

OCORA

Open CCS On-board Reference Architecture

Modelling activities intermediate report

WP04 Train Display System

This OCORA work is licensed under the dual licensing Terms EUPL 1.2 (Commission Implementing Decision (EU) 2017/863 of 18 May 2017) and the terms and condition of the Attributions- ShareAlike 3.0 Unported license or its national version (in particular CC-BY-SA 3.0 DE).



Document ID: OCORA-TWS01-210

Version: 1.01

Date: 31.01.2025

Management Summary

The ERTMS system allows the supervision and automatic control of trains. It is made up of sub-systems on board the trains. Among them, the EVC (European Vital Computer) performs the calculations, and the TDS (Train Display System) is a man-machine-interface which displays the information to the driver. These two pieces of equipment communicate with each other.

The technical specification for interoperability defines a set of standards which specify the ERTMS system. Some of these standards define the interface between two subsystems. Subset 121 specifies the interface between the EVC and the TDS.

The model-based engineering approach aims to improve the quality of system design. Indeed, this method relies on standards, processes and specific tools. It aims to obtain a formalized, structured and verifiable definition of systems. It is applied throughout the development cycle. Thus, this approach compensates for the weaknesses of the traditional approach of document specification written in natural language. This approach is part of the wider systems engineering approach which includes configuration management, requirements management, synthesis of the various specialities involved, and interface management. The specialities may be technical (aerodynamics, thermodynamics, etc.) or support (operating safety, cyber security, etc.). Thus, the model-based engineering approach is interwoven with the system engineering approach, and these two approaches feed off each other.

Matlab/Simulink is a development environment allowing the modelling of a system. Programming can be done visually by using predefined blocks or by writing scripts. The structure of the system can be represented by the definition of "subsystems". The behaviour of the system is implemented and can be simulated. Toolboxes contain ready-to-use functions dedicated to technical domains (automobiles, etc.), technologies (artificial intelligence, etc.), techniques (signal analysis, code verification, proof of concept, etc.).

In addition to the definition provided by subset 121, it is proposed to develop a model under Matlab/Simulink. The model implements the structure and behaviour specified in the subset. Test scenarios are used to dynamically explain the expected behaviour.

The objective is to obtain a set of specifications, in the form of the subset and the model, which:

- Clarifies the behaviour
- Justifies the requirements by verifying the specification through the implementation of tests.

Revision history

Version	Change Description	Initial	Date of change
1.00	Official version for OCORA Release R5	MT	10.11.2023
1.01	Editorial updates in the introduction chapter	ML	31.01.2025

Table of contents

1	Introduction	8
1.1	Purpose of the document.....	8
1.2	Applicability of the document	8
1.3	Context of the document.....	8
2	Modelling objectives	11
3	Scope.....	12
4	Modelling principles.....	13
4.1	Development environment.....	13
4.2	High Level Architecture.....	14
4.3	Modelling/Simulation process	15
4.4	Incremental implementation.....	16
4.5	Boundary.....	17
4.6	Configuration.....	17
5	Model architecture	18
5.1	Model Input	18
5.1.1	Test sequence	18
5.1.2	Definition of Message / Packet.....	19
5.2	Simulation output	21
5.2.1	Dashboard	21
5.2.2	Sequence viewer	22
5.2.3	Graphic	23
5.2.4	CSV file.....	23
5.3	Internal architecture	25
6	Simulation	27
6.1	Coverage	27
6.2	Sequence.....	27
6.2.1	Unit test	28
6.2.2	Operational test	32
7	Model description.....	37
7.1	Model hierarchy	37
7.2	Transmission models	37
7.2.1	CRC 32 bits calculation	37
7.2.2	Safety code.....	38
7.3	Reception models	39
7.4	Generic Block.....	44
7.4.1	Signal to byte array.....	45
8	Analysis of Subset 121	46
9	Conclusion.....	52
10	ANNEX – PACKETS.....	53

11	ANNEX Relation between variables and TDS generated views	55
	11.1.1 Relation between variable and graphical elements in data entry window	55
	11.1.2 Relation between variable and graphical elements in default window	56
	11.1.3 Sound	64
12	ANNEX – TDS class diagram	65
13	ANNEX - Data Modelling of Subset 121 Configuration.....	66
14	ANNEX - TEST CASE TEMPLATE	67

Table of figures

Figure 1 – ERTMS Screen.....	9
Figure 2 – ERTMS Standard	10
Figure 3 – SDTV2 modelling scope	12
Figure 4 – Simulink development and simulation environment	13
Figure 5 – high level architecture	14
Figure 6 – modelling phases	15
Figure 7 – Modelling process	15
Figure 8 – building model in the Simulink development environment	15
Figure 9 – Simulation process	16
Figure 10 – incremental implementation	17
Figure 11 – TDS configuration.....	17
Figure 12 – Model Architecture	18
Figure 13 – test sequence file	19
Figure 14 – application for message definition	21
Figure 15 - pacing button in development environment	21
Figure 16 – simulation dashboard	22
Figure 17 – sequence viewer	22
Figure 18 - signal logging	23
Figure 19 - data inspector menu button.....	23
Figure 20 – graphic.....	23
Figure 21 – Internal architecture.....	25
Figure 22 – interface controller	26
Figure 23 – Unit test – Connection request	29
Figure 24 – display architecture 1	46
Figure 25 – display architecture 2	46
Figure 26 - subset 121 fig 6 proposed improvement.....	48
Figure 27 - sequence of PD and MD	49
Figure 28 - extract from subset 121 packet structure (non safe).....	49
Figure 29 - extract from subset 121 packet structure (safe).....	49
Figure 30 - clarification of the display window sequence	50
Figure 31 - subset 121 § 6.3.1.2.....	50
Figure 32 - soft key screen key names.....	50
Figure 33 - extract from Subset 121 §10.3.4.2	51

Table of tables

Table 1 – EVC-TDS interface OSI Layer.....	12
Table 2 – model decomposition.....	25
Table 3 – list of test cases	28
Table 4 - TDS architecture comparison.....	47

References

Reader's note: please be aware that the numbers in square brackets, e.g. [1], as per the list of referenced documents below, is used throughout this document to indicate the references to external documents. Wherever a reference to a TSI-CCS SUBSET is used, the SUBSET is referenced directly (e.g., SUBSET-026). OCORA always reference to the latest available official version of the SUBSET, unless indicated differently.

- [1] OCORA-BWS01-010 – Release Notes
- [2] OCORA-BWS01-020 – Glossary
- [3] OCORA-BWS01-030 – Question and Answers
- [4] OCORA-BWS01-040 – Feedback Form
- [5] OCORA-BWS03-010 – Introduction to OCORA
- [6] OCORA-BWS03-020 – Guiding Principles
- [7] OCORA-BWS04-010 – Problem Statements
- [8] OCORA-BWS05-010 – Road Map
- [9] OCORA-TWS01-035 – CCS On-Board (CCS-OB) – Architecture
- [10] OCORA-TWS01-201 Train display system (TDS) discussion paper
- [11] PR EN 50716 - Cross-functional Software Standard for Railways
- [12] SUBSET-026 - System Requirements Specification – 4.0.0
- [13] ERA_ERTMS_015560 - ETCS Driver Machine Interface – 4.0.0
- [14] IEC61375-2-3 - TCN Train communication profile, Edition1.0, 2015-07
- [15] SUBSET-121 – TDS / ETCS On-board Interface FFFIS – 1.1.0
- [16] SUBSET-121_Conf – TDS / ETCS On-board Interface FFFIS Configuration – 1.1.0
- [17] OCORA-TWS01-201 Train display system (TDS) discussion paper

1 Introduction

1.1 Purpose of the document

The objective of this document is to specify the model of subset 121 by defining:

- the modelling principles
- modelling assumptions and scope
- the preliminary architecture
- the design and coding quality rules
- operational scenarios
- verification principles

This document is addressed to experts in the CCS domain and to any other person, interested in the OCORA concepts for on-board CCS. The reader is invited to provide feedback to the OCORA collaboration and can, therefore, engage in shaping OCORA. Feedback to this document and to any other OCORA documentation can be given by using the feedback form [\[4\]](#).

If you are a railway undertaking, you may find useful information to compile tenders for OCORA compliant CCS building blocks, for tendering complete on-board CCS system, or also for on-board CCS replacements for functional upgrades or for life-cycle reasons.

If you are an organization interested in developing on-board CCS building blocks according to the OCORA standard, information provided in this document can be used as input for your development.

1.2 Applicability of the document

The document is informative. Subsequent releases of this document will be developed based on a modular and iterative approach, evolving within the progress of the OCORA collaboration.

1.3 Context of the document

This document is published as part of the OCORA Release R5, together with the documents listed in the release notes [\[1\]](#). Before reading this document, it is recommended to read the Release Notes [\[1\]](#). If you are interested in the context and the motivation that drives OCORA we recommend to read the Introduction to OCORA [\[5\]](#), and the Problem Statements [\[7\]](#). The reader should also be aware of the Glossary [\[2\]](#) and the Question and Answers [\[3\]](#).

The document presents the results of work conducted within the scope of the OCORA project that aims to specify generic interfaces between CCS on-board equipment (in particular interface identified as SCI-ETCS-DMI). In particular, the analysis of the ERTMS specifications (subset) is a part of this work.

This document is part of the work carried out within the WP04 Train Display system.

The Train Display System (TDS) facilitates interaction between the Control Command Signalling System (CCS) and the train driver. It accomplishes this through various windows (see Fig. 1), which serve the following purposes:

- Providing Information to the Driver:
 - Signalling Information: This includes details such as target distance, target speed, and operational mode.
 - Driving Information: This encompasses data related to train speed, planning information, and track conditions.
- Enabling Driver Interaction with the System:

- Data Entry: Drivers can input train-related data.
- Function Enablement: This allows for actions like override and activation of shunting mode.

The TDS ensures communication with other CCS and TCMS systems.

Refer to the Train Display System (TDS) discussion paper for a more detailed definition of the TDS.

The scope of TDS in this document is limited to the data exchange between ETCS and a display.

The standards relevant to the TDS include:

- ERA DMI [13]: This standard defines the expected behaviour and information displayed by the TDS.
- Subset-026 [12]: It specifies the ERTMS (European Rail Traffic Management System) system.
- Subset-121 [15]: This standard outlines the communication interface between the EVC (European Vital Computer) and the TDS. The primary objective of creating a model is to evaluate this specification (see also Figure 2).

In addition, the management of multiple train displays is also part of the work; i.e., capability of communication with two distinct sources as defined in subset-121 [15].

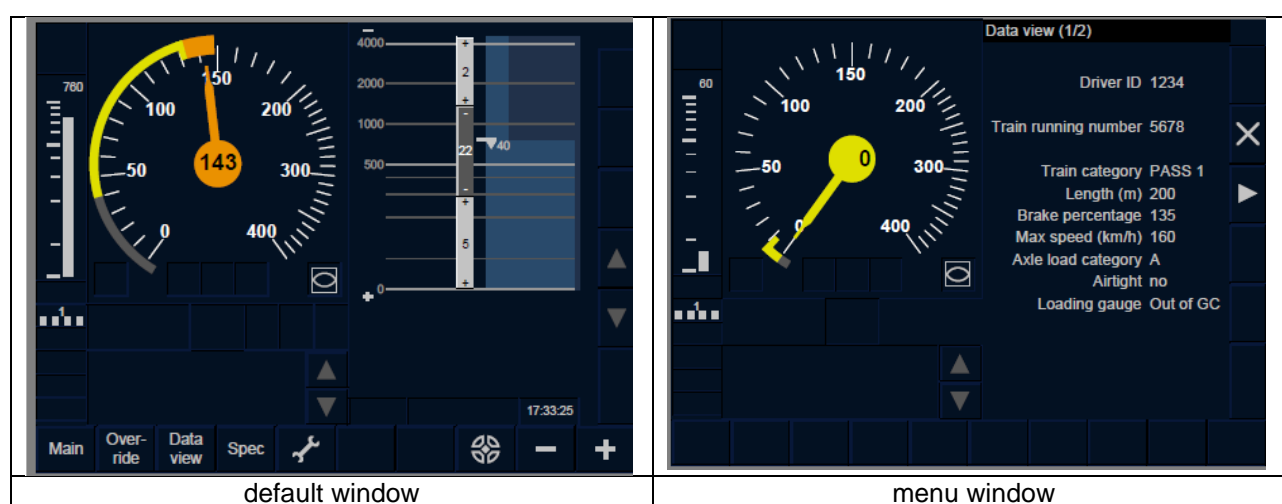


Figure 1 – ERTMS Screen

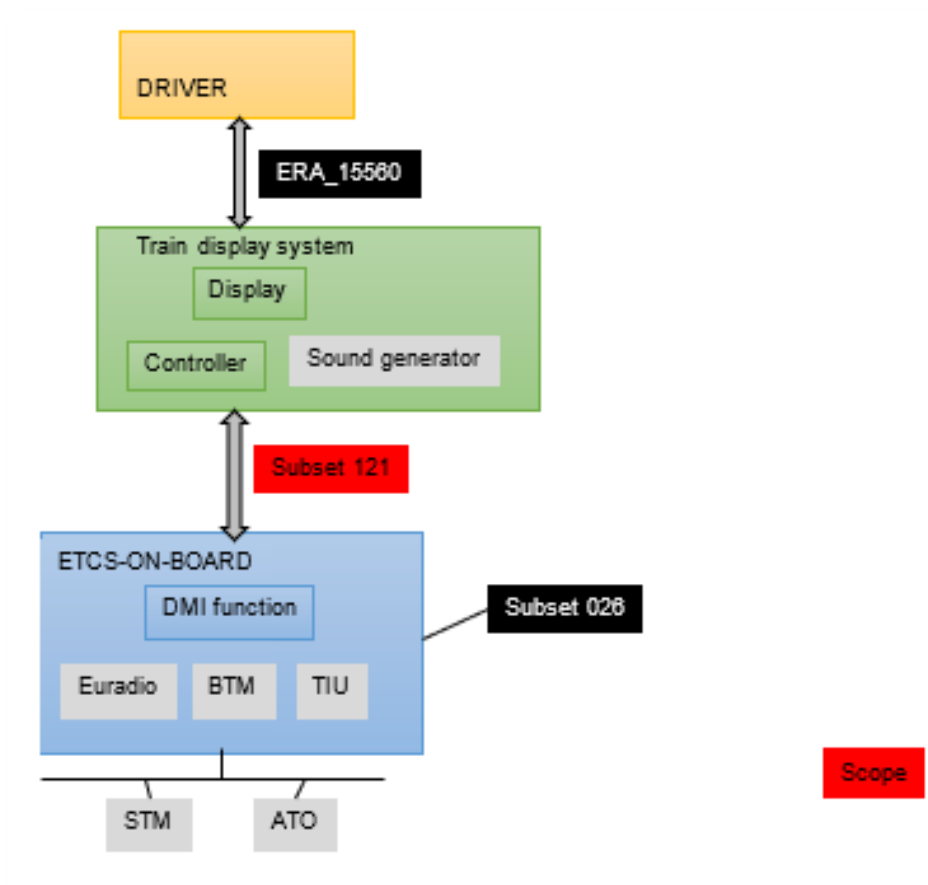


Figure 2 – ERTMS Standard

According to NF EN 50716 standard (evolution of 50128 and 50657) concerning railway software design, modelling is a highly recommended techniques for safety software (SIL 2 to SIL 4).

2 Modelling objectives

Model-Based Design (MBD) is a methodology that aims to provide methodologies, standards and tools to enhance overall system design. It focuses on early error detection, inherent traceability, and simplified communication among engineers by transitioning from a document-centric approach to a centralized, model-centric specification.

MBD facilitates design reuse, allowing the extraction and incorporation of models or their components into other models for various purposes. Representing a system as a virtual model enables the simulation of its behaviour and early testing, even during the specification phase.

Simulink, a graphical MBD environment, simplifies the complexity of coding by describing systems through hierarchical representations of blocks.

The objectives of translating the TDS/EVC interface specification (Subset 121) into a Simulink model include:

- **Verification of Specification:** This aims to assess the specification for completeness, consistency, and clarity. Creating a model helps in identifying gaps in the specification and resolving conflicting or ambiguous requirements.
- **Formalized Model-Based Specification:** The process seeks to transform the textual specification into a formalized model-based specification. The resulting model could serve as an annex to Subset 121.
- **Test Definition and Execution:** Finally, it aims to define and carry out tests based on the Simulink model."

3 Scope

Subset 121 specifies the interface between the European Vital Computer (EVC) and the Train Display System (TDS). From a modelling perspective, the primary focus will be on modelling the application layer and the Safe Data Transmission protocol (refer to the OSI Layer Table 1 below), as defined by IEC 61735, with enhancements outlined in Subset 121. It's important to note that the lower layers of the communication channel, as specified in Subset 147, are considered out of scope for this modelling effort.

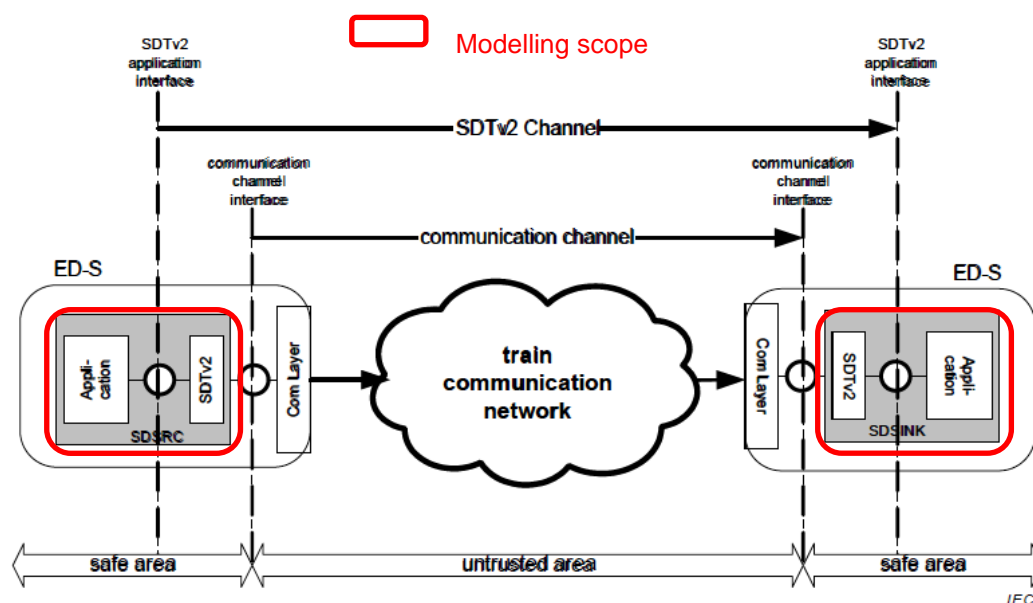


Figure 3 – SDTv2 modelling scope

Layer	
Application	Subset 121
	Safety layer: IEC 61375 ANNEX B – Safe Data Transmission SDT Subset 121 modification of SDT Subset 121 enhancement of SDT
Presentation	Subset 147
Session	
Transport	
Network	
Data link	
Physical	

Table 1 – EVC-TDS interface OSI Layer

4 Modelling principles

To meet the objective of assessment of the subset 121 the model shall implement these principles:

- The model shall translate requirements of the specification into functional model.
- The model shall implement behaviour of EVC and TDS in order to generate representative exchange on the interface.
- The model shall allow simulation of any nominal scenario which constitutes a representative sample of information exchanged on the interface.
- The model shall allow simulation of degraded mode of the interface.
- The model shall be configured outside the development environment in order to define a set of test cases.
- The model shall provide output outside the development environment that can be interpretable.

4.1 Development environment

The model is developed in the Matlab/Simulink development environment.

Simulink allows to model dynamic behaviour of the system. It's a graphical programming language based on a bloc library which enable generic function. Specific function can be coded in a script way. It enables simulation but also code generation which can be implemented in a hardware target.

The version 2022b is the version used for modelling.

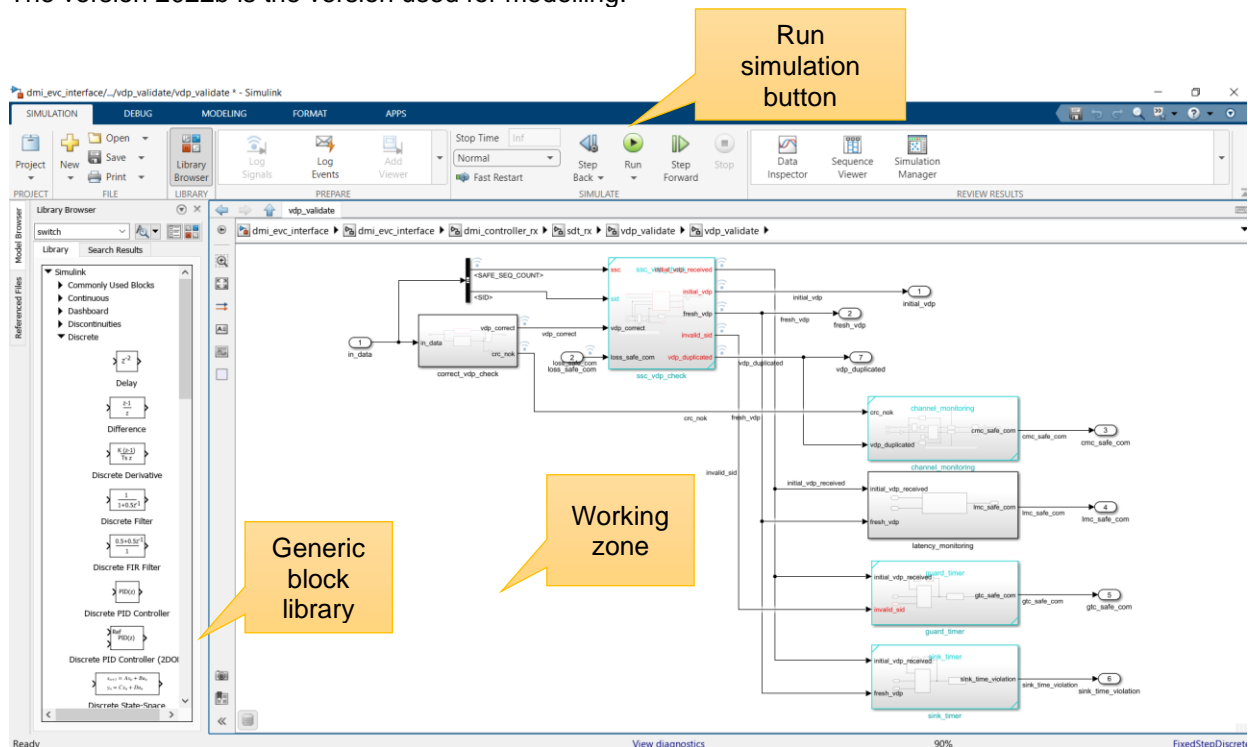


Figure 4 – Simulink development and simulation environment

4.2 High Level Architecture

The term “System under Test (SuT)” refers to the system being assessed.

The architecture of the model is decomposed in 3 elements:

- The model implements the specification of the interface as specified by the Subset 121.
- The data outside the model enables the parametrization of the model.
- The environment includes the models which represent the TDS and the EVC behaviour.

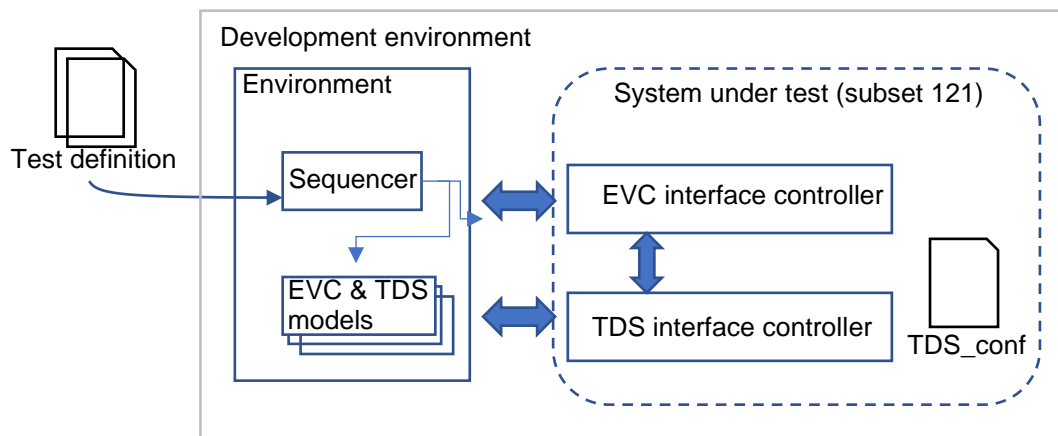


Figure 5 – high level architecture

- Test definition: configuration files where the tests are defined. It defines the input generated at each step of the test. This input can be of different format (simple variable or structure of data).
- Implemented inside the development environment.
 - The SuT environment
 - Sequencer: model which manage the successive steps of the simulation
 - model which represents the EVC and TDS behaviour and which generates application information exchanged through the communication interface.
 - System under test: modelling of EVC and TDS interface controller as defined by the Subset 121

In order to assess the robustness of the interface it shall be possible to inject error in the interface (message lost, duplicated, latency, etc.)

The OSI Layers lower than Application Layer are not modelled. They should be considered as black box.

4.3 Modelling/Simulation process

The modelling phases (refer to Figure 6 – modelling phases) follows these steps:

1. The definition of the model defines the objectives, the principles, the operational use, the architecture, the perimeter and the test scenarios.
2. The model implementation (refer to Figure 8 – building model in the Simulink development environment) consists in building the model in the development environment. Simulink is based on a library a functional blocks that can be drag-and-dropped in the working space. These blocks can be parametrized for a specific use. These blocks are combined (linked by signal) in order to produce a desired functionality. Blocks can be grouped into consistent subsystem. It is also possible to define a function by means of a script. Blocks can be linked by a signal (represented as a variable or a vector if mixed) or a bus (structured set of variables).
In parallel configuration files are defined according to the scenarios.
Verifications are conducted in order to control the correct operation.
3. Simulation (refer to Figure 9 – Simulation process) are run based on a specific configuration of the model including test sequence file.
The operator shall configure the simulation according to a specific scenario or a set of scenarios.
The operator shall manually launch the simulation.

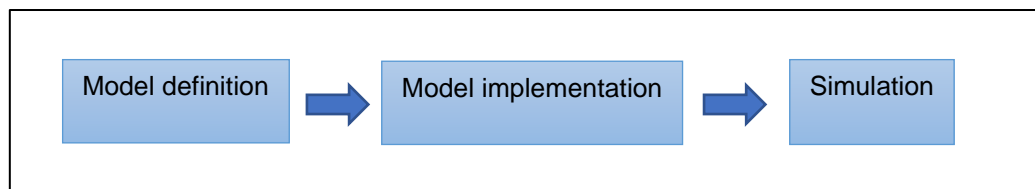


Figure 6 – modelling phases

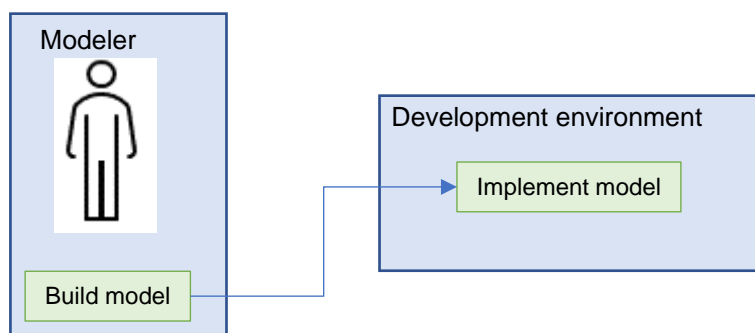


Figure 7 – Modelling process

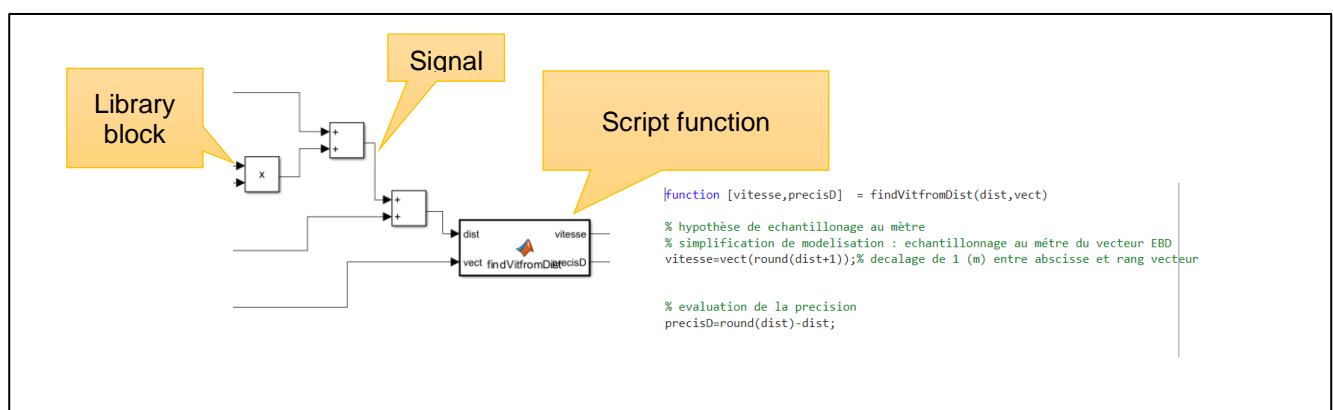


Figure 8 – building model in the Simulink development environment

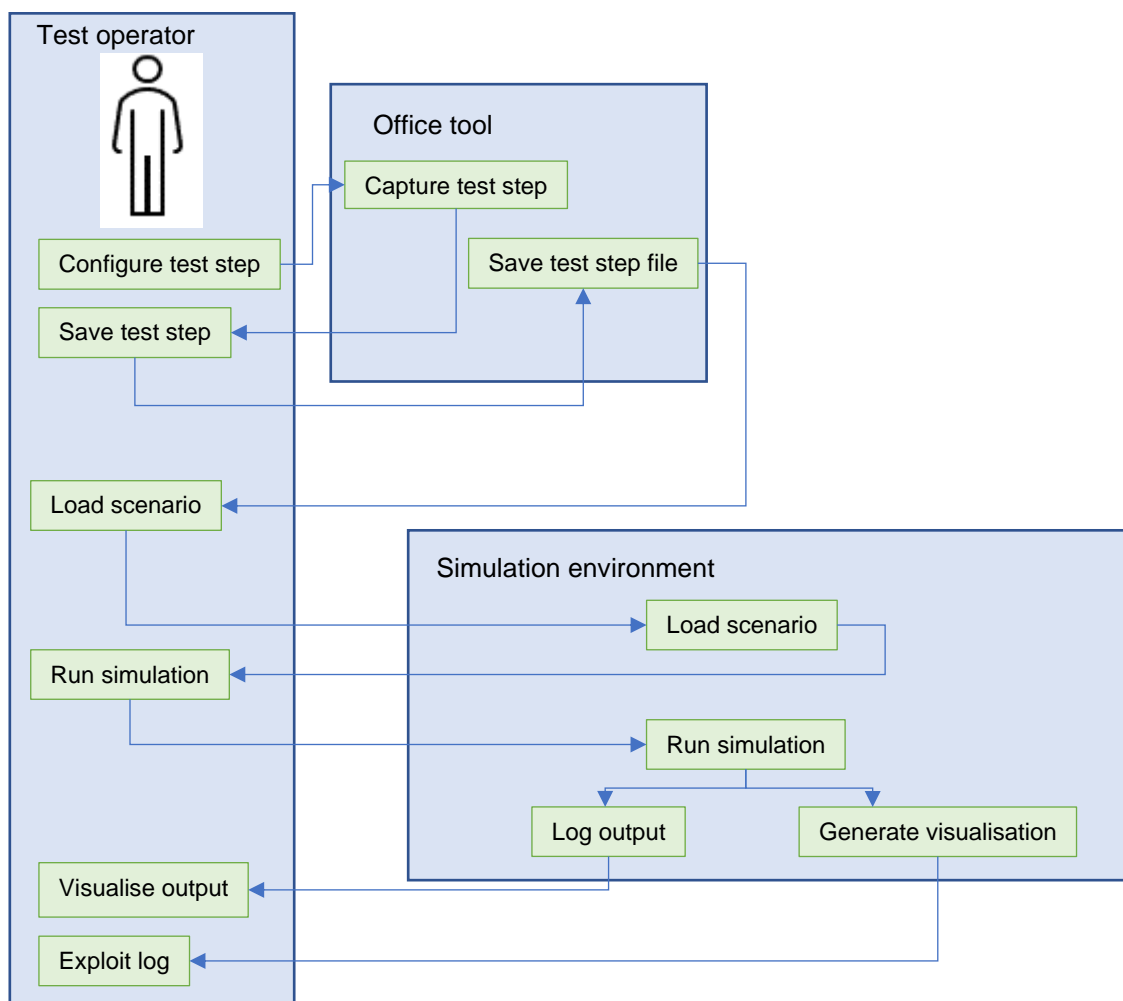


Figure 9 – Simulation process

4.4 Incremental implementation

The implementation of the model will be incremental.

1. Firstly, the backbone of the model will be implemented
 - a. The sequencer that loads the test parameters
 - b. The interface controllers
2. In a second step, the environmental model will be implemented
 - a. EVC “start of mission” function and the corresponding TDS function which allow to display the menu of this phase including data entry (ID, TRN, train configuration) which lead to Full supervision mode
 - b. EVC “speed and distance monitoring function” (braking curve, supervision limit, monitoring CSM/TSM/RSM as defined in subset 026 §3.13 [12]) and the corresponding TDS function (allowed speed and distance displayed in ceiling speed gauge and bar graph)

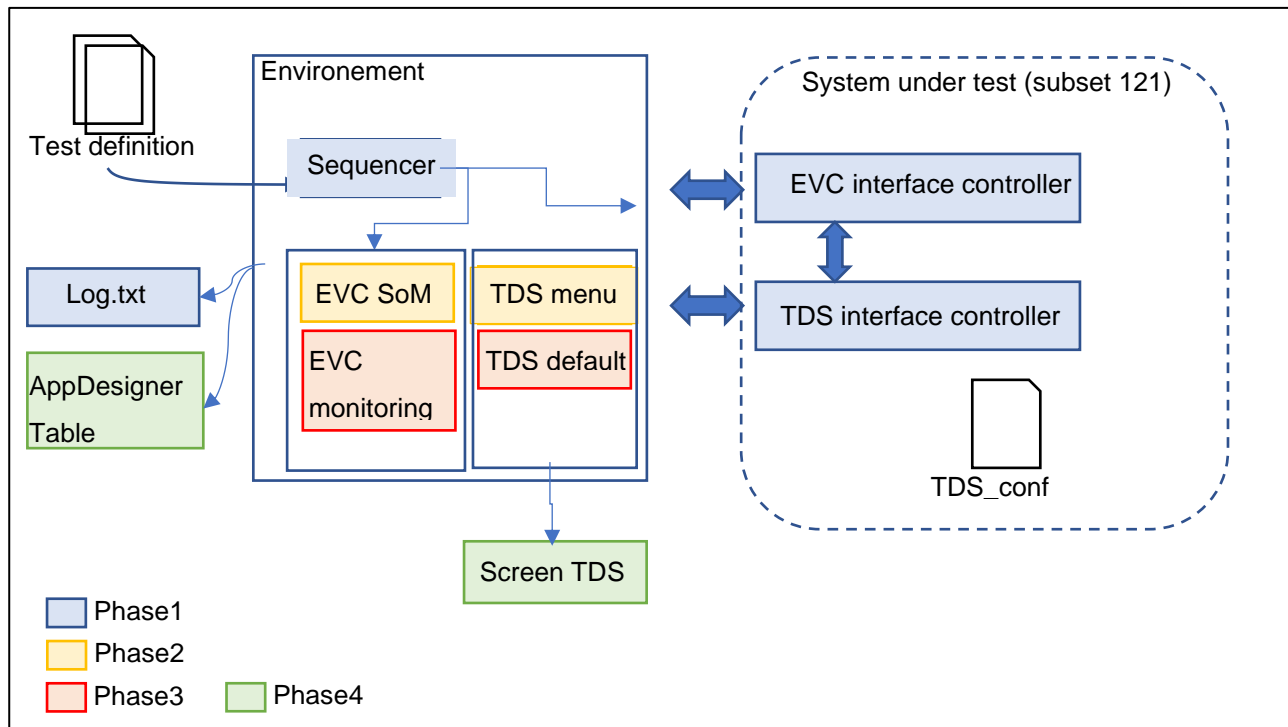


Figure 10 – incremental implementation

4.5 Boundary

These elements are excluded from the model:

- MVB is out of scope (only TCN-TCP due to control command network CCN as defined by OCORA)
- STMs are out of scope (due to simplification of the model)
- ATO system is out of scope (due to simplification of the model)

4.6 Configuration

The model shall represent the TDS configuration allowed by the subset 121 [15] including:

- Train configurations with single or dual cabin have to be modelled (see Figure 11 – TDS configuration)
- TDS Configuration soft key and touch screen
- Configuration as defined by [16]

Dual or single channel configuration between an EVC and a TDS (one channel at a time) are in the scope

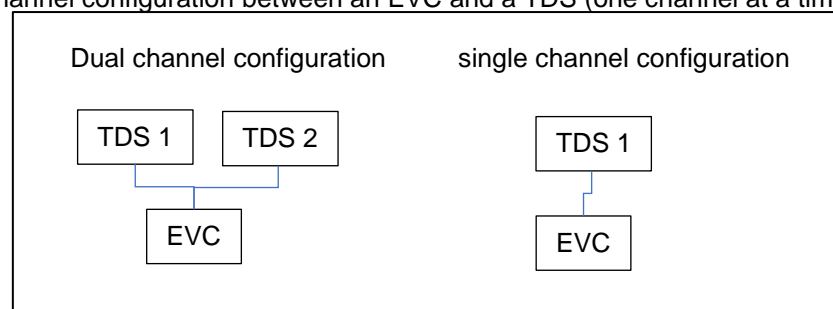


Figure 11 – TDS configuration

5 Model architecture

The model architecture is defined by the external links between the model and external actors, as well as by the logical components that compose the model.

The selection of a specific scenario by the operator enables the model to load the correct test configuration files. At the end of the simulation, the simulation environment computes graphical visualizations and writes log files that include variations of selected variables over time.

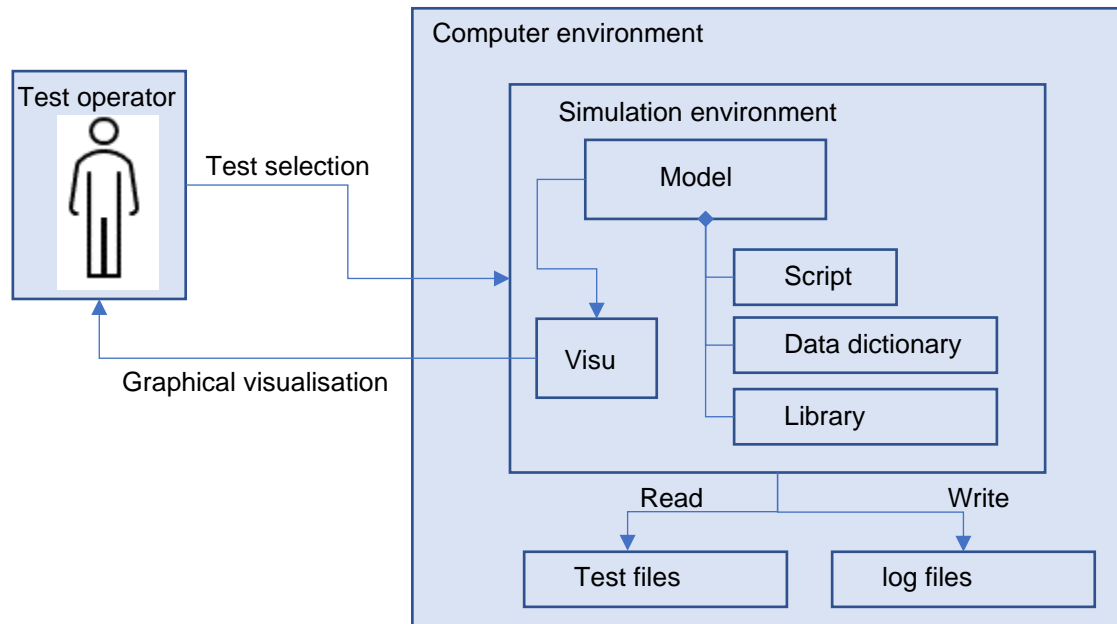


Figure 12 – Model Architecture

5.1 Model Input

To run a simulation, a model uses the definition of the sequence of the tests stored in a excel file. Message data called in the test sequence are defined in a separate file.

5.1.1 Test sequence

The test sequence is defined by a table describing a list of tests. This table will be implemented in a excel file.

There are two types of test steps:

- test steps generate output: a message data, or a change in a variable of a process data
- test steps that compare recorded signal with expected one.

Each test step is defined by:

- Column 1: the order of the steps
- Column 2: an input from the model used as a condition to execute the step: especially an odometer information
- Column 3: a time condition which has to be elapsed before the step is executed
- Column 4: to which model data are transmitted
- Column 5: the kind of data; message or process data
- Column 6: the port; the identifier of the process data
- Column 7:
 - For process data, the field of the modified variable
 - For message data: "SEQUENCE" directive
- Column 8:

- For process data: the value of the variable to change
- For message data: the number of the message among the list of the message stored in the xml file
- Column 9: the expected result
- Column 10: the recorded result

step	input	time_ms	source	type	port	field	value	read_input	result
1	0	0	EVC_Ctrl	MessageData		SEQUENCE	1		
2	0	0	EVC_Ctrl	ProcessData	PD_1_ETCS_TO_TDS	D_TUNNELSTOP	100		
2	0	0	EVC_Ctrl	ProcessData	PD_1_ETCS_TO_TDS	M_GEOPOSITION	15		
2	0	0	EVC_Ctrl	ProcessData	PD_1_ETCS_TO_TDS	T_TTI	14		
3	0	0	EVC_Ctrl	MessageData		SEQUENCE	1		
4	0	0	EVC_Ctrl	MessageData		SEQUENCE	2		
5	200	0	EVC_Ctrl	ProcessData	PD_1_ETCS_TO_TDS	D_TUNNELSTOP	50		
6	1000	0	Supervision	ProcessData		SET_SPEED	15		
6	1000	0	Supervision	ProcessData		DELTA_DIST	100		
6	1000	0	Supervision	ProcessData		TIME	20220725111810		

Figure 13 – test sequence file

5.1.2 Definition of Message / Packet

Message data is composed of packets. Packets are a variable sized set of variables. Message and packet are defined in a xml file.

Subset 121 §8.1.9 an §8.1.10 defined messages structure

Message header	
Packet (non safe)	NID_PACKET
	L_PACKET
	other variables
Packet (safe)	NID_PACKET
	L_PACKET
	other variables
	safety trailer

Example: Packet TDS-10: PKT_DATAENTRY_LABEL

NID_PACKET	10	
L_PACKET	64	
NID_NTC	14	Identifier of level NTC
N_ITER	1	
NID_DATA	25	Identifier of ETCS or STM specific data
L_LABEL	12	Length of a label string
X_LABEL	18	One byte of a string

Values are arbitrary set for the example

This structured definition of packets is translated in a xml format:

I

<pre> <Messages> <Message> <SEQUENCE>1</SEQUENCE> <LENGTH>6</LENGTH> <NB_PKT>2</NB_PKT> <Packet> <NID_PACKET>9</NID_PACKET> <L_PACKET>5</L_PACKET> <NID_LANGUAGE>1</NID_LANGUAGE> </Packet> <Packet> <NID_PACKET>22</NID_PACKET> <L_PACKET>11</L_PACKET> <D_PA>0</D_PA> <N_ITER>1</N_ITER> <d_section>1</d_section> <q_pa_sp>2</q_pa_sp> <v_pa_sp>3</v_pa_sp> </Packet> </Message> <Message> <SEQUENCE>2</SEQUENCE> <LENGTH>6</LENGTH> <NB_PKT>2</NB_PKT> <Packet> <NID_PACKET>9</NID_PACKET> <L_PACKET>5</L_PACKET> <NID_LANGUAGE>2</NID_LANGUAGE> </Packet> <Packet> <NID_PACKET>22</NID_PACKET> <L_PACKET>11</L_PACKET> <D_PA>0</D_PA> <N_ITER>1</N_ITER> <d_section>2</d_section> <q_pa_sp>3</q_pa_sp> <v_pa_sp>4</v_pa_sp> </Packet> </Message> </Messages> </pre>	<p>Specific balise are create in order to manage the test sequence. Especially to make the link between the test sequence file and the message data definition file:</p> <p>MESSAGES: include all the message of a test sequence</p> <p>MESSAGE: define a unique message</p> <p>PACKET: define a packet</p> <p>SEQUENCE: message number linked to the test sequence</p>
---	---

An application is developed which facilitate the design of the xml files. It provides a user interface which allow to build message exchanged by EVC and TDS.

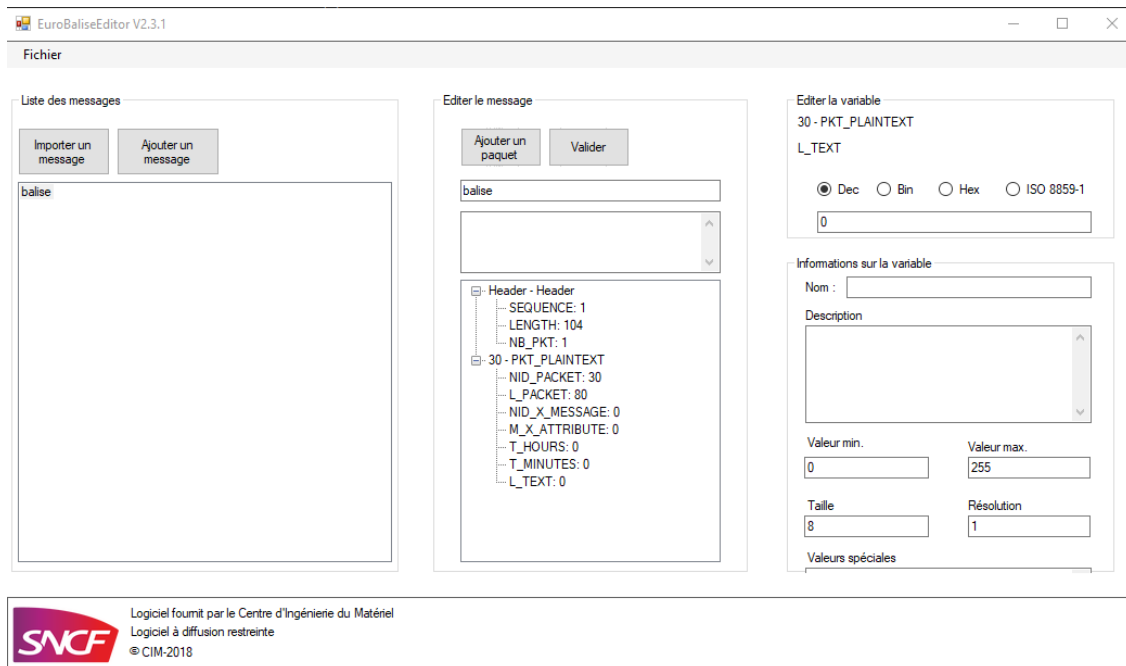


Figure 14 – application for message definition

5.2 Simulation output

Result of simulation are:

- Figure that shows the evolution of variable over time
- Test steps file updated with expected results
- Raw data stored in matlab file (.mat)

5.2.1 Dashboard

The simulation can be slowed down with the “simulation pacing” function.

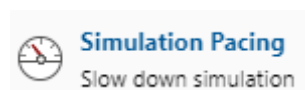


Figure 15 - pacing button in development environment

So, it is possible to visualize internal signal values of the model in real time and display their value with “display block”. “Dashboard” block allows also to display the variable in a more graphical way.

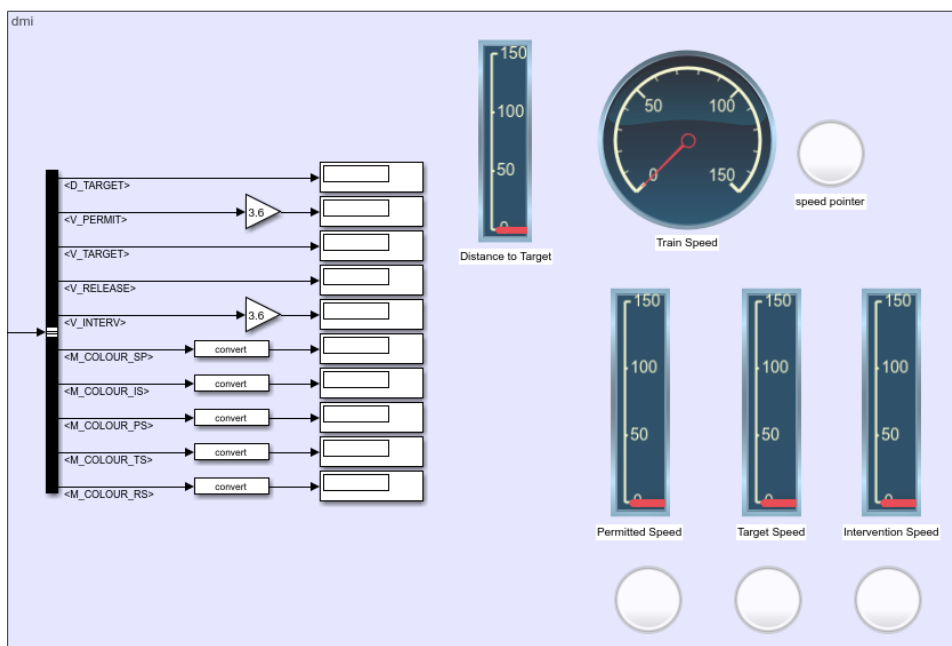


Figure 16 – simulation dashboard

5.2.2 Sequence viewer

The “sequence viewer” block in simulink allows to display exchange of data in simulink block. It is suitable to visualize the exchange between the interface controllers of the EVC and the TDS.

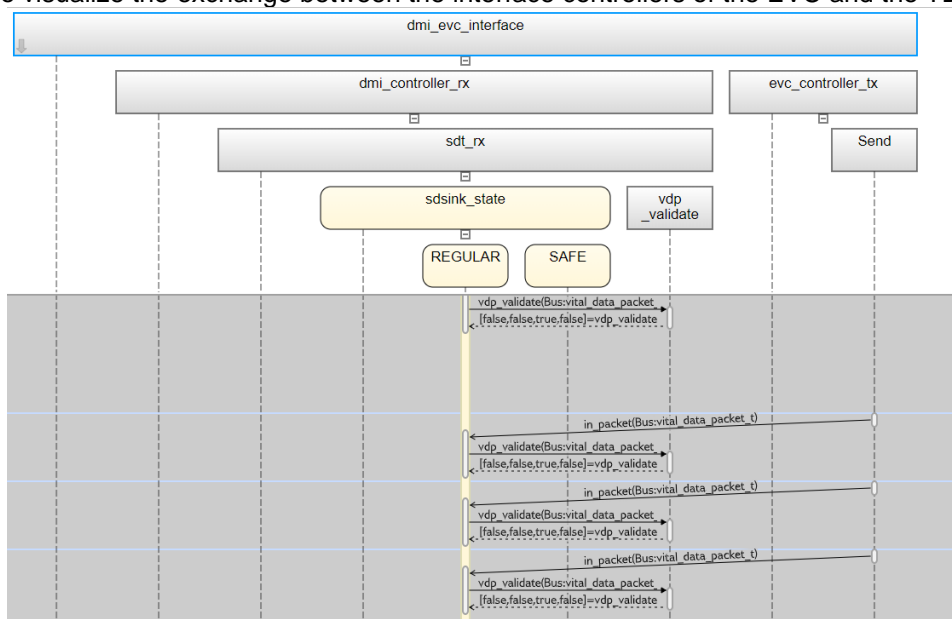


Figure 17 – sequence viewer

5.2.3 Graphic

Signal can be monitored during simulation.

A signal can be logged by selecting it and attaching the corresponding function.

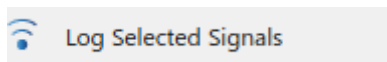


Figure 18 - signal logging

After simulation, all the signal can be viewed with the simulink embedded "data inspector" user interface.

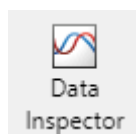


Figure 19 - data inspector menu button

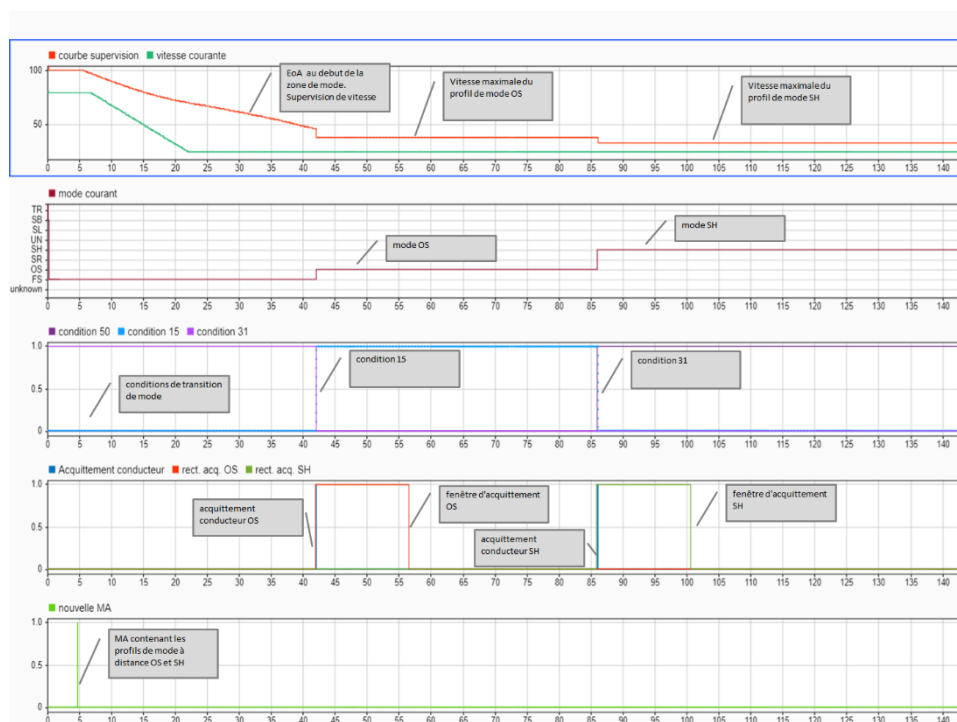


Figure 20 – graphic

5.2.4 CSV file

A *.csv file is produced, which contains the following information:

- Step
- Time (in ms)
- Source
- Destination
- Type of data: message or process
- Value: value of change of a process data, reference of a message data that is stored apart.

5.3 Internal architecture

The sequencer shall import test sequences file and according to each step shall produce command to subsystem.

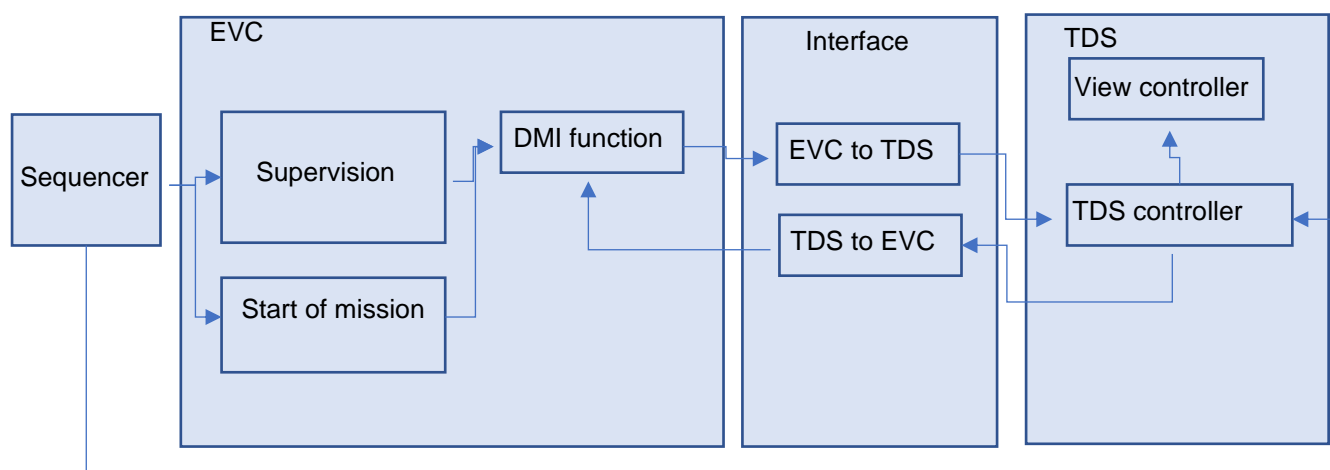


Figure 21 – Internal architecture

The commands produced by the sequencer as input to the models are as follows:

Model	Sequencer command
Supervision	Driver activity: set in motion, ...
Start of mission	Internal state of the EVC (position validity, ...), Train interface (cabin opening, ...)
TDS controller	interaction of the driver with the TDS: button press, entry of data

EVC models are implemented based on requirement of the subset 026 “system requirement specification”.

Model	Standard	Chapter
Supervision	Subset 026	§3.13 speed and distance monitoring
Start of mission	Subset 026	§5.4 Start of mission
DMI function	ERA DMI	§ 7 speed and distance monitoring §8.2.1.2 current train speed pointer §8.2.1.3 current train speed digital §8.2.1.4 circular speed gauge §8.2.1.5 basic speed hook §8.2.1.5 release speed §8.2.2.1 distance to target bar §8.2.3 supplementary driving information §10/11 sub level windows

Table 2 – model decomposition

Interface models are implemented based on requirement of the Subset 121 and IEC 61375-2-3 §SDTv2. A transmission model manages the sequence of process and data model. A transmission model evaluates the “vital data packet” according SDTv2 protocol: data version, checksum, sequence counter, channel monitoring, latency monitoring, guard timer, sink timer. The model evaluates the safe or regular state according to the result of vdp (vital data packet) evaluation.

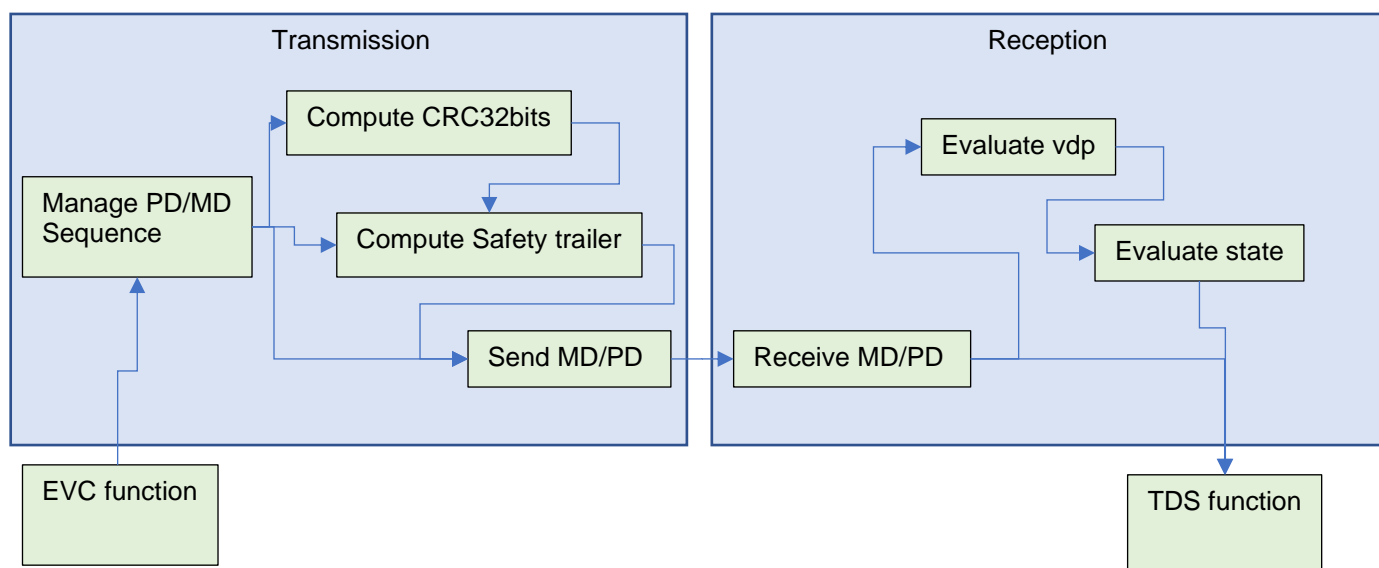


Figure 22 – interface controller

6 Simulation

The simulation aims to evaluate Subset 121 with respect to its capability to enable communication between EVC and TDS according to functional interface and performance.

Coverage criteria are defined to guide the evaluation effort.

Test sequences are defined based on a specific formalism (see annex) to specify the expected behaviour and to configure the steps in the test configuration file. The sequence model responsible for managing the steps is responsible for generating commands according to these definitions.

6.1 Coverage

A partitioning principle can be applied in order to determine the coverage.

In fact, the complexity and the very large number of parameters don't allow for 100% coverage with reasonable effort. That's why test cases are broken down into smaller and more manageable subsets based on criteria.

Especially, equivalence partitioning aims to identify classes or groups of software elements. Test cases are designed to cover each class at least once. By extension, we consider that if the test is successful for one member of the class, other members of the class are also likely to be good.

The following criteria shall be covered:

- All operational phases
 - Start of mission
 - Mission in full supervision
- All TDS elements have to be commanded by the EVC:
 - Ack
 - Button
 - Window
 - Menu
 - Data entry
 - Data view
 - Indication
 - Indicator
 - Planning area
 - Sound
 - Toggling
 - Text
 - Transfer STM data
- All packets included at least once in the different data exchanges: packet 1-33 & 44,45
- At least one change for each variable of a process data
- All Subset 121 and SDT mechanism shall be tested in nominal and degraded modes

6.2 Sequence

Two types of sequence are defined:

- Unit test which describes a sequence of exchange for a specific functionality
- Operational test which describes the behaviour of the whole system in a operational context and implies several functions

A preliminary lists of test case is established:

Type	Doc Ref	Chapter	Description	Nominal / Degraded
unit test	subset 121	§6.2.1.1	connection sequence	Nominal
unit test	subset 121	§6.2.1.2	ack button dialog	Nominal

			sequence	
unit test	subset 121	Tbd	button dialog sequence of an active window	Nominal
unit test	subset 121	Tbd	window management sequence	Nominal
unit test	subset 121	Tbd	menu management sequence	Nominal
unit test	IEC 61375	Tbd	increase of latency between process data	Degraded
unit test	IEC 61375	Tbd	wrong safety code	Degraded
unit test	IEC 61375	Tbd	change of SID (sender identifier) when sending process data	Degraded
operational test	Susbet 026	§6.2.2.1	start of mission – start up sequence	Nominal
operational test	subset 026	§6.2.2.2	start of mission – data entry	Nominal
operational test	subset 026	§6.2.2.3	mission in level 2 switching from ceiling speed to target speed to release speed monitoring	Nominal
operational test	subset 026	Tbd	level transition	Nominal
operational test	subset 026	Tbd	override	Nominal
operational test		Tbd	failure of a TDS panel and switch to other TDS	Degraded

Table 3 – list of test cases

6.2.1 Unit test

6.2.1.1 Connection request

Connection sequence to initiate communication between EVC and TDS.

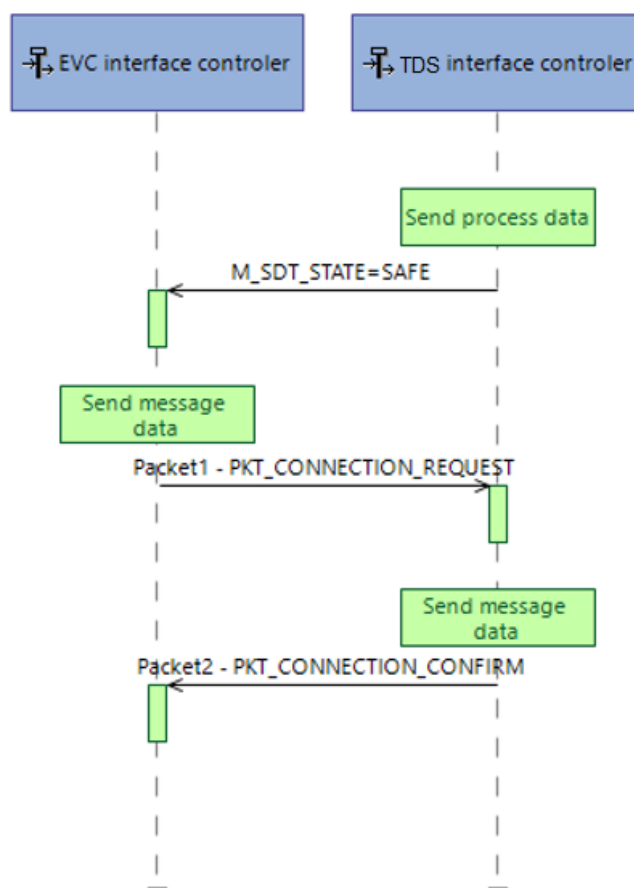
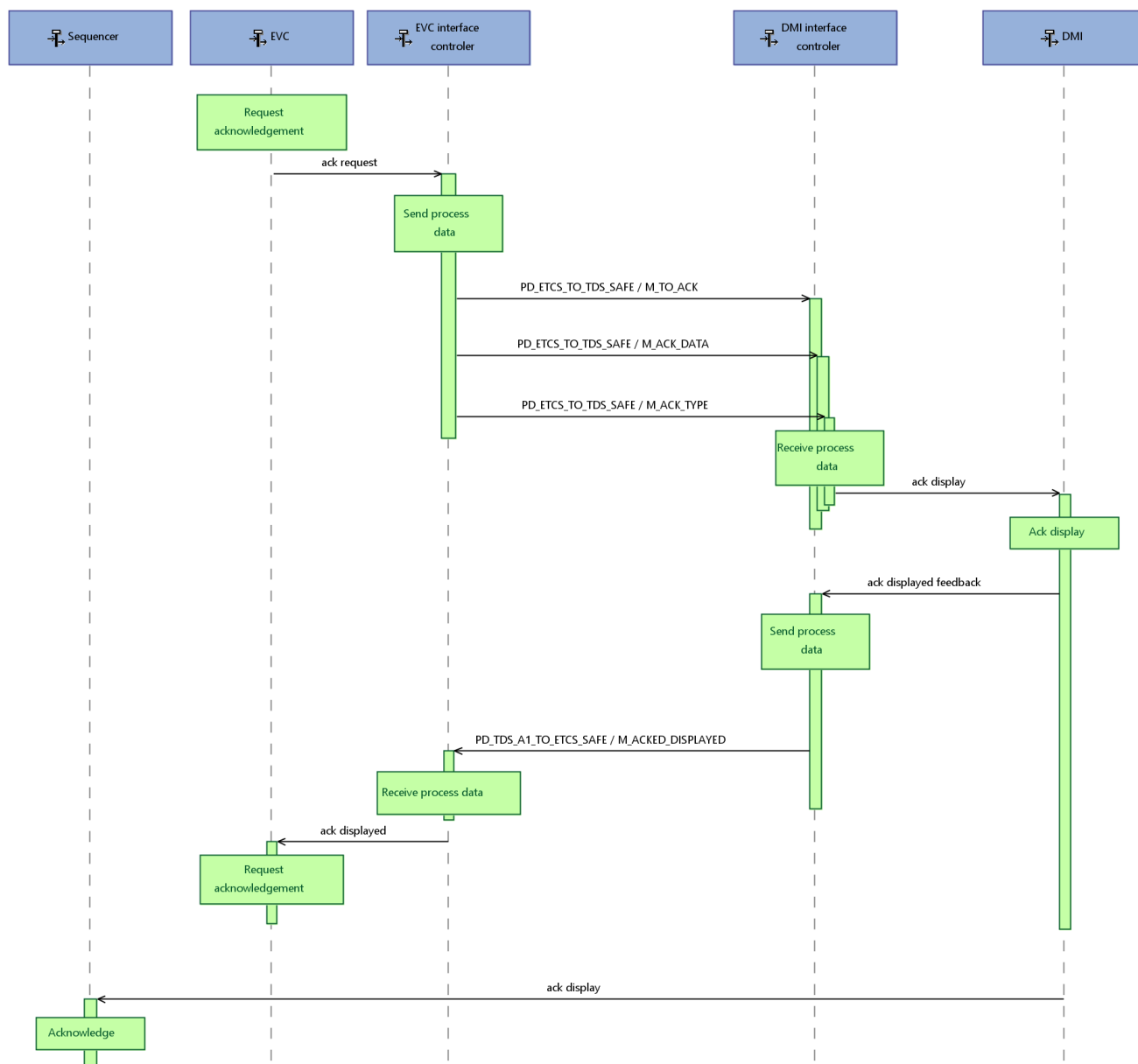


Figure 23 – Unit test – Connection request

6.2.1.2 Acknowledgement

EVC requests the display for the acknowledgement, Driver acknowledges, the TDS informs the EVC.



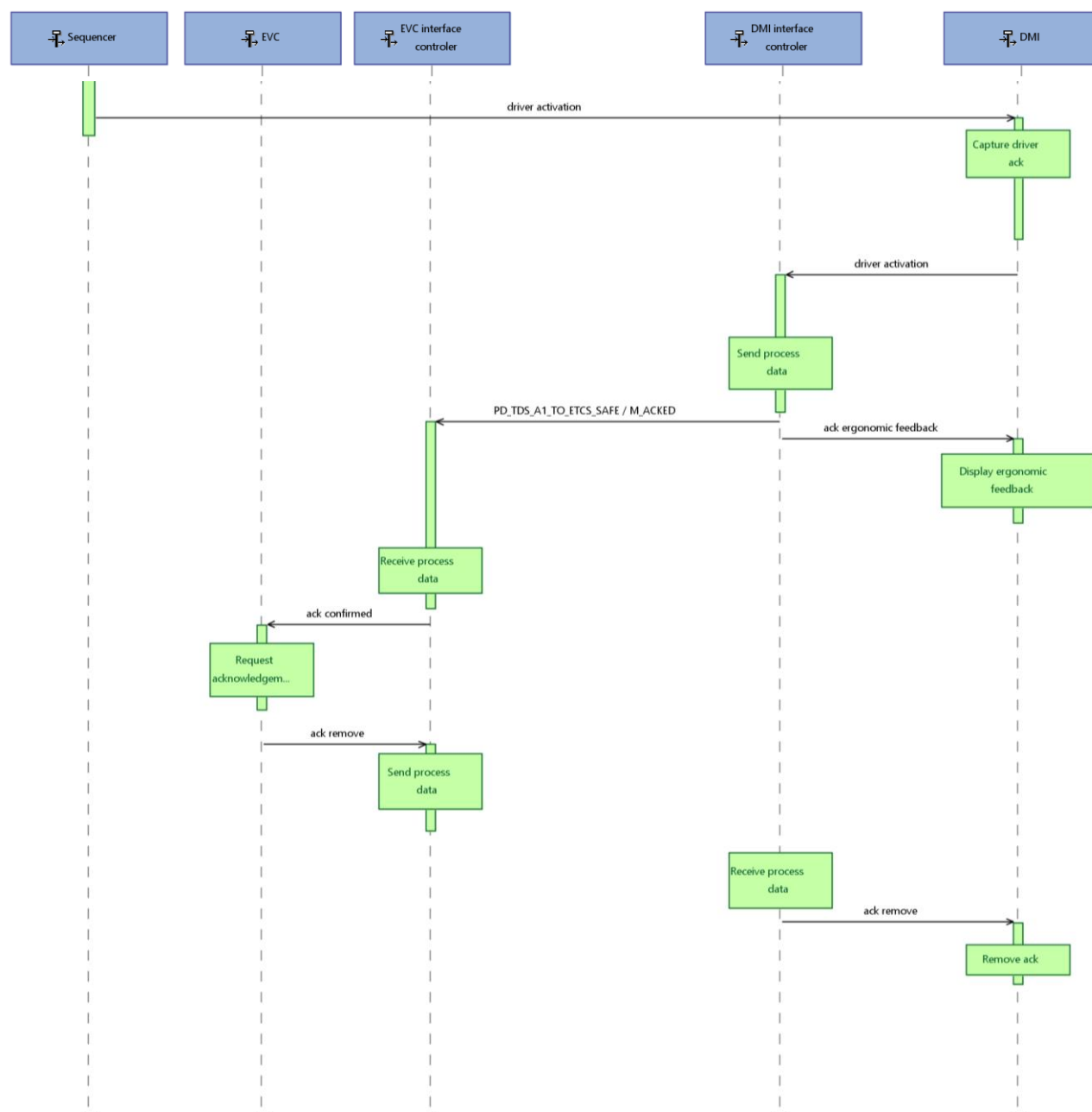


Figure 21 – Unit test - acknowledgement

6.2.2 Operational test

6.2.2.1 Start of mission

When power up, the connection between EVC and TDS is initiated. As next the EVC requests the driver ID.

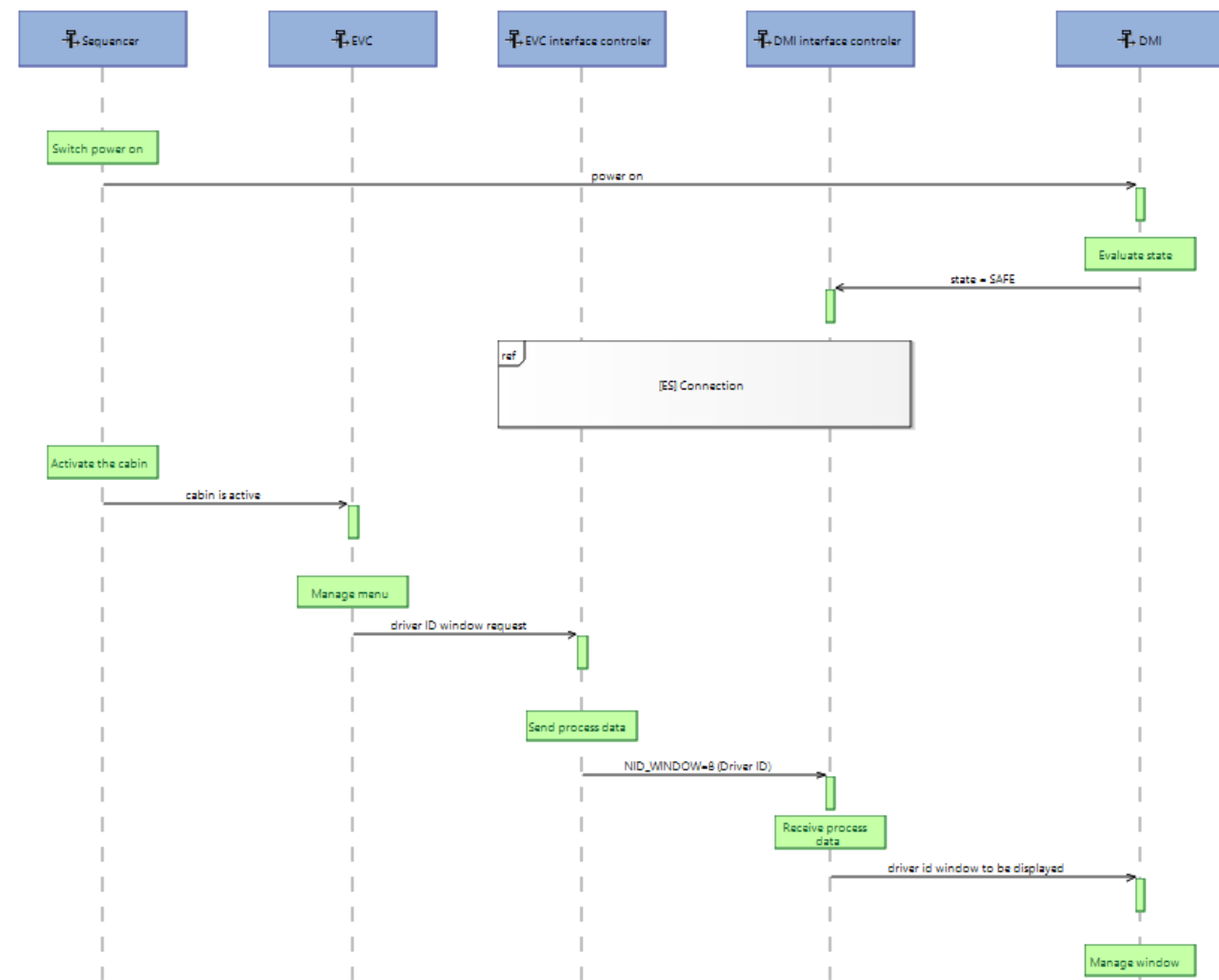
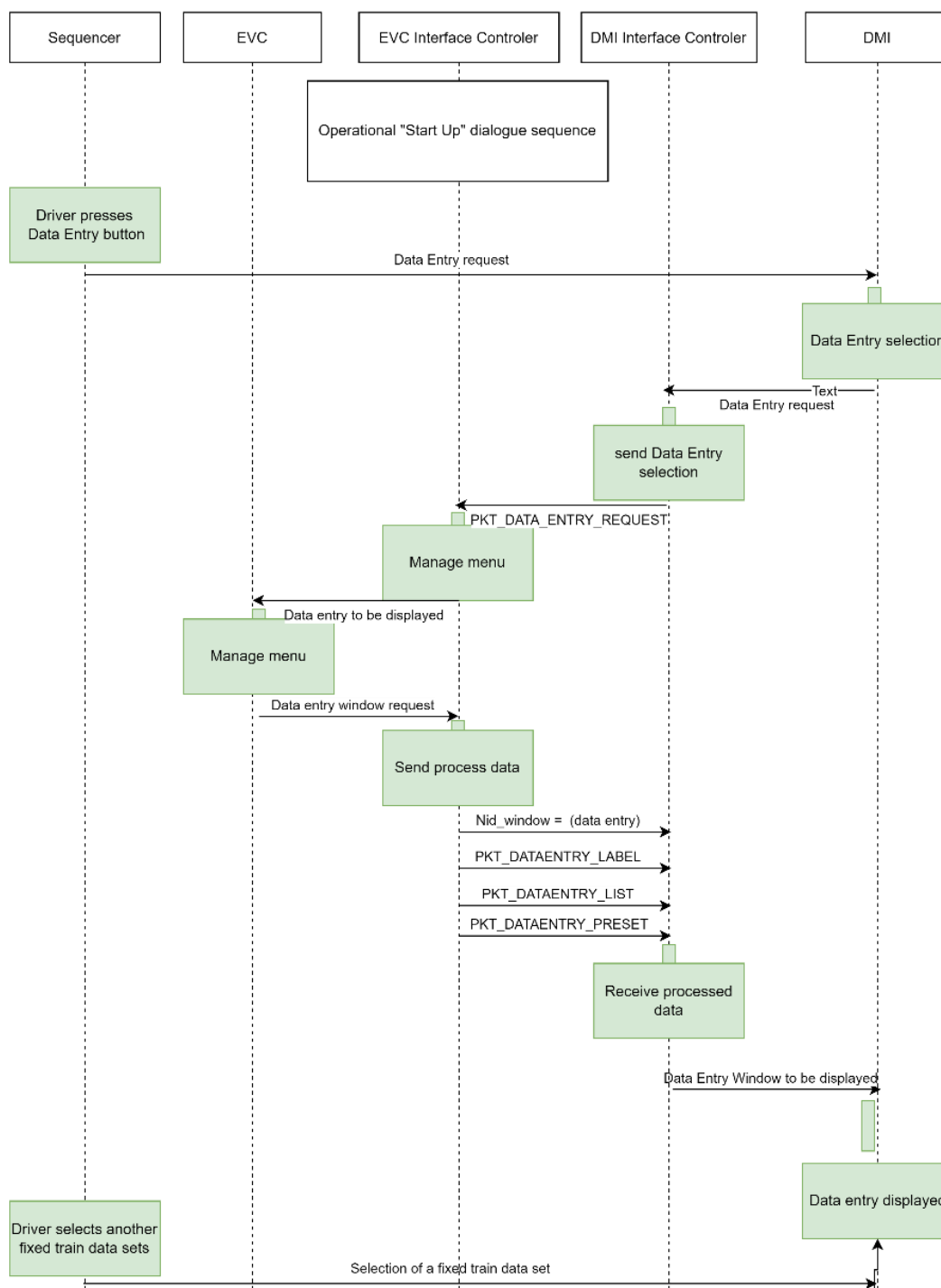
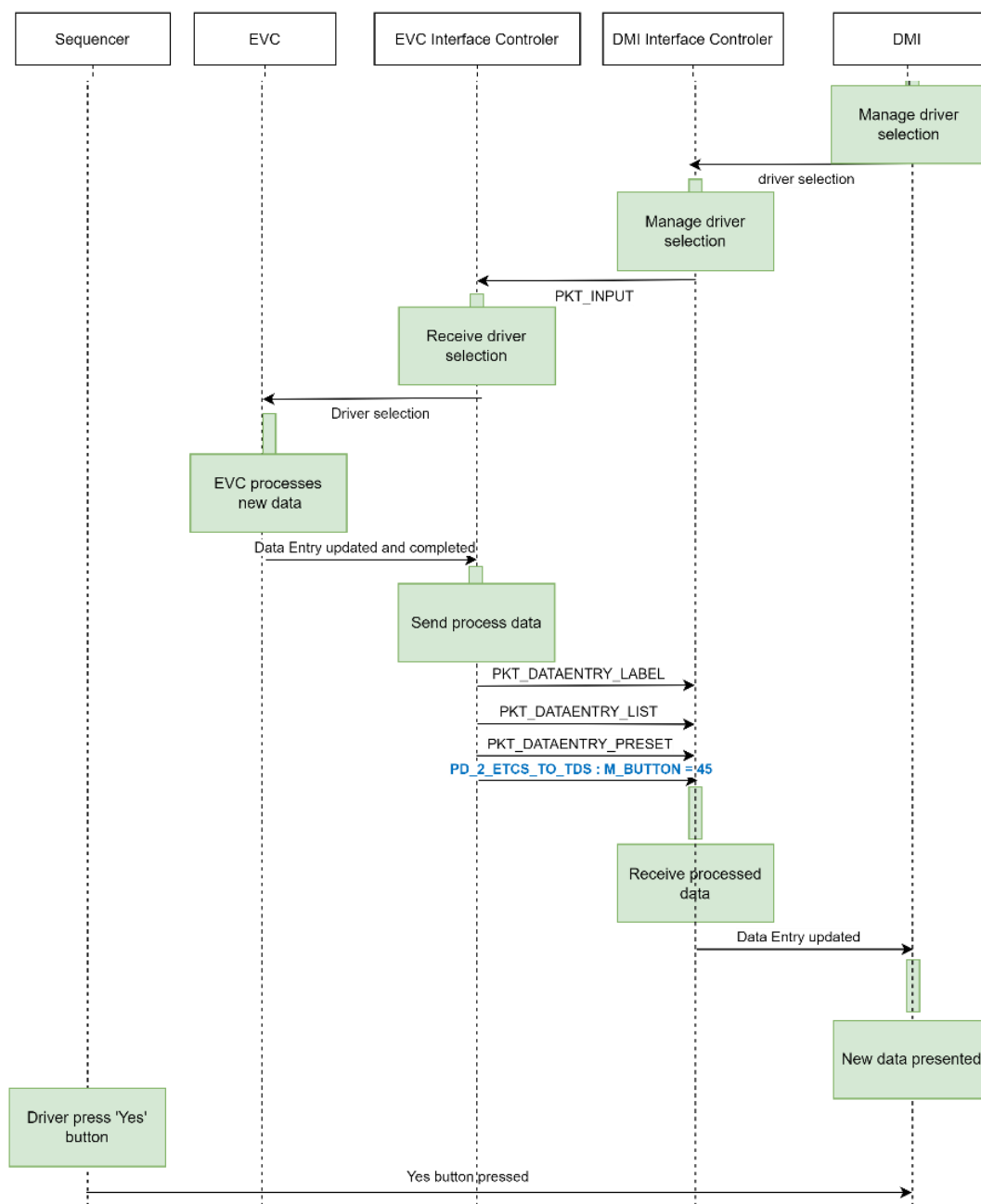


Figure 22 – Operational - Start of mission sequence

6.2.2.2 Start of mission – data entry

During start of mission, after ID and TRN are configured, Driver configures the train data: train length, brake characteristics, ...





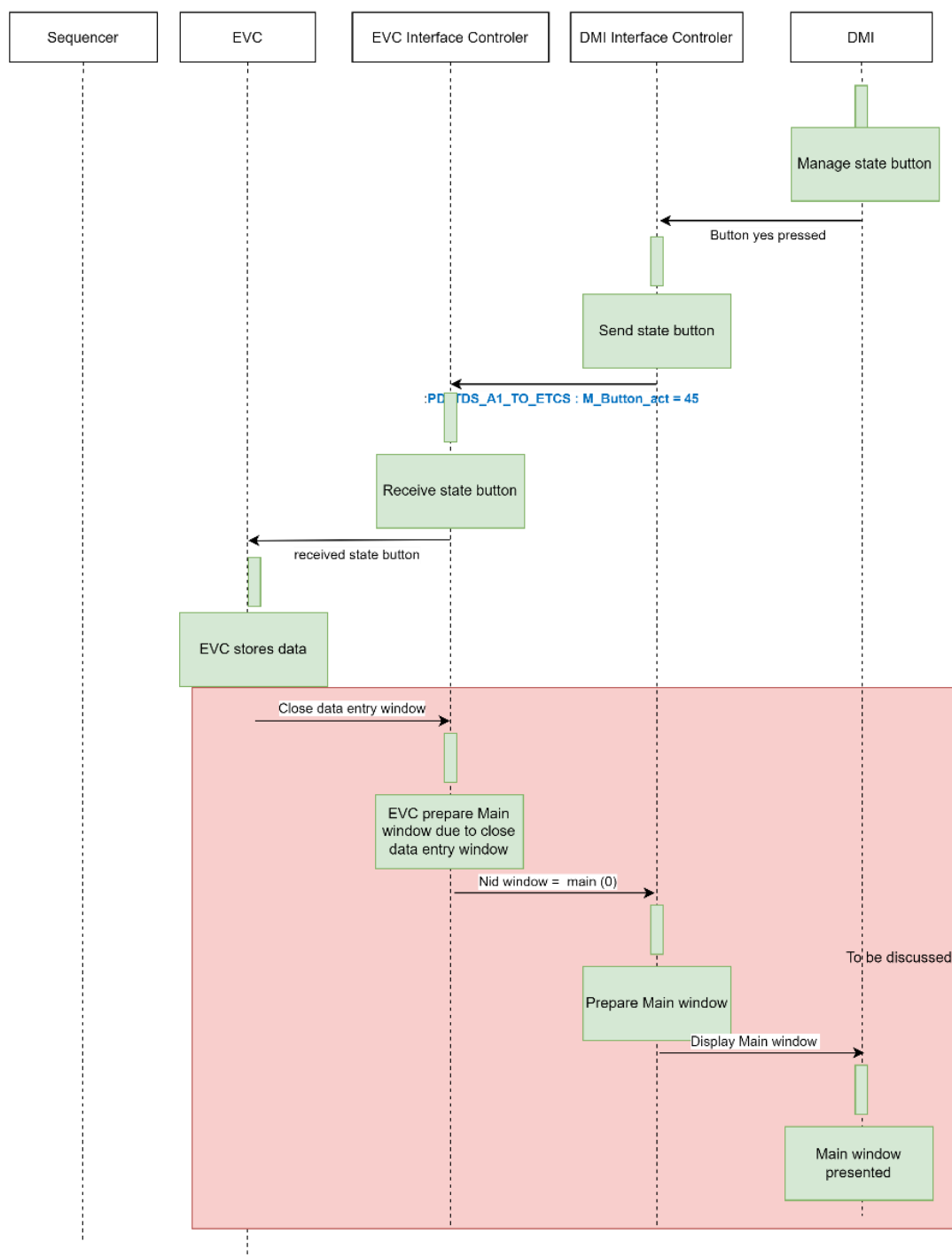


Figure 23 – Operational - Start of mission data entry

6.2.2.3 Mission in level 2 FS

CCS-OB enters into a braking to target sequence. A static speed profile includes a discontinuity with a decrease of speed. The CCS-OB crosses the successive supervision limit.

Precondition: CCS-OB is in Full Supervision Mode. A Movement Authority (MA) is received and used. Position of the train is before starting the indication of supervision limits.

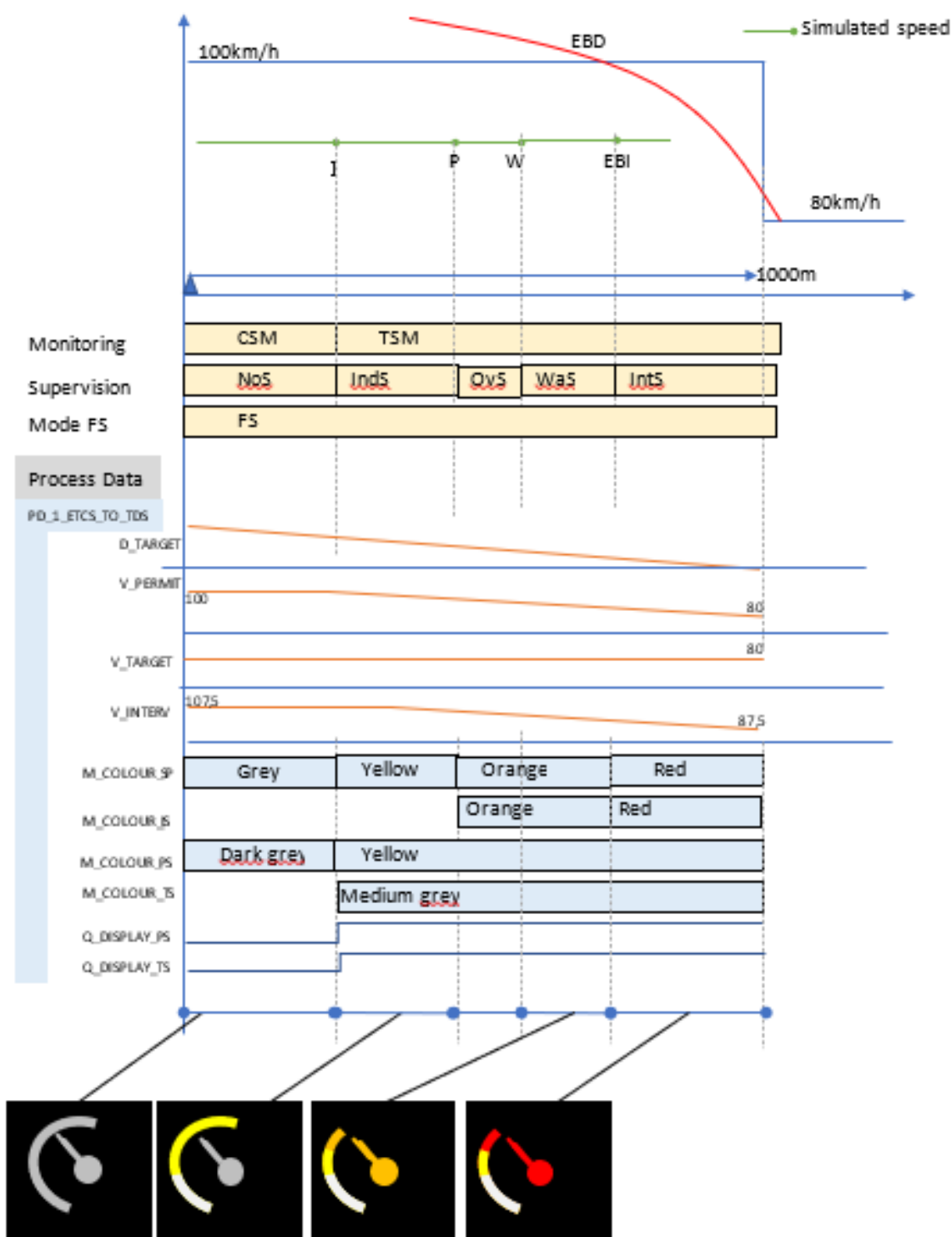


Figure 24 – mission in level 2 display

7 Model description

This chapter describes the different models that implement the Subset 121 and SDTv2 protocol.

7.1 Model hierarchy

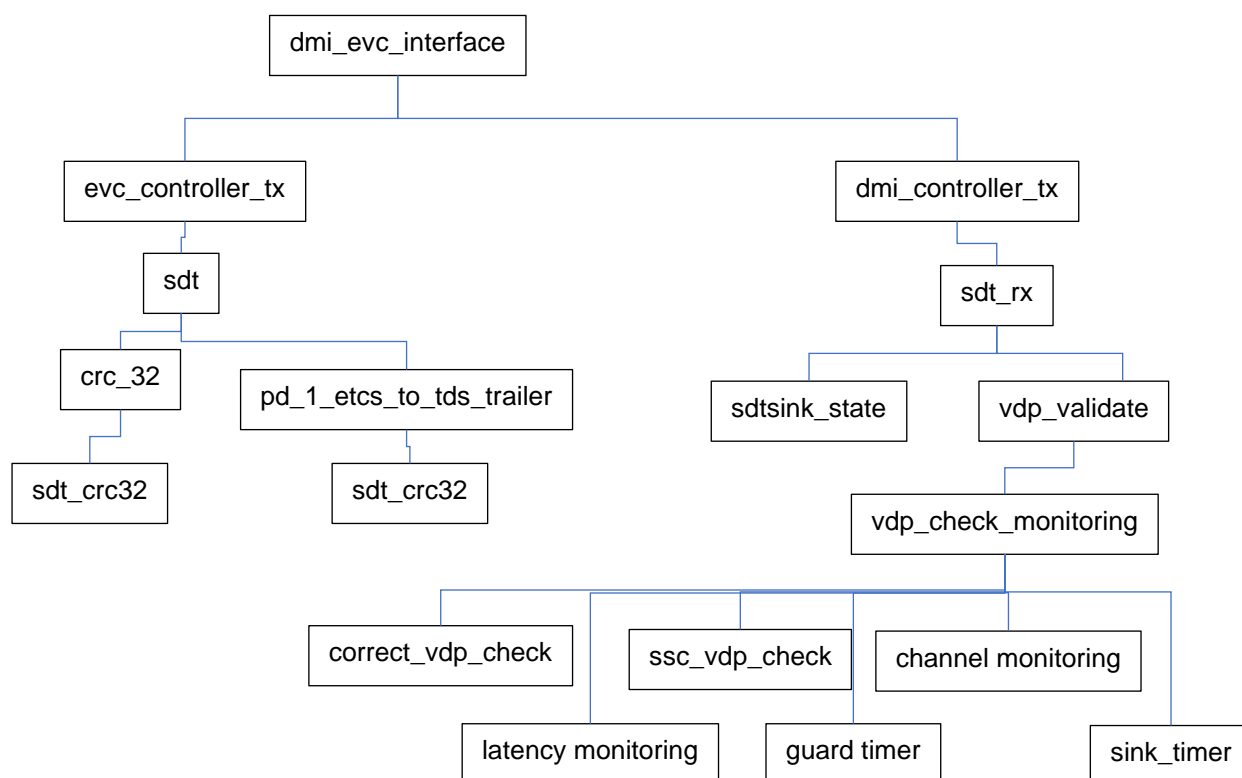


Figure 25 – model hierarchy

7.2 Transmission models

7.2.1 CRC 32 bits calculation

Name sdt_crc32
Description Calculation of the CRC of the SID used to compute the source identifier and the safety code
Reference NF EN 61375-2-3 §B7 SC-32
Input buf:SID crc:length of the CRC CRC_TAB_32: NF EN 61375-2-3 §B7 SC-32 – figure B.4
Output out_crc: calculated crc
Model function out_crc = crc32(buf, crc, CRC_TAB_32)

```

% Input reflected (not present in the IEC 61375-2-3)
buf = bin2dec(flip1r(dec2bin(buf, 8)));

for byte = 1:length(buf)
    idx = bitand(bitxor(bitshift(crc, -24), uint32(buf(byte))), uint32(0xFF)) + 1;
    crc = bitxor(CRC_TAB_32(idx), bitshift(crc, 8));
end

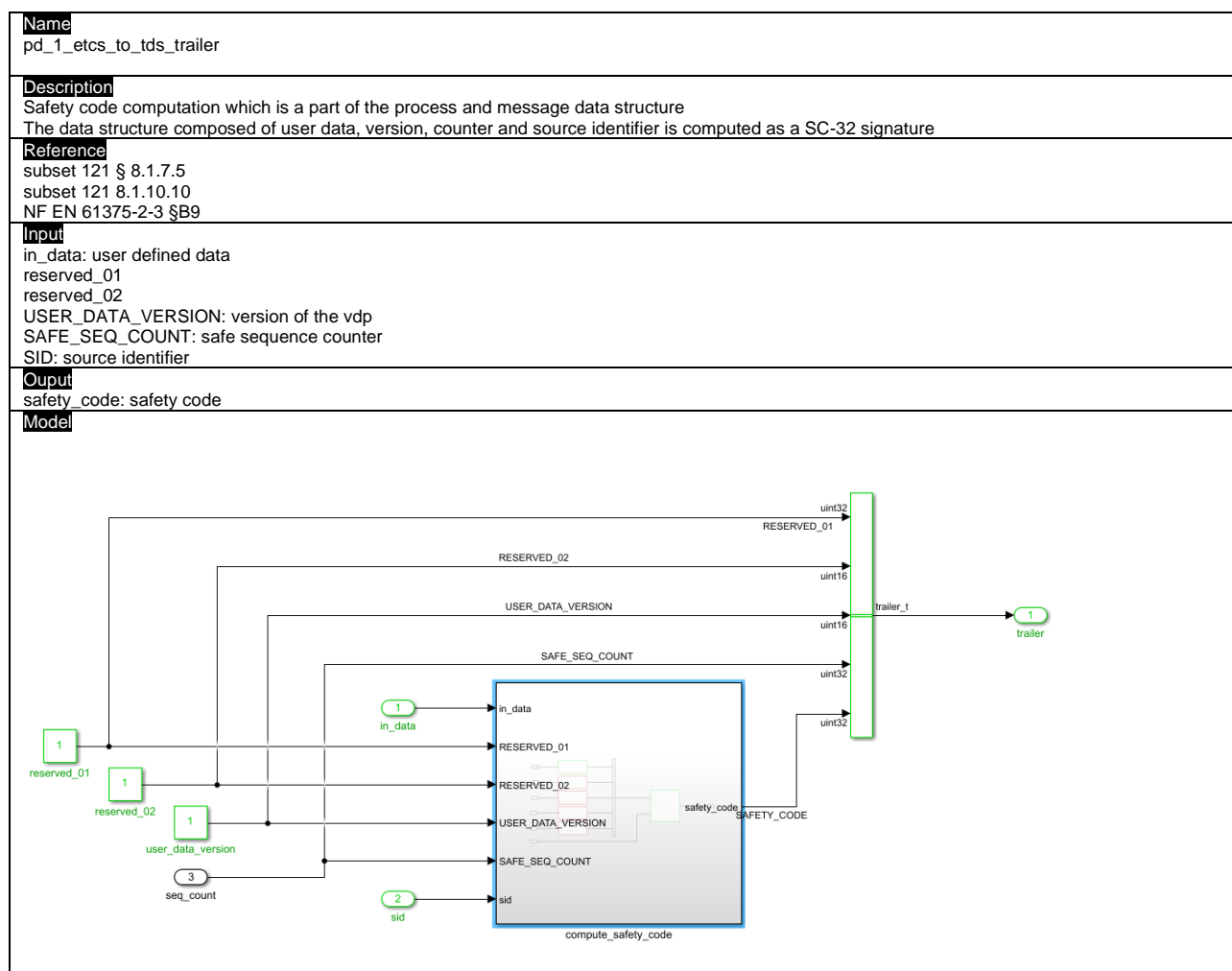
% Result reflected (not present in the IEC 61375-2-3)
crc = bin2dec(flip1r(dec2bin(crc, 32)));

% Final XOR value (not present in the IEC 61375-2-3)
out_crc = bitxor(crc, uint32(0xFFFFFFFF));

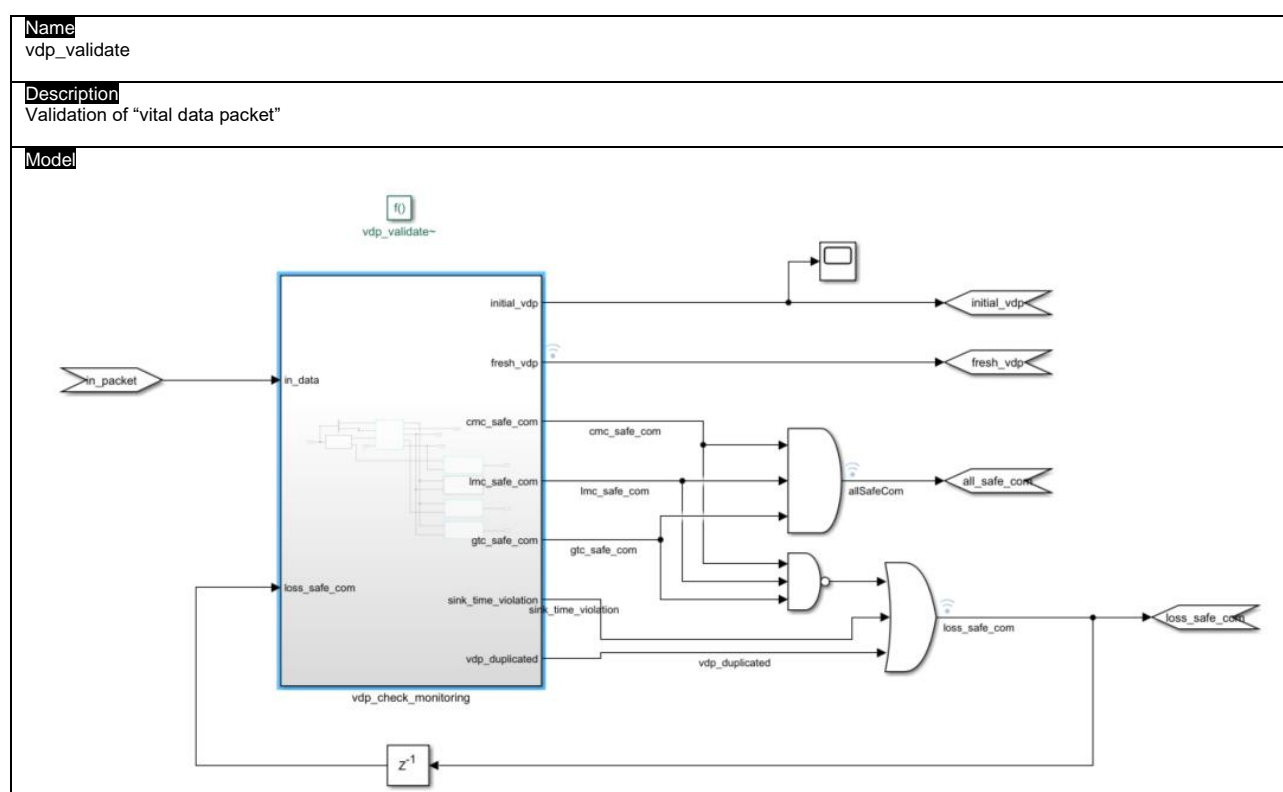
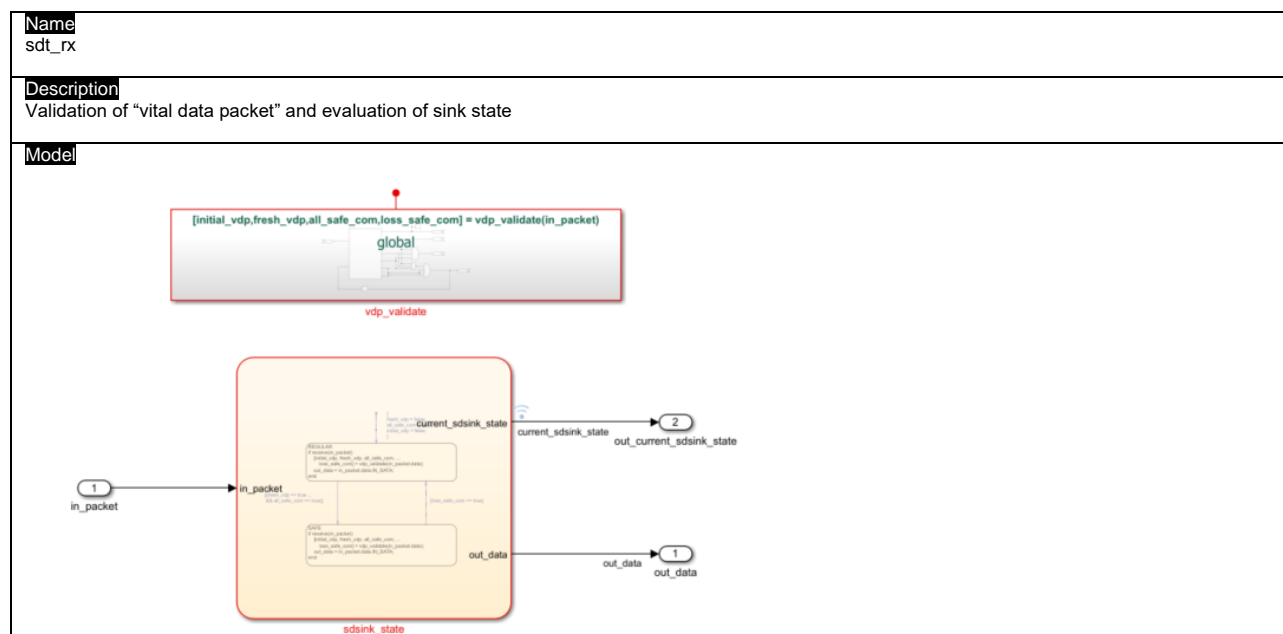
end

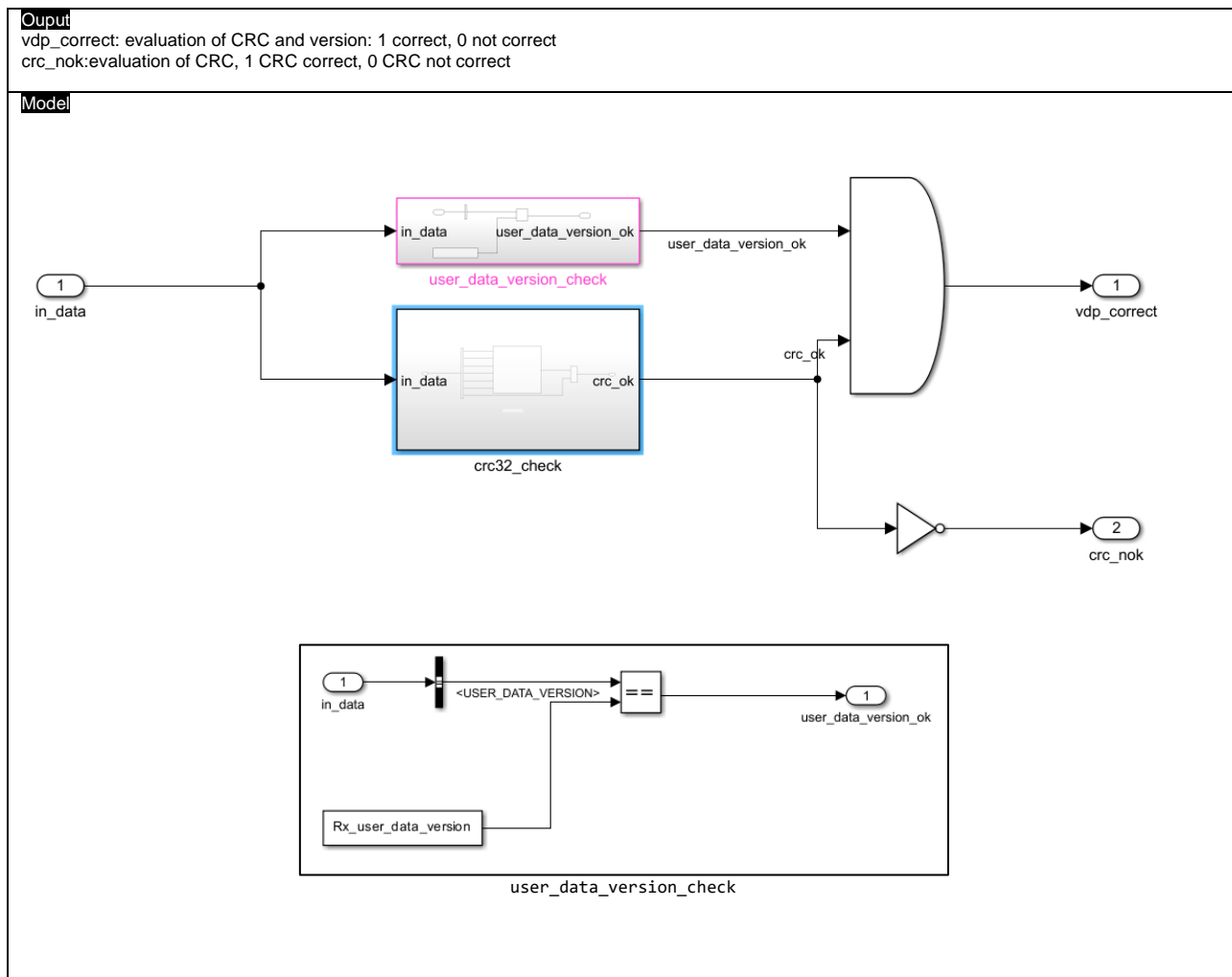
```

7.2.2 Safety code

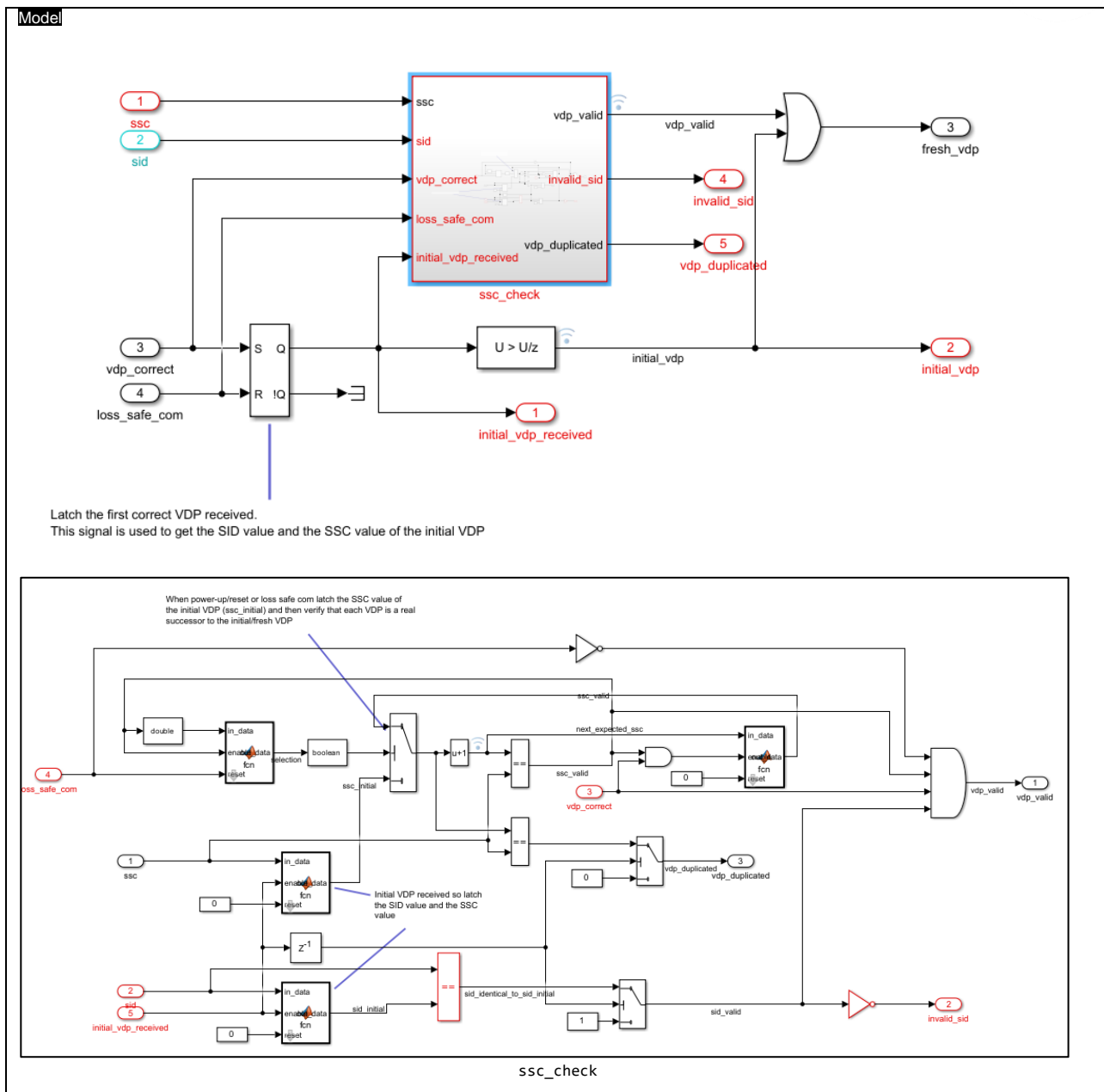


7.3 Reception models

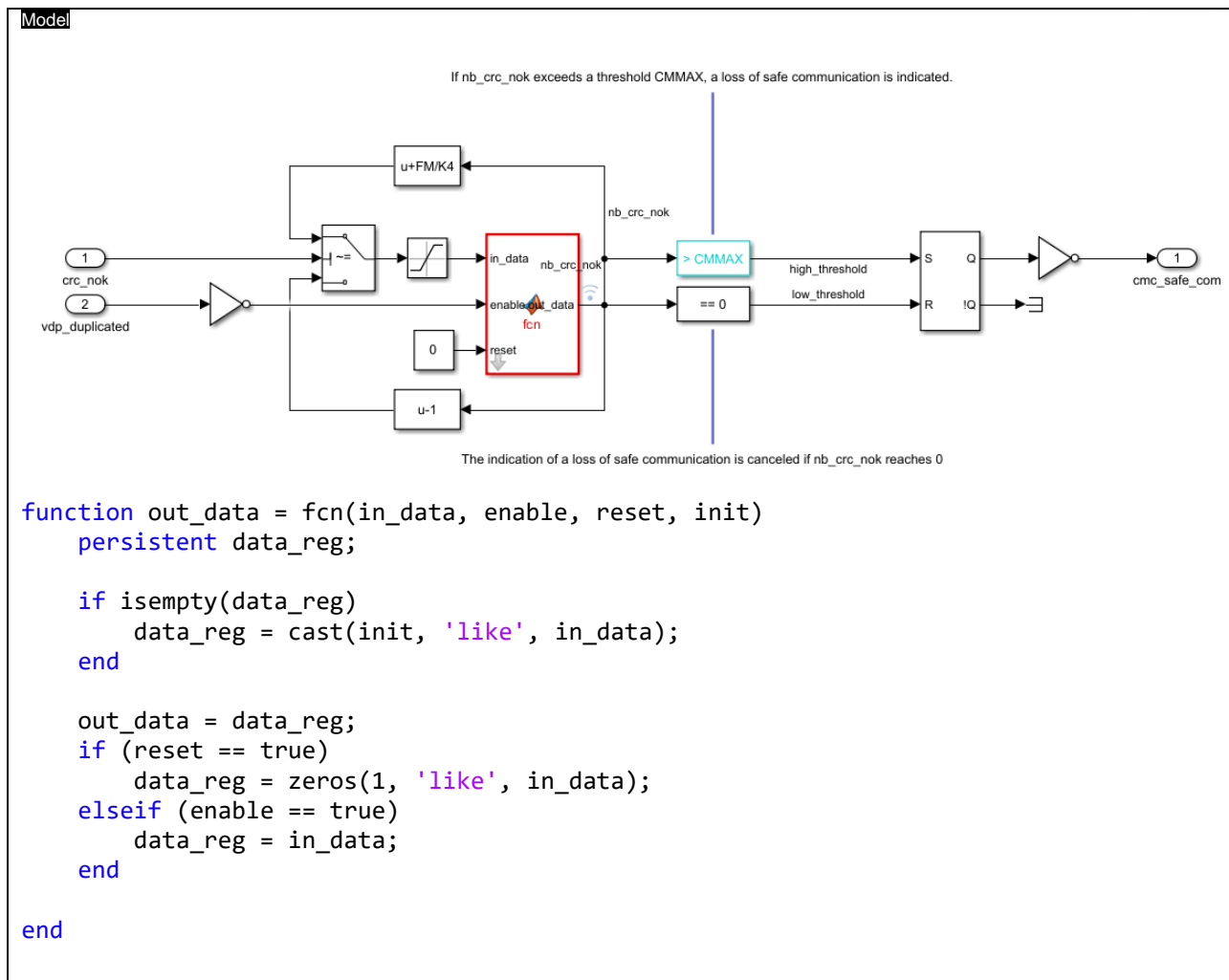




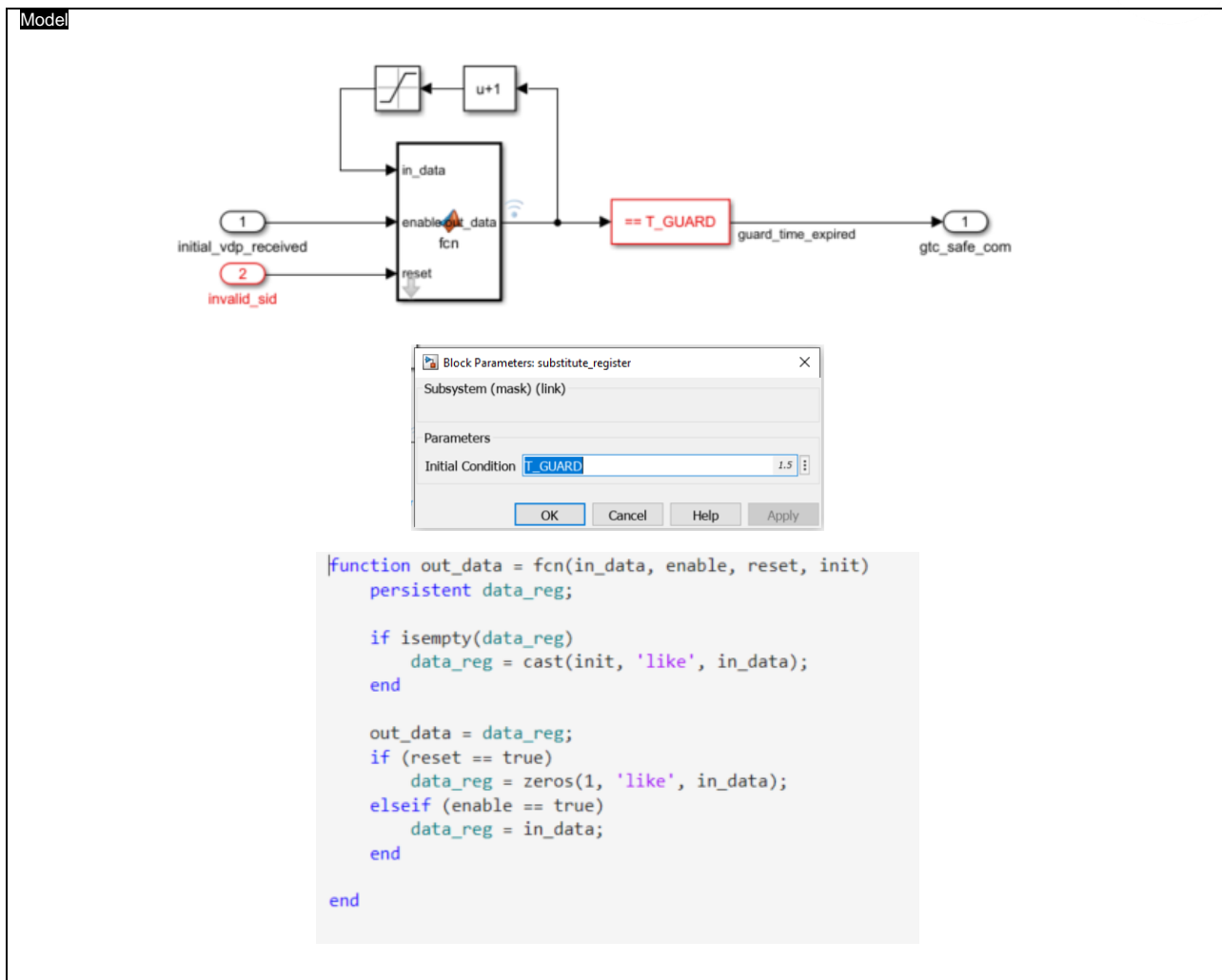
Name	ssc_vdp_check
Description	Evaluate the freshness of the received vdp
Reference	EN 61375-2-3 §B13.2.2.3 EN 61375-2-3 §B13.2.2.4 EN 61375-2-3 §B13.2.2.5
Input	ssc: safe counter sid: source identifier vdp_correct: correctness of vdp according to version and checksum loss_safe_com: loss of communication
Output	fresh_vdp: is vdp evaluate as fresh



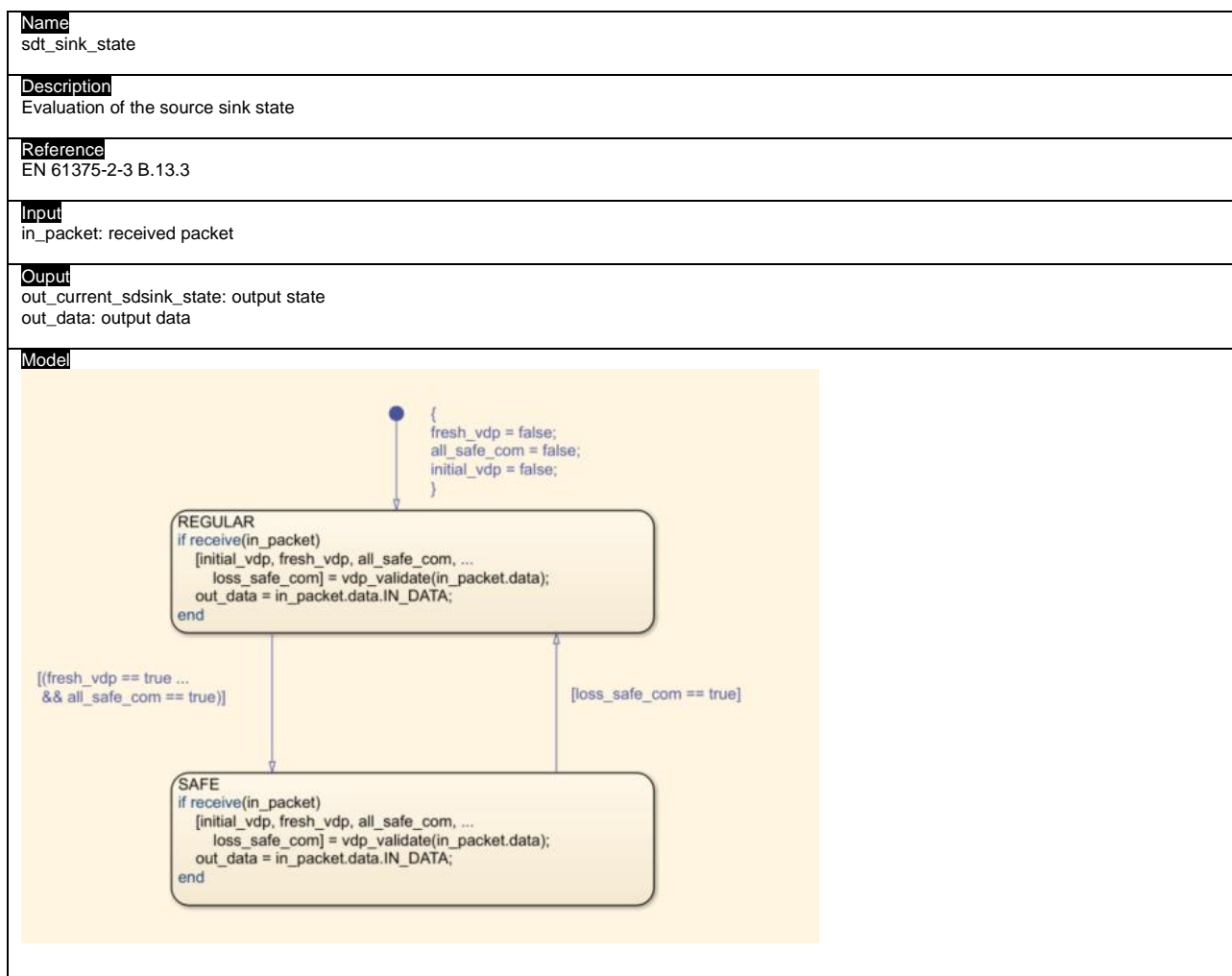
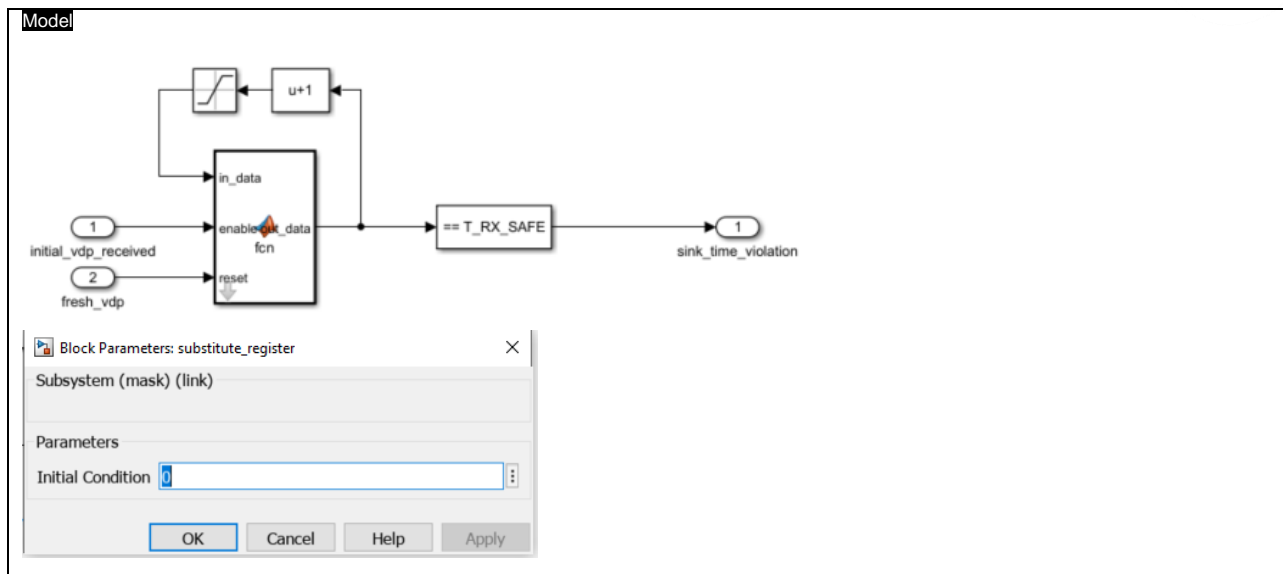
Name	channel monitoring
Description	detection of sudden increase of the transmission failure rate
Reference	NF EN 61375-2-3 §B13.9
Input	crc_no vdp_duplicated
Output	cmc_safe_com



Name	guard_timer
Description	The guard time check intends to detect two redundant active sending sources
Reference	EN 61375 §B.13.7.1
Input	initial_vdp_received ; initial vdp received that trigger the guard timer invalid_sid: new SID detected different from the previous one
Output	gtc_safe_com: guard time is expired




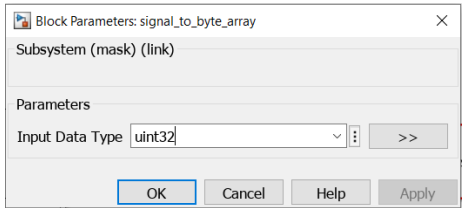
Name	sink_timer
Description	Timer expiration since the reception of the first fresh vdp
Reference	EN 61375-2-3 §B.13.6
Input	initial_vdp: vdp received which trigger the timer fresh_vdp: reception of a new fresh vdp
Output	sink_time_violation: expiration of the sink time



7.4 Generic Block

For specific needs generic blocks have been implemented.

7.4.1 Signal to byte array

Description	Convert a signal to a array of byte
Reference	n.a
Input	<p>in_data: input data</p> <p>the data type of the input is defined through a block mask</p>
Output	bytes_array: array of bytes
Model	<div data-bbox="359 577 598 712">  </div> <div data-bbox="810 524 1273 730">  </div> <pre> function bytes_array = signal_to_byte_array(in_data, in_data_size) fields_array = zeros(in_data_size, 1, "uint8"); fields_array(:) = typecast(in_data, "uint8"); bytes_array = fields_array; end </pre>

8 Analysis of Subset 121

Modelling activities enables to identify enhancements for Subset 121.

Subset 121 version 1.1.0 is the referenced version for this analysis.

These remarks will be transmitted to ERJU Train CS domain and ERA with a formal review sheet of OCORA.

We categorised the remarks as follow:

- Form: proposal to bring clarity to the document
- Technical: evolution proposal
- Open Point: point requiring clarification

1. Technical: §3.1 Reference: versions of the document are missing
2. Technical: Figure 1: ETCS reference Architecture: Why is there a link between Driver and ETCS onboard DMI function? I think it could be more suitable to add ERA_ERTMS_015560 to the TDS to DMI function interface.
3. Technical: §8.1.8.4/6.9.4.5 M_COLOUR_xx: is it relevant that the ETCS-OB computes the colour? The TDS can compute the colour based on mode and supervision status.
4. Technical: §6.11.4.1 M_COLOUR_PA_IND: the indication marker is yellow. Why there is the need to have a dedicated parameter?
5. Technical: What is the motivation to make the distinction between digit (X_V_TRAIN_DIG100, X_V_TRAIN_DIG10, X_V_TRAIN_DIG1) and train speed variable (V_TRAIN)? The TDS can compute the digit based on train speed value.
6. Technical: Functional repartition between ETCS on board “DMI function” and TDS. For now the controller function which computes information that needs to be displayed are mostly in the ETCS OB side (Archi 1). But it shall be considered also to allocate this function in the TDS side (Archi 2). See the figure and the table below for the difference and pro and con of these two different architectures.

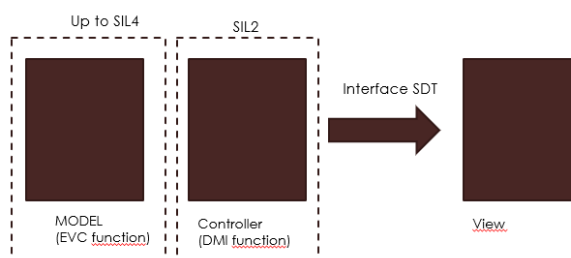


Figure 24 – display architecture 1

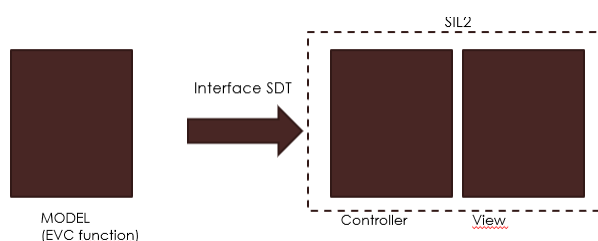


Figure 25 – display architecture 2

Architecture	Pros
Display Architecture 1	<ul style="list-style-type: none"> + state machine in ETCS-OB side only. No synchronization issue of state machine in both side =more simple + TDS side is more generic

Display Architecture 2	<ul style="list-style-type: none"> + all SIL2 display functionality are grouped in one logical component. A modification don't affect ETCS OB + separation of display logic from supervision logic + all graphic functions are in the TDS side + less latency constraint in the interface + integrated management of display elements
------------------------	--

Table 4 - TDS architecture comparison

7. Technical: Functional definition of the logical component (ETCS OB DMI function, TDS) are spread in different document Subset 026, ERA DMI and Subset 121. A principle shall be defined in order to clarify the functional allocation.

- a. ETCS on board DMI function: in subset 026
- b. TDS function: in ERA DMI
- c. Interface between ETCS OB and TDS: Subset 121

In fact, ERA DMI includes ETCS OB and TDS functional requirements. Subset 121 includes TDS function (mode and safety reaction).

This can also be performed by using a requirement-based approach supported by tool (Polarion for e.g.) which allow to allocated properties to requirement (like allocation to logical component). Export from the tool allow to conserve word format (same as today) but in addition ReqIf format can also be produced which provide additional information linked to the requirement (versions, historic, allocation to logical component, links with others requirements, ...)

Another way of doing this, is to provide a global architectural view which provide a synthesis of functional allocation. That can be also supported by a tool (capella for e.g.)

8. Technical: §5.2.7.5 What are the consequence of “abort the data entry procedure”? back to “main window”?
9. Open point: §7.6.1.2 Juridical information. How is this an exception? You mean that there is an interpretation of the ETCS-OB if TDS provide a speed with invalid status? There is also a conversion between M_BUTTONS (Subset 121) and M_DRIVER_ACTIONS (Subset 027) for e.g.
10. Open point: DMI_Conf , fixed data, M_BUTTON: why do we need to send brightness to the ETCS OB?
11. Open point: do we consider that sound generator is a part of the TDS? or can it be external? linked to the ETCS OB or to the TDS?
12. Technical: TDS_conf: is it relevant to define the parameter in a excel file? It would be more relevant to do it in a metalanguage (xml for example). So, it can directly be used to configure the TDS. Avoid transposition and risk of error. This file has to be certified for the use in SIL2 application (functions) and make it available to suppliers which can reuse it directly as it is.
13. Technical: Use standardized modelling language to define structure of data (especially message and packet) (cf ANNEX 11 - 12)
14. Technical: §8.1.10.8.1 “8.1.10.8.1 The SMIs (see TCN-TCP Annex B.8) and the UserDataMainVersion (see TCN-TCP Annex B.15) shall be configured in EVC-TDS_Conf. Each SMI shall be unique in the network.”
- a. UserDataMainVersion not find in TDS_conf
 - b. The SMI is used for distinction between process and message data. (deviation from standard IEC 61375-2-3 identification of device)
15. Technical: IEC 61375-2-3: 2015 §ANNEXE B.6 p.265 Variable length of data structures (open arrays, records) shall not be used. SDTv2 is used to justify integrity of the interface, but by not respecting the

principles it is questionable if this works.

16. Open point: Subset 121, §222.1.1.20: How to configure the PKT_INPUT with fixed train data set? Shall we transmit a defined value for L_X_VALUE or attribute the value 0 and create a specific NID_KEY (spare values defined in configuration file)?
17. Technical: Related to the configuration file of the Subset 121, it seems that the process data structure is only provided as an example. It means that each project can have a different structure for the process data. As an interface specification, the subset should define the structure of the process data. So, software of TDS and EVC are directly compatible with no additional development.
18. Technical: M_ACK_DISPLAYED to be defined in §8.1.11
19. Open point: Process data “remove ack” not found
20. Open point: It is not clear how the TDS returns to the main window after the data entry. There is a need for clarification in the sequence. Which command is used and in which order “close”, “display” “nid_window”.
21. Form: Describe the relation between messages in the sequence diagram and the packet/process variables

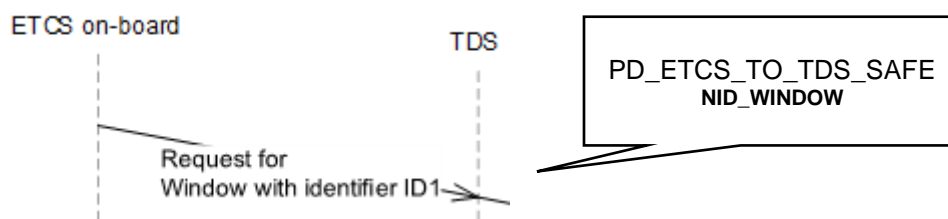


Figure 26 - subset 121 fig 6 proposed improvement

22. Open point: According to the sequence of the process data, the order of the PD and how message data is inserted is not described. Cycle time of PD is 128ms.
 Does it means that all PD and MD has to be send in this period ?
 What in case there are several MDs to be sent?
 Is there a queuing mechanism defined?
 Is there a priority mechanism involved?
 Does the order of the PD has to be defined ?
 What are the criteria for MD insertion between PD ?
 What is the time between each PD ?

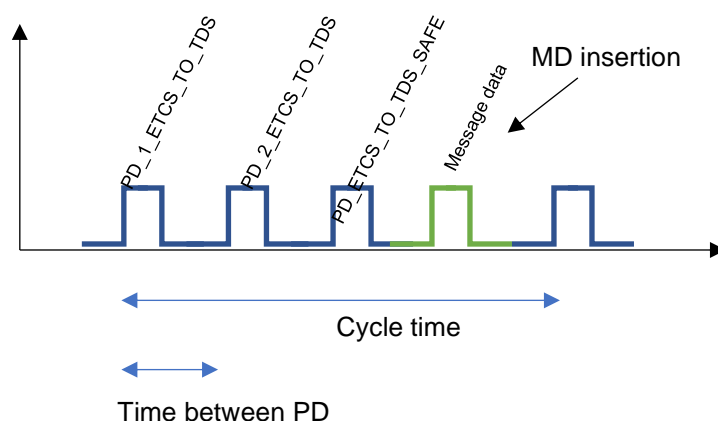


Figure 27 - sequence of PD and MD

23. Open point: What is the meaning of “- 3 bytes” ?

Packet (non safe)		
NID_PACKET [0..127]	L_PACKET	Data
1 byte	2 bytes	L_PACKET - 3 bytes

8.1.10.6 Safety related data shall be transmitted in safe packets containing a safety trailer.

Figure 28 - extract from subset 121 packet structure (non safe)

8.1.10.10 Packet structure used for safety related data

Packet (safe)			
NID_PACKET [128..255]	L_PACKET	Data	Safety Trailer according to [TCN-TCP] Annex B.9 (ETH) Annex B.15 (MVB)
1 byte	2 bytes	L_PACKET - 3 bytes	6 bytes on MVB 16 bytes on ETH

Figure 29 - extract from subset 121 packet structure (safe)

24. Open point: There is no size limit (maximum) of message data.
25. Technical: Generic default maximum value of N_ITER is 255 for packets. Is it possible to define a maximum value for each N_ITER in the packets?
26. Form: Use octet rather than byte. Octet is not only European but international and is less likely to be confused. Especially when using bit, kb => kilobit, KB => KiloBytes => Ko Kilo-octet
27. Technical: The definition of the display of one specific window is split between different sequence diagrams spread throughout the document. Provide a synthesis diagram for specific windows which allows understanding the exchange based on all the needed elements to build a window (driver ID for example).
28. Technical: Figure 9 in Subset 121 – add the case where request for data entry is automatic (driver ID when switching to SB mode from powerup)
29. Technical: Figure 9 in Subset 121 – add the case where data is acquired by an external source

(driver ID and train running number)

30. Technical: The sequence diagram requires to display the window on receipt of the window identifier. But the display shall only display the window when all the necessary information has been received beforehand (window, label, buttons, ...). So, the display is only shown when all PD containing the data are received.

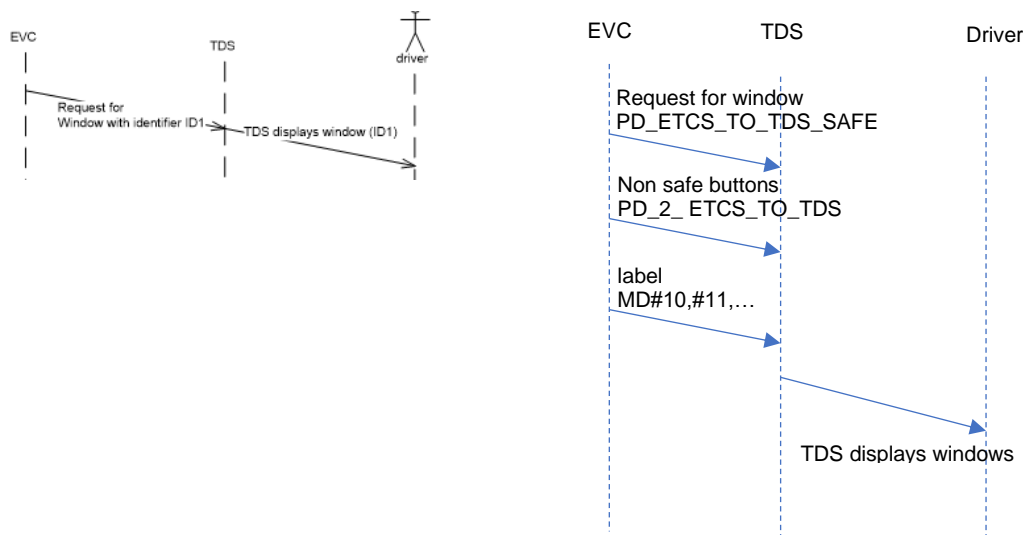


Figure 30 - clarification of the display window sequence

31. Form: exception in an exception – list the button controlled by EVC and the ones controlled by the TDS separately

6.3.1.2 **Exception:** The Navigation buttons including the 'Enter' button, but without the 'Close' button are controlled by the TDS. Refer to ERA-DMI § 5.3.2.7.

Figure 31 - subset 121 § 6.3.1.2

32. Open point: Is the keyboard controlled by TDS or by EVC
33. Technical: Keyboard identifier in DMI_conf by "informal"
34. Technical: § 3.1 add the version of the referenced document
35. Technical: §8.1.1.15 add key names for soft key screen

1-A
2-B1
3-B2
4-C2
5-C3
6-C4
7-D2
8-D3
9-D4

Figure 32 - soft key screen key names

36. Technical: Subset121 implies a functional allocation between EVC and TDS. Provide identification of

functional allocation based on ERA_DMI and subset 026 documents.

10.3.4.2.1 The check versus the technical permitted range shall take place when an entered data value is accepted by the driver.

Requirement for EVC

10.3.4.2.2 When the entered data value of an input field is outside its pre-configured technical permitted range, the data part of its echo text shall indicate '++++' in red.

Requirement for TDS

Figure 33 - extract from Subset 121 §10.3.4.2

9 Conclusion

The activities enable the definition and the implementation of a model of the communication interface as defined by Subset 121. We can evaluate the progress of work and the planning for the OCORA R6 release as follow:

Activities	% progress OCORA R5	planned OCORA R6
Definition of the model	80%	100%
Definition of the test sequence	40%	60%
Implementation of the model	60%	100%
Simulation	10%	50%
Evaluation of alternative architecture	0%	0%
Implementation of management of several train display	0%	0%

For the model implementation:

Part fo the model	% progress OCORA R5	planned OCORA R6
Sequencer	70%	100%
EVC model	60%	80%
TDS model	40%	80%
Output of model	40%	80%
Interface	80%	100%

For the test:

Type of test	% progress OCORA R5	planned OCORA R6
Unit test	60%	100%
Simulation – nominal	50%	100%
Simulation – degraded mode	10%	50%

Even if the tests are not carried out in an exhaustive manner, the implementation activity has raised points relating to the specification Subset 121.

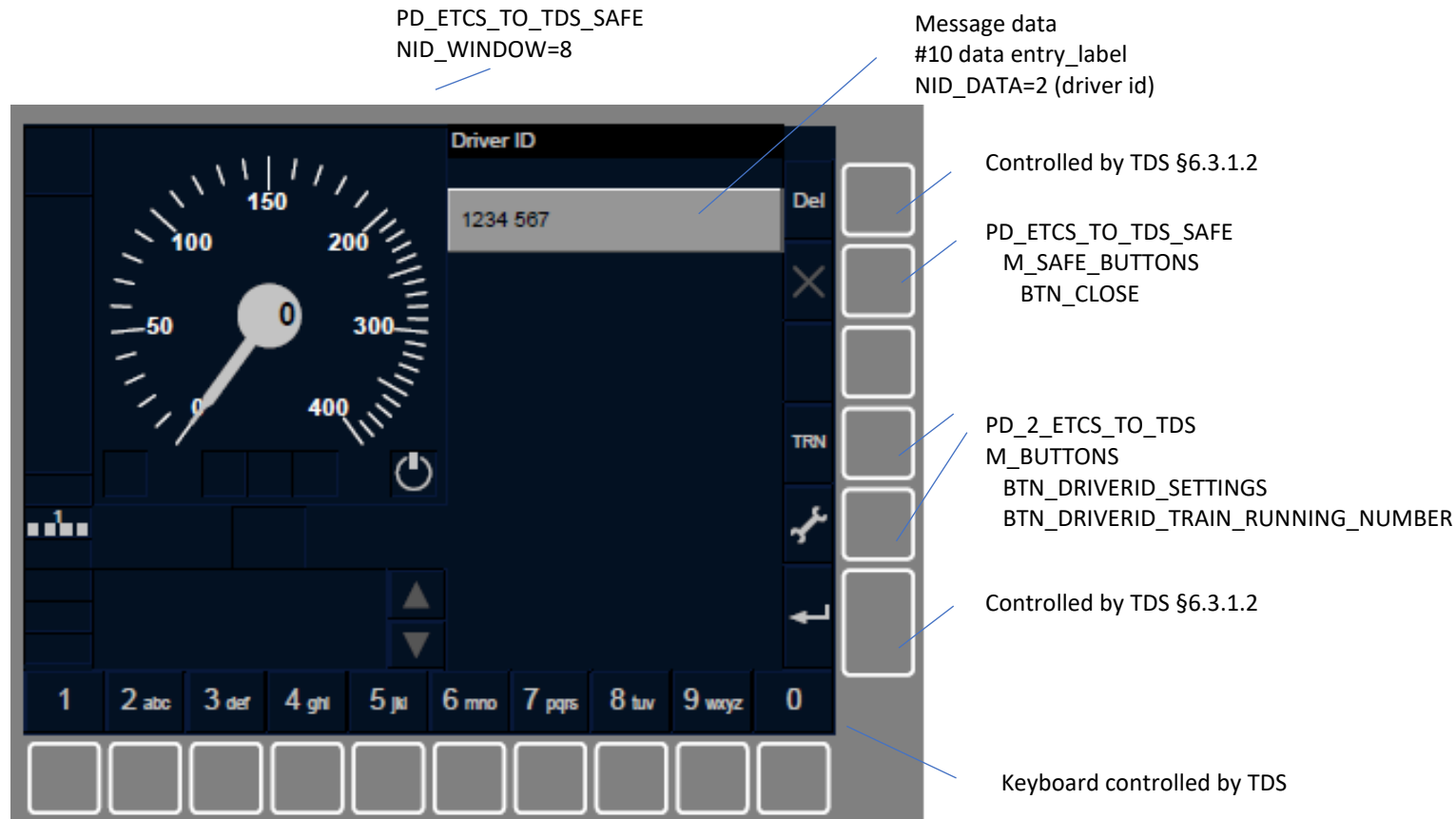
10 ANNEX – PACKETS

Source	N°	Title	Direction	Function	Safe
TDS-	1	PKT_CONNECTION_REQUEST	EVC -> TDS	TDS: Connection EVC/TDS	
	3	PKT_DISCONNECTION_REQUEST	EVC -> TDS	TDS: Connection EVC/TDS	
	10	PKT_DATAENTRY_LABEL	EVC -> TDS	TDS: Data entry	
	11	PKT_DATAENTRY_LIST	EVC -> TDS	TDS: Data entry	
	128	PKT_DATAENTRY_PRESET	EVC -> TDS	TDS: Data entry	X
	129	PKT_ECHO_TEXT	EVC -> TDS	TDS: Data entry	X
	14	PKT_DATAVIEW	EVC -> TDS	TDS: Data view	
	15	PKT_SPECIFIC_DATAVIEW	EVC -> TDS	TDS: Data view	
	20	PKT_PA_IND	EVC -> TDS	TDS: Planning area (ATO)	
	21	PKT_PA_GP	EVC -> TDS	TDS: Planning area (gradient)	
	22	PKT_PA_SP	EVC -> TDS	TDS: Planning area (speed profile)	
	30	PKT_PLAINTEXT	EVC -> TDS	TDS: Text messages (area E)	







	31	PKT_PLAINTEXT_REMOVE	EVC -> TDS	TDS: Text messages (area E)	
	32	PKT_FIXEDTEXT	EVC -> TDS	TDS: Text messages (area E)	
	33	PKT_FIXEDTEXT_REMOVE	EVC -> TDS	TDS: Text messages (area E)	
	40	PKT_ATO_NEXT_STOPPING_POINT	EVC -> TDS	TDS: stopping points name (G2/3/4)	
	44	PKT_USER_DATA_EXCHANGE	EVC -> TDS		
	45	PKT_USER_DATA_EXCHANGE	TDS -> EVC		
	2	PKT_CONNECTION_CONFIRM	TDS -> EVC	EVC: Connection EVC/TDS	
	9	PKT_LANGUAGE	TDS -> EVC		
	130	PKT_INPUT	TDS -> EVC	EVC: store Train data	X






11 ANNEX Relation between variables and TDS generated views








11.1.1 Relation between variable and graphical elements in data entry window















11.1.2 Relation between variable and graphical elements in default window

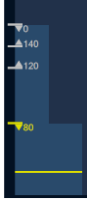





		Symbol	Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
Zone A								
A1								
	Limited supervision		+symbol		§8.2.1.7.4 p.67	Mode management (LS)		Sender ETCS: PD / Q_DISPLAY_LSSMA PD / V_LSSMA
	LSSMA		+digit		§8.2.1.7 p.66	Mode management (LS)		Sender ETCS: PD / Q_DISPLAY_LSSMA
	Time to indication		+digit		§8.2.2.5 p.74	speed and distance monitoring +time to indication +mode +supervision status	National value: +TdispTTI +TTI in CSM	Sender ETCS: PD / Q_DISPLAY_TTI PD / T_TTI
A2	distance to target digital		+digit		§8.2.2.2 p.70	speed and distance monitoring +remaining distance +mode +supervision status		Sender ETCS: PD / Q_DISPLAY_TD_DIG PD / D_TARGET
A3	Distance to target		+bar +scale		§8.2.2.1 p.68	speed and distance monitoring +remaining distance +mode +supervision status		Sender ETCS: PD / Q_DISPLAY_TD_BAR PD / D_TARGET
A4	Adhesion factor		+symbol		§8.2.3.7 p.92	Manage adhesion factor		Sender ETCS: PD / M_INDICATORS




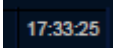




Symbol		Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
Zone B							
B0							
	Speed dial		+short/long speed indicator +range: 4 configurations +speed number		§8.2.1.1 p.46	internal TDS: configuration of the train (max speed)	
	set speed		+circle		§8.2.3.9 p.95		Sender ETCS: PD / V_SET
B1							
	Current train speed pointer		+pointer +color		§8.2.1.2 p.52	speed measurement	Sender ETCS: PD / Q_DISPLAY_CTS PD / V_TRAIN Sender TDS: M_SPEED_STATUS
	Current train speed digital		+digit +color: black/white		§8.2.1.3 p.54	speed measurement	Sender ETCS: PD / Q_DISPLAY_CTS_DIG PD / X_V_TRAIN_DIG100 PD / X_V_TRAIN_DIG10 PD / X_V_TRAIN_DIG1
B2							
	Circular gauge speed		+gauges +color		§8.2.1.4 p.55	speed and distance monitoring +Vperm +Vtarget +Vsbi +mode +supervision status	Sender ETCS: PD / Q_DISPLAY_PS PD / Q_DISPLAY_TS PD / Q_DISPLAY_RS PD / Q_DISPLAY_IS PD / V_PERMIT PD / V_TARGET PD / V_INTERV PD / M_COLOUR_IS PD / M_COLOUR_PS PD / M_COLOUR_TS PD / Q_DISPLAYMODE_PS PD / Q_DISPLAYMODE_TS PD / Q_DISPLAYMODE_IS


Symbol		Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
	Basic speed hook		+gauges +color		§8.2.1.5	speed and distance monitoring +Vperm +Vtarget +mode +supervision status	Sender ETCS: PD / Q_DISPLAY_HOOK_PS PD / Q_DISPLAY_HOOK_TS
	Release speed		+gauge		§8.2.1.6 p.64	speed and distance monitoring + Vrelease +mode supervision status	Sender ETCS: Q_DISPLAY_RS_DIG PD / V_RELEASE PD / M_COLOUR_RS
B3/4/5							
	Track condition		+symbol			Track condition	Sender ETCS: PD / M_INDICATORS Sender TDS:
	LX not protected		+symbol			not protected level crossing	Sender ETCS: PD / M_SAFE_INDICATORS Sender TDS:
B6	Release speed		+digit			speed and distance monitoring	Sender ETCS: PD / V_RELEASE
B7	Mode		+symbol			Mode management	Sender ETCS: PD / M_SAFE_INDICATORS Sender TDS: PD / M_MODE_STATUS
Zone C							
C1							
	Level announcement		+symbol			Level transition	Sender ETCS: PD / M_INDICATORS PD / NID_NTC_ANNOUNCED

		Symbol	Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
	Level acknowledgement		+symbol +with a flashing frame (always)			Level transition		Sender ETCS: PD / M_INDICATORS
	Mode acknowledgement		+symbol +with a flashing frame (always)			Mode profile		Sender ETCS: PD / M_INDICATORS
C2/3/4								
	Toggle function for tunnel stopping area		+symbol	touch		internal TDS		Sender ETCS: PD / M_INDICATORS
	Tunnel stopping area		+symbol + digit					Sender ETCS: PD / Q_DISPLAY_TSA_RD PD / D_TUNNELSTOP Sender TDS:
C5								
C6	Reversing permitted		+symbol			Reversing		Sender ETCS: PD / M_INDICATORS
C7	Override active					Override		Sender ETCS:
C8	Level		+symbol			Level management		Sender ETCS: PD / M_INDICATORS
C9	Emergency brake		+symbol +with a flashing frame (conditional)			Emergency brake		Sender ETCS: PD / M_INDICATORS

Symbol		Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
Zone E							
E1	safe connection radio					Radio connection	Sender ETCS: PD / M_INDICATORS
E5/6/7/8/9							Sender ETCS: MD / PKT_PLAINTEXT MD / PKT_PLAINTEXT_REMOVE MD / PKT_FIXEDTEXT MD / FIXEDTEXT_REMOVE
Zone D							
D	Track ahead free					Track ahead free	Sender ETCS: PD / M_INDICATORS PD / Q_DISPLAY_PA
D1	Distance scale					Internal TDS	Sender ETCS: PD / Q_DISPLAY_PA
D2/3/4	Track condition		+symbol			Track condition	Sender ETCS: PD / Q_DISPLAY_PA PD / D_PA MD / PKT_PA_IND
D5	Gradient						Sender ETCS: PD / Q_DISPLAY_PA PD / D_PA MD / PKT_PA_GP
D6	Speed profile discontinuity		+symbol +number				Sender ETCS: PD / Q_DISPLAY_PA PD / D_PA
D7							
	Planning area speed profile		+diagram +cropped +scaled		§8.3.7 p.105		Sender ETCS: PD / D_PA MD / PKT_PA_SP

		Symbol	Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
	Indication marker		+ line + yellow		§8.3.8 p.107			Sender ETCS: PD / Q_DISPLAY_PA PD / D_PA_INDICATION PD / M_COLOUR_PA_IND
D8								Sender ETCS: Sender TDS:
D9	scale up		+symbol	touch		internal TDS		
D12	scale down		+symbol	touch		internal TDS		
Zone F								
F1	Main							Sender ETCS PD / M_BUTTONS PD / M_SAFE_BUTTONS Sender TDS PD / M_BUTTONS_ACT PD / M_SAFE_BUTTONS_ACT
F2	Override							
F3	Data view							
F4	Spec							
F5	Settings							
F6	Toggling function for tunnel stopping area			soft key				
F7	Toggling function for speed/distance information							
F8	geographical position			soft key				Sender ETCS: PD / Q_DISPLAY_GP PD / M_GEOPOSITION

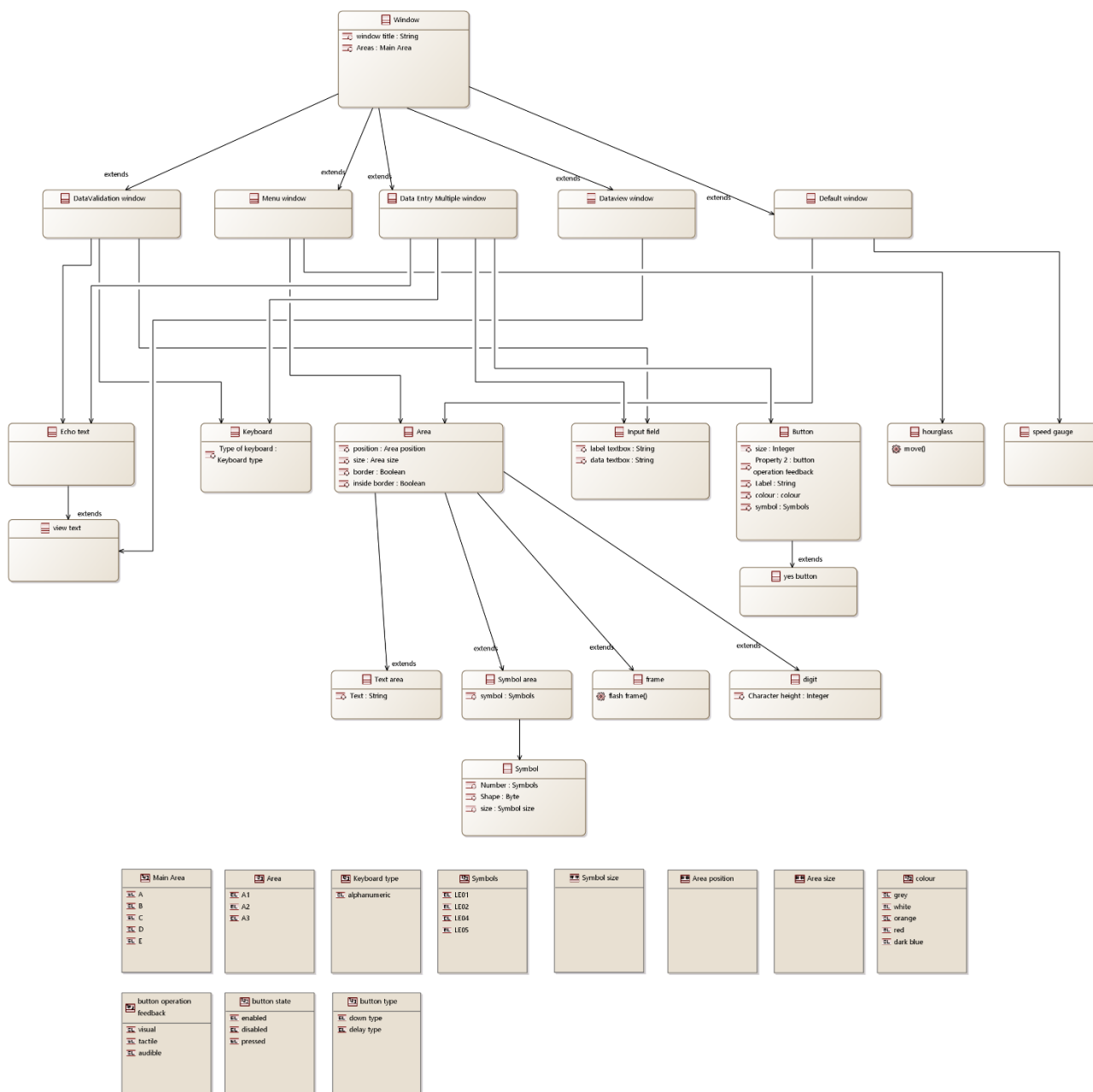
		Symbol	Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
F9	scale up		+symbol	soft key		internal TDS		
F10	scale down		+symbol	soft key		internal TDS		
Zone G								
G12	geographical position			touch				Sender ETCS: PD / Q_DISPLAY_GP PD / M_GEOPOSITION
G13	Local time		+hour, minute, second			internal TDS		Sender ETCS: PD / T_HOURS PD / T_MINUTES PD / T_SECONDS
Zone H								
H3	Previous			soft key		internal TDS		
H4	Next			soft key		internal TDS		
H5								
	scroll up			soft key		internal TDS		
	close window			soft key		internal TDS		

	Symbol	Behaviour	screen	ERA TDS	Function	Parameter	Subset 121 variable PD: process data MD: message data
H6			soft key		internal TDS		
H7	Ack	+symbol	soft key				Sender ETCS: PD / M_TO_ACK PD / M_ACK_TYPE PD / M_ACK_DATA Sender TDS: PD / M_ACK_DISPLAYED PD / M_ACKED

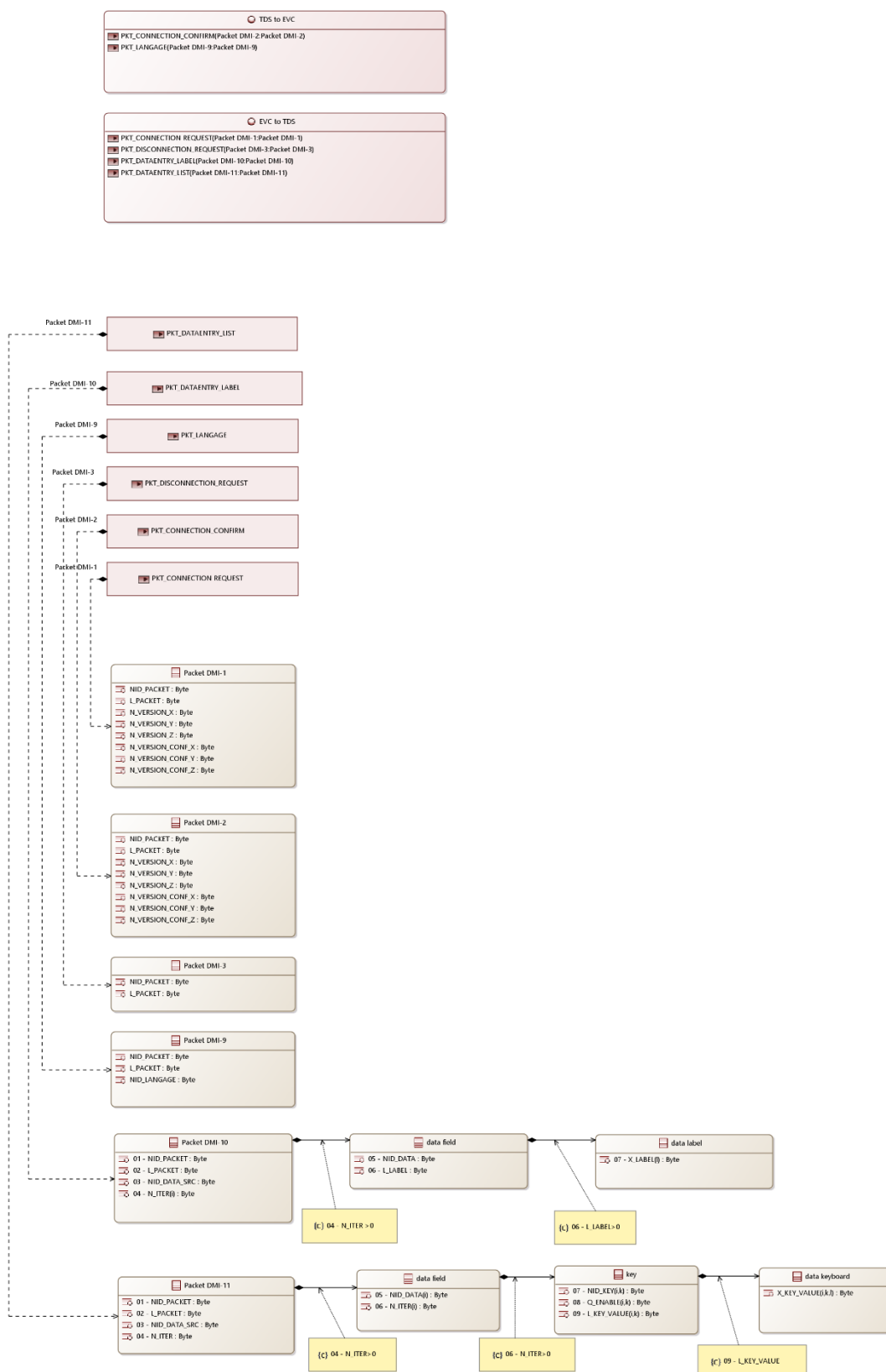
11.1.3 Sound

	ERA TDS	Function	Subset 121 type	
S_info	§14.3.1 p.259		PD	ETCS sender NID_SOUND
Overspeed	§14.3.2 p.259		PD	ETCS sender NID_SOUND
Warning	§14.3.3 p.259		PD	ETCS sender NID_SOUND

12 ANNEX – TDS class diagram



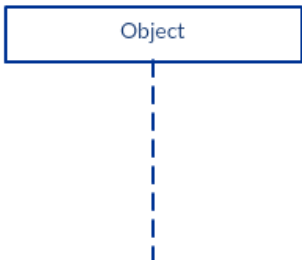
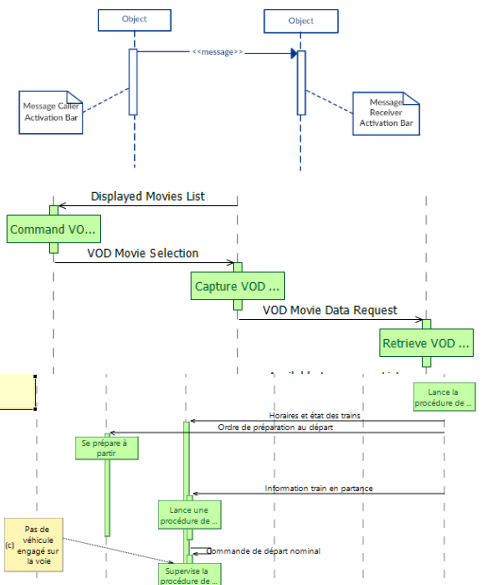

13 ANNEX - Data Modelling of Subset 121 Configuration

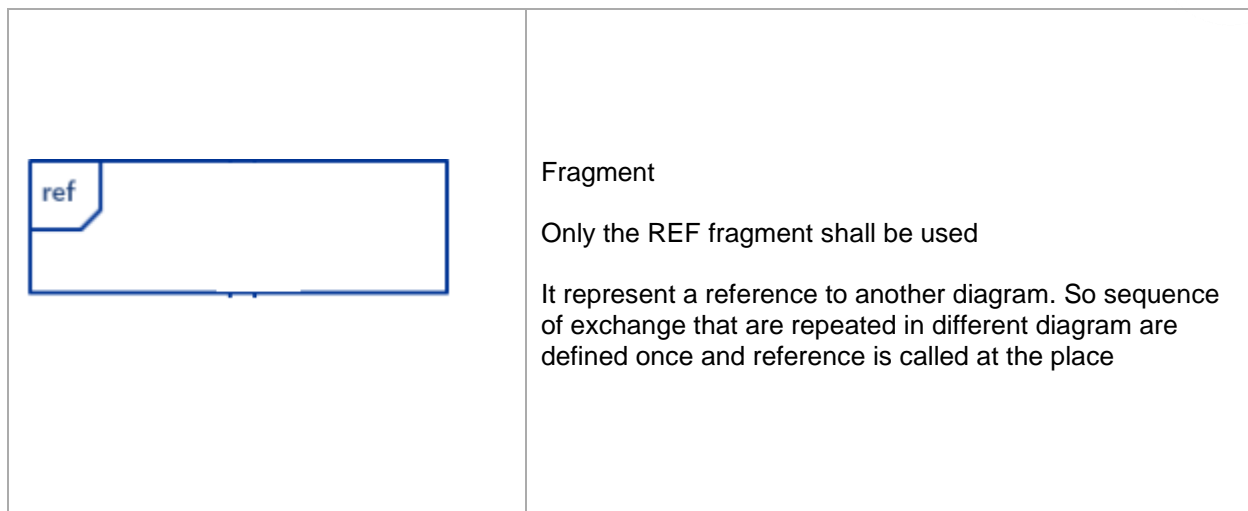


14 ANNEX - TEST CASE TEMPLATE

Test cases are represented in a test sequence format.

Sequence diagram can have several objects:

	<p>swimlane</p> <p>for each actor, include function, mode, execution</p> <p>start point and end point of exchange</p>
	<p>activation box (rectangle)</p> <p>A functional exchange trigger the activation of a function</p> <p>The box represents the duration of activation of function.</p> <p>If a function triggers itself exchange, the arrow starts inside the box</p>
	<p>Functional exchange</p> <p>according to MD and PD defined in ss121 for exchange between controller</p> <p>For functional exchange between TDS and EVC interface controller, the text associated to the line is defined as follow</p> <ul style="list-style-type: none"> • Message data <ul style="list-style-type: none"> ◦ MD - Packet TDS 14: PKT_DATAVIEW • Process data: only on change of value of a variable <ul style="list-style-type: none"> ◦ PD - PD_ETCS_TO_TDS_SAFE - NID_WINDOW = 20 (dataview)

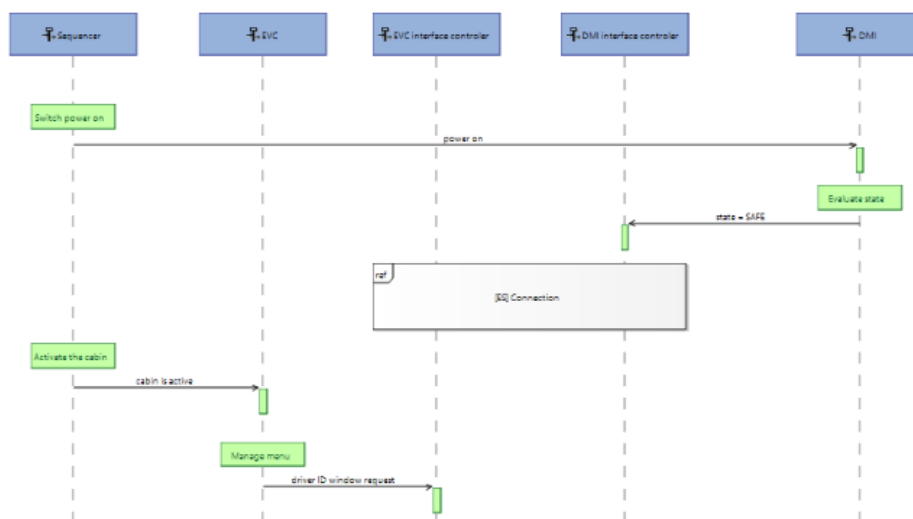


In the upper part of the diagram, we define an actor for each header of swimlanes.
The number and the order shall always be the same:

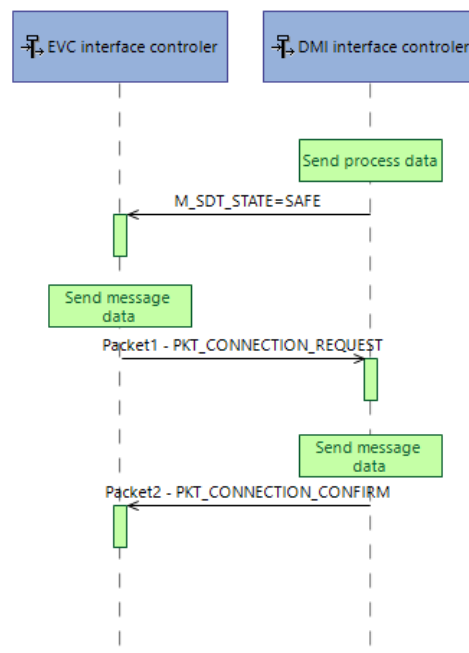
1. Sequencer
2. EVC
3. EVC interface controller
4. TDS interface controller
5. TDS



Below are represented two examples:



Example of an operational test case



Example of a unit test case

Predefined function

Function	Component
Send message data	EVC interface controller
Send process data	EVC interface controller
.....	