CAN-based Data Acquisition System for Hybrid Powertrain HIPPO-2

Konstantinos Mavrogiannakis

School of Naval Architecture and Marine Engineering
National Technical University of Athens
Greece

Supervisor: Assistant Professor George Papalambrou

Committee Member : Prof. N. Kyrtatos
Committee Member : Prof. K. Kyriakopoulos

November 2018

Abstract

In this thesis, a complete communication and remote control system has been designed and implemented in order to provide data acquisition and remote control capabilities to the new Hybrid Integrated Propulsion Powertrain (HIPPO-2) testbed at the Laboratory of Marine Engineering, NTUA. HIPPO-2 consists of a Caterpillar C9.3 Diesel Engine and two ABB electric motors, one applies the total load to the system (Electric Brake) and the other is used as assistance to diesel engine.

The data acquisition and the control interface is carried through dSpace MicroAutobox II. This system gives the ability to have Analog and Digital Input/Output but also 4 CAN channels; all in one compact and standalone unit. The drivers of electric motors and the ECU of diesel engine communicate with dSpace through CAN bus physical layer but with different higher layer protocols.

This thesis will describe in detail each device and communication protocol that take part in the operation of HIPPO-2 testbed. The main tasks which took place during this thesis are study and understanding of CAN bus architecture and higher layer protocols of CAN bus, CANopen and J1939; study of ABB's CANopen Adapter Module RCAN-01; study the operation and signalling attributes of all peripheral devices (Torque Flange, Corriolis Flowmeter, Signal Conditioners, Frequency Transducer and isolation amplifiers); study of dSpace MicroAutobox II, and its Control Desk interface; integration of above and robust and effective data acquisition and control system for HIPPO-2; parametrisation of signals in Simulink and set up the interface of Control Desk to monitor HIPPO-2; experiments during the operation of HIPPO-2 and finally display and storage of results.

Contents

List of Figures **5**

List of Tables **7**

1 Introduction **9**

1.1 Framework . . . 9

1.2 Hippo-2 testbed . . . 9

1.3 Scope . . . 10

2 Communication Protocols **12**

2.1 Introduction in CAN . . . 12

2.2 CAN lower- and higher-layer protocols . . . 13

2.2.1 Physical layer options . . . 14

2.2.2 CAN data link layer . . . 14

2.2.3 Standardized higher-layer protocols . . . 15

2.2.4 Message Transfer . . . 15

2.2.5 Frame Formats . . . 15

2.2.6 Frame Types . . . 16

2.2.7 Data and remote Frame . . . 16

2.2.8 Error Frame . . . 18

2.2.9 Overload Frame . . . 18

2.2.10 Bus Arbitration . . . 19

2.2.11 Message Broadcasting . . . 19

2.2.12 Data Transfer Synchronization . . . 20

2.2.13 Error detection and Fault confinement . . . 20

2.2.14 Benefits of using CAN . . . 21

2.2.15 Designing a CAN network . . . 21

2.3 CANopen Protocol . . . 22

2.3.1 Pre-tested components and devices . . . 23

2.3.2 The Object Dictionary . . . 23

2.3.3 Types of communication in CANopen . . . 24

2.3.4 Communication Types . . . 26

2.4 SAE CAN J1939 . . . 27

2.4.1 Introduction to J1939 . . . 27

2.4.2 Physical Layer . . . 27

2.4.3 Network Layer . . . 28

2.4.4 Communication Characteristics . . . 29

3 Experimental Facility **32**
3.1 HIPPO-2 Integration 33
3.2 Sensors and Data Acquisition System 36
3.2.1 Coriolis Mass Flowmeter FCM2000 36
3.2.2 HBM T10F torque meter 37
3.3 ABB engine controllers 38
3.4 Description of Data Acqutision System 38
3.4.1 Electrical connection 40

4 HIPPO 2 data acquisition **43**
4.1 Introduction 43
4.2 CAN message communication setup 43
4.2.1 Programming of RTI_CAN1 44
4.2.2 Programming of RTI CANopen ABB 51
4.2.3 NMT configuration 56
4.2.4 PDO configuration 57
4.3 I/O of dSpace Microautobox II 61
4.4 ControlDesk 62
4.4.1 Presentation 62
4.4.2 CANopen Initialisation sequence 64
4.4.3 Electric motor operation 65

5 Operation of HIPPO-2 and Experimental Results **68**
5.1 Scenario of HIPPO-2 68

6 Conclusion and Future Work **73**

Appendices **74**

A HIPPO-2 data **75**
A.1 Electric Drawings 75
A.2 CAT C9.3 data 81
A.3 dSpace MicroAutobox II data and electric designs 84
A.4 ABB data 85
A.5 PDO messages 87
A.6 Analog Messages 87

Bibliography **90**

Acknowledgement

This thesis beccomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

I would like to thank Professor Nikolaos Kyrtatos for providing the opportunity to work with the full-scale hybrid diesel-electric marine propulsion powertrain of LME. Access to LME experimental facilities was essential in order to verify theoretical concepts from a practical perspective and evaluate experimentally the designed control system. I also thank him for being a member of my supervisors committee.

I would like to express the deepest appreciation to my thesis supervisor Assistant Professor George Papalambrou firstly for giving me the opportunity to study and for his immense knowledge which offer to me vulnerable knowledge assets through this thesis. I would also like to thank him for motivation to look beyond the problems. His expertise and work ethics taught me a lot.

Special thanks to Mr. Nikolaos Planakis with his patient guidance, enthusiastic encouragement and useful critiques of this research work. Every time I run into unexpected obstacle, always pointed out the problems and offered suggestions and solutions, which inspired my research a lot. Without his guidance and help, this thesis would not have been possible.

I would also like to extend my thanks to the technicians of the laboratory of the Marine Engineering (LME) department for their help in offering me the resources in running the program and all LME fellow members for their help and support.

I would like to express my deep gratitude to Mr Nikolaos Alexandrakis for his contribution through his advices.

I would like to express my gratitude towards my family for the encouragement which helped me in completion of this paper. Specially to my brother who supported me through this period of time bearing most professional obligations.

I would also like to thank Professor Konstantinos Kyriakopoulos for evaluating my work and being a member of my supervisors committee.

List of Figures

1.1 3-D representation of the units of the hybrid test bed HIPPO-2. 10

2.1 ISO/OSI Reference model for CAN protocol. 13
2.2 CAN is one of the most robust network technologies, especially if you are using a line-topology with very short stubs and a twisted-pair of cable. . . . 14
2.3 Architecture of Data and Remote frame in CAN 2.0 B protocol. 16
2.4 Architecture of Data and Remote frame in CAN protocol. 23
2.5 SDO Mechanism. 25
2.6 General format of SDO communication. 25
2.7 PGN and SPN in SAE J1939 message. 30
2.8 PGN example - Electronic Engine Controller 1 - EEC1. 30
2.9 Message ID example. 31

3.1 The HIPPO-2 hybrid diesel-electric testbed of LME. 32
3.2 HIPPO-2 speed and torque outputs. 33
3.3 Engine CAT C9.3. 34
3.4 ABB Electric Motor 90 kW. 34
3.5 ABB Electric Brake 315 kW. 35
3.6 HIPPO-2 overview. 36
3.7 ABB Coriolis flowmeters. 37
3.8 ABB Coriolis panel. 37
3.9 HBM T10F torque meter. 38
3.10 ABB ACS 800 controlling panels. 39
3.11 The HIPPO-2 dSpace monitoring and control board. 39
3.12 The general arrangement of data acquisition system for HIPPO-2. 40
3.13 The general design of data acquisition system. 41
3.14 +Y12-H15 Electric Panel. 42

4.1 Simulink main blocks. 44
4.2 RTI CAN1 general overview. 44
4.3 RTI CAN1 detailed (1 of 3). 45
4.4 RTI CAN1 detailed (2 of 3). 45
4.5 RTI CAN1 detailed (3 of 3). 46
4.6 CAN Controller set up. 47
4.7 J1939 Supported Parameters example. 48
4.8 EEC1 set up example 1/2. 49
4.9 EEC1 set up example 2/2. 49
4.10 TSC 1 configuring example 1/2. 50
4.11 TSC 1 configuring example 2/2. 51
4.12 CANopen_ABB simulink box. 51
4.13 CANopen_ABB controller setup. 52

4.14 EB_SDO request. 53
4.15 EB_SDO message composition. 54
4.16 Excel table of PDO21Rx of Node 2 (EM) mapping parameters. 54
4.17 SDO simulink. 55
4.18 Subsystem for selection 32 SDO packets. 55
4.19 SDO Transmit message. 56
4.20 SDO Transmit. 56
4.21 NMT message Identifier. 57
4.22 NMT message composition. 57
4.23 NMT message composition. 58
4.24 PDO23 message transmit block - message parameters. 60
4.25 PDO23 message transmit block - message composition. 60
4.26 Simulink Blocks for Analog signals. 61
4.27 ADC blocks for the four analog signals. 62
4.28 Filtration and calibration of HBM analog signals. 62
4.29 Filtration and calibration of ABB Coriolis analogue signals. 63
4.30 Initialisation of CANopen communication. 63
4.31 HIPPO-2 control screen. 64
4.32 ABB Drives Profile for AC-drives (p.312 from ACS800 firmware manual). . 66
4.33 ABB motor control example (from ACS800 firmware manual, p. 313). . . . 67

5.1 Torque, speed, pressure and fuel consumption. 69
5.2 ,NOx and EGR valve opening(%). 70
5.3 Exhaust gas mass flow, gas temperature and coolant temperature of diesel engine. 71
5.4 Torque reference from EM as compared to Torque from HBM sensor. 72
5.5 Voltage and Current from EB. 72

A.1 CAN cable 75
A.2 FCM2000 Panel 76
A.3 Electirc Drawing of Rack 1 inside +Y12-H15 Electric Panel 77
A.4 Electirc Drawing of Rack 2 inside +Y12-H15 Electric Panel 78
A.5 Electirc Drawing of Rack 3 inside +Y12-H15 Electric Panel 79
A.6 Electric Drawing of Fuel Meter Box 1 79
A.7 Electric Drawing of Fuel Meter Box 2 79
A.8 Electric Drawing of +Y12-BOX1 80
A.9 The CAT C9.3 diesel engined installed sensors(1). 81
A.10 The CAT C9.3 diesel engined installed sensors(2). 83
A.11 ZIF connector electric drawing for dSpace MicroAutobox II 84
A.12 ZIF connections with HIPPO-2 84
A.13 Power Supply electric drawing for dSpace MicroAutobox II 85
A.14 ABB (EB) motor test report 86
A.15 Cable connection for T10F torque meter sensor 87
A.16 Data label from Coriolis Flow meter 89

List of Tables

2.1 CANbus length and data ratess. 14
2.2 Setting of DLC field for required amount of data. 17
2.3 BUS arbitration on CAN bus. 19
2.4 J1939 Conductor Specifications 29

4.1 Table of NMT function. 58
4.2 Table of NMT function. 59
4.3 MCW bit values. 65

A.1 List of selected signals of SAE J1939 82
A.2 EM ABB drive ACS800-17-0100-3 +E202+F250+K457+L504+M600+N678+Q951 data 85
A.3 EB ABB drive ACS800-17-0390-3 +E202+H359+K457+L504+M601+N678+Q951 data 85
A.4 Setting points in ACS800 86
A.5 PDO messages 87
A.6 Settings of HBM speed analog signal 88
A.7 Settings of HBM speed torque signal 88
A.8 Settings of Return Fuel Flow analog signal 88
A.9 Settings of Feed Fuel Flow analogue signal 89

Abbreviations

Abbreviation	Description
NTUA	National Technical University of Athens
HIPPO	Hybrid Integrated Propulsion POwertrain
CAT	Caterpilar trade mark
EM	Electric Motor
EB	Electric Brake
CAN	Controller Area Network
RPM	Rotations Per Minute
NRS	NOx Reduction System
CEM	Clean Emissions Module
DPF	Diesel Particulate Filter
DOC	Diesel Oxidation Catalyst
HD	Hydrocarbons
ZIF	Zero Insertion Force
AT	Aftertreatment
dSMA II	dSpace MicroAutobox II
I/O	Input/Output
SCADA	Supervisory Control and Data Acquisition

Chapter 1

Introduction

1.1 Framework

Today more and more communication protocols are based on bus topology. Bus topology gives the great advantage of less cable installation and faster data transmittance in large scale projects. Bus topology is widely used in every automation system; in building sector, automotive industry, construction machineries as well as in marine sector. One of the most popular bus technologies is the CAN bus (Controller Area Network). Through CAN bus we have the opportunity to transmit information from more than 100 nodes with just one cable installation without any signal losses.

During recent years, the Laboratory of Marine Engineering, at NTUA, investigated marine hybrid propulsion technology with test bed HIPPO-1 and integrated control simulation. In 2017 it installed the latest and most modern hybrid powertrain system (HIPPO-2) for research in marine hybrid propulsion and aftertreatment technologies. This system will be used in a number of experiments about the modelling, controlling and data collection of the operation of hybrid technology in marine industry. In order to minimise the cost of installation and difficulties in operation, a very efficient and friendly-user data acquisition system was also designed.

In this work, the approach was towards the operational characteristics of signals in HIPPO-2, including their comprehensive documentation. My contribution in this project was to investigate and analyze the installed electrical infrastructure: data acquisition modules (i.e. DSpace MicroAutobox II), ECU and ABB electrical drives. In addition, the study continued by setting up the communication between the diesel engine and the electric motors through CAN bus protocol.

1.2 Hippo-2 testbed

HIPPO stands for Hybrid Integrated Propulsion POwertrain. HIPPO-2 is the new test bed at the Laboratory of Marine Engineering. The installation of HIPPO-1 provided valuable insight into hybrid power plant operation and control. The design of HIPPO-2 was based on the experience gathered from this first installation. The HIPPO-2 comprises of state-of-art components: a Caterpillar 9.3, 261 kW diesel engine with Tier 4 emission capability, incorporating EGR and a Particulate filter, a Selective Catalytic Reduction unit, an Oxidation Catalyst as well as an Ammonia trap, all controlled from a central unit through a multitude of sensing elements and actuators. The test bed dynamo-meter is an ABB 315 kW AC induction motor. The electric motor/generator is an ABB 90 kW AC induction motor. All three machines are based on a common bed-plate and the whole drive train rotates in unison(Fig. 1.1). The monitoring and control of the HIPPO-2 test bed is based on the dSPACE Microautobox II platform.

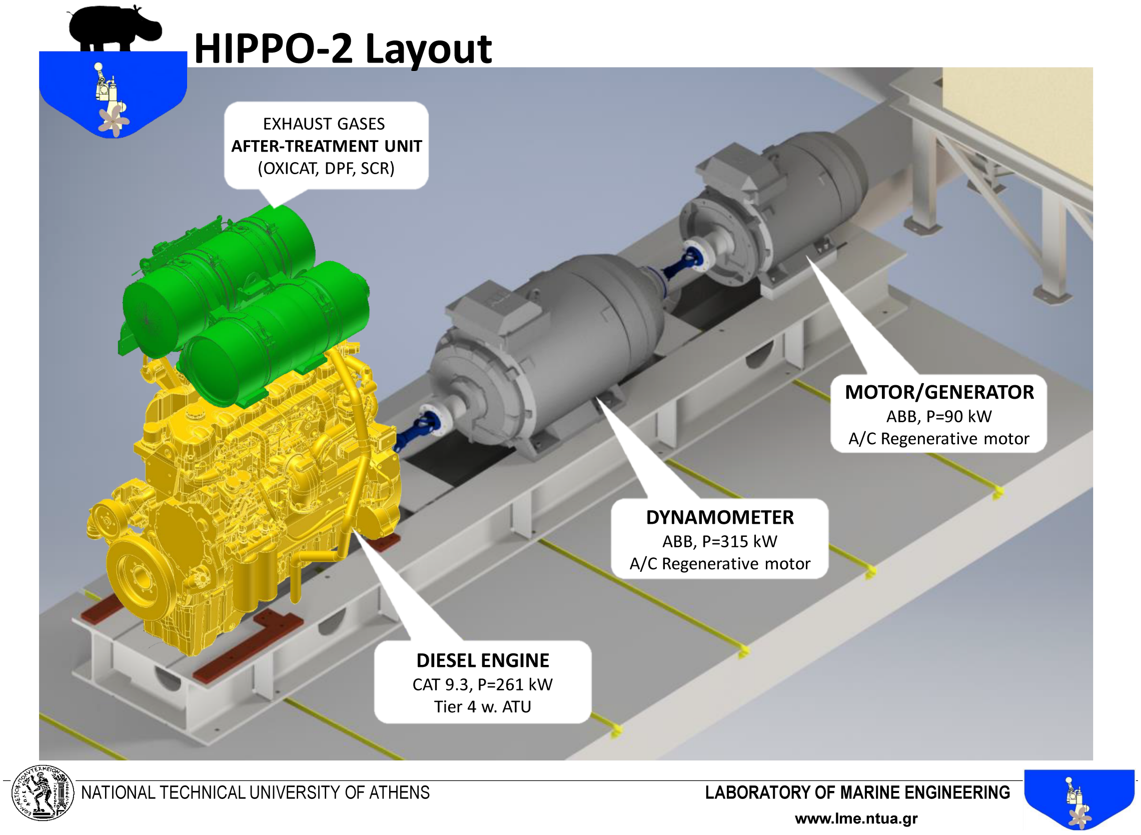


Figure 1.1: 3-D representation of the units of the hybrid test bed HIPPO-2.

1.3 Scope

This thesis studies the signal management of HIPPO-2. Signal communication of HIPPO-2 is based on CAN protocol for communication between diesel and electric engines and analog signal communication from sensors for torque, speed and fuel flow. The signal integration and data-logging is achieved by dSMA II as it establishes robust communication and control of the test bed.

In this thesis the procedure of programming and set up of configuration parameters of HIPPO-2 and control room are described in detail. The main goal is the communication integration through dSMA II and the integration of different devices, and it was approached through the following steps:

- study of CAN protocol and more specific of SAE J1939 and CANopen higher layer protocols
- analysis of analog signal
- acquisition and analysis of CAN and analog signals
- operation of HIPPO-2 with SCADA system for experimental purposes
- evaluation of HIPPO-2's safe operation with dSpace Control Desk
- detailed description of the plant electronic and electric parts as well as signal acquisition

The main outcome is the documentation of the most important parameters of HIPPO-2 data acquisition system so that future end users who will work in HIPPO-2 test bed will have a comprehensive guide for configuration and troubleshooting purposes.

Chapter 2

Communication Protocols

2.1 Introduction in CAN

Imagine that your car is like a human body, the Controller Area Network system (CAN bus) is the nervous system, facilitating communication between all parts of the body. Similarly 'nodes' or electronic control units (ECU) are connected via the CAN bus, which acts as a central networking system.In an automotive CAN bus system, ECUs can e.g. be the engine control unit, airbags or the audio system. A modern car can have up to 70 ECUs. This is where the CAN standard comes in handy as it allows ECUs to communicate with each other without complex dedicated wiring in between. In turn this allows for several features to be added via software alone e.g. electronic gearbox control. Indeed, that is the primary purpose of CAN: To allow any ECU to communicate with the entire system without causing an overload to the controller computer. CAN is quite popular for five key reasons [8]:

1. Low cost: ECUs communicate via a single CAN interface, i.e. not direct analog signal lines, reducing errors, weight, costs
2. Centralized: The CAN bus system allows for central error diagnosis and configuration across all ECUs
3. Robust: The system is robust towards failure of subsystems and electromagnetic interference, making it ideal for e.g. vehicles
4. Efficient: CAN messages are prioritized based on IDs so that the highest priority IDs are non-interrupted.
5. Flexible: Each ECU contains a chip allowing it to receive all transmitted messages, decide relevance and act accordingly. This allows easy modification and inclusion of additional nodes such as CAN bus data loggers

A fast reference in CAN history [2]:

- PreCAN : Car ECUs relied on increasingle complex point to point wiring
- 1986: Bosh developed the CAN protocol as a solution, releasing it at a SAE congress
- 1991: Bosh published CAN 2.0, CAN 2.0A (11 bit) and CAN 2.0B (29 bit)
- 1993: CAN is adopted as international standard (ISO 11898)
- 2012: Bosch released the CAN FD 1.0, aka CAN with flexible data rate

Controller Area Network (CAN) is an asynchronous serial communication protocol which follows ISO 11898 standards and is widely accepted in automotive due to its real time performance, reliability and compatibility with wide range of devices. CAN is a two wire differential bus with data rates up to 1Mbps and offers a very high level of security. Its robust, low cost and versatile technology made CAN applicable in other areas of applications where inter processor communication or elimination of excessive wiring is needed. Some of the areas it is widely used are industrial machinery, avionics, IOT devices, medical equipments, home automation etc. [3]

The main characteristics of CAN protocol are:

- Multi master hierrarchy
- Priority based bus access
- Baud rate up to 1 Mbps
- Error detection and fault confinement

2.2 CAN lower- and higher-layer protocols

Like all modern communication technologies, CAN-based network solutions follow the Open Systems Interconnect (OSI) reference model. It is standardized in ISO/IEC 7498-1. For CAN networks, this model has been adapted. The CAN reference model comprises the CAN physical layer (layer-1), the CAN data link layer (layer-2), and the CAN application layer (layer-7). The other layers (3 to 6) are often not explicitly implemented or are implemented in higher layer protocols like CANopen, J1939 etc. It creates a bridge from Data link layer to Application layer(as shown in 2.1 in order to limit the resources and to improve the performance. However, some application layers also include the network and transport layer functionality. All these communication services and protocols specify only the communicating behaviour. The application functionality is not in the scope of the OSI model. It is part of the device or application profile specifications.

The physical layer and data link layer are integrated on the CAN controller chips and the libraries for connection between the data link layer and the application layer are provided by the CAN chip manufacturers.

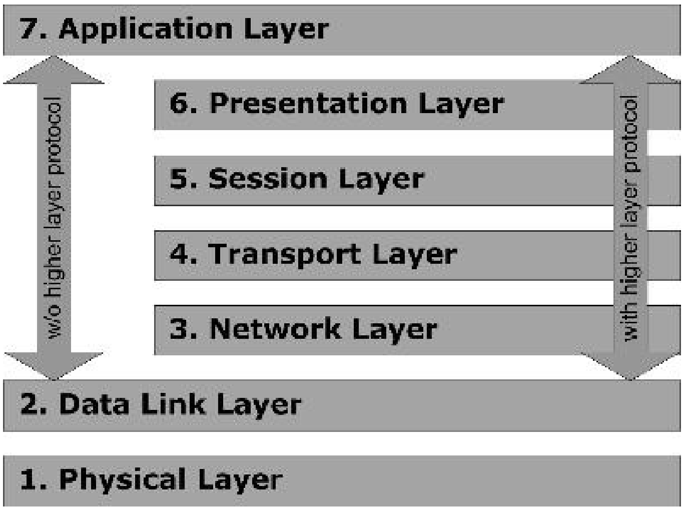


Figure 2.1: ISO/OSI Reference model for CAN protocol.

2.2.1 Physical layer options

CAN is one of the most robust network technologies, especially if you are using a line-topology with very short stubs and a twisted-pair of cable. In figure 2.2) depicted a single twisted pair wire with three nodes terminated on each end with 120 Ohm resistors.

A twisted-pair copper cable with common ground usually realizes the physical transmission. Of course, all connected nodes need to support the same data-rate and the same bit-timing settings.

Most common is the high-speed transmission as standardized in ISO 11898-2:2003. It supports data-rates up to 1 Mbit/s. There are also high-speed transceivers with low-power capability (ISO 11898-5) and selective wake-up functionality for partial networking (ISO 11898-6). The three standards mentioned above are merged into one standard (ISO 11898-2:2016). Depending on the bit rate there is a maximum length restriction (table 2.1)

There is not specified maximum number of nodes are allowed on CAN bus but the networks are really limited by the electrical loading of the revolving nodes and transmitter but typically up to 64 nodes is normal.

Bus length	Bit rate tradeoff
1M bit/sec	40 meters
500k bit/sec	100 meters
250k bit/sec	200 meters
125k bit/sec	500 meters

Table 2.1: CANbus length and data ratess.

The robustness of high-speed CAN networks is excellent. In-vehicle networks often use star-topologies, sometimes with multiple stars. In some applications, hybrid topologies are used, combining line and star. However, the most robust topology is a bus-line with very short stubs.

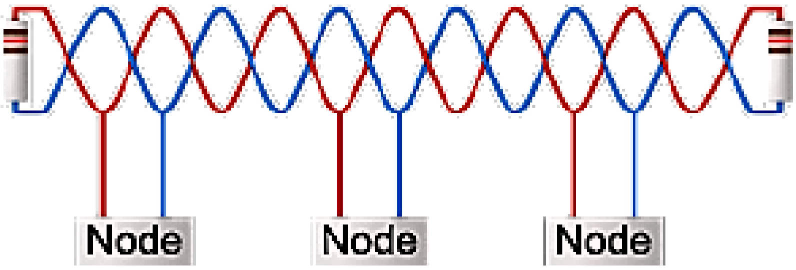


Figure 2.2: CAN is one of the most robust network technologies, especially if you are using a line-topology with very short stubs and a twisted-pair of cable.

2.2.2 CAN data link layer

CAN is a very reliable multi-master serial bus system with multi-drop capabilities. The bus arbitration method is the same for both CAN data link layer protocols. The CAN messages are broadcasted. This means every node is able to consume any message produced by any other node in the CAN bus system.

Originally developed for use as an in-vehicle network in passenger cars, nowadays CAN is used in many other industries. This includes applications in any kind of transportation system (rail vehicle, aircraft, marine, etc.), in industrial machine control systems, in home and building automation (e.g. HVAC, elevators), in mobile machines (construction and agriculture equipment), in medical devices and laboratory automation, as well as in many other embedded and deeply embedded applications. Each year, about 1 billion CAN nodes are sold. The price for CAN protocol controllers is very reasonable.

Originally, there was just the data link layer protocol, what we today refer to as Classical CAN. Introduced by Bosch as CAN 2.0 A/B, it has been internationally standardized in the ISO 11898 series since 1993. CiA doesnt use the term CAN 2.0 anymore, calls it Classical CAN now [4].

2.2.3 Standardized higher-layer protocols

There are several CAN-based higher-layer protocols. The most generic one is CANopen. It comprises the CiA 301 application layer standardized internationally in EN 50325-4 and several device, interface, and application profiles (CiA 4xx series). Additionally, there are other protocol specifications for additional communication functions (CiA 3xx series). CANopen is used in many embedded and deeply embedded control applications.

The family of CAN protocol J1939-based application profiles is another approach. The original J1939 specifications were designed for in-vehicle networks in trucks and buses. The Society of Automotive Engineers (SAE) develops and maintains these recommended practices. The internationally standardized ISO 11783 series (also known as Isobus), one of the J1939 derivatives, is dedicated to implementing communication in agriculture tractors. It can also be used for forestry mobile machines. The IEC 61162-3 (also known as NMEA2000) is another J1939 derivate; it is optimized for CAN-based marine navigation networks. The ISO 11992 series standardizes the communication between trucks and trailers. This J1939 derivate is a base for the European regulation of towed vehicles.

In the passenger car industry, application layers are usually proprietary. Exceptions are the CAN transport layer and the diagnostic services, which are internationally standardized in ISO 15765-2 respectively in the ISO 15031 series. Another exception is an open network for add-on devices. This is specified in the CANopen application profile CiA 447. [4]

2.2.4 Message Transfer

The basic characteristics of CAN messages are:

- All messages are broadcast
- Any node is allowed to broadcast a message
- Each message contains an ID that identifies the source or content of a message
- Each receiver decides to process or ignore each message

In the CAN protocol, nodes communicate data or information through messages termed as frames.

2.2.5 Frame Formats

There are two different formats which differ in the length of the Identifier field: Frame with the number of 11 bit Identifier are denoted Standard Frames. In contrast, frames containing 29 bit Identifier are denote Extended Frames

2.2.6 Frame Types

A frame is transmitted on the bus only when the bus is in idle state, message transfer is manifested and controlled by four different frame types:

- Data Frame used to transmit data from a transmitter to the receivers.
- Remote Frame is used to request data transmission of the Data Frame with the same Identifier.
- Error Frame is transmitted by any unit that detects a bus error
- Overload Frame is used to provide for an extra delay between the preceding and the succeeding Data or Remote Frames

Data frame and Remote frame can be used both in Standard Frame format and Extended Frame format; they are seperated from preceding frames by an Interframe Space

The frames transmitted from one node will be received by all the other nodes on the network using message broadcasting. The message filtering which is provided by the CAN controller hardware decides wether the received frame is relevant to that node or not. If an error occurs due to reception or transmission, an error frame will be transmitted on the bus to let the network know of the error frame starts with a 6 dominant bits, it will have highest priority when the bus is idle. As soon as the error is detected, the CAN protocol implements the fault confident techniques to overcome the error. The fault confinement feature in the CAN protocol differentiates between a temporary error and a permanent failure of a node. If the error is due to permanent failure of the node, it automatically detaches the defective node from the bus without causing any problems to the network

As mentioned earlier CAN provides four different types of message frames for communication, the architecture of each frame is discussed in this section.

2.2.7 Data and remote Frame

The architecture of the data and the remote frame are exactly the same. A data frame has higher priority than a remote frame. Each data and remote frame starts with a Start Of Frame (SOF) field and end with an End Of Frame (EOF) field. The figure 2.3 gives architecture of data and remote frames for CAN 2.0 B -extended format [1].

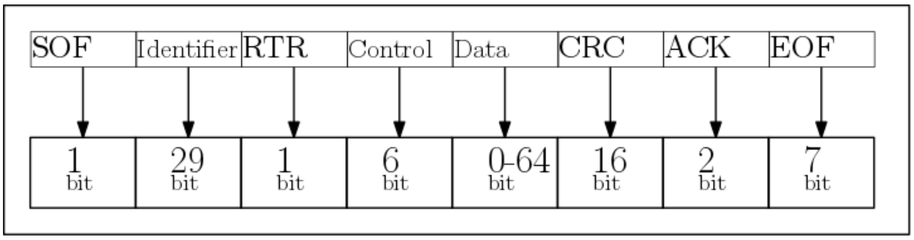


Figure 2.3: Architecture of Data and Remote frame in CAN 2.0 B protocol.

The following are the fields in data and remote frame:

- SOF field (1 bit) – Indicates the beginning of the frame. A single dominant bit represents a start of a frame to tell the other ECUs that a message is coming. It is also used for data transfer synchronization.
- Arbitration Field – This contains two sub fields, Message Identifier or CAN-ID and RTR field.

  - CAN ID contains the message priority as well as functional address (e.g. RPM, wheel speed..)(29 bits) – This field contains a message ID for each frame which is 29 (Extended ID) bits. No two message frames in the CAN network should have the same message ID. A message ID which has a low decimal value is considered as a high priority message.
  - Remote Transmission Request (RTR) (1bit) – The RTR field distinguishes a data frame from a remote frame.The Remote Transmission Request allows ECUs to "request" messages from other ECUs

- Control Field (6 bits) – This contains two sub fields, IDE and DLC field
  - Identifier Extension (IDE) Bit (1 bit) – This bit indicates the format of the message ID in the frame, either a stadard 11-bit format or extenf 29-bit format.
  - Data Length Code (DLC) field (4 bits) – This field is used to set the length of the Data in bytes (0 to 8 bytes) being transferred from one node to other node. In a remote frame, these bits represent the amount of data it is requesting. In the table 2.2 are the values of the DLC bits for the corresponding amount of data.

- Data Field – This field contains the actual data values, which need to be "scaled" or converted to be readable and ready for analysis. It is not applicable for remote frame.

- CRC field (16 bits) – Cyclic Redundancy Check checks data integrity. The CRC field consists of the CRC Sequence and a CRC Delimiter bit.
  - CRC Sequence field (15 bits) – This 15 bit field contains the frame check sequence without the stuffing bits.
  - CRC Delimiter (1 bit) – This bit is used to provide processing time for the CRC Sequence field

- ACK field (2 bits) – The ACK slot indicates if the CRC process is OK. The ACK field consists of a 1 bit Acknowledgement Slot field and Acknowledgement Delimiter bit (which is always recessive).

- EOF field (7 bits) – Indicates the end of the frame. A seven bit continuous recessive bit represents the end of frame.

DLC (3 bits)	Amount of data
dddd	0 bytes
dddr	1 byte
ddrd	2 bytes
ddrr	3 bytes
drdd	4 bytes
drdr	5 bytes
drrd	6 bytes
drrr	7 bytes
rddd	8 bytes

Table 2.2: Setting of DLC field for required amount of data.

A node uses data frane to transmit data to any other node on the network. The RTR field determines whether the message frame should act as data frame or a remote frame. When the RTR bit is set to dominant level, then the message frame will act as a data frame, when the RTR bit is set to recessive level, then the message frame will act as a remote frame. A maximum of 8 bytes of data can be transferred using a single data frame. In each data frame will be assigned a unique message ID from which the node decides whether the data is relevant or not.

A remote frame is used to request a data frame from any node on the network. When the RTR bit is set to recessive level, then the message frame will act as a remote frame. While requesting data from a node, the length of the data field in control field (DLC bits) of the remote frame should be same as the requesting data frame otherwise a bus collision occurs. As soon as the remote frame is accepted by a node, a data frame will be transmitted on to the bus with the requested data. When two or more nodes on the netwwork request the same message at the same time a bus collision occurs.

Only 3 of the 8 key fields of data frame are practically relevant in most data logging scenarios: The CAN ID, the Control and the Data field.

2.2.8 Error Frame

An error frame is transmitted onto the bus whenever a transmission or reception error occurs due to a faulty node or bus problems. An error frame consists of three fields

- Error flag (6 bits) – It is 6 dominant bits which indicates the transmitting or receiving error on the bus

- Error Delimiter (8 bits) – It is represented as a sequence of 8 recessive bits. After transmitting the error flag each node transmits a single recessive bit and waits for the bus level to change to recessive. Only after the bus level is recessive, the remaining 7 recessive bits will be sent onto the bus.

- Interframe Space (3 bits) – It is represented as minimum space between any type (data, remote, error, overload) of frame and a following data or remote frame. It contains of 3 recessive bits

2.2.9 Overload Frame

The overload frame takes the same form of the frame but the overload frame is used to request a delay between the transmission of the next data or remote frame. It consists of two fields, Overload flag and an Overload delimiter.

- Overload flag (6 bits) – It contains 6 dominant bits which indicates the transmitting or receiving error on the bus.

- Overload Delimiter (8 bits) – It is represented as a sequence of 8 ressecive bits. After traansmitting the error flag each node transmits a single recessive bit and waits for the bus level to change recessive. Only after the bus level is recessive, the remaining 7 recessive bits will be sent onto the bus.

- Interframe Space (3 bits) – It is represented as minimum space between frames of any type (data, remote, error, overload) of frame and a following data or remote frame. It transmits 3 consecutive recessive bits on the bus. When the Interframe space is being transmitted, no node on the network is alloed to transmit any of the frames except the overload frame.

2.2.10 Bus Arbitration

In a single bus communication protocol, when two or more nodes request access to the bus then the bus arbitration technique comes in. Usually bus access will be given to a node with a high priority. The bus arbitration technique also reduces data collisions.
CAN protocol provides a non-destructive bus arbitration mechanism. It assigns a recessive level to the bus only of all the nodes on the network output a dominant level. When a dominant bit and recessive bit request access for the bus, the dominant bit is given the access as it is considered as the high priority. So the bus arbitration on CAN network follows an AND gate logic as shown in table A.4

A CAN bus is a half duplex, two wire differential bus. The two lines, CAN_L and CAN_H, form the communication bus for the nodes to transmit data or information. The logic levels used on the bus are dominant and recessive level, where dominant level is referred when TTL=0V and recessive level, where dominant level is referred when TTL=5V. The dominant level always overrides recessive level and this concept is used to implement the bus arbitration. The voltage levels on the CAN bus varies from 1.5 volts to 3.5 volts. The logic levels are calculated as the voltage difference between the two lines.

$$V_{diff} = V_{CAN_H} - V_{CAN_L} \tag{2.1}$$

If the difference voltage (V_{diff}) is 2 volts, it is considered as a dominant level and if it is 0 volts, it is considered as a recessive level.

When transmitting a frame on the bus, the bus access to a node will be a given based upon the message ID of the frame. As the dominant level is considered as high priority, the message ID with more dominant bits is considered as a high priority message. When the bus is idle, the bus access will be assigned to the node which transmits a message with a higher priority.

Node 1	Node 2	Bus logic level
Dominant	Dominant	Dominant
Dominant	Recessive	Dominant
Recessive	Dominant	Dominant
Recessive	Recessive	Recessive

Table 2.3: BUS arbitration on CAN bus.

2.2.11 Message Broadcasting

CAN protocol is based on message broadcasting mechanism, in which the frames transmitted from one node is received by every other node on the network. The receiving nodes will only react to the data that is relevant to them. Messages in CAN are not acknowledged due to unnecessary increase of traffic. But the receiving node checks for the frame consistency and acknowledges the consistency. If the acknowledge is not received from any or all the nodes of the network, the transmitting node posts an error message to the bus. If any of the nodes are unable to decode the transmitted message due to internal malfunction or any other problem, the entire bus will be notified of the error and the node re-transmits the frame. If there is an internal malfunction in a node, that particular node reports an error for each frame it receives. Due to this most bandwidth of the network will be allocated to error frames as they have higher priority (starts with 6 consecutive dominant bits). To overcome this problem the CAN protocol supports a bus off state in a node, in which the node will be detached from the bus if it reports an error for more than a pre defined value. The bus off state of a node is implemented to avoid the breakdown of the network due to a single node.

While broadcasting data frames on the bus, each node on the bus receives every data frame transmitted on to the bus. As CAN protocol does not support IDs for the nodes and the receiver does not know the information of the transmitter of the frame, each data frame goes through an acceptance filtering process at the receiving node, which is dependent on the message ID (standard or extended) of the frame.

The process of data requesting in CAN protocol is carried out by the remote frame. The RTR bit in a frame decides whether the frame is a remote frame or data frame. When the RTR bit is set to recessive level the frame will act as a remote frame. When a node is requesting a data frame from another node, the message identifier section (ID bits) and data length section (DLC bits) in the remote frame should be of same value of that in the data frame that is requested otherwise an error will be reported on the bus.

2.2.12 Data Transfer Synchronization

Each node in the CAN network will have different oscillators running at different frequencies, so to make all the nodes work synchronously while transferring data, the CAN protocol uses the falling edge of the SOF bit (transition from recessive to dominant bus level).

The bit coding used in CAN bus is Non-Return-to-Zero principle in which the bit level remains constant during the entire bit time, which creates a node synchronization problem during the transmission of larger bit blocks of same polarity. To overcome this problem CAN protocol uses bit stuffing mechanism.

Bit Stuffing: The CAN protocol allows only 5 consecutive bits of same polarity between the SOF bit and Data field. If more than 5 consecutive bits of same polarity are transmitted on to the bus it will be considered as an error condition. So to transmit data with more than 5 consecutive bits of same polarity the CAN protocol inserts a complementary bit of opposite polarity at the transmitter end and at the receiver end the filtering should be performed to get rid of the stuffed bit. Bit stuffing is applied only in data and remote frames and it is not applicable after CRC field.

2.2.13 Error detection and Fault confinement

The CAN protocol implements a series of error detection mechanisms which contributes to the high level of reliability and error resistance. It also implements fault confinement mechanisms for proper function of the network. The error detection mechanisms implemented are:

- Bit monitoring – Transmitter compares each bit that is transmitted on to the bus with the data it is transmitting and reports an error if there is change in the data transmitted.

- Checksum check – Every data and remote frame has a 15-bit CRC filed which carries the checksum of the frame and is used to detect errors at the receiver.

- Bit stuffing – CAN protocol allows only 5 consecutive bits of same polarity. Bit stuffing is implemented during transmission of more than 5 consecutive bits of same polarity. If more than 5 consecutive bits of same polarity are transmitted, the bus takes it as an error frame as the first 6 bits of the error frame are dominant bits.

- Frame check – Each transmitting and receiving node checks for the consistency of the frames.

- Acknowledge check – Each receiving node transmits frame consistency acknowledge to the transmitting node.

Whenever an error occurs on the bus, each node on the network receives the error frame and the transmitting node serves the error by re-transmitting the frame.

The CAN protocol implements fault confinement techniques to ensure that the communication on the network never fails.Consider a situation in which a node has an internal malfunction caused due to electrical disturbances and transmits error frame for every frame it receives. For serving these kind of errors, CAN protocol is supplied with two counters, a transmit error counter and a receive error counter. The corresponding counter is incremented each time a failure in the transmission/reception occurs. The counter is decremented whenever there is a successful transmission/reception. The counter value does not decrement when the value is zero. Based upon the values of two counters the CAN nodes will have three states, Error active, Error passive and Bus off state.

- Error active state – Every node after reset starts with this state in which it transmits an error flag (6 consecutive dominant bits) whenever it receives an error frame.

- Error passive state – A node enters into this state when a receive error counter or transmit error counter value is equal to or greater than 127. When the node is in error passive state, it transmits an error flag with 6 consecutive recessive bits.

- Bus off state – A node enters into this state when a transmit error counter value increases more than 255.

2.2.14 Benefits of using CAN

Controller Area Network is a serial communication protocol which is mainly used for reducing wired interconnections in a vehicle. Some of the benefits in implementing CAN protocol in automotive are:

- Reduced wired interconnections

- Low cost implementation

- Speed, reliability and error resistance

- Worldwide acceptance

2.2.15 Designing a CAN network

When designing a CAN physical network, you have to make some general decisions. First of all, you need to choose the required data throughput. In theory, you can't overload the CAN network. You can only delay lower-prior messages, but then your application does not work properly. For a short time, busloads above 50 percent are acceptable. But an average busload of 50 percent should not be exceeded.

Now you can calculate the necessary transmission speed, which is the number of message bits doubled to be sent in a dedicated time window (e.g. one second). You can and should ask the software system designer to reduce the communication as much as possible. It is recommended to communicate as little as possible, but as much as necessary. Avoid communication without new information (CANopen offers you many options to do so, due to its flexible parametrization of messages). If you have optimized your busload and know the minimum bit-rate you need, you can calculate the maximal achievable network length.

You can also do it the other way around, of course: You take the needed network length and determine the possible maximum bit-rate. Here, you should optimize the communication to a minimum too. If possible, always use the minimum bit-rate, because the lower the speed, the more robust the communication.

You have more options to optimize your design. First of all is the topology: use a line topology terminated at both ends with resistors matching the impedance of all other physical layer elements (cable, connector, etc.). When using bit-rates much higher than 1 Mbit/s (CAN FD), even the impedance of micro-controller ports can be an issue. Star and mixed topologies limit the achievable speed/length ratio. It is recommended to keep the not terminated stubs and branches as short as possible. Even carmakers often using star topologies consider going for line topologies when introducing CAN FD communication.

If the network is quite long, it is recommended to use galvanic isolation. But this shortens the maximum length or speed due to the additional delay caused by the transformer or optocoupler. You should also use twisted-pair cables and if necessary you should shield them, in particular in noisy environments.

Most new designs are based on the CAN high-speed physical layer (ISO 11898-2). The transceivers are optionally available with low-power functionality. This means, the ISO 11898-3 approach (low-power, fault-tolerant limited to 125 kbit/s) is not state-of-the-art any longer. The single wire CAN (SAE J1411) is no option anyway, due it being slow and not robust.

2.3 CANopen Protocol

Real world problems require the integration of several devices that have same or similar functionality. However, more than often, these devices may not be interchangeable due to different implementation details, communication protocols or parameter setups. The CIA CAN in Automation organization has created several standards to define functionality of such devices so that they may be used interchangeably. Devices profiles like generic I/O modules, drives and motion control have been created. Today, there are many vendors selling products with integrated CANopen protocol. CANopen is used widely in e.g industrial automation applications. It builds on CAN in terms of the physical layer (i.e. lines used, voltages,..) and the data link layer (i.e. utilizing a CAN frame message based on protocol). In practice this means that a data logger like dSpace Autobox II is able to log data from CANopen applications, though the method for ex post data conversion will vary depending on the specific application. The CANopen standard consists of an addressing scheme, several communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation/de-segmentation. The basic CANopen device and communication profiles are given in the CiA 301 specification released by CAN in Automation. CAN may be implemented over a number of physical media as long as the drivers are open-collector and each node can hear itself (and others) while transmitting (this is necessary for its message priority and error handling mechanisms). The message frames generally used to carry data are shown in figure 2.4. They contain the following fields (this is a simplified description as the controller takes care of the detail)

- Start of frame (SOF)
- Message Identifier (MID) the Lower the value the Higher the priority of the message length is either 11 or 29 bits long depending on the standard being used

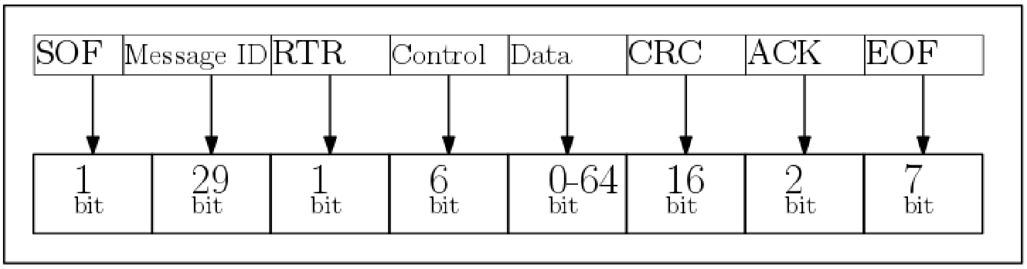


Figure 2.4: Architecture of Data and Remote frame in CAN protocol.

- Remote Transmission Request (RTR)
- Control field (CONTROL) this specifies the number of bytes of data to follow
- Data Field (DATA length 0 to 8 bytes)
- CRC field containing a fifteen bit cyclic redundancy check code
- Acknowledge field (ACK) an empty slot which may be filled by any and every node that receives the frame (it does NOT say that the node you inteded the data for got it, just that at least one node on the whole network got it.)
- End of Frame (EOF)

CANopen allows the implementation of up to 127 CAN-nodes on a single bus. Each node must be provided with a unique node ID. Each unique node allows the use of more than one address number. In CANopen instead of addresses, they are referred to as Communication Object Identifiers (COB ID).

2.3.1 Pre-tested components and devices

Most of the micro-controllers with on-chip CAN modules are tested. The automotive industry requires CAN controllers that have passed the conformance testing according to the ISO 16845 series. Device makers implementing CANopen or DeviceNet or Isobus should also test the conformance of the application layer functions.

Conformance testing is like spell checking: It doesnt prove interoperability. The CANopen conformance test tool includes an EDS (electronic data sheet) test, which proves the interoperability with CANopen design and configuration tools. EDSs are used as an exchange format between tools.

2.3.2 The Object Dictionary

One of the central themes of CANopen is the object dictionary (OD), which is essentially a table that stores configuration and process data. It is a requirement for all CANopen devices to implement an object dictionary. The CANopen standard defines a 16-bit bit index and an 8-bit sub-index. That is, it is permissible to have up to 65536 indices and up to 256 subentries at each index. The standard defines that certain addresses and address ranges must contain specific parameters. As such, any CANopen master can read this index from a network of CANopen slaves in order to uniquely identify each slave by name. Some object dictionary indices, such as the device type (1000h) are mandatory, and others, such as the manufacturer software version (100Ah) are optional. The collection of mandatory indices represents the minimum object dictionary, which is required to brand a device CANopen compliant.

The object dictionary is the method by which a CANopen device can be communicated with. For example, one could write a true to the index in the manufacturer-specific section of the object dictionary (2000h-5FFFh), which the device could interpret as an enable signal for acquiring data from a voltage input. Conversely, the master may also want to read information from the object dictionary to get the acquired data, or to find out how to device is currently configured. The two communication mechanisms for accessing the object dictionary are Service Data Objects (SDOs) and Process Data Objects (PDOs), which will be explained in this section.

2.3.3 Types of communication in CANopen

CANopen communication can be sub-divided into four classes:

- *Service data object (SDO) communication*: Based on existing CAL CMS domain protocol services; SDOs are mainly used for device parametrisation and configuration.
- *Process data object (PDO) communication*: Used during normal network operation to transfer data in real-time with little or no protocol processing overhead.
- *Network management (NMT) functions*: These tele-grams are used for co-ordinating device and network operation in a controlled and well-defined manner.
- *Predefined format messages*: These telegrams support the implementation of timing, sunchronisation an error reporting mechanisms.

Each of these types is discussed in the following sections.

Service Data Objet (SDO) The CANopen protocol specifies that each node on the network must implement a server that handles read/write requests to its object dictionary. This allows for a CANopen master to act as a client to that server. The mechanism (fig 2.5) for direct access (read/write) to the servers object dictionary is the Service Data Object (SDO). The node whose object dictionary is accessed is referred to as the SDO server, and the node grabbing the data is referred to as the SDO client. The transfer is always started by the SDO client. SDOs allow access to a single entry in the Object Dictionary,specified by index and sub-index. They use the clientserver communication model, where the client accesses the data and the server owns the target Object Dictionary. (All relevant data that characterizes a device profile of a CANopen device is implemented in an Electronic Data Sheet (EDS) file. The EDS organizes objects in a dictionary that stores the relevant data.) SDOs are typically used for device configuration or for accessing a large amount of data at a very low rate. The difference between a Transmit and Receive SDO is that in case of a Receive SDO a request to obtain the contents of an object must be sent first. Data greater then 4 bytes in length can be transferred using multiple telegrams. The general format of SDO transfer is illustrated in figure 2.6

The CANopen protocol specifies that each node on the network must implement a server that handles read/write requests to its object dictionary. This allows for a CANopen master to act as a client to that server. The mechanism for direct access (read/write) to the servers object dictionary is the Service Data Object (SDO). The node whose object dictionary is accessed is referred to as the SDO server, and the node grabbing the data is referred to as the SDO client. The transfer is always started by the SDO client. SDOs allow access to a single entry in the Object Dictionary, specified by index and sub-index. They use the clientserver communication model, where the client accesses the data and the server owns the target Object Dictionary. (All relevant data that characterizes a device

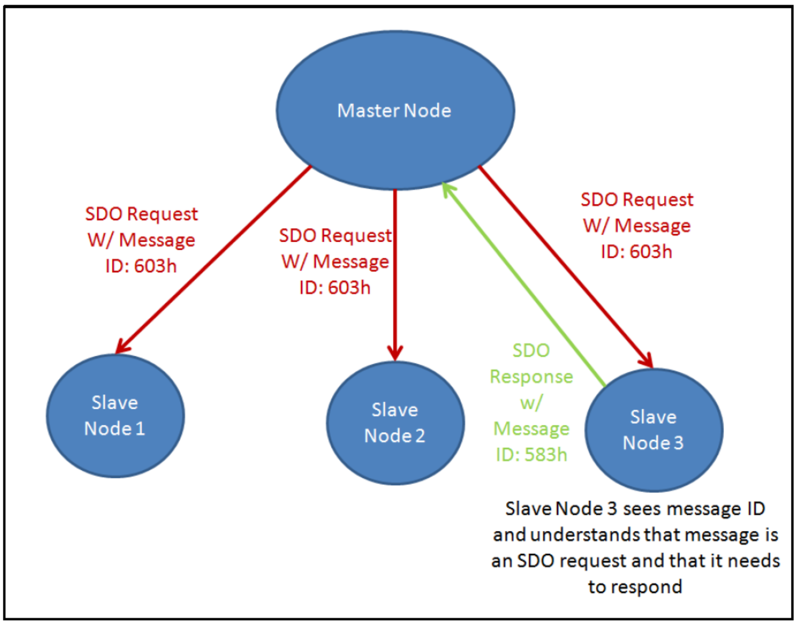

Master Node
SDO Request W/ Message ID: 603h
SDO Request W/ Message ID: 603h
SDO Request W/ Message ID: 603h
SDO Response w/ Message ID: 583h
Slave Node 1
Slave Node 2
Slave Node 3
Slave Node 3 sees message ID and understands that message is an SDO request and that it needs to respond

Figure 2.5: SDO Mechanism.

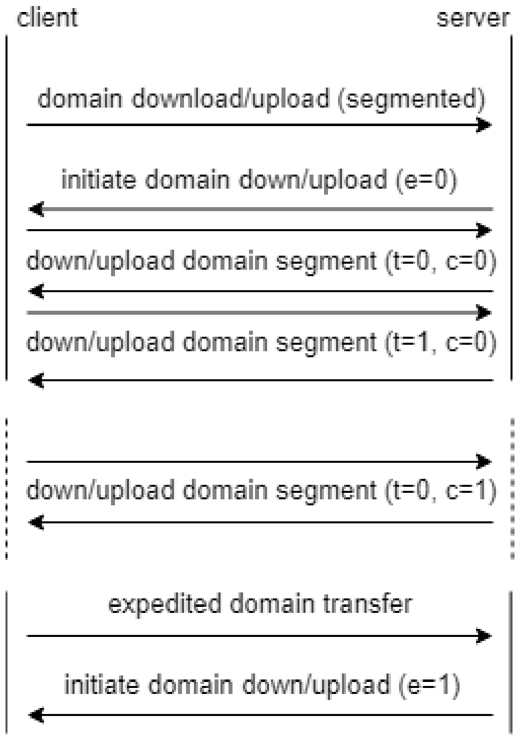

client
server
domain download/upload (segmented)
initiate domain down/upload (e=0)
down/upload domain segment (t=0, c=0)
down/upload domain segment (t=1, c=0)
down/upload domain segment (t=0, c=1)
expedited domain transfer
initiate domain down/upload (e=1)

Figure 2.6: General format of SDO communication.

profile of a CANopen device is implemented in an Electronic Data Sheet (EDS) file. The EDS organizes objects in a dictionary that stores the relevant data.) SDOs are typically used for device configuration or for accessing a large amount of data at a very low rate. The difference between a Transmit and Receive SDO is that in case of a Receive SDO a request to obtain the contents of an object must be sent first.

Process Data Object (PDO) PDOs are used by connected nodes, for example in a twin motor configuration, to exchange real time data during operation. PDOs allow up to 8 bytes of data to be transmitted in one CAN message. This reduces the time to send the PDO data over the CAN-bus. Process data represents data that can be changing in time, such as the inputs (i.e. sensors) and outputs (i.e. motor drives) of the node controller. Process data is also stored in the object dictionary.

Network management (NMT) NMT is the final communication type defined in CANopen, and is primarily responsible for application co-ordination. Normally one node on the network acts as the NMT master. Using network management, nodes may be made 'operational' or 'pre operational' either individually or across the whole network. A node in its operational state may communicate using SDOs and PDOs, but while in its pre operational state may communicate using SDOs only This ensures that it is possible to start and stop an application in a well-defined fashion.
Other network management functions are available for the distribution of CAN identifiers for communication.
Finally an important part of network management is a life-guarding facility This works by periodically polling each node on the network to determine its operating state and to check that it is still communicating correctly Obviously a node that does not reply at all within a certain period or replies with an unexpected operating state is regarded as malfunctioning. Similarly nodes can guard the NMT master by timing the period between successive life-guarding polls. If this period is too great, the node can assume the Nh4T master is in error.

2.3.4 Communication Types

CANopen defines a number of communication classes for the input and output data (process data objects):

- Event Driven – Telegrams are sent as soon as their contents have changed. This means that the process image as a whole is not continuously transmitted, only its changes.For example, an analog input level changing by more than a specified interval or exceeding a predefined limit. The mechanism causing event transmission is configurable in the object dictionary depending on the capabilities of the device. This method of communicating process data means that CANopen networks can have extremely fast response times.

- Cyclic synchronous –Another important type of communication is CANopens synchronous communication mode.In this mode, a synchronisation telegram (SYNC) is broadcast to all nodes configured to receive it. On reception of the SYNC, some nodes produce messages for example PDOs and other nodes consume them. Using this mode, it is possible to produce PLC-type behaviour where output information is sent to various devices who then set their outputs on reception of the next SYNC telegram. It is also possible to reduce bus loading by configuring nodes to produce PDOs on reception of every nth SYNC telegram.// Synchronous communications mean that the bus traffic is very predictable, although response times are determined by the SYNC transmission period. However, both synchronous and event communication can be mixed and, in practice, most CANopen networks use a mixture of transmission types.

The desired communication type is set by the Transmission Type parameter.

2.4 SAE CAN J1939

2.4.1 Introduction to J1939

The J1939 standard is a widely used protocol, which operates on a standard CAN bus system. J1939 is a standard maintained by the Sociery of Automotive Engineers (SAE).The SAE standard was initially developed for the U.S. truck and bus industry. It has been expanded and is now the most widely used data link standard for industrial power trains, with compliance from almost all engine and transmission manufacturers.
The standard defines how information is transferred across a network to allow ECUs (i.e. computers to communicate information. (e.g. vehicle speed). In fact, J1939 provides a higher layer protocol (HLP) using CAN as a physical layer basis and so has all the architecture of CAN protocol concerning the data communication. In simple terms this means that J1939 offers a standardised method for communication across ECU's e.g. one language across all manufacturers. All J1939 enabled devices will operate on this data link and the remainder of this section details the basic requirements for J1939 communication.
The J1939 standard is defined in multiple documents corresponding to five of the seven OSI layers (figure 2.1). For example J1939-11 defines the physical layer, J1939-21 the data link and transport layer, J1939-31 the network layer and J1939-71,73,75 defines application layers. J1939-81 describes network management. A list of SAE J1939 documentation which should be used as reference when installing a J1939 network is listed below:

- SAE J1939-11 Physical Layer, 250Kbits/s, Twisted Shielded pair
- SAE J1939-15 Reduced Physical Layer, 250Kbits/s, Unshielded Twisted pair
- SAE J1939-21 Data Link Layer
- SAE J1939-31 Network Layer
- SAE J1939-71 Vehicle Application Layer
- SAE J1939-75 Application Layer Generator Sets and Industrial
- SAE J1939-73 Application Layer Diagnostics
- SAE J1939-81 Network Management

In order to be more specific and focused in our work-bench which is the scope of this diploma thesis, it will not be a thorough description for every documentation. Instead, it will be a more explanatory reference at the construction and transmission of messages as defined by J1939-21.

2.4.2 Physical Layer

The main attributes which are described and are specified in SAE J1939-11 are the next:

- The data rate is 250K bits/sec like CANopen
- Twisted pair cable, of a 120 Ohm impedance characteristic, should be used throughout. Note that most commercially available twisted pair cable is not suitable.
- It is recommended that this cable is shielded (as per J1939-11) and that the screen is grounded at a central point in the network. Unshielded twisted pair cable is used by some application manufacturers, however (as per J1939-15), offering lower cost but lower immunity to electromagnetic noise.

- The bus is linear and should be terminated with 120 Ohm resistors at either end. It is a common mistake to use one 60 Ohm resistor instead of two 120 Ohm resistors. However, this does not work correctly.
- Maximum bus length is 40 m.
- Maximum number of network nodes is 30 for J1939-11 and 10 for J1939-15.
- The terminating resistors should not be contained in network nodes.
- Network nodes are connected to the bus via stubs of maximum recommended length 1m (J1939-11) or 3m (J1939-15).

J1939 Data Bus Harness Design for Caterpiral C9.3 The data bus connector that Caterpillar uses is a modified DT connector, special wedge, cable (see A.1), and extended socket. The harness assembly requirements are unique to typical Cat wire harnesses. Caterpillar recommends 2 conductor shielded cable from Raychem Corp (Raychem part number 2019D0309-0 or Cat part number 153-2707) for all J1939 data link wiring. This is twisted pair wiring. If the Caterpillar recommended cable is not used, the cable must meet J1939 specifications for conductors (refer to table 2.4). For additional information regarding the electrical system design see the SAE publication J1939/11 Physical Layer. The minimum bend radius for the data bus cable is 40 mm. For the data bus to function as intended, the following requirements must be identified on the customer wire harness print.

1. Remove 75 mm of the outer jacket of data link shielded cable. (Reference Cat part number 153-2707)
2. Remove the foil shield from the exposed wires to within 3 mm of the cable jacket end.
3. Crimp gold-plated socket terminals to the wires and the extended socket terminal to the drain wire.
4. Slide heat shrink tube over the cable end. (Reference Cat part number 125-7876)
5. Install the terminals into the appropriate connector cavity positions.
6. Install the wedge into the connector.
7. Apply the heat shrink tube over the back of the connector body and the jacket of the cable.
8. The above components and assembly procedures must be used to ensure the cable to connector joint will be sealed. Failure to conform to these requirements will result in cable contamination and result in loss of shield performance. See Figure 17.1.

2.4.3 Network Layer

- J1939 recommends a bit sample point of 87 percent. This relatively late sampling point, which gives the best immunity to noise and propagation delay. It does restrict the size of the software jump width (SJW), however.
- All nodes must have the same bit timing.

J1939 Specifications for Conductors			
	Minimum	Nominal	Maximum
Impedance (ohm)	108	120	132
Capacitance between conductors (pF/m)	0	40	75
Capacitance between conductors and shield (pF/m)	0	70	110

Table 2.4: J1939 Conductor Specifications

- Accurate bit timing is essential (4 ms +/- 0.2 percent).
- It is recommended that the average busload is not greater than 40 percent.
- Hardware filtering (masking) of CAN messages should be used under high busload limit demands on processors.
- The engine ECU always assumes a fixed address zero. It will not change its address in the arbitration process described in J1939-81.
- The multi protocol (described in J1939-21) is used for sending messages with more than eight bytes of data. In the Caterpillar application this will be used principally for the diagnostic messages DM1, DM2, and DM3.
- Information may be broadcast or requested at regular intervals. For example, the engine will broadcast its current speed every 20ms but it will only send hours run information if another node requests it.

2.4.4 Communication Characteristics

J1939 protocol uses CAN 2.0B, an extended 29 bit identifier. Messages are identified through PGNs (Parameter Group Numbers) which comprise 18 out of 29 bit identifiers. A PGN will thus contain a number of SPNs (Suspect Parameter Numbers) in the 8 data bytes reflecting parameters (e.g. RPM). J1939 includes a large range of standard PGNs, though PGNs 00FF00 through 00FFFF are reserved for proprietary use. A data byte of 0xFF (255) reflects N/A data, while 0xFE reflects an error. Multibyte variables are sent least significant byte first (Intel byte order). J1939 supports PGNs with up to 1785 bytes using a transport protocol.

How to read the J1939 message format In order to analyse converted SAE J1939 data in human-readable form, firstly must be understood how to interpret the raw SAE J1939 packets. The basics are PGNs and SPNs

Parameter Group Number (PGN)

PGN is the SAE J1939 identifier. A PGN acts as a unique ID for looking up the function of a J1939 message and the associated data parameters (i.e.the SPNs). These information can be found in SAE J1939-71 document, which provides lists of PGNs and SPNs, as well as information on how to interpret and convert the data. You cannot match PGNs vs the full 29 bit CAN identifier. Instead, you need to separate out the 18 bit PGN as figure 2.7.

Next we look a PGN example 2.8:

We have a J1939 logged message with ID 0x0CF00401 2.9. Here, the PGN starts at bit 9. The outcome is PGN 0x0F004 or in decimal 61444. Looking this up in the SAE J1939-71 documentation we see that its the Electronic Engine Controller 1 - EEC1. Further, the document will have details on the PGN including priority, transmission rate and a list of the associated SPNs. For this PGN, there are seven SPNs (e.g. Engine Speed, RPM),

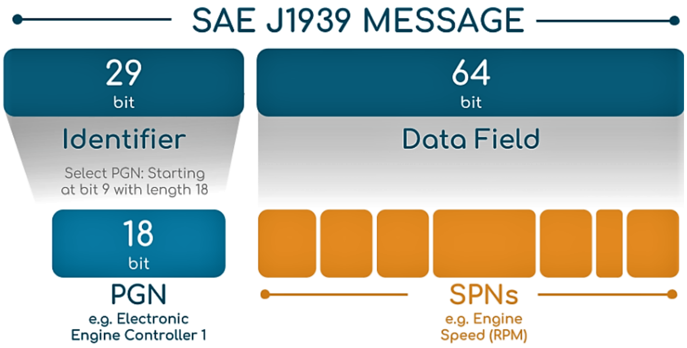
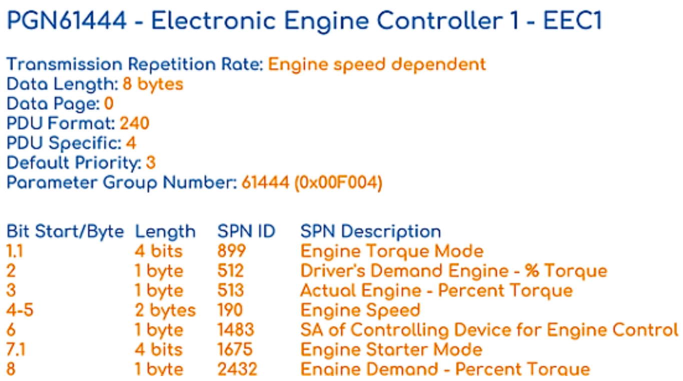


Figure 2.7: PGN and SPN in SAE J1939 message.



PGN61444 - Electronic Engine Controller 1 - EEC1

Transmission Repetition Rate: Engine speed dependent
Data Length: 8 bytes
Data Page: 0
PDU Format: 240
PDU Specific: 4
Default Priority: 3
Parameter Group Number: 61444 (0x00F004)

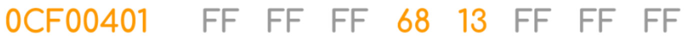
Bit Start/Byte	Length	SPN ID	SPN Description
1.1	4 bits	899	Engine Torque Mode
2	1 byte	512	Driver's Demand Engine - % Torque
3	1 byte	513	Actual Engine - Percent Torque
4-5	2 bytes	190	Engine Speed
6	1 byte	1483	SA of Controlling Device for Engine Control
7.1	4 bits	1675	Engine Starter Mode
8	1 byte	2432	Engine Demand - Percent Torque

Figure 2.8: PGN example - Electronic Engine Controller 1 - EEC1.

each of which can be looked up in the J1939-71 documentation for further details. In order to make it more simple when the J1939 29-bit identifier can be broken further down into sub components. Specifically, the ID comprises the Priority (3 bits), PGN (18 bits) and Source Address (8 bits). Further, the PGN can be broken into four parts: Reserved (1bit), Data Page (1 bit), PDU Format (8 bits) and PDU Specific (8 bits).

Suspect Parameter Number (SPN)

The SPNs of a J1939 message reflect data parameters - such as speed and RP. To make it more clear an example is given. Let's assume we've identified a PGN (e.g. 61444) based on a raw 29 bit message ID (e.g.0x0CF00401). For a given entry of this message ID, we also logged 8 bytes of raw data. In order to interpret and covert this into a human readable data we need to look at the SPN, which reflects the ID of a specific parameter contained within the data of a given PGN. For example consider SPN 190, Engine Speed, mentioned in the previous example and we are interested in converting and analysing this particular parameter. Looking into the PGN info from figure 2.8, we see that relevant data is in byte 4 and 5, i.e. 0x6813 (fig.2.9). Taking the decimal form of 0x1368 (Intel byte order), we get 4968 decimal (0x1368=4968(DEC)). To have the final value of the RPM, we conduct a scaling of this value using offset 0 and the scale 0.125 RPM/bit - 0 + (0.125*4968) = 621 RPM -. Note that some data bytes from figure 2.9, are FF or 255 decimal meaning 'not available'. While the PGN may support SPNs in this range, this specific application does not support these parameters.



```
0CF00401   FF  FF  FF  68  13  FF  FF  FF
```

Figure 2.9: Message ID example.

J1939 Request Messages

The SAE J1939 protocol supports a number of more advanced operations. These include requests, multi-packet messages, multiplexing and more.

Most J1939 messages are broadcast to the CAN bus, but some need to be requested. This is achieved using the request message (PGN 59904), which is the only 1939 message with only 3 bytes of data, it has priority 6, a variable transmit rate and can either be sent as a global or specific address request. The data bytes 1-3 should contain the requested PGN (Intel by order). Examples of requested J1939 messages include the diagnostic message (DM).

J1939 Multi-Packet Messages

The PGN and SPN examples from before are based on a J1939 message size of 8 data bytes, which will be the case for the majority of messages. However two other possible sizes exist: The 3 bytes request message above - and variable size messages. The later is used in the J1939 standard to allow communication of data packets beyond the usual 8 bytes limit of the CAN bus format. These are referred to as J1939 multi-frame or multi-packet messages. J1939 specifies how to deconstruct, transfer and readdemble the packets - process referred to as the Transport Protocol (J1939-21). Two types exist: The Connection Mode (intended for a specific device) and the BAM (Broadcast Announce Message) which is intended for the entire network. In simple terms, the BAM works by the transmitting ECU sending an initial BAM packet to set up the transfer. The BAM specifies the multi-packet PGN identifier as well as the number of data bytes and packets to be sent. It is then followed by up to 255 packets of data. Each of these 255 packets use the first data byte to specify the sequence number (1 up to 255), followed by 7 bytes of data. The max number of bytes per multi-packet message is therefore 7 bytes x 255 = 1785 bytes. The final packet will contain at least one byte of data, followed by unused bytes set to FF. In the BAM type scenario, the time between messages is 50-200 ms. Finally, a conversion software can reassemble the multiple entries of 7 data bytes into a single string and handle it according to the multi-packet PGN and SPN specifications.

Chapter 3

Experimental Facility

The HIPPO-2 hybrid diesel-electric power plant consists of a internal combustion engine (ICE) in serial connection to an electric motor (EM). In this configuration the rotational speed of the ICE and the EM are identical and the supplied torques add together to maintain the total torque demand applied by a electric motor brake (EB). In Fig. 3.1 and 3.2 the experimental hybrib powertrain of LME is presented, along with a schematic representation of the speed and torque outputs.

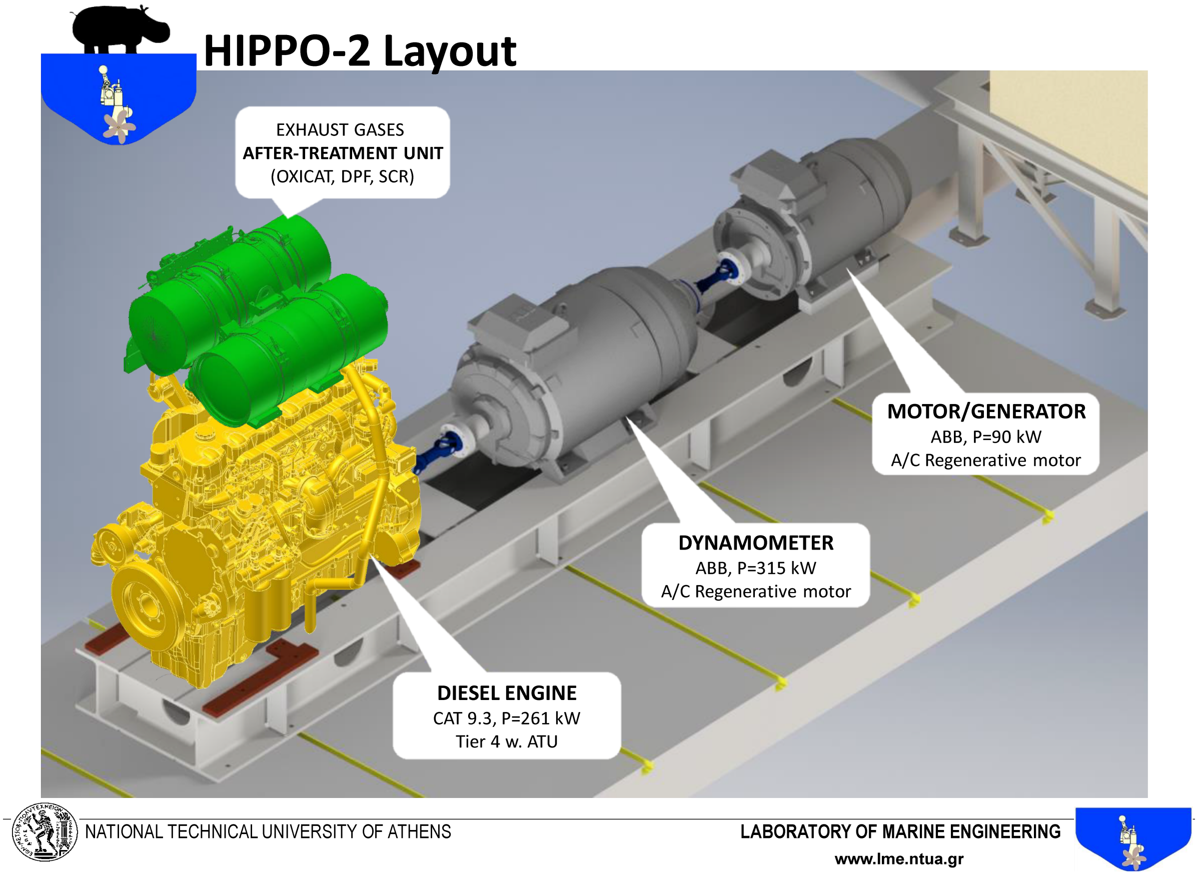


Figure 3.1: The HIPPO-2 hybrid diesel-electric testbed of LME.

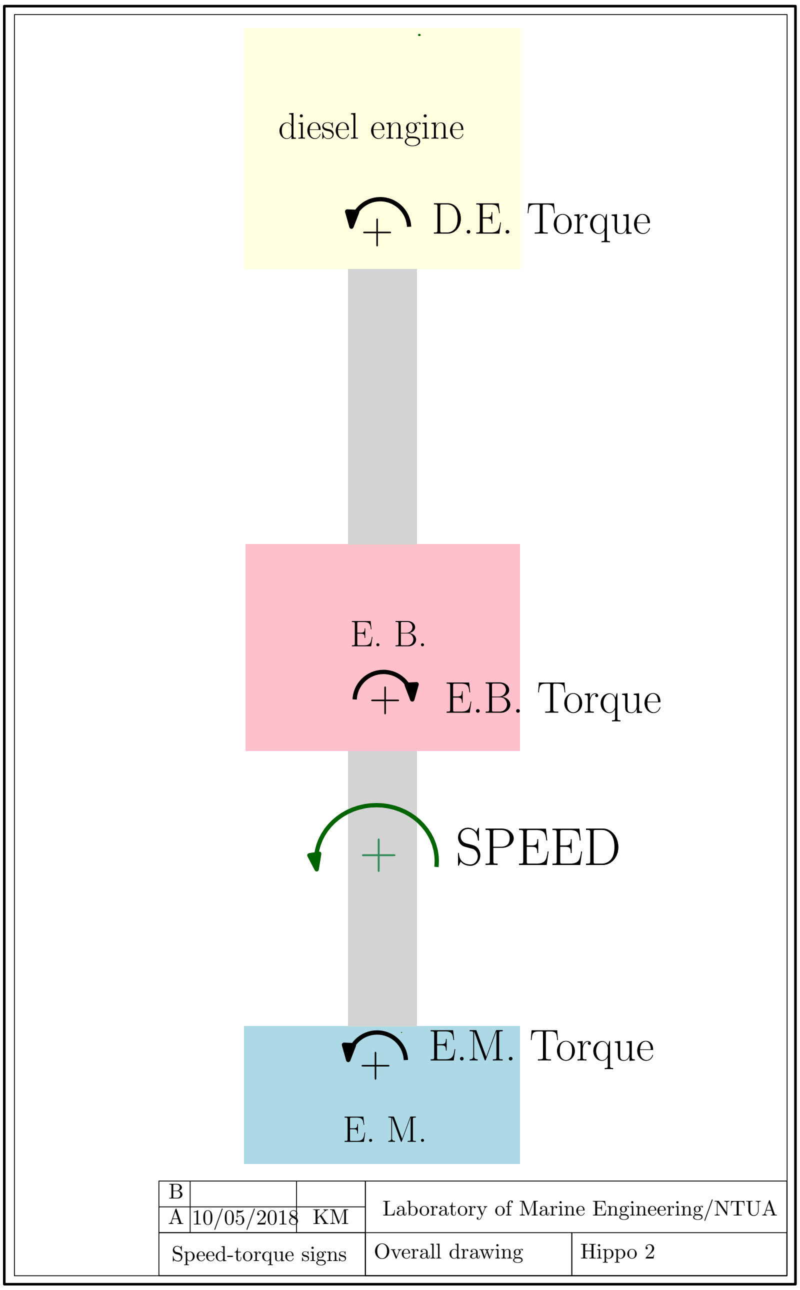


Figure 3.2: HIPPO-2 speed and torque outputs.

3.1 HIPPO-2 Integration

The ICE is a turbocharged CATERPILLAR 6-cylinder 9.3-liter 4-stroke industrial diesel engine, model C9.3 ACERT figure 3.3, producing 261 kW at 2200 rpm and maximum torque 1596 Nm at 1400 rpm. According to the speed reference and the deviation of speed measurement, the electronic control unit (ECU) of the ICE controls the fuel injection in the cylinders in closed loop control, using controller in the form of look-up tables.

The EM is a standard AC induction 3-phase motor, fig 3.4, with a rated power of 90 kW at 1483 rpm, type M3BP 280SMB 4 IMB3/IM1001, manufactured by ABB.

The EB is a standard AC induction 3-phase motor manufactured also by ABB, type M3BP 355SMB 4 IMB3/IM1001, with 315 kW load capacity, operating at 1488 rpm. A test report can be found in A.14.

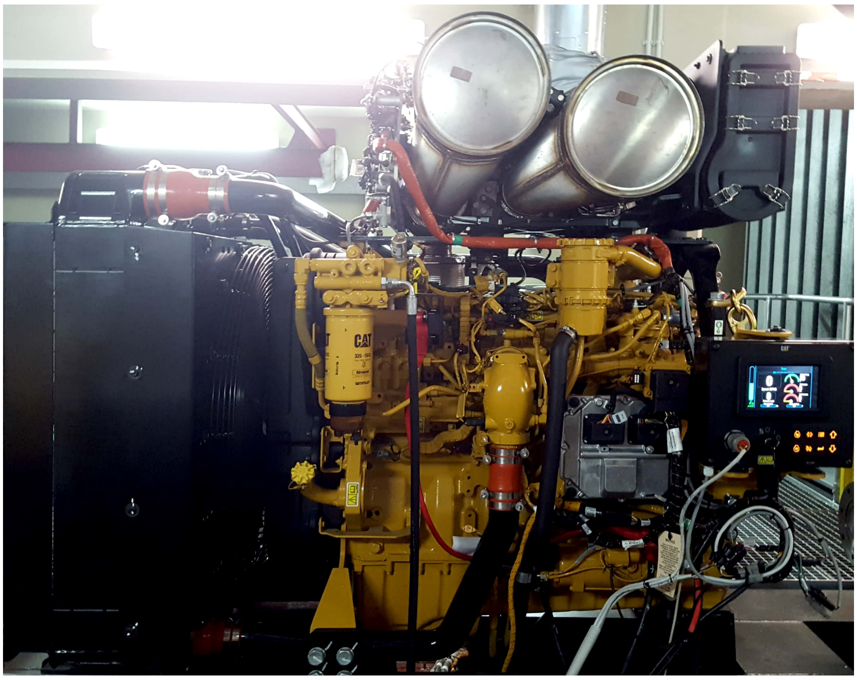


Figure 3.3: Engine CAT C9.3.

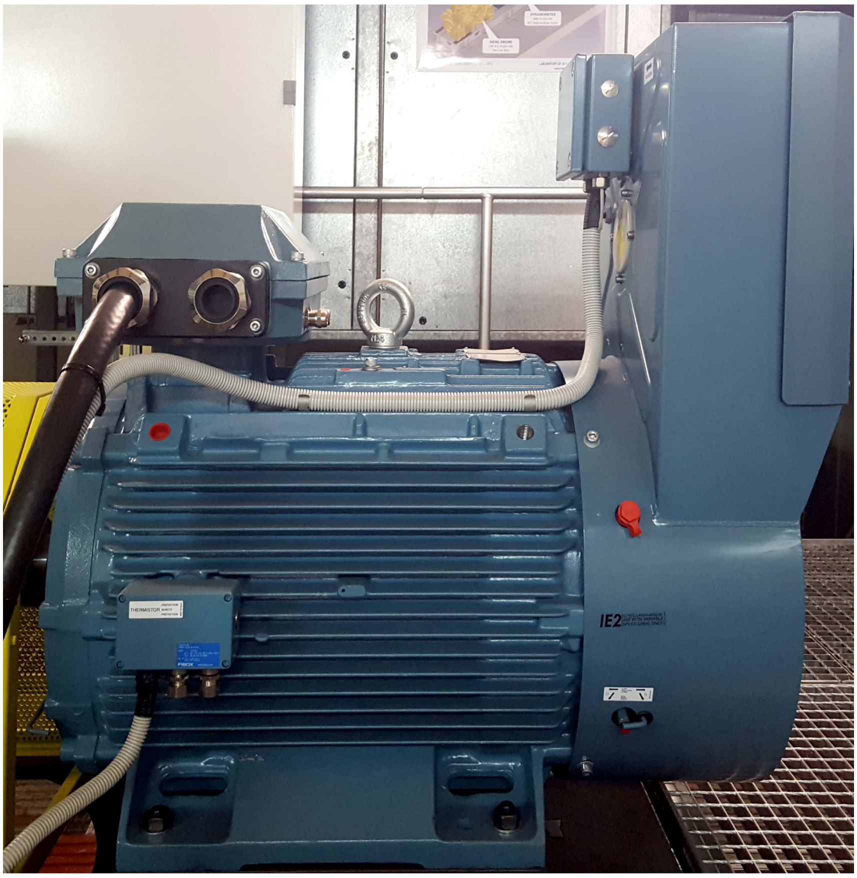


Figure 3.4: ABB Electric Motor 90 kW.

The 3 motors are connected in series, thus the operating speed range of HIPPO-2 is from 600 to 2200 rpm.

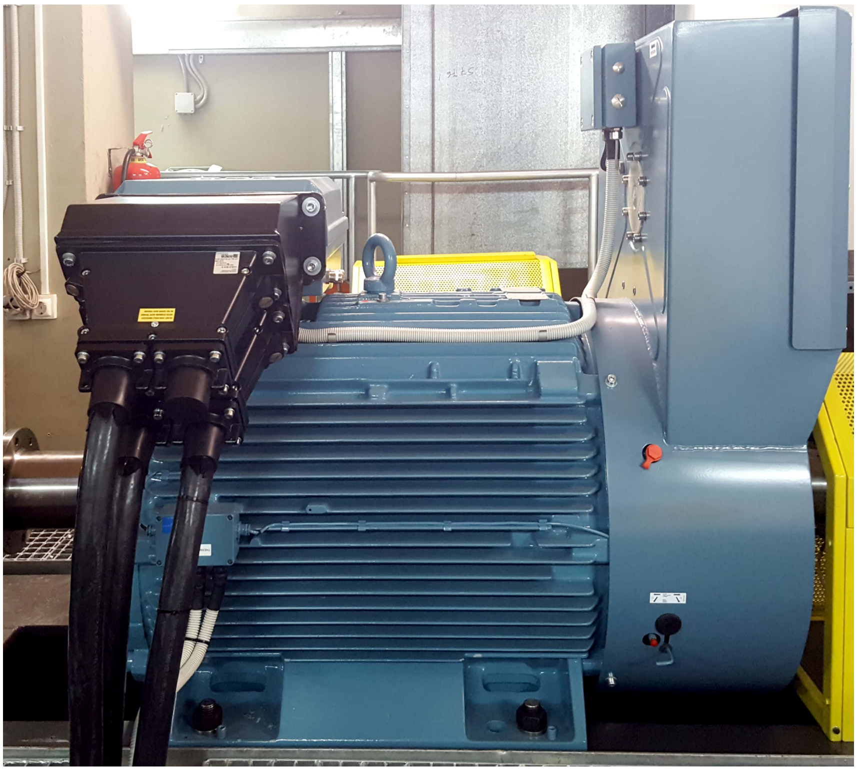


Figure 3.5: ABB Electric Brake 315 kW.

Facility's total overview is in figure 3.6.

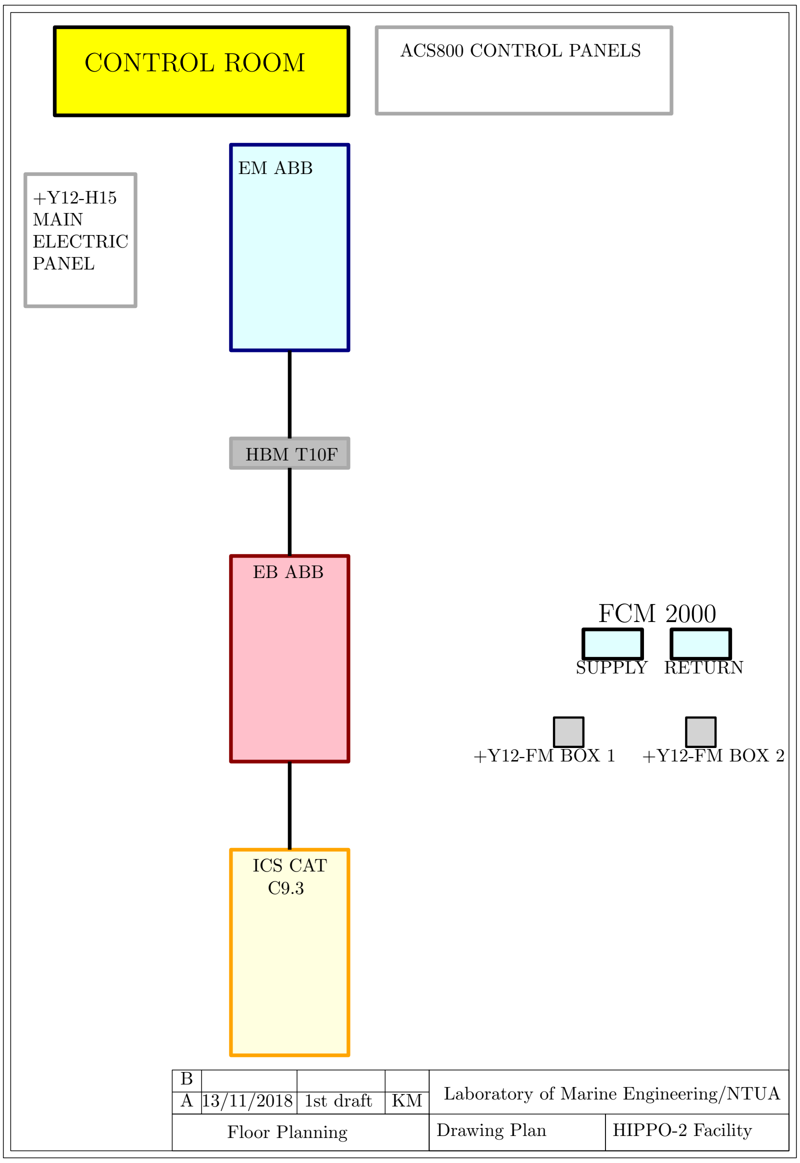


Figure 3.6: HIPPO-2 overview.

3.2 Sensors and Data Acquisition System

The installed sensors in the diesel engine are presented in Fig. A.9 and A.10. The NOx and λ values are provided by a *SmartNOx* sensor in the manifold downstream of the turbocharger (TC). Fuel mass flow measurements are provided by two ABB Coriolis flowmeters, one at supply and one at return fuel lines. TC speed and intake manifold pressure are also measured.

3.2.1 Coriolis Mass Flowmeter FCM2000

The fuel mass flow from the diesel tank as well the fuel return are measured by two identical Coriolis flowmeters manufactured by ABB, fig. 3.7. Each of the Coriolis flow meters has one panel for data acquisition, fig. 3.8.

The Model number of these two Coriolis flowmeters is ME23A3AA1GEC, fig. A.16.

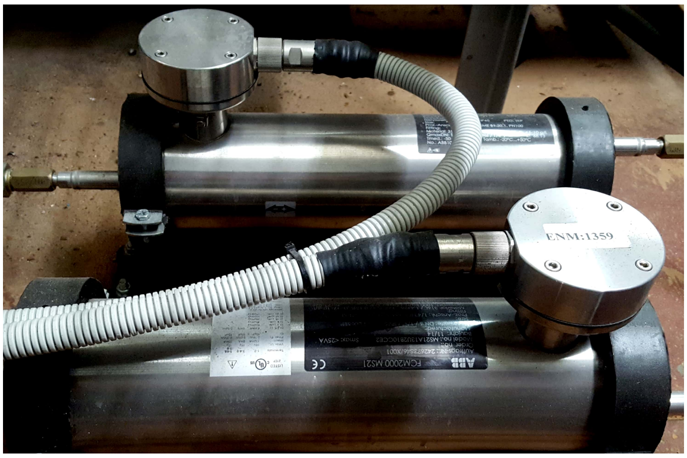


Figure 3.7: ABB Coriolis flowmeters.

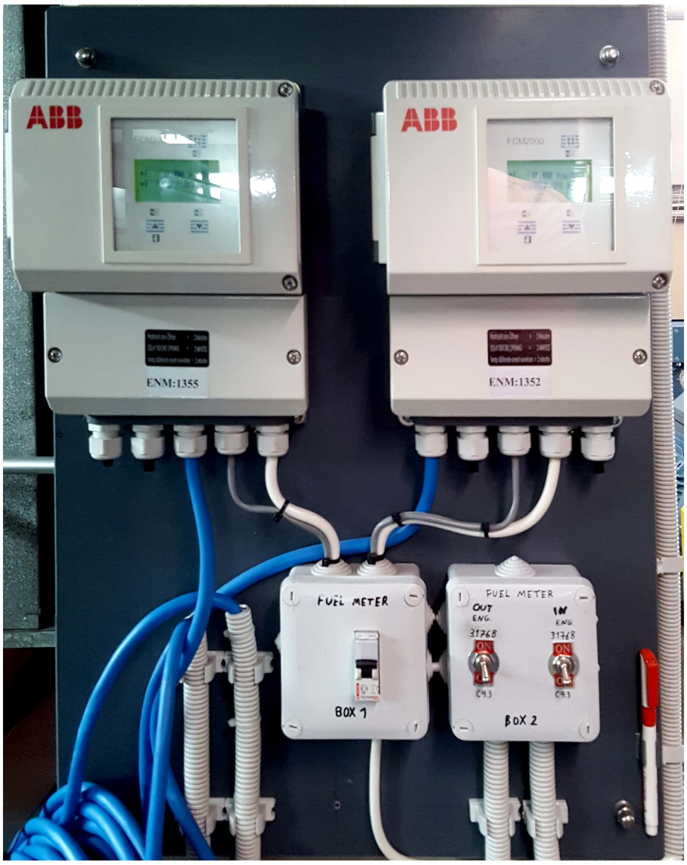


Figure 3.8: ABB Coriolis panel.

3.2.2 HBM T10F torque meter

Between EM and EB, a torque meter, HBM T10F, is installed (fig.3.9).

HBM torque meter T10F has 3 connectors. At HIPPO-2, connected for speed measurement is connector 2 while for torque measurement is connector 3. More specific info about the pins and their terminal connections can be found in figure A.15.

Figure 3.9: HBM T10F torque meter.

3.3 ABB engine controllers

The speed and torque outputs for both EM and EB are being controlled and supervised by ABB drives. The type of ABB driver which has been used for EB and EM is ACS800 (fig 3.10).

The data acquisition and the control of the ABB engines are achieved through ACS800 panels and CANopen bus protocol. The data of each cabinet driver is shown in table A.2 for EM and in table A.3 for EB.

3.4 Description of Data Acqutision System

The data acquisition system of HIPPO 2 consists of:

1. DSpace Microautobox II
2. CAN bus
3. RCAN modules
4. ABB Drivers
5. Signalling Cables for I/O
6. Ethernet cable

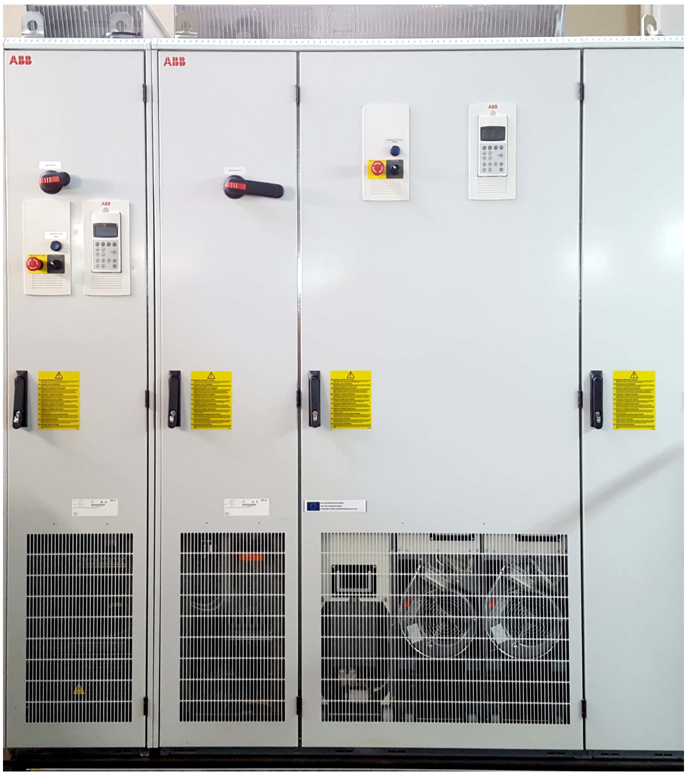


Figure 3.10: ABB ACS 800 controlling panels.

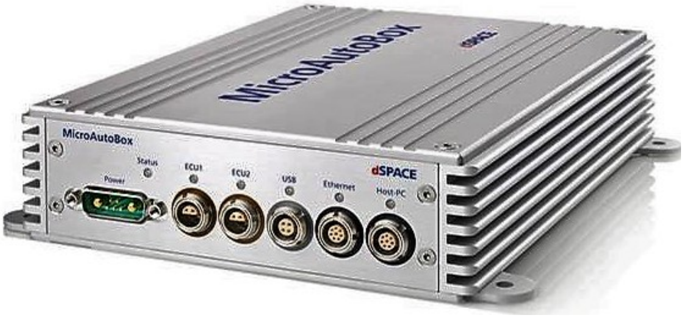


Figure 3.11: The HIPPO-2 dSpace monitoring and control board.

The platform for Data Acquisition and control of the powertrain is based on the dSpace Micro Autobox II DS1401/1511 (Fig. 3.11) controller board, with rapid control prototyping capability, programmed under the MATLAB/Simulink environment.

The simplicity of communication topography is depicted in figure 3.12.

dSMA II is a real-time system for performing fast function prototyping in fullpass and bypass scenarios. It operates without user intervention, just like an ECU. It has the ability to connect with many sensors from the testbed through of its Analog, Digital and CAN ports. In fact, with dSMA II we can read all the data from any sensor which is mounted on C9.3 diesel as well n ABB electric motor through CAN bus. The backbone of CAN bus is a twisted pair cable; in our case we used 3 CAN channels from dSpace. The first is CAN A from CAT engine's ECU and is connected to CAN 2 in dSMA II connector, the second is ABB CAN from RCAN modules which is connected to CAN 3 in dSMA II; the third CAN channel, CAN 4, has been installed for future use and is connected to AT's ECU.

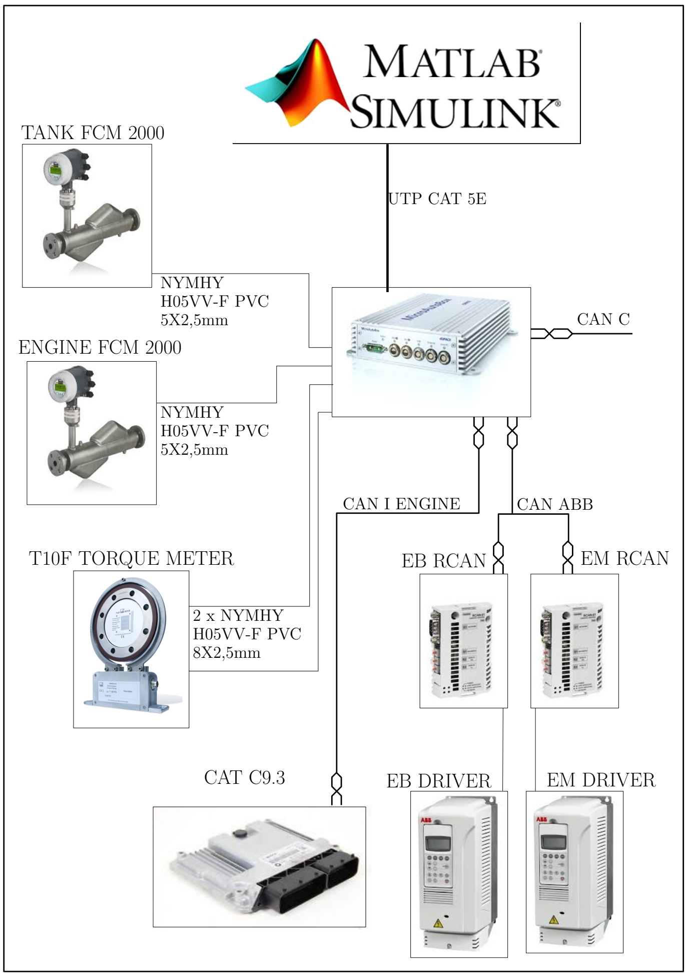


Figure 3.12: The general arrangement of data acquisition system for HIPPO-2.

The CAT C9.3 engine together with the two ABB motors have almost 150 sensors and 700 signals that can be acquired and used for monitoring every parameter of the HIPPO-2 test bench. It is very important for the prototype test bench to have the capability to monitor a diverse set of parameters for basic engine, TC, AT as well as for those parameters related to the operation of electric drives. Taking into consideration all the above complexity of sensoring, the most efficient way of connecting and controlling the motors is through CAN bus.

In figure 3.13 is HIPPO-2's signal cable connection overview.

3.4.1 Electrical connection

HIPPO_2 has one main electric panel '+Y12-H15', three smaller electric boxes for Coriolis flow meters and ACS 800 panels.

FCM 2000 Electric connection

Figure A.2 shows the drawing of two FCM 2000 panels.

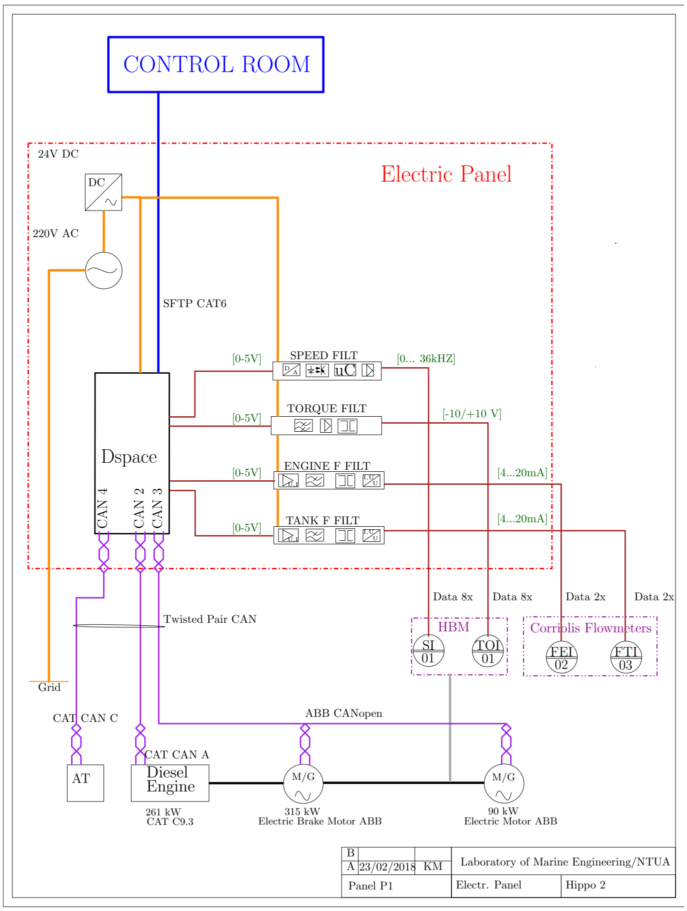


Figure 3.13: The general design of data acquisition system.

+Y12-H15 Electric Panel

In the main electric panel are connected the analog signal cables, the CAN bus cables and their filters with dSpace. The main control of HIPPO_2 is done by Control Room with only one Ethernet cable which connect the PC from Control Room with the dSpace MicroAutobox II. In fig 3.14 is the picture of +Y12-H15 and in fig A.3,A.4,A.5 the three electric drawings for each rack inside panel +Y12-H15. The CAN bus connection between ICE C9.3 and the electric panel is made with a twisted pair and like every connection of CAN bus cables, Yellow or White cable is CAN_H and Green or Brown is CAN_L.

ZIF Connector and Power Supply of dSpace MicroAutobox II

The figure A.11 shows the electric cabling of Zero Insertion Force connector of MicroAutobox II and the figure A.13 is the schematic of power supply connection for dSpace MicroAutobox II.

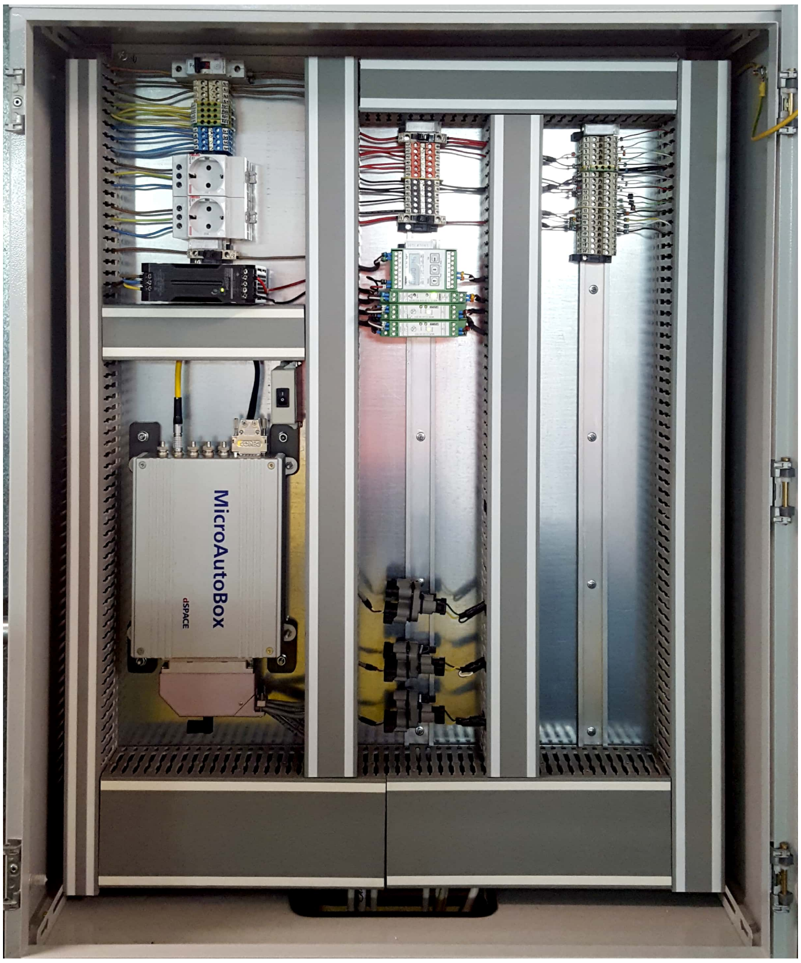


Figure 3.14: +Y12-H15 Electric Panel.

Electric boxes

Except from main electric panel, the test bed has 3 more small electric boxes used for connecting the Coriolis flow meters with Hippo-2 and Hippo-1 if needs to operate. The first box called Fuel Meter Box 1 (FM BOX 1) and the electric drawing is in figure A.6. The FM BOX 2 used for the selection of the operating test bed; its electric drawing is shown in figure A.7. The third box, +Y12-BOX1 fig.A.8, was installed for HIPPO-1 but is referred here for completion.

Chapter 4

HIPPO 2 data acquisition

4.1 Introduction

In this chapter the integration of the controllers, sensors and actuators which are connected to dSMA II will be analysed. The integration is done through ControlDesk and Matlab. Also the initialisation of DSpace control desk using Matlab/Simulink will be thoroughly described. All CAN messages and analog signals are obtained and processed by dSpace and Control Desk, the latter programmed with Matlab/Simulink. At the beginning of this chapter the procedure of the programming Control Desk in order to receive and send messages from and to HIPPO-2 will be described. Subsequently, the final view from Control Desk which is used to operate HIPPO 2 test bed is depicted. This chapter ends up with an example from experiments.

4.2 CAN message communication setup

In order to setup the communication in dSpace, Matlab R2015a with configured dSpace software for MATLAB 8.5.0.197613 (R2015a) is used. The blocks for CAN signals in Simulink can be found from simulink library browser in dSPACE RTI CAN blockset. A Simulink program named HIPPO_2 has been created, with the main blocks as in fig.4.1 and described as follows.

- RTI Data, which is used by Real Time Interface to store internal model-dependent data
- RTI_CAN1
- CAN_open ABB
- ANALOG INPUTS
- HBM_FLANGE
- ABB FLOWMETER

Each block refers to each different controller or device. ICE CAT C9.3 refers to RTI_CAN1; ABB machines (brake and motor) refer to CANopen_ABB; HBM torque flange refers to HBM_FLANGE and both Coriolis flow meters refer to ABB FLOWMETER. A block also exists for ANALOG INPUTS, which refers to dSpace's analog inputs.

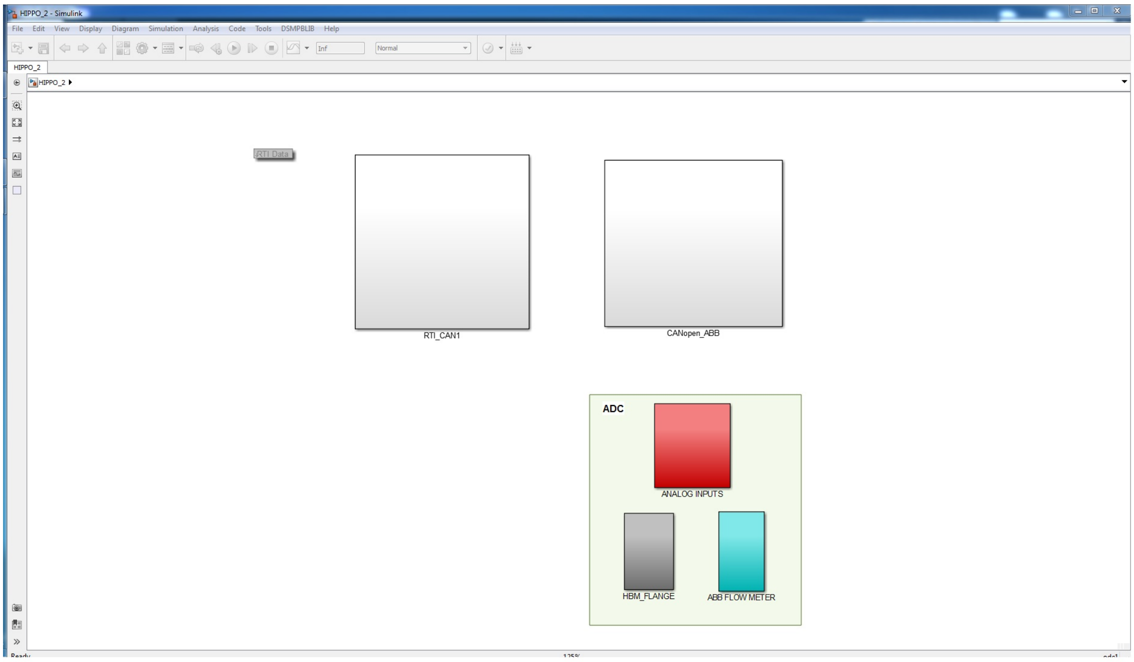


Figure 4.1: Simulink main blocks.

4.2.1 Programming of RTI_CAN1

Inside the block of RTI_CAN1, there are all the blocks of receiving and transmitting messages between dSpace and CAN bus from CAT C9.3. In figure 4.2 the blocks of RTI_CAN1 are shown in abstract form. (More detailed pictures are shown in fig. 4.3, 4.4, 4.5).

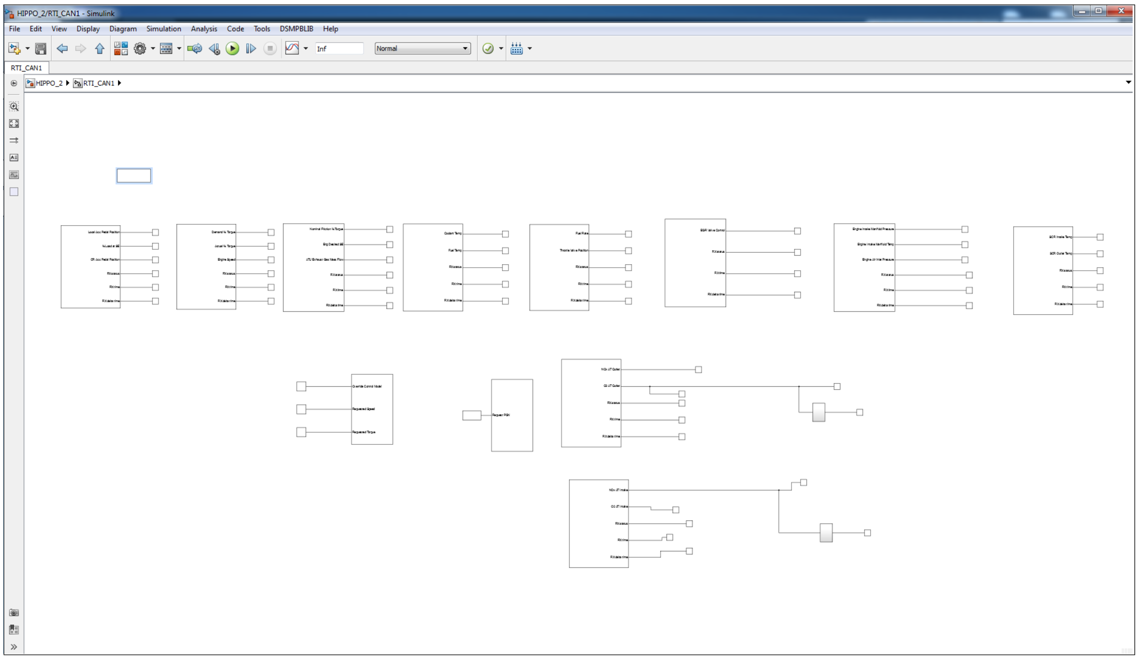

HIPPO_2/RTI_CAN1 - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools DSMPBLIB Help
Inf
Normal
RTI_CAN1
HIPPO_2
RTI_CAN1

Figure 4.2: RTI CAN1 general overview.

In figure 4.3 the messages for Parameter Group EEC1, EEC2, EEC3, ET1 and LFE1 as well the CAN Controller set-up box which is necessary for CAN communication are shown.

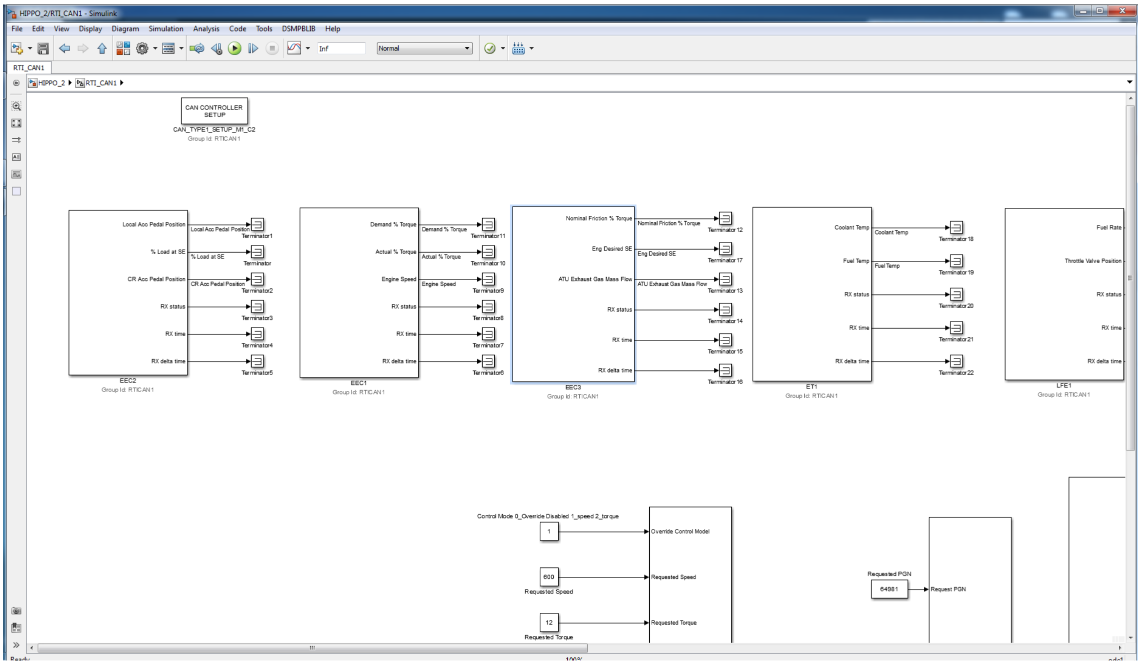


Figure 4.3: RTI CAN1 detailed (1 of 3).

In figures 4.4 and 4.5 there are the rest receive Parameter Groups from C9.3 engine and two RTI CAN transmit messages.

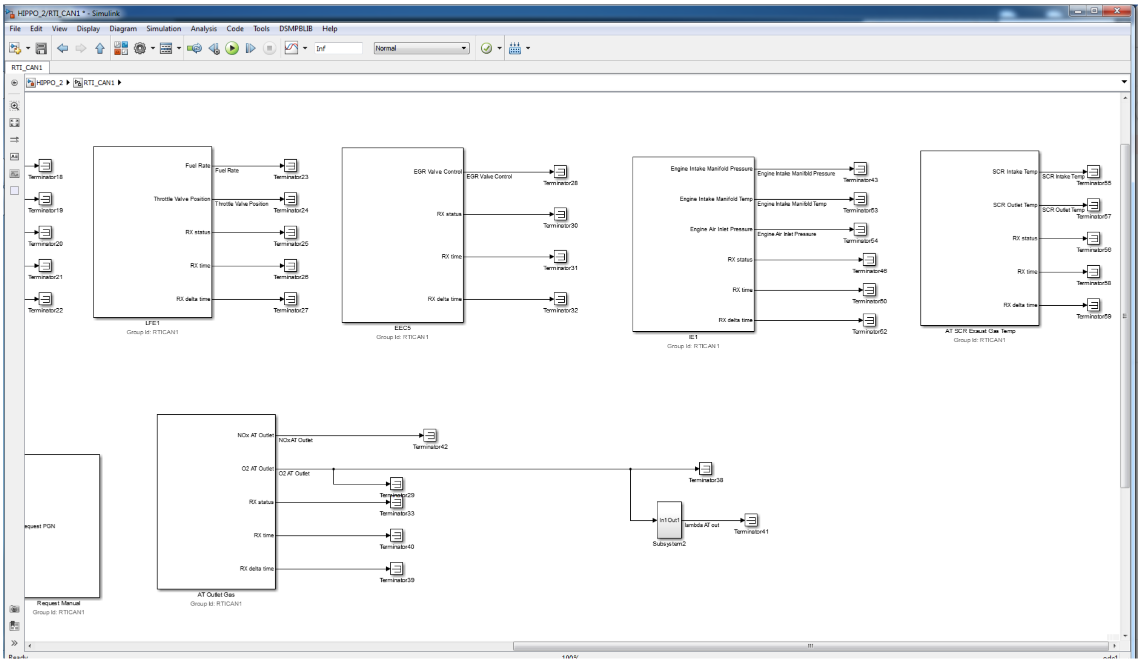


Figure 4.4: RTI CAN1 detailed (2 of 3).

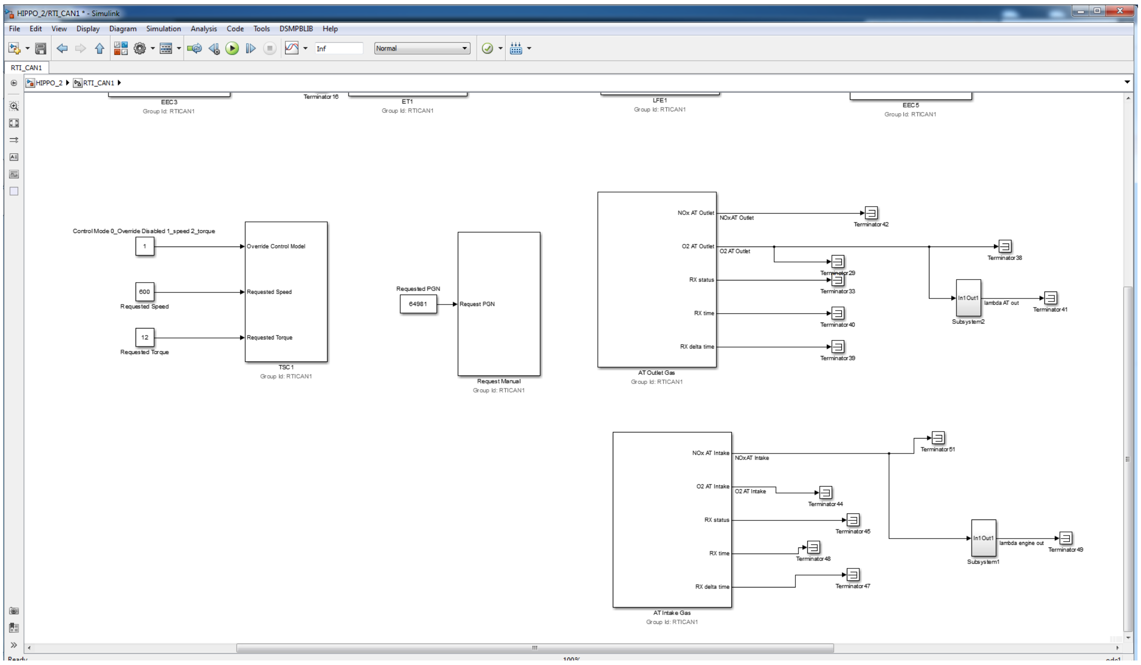


Figure 4.5: RTI CAN1 detailed (3 of 3).

CAN Controller setup

The establishment of communication between CAN SAE J1939 and dSpace Controlldesk is done by configuring the Matlab RTI CAN controller Setup box (fig. 4.6).

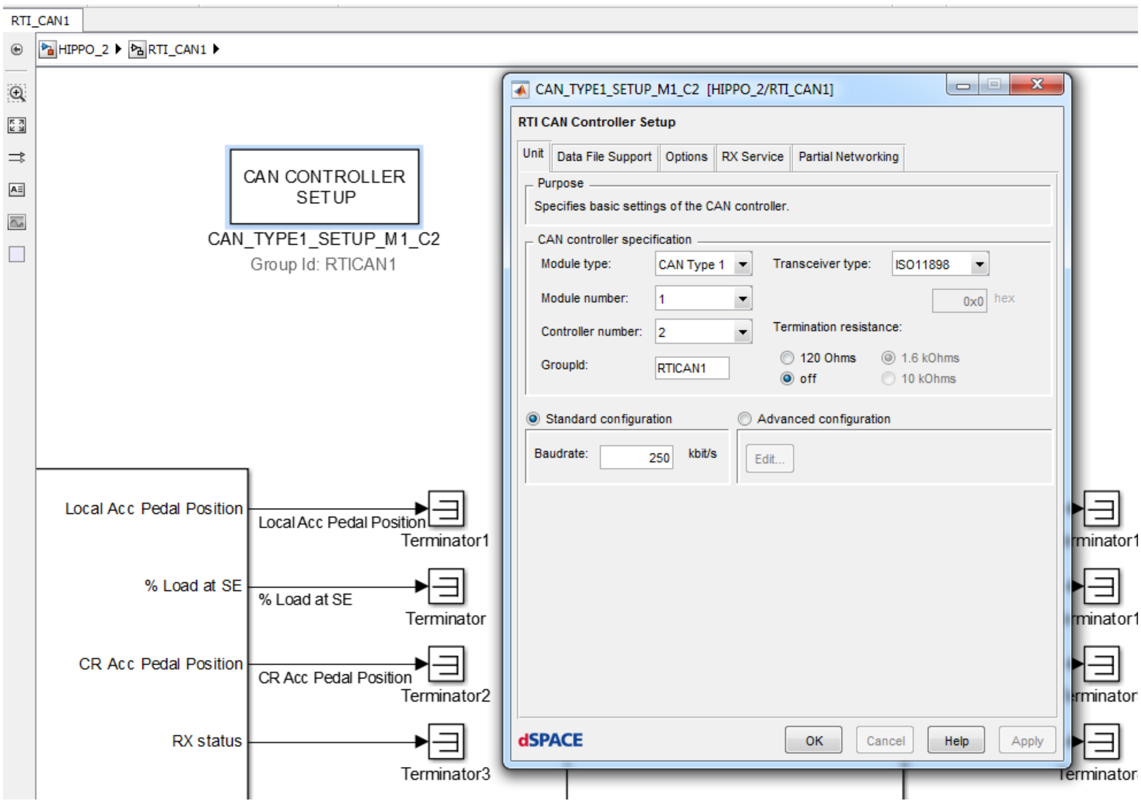
The parameters that are configured are :

- Module Type
- Module Number
- Controller Number
- Groupid

MicroAutobox II has 4 signal I/O ports for CAN and is configured accordingly from RTI CAN controller.

- CAN 1 : Module Number is 1 and Controller Number is 1
- CAN 2 : Module Number is 1 and Controller Number is 2
- CAN 3 : Module Number is 2 and Controller Number is 1
- CAN 4 : Module Number is 2 and Controller Number is 2

The baudrate of SAE J1939 is 250 kbit/s.



CAN_TYPE1_SETUP_M1_C2 [HIPPO_2/RTI_CAN1]

RTI CAN Controller Setup

Unit | Data File Support | Options | RX Service | Partial Networking

Purpose

Specifies basic settings of the CAN controller.

CAN controller specification

Module type: CAN Type 1

Transceiver type: ISO11898

Module number: 1

0x0 hex

Controller number: 2

Termination resistance: ○ 120 Ohms ● off ● 1.6 kOhms ○ 10 kOhms

GroupId: RTICAN1

● Standard configuration ○ Advanced configuration

Baudrate: 250 kbit/s

Figure 4.6: CAN Controller set up.

SAE J1939 receive message setup

By the term 'receive' the sent message from CAT C9.3 and received by dSMA II is declared. In order to set the message parameters of SAE J1939 in dSpace first one looks at the Parameter Group of J1939 supported parameters from Electronics Application and Installation Guide in Data Link Support chapter. In figure 4.7 there is an example of PGN Electronic Engine Control 1.

Parameter Groups contain information on parameter assignments within the 8-byte CAN data field of each message as well repetition rate and priority.

The following is an example of a parameter group definition as listed in SAE J1939/71:

PGN 61444 - Electronic Engine Control 1

Transmission Rate: Engine speed dependent

Data Length: 8 bytes

Default Priority: 3

PG Number: 61444 (F004hex)

Description of Data

Byte 1.1: Engine Torque Mode SPN 899

Byte 1.5: Actual Engine- Percent Torque High Resolution SPN 4154

Byte 2: Driver's Demand Engine - Percent Torque **SPN 512**

Byte 3: Actual Engine - Percent Torque - **SPN 513**

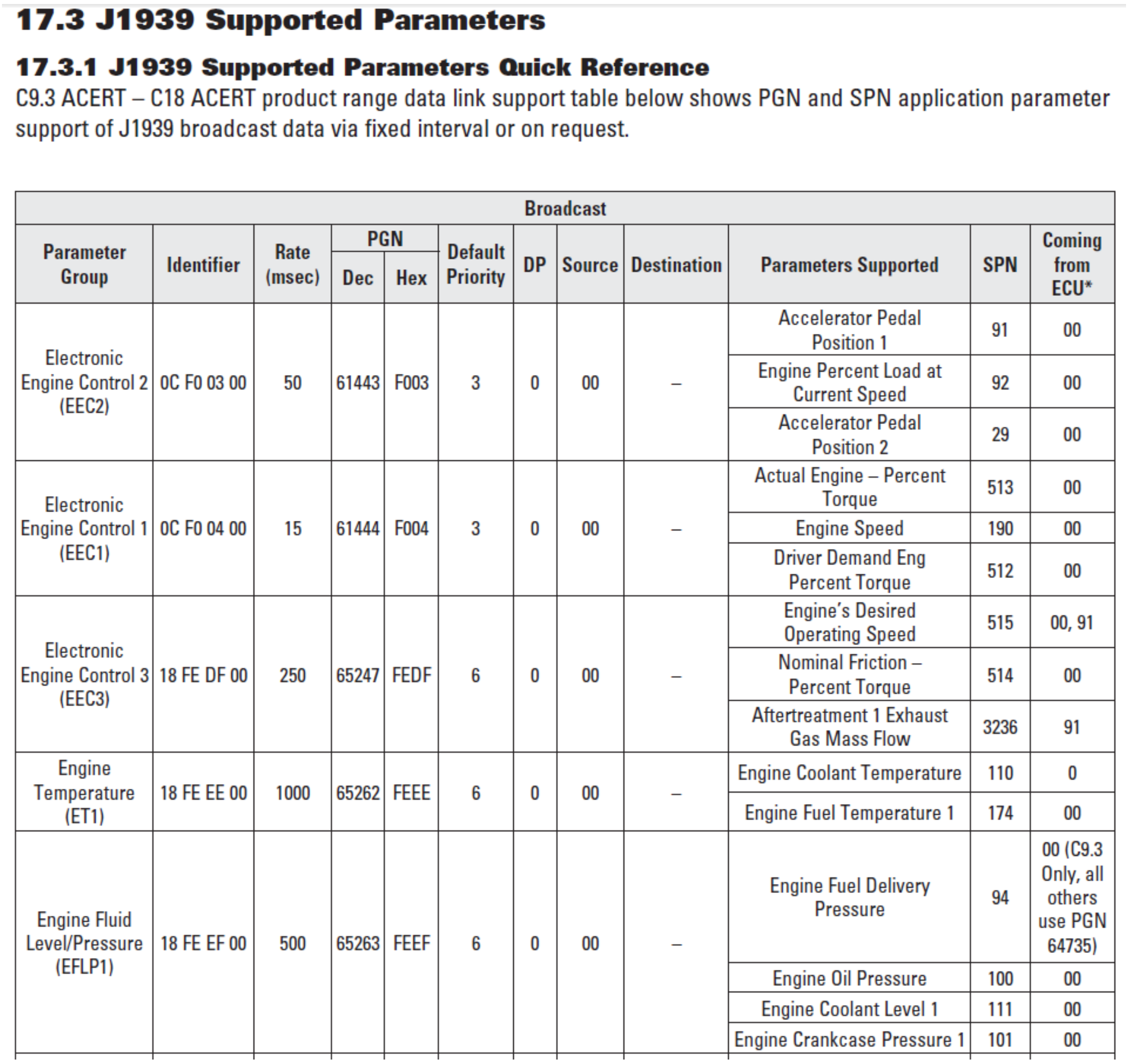
Byte 4,5: Engine Speed - **SPN 190**

Byte 6: Source Address of Controlling Device for Engine Control - SPN 1483

Byte 7: Engine Starter Mode - SPN 1675

Byte 8: Engine Demand - Percent Torque - SPN 2432

In bold are the SPN that are used from C9.3 ICE engine.



17.3 J1939 Supported Parameters

17.3.1 J1939 Supported Parameters Quick Reference

C9.3 ACERT – C18 ACERT product range data link support table below shows PGN and SPN application parameter support of J1939 broadcast data via fixed interval or on request.

Broadcast											
Parameter Group	Identifier	Rate (msec)	PGN		Default Priority	DP	Source	Destination	Parameters Supported	SPN	Coming from ECU*
			Dec	Hex							
Electronic Engine Control 2 (EEC2)	0C F0 03 00	50	61443	F003	3	0	00	–	Accelerator Pedal Position 1	91	00
									Engine Percent Load at Current Speed	92	00
									Accelerator Pedal Position 2	29	00
Electronic Engine Control 1 (EEC1)	0C F0 04 00	15	61444	F004	3	0	00	–	Actual Engine – Percent Torque	513	00
									Engine Speed	190	00
									Driver Demand Eng Percent Torque	512	00
Electronic Engine Control 3 (EEC3)	18 FE DF 00	250	65247	FEDF	6	0	00	–	Engine's Desired Operating Speed	515	00, 91
									Nominal Friction – Percent Torque	514	00
									Aftertreatment 1 Exhaust Gas Mass Flow	3236	91
Engine Temperature (ET1)	18 FE EE 00	1000	65262	FEEE	6	0	00	–	Engine Coolant Temperature	110	0
									Engine Fuel Temperature 1	174	00
Engine Fluid Level/Pressure (EFLP1)	18 FE EF 00	500	65263	FEEF	6	0	00	–	Engine Fuel Delivery Pressure	94	00 (C9.3 Only, all others use PGN 64735)
									Engine Oil Pressure	100	00
									Engine Coolant Level 1	111	00
									Engine Crankcase Pressure 1	101	00

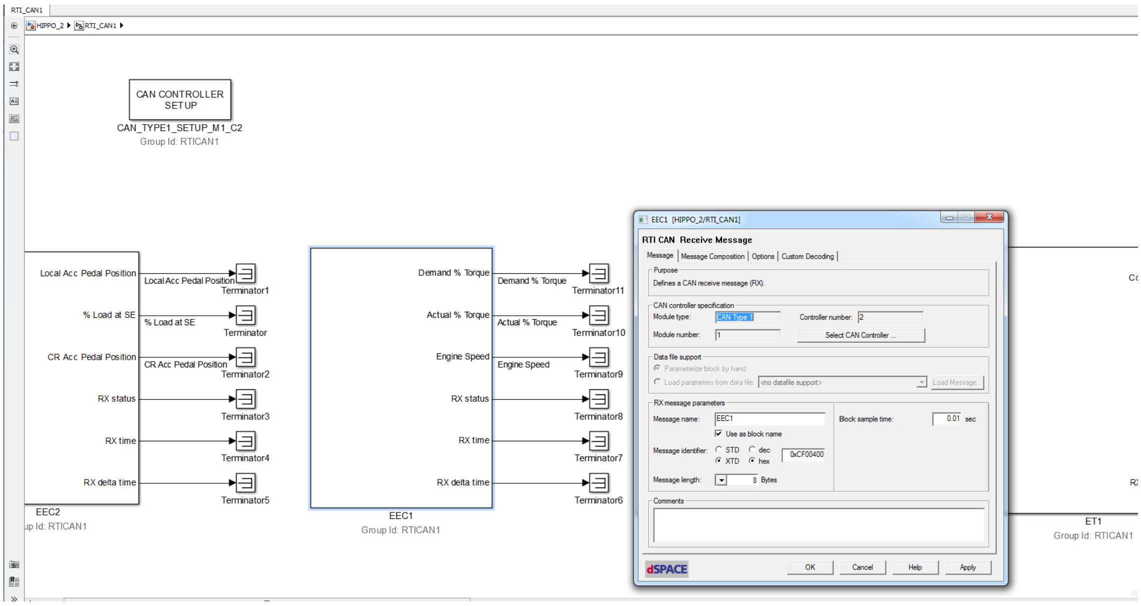
Figure 4.7: J1939 Supported Parameters example.

In order to set up the message acquisition in Simulink, a new RTI CAN Receive message block should be created, in which the next areas should be set up, fig.4.8 :

- Message Name
- Message Identifier
- Message Length

The next parameters need to be full filed are from Message Composition table figure 4.9 . In that table configuring the signal (SPN) parameters which are:

- Signal Name (e.g. Actual% Torque)
- Start Bit (SB) (e.g.16)
- Signal Length (SL) (e.g. 8 bits)
- Factor(e.g. 1)
- Offset (e.g. -125)



EEC1 [HIPPO_2/RTI_CAN1]

RTI CAN Receive Message

Message | Message Composition | Options | Custom Decoding

Purpose: Defines a CAN receive message (RX).

CAN controller specification
Module type: CAN Type 1
Controller number: 2
Module number: 1
Select CAN Controller ...

Data file support
● Parameterize block by hand
○ Load parameters from data file: <no datafile support>
Load Message...

RX message parameters
Message name: EEC1
☑ Use as block name
Message identifier: ○ STD ● XTD ○ dec ● hex 0xCF00400
Message length: 8 Bytes
Block sample time: 0.01 sec

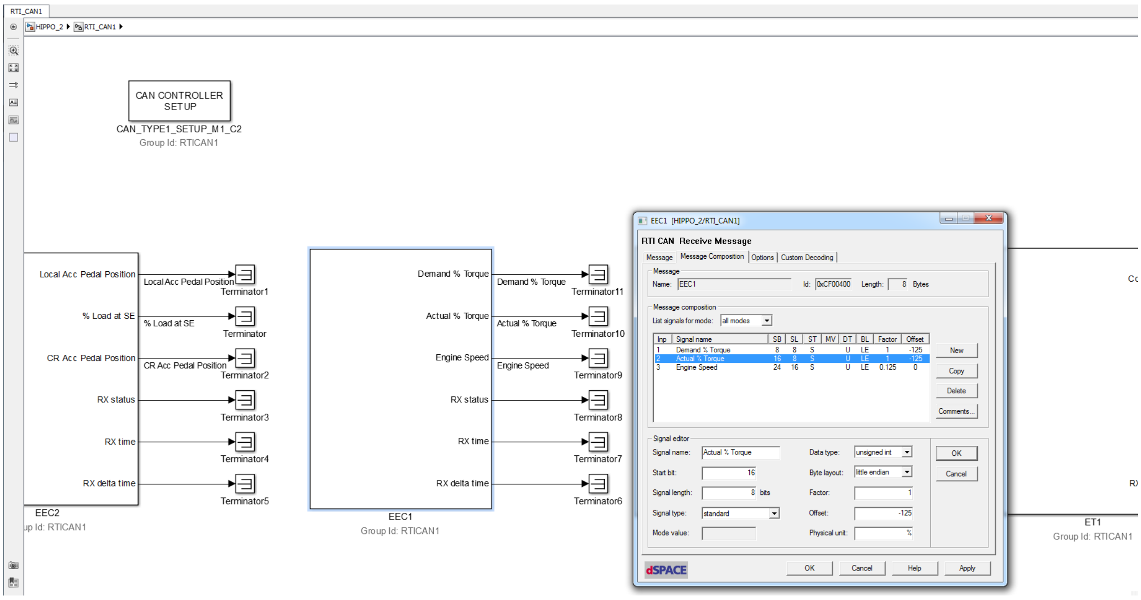
Comments

dSPACE OK Cancel Help Apply

Figure 4.8: EEC1 set up example 1/2.

- Physical unit (e.g. %)

After setting up all the parameters the signals are terminated.



EEC1 [HIPPO_2/RTI_CAN1]

RTI CAN Receive Message

Message | Message Composition | Options | Custom Decoding

Message
Name: EEC1 Id: 0xCF00400 Length: 8 Bytes

Message composition
List signals for mode: all modes

Inp	Signal name	SB	SL	ST	MV	DT	BL	Factor	Offset
1	Demand % Torque	8	8	S		U	LE	1	-125
2	Actual % Torque	16	8	S		U	LE	1	-125
3	Engine Speed	24	16	S		U	LE	0.125	0

Signal editor
Signal name: Actual % Torque
Start bit: 16
Signal length: 8 bits
Signal type: standard
Mode value:
Data type: unsigned int
Byte layout: little endian
Factor: 1
Offset: -125
Physical unit: %

dSPACE OK Cancel Help Apply

Figure 4.9: EEC1 set up example 2/2.

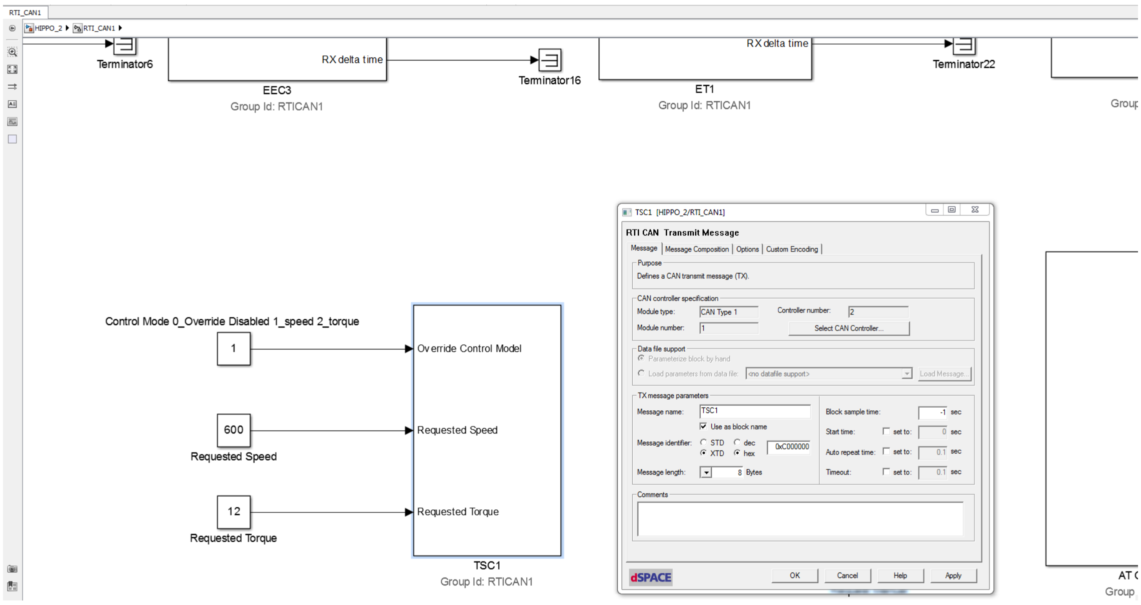
SAE J1939 transmit message setup

The transmit messages are configured similarly to receive messages.

To set up the message transmit in Simulink, one should create a new RTI CAN Transmit message block in which the next areas are configured, fig.4.10 :

- Message Name

- Message Identifier
- Message Length



TSC1 [HIPPO_2/RTI_CAN1]

RTI CAN Transmit Message

Message | Message Composition | Options | Custom Encoding

Purpose
Defines a CAN transmit message (TX).

CAN controller specification
Module type: CAN Type 1 Controller number: 2
Module number: 1 Select CAN Controller...

Data file support
● Parameterize block by hand
○ Load parameters from data file: <no datafile support> Load Message...

TX message parameters
Message name: TSC1
☑ Use as block name
Message identifier: ○ STD ● XTD ○ dec ● hex 0xC000000
Message length: 8 Bytes
Block sample time: -1 sec
Start time: □ set to: 0 sec
Auto repeat time: □ set to: 0.1 sec
Timeout: □ set to: 0.1 sec

Comments

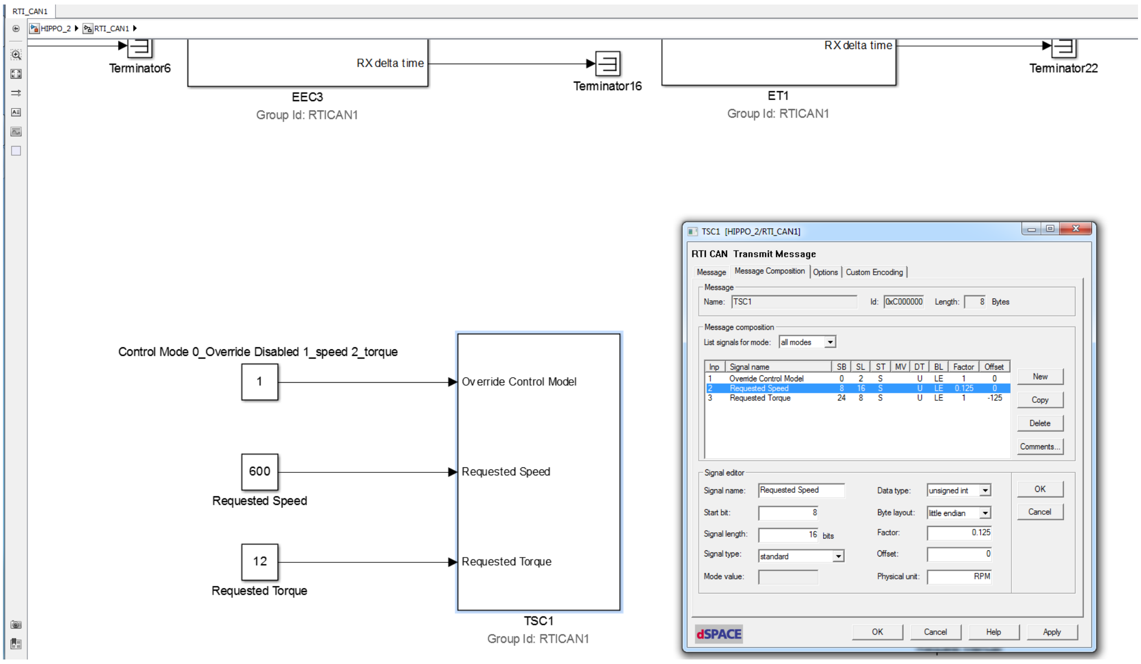
OK Cancel Help Apply

Figure 4.10: TSC 1 configuring example 1/2.

The next parameters which need to be configured are from Message Composition table figure 4.11. In that table, the signal (SPN) parameters that are configured are:

- Signal Name (e.g. Requested Speed)
- Start Bit (SB) (e.g.8)
- Signal Length (SL) (e.g. 16 bits)
- Factor(e.g. 0.125)
- Offset (e.g. 0)
- Physical unit (e.g. RPM)

After setting up all the parameters, one should create an input box for each input. In table A.1 the signals of CAN SAE J1939 are shown.



TSC1 [HIPPO_2/RTI_CAN1]

RTI CAN Transmit Message

Message

Name: TSC1 Id: 0xC000000 Length: 8 Bytes

Message composition

List signals for mode: all modes

Inp	Signal name	SB	SL	ST	MV	DT	BL	Factor	Offset
1	Override Control Model	0	2	S		U	LE	1	0
2	Requested Speed	8	16	S		U	LE	0.125	0
3	Requested Torque	24	8	S		U	LE	1	-125

Signal editor

Signal name: Requested Speed
Data type: unsigned int
Start bit: 8
Byte layout: little endian
Signal length: 16 bits
Factor: 0.125
Signal type: standard
Offset: 0
Mode value:
Physical unit: RPM

Figure 4.11: TSC 1 configuring example 2/2.

4.2.2 Programming of RTI CANopen ABB

CANopen and J1939 are two different higher layer protocols but with the same low layer CAN protocol. Thus the configuration of the communication has similar attributes. To begin with, inside the CANopen_ABB box (fig. 4.12) is the CAN Controller Setup box (fig. 4.13).

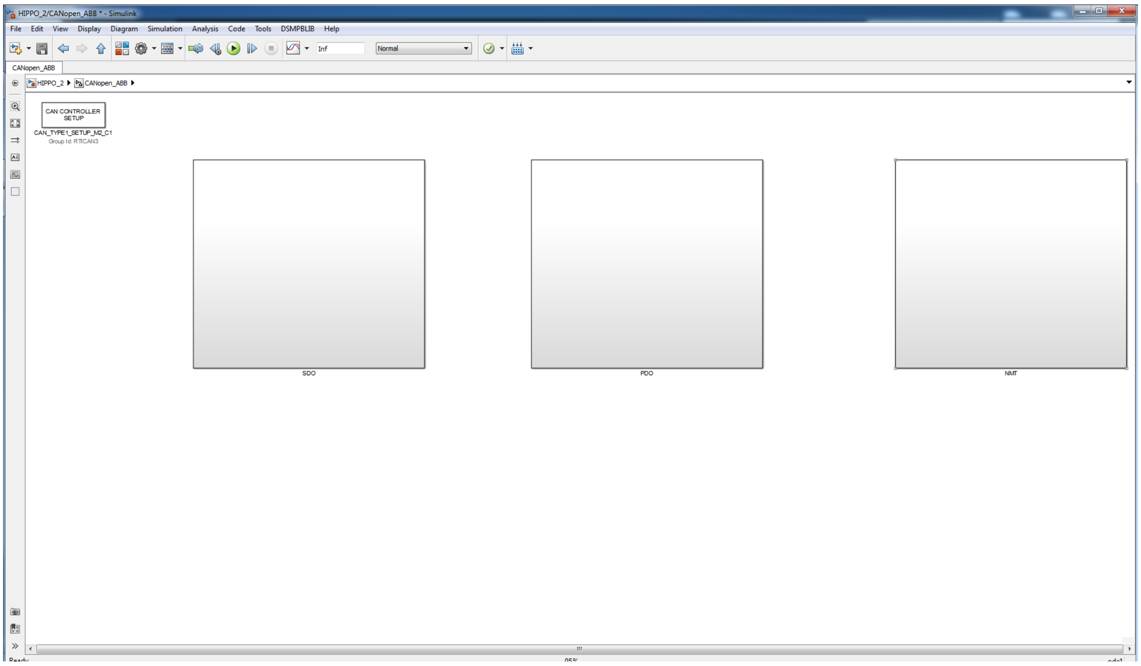
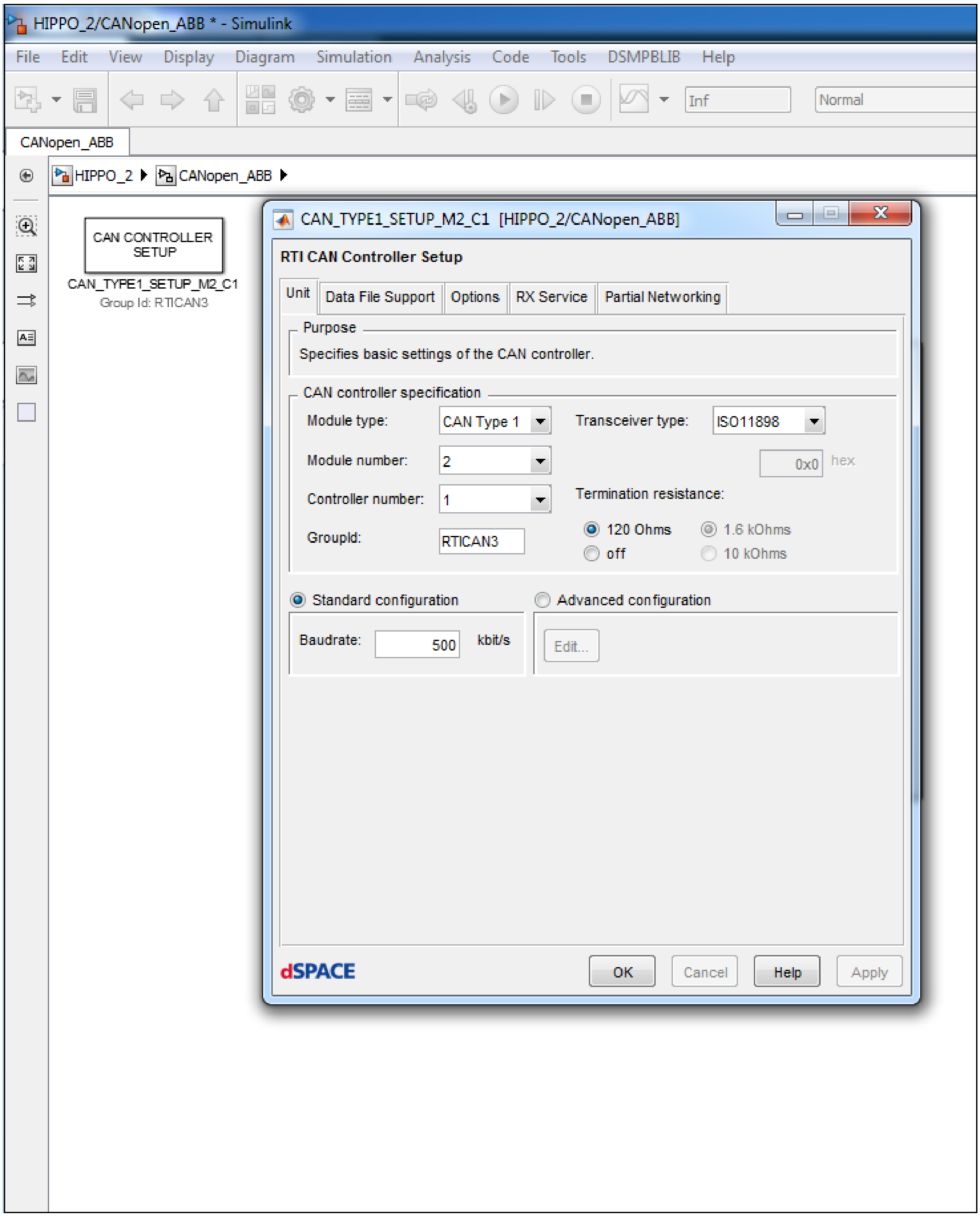


Figure 4.12: CANopen_ABB simulink box.

Similar to RTI Can Controller of J1939, the specified parameters of CAN communication are:



RTI CAN Controller Setup

Purpose

Specifies basic settings of the CAN controller.

CAN controller specification

Module type: CAN Type 1

Module number: 2

Controller number: 1

GroupId: RTICAN3

Transceiver type: ISO11898

0x0 hex

Termination resistance:

● 120 Ohms ○ 1.6 kOhms

○ off ○ 10 kOhms

● Standard configuration ○ Advanced configuration

Baudrate: 500 kbit/s

Figure 4.13: CANopen_ABB controller setup.

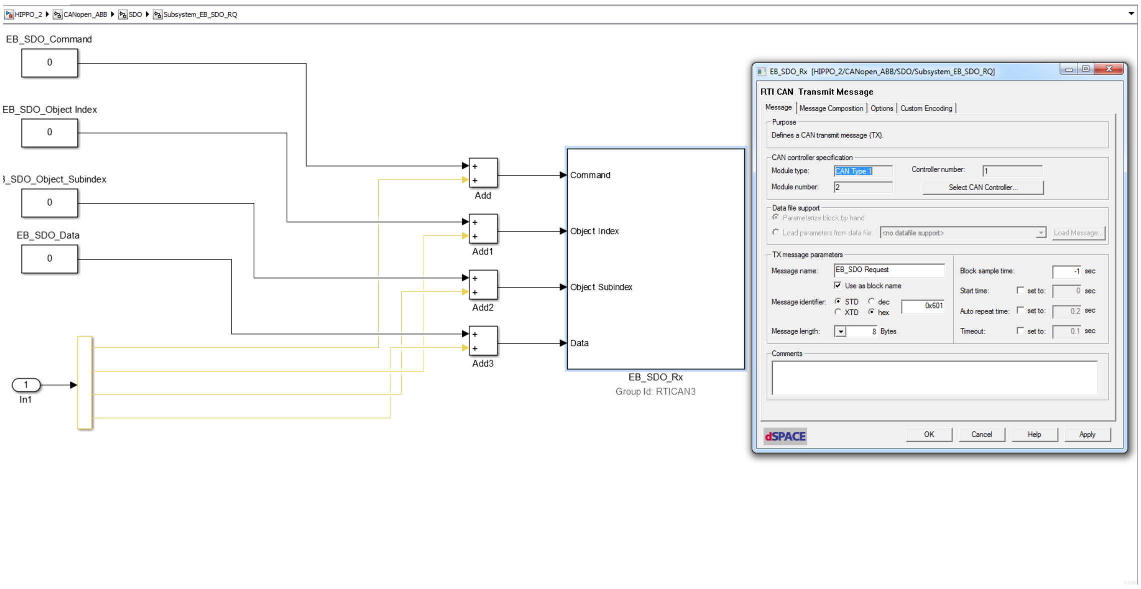
- Module Number set to 2
- Controller Number set to 1
- Groupid set to RTICAN 1
- Baudrate set to 500 kbit/s

SDO configuration

According to paragraph 2.3.3 where the Service Data Object is explained, in order to establish the communication with CANopen, we use SDO to have access to the entries in the device Object Dictionary. This will be made clear with an example.

Example of SDO message We send an SDO message with message identifier 0x601. Number 0x601 analysed in 0x600, where 0x600 is the Communication Object (COB) ID from Master (Control Desk) to Slave (Node). Number 1 is the Node ID, in that case 1 is Electric Brake motor.(fig/ 4.14) Next is need to predefine the message format of the transmited Message.(4.15) Each SDO message has the next format (page 51 from CANopen, Adapter Module RCAN-01 User's Manual):

- Command: Starts from bit 0 and has 8 bits length
- Object Index: Starts from bit 8 and has 16 bits length
- Object Subindex: Starts from bit 24 and has 8 bits length
- Data: Starts from bit 32 and has 32 bits length



EB_SDO_Rx [HIPPO_2/CANopen_ABB/SDO/Subsystem_EB_SDO_RQ]

RTI CAN Transmit Message

Message | Message Composition | Options | Custom Encoding

Purpose
Defines a CAN transmit message (TX).

CAN controller specification
Module type: CAN Type 1
Controller number: 1
Module number: 2
Select CAN Controller...

Data file support
● Parameterize block by hand
○ Load parameters from data file: <no datafile support>
Load Message...

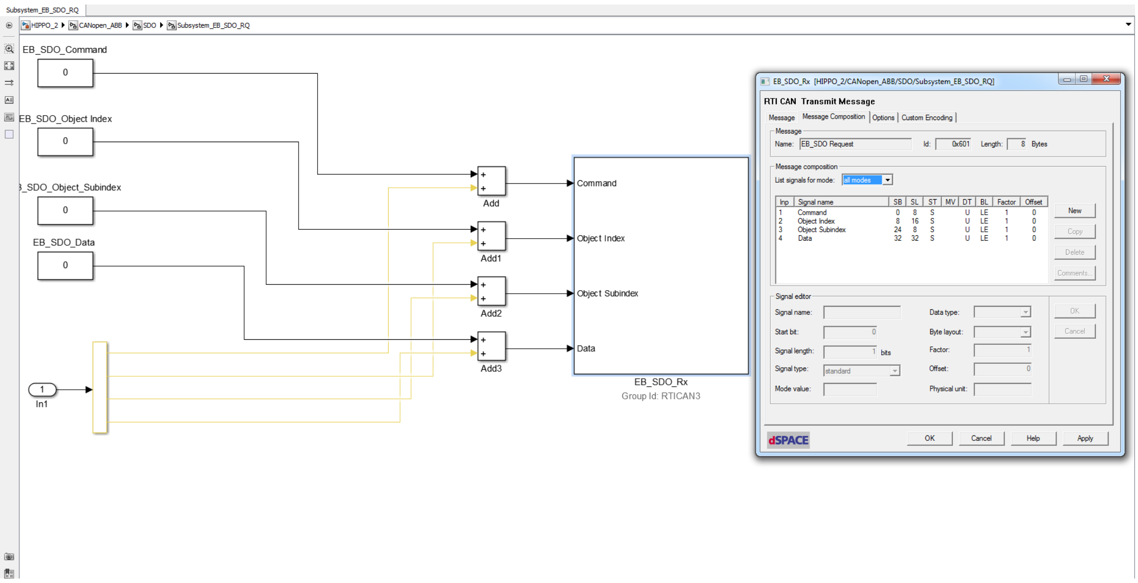
TX message parameters
Message name: EB_SDO Request
☑ Use as block name
Message identifier: ● STD ○ XTD ○ dec ● hex 0x601
Message length: 8 Bytes
Block sample time: -1 sec
Start time: □ set to: 0 sec
Auto repeat time: □ set to: 0.2 sec
Timeout: □ set to: 0.1 sec

Comments

Figure 4.14: EB_SDO request.

In order to send and receive messages with CANopen protocol an initialisation procedure need to be done before every start up. This initialisation is the mapping of PDO through Object dictionary. One message of SDO which is send to configure the PDOs that we receive from the nodes, should have the next attributes:

- Message Identifier: consist of hex code of message sender (Master or Slave) and Node ID of the receiver
- Command: From [9] insert the hex code depending the following message, e.g. Write Dictionary Object reply, expedited, 4 bytes sent (because Data in SDO message consist of 4 bytes), the command is 0x23 in Hex or 35 in decimal form.



EB_SDO_Rx [HIPPO_2/CANopen_ABB/SDO/Subsystem_EB_SDO_RQ]

RTI CAN Transmit Message

Message | Message Composition | Options | Custom Encoding

Message

Name: EB_SDO Request Id: 0x601 Length: 8 Bytes

Message composition

List signals for mode: all modes

Inp	Signal name	SB	SL	ST	MV	DT	BL	Factor	Offset
1	Command	0	8	S		U	LE	1	0
2	Object Index	8	16	S		U	LE	1	0
3	Object Subindex	24	8	S		U	LE	1	0
4	Data	32	32	S		U	LE	1	0

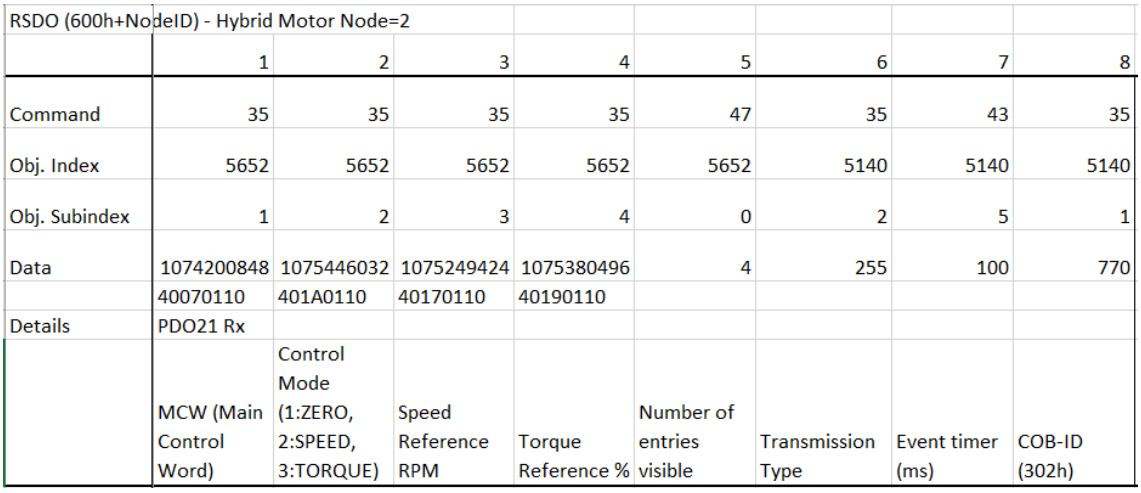
Signal editor

Signal name:
Start bit: 0
Signal length: 1 bits
Signal type: standard
Mode value:
Data type:
Byte layout:
Factor: 1
Offset: 0
Physical unit:

Figure 4.15: EB_SDO message composition.

- Object Index: Is written in CANopen Object Dictionary of [10]. e.g. for mapping PDO21 the Index is 0x1614 or 5652 in decimal
- Object Subindex: Is also written in CANopen Object Dictionary of [10]. e.g. for Mapping Obj1 of PDO21 (p.45 from [10]) the subindex is 1
- Data: In that field is entered the CANopen data which will be sent through PDO 21 Obj1. e.g. The 07.01 Control words is a 16-bit data word, so from p.66 from [10] the Data is 0x40070110 : (4000h+07=4007) and subindex=01 and the length in bits is 2 byte (p.150 from [11]) which translated to 10h (p.86 from [10]). So the final number is 0x40070110 or 1074200848 in decimal form.

In figure 4.16 is an example of the setting parameters of PDO21RX message configuration for Node 2 = Electric Motor throug SDO messages. It has been created similar tables for PDO21Tx, PDO22Tx, PDO23Tx both for Electric Brake (Node 1) and Electric Motor (Node 2)



RSDO (600h+NodeID) - Hybrid Motor Node=2								
	1	2	3	4	5	6	7	8
Command	35	35	35	35	47	35	43	35
Obj. Index	5652	5652	5652	5652	5652	5140	5140	5140
Obj. Subindex	1	2	3	4	0	2	5	1
Data	1074200848	1075446032	1075249424	1075380496	4	255	100	770
	40070110	401A0110	40170110	40190110				
Details	PDO21 Rx							
	MCW (Main Control Word)	Control Mode (1:ZERO, 2:SPEED, 3:TORQUE)	Speed Reference RPM	Torque Reference %	Number of entries visible	Transmission Type	Event timer (ms)	COB-ID (302h)

Figure 4.16: Excel table of PDO21Rx of Node 2 (EM) mapping parameters.

During the initialisation of PDOs need to be sent 32 sdo packets for each ABB engine (motor and brake). To be done, it has been developed in Simulink an SDO matlab function (fig. 4.17). This function send 32 sdo packets for EM and EB motors for communication initialisation in PRE-OPERATIONAL mode. The configuration of 32 packets is auto-programmed by a subsystem which input the numbers 1 to 32 queueing (fig. 4.18)

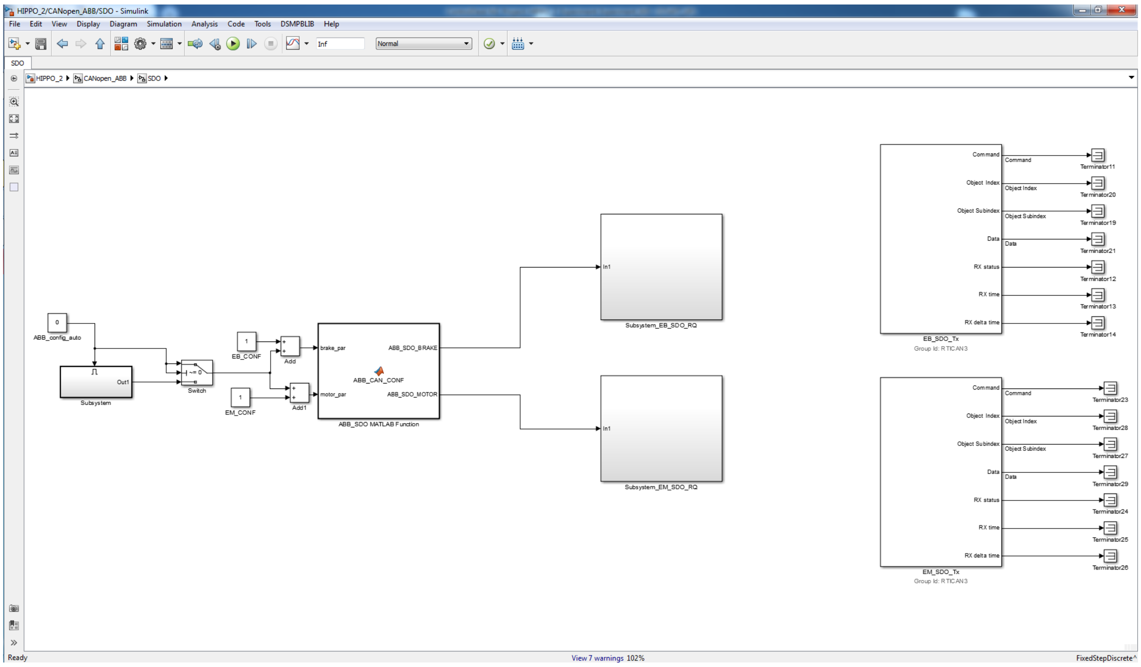


Figure 4.17: SDO simulink.

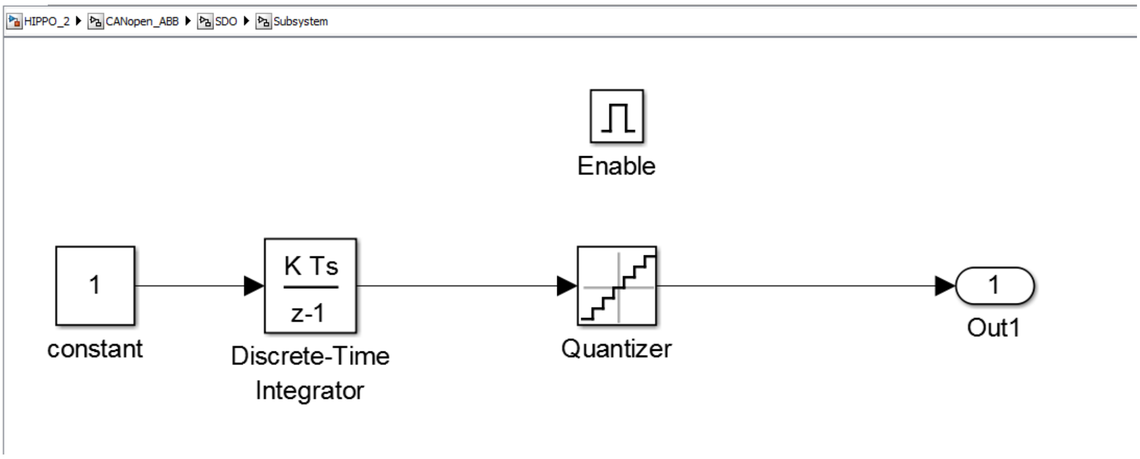
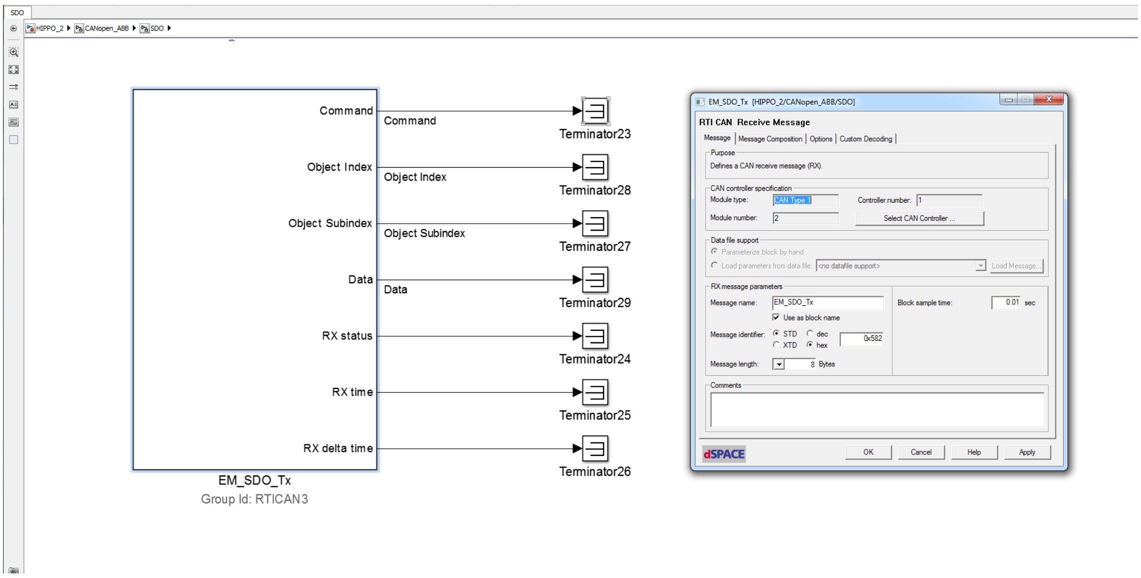


Figure 4.18: Subsystem for selection 32 SDO packets.

SDO receive block In figures 4.19) and 4.20) there is an example of configuration an SDO receive packet. The main difference from SDO transmit packet is the message identifier which is 0x580 + the Node Id instead of 0x600 from transmit packets. Here Receive means the message which sent from Slave to Master. In this case, Master is the dSMA II and slaves are the EM and EB RCAN modules.

The Message format for receive message is the same as transmit message.



EM_SDO_Tx [HIPPO_2/CANopen_ABB/SDO]

RTI CAN Receive Message

Message | Message Composition | Options | Custom Decoding

Purpose

Defines a CAN receive message (RX).

CAN controller specification

Module type: CAN Type 1 Controller number: 1

Module number: 2 Select CAN Controller ...

Data file support

● Parameterize block by hand

○ Load parameters from data file: <no datafile support> Load Message...

RX message parameters

Message name: EM_SDO_Tx Block sample time: 0.01 sec

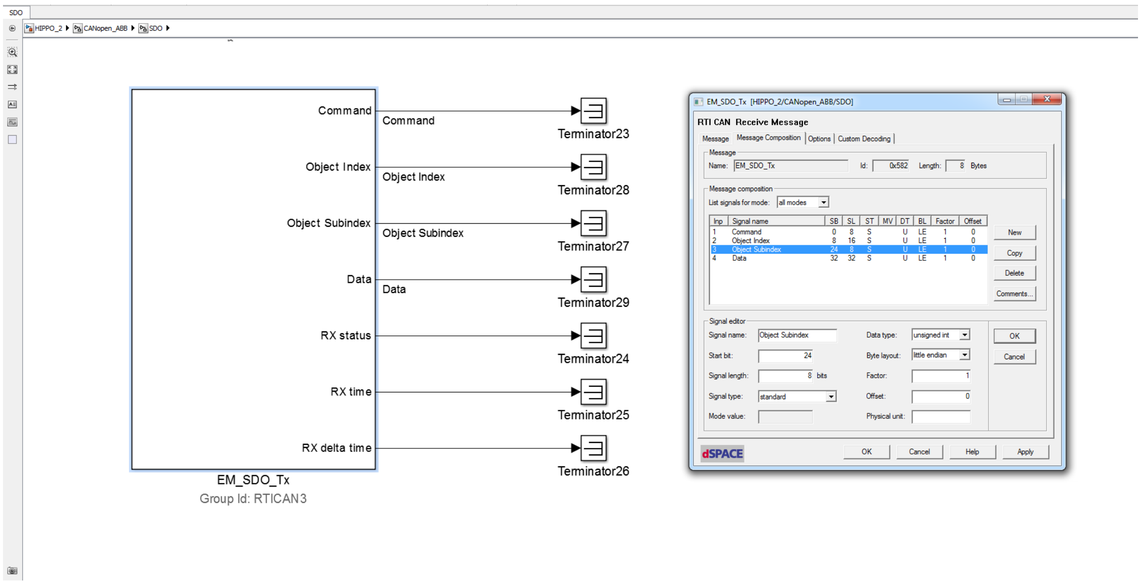
☑ Use as block name

Message identifier: ● STD ○ dec ○ XTD ● hex 0x582

Message length: 8 Bytes

Comments

Figure 4.19: SDO Transmit message.



EM_SDO_Tx [HIPPO_2/CANopen_ABB/SDO]

RTI CAN Receive Message

Message | Message Composition | Options | Custom Decoding

Message

Name: EM_SDO_Tx Id: 0x582 Length: 8 Bytes

Message composition

List signals for mode: all modes

Inp	Signal name	SB	SL	ST	MV	DT	BL	Factor	Offset
1	Command	0	8	S		U	LE	1	0
2	Object Index	8	16	S		U	LE	1	0
3	Object Subindex	24	8	S		U	LE	1	0
4	Data	32	32	S		U	LE	1	0

Signal editor

Signal name: Object Subindex Data type: unsigned int

Start bit: 24 Byte layout: little endian

Signal length: 8 bits Factor: 1

Signal type: standard Offset: 0

Mode value: Physical unit:

Figure 4.20: SDO Transmit.

4.2.3 NMT configuration

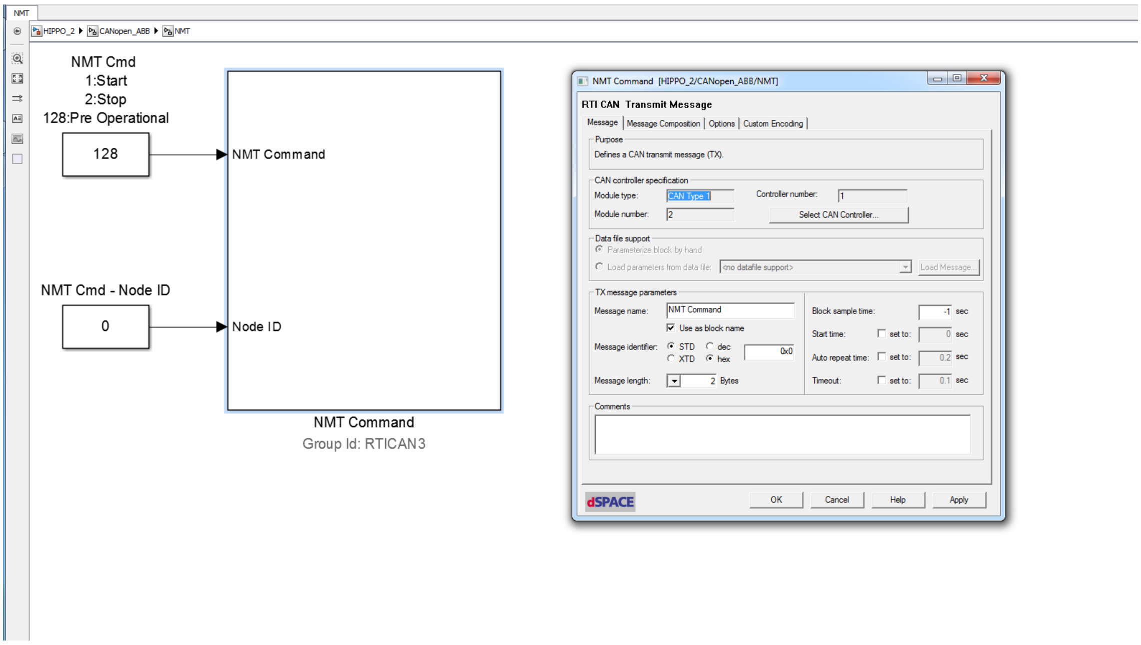
As already referred in section 2.3.3, the NMT Protocol is used by the master node to start/stop and reset the slave nodes of the system.

The NMT packet is always sent with a COB-ID of 0x000; this is the highest priority message that can be sent and it has the following format in figure 4.21

The NMT packet format is shown in figure 4.22

NMT Function can be one of the following:

After power on the node automatically enters initialization, it then moves to pre-operational and emits a Guard 0x700 Message of type boot-up to signal to other nodes that it is present and awake. Depending on the nodes configuration it may then automatically enter Operation state. The Initialisation state is formed from two sub states, reset application and reset communications. Reset application resets all the object dictionary



NMT Command [HIPPO_2/CANopen_ABB/NMT]

RTI CAN Transmit Message

Message | Message Composition | Options | Custom Encoding

Purpose: Defines a CAN transmit message (TX).

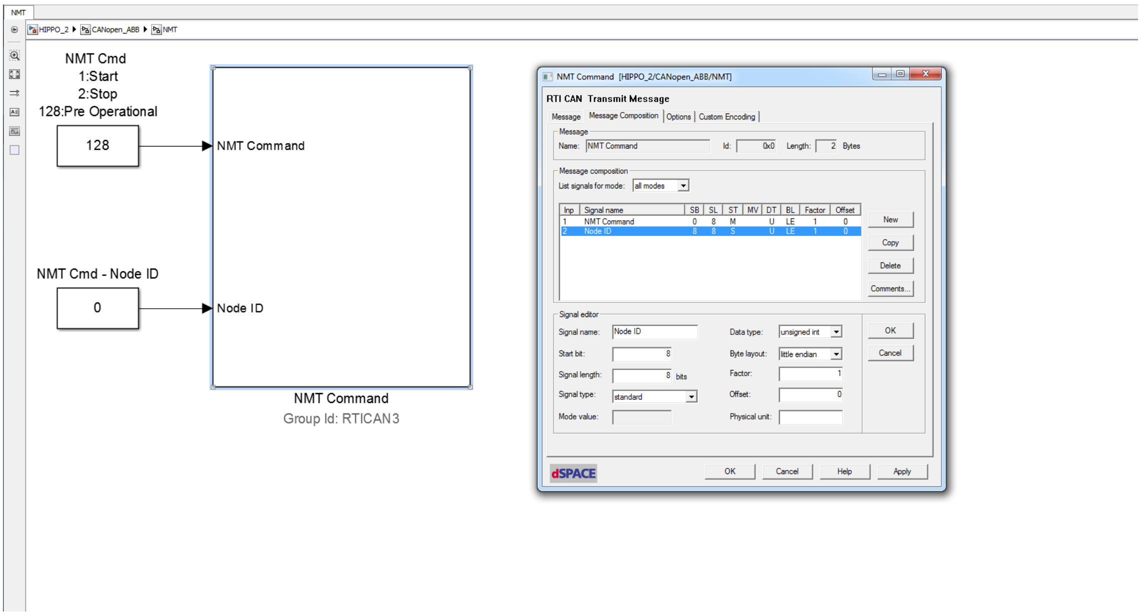
CAN controller specification
Module type: CAN Type 1
Controller number: 1
Module number: 2

Data file support
● Parameterize block by hand
○ Load parameters from data file: <no datafile support>

TX message parameters
Message name: NMT Command
☑ Use as block name
Message identifier: ● STD ○ dec ○ XTD ● hex 0x0
Message length: 2 Bytes
Block sample time: -1 sec
Start time: □ set to: 0 sec
Auto repeat time: □ set to: 0.2 sec
Timeout: □ set to: 0.1 sec

Comments

Figure 4.21: NMT message Identifier.



NMT Command [HIPPO_2/CANopen_ABB/NMT]

RTI CAN Transmit Message

Message | Message Composition | Options | Custom Encoding

Message
Name: NMT Command Id: 0x0 Length: 2 Bytes

Message composition
List signals for mode: all modes

Inp	Signal name	SB	SL	ST	MV	DT	BL	Factor	Offset
1	NMT Command	0	8	M		U	LE	1	0
2	Node ID	8	8	S		U	LE	1	0

Signal editor
Signal name: Node ID
Start bit: 8
Signal length: 8 bits
Signal type: standard
Mode value:
Data type: unsigned int
Byte layout: little endian
Factor: 1
Offset: 0
Physical unit:

Figure 4.22: NMT message composition.

entries from the standard profile and the manufacture specific sections. Reset communication resets the object dictionary entries from the communications profile to their power on values.Either of these reset states may be entered directly at any other time (fig.4.23)

In different states various protocols are available to the node, in operational all protocols are available, in Stopped only a limited subset is available, as shown below.

4.2.4 PDO configuration

Process Data objects (PDO) can be configured to move multiple object dictionary entries from one node to another. The mapping of which dictionary get end and where they end

Code Hex	Code Dec	NMT function
0x01	1	Enter Operation
0x02	2	Enter Stop
0x80	128	Enter Pre-operational
0x81	129	Reset Node
0x82	130	Reset Communication

Table 4.1: Table of NMT function.

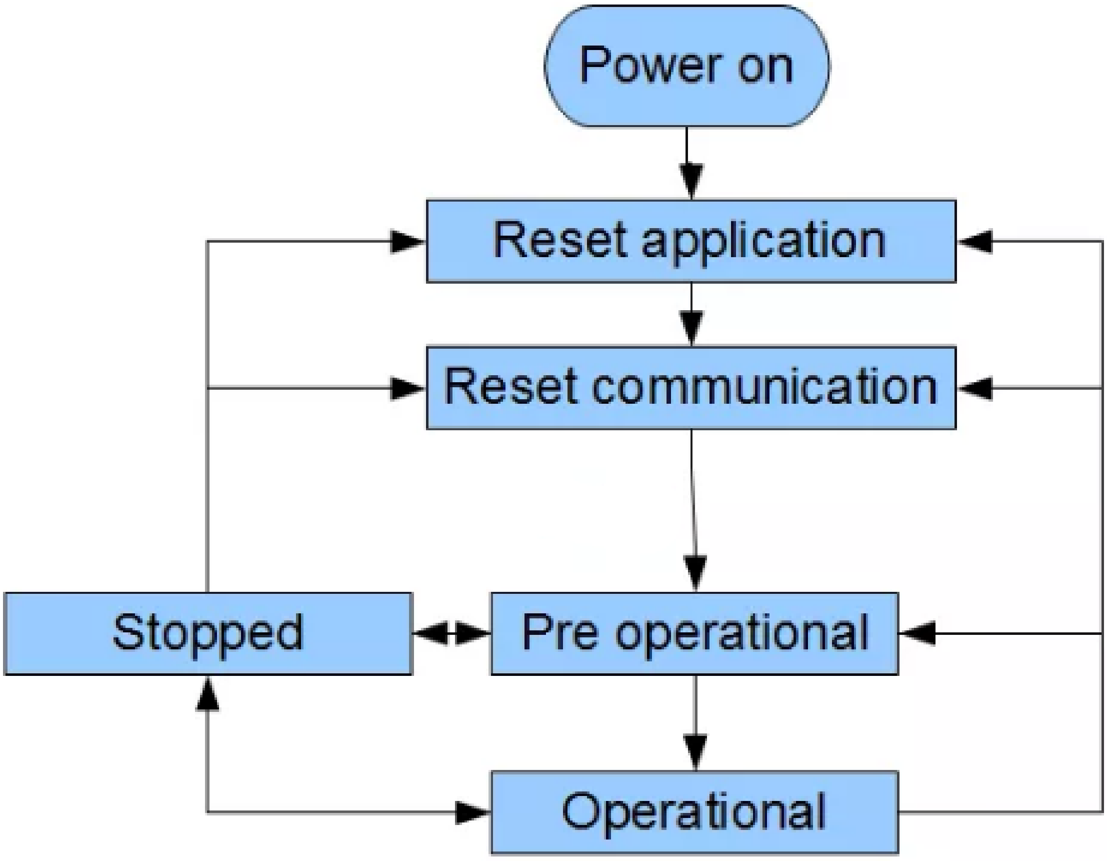


Figure 4.23: NMT message composition.

up is totally programmable, as described in SDO configuration subsection. PDOs are a maximum of 8 bytes of data but in those 8 data bytes you can pick and choose which object dictionary values to send. In HIPPO-2 it has been programmed to send and receive the next signals :

Receiving messages from each engine (EB and EM)

- MSW (Main Status Word) - PDO21
- Actual Control Mode (Zero, Speed or Torque Driven) - PDO21
- Motor Speed (rpm) - PDO21
- Motor Torque (%) - PDO21
- Motor Power (%) - PDO22
- Drive DC Voltage (V) - PDO22

Function	**Stopped**	**Pre-Operational**	**Operational**
NMT	YES	YES	YES
PDO	NO	NO	YES
SDO	NO	YES	YES

Table 4.2: Table of NMT function.

- Motor AC Voltage (V) - PDO22
- Motor Current (A) - PDO22
- Frequency (Hz) - PDO23
- kWh - PDO23
- Used Main Control Word - PDO23

Transmitting messages to each engine (EB and EM)

- MSW (Main Status Word) - PDO21
- Torque Mode selector (1.Speed or 2.Torque selection) - PDO21
- Motor Speed Reference (rpm) - PDO21
- Torque reference (%) - PDO21

Programming of PDO messages The RTI CAN receive message block for PDO's need to configure the next parameter (fig.:4.24) :

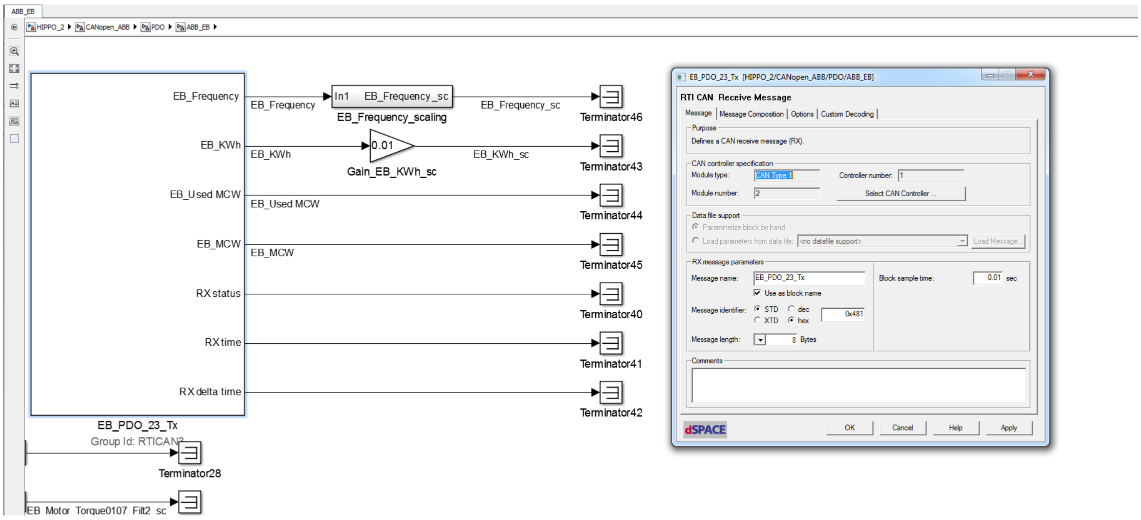
- Message name: the name of RTI CAN Receive message e.g. EB_PDO_23_Tx = Electric Brake PDO23 transmit (by the prepsective of the slave node)
- Message identifier e.g. 0x480+Node ID:1 = 0x481 (**receive** from Node **1**)
- Message length : 8 bytes (PDOs may have 1 byte of data to 8)
- Block Sample time : 0.01 sec (10 ms)

The message composition is configured according to each signal acquisition (fig.:4.25):

- Signal name: the name of Receive signal e.g. EB_KWh = Electric Brake operating KWh
- Start bit, each signal has 2 bytes of data according to p.48 of User's Manual RCAN-01. So the second signal will be in third and fourth byte = the starting bit is 16
- Signal length : 2 bytes, explained in start bit.
- Factor :Is writen in ACS800 Firmware manual, in section Actual signals and parameters in column FbEq for ea signal.
- Offset is 0

The configured receiving and transmitting signals both from and to Electric Brake as well Electric Motor are shown in table A.5

PDO21	Rx	500h+node	ID
PDO21	Tx	480h+node	ID
PDO22	Rx	400h+node	ID
PDO22	Tx	380h+node	ID
PDO23	Rx	300h+node	ID
PDO23	Tx	280h+node	ID



EB_PDO_23_Tx [HIPPO_2/CANopen_ABB/PDO/ABB_EB]

RTI CAN Receive Message

Message | Message Composition | Options | Custom Decoding

Purpose: Defines a CAN receive message (RX).

CAN controller specification
Module type: CAN Type 1
Controller number: 1
Module number: 2
Select CAN Controller ...

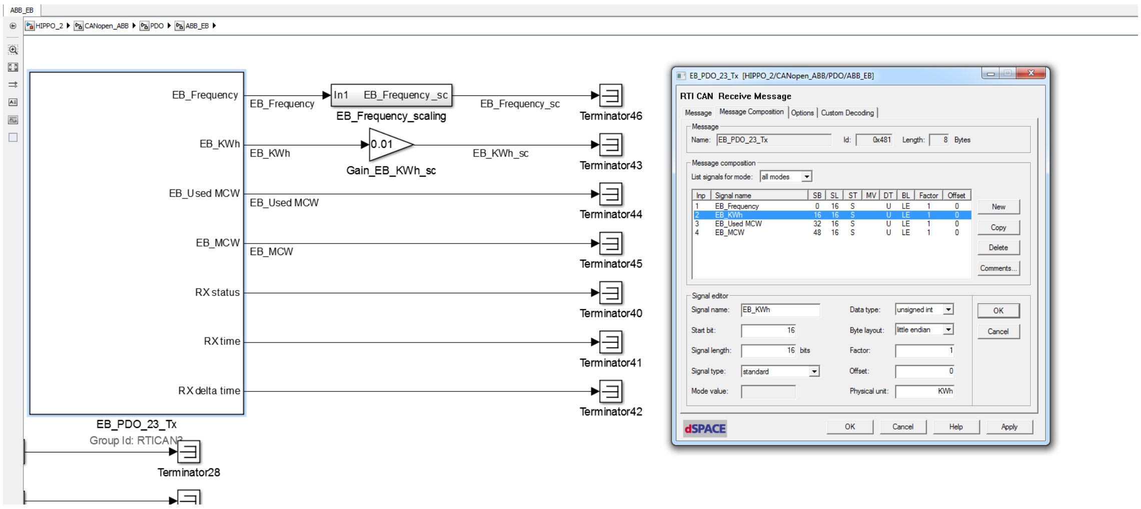
Data file support
● Parameterize block by hand
○ Load parameters from data file: <no datafile support> Load Message...

RX message parameters
Message name: EB_PDO_23_Tx
☑ Use as block name
Message identifier: ● STD ○ XTD ○ dec ● hex 0x481
Message length: 8 Bytes
Block sample time: 0.01 sec

Comments

dSPACE OK Cancel Help Apply

Figure 4.24: PDO23 message transmit block - message parameters.



EB_PDO_23_Tx [HIPPO_2/CANopen_ABB/PDO/ABB_EB]

RTI CAN Receive Message

Message | Message Composition | Options | Custom Decoding

Message
Name: EB_PDO_23_Tx Id: 0x481 Length: 8 Bytes

Message composition
List signals for mode: all modes

Inp	Signal name	SB	SL	ST	MV	DT	BL	Factor	Offset
1	EB_Frequency	0	16	S		U	LE	1	0
2	EB_KWh	16	16	S		U	LE	1	0
3	EB_Used MCW	32	16	S		U	LE	1	0
4	EB_MCW	48	16	S		U	LE	1	0

New Copy Delete Comments...

Signal editor
Signal name: EB_KWh
Start bit: 16
Signal length: 16 bits
Signal type: standard
Mode value:
Data type: unsigned int
Byte layout: little endian
Factor: 1
Offset: 0
Physical unit: KWh

OK Cancel

dSPACE OK Cancel Help Apply

Figure 4.25: PDO23 message transmit block - message composition.

4.3 I/O of dSpace Microautobox II

According to the electric drawing of figure A.12 there are 4 analog channels for I/O to dSMA II and 3 CAN connections with HIPPO2. Through 4 analog channels, it is achieved the control desk's signal acquisition from HBM torque meter and the two ABB Coriolis flow meters. Similar with CAN messages, the analog signal acquisition is configured from Matlab/Simulink. The blocks for analog signals in Simulink can be found from Simulink library browser in dSPACE RTI 1401 then MicroAutoBox II DS1511, ADC Type 4 and the selected block for analog input signals is ADC_TYPE4_BL1. For better visualisation, an ADC area has been created, referred to ADC (analog to digital converter) in figure 4.26 with three sub sections. The first is the Analog Inputs block section; there the four analog signals are configured:

- Analog Chanel 1 : Speed HBM
- Analog Chanel 2 : Torque HBM
- Analog Chanel 3 : Fuel return (Tank)
- Analog Chanel 4 : Fuel feed (Engine)

In each ADC_TYPE4_BL1 Simulink block are configured the module, the channel and the input voltage range. Each block ends to one Goto block (fig:4.27)

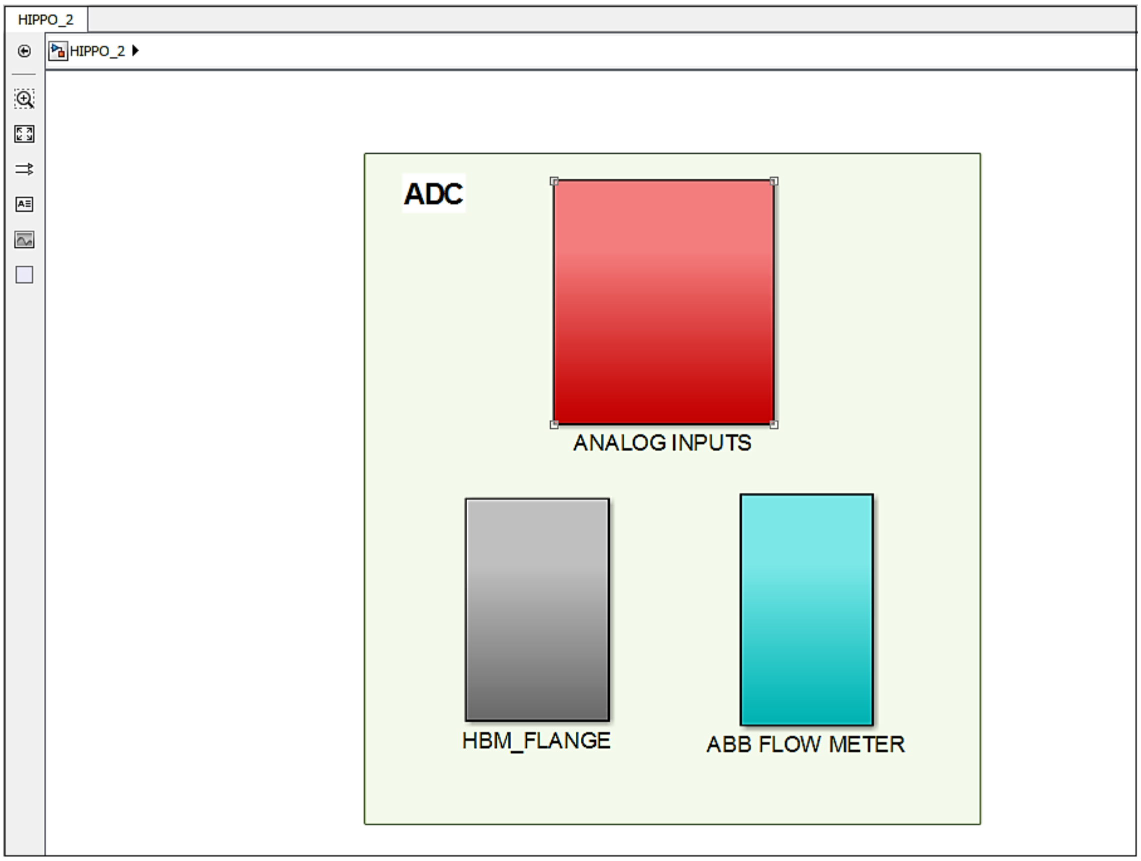
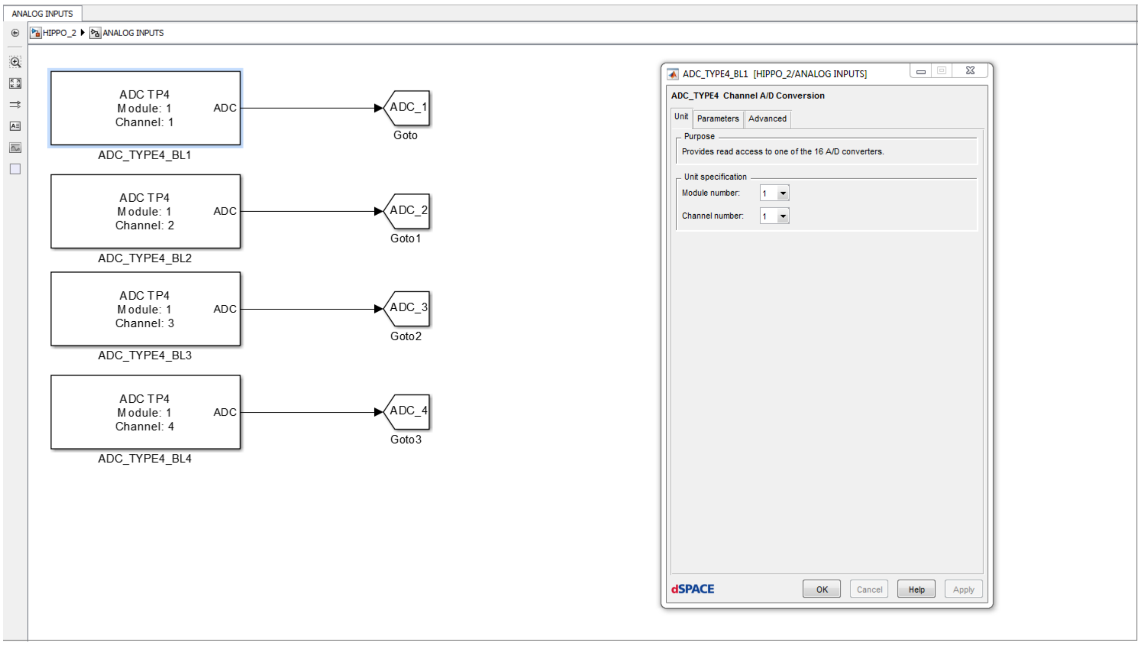


Figure 4.26: Simulink Blocks for Analog signals.

After the acquisition, the analog signals are processed and calibrated with filters gains and constants (figures 4.28 and 4.29)



ADC_TYPE4 Channel A/D Conversion

Purpose

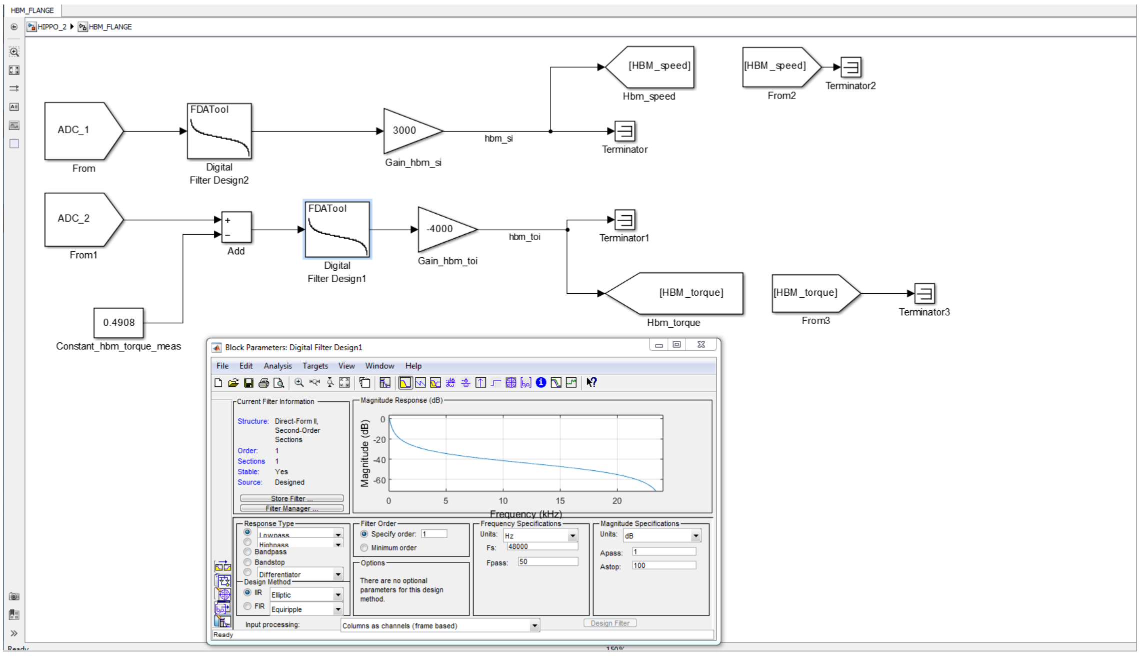
Provides read access to one of the 16 A/D converters.

Unit specification

Module number: 1

Channel number: 1

Figure 4.27: ADC blocks for the four analog signals.



Current Filter Information

Structure: Direct-Form II, Second-Order Sections

Order: 1

Sections 1

Stable: Yes

Source: Designed

Response Type

- ● Lowpass
- ○ Highpass
- ○ Bandpass
- ○ Bandstop
- ○ Differentiator

Design Method

- ● IIR Elliptic
- ○ FIR Equiripple

Filter Order

- ● Specify order: 1
- ○ Minimum order

Options

There are no optional parameters for this design method.

Frequency Specifications

Units: Hz

Fs: 48000

Fpass: 50

Magnitude Specifications

Units: dB

Apass: 1

Astop: 100

Input processing: Columns as channels (frame based)

Figure 4.28: Filtration and calibration of HBM analog signals.

The data for calibration and filtration of analog signals shown in tables A.6, A.7, A.8, A.9

4.4 ControlDesk

4.4.1 Presentation

ControlDesk is the dSPACE experiment software for seamless ECU development. Through an configurable environment the user is able to operate troubleshooting and collect data

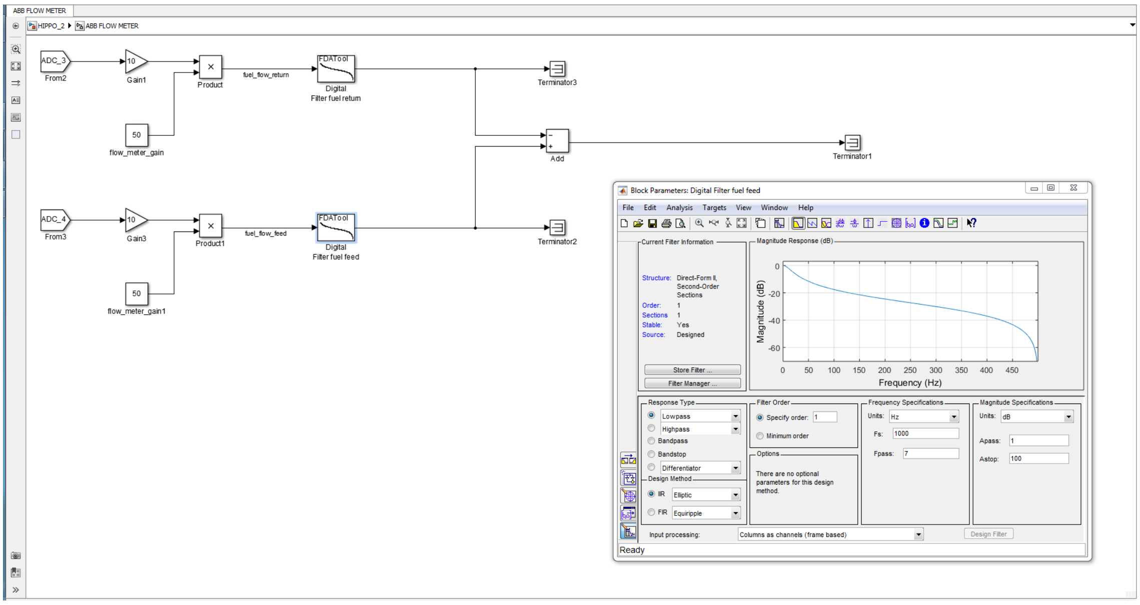


Figure 4.29: Filtration and calibration of ABB Coriolis analogue signals.

from any test bed operating under dSMA II. Some of the possibilities that ControlDesk provide are the development process stages of rapid control prototyping, hardware-in-the-loop simulation, ECU calibration and diagnostics, analysis of vehicle bus systems and more. ControlDesk Next Generation makes it easy to access data and to control simulation platforms, measurement devices, bus interfaces and ECUs.

For HIPPO-2 testbed is developed two ControlDesk's platforms. The first one is created for initialisation of CANopen bus communication (fig. 4.30) and the second for operation and troubleshooting the HIPPO-2 testbed (fig.:4.31).

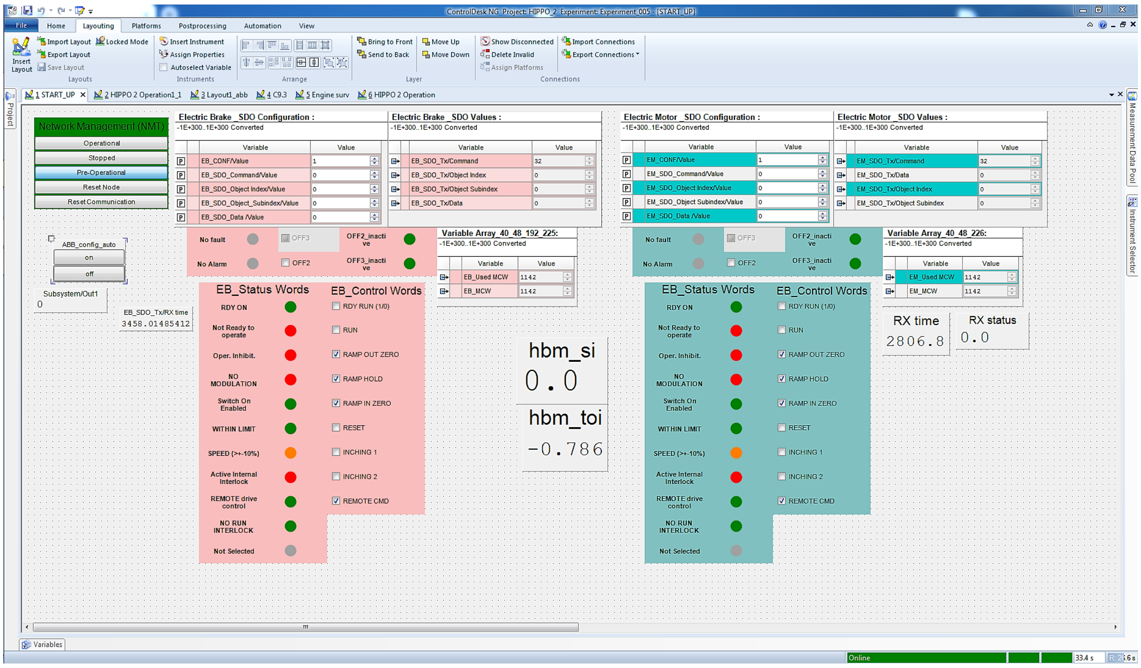
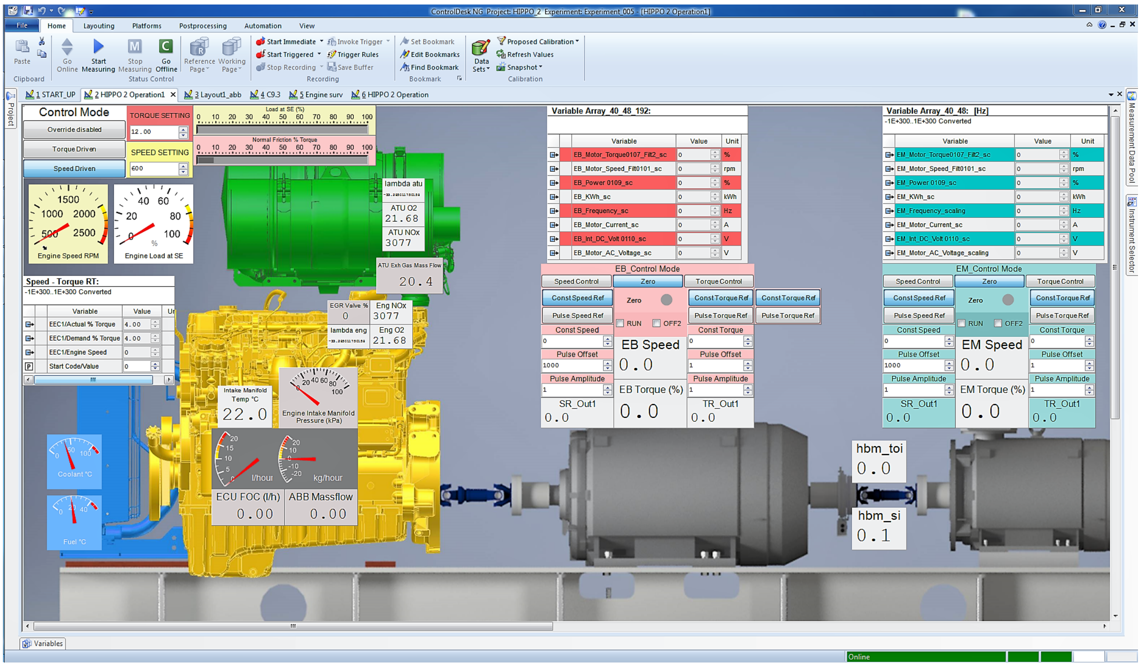


Figure 4.30: Initialisation of CANopen communication.



ControlDesk NG Project: HIPPO_2 Experiment: Experiment_005 - [HIPPO 2 Operation1]

Control Mode

TORQUE SETTING: 12.00

SPEED SETTING: 600

Variable Array_40_48_192:

Variable	Value	Unit
EB_Motor_Torque0107_Filt2_sc	0	%
EB_Motor_Speed_Filt0101_sc	0	rpm
EB_Power 0109_sc	0	%
EB_KWh_sc	0	kWh
EB_Frequency_sc	0	Hz
EB_Motor_Current_sc	0	A
EB_Int_DC_Volt 0110_sc	0	V
EB_Motor_AC_Voltage_sc	0	V

Variable Array_40_48: [Hz]
-1E+300..1E+300 Converted

Variable	Value	Unit
EM_Motor_Torque0107_Filt2_sc	0	%
EM_Motor_Speed_Filt0101_sc	0	rpm
EM_Power 0109_sc	0	%
EM_KWh_sc	0	kWh
EM_Frequency_scaling	0	Hz
EM_Motor_Current_sc	0	A
EM_Int_DC_Volt 0110_sc	0	V
EM_Motor_AC_Voltage_scaling	0	V

Speed - Torque RT:
-1E+300..1E+300 Converted

Variable	Value	Ur
EEC1/Actual % Torque	4.00	
EEC1/Demand % Torque	4.00	
EEC1/Engine Speed	0	
Start Code/Value	0	

lambda atu: -33.28501175015

ATU O2: 21.68

ATU NOx: 3077

ATU Exh Gas Mass Flow: 20.4

EGR Valve %: 0

Eng NOx: 3077

lambda eng: -33.28501175015

Eng O2: 21.68

Intake Manifold Temp °C: 22.0

ECU FOC (l/h): 0.00

ABB Massflow: 0.00

EB_Control Mode

Const Speed: 0

Pulse Offset: 1000

Pulse Amplitude: 1

SR_Out1: 0.0

EB Speed: 0.0

EB Torque (%): 0.0

Const Torque: 0

Pulse Offset: 1

Pulse Amplitude: 1

TR_Out1: 0.0

EM_Control Mode

Const Speed: 0

Pulse Offset: 1000

Pulse Amplitude: 1

SR_Out1: 0.0

EM Speed: 0.0

EM Torque (%): 0.0

Const Torque: 0

Pulse Offset: 1

Pulse Amplitude: 1

TR_Out1: 0.0

hbm_toi: 0.0

hbm_si: 0.1

Online

Figure 4.31: HIPPO-2 control screen.

4.4.2 CANopen Initialisation sequence

CANopen protocol require an initialisation procedure to enable the communication with PDO messages between the nodes according to theoretical approach of chapter 2. The communication initialisation is enabled through next steps:

- Activation of Pre-operational mode from NMT block
- Configure the EB and EM PDO messages. This achieved by ABB_auto_config block in START_UP area in figure 4.30 which configure each SDO packet as explained in paragraph SDO configuration.

Now the CANopen communication is sett up and the two ABB motors are capable of receiving and transmitting the PDO messages.

4.4.3 Electric motor operation

Electric motor's start, run and stop modes are controlled by Main Control Words (MCW). The Control Word is the principal means of controlling the drive from an external control system. The Control Word is sent by the external control system to the drive. The drive switches between its states according to the bit-coded instructions of the Control Word. Each bit of MCW message is explained in ACS800 firmware manual (page 150).

For the *ready to run mode* the MCW's bits should have the values from table 4.3

Bit	Bit Value
0	1
1	1
2	1
3	0
4	1
5	1
6	1
7	0
8	0
9	0
10	1

Table 4.3: MCW bit values.

The operation profile of ABB Drives is shown in figure 4.32

In addition, one control example is presented in figure 4.33. The control example is: START by AUTO- or DC MAGN mode, STOP by ramp generator.

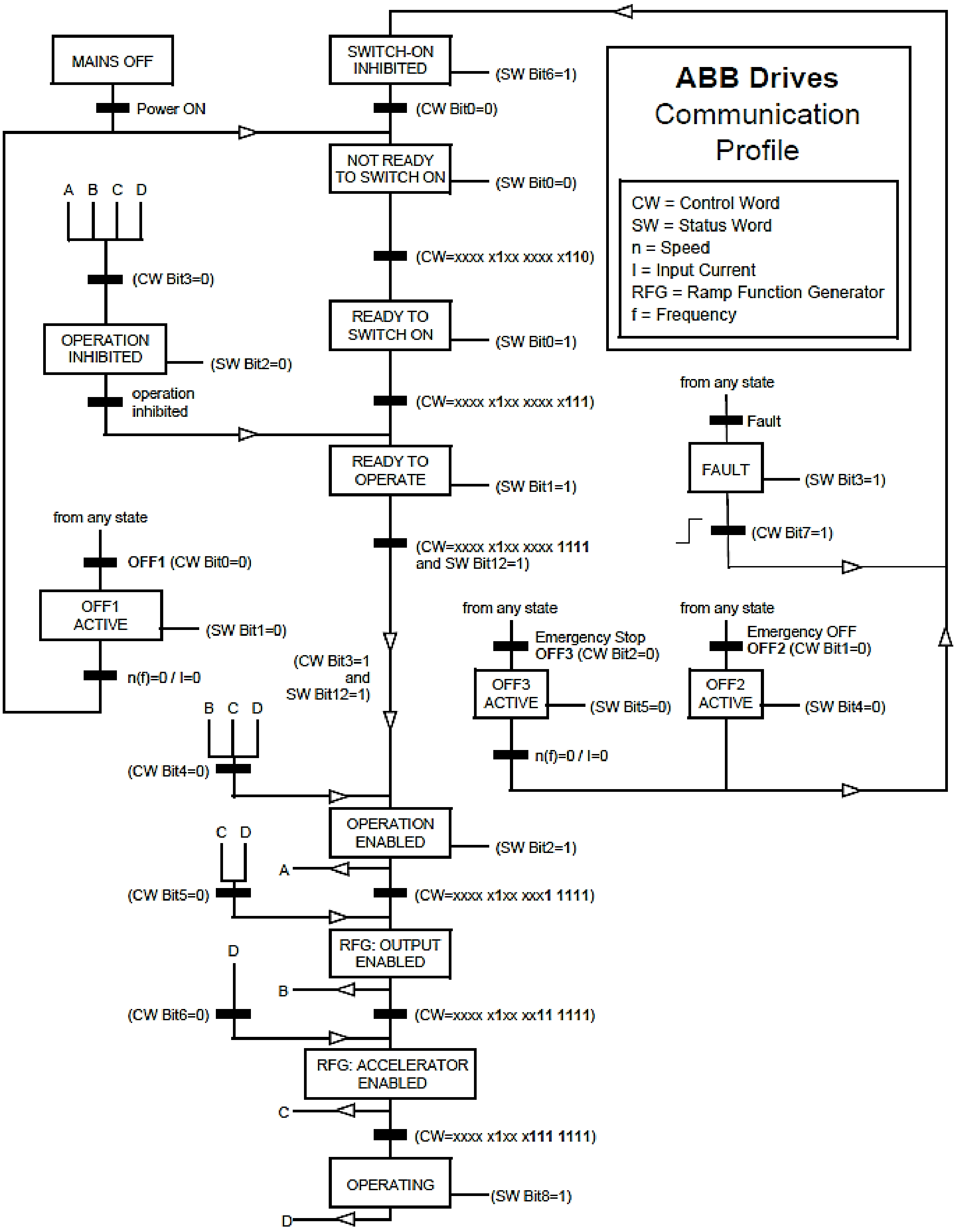


Figure 4.32: ABB Drives Profile for AC-drives (p.312 from ACS800 firmware manual).

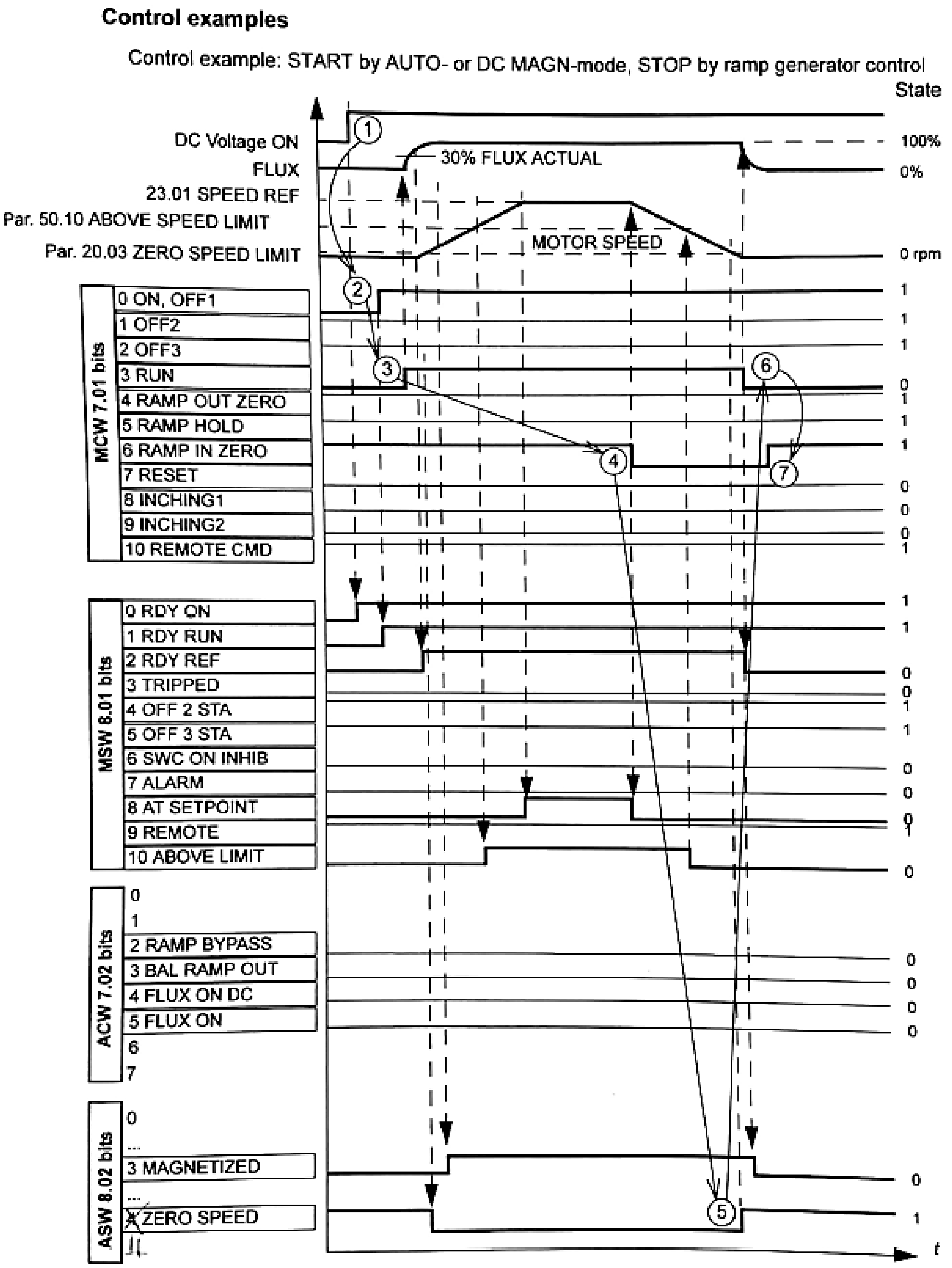


Figure 4.33: ABB motor control example (from ACS800 firmware manual, p. 313).

Chapter 5

Operation of HIPPO-2 and Experimental Results

Following the completion of the communication between the different devices and technology platforms, a final test is performed so as ensure the proper operation of HIPPO-2. For that purpose, it a simulation scenario for HIPPO-2 has been developed and its results are documented bellow.

5.1 Scenario of HIPPO-2

In October and November 2018 the following scenario took place in HIPPO-2 test bed, in order to verify the communication between the various diverse parts and subsystems of HIPPO-2:

- The diesel engine operates both in steady speed and torque during the test.
- Electric Brake applied reverse torque to the diesel engine.
- The Electric Motor applied reverse torque as well.
- Except from the essential signals of speed and torque measurement from EB and CAT diesel engine, some other signals were also recorded (HBM Torque meter, Coriolis flow meters, EM engine etc).

Following are the figures from the experiments.

In figure 5.1 the speed and torque from three motors as well as inlet/output pressures and fuel consumption from diesel engine are shown. Following, in figure 5.2 lambda, NOx emissions and EGR valve opening percentage are shown. In figure 5.3 the exhaust gas mass flow, temperature and coolant temperature of CAT engine are presented. The comparison between the two torques, the one from Electric Motor (EM) and HBM torque meter is shown in figure 5.4. At last but not least in figure 5.5 it is possible to compare current consumption with the voltage of Electric Brake (EB) during the test.

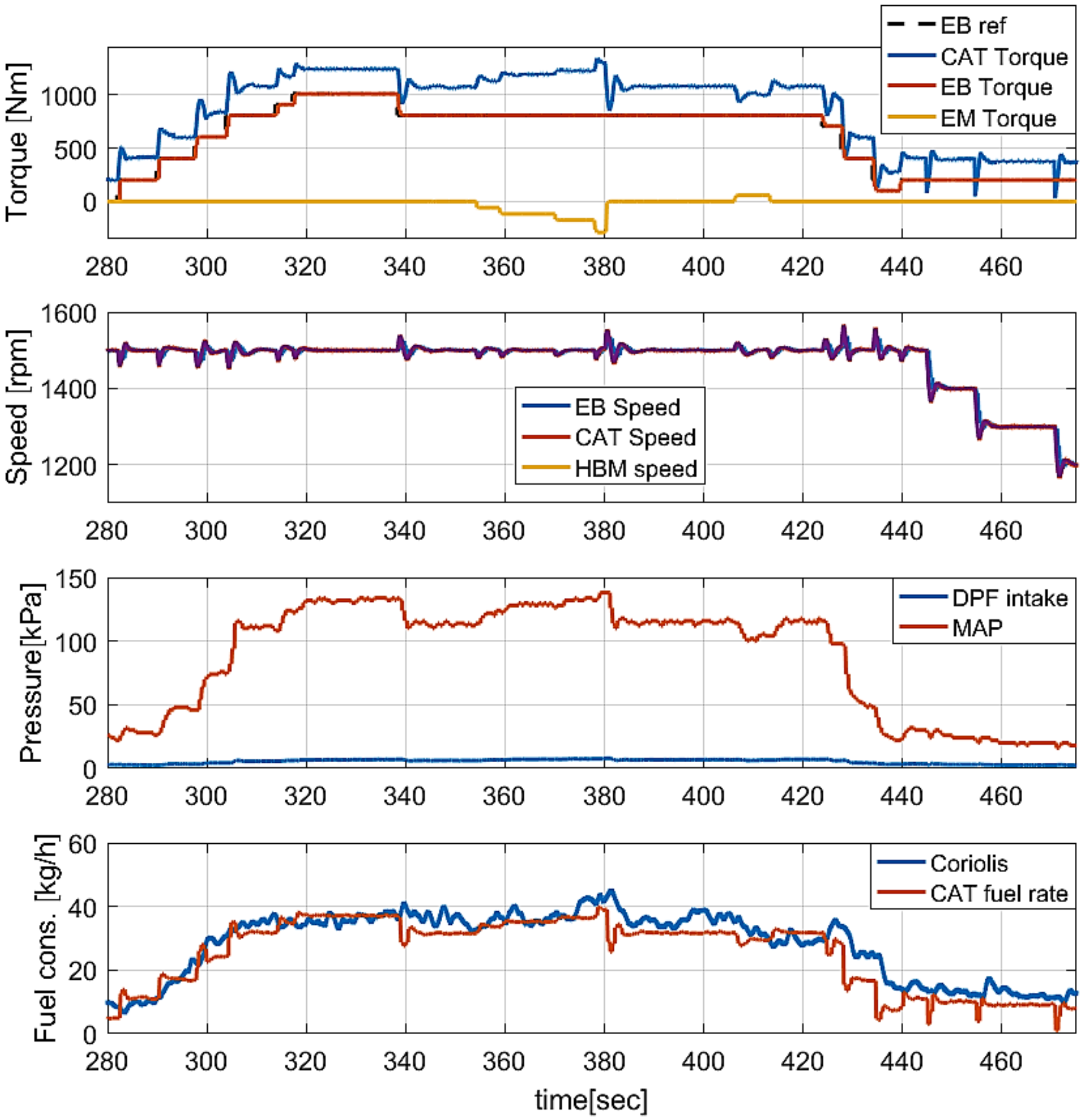


Figure 5.1: Torque, speed, pressure and fuel consumption.

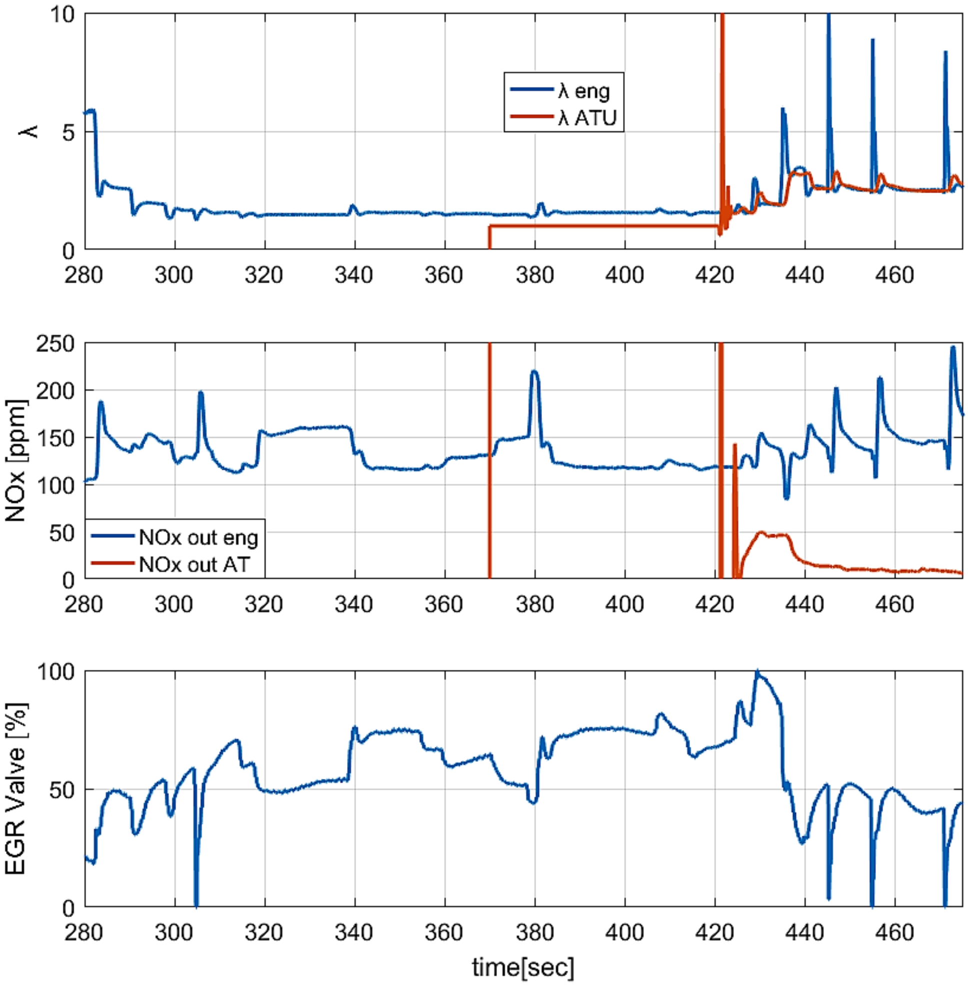


Figure 5.2: ,NOx and EGR valve opening(%).

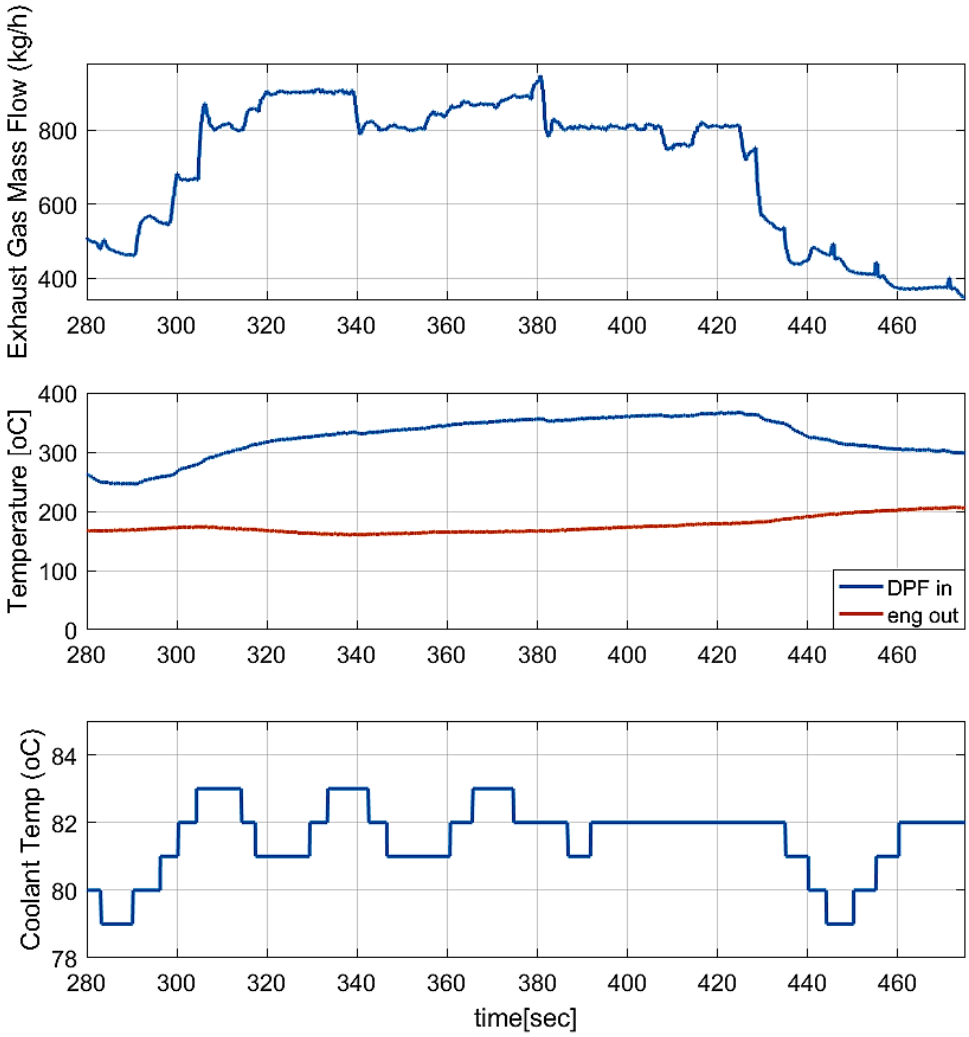


Figure 5.3: Exhaust gas mass flow, gas temperature and coolant temperature of diesel engine.

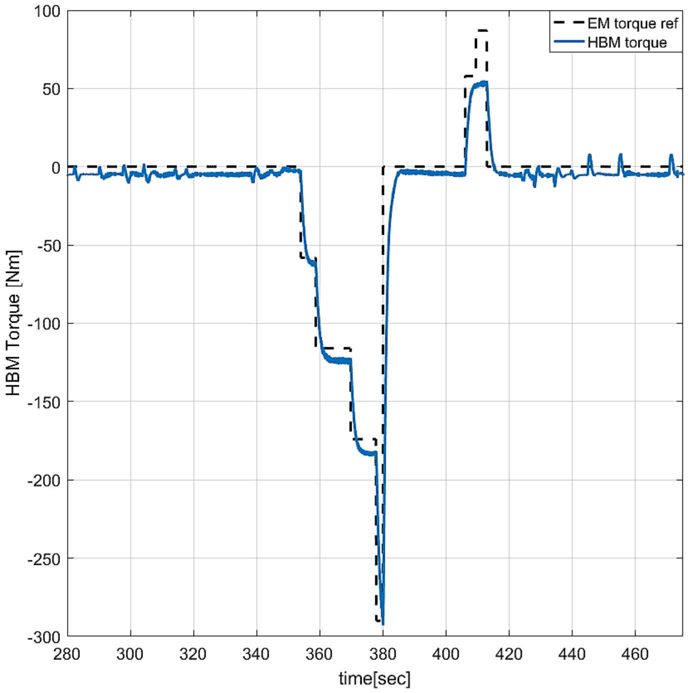


Figure 5.4: Torque reference from EM as compared to Torque from HBM sensor.

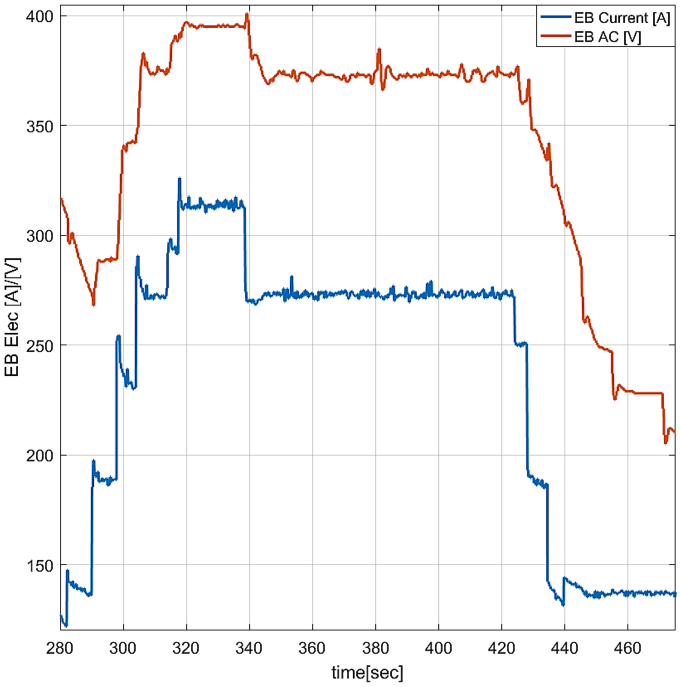


Figure 5.5: Voltage and Current from EB.

Chapter 6

Conclusion and Future Work

This thesis provided two major contributions including: (1) communication establishment of analog signals and two different high-level CAN protocols; (2) registration of every parameter and procedure which took place during the data acquisition system set up, with the main focus being on developing prototype engine controllers and instrumentation.

During the installation of cabling, several difficulties were encountered from the right equipment selection to thoroughly parametrisation of signal communication. Especially attention was needed in understanding the CAN protocol and parametrising all the necessary blocks in Simulink. Even some unexpected mechanical problems became evident during the initial test of the test bed but were successfully dealt with by the collaboration and contribution of Laboratory staff.

Future Work

HIPPO-2 is a unique test-bed which provides the opportunity for many experimental procedures to take place in future. At this time, the following ideas are provided for consideration.

Low cost - High efficiency control During the last decade significant progress in control with low cost processors has taken place. It would be a very interesting approach to try controlling such high output engines with CAN protocol, using a low cost and open source controllers like Rasberry and Arduino.

Deeper signal analysis HIPPO-2 consists of dozens sensors and actuators mainly located in CAT C9.3 diesel engine. A more analytical approach to each sensor could lead to more experimental and research work, like diesel engine modelling and more broadly to marine hybrid system propulsion modelling. Thus a future work could have been towards the understanding of sensors and recording their technical as well CAN characteristics.

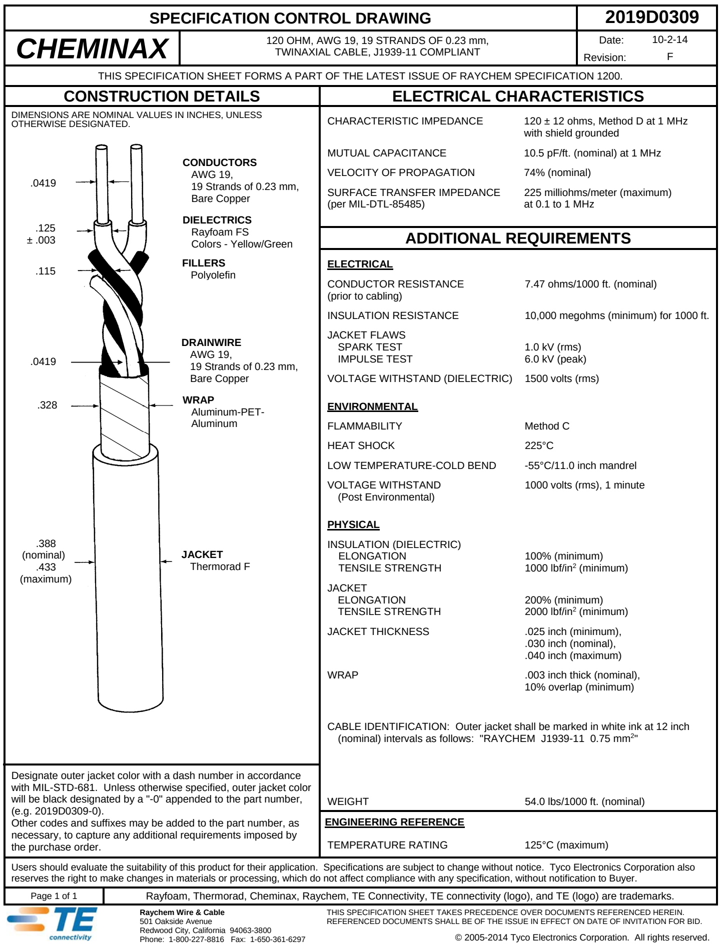
Emission reduction studies A more accurate exhaust gas temperature model can be developed by adding gas flow dynamics. EGR and waste gate are controlled by C9.3 controller. A potential area of improvement is to make a communication between EGR controller and dSpace MicroAutobox II in order to study furthermore NOx and particles reduction.

Appendices

Appendix A

HIPPO-2 data

A.1 Electric Drawings



SPECIFICATION CONTROL DRAWING		2019D0309
CHEMINAX	120 OHM, AWG 19, 19 STRANDS OF 0.23 mm, TWINAXIAL CABLE, J1939-11 COMPLIANT	Date: 10-2-14 Revision: F

THIS SPECIFICATION SHEET FORMS A PART OF THE LATEST ISSUE OF RAYCHEM SPECIFICATION 1200.

CONSTRUCTION DETAILS

DIMENSIONS ARE NOMINAL VALUES IN INCHES, UNLESS OTHERWISE DESIGNATED.

.0419
.125
± .003
.115
.0419
.328
.388
(nominal)
.433
(maximum)

CONDUCTORS
AWG 19,
19 Strands of 0.23 mm,
Bare Copper

DIELECTRICS
Rayfoam FS
Colors - Yellow/Green

FILLERS
Polyolefin

DRAINWIRE
AWG 19,
19 Strands of 0.23 mm,
Bare Copper

WRAP
Aluminum-PET-
Aluminum

JACKET
Thermorad F

Designate outer jacket color with a dash number in accordance with MIL-STD-681. Unless otherwise specified, outer jacket color will be black designated by a "-0" appended to the part number, (e.g. 2019D0309-0).
Other codes and suffixes may be added to the part number, as necessary, to capture any additional requirements imposed by the purchase order.

ELECTRICAL CHARACTERISTICS

CHARACTERISTIC IMPEDANCE	120 ± 12 ohms, Method D at 1 MHz with shield grounded
MUTUAL CAPACITANCE	10.5 pF/ft. (nominal) at 1 MHz
VELOCITY OF PROPAGATION	74% (nominal)
SURFACE TRANSFER IMPEDANCE (per MIL-DTL-85485)	225 milliohms/meter (maximum) at 0.1 to 1 MHz

ADDITIONAL REQUIREMENTS

ELECTRICAL

CONDUCTOR RESISTANCE (prior to cabling)	7.47 ohms/1000 ft. (nominal)
INSULATION RESISTANCE	10,000 megohms (minimum) for 1000 ft.
JACKET FLAWS	
SPARK TEST	1.0 kV (rms)
IMPULSE TEST	6.0 kV (peak)
VOLTAGE WITHSTAND (DIELECTRIC)	1500 volts (rms)

ENVIRONMENTAL

FLAMMABILITY	Method C
HEAT SHOCK	225°C
LOW TEMPERATURE-COLD BEND	-55°C/11.0 inch mandrel
VOLTAGE WITHSTAND (Post Environmental)	1000 volts (rms), 1 minute

PHYSICAL

INSULATION (DIELECTRIC)	
ELONGATION	100% (minimum)
TENSILE STRENGTH	1000 lbf/in^2 (minimum)
JACKET	
ELONGATION	200% (minimum)
TENSILE STRENGTH	2000 lbf/in^2 (minimum)
JACKET THICKNESS	.025 inch (minimum), .030 inch (nominal), .040 inch (maximum)
WRAP	.003 inch thick (nominal), 10% overlap (minimum)

CABLE IDENTIFICATION: Outer jacket shall be marked in white ink at 12 inch (nominal) intervals as follows: "RAYCHEM J1939-11 0.75 mm^2"

WEIGHT	54.0 lbs/1000 ft. (nominal)

ENGINEERING REFERENCE

TEMPERATURE RATING	125°C (maximum)

Users should evaluate the suitability of this product for their application. Specifications are subject to change without notice. Tyco Electronics Corporation also reserves the right to make changes in materials or processing, which do not affect compliance with any specification, without notification to Buyer.

Page 1 of 1

Rayfoam, Thermorad, Cheminax, Raychem, TE Connectivity, TE connectivity (logo), and TE (logo) are trademarks.

Raychem Wire & Cable
501 Oakside Avenue
Redwood City, California 94063-3800
Phone: 1-800-227-8816 Fax: 1-650-361-6297

THIS SPECIFICATION SHEET TAKES PRECEDENCE OVER DOCUMENTS REFERENCED HEREIN. REFERENCED DOCUMENTS SHALL BE OF THE ISSUE IN EFFECT ON DATE OF INVITATION FOR BID.

© 2005-2014 Tyco Electronics Corporation. All rights reserved.

Figure A.1: CAN cable

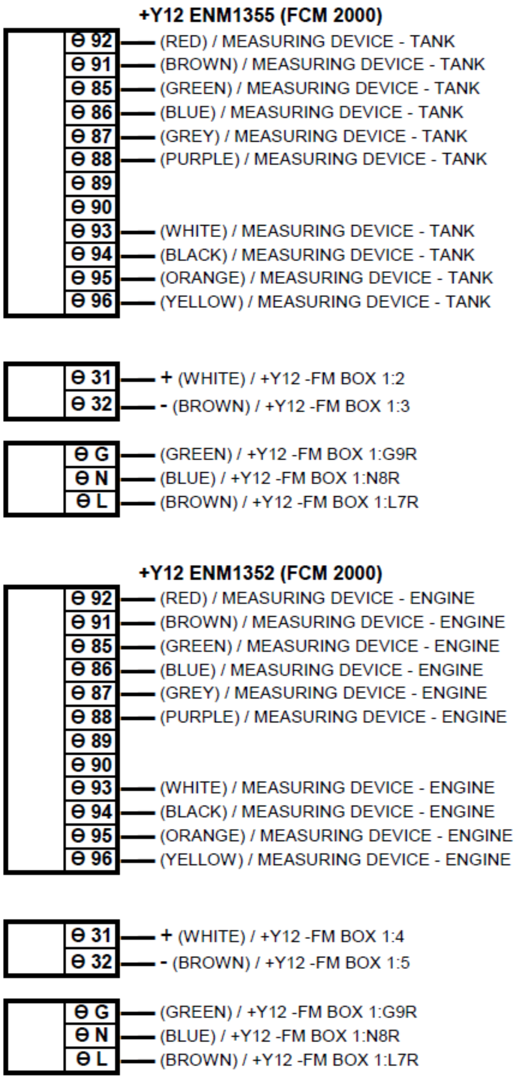

+Y12 ENM1355 (FCM 2000)
ϴ 92
(RED) / MEASURING DEVICE - TANK
ϴ 91
(BROWN) / MEASURING DEVICE - TANK
ϴ 85
(GREEN) / MEASURING DEVICE - TANK
ϴ 86
(BLUE) / MEASURING DEVICE - TANK
ϴ 87
(GREY) / MEASURING DEVICE - TANK
ϴ 88
(PURPLE) / MEASURING DEVICE - TANK
ϴ 89
ϴ 90
ϴ 93
(WHITE) / MEASURING DEVICE - TANK
ϴ 94
(BLACK) / MEASURING DEVICE - TANK
ϴ 95
(ORANGE) / MEASURING DEVICE - TANK
ϴ 96
(YELLOW) / MEASURING DEVICE - TANK
ϴ 31
+ (WHITE) / +Y12 -FM BOX 1:2
ϴ 32
- (BROWN) / +Y12 -FM BOX 1:3
ϴ G
(GREEN) / +Y12 -FM BOX 1:G9R
ϴ N
(BLUE) / +Y12 -FM BOX 1:N8R
ϴ L
(BROWN) / +Y12 -FM BOX 1:L7R
+Y12 ENM1352 (FCM 2000)
ϴ 92
(RED) / MEASURING DEVICE - ENGINE
ϴ 91
(BROWN) / MEASURING DEVICE - ENGINE
ϴ 85
(GREEN) / MEASURING DEVICE - ENGINE
ϴ 86
(BLUE) / MEASURING DEVICE - ENGINE
ϴ 87
(GREY) / MEASURING DEVICE - ENGINE
ϴ 88
(PURPLE) / MEASURING DEVICE - ENGINE
ϴ 89
ϴ 90
ϴ 93
(WHITE) / MEASURING DEVICE - ENGINE
ϴ 94
(BLACK) / MEASURING DEVICE - ENGINE
ϴ 95
(ORANGE) / MEASURING DEVICE - ENGINE
ϴ 96
(YELLOW) / MEASURING DEVICE - ENGINE
ϴ 31
+ (WHITE) / +Y12 -FM BOX 1:4
ϴ 32
- (BROWN) / +Y12 -FM BOX 1:5
ϴ G
(GREEN) / +Y12 -FM BOX 1:G9R
ϴ N
(BLUE) / +Y12 -FM BOX 1:N8R
ϴ L
(BROWN) / +Y12 -FM BOX 1:L7R

Figure A.2: FCM2000 Panel

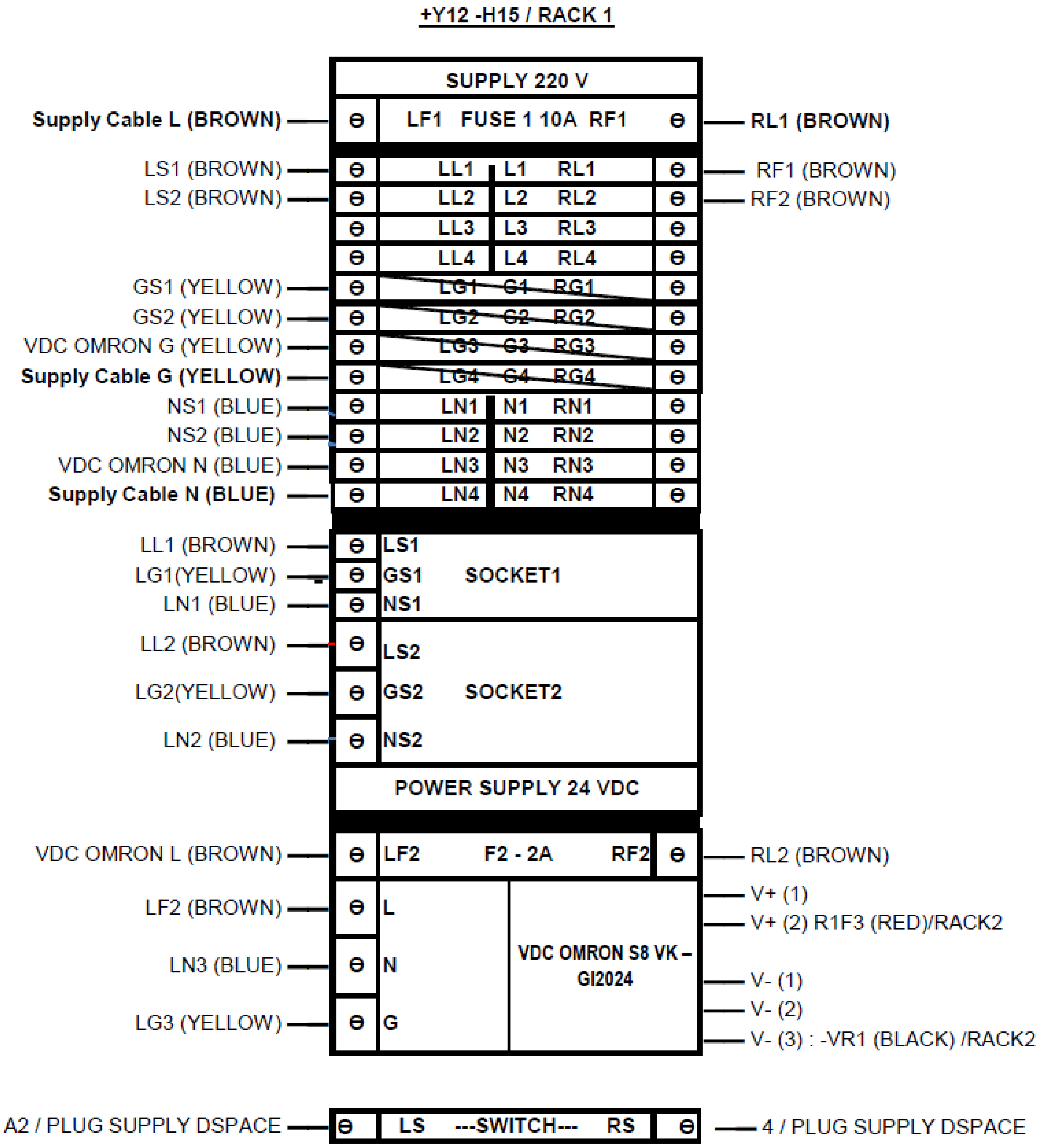


Figure A.3: Electirc Drawing of Rack 1 inside +Y12-H15 Electric Panel

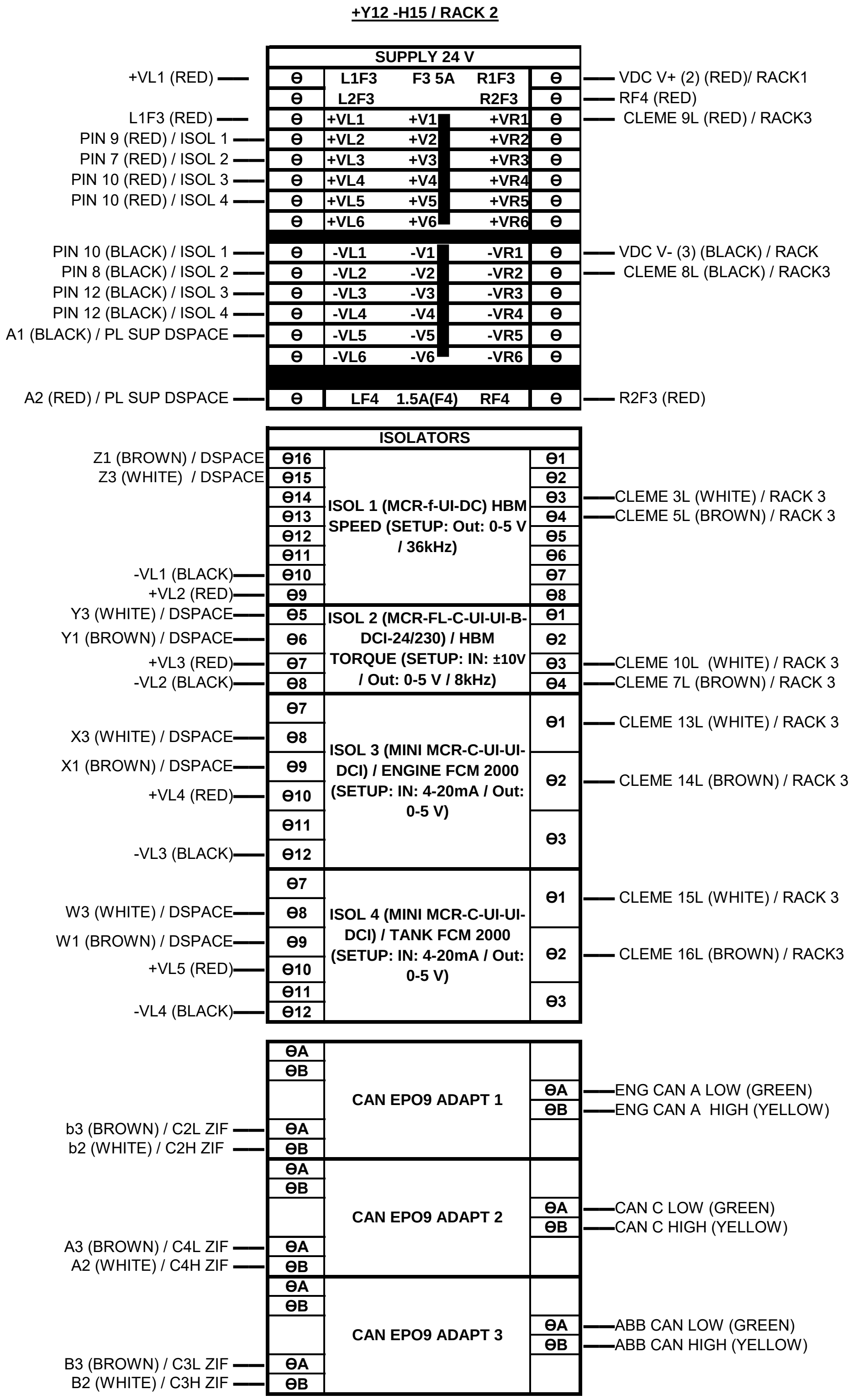


Figure A.4: Electirc Drawing of Rack 2 inside +Y12-H15 Electric Panel

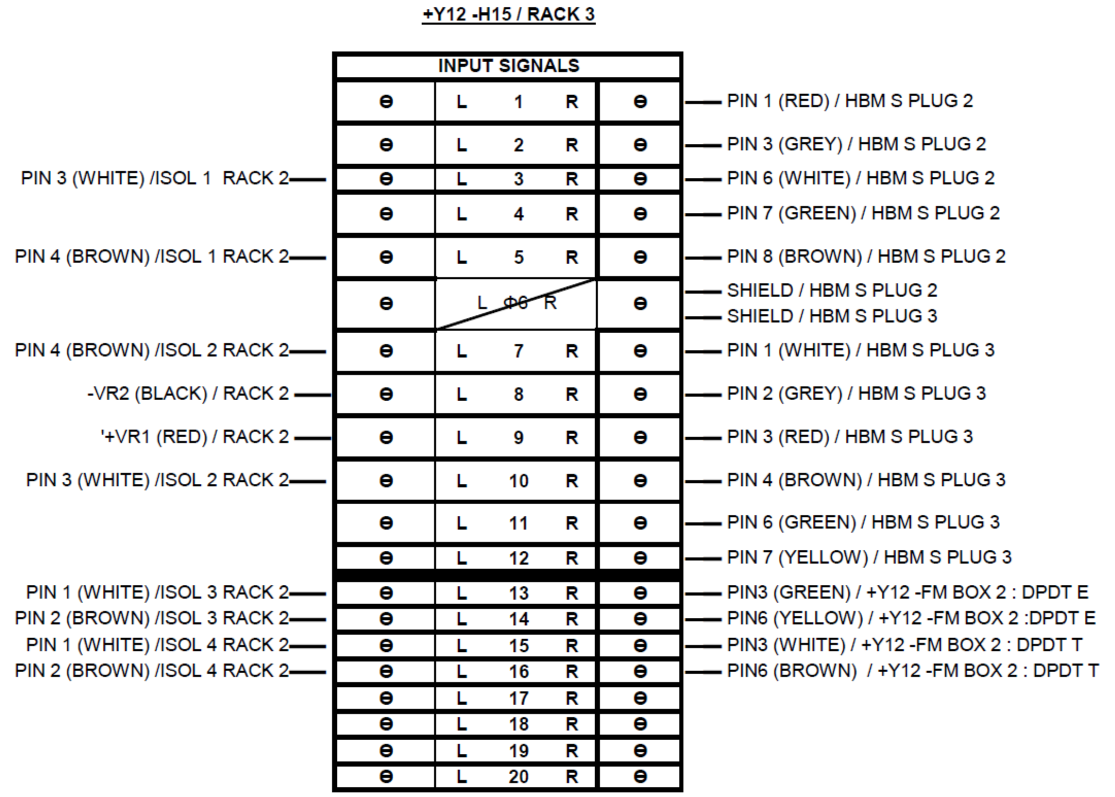

+Y12 -H15 / RACK 3
INPUT SIGNALS
L 1 R
L 2 R
L 3 R
L 4 R
L 5 R
L Φ6 R
L 7 R
L 8 R
L 9 R
L 10 R
L 11 R
L 12 R
L 13 R
L 14 R
L 15 R
L 16 R
L 17 R
L 18 R
L 19 R
L 20 R
PIN 3 (WHITE) /ISOL 1 RACK 2
PIN 4 (BROWN) /ISOL 1 RACK 2
PIN 4 (BROWN) /ISOL 2 RACK 2
-VR2 (BLACK) / RACK 2
'+VR1 (RED) / RACK 2
PIN 3 (WHITE) /ISOL 2 RACK 2
PIN 1 (WHITE) /ISOL 3 RACK 2
PIN 2 (BROWN) /ISOL 3 RACK 2
PIN 1 (WHITE) /ISOL 4 RACK 2
PIN 2 (BROWN) /ISOL 4 RACK 2
PIN 1 (RED) / HBM S PLUG 2
PIN 3 (GREY) / HBM S PLUG 2
PIN 6 (WHITE) / HBM S PLUG 2
PIN 7 (GREEN) / HBM S PLUG 2
PIN 8 (BROWN) / HBM S PLUG 2
SHIELD / HBM S PLUG 2
SHIELD / HBM S PLUG 3
PIN 1 (WHITE) / HBM S PLUG 3
PIN 2 (GREY) / HBM S PLUG 3
PIN 3 (RED) / HBM S PLUG 3
PIN 4 (BROWN) / HBM S PLUG 3
PIN 6 (GREEN) / HBM S PLUG 3
PIN 7 (YELLOW) / HBM S PLUG 3
PIN3 (GREEN) / +Y12 -FM BOX 2 : DPDT E
PIN6 (YELLOW) / +Y12 -FM BOX 2 :DPDT E
PIN3 (WHITE) / +Y12 -FM BOX 2 : DPDT T
PIN6 (BROWN) / +Y12 -FM BOX 2 : DPDT T

Figure A.5: Electirc Drawing of Rack 3 inside +Y12-H15 Electric Panel

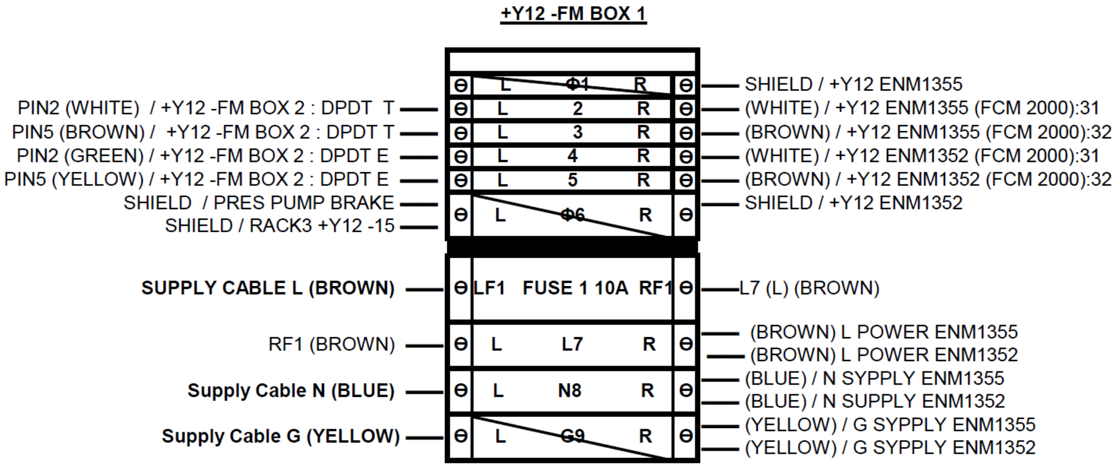

+Y12 -FM BOX 1
L Φ1 R
L 2 R
L 3 R
L 4 R
L 5 R
L Φ6 R
LF1 FUSE 1 10A RF1
L L7 R
L N8 R
L G9 R
PIN2 (WHITE) / +Y12 -FM BOX 2 : DPDT T
PIN5 (BROWN) / +Y12 -FM BOX 2 : DPDT T
PIN2 (GREEN) / +Y12 -FM BOX 2 : DPDT E
PIN5 (YELLOW) / +Y12 -FM BOX 2 : DPDT E
SHIELD / PRES PUMP BRAKE
SHIELD / RACK3 +Y12 -15
SUPPLY CABLE L (BROWN)
RF1 (BROWN)
Supply Cable N (BLUE)
Supply Cable G (YELLOW)
SHIELD / +Y12 ENM1355
(WHITE) / +Y12 ENM1355 (FCM 2000):31
(BROWN) / +Y12 ENM1355 (FCM 2000):32
(WHITE) / +Y12 ENM1352 (FCM 2000):31
(BROWN) / +Y12 ENM1352 (FCM 2000):32
SHIELD / +Y12 ENM1352
L7 (L) (BROWN)
(BROWN) L POWER ENM1355
(BROWN) L POWER ENM1352
(BLUE) / N SYPPLY ENM1355
(BLUE) / N SUPPLY ENM1352
(YELLOW) / G SYPPLY ENM1355
(YELLOW) / G SYPPLY ENM1352

Figure A.6: Electric Drawing of Fuel Meter Box 1

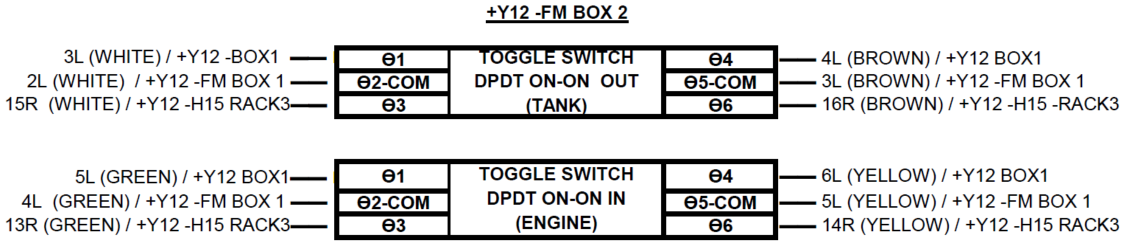

+Y12 -FM BOX 2
ϴ1
ϴ2-COM
ϴ3
TOGGLE SWITCH
DPDT ON-ON OUT
(TANK)
ϴ4
ϴ5-COM
ϴ6
3L (WHITE) / +Y12 -BOX1
2L (WHITE) / +Y12 -FM BOX 1
15R (WHITE) / +Y12 -H15 RACK3
4L (BROWN) / +Y12 BOX1
3L (BROWN) / +Y12 -FM BOX 1
16R (BROWN) / +Y12 -H15 -RACK3
ϴ1
ϴ2-COM
ϴ3
TOGGLE SWITCH
DPDT ON-ON IN
(ENGINE)
ϴ4
ϴ5-COM
ϴ6
5L (GREEN) / +Y12 BOX1
4L (GREEN) / +Y12 -FM BOX 1
13R (GREEN) / +Y12 -H15 RACK3
6L (YELLOW) / +Y12 BOX1
5L (YELLOW) / +Y12 -FM BOX 1
14R (YELLOW) / +Y12 -H15 RACK3

Figure A.7: Electric Drawing of Fuel Meter Box 2

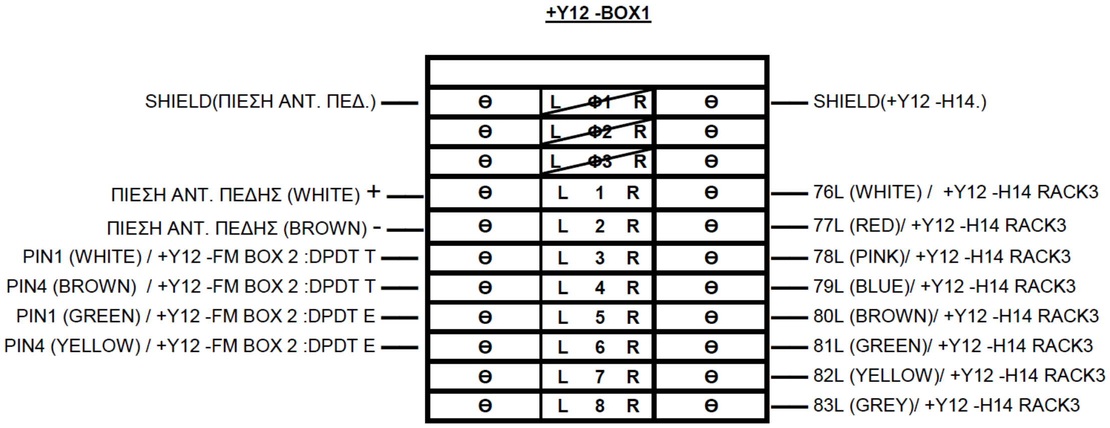
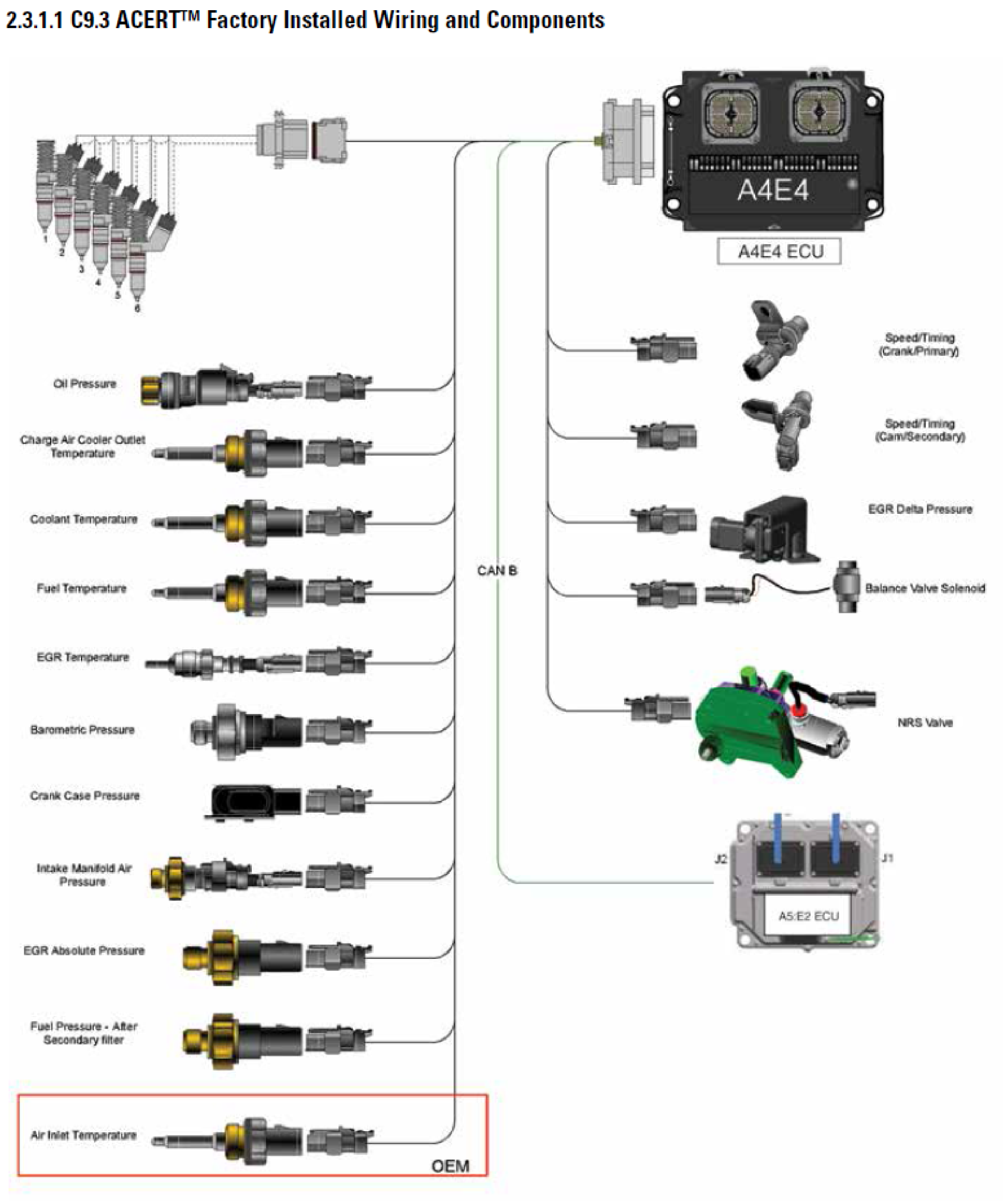

+Y12 -BOX1
SHIELD(ΠΙΕΣΗ ΑΝΤ. ΠΕΔ.)
ΠΙΕΣΗ ΑΝΤ. ΠΕΔΗΣ (WHITE) +
ΠΙΕΣΗ ΑΝΤ. ΠΕΔΗΣ (BROWN) -
PIN1 (WHITE) / +Y12 -FM BOX 2 :DPDT T
PIN4 (BROWN) / +Y12 -FM BOX 2 :DPDT T
PIN1 (GREEN) / +Y12 -FM BOX 2 :DPDT E
PIN4 (YELLOW) / +Y12 -FM BOX 2 :DPDT E
L Φ1 R
L Φ2 R
L Φ3 R
L 1 R
L 2 R
L 3 R
L 4 R
L 5 R
L 6 R
L 7 R
L 8 R
SHIELD(+Y12 -H14.)
76L (WHITE) / +Y12 -H14 RACK3
77L (RED)/ +Y12 -H14 RACK3
78L (PINK)/ +Y12 -H14 RACK3
79L (BLUE)/ +Y12 -H14 RACK3
80L (BROWN)/ +Y12 -H14 RACK3
81L (GREEN)/ +Y12 -H14 RACK3
82L (YELLOW)/ +Y12 -H14 RACK3
83L (GREY)/ +Y12 -H14 RACK3

Figure A.8: Electric Drawing of +Y12-BOX1

A.2 CAT C9.3 data



2.3.1.1 C9.3 ACERT™ Factory Installed Wiring and Components

Figure A.9: The CAT C9.3 diesel engined installed sensors(1).

PGN	Paraneter Group Label	Identifier	SPN
61443	Electronic Engine Controller 2 (EEC2)	0C F0 03 00	91
61443	Electronic Engine Controller 2 (EEC2)	0C F0 03 00	92
61443	Electronic Engine Controller 2 (EEC2)	0C F0 03 00	29
61444	Electronic Engine Controller 1 (EEC1)	0C F0 04 00	512
61444	Electronic Engine Controller 1 (EEC1)	0C F0 04 00	513
61444	Electronic Engine Controller 1 (EEC1)	0C F0 04 00	190
65247	Electronic Engine Controller 3 (EEC3)	18 FE DF 00	515
65247	Electronic Engine Controller 3 (EEC3)	18 FE DF 00	3236
65247	Electronic Engine Controller 3 (EEC3)	18 FE DF 00	514
65247	Electronic Engine Controller 3 (EEC3)	18 FE DF 00	515
65262	Engine Temperature 1 (ET1)	18 FE EE 00	110
65262	Engine Temperature 1 (ET1)	18 FE EE 00	174
65266	Fuel Economy (Liquid) (FLE1)	18 FE F2 00	183
65266	Fuel Economy (Liquid) (FLE1)	18 FE F2 00	51
64981	Electronic Engine Controller 5 (EEC5)	18 FD D5 00	2791
65270	Inlet/Exhaust Conditions 1 (IE1)	18 FE F6 00	102
65270	Inlet/Exhaust Conditions 1 (IE1)	18 FE F6 00	105
65270	Inlet/Exhaust Conditions 1 (IE1)	18 FE F6 00	106
65270	Inlet/Exhaust Conditions 1 (IE1)	18 FE F6 00	173
64877	After Treatment 1 Air Control 2	18 FD 6D 91	3832
64877	After Treatment 1 Air Control 2	18 FD 6D 91	3837
64908	Aftertreatment 1 Gas Parameters	18 FD 8C 91	3609
64927	Aftertreatment 1 Air Control 1	18 FD 9F 91	3487
64927	Aftertreatment 1 Air Control 1	18 FD 9F 91	3488
64948	Aftertreatment 1 Intake Gas 2	18 FD B4 91	3241
64948	Aftertreatment 1 Intake Gas 2	18 FD B4 91	3242
64830	Aftertreatment 1 SCR Reagent Information	18 FD 3E 91	4360
64830	Aftertreatment 1 SCR Reagent Information	18 FD 3E 91	4363
65194	Alternate Fuel 2	18 FE AA 00	1693
65194	Alternate Fuel 2	18 FE AA 00	1694

Table A.1: List of selected signals of SAE J1939

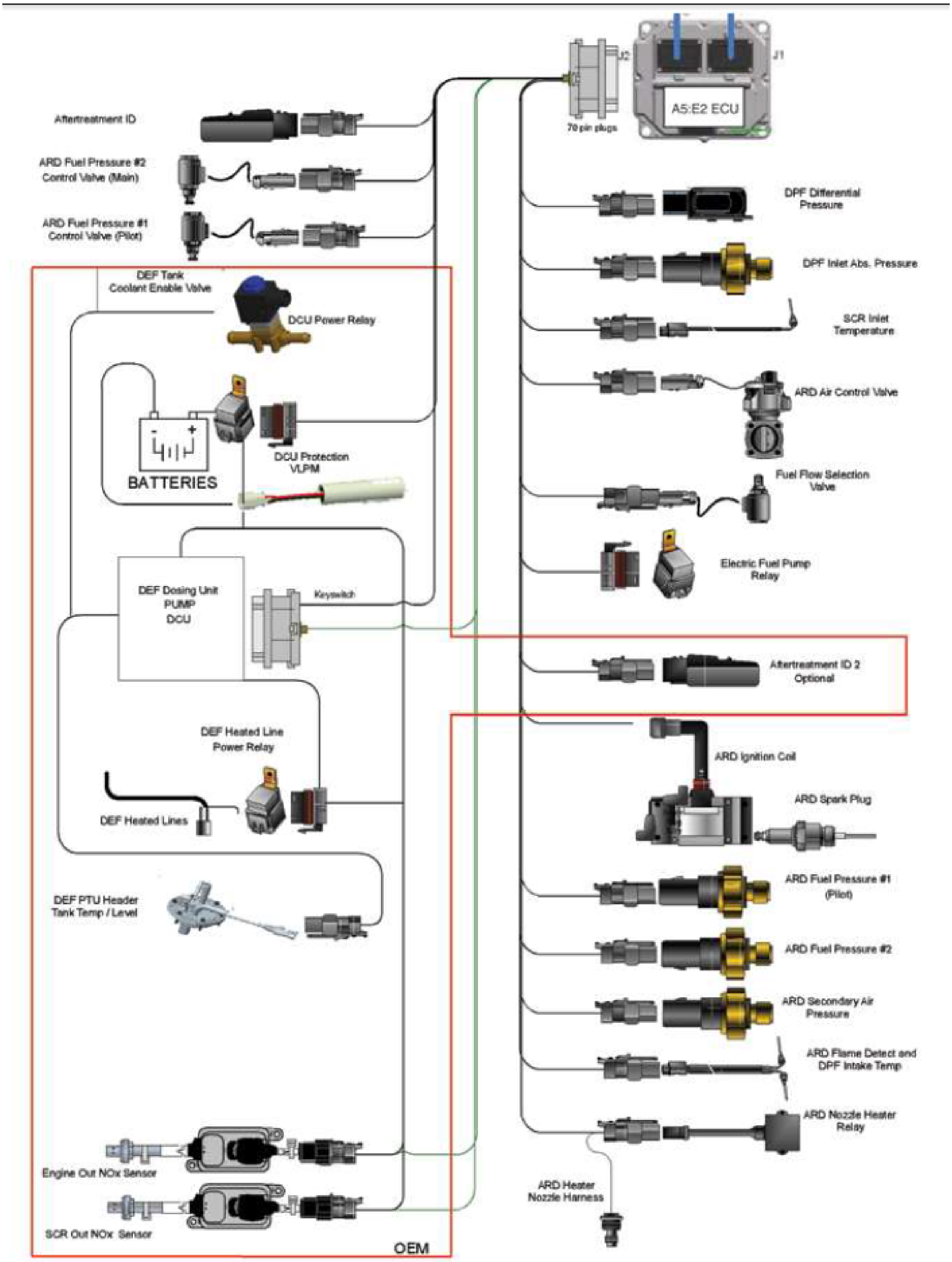


Figure A.10: The CAT C9.3 diesel engined installed sensors(2).

A.3 dSpace MicroAutobox II data and electric designs

- Analogue I/O : 0-5 V

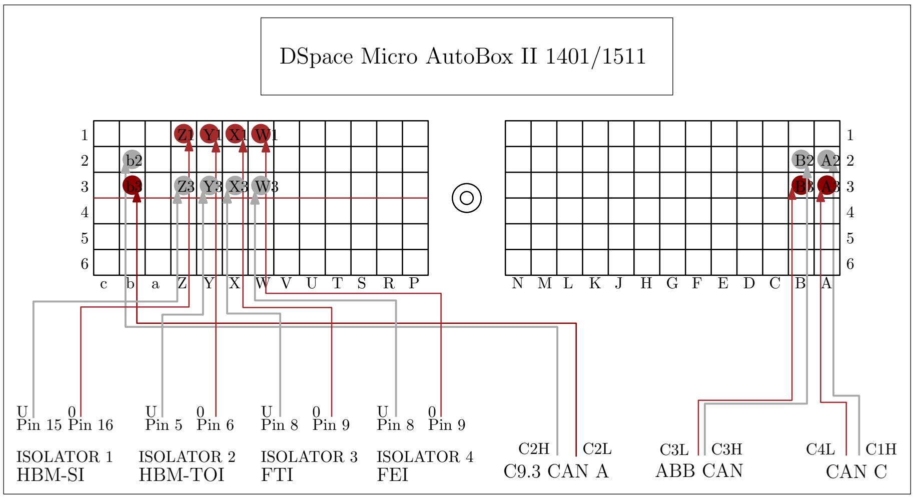
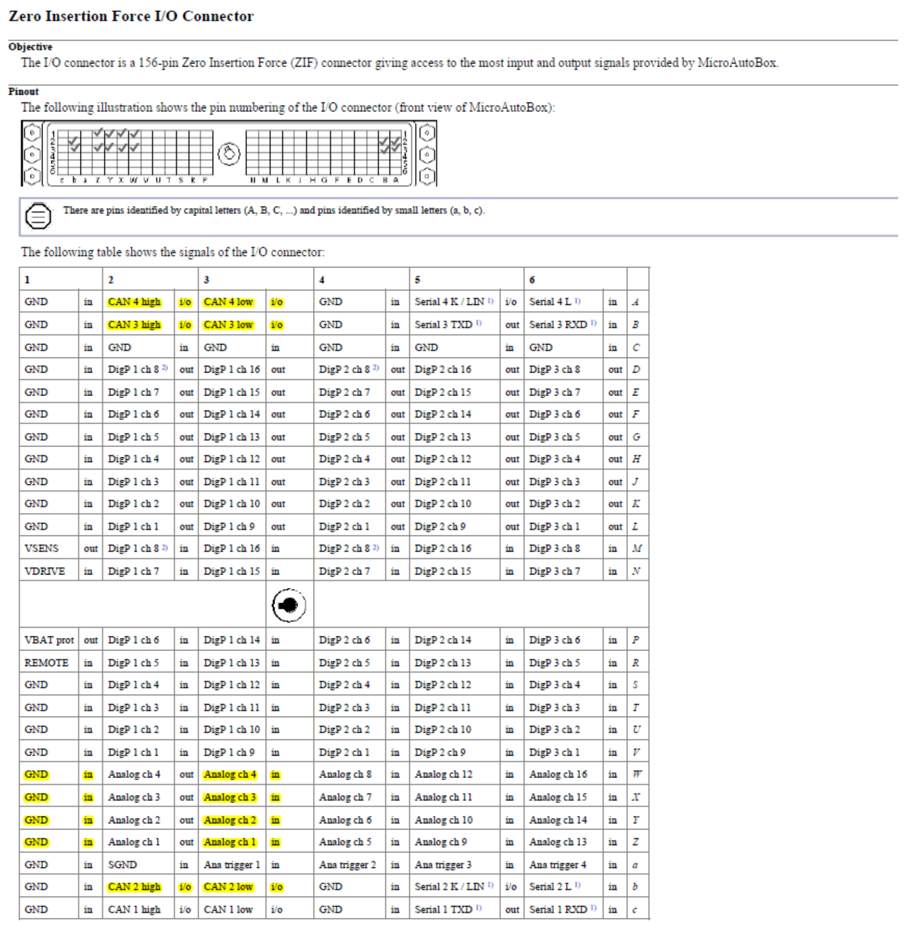


Figure A.11: ZIF connector electric drawing for dSpace MicroAutobox II



Zero Insertion Force I/O Connector

Objective

The I/O connector is a 156-pin Zero Insertion Force (ZIF) connector giving access to the most input and output signals provided by MicroAutoBox.

Pinout

The following illustration shows the pin numbering of the I/O connector (front view of MicroAutoBox):

There are pins identified by capital letters (A, B, C, ...) and pins identified by small letters (a, b, c).

The following table shows the signals of the I/O connector:

1		2		3		4		5		6		
GND	in	CAN 4 high	i/o	CAN 4 low	i/o	GND	in	Serial 4 K / LIN 1)	i/o	Serial 4 L 1)	in	*A*
GND	in	CAN 3 high	i/o	CAN 3 low	i/o	GND	in	Serial 3 TXD 1)	out	Serial 3 RXD 1)	in	*B*
GND	in	GND	in	GND	in	GND	in	GND	in	GND	in	*C*
GND	in	DigP 1 ch 8 2)	out	DigP 1 ch 16	out	DigP 2 ch 8 2)	out	DigP 2 ch 16	out	DigP 3 ch 8	out	*D*
GND	in	DigP 1 ch 7	out	DigP 1 ch 15	out	DigP 2 ch 7	out	DigP 2 ch 15	out	DigP 3 ch 7	out	*E*
GND	in	DigP 1 ch 6	out	DigP 1 ch 14	out	DigP 2 ch 6	out	DigP 2 ch 14	out	DigP 3 ch 6	out	*F*
GND	in	DigP 1 ch 5	out	DigP 1 ch 13	out	DigP 2 ch 5	out	DigP 2 ch 13	out	DigP 3 ch 5	out	*G*
GND	in	DigP 1 ch 4	out	DigP 1 ch 12	out	DigP 2 ch 4	out	DigP 2 ch 12	out	DigP 3 ch 4	out	*H*
GND	in	DigP 1 ch 3	out	DigP 1 ch 11	out	DigP 2 ch 3	out	DigP 2 ch 11	out	DigP 3 ch 3	out	*J*
GND	in	DigP 1 ch 2	out	DigP 1 ch 10	out	DigP 2 ch 2	out	DigP 2 ch 10	out	DigP 3 ch 2	out	*K*
GND	in	DigP 1 ch 1	out	DigP 1 ch 9	out	DigP 2 ch 1	out	DigP 2 ch 9	out	DigP 3 ch 1	out	*L*
VSENS	out	DigP 1 ch 8 2)	in	DigP 1 ch 16	in	DigP 2 ch 8 2)	in	DigP 2 ch 16	in	DigP 3 ch 8	in	*M*
VDRIVE	in	DigP 1 ch 7	in	DigP 1 ch 15	in	DigP 2 ch 7	in	DigP 2 ch 15	in	DigP 3 ch 7	in	*N*
					◉							
VBAT prot	out	DigP 1 ch 6	in	DigP 1 ch 14	in	DigP 2 ch 6	in	DigP 2 ch 14	in	DigP 3 ch 6	in	*P*
REMOTE	in	DigP 1 ch 5	in	DigP 1 ch 13	in	DigP 2 ch 5	in	DigP 2 ch 13	in	DigP 3 ch 5	in	*R*
GND	in	DigP 1 ch 4	in	DigP 1 ch 12	in	DigP 2 ch 4	in	DigP 2 ch 12	in	DigP 3 ch 4	in	*S*
GND	in	DigP 1 ch 3	in	DigP 1 ch 11	in	DigP 2 ch 3	in	DigP 2 ch 11	in	DigP 3 ch 3	in	*T*
GND	in	DigP 1 ch 2	in	DigP 1 ch 10	in	DigP 2 ch 2	in	DigP 2 ch 10	in	DigP 3 ch 2	in	*U*
GND	in	DigP 1 ch 1	in	DigP 1 ch 9	in	DigP 2 ch 1	in	DigP 2 ch 9	in	DigP 3 ch 1	in	*V*
GND	in	Analog ch 4	out	Analog ch 4	in	Analog ch 8	in	Analog ch 12	in	Analog ch 16	in	*W*
GND	in	Analog ch 3	out	Analog ch 3	in	Analog ch 7	in	Analog ch 11	in	Analog ch 15	in	*X*
GND	in	Analog ch 2	out	Analog ch 2	in	Analog ch 6	in	Analog ch 10	in	Analog ch 14	in	*Y*
GND	in	Analog ch 1	out	Analog ch 1	in	Analog ch 5	in	Analog ch 9	in	Analog ch 13	in	*Z*
GND	in	SGND	in	Ana trigger 1	in	Ana trigger 2	in	Ana trigger 3	in	Ana trigger 4	in	*a*
GND	in	CAN 2 high	i/o	CAN 2 low	i/o	GND	in	Serial 2 K / LIN 1)	i/o	Serial 2 L 1)	in	*b*
GND	in	CAN 1 high	i/o	CAN 1 low	i/o	GND	in	Serial 1 TXD 1)	out	Serial 1 RXD 1)	in	*c*

Figure A.12: ZIF connections with HIPPO-2

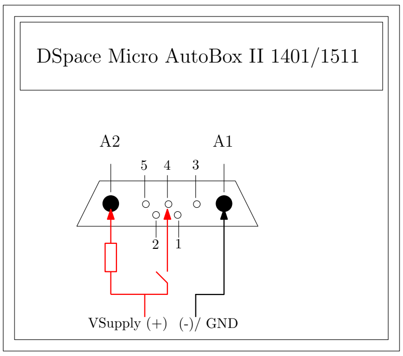


Figure A.13: Power Supply electric drawing for dSpace MicroAutobox II

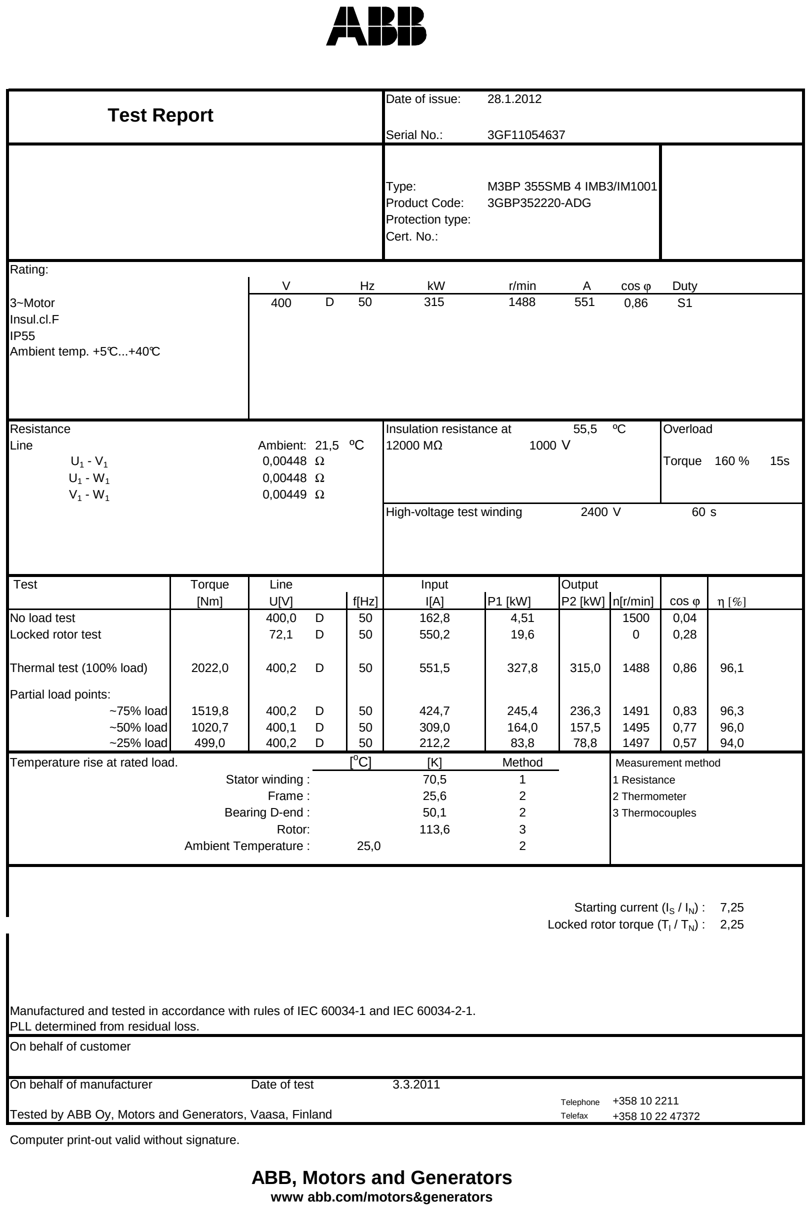
A.4 ABB data

Input	U	3 380...415 V
	I	153 A
	f	48...63 Hz
	Icc	65 kA
Output	U	3 0...UInput V
	I	165 A
	f	0...300 Hz

Table A.2: EM ABB drive ACS800-17-0100-3 +E202+F250+K457+L504+M600+N678+Q951 data

Input	U	3 380...415 V
	I	509 A
	f	48...63 Hz
	Icc	65 kA
Output	U	3 0...UInput V
	I	565 A
	f	0...300 Hz

Table A.3: EB ABB drive ACS800-17-0390-3 +E202+H359+K457+L504+M601+N678+Q951 data



ABB

Test Report

Date of issue: 28.1.2012
Serial No.: 3GF11054637

Type: M3BP 355SMB 4 IMB3/IM1001
Product Code: 3GBP352220-ADG
Protection type:
Cert. No.:

Rating:

	V		Hz	kW	r/min	A	cos φ	Duty
3~Motor	400	D	50	315	1488	551	0,86	S1

Insul.cl.F
IP55
Ambient temp. +5℃...+40℃

Resistance
Line Ambient: 21,5 °C

U_1 - V_1	0,00448	Ω
U_1 - W_1	0,00448	Ω
V_1 - W_1	0,00449	Ω

Insulation resistance at 55,5 °C
12000 MΩ 1000 V

Overload
Torque 160 % 15s

High-voltage test winding 2400 V 60 s

Test	Torque [Nm]	Line U[V]		f[Hz]	Input I[A]	P1 [kW]	Output P2 [kW]	n[r/min]	cos φ	η [%]
No load test		400,0	D	50	162,8	4,51		1500	0,04	
Locked rotor test		72,1	D	50	550,2	19,6		0	0,28	
Thermal test (100% load)	2022,0	400,2	D	50	551,5	327,8	315,0	1488	0,86	96,1
Partial load points:										
~75% load	1519,8	400,2	D	50	424,7	245,4	236,3	1491	0,83	96,3
~50% load	1020,7	400,1	D	50	309,0	164,0	157,5	1495	0,77	96,0
~25% load	499,0	400,2	D	50	212,2	83,8	78,8	1497	0,57	94,0

Temperature rise at rated load.	[°C]	[K]	Method
Stator winding :		70,5	1
Frame :		25,6	2
Bearing D-end :		50,1	2
Rotor:		113,6	3
Ambient Temperature :	25,0		2

Measurement method
1 Resistance
2 Thermometer
3 Thermocouples

Starting current (I_S / I_N) : 7,25
Locked rotor torque (T_l / T_N) : 2,25

Manufactured and tested in accordance with rules of IEC 60034-1 and IEC 60034-2-1.
PLL determined from residual loss.

On behalf of customer

On behalf of manufacturer Date of test 3.3.2011

Tested by ABB Oy, Motors and Generators, Vaasa, Finland

Telephone +358 10 2211
Telefax +358 10 22 47372

Computer print-out valid without signature.

ABB, Motors and Generators
www abb.com/motors&generators

Figure A.14: ABB (EB) motor test report

Index	Name	Brake (EB)	Hybrid (EM)
20.01	Min Speed	-2200 rpm	-2200 rpm
20.02	Max Speed	0 rpm	0 rpm
99.05	Motor Nom Speed (1:1rpm)	1488	1483
99.06	Motor Nom Power (10:1kW)	315 kw	90 kw
27.01	Flux Optimisation (0=NO)	Yes	Yes

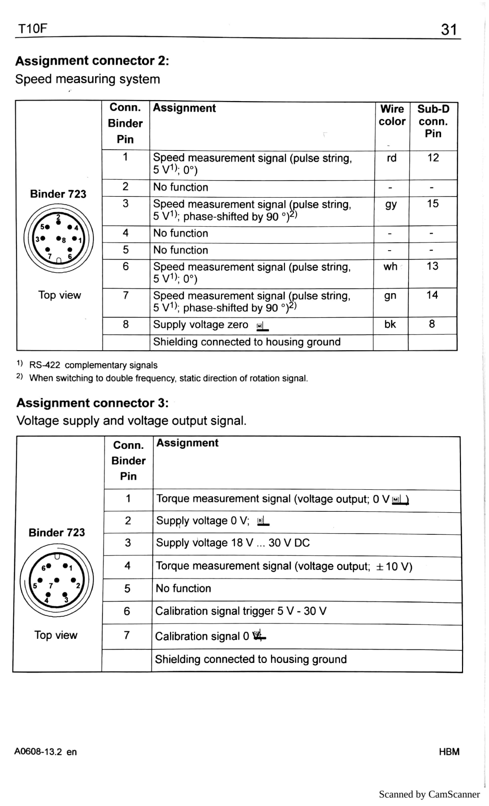
Table A.4: Setting points in ACS800

A.5 PDO messages

PDO21Tx		**PDO21Rx**	
MSW	08.01	MCW	07.01
Control Mode	01.25	Control Mode	26.01
Motor Speed Filt (rpm)	01.01	Speed reference	23.01
Motor Torque (%)	01.07	Torque reference %	25.01
PDO22Tx		**PDO23Tx**	
Motor Power (%)	01.09	Frequency (Hz)	01.05
Drive DC Voltage (V)	01.10	KWh	01.14
Motor AC Voltage (V)	01.11	USED MCW	02.26
Motor Current (A)	01.06	MCW	07.01

Table A.5: PDO messages

A.6 Analog Messages



T10F 31

Assignment connector 2:

Speed measuring system

	Conn. Binder Pin	Assignment	Wire color	Sub-D conn. Pin
Binder 723 Top view	1	Speed measurement signal (pulse string, 5 V[1); 0°)	rd	12
	2	No function	-	-
	3	Speed measurement signal (pulse string, 5 V[1); phase-shifted by 90 °)[2)	gy	15
	4	No function	-	-
	5	No function	-	-
	6	Speed measurement signal (pulse string, 5 V[1); 0°)	wh	13
	7	Speed measurement signal (pulse string, 5 V[1); phase-shifted by 90 °)[2)	gn	14
	8	Supply voltage zero ⏚	bk	8
		Shielding connected to housing ground		

[1) RS-422 complementary signals

[2) When switching to double frequency, static direction of rotation signal.

Assignment connector 3:

Voltage supply and voltage output signal.

	Conn. Binder Pin	Assignment
Binder 723 Top view	1	Torque measurement signal (voltage output; 0 V ⏚)
	2	Supply voltage 0 V; ⏚
	3	Supply voltage 18 V ... 30 V DC
	4	Torque measurement signal (voltage output; ± 10 V)
	5	No function
	6	Calibration signal trigger 5 V - 30 V
	7	Calibration signal 0 V ⏚
		Shielding connected to housing ground

A0608-13.2 en HBM

Figure A.15: Cable connection for T10F torque meter sensor

	HBM speed - FIlter		
Filter:	FDA tool		
Response Type:	Lowpass		
Design Method:	IIR Eliptic		
Filter Order:	Specify order:1		
Frequency Specs:	Units: HZ	Fs: 48000	Fpass: 50
Magnitude Specs:	Units: dB	Apass: 1	Astop: 100
	HBM speed calibration		
	GAIN after filtr.	3000	
	Constant	0	

Table A.6: Settings of HBM speed analog signal

	HBM torque - FIlter		
Filter:	FDA tool		
Response Type:	Lowpass		
Design Method:	IIR Eliptic		
Filter Order:	Specify order:1		
Frequency Specs:	Units: HZ	Fs: 480	Fpass: 20
Magnitude Specs:	Units: dB	Apass: 1	Astop: 100
	HBM torque calibration		
	GAIN after filtr.	-4000	
	Constant before filtr.	0,489785	
	Sample time:	-1	

Table A.7: Settings of HBM speed torque signal

	Fuel flow Return - Filter		
Filter:	FDA tool		
Response Type:	Lowpass		
Design Method:	IIR Eliptic		
Filter Order:	Specify order:1		
Frequency Specs:	Units: HZ	Fs: 100	Fpass: 7
Magnitude Specs:	Units: dB	Apass: 1	Astop: 100
	Fuel flow Return calibration		
	GAIN	10	
	Constant	50	
	Product before filtr.	Gain, Constant	

Table A.8: Settings of Return Fuel Flow analog signal

	Fuel flow Feed - Filter		
Filter:	FDA tool		
Response Type:	Lowpass		
Design Method:	IIR Eliptic		
Filter Order:	Specify order:1		
Frequency Specs:	Units: HZ	Fs: 100	Fpass: 7
Magnitude Specs:	Units: dB	Apass: 1	Astop: 100
	Fuel flow Feed calibration		
	GAIN	10	
	Constant	50	
	Product before filtr.	Gain, Constant	

Table A.9: Settings of Feed Fuel Flow analogue signal

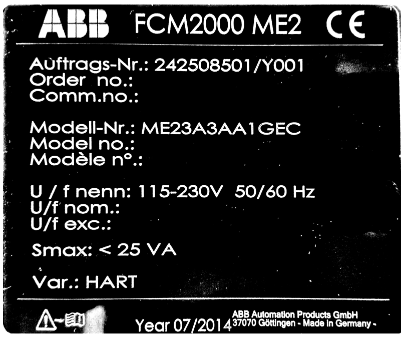


ABB FCM2000 ME2 CE

Auftrags-Nr.: 242508501/Y001
Order no.:
Comm.no.:

Modell-Nr.: ME23A3AA1GEC
Model no.:
Modèle n°.:

U / f nenn: 115-230V 50/60 Hz
U/f nom.:
U/f exc.:

Smax: < 25 VA

Var.: HART

Year 07/2014
ABB Automation Products GmbH
37070 Göttingen - Made in Germany -

Figure A.16: Data label from Coriolis Flow meter

Bibliography

[1] Robert Bosch GmbH 1991 CAN Specification Version 2.0 $bosch - semiconductors.de/media/ubk_{s}emiconductors/pdf_{1}/canliteratur/can2spec.pdf$

[2] Steve Corrigan SLOA101BAugust 2002 Introduction to the Controller Area Network (CAN) *ti.com/lit/an/sloa*101*b*/*sloa*101*b.pdf*

[3] Sunil Kumar Reddy Gurram 2011 Thesis Implementation of Controller Area Network (CAN) bus in an autonomous All-Terrain vehicle

[4] https://www.can-cia.org/

[5] Pfeiffer, O.; Ayre, A. & Keydel, C. 2008. Embedded Networking with CAN and CANopen. *Copperhill Technologies Corporation.*

[6] Olaf Pfeiffer, Christian Keydel, Andrew Ayre 2005. CANopen on general serial networks. *CAN in Automation, iCC.*

[7] dSPACE. Technical Manual of dSPACE RapidPro and MicroAutoBox, 2013.

[8] CSS Electronics, www.csselectronics.com. *css electronics.*

[9] http://www.byteme.org.uk/canopenparent/canopen/

[10] User's Manual CANopen Adapter Module RCAN-01 - 3AFE64504231 REV B EN / EFFECTIVE: 16.12.2008

[11] Firmware manual - System Control Program of ACS800 ABB - 3AFE64670646 REV H EN / EFFECTIVE: 2014-06-02

[12] HBM Torque Flange T10F mounting instructions - A0608-13.1 en