OETCS/WP3/D3.5.2

openETCS

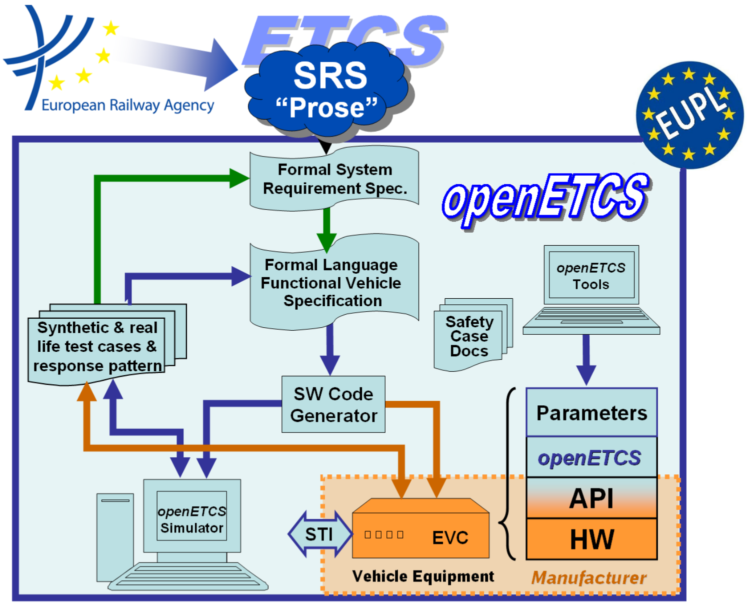
ITEA2 Project
Call 6 11025
2012 – 2015

Work Package 3: "Modeling"

openETCS Design Specification

Baseliyos Jacob, Peter Mahlmann

May 2015



Funded by:

This page is intentionally left blank

Work Package 3: "Modeling" **OETCS/WP3/D3.5.2**
May 2015

openETCS Design Specification

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Baseliyos Jacob (DB Netz AG)	signature Jan Welte (Technische Universität Braunschweig)	signature Izaskun de la Torre (SQS)	signature Klaus-Rüdiger Hase (DB Netz)

Baseliyos Jacob, Peter Mahlmann
DB Netz AG

Architecture and Design Specification

Prepared for openETCS@ITEA2 Project

Abstract: This document gives an introduction to the software and component design of the openETCS OBU model. The functional scope is tailored to cover the functionality required for the openETCS demonstration as an objective of the ITEA2 project. The goal is to develop a formal model and to demonstrate the functionality during a proof of concept on the ETCS Level 2 Utrecht Amsterdam track with real scenarios. It has to be read as a complement to the models in SysML and Scade languages.

Modification History

Version	Section	Modification / Description	Author	Date
0.1	Document	Initial document providing structure	Peter Mahlmann	27.05.2015

Table of Contents

Figures and Tables

Figures

Tables

1 Runtime API

1.1 Functional breakdown

1.1.1 F1: openETCS Runtime System and Input to the EVC)

openETCSAPI.png

Figure 1. openETCS API Highlevel View

Figure 3 shows the structure of API with respect of the software architecture. Input boxes and output boxes not implemented in this stage are marked as red, other interfaces are marked as green. The System covers functions for processing Inputs from other Units, functions for processing Outputs to other functions and a basic runtime system. Inputs are used to feed the input to the executable model before calling it, outputs are used for collecting information provided by the executable model to be passed to the relevant interfaces after the execution cycle has finished.

1.1.1.1 Principles for Interfaces (openETCS)

Information is exchanged *messages* in an asynchronous way. A message is a set of information corresponding to an event of a particular unit, e.g. a balise received from the . The possible kind of messages are described in chapter **??**.

The information is passed to the executable model as parmeters to the snychronous call of a procedure (Interface to the executable model). Since the availability of input messages to the application is not guaranteed the parts of the interfaces are defined with a "present" flag. In addition, fields of input arraysquite often is of variable size. Implementation in the concrete interface in this use-case is the use of a "size" parameter and a "valid"-flag.

1.1.1.2 openETCS Model Runtime System

The openETCS model runtime system also provides:

- Input Functions From other Units
  In this entity messages from other connected units are received.
- Output Functions to other Units
  The entity writes messages to other connected units.
- Conversation Functions for Messages (Bitwalker)
  The conversion function are triggered by Input and Ouput Functions. The main task is to convert input messages from an bit-packed format into logical ETCS messages (the ETCS language) and Output messages from Logical into a bit-packed format. The logical format of the messages is defined for all used types in the openETCS data dictonary.
  Variable size elements in the Messages are converted to fixed length arrays with an used elements indicator.
  Optional elements are indicated with an valid flag. The conversion routines are responsible for checking the data received is valid. If faults are detected the information is passed to the openETCS executable model for further reaction.
- Model Cycle

  The version management function is part of the message handling.This implies, conversions from other physical or logical layouts of messages are mapped onto a generic format used in the EVC. Information about the origin version of the message is part of the messages.

  The executable model is called in cycles. In the cycle

  – First the received input messages are decoded
  – The input data is passed to the executable model in a predefined order. **(Details for the interface to be defined)**.
  – Output is encoded according to the and passed to the buffers to the units.

1.1.1.3 Input Interfaces of the openETCS API From other Units of the OBU

Interfaces are defined in the Scade project APITypes (package API_Msg_Pkg.xscade).

In the interfaces the following principles for indicating the quality of the information is used:

Indicator	Type	Purpose
present	bool	True indicates the component has been changed compared to the previous call of the routine
valid	bool	True indicates the component is valid to be used.

In the next table we can see the interfaces being used in the openETCS system. Details on the interfaces are defined further down.

Unit	Name	Processing Function
	Balise Telegram	Receive Messages
	Driver Machine Interface	DMI Manager
EURORADIO	Communication Management	Communication Management
EURORADIO	Radio Messages	Receive Messages
	Odometer	All Parts
System TIME	Time system of the OBU	All Parts
TIU	Train Data	All Parts

Information in the following sections gives an more detailed overview of the structure of the interfaces.

1.1.1.4 Message based interface (BTM, RTM)

Balise Message (Track to Train)

Message Name	Optional Packets	Restrictions in the current scope
Balise Telegram	3: National Values 41: Level Transition Order 42: Session Management 45: Radio Network registration 46: Conditional Level Transition Order 65: Temporary Speed Restriction 66: Revoke Temporary Speed Restriction 72: Packet for sending plain text messages 137: Stop if in Staff Responsible 255: End of Information	Used in Scenario

Balise Telegram	0, 2, 3, 5, 6, 12, 16, 21, 27, 39, 40, 41, 42, 44, 45, 46, 49, 51, 52, 65, 66, 67, 68, 69, 70, 71, 72, 76, 79, 80, 88, 90, 131, 132, 133, 134, 135, 136, 137, 138, 139, 141, 145, 180, 181, 254	Not Used in Scenario

Radio Messages (Track to Train)

Message Name	Optional Packets	Restrictions in the current scope
2: SR Authorisation	63: List of Balises in SR Authority	Message Not Supported
3: Movement Authority	21: Gradient Profile 27: International Static Speed Profile 49: List of balises for SH Area 80: Mode profile plus common optional packets	a
9: Request To Shorten MA	49: List of balises for SH Area 80: Mode profile	
24: General Message	From RBC: 21: Gradient Profile 27: International Static Speed Profile plus common optional packets From RIU: 44, 45, 143, 180, 254	Messages from RIU are not supported
28: SH authorised	3, 44, 49	
33: MA with Shifted Location Reference	21: Gradient Profile 27: International Static Speed Profile 49: List of balises for SH Area 80: Mode profile plus common optional packets	
37: Infill MA	5, 21, 27, 39, 40, 41, 44, 49, 51, 52, 65, 66, 68, 69, 70, 71, 80, 88, 138, 139	Message Not Supported
List of common optional parameters	3, 5, 39, 40, 51, 41, 42, 44, 45, 52, 57, 58, 64, 65, 66, 68, 69, 70, 71, 72, 76, 79, 88, 131, 138, 139, 140, 180	

The runtime system is in charge to transfer the messages from its stream mode first to compressed message format.

1.1.1.5 Interfaces to the Time System

The interface types are defined in the OBU_Basic_Types_Pkg Package. The system time is defined in the basic software.

The system TIME is provided to the executable model at the begin of the cycle. It is not refreshed during the cycle. The time provided to the application is equal to 0 at power-up of the EVC (it is not a “UTC time” nor a “Local Time”), then must increase at each cycle (unit = 1 msec), until it reaches its maximum value (i.e current EVC limitation = 24 hours)

- TIME (T_internal_Type, 32-bit INT)
  Standardized system time type used for all internal time calculations: in ms. The time is defined as a cyclic counter: When the maximum is exceeded the time starts from 0 again.
- CLOCK (to be implemented)
  The clocking system is provided by the JRU. A GPS based clock is assumed to provide the local time.

1.1.1.6 Interfaces to the Odometry System

The interface types are defined in the OBU_Basic_Types_Pkg Package. The odometer gives the current information of the positing system of the train. In this section the structure of the interfaces are only highlighted. Details, including the internal definitions for distances, locations speed and time are implemented in the package.

- Odometer (odometry_T)
  - valid (bool)
    valid flag, i.e., the information is provided by the ODO system and can be used.
  - timestamp (T_internal_Type)
    of the system when the odometer information was collected. Please, see also general remarks on the time system.
  - Coordinate (odometryLocation_T)
    - nominal (L_internal_Type) [cm]
    - min (L_internal_Type) [cm]
    - max (L_internal_Type) [cm]

    The type used for length values is a 32 bit integer. Min and max value give the interval where the train is to be expected. The bounderies are determined by the inaccuracy of the positioning system. All values are set to 0 when the train starts.
  - speed (OdometrySpeeds_T) [km/h]
    - v_safeNominal (speed internal type) [km/h]
      The safe nominal estimation of the speed which will be bounded between 98% and 100% of the upper estimation
    - v_rawNominal (speed internal type) [km/h]
      The raw nominal estimation of the speed which will be bounded between the lower and the upper estimations
    - v_lower (speed internal type) [km/h]
      The lower estimation of the speed
    - v_upper (speed internal type) [km/h]
      The upper estimation of the speed

    The type used for speed values is a 32 bit integer. Min and max value give the interval where the train is to be expected. The bounderies are determined by the inaccuracy of the positioning system. All values are set to 0 when the train starts.

- acceleration (A_internal_Type)[0.01 m/s2],
  Standardized acceleration type for all internal calculations : in
- motionState (Enumeration)
  indicates whether the train is in motion or in no motion
- motionDirection (Enumeration)
  indicates the direction of the train, i.e., CAB-A first, CAB-B first or unknown.

1.1.1.7 Interfaces to the Train Interfaces (TIU)

The following infomration is based on the implementation of the Alstom API. The interface is organised in packets. The packets of the Alstom implementation are listed in the appendix to this document.

The description of interfaces needed for the current scope will be added according to the use.

1.1.1.8 Output Interfaces of the openETCS API TO other Units of the OBU

From Function	Name	To Unit	Description
	Radio Output Message	EURORADIO	
	Communication Management	EURORADIO	
	Driver Information		
	Train Data	TIU	

Packets: to be completed

Radio Messages to be completed