Non-automotive application of CAN-based distributed electronic control systems.
CCP is a master-slave communication type, figure N. The master device is connected to the ECU (slave device) via the CAN network. The master device always initiates a conversation by sending a CAN message. The ECU is responsible for responding to the received message.

![Diagram of CCP communication type]

**Figure N: CCP communication type**

### 5.3.2 Message flow

All messages and transferred data are packed in CAN frames, called message objects. These contain up to 8 byte of data [6].

Two messages objects are used:
- Command Receive Object, CRO
- Data Transmission Object, DTO

A CRO is sent from the master device to the ECU (slave devices). The ECU answers with a Data Transmission Object containing a Command Return Message, CRM. The Command Return Message purpose is to determine whether the corresponding command has been successfully completed or not.
The DTO’s carries all outgoing ECU messages and data as data packets. The first byte of a data packet is used as the Packet ID. A DTO can assume three different messages [6].

- Command Return Message, CRM
- Data Acquisition Message, DAQ
- Event Message

An Event Message reports internal slave status changes in order to invoke error recovery or other services. A Data Acquisition Message is used if a Packet ID points to a corresponding Object Descriptor Table, ODT, that describes which data acquisition elements are contained in the remaining 7 data bytes of the packet. The ODT’s are initialized and modified via protocol commands [6].
5.3.3 Organization of Data Acquisition

The master device initializes a data acquisition from an ECU. The ECU sends a response with the defined data in DAQ – DTO's. See figure P.

![Figure P: Organization of Data Acquisition](image)

Data located in an ECU memory are assigned to an ODT. This table holds the address and the length in bytes of the data. The ODT can have up to 7 element references [6].

The ODT is transferred to a DAQ message and sent back to the master device.

CCP allows a number of DAQ lists, which can be active at the same time. The sampling and transfer of the DTO's of each DAQ list is triggered by individual events in the ECU. When a DAQ list is triggered the data for all or one ODT is sampled in a consistent way. The ECU may need time to send the sampled DTO messages on the bus. There are two possibilities for the ECU to react when a new event cycle is triggered before the transfer of the previous cycle is done:

1. The transmission of the previous cycle is skipped [6].
   Advantage: There are some ODT’of a DAQ-list missing and the tool is able to show this to the user.
   Problem: If those failures occur frequently, there are some signals that are not measured at all.
2. The transmission of the new cycle is skipped.
   Advantage: The received samples are always complete and the resulting acquisition raster is reduced.
   Problem: In case of event-triggered samples, the tool has no possibility to check the plausibility of the acquisition-rate.

In addition, the ECU may transmit a DAQ processor overload event message to inform the master device. The ECU must take care not to overflow another cycle with this additional message.

When defining the measuring component in the ECU, the developer should take care that the sampled data can be transmitted via the CAN.
6 Results

6.1 C200 communication interfaces

6.1.1 RS-232

The serial interface is used for testing and setup, text input/output, boot loader and satellite communication for the future [16]. The serial interface is used by the Sniffer program. There are two major disadvantages with the serial port connection. It is too slow for the large amount of data which will be the case with several parameters logging at the same time with a sampling frequency somewhere between one and four Hz. Can Calibration Protocol is dependent on a CAN interface. The serial interface is not good enough for this project.

+ For this application none
- Slow data rate
- CAN Calibration Protocol need a CAN interface

6.1.2 Yellow CAN bus

The yellow CAN bus is the internal bus for communication between C200 and the vehicle. It is a heavy loaded bus since a lot of other ECU’s are connected to the bus. To use this bus for logging parameters from C200 would increase the traffic on the bus even more and that can cause problems in form of slow communication and interrupts. The positive thing using the yellow bus is that the OBD-socket in the vehicle can be used to connect VCI2 and computer to the vehicle and C200.

+ OBD connection
+ CAN interface suitable for many protocols
- Heavy loaded bus

6.1.3 Blue CAN bus

The blue CAN bus is an extern bus managing the digital tachograph download. The download is not performed continuously only on demand. When a tachograph download is performed the bus is heavily loaded. Other traffic will not be able to use the bus while a download is in progress. The time for a tachograph download is about five minutes. Since this is the only traffic on the bus and a download is not performed continuously most time the bus is unused. High performance, low traffic and suitable for many communication protocols makes the blue bus the best choice. The only identified disadvantage using the extern bus is the OBD-socket in the vehicle can not be used. It only connects to the intern buses. Instead the VCI2 needs to be connected direct on the blue bus in the back of the C200 unit.

+ Low traffic bus
+ CAN interface suitable for many protocols
- No OBD connection
6.2 Protocol implementation

6.2.1 KWP2000

To use KWP2000 as a protocol for logging vehicle data several changes needs to be done in C200 and KWP2000 server. The investigation about communication interface resulted in the blue CAN bus. KWP2000-server uses the yellow CAN bus for both incoming and outgoing data transfer. CAN server handles the CAN traffic and it is there changes needs to be done so it listens to the blue bus instead of the yellow. All parameters must be implemented and reserved in both KWP2000 server and PC-client.

Figure Q shows how a setup with KWP2000 shall look like. C200 transfers data on the blue CAN bus via a VCI2 connector to the computer. In the computer an application controls the communication. The application can be written in C# (C-sharp), python or CDev, depending on what the application shall perform. The application is linked together with SCOMM and SMAPi to enable a communication with C200. The application communicates with a suitable PC-program.

![Figure Q: KWP2000 communication system](image-url)
As read in previous chapter, KWP2000 is a passive protocol. Which means it have to be maintained to keep its activity. If no request is sent to the server it assumes a standby mode. To solve that problem a while loop can be initiated. An example of a while loop for XCOM\(^1\) is shown below.

```plaintext
while()
{
    Send request to ECU
    Parse response from ECU
    Print signal value to screen and/or file
    Sleep (until next sampling)
}
```

### 6.2.2 CCP

CCP is developed and optimized for reading data from an ECU’s memory. As read in chapter 2.2 about C200 the unit architecture is based on parameters. Applications is sending notifications and subscribing for parameters instead of reading from memory cells. The architecture is chosen from the facts that C200 have Linux as operation system. Linux has a memory protection which prevents a process to read and write in and from another process memory. That problem could be solved using a shared memory or the Linux/UNIX command ptrace. The ptrace system call provides a means by which a parent process may observe and control the execution of another process and examine and change its core image and registers. Though it is possible to control another process, the main problem is not solved by using these functions. C200 still doesn’t support memory management.

To use a shared memory all applications in C200 needs to be redesigned to read and write from the shared memory instead of listening and subscribing for parameters. Change the whole C200 system is not an option for an implementation of a communication protocol. The implication is to change CCP to use parameters instead of change C200 to use memory management.

Two implementation alternatives of the CCP server in C200 are conceivable. These are shown in figure R. Proposal number one is to implement the CCP server as an own application in the application layer. The other proposal is to implement it together with the application Truck Monitor.

To implement CCP as an own application demands memory management by C200. Data needs to be obtained from both CAN server and Truck Monitor, different processes and memories. This solution can be an option if C200 have a shared memory. The other alternative is to implement the CCP server together with Truck Monitor. Truck Monitor supplies most of the parameters interesting for this project. CCP server access Truck Monitors memory and read out as parameters. CCP sends it to CAN server and via the blue CAN bus to a computer.

\(^1\) Scania developed PC-client
The CCP-server from Scania Common Platform is written in C code while the applications in C200 are written in C++. To enable an implementation of the CCP-server the C code must be translated to C++. A converter can viable such an operation.

An implementation of CCP includes changes in following applications:
- CCP-server from Scania Common Platform
- Truck Monitor
- CAN server

Following five commands are necessary to implement in the CCP-server for a DAQ list initialization:

CONNECT - connect
GET_DAQ_SIZE - allocated DAQ list
SET_DAQ_PTR - set destination pointer
WRITE_DAQ - write list data
START_STOP - Start transmission of DAQ – DTO: s and set parameters
6.3 Protocol summary

6.3.1 KWP2000

+ KWP2000 server is implemented in C200.
+ Scania developed PC-clients like XCOM.
+ Scania framework SCOMM for communication between KWP2000 and a PC.
- Limited numbers of parameters to reserve in KWP2000 specification.
- Adding new parameters are complicated and time-consuming.
- One request gives one response.
- Data is not that well packed in CAN-frames as in CCP.

6.3.2 CCP

+ CCP server from Scania Common Platform
+ Higher sampling frequency then KWP2000
+ One request gives many responses
+ Compact data in CAN-frames
- Few and expensive PC-clients
- CCP server written in C
- Not implemented in C200
  ? Functionality in C200

6.4 Sampling frequency

C200 can sample signals with a high frequency. For most parameters from Truck Monitor one Hz is enough to show a reliable result. For example: Measure a harsh acceleration. It takes several seconds for a truck to accelerate and trigger a harsh acceleration.

KWP2000 is more sensitive for logging many signals than CCP. CCP can log several signals with a retained high sampling frequency. KWP2000 must decrease the sampling frequency with increasing number of signals logged simultaneously. Below are three examples from the Scania developed PC-program XCOM logging signals with KWP2000.

1-5 signals (1 byte each): 50 Hz
5-8 signals (1 byte each): 15-20 Hz
8 signals (4 byte each) 9-10 Hz
There are no problems logging signals with a lower sampling frequency.
6.5 PC programs

The first weeks of this thesis a study of PC programs to show logged data from C200 were made. The investigation did not last to long therefore only an overview of available PC programs and protocol is presented.

PC program and protocols used by Scania

<table>
<thead>
<tr>
<th>PC program</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision</td>
<td>CCP, J1939</td>
</tr>
<tr>
<td>XCOM</td>
<td>KWP2000</td>
</tr>
<tr>
<td>Gredi</td>
<td>CCP</td>
</tr>
<tr>
<td>M-Log (hardware logger)</td>
<td>CCP, J1939</td>
</tr>
<tr>
<td>FREC (hardware logger)</td>
<td>CCP, J1939</td>
</tr>
<tr>
<td>Diadem</td>
<td>KWP2000</td>
</tr>
<tr>
<td>CANalyzer</td>
<td>KWP2000</td>
</tr>
<tr>
<td>CANoe</td>
<td>KWP2000</td>
</tr>
<tr>
<td>CANape</td>
<td>CCP</td>
</tr>
</tbody>
</table>
7 Analysis

The goal of this thesis was to find how a real-time communication between C200 and a computer should look like. The result shows that a communication can be designed in several ways.

The CAN-buses are superior the RS-232 communication interface. The transfer rates and reliability are higher and safer. The blue bus is most of the time unused and fits perfect for this application though tests is not performed continuously. In the nearest future no other application is detected to be connected to the blue bus and use its capacity. In a comparison with the other two alternatives the blue bus is the best choice in all areas.

The choice of a communication protocol is more complicated. Both KWP2000 and CCP have benefits against the each other. If only the functionality of the protocols is considered the choice is easy, CCP. But the protocol must also be implemented which makes it more complex. KWP2000 is pre implemented in C200 but has low real-time performance as well as time consuming maintenance of parameters. CCP is the opposite and has very good real-time performance but the implementation demands a lot more effort to get it done.

Scania came with the idea of developing a test driver interface for C200. C200 certainly needs an interface to make the tests of functions in the unit more efficient. Scania will save both money and resources with a real-time communication interface. Here are some of the advantages: There is no need to have engineers supervising the test from the office while a test is performed, failures can be detected directly in the vehicle and evaluation of test results can be done in the vehicle.
8 Conclusion

The conclusion of this project is to use the extern blue CAN-bus as communication interface and CCP as communication protocol. These two gives the best possibilities to create a real-time interface between C200 and a computer.

The blue CAN-bus has low traffic intensity, suitable for real-time communication and supports CCP. The communication is fast and has a high safety which guaranties a reliable data transfer.

CCP is made for real-time communication and its advantages outweigh those for KWP2000. CCP can log many parameters simultaneously with a high frequency. The problem with CCP is the implementation, Linux and C200 architecture prevents CCP to use its normal memory management. The solution of that problem is to implement the CCP server together with Truck Monitor and get the access to the parameters from that application. Despite the implementation problem, CCP it is the best protocol for logging parameters in real-time, especially when many parameters are logged simultaneously.
9 Future work

This thesis has comprised an investigation about communication between C200 and a computer. The most obvious work in the nearest future is to build a demonstration system with CCP. Even the author is interested in how CCP will work as communication protocol in C200.

When a communication is established between C200 and a computer, hopefully with CCP, an investigation of this thesis original assignment is the next step. What PC-program is suitable for showing parameters from C200? There are a lot of programs with different layouts and purposes. The investigation may answer questions like: best performance, price, maintenance, layout etc. The work should contain testing programs in a vehicle and on a test bench.
References

Literature

Specifications

Scania internal documents
[9] SCOMM presentation.ppt
[10] Project_def_C200_MECO_384802_1.2.doc – C200 Project definition
[12] CAN Network with C200.doc
[15] REIV’s homepage at Scania Inline

Internet

Contacts with engineers at Scania
Magnus Eriksson RESD – Diagnostic Communication
Carl Blumenthal RESD – Diagnostics Communication
Tom Nyström REVM – Vehicle Management Controls
Pär Sundbäck REST – System and Integration Test
Martin Altin RCIV – ECU Software Verification
Jonas Fransson NEVE – ECU Support Tools
Anders Eskilson NEP – System Platform
Håkan Nilsson REIV – On-Board Platform
Jan Lindman REIV – On-Board Platform
Anders Björkman REIV – On-Board Platform
Niklas Törnkvist REIT – Test and Verification