openETCS SRS Analysis and Modelling Guidelines

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Author : | | Uwe Steinke | | Date : | 2013-12-03 | | | | | |
|  | | | | | | | | | | |
| **Subject :** | | | | | | | Document Review : | | 🗹 | |
| **OpenETCS WP3 Task Force** | | | | | | | Design Review : | | ❒ | |
| - provisional - | | | | | | | Other : | |  | |
|  | |  | | | | | | | | |
|  |  | |  | | |  | |  | |  |
|  |  | |  | | |  | |  | |  |

|  |  |  |
| --- | --- | --- |
| **Name** | **Position** | **Company / Department** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Distribution to:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Position** | **Company / Department** |
| datadictionary@openetcs.org; openETCS srs-analysis@openetcs.org |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Content

1. openETCS SRS Analysis and Modeling Guidelines 3

1.1 references 3

1.2 object 3

2. Data Type Definitions 4

2.1 Use of the ETCS Language 4

2.1.1 Transformations between native ETCS language and OBU internal data types 4

2.1.2 Mapping of basic data types: Native ETCS language ↔OBU internal data types 5

2.1.3 Mapping of type names: Native ETCS language ↔OBU internal data types 6

2.2 Self-Defined Data types for OBU-internal Usage 7

2.2.1 Use of ETCS language type definitions whenever possible 7

2.2.2 Uniform scaled (physical) dimensions 7

2.3 Basic concepts 8

2.3.1 Replacing dynamic data structures with static type defintions 8

2.3.2 Concept of “Valid”-Flags and “Valid”-Strobes 8

2.3.3 Time Stamps 9

2.3.4 Synchronism, interdependencies and execution order of functions 9

# openETCS SRS Analysis and Modeling Guidelines

## references

UNISIG Subset\_026 version\_3.3.0

Chapter 7 : ERTMS / ETCS Language

Chapter 8 : Messages

## object

For the openETCS analysis task force three working groups have been set up for detailing the very basic functions of an ETCS OBU kernel. The working groups are focused on

* Reception and management of the balise information
* Determine train location
* Data definition / data base

The structuring of elementary functions, interface definition and data type definitions can be eased by the agreement on basic principles and methods.

The intention of this document is to define basic concepts, recommendations and guide lines for

* The definition of self-defined internal data types within the openETCS OBU kernel functions,
* The use of the standardized ETCS language defined in Subset 026, chapter 7 and 8,
* The mapping between the ETCS language (Subset 026, chapter 7 and 8) and openETCS OBU internal data types.

# Data Type Definitions

## Use of the ETCS Language

The ETCS Language, i. e. the definition of ETCS data types, packets and messages is specified in Subset 026, chapter 7 and 8.

These definitions reflect the data formats at external interfaces, especially at the air gap between track and train. The data formats have been defined to save bits during their transmission between balises, loop, RBC and train. The various different bit lengths lead to field borders on bit boundaries in data structures and are very unhandy to be used in modelling and implementation languages.

In addition, the ETCS language terminology talks about “variables” instead of “types”. For clarification: all “variable”, “packet” and “message” types are in fact data type definitions and should not be misunderstood as global variables.

To unburden the OBU software modelling and implementation from the inconvenience of arbitrary bit lengths and bit boundaries, the following provisions should be made:

### Transformations between native ETCS language and OBU internal data types

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-ETCS-Language-001#DEF# |
| Titel: | **Transformations betw. native ETCS language elements and OBU internal data types** |
| Description: | Native ETCS language data types must be converted into OBU internal data types at the external interfaces of the OBU software.  This enables the internal OBU functions to deal with internal data types only. |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: | ETCS language in XML-Format:   * <https://github.com/openETCS/dataDictionary/blob/master/Artifacts/subset-026-7/Decoder/Delivery/SubsetFromWord/SubSet026_7.xml> * <https://github.com/openETCS/dataDictionary/blob/master/Artifacts/subset-026-8/Decoder/Delivery/SubsetFromWord/subset8.xml> |
| Test Hints: : |  |
| Comments: |  |

### Mapping of basic data types: Native ETCS language ↔OBU internal data types

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-ETCS-Language-002#DEF# |
| Titel: | **Mapping of basic data types** |
| Description: | The transformations between ETCS data types and the corresponding OBU internal data types are  **Native ETCS language types ↔ OBU internal data types**  Integer 2 – 31 bit ↔ integer32  Enumeration ↔ enumeration  Integer 1 bit ↔ boolean (if character is boolean)  Integer 1 bit ↔ enumeration (if character is enum)  Physical dimensions w. int resolution ↔ real  Structures ↔ structures with fields on at least byte  boundaries  Dynamic iterations of structures ↔ Iterations unrolled into static arrays of  structures of constant maximum size |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: | #REQ-SRS-Analysis-ModellingGuidelines-001#REF#  ETCS Language transformed into OBU internal data types:   * In C format:  <https://github.com/openETCS/dataDictionary/tree/master/Artifacts/subset-026-7/Decoder/Delivery/GeneratedC>  <https://github.com/openETCS/dataDictionary/tree/master/Artifacts/subset-026-8/Decoder/Delivery/GeneratedC> * In SCADE format:  <https://github.com/openETCS/dataDictionary/tree/master/Artifacts/subset-026-7/Decoder/Delivery/ScadeStruct>  <https://github.com/openETCS/dataDictionary/tree/master/Artifacts/subset-026-8/Decoder/Delivery/ScadeStruct> |
| Test Hints: : |  |
| Comments: |  |

### Mapping of type names: Native ETCS language ↔OBU internal data types

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-ETCS-Language-003#DEF# |
| Titel: | **Mapping of type names** |
| Description: | Types defined in the ETCS language shall preserve their names when transformed into their corresponding OBU internal data types. The distinction between both shall be made via different name spaces. |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: | #REQ-SRS-Analysis-ModellingGuidelines-001#REF# |
| Test Hints: : |  |
| Comments: | Justification: Avoids confusion by different names for internal and external data types and preserves the corresponding relationship between both. |

## Self-Defined Data types for OBU-internal Usage

In principle, data type used internally within the ETCS OBU software can be defined freely. On the other hand, the distributed development in the openETCS team could be improved by common understanding and agreement to some basic principles.

### Use of ETCS language type definitions whenever possible

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-Self-Defined-Data-Types-001#DEF# |
| Titel: | **Use of ETCS language type definitions** |
| Description: | Self defined data types shall be based upon ETCS language elements (transformed as specified in #REQ-SRS-Analysis- ETCS-Language-003#REF#) whenever possible.  Type definitions in parallel to existing ETCS language terms shall be avoided. |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: | #REQ-SRS-Analysis- ETCS-Language-003#REF# |
| Test Hints: : |  |
| Comments: | The definition of data types without referring to ETCS language elements can cause ambiguity problems and should be avoided from the beginning. |

### Uniform scaled (physical) dimensions

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-Self-Defined-Data-Types-002#DEF# |
| Titel: | **Uniform scaled (physical) dimensions** |
| Description: | Several elements of the ETCS language use combinations of scaling factors (example: Q\_SCALE) and values (example: D\_LINK) to quantify (physical) dimensions. Self-defined OBU internal types shall be based upon a unique scaling factor instead (example: all lengths and distances in cm). |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: |  |
| Test Hints: : |  |
| Comments: | For consistent resolutions of (physical) dimensions. |

## Basic concepts

### Replacing dynamic data structures with static type defintions

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-Basic-Concepts-001#DEF# |
| Titel: | **Replacing dynamic data structures with static type defintions** |
| Description: | Several structured types of the ETCS language use iterations of substructure loops of arbitrary lengths; an actual iterator value then indicates the actual length of the loop. Since such dynamic allocations are not suited for dependable and safety-related software, the dynamic structures shall be replaced by static structure arrays of a constant maximum length. “valid”-Flags then shall designate array elements filled with data. |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: |  |
| Test Hints: : |  |
| Comments: |  |

### Concept of “Valid”-Flags and “Valid”-Strobes

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-Basic-Concepts-002#DEF# |
| Titel: | **“Valid”-Flags and “Valid”-Strobes** |
| Description: | Since resource allocations like memory space for variables must not be performed dynamically, all resources must be allocated statically and variables like structures and arrays must be of fixed size. This requires a method to designate the array elements arrays filled with valid data and that a structure is filled with valid data. This shall be achieved by adding a boolean “valid” flag to the structures and array elements.  Data flows with event character (example: a just received balise datagram) shall in the same way make use of such “valid” strobes within the data flow type definitions: a “valid” strobe set to true then designates the presence of an event, set to false designates the absence of an event. |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: |  |
| Test Hints: : |  |
| Comments: |  |

### Time Stamps

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-Basic-Concepts-003#DEF# |
| Titel: | **Time Stamps** |
| Description: | All data received via an external sensor shall be marked with the actual system time at the earliest moment of reception. |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: |  |
| Test Hints: : |  |
| Comments: | Serves to eliminate the impact of system internal propagation delays, if required. |

### Synchronism, interdependencies and execution order of functions

|  |  |
| --- | --- |
| ID: | #REQ-SRS-Analysis-Basic-Concepts-004#DEF# |
| Titel: | **Synchronism, Interdependencies and execution order of functions** |
| Description: | * Models shall be created with the awareness that the models will be executed on a synchronously clocked platform * Models shall be created without considering feedback interdependencies of functions or the execution order of functions. This will be solved during the software implementation phase. |
| Category: | Modelling Guideline |
| Priority: | Mandatory |
| Stability: |  |
| Classification: |  |
| Origin: |  |
| References: |  |
| Test Hints: : |  |
| Comments: |  |