MSR Engineering Documentation (MEDOC)

Structure Principles of the MSRNET DTD

Scope: Networks

MSRNET-SP-EN

Signatures:

Companies

MSR MEDOC:
BMW AG
Daimler Chrysler AG
Dr.Ing.h.c.F. Porsche AG
Volkswagen AG
Hella KG Hueck & Co
Robert Bosch GmbH
Siemens AG
TZ Kommunikationstechnik GmbH

Company MSR MEDOC:

Company BMW AG

Team member Expert Dipl.-Ing. U. Vogel

Phone +49-89-382 34117

FAX +49-89-382 33336

Email ulrich.vogel@bmw.de

Company Daimler Chrysler AG

Team member Chairman Dipl.-Ing. P. Rauleder

Phone +49-711-17 41133

FAX +49-711-17 41717

Team member Expert Dipl.-Ing. (FH) U. Bless

Phone +49-711-17 41723

FAX +49-711-17 41717

Email uwe.bless@daimlerchrysler.com

Company Dr.Ing.h.c.F. Porsche AG

Team member Expert Dipl.-Ing. Adams

Phone +49-7044-35 2084

FAX +49-7044-35 2403

Email winfried.adams@porsche.de

Company Volkswagen AG

Team member Expert Dipl.-Ing. O. Marcks

Phone +49-5351-9 73795

FAX +49-5351-9 21881

Email oliver.marcks@volkswagen.de

Company Hella KG Hueck & Co

Team member Expert M. Epping

Phone +49-2941-38 8572

FAX +49-2941-38 2510

Email eppimi1@hella.de

Company Robert Bosch GmbH

Team member Expert Dipl.-Ing. B. Weichel

Phone +49-711-811 8322

FAX +49-711-8118262

Email bernhard.weichel@de.bosch.com

Team member Expert Dipl.-Inf. H. Gengenbach

Phone +49-711-811 20521

FAX +49-711-811 3960

Email helmut.gengenbach@de.bosch.com

Company Siemens AG

Team member Expert Dipl.-Ing. E. Jakobi

Phone +49-941-790 4931

FAX +49-941-790 5805

Email eckard.jakobi@at.siemens.de

Company TZ Kommunikationstechnik GmbH

Team member DTD architect Dipl.-Ing. (FH) H. Klein

Phone +49-711-4609917

FAX +49-711-4609999

Email herbert.klein@tzkom.de

Configuration Parameters

Company
USED MSR-MEDOC

Language
Given There was no language hand over
USED English

NA Proceed
Given NA Treatment not specified
USED NA Elements will not be printed

Filename
USED msrnet-sp-en.sgm

MetaMorphosis-Version
Version 2.2

Form Version
Version 1.12

Table of Contents

I Abstract **10**
II Introduction **11**
III The role of MSR **12**
IV How to read this document **13**
The following conventions apply 13
Examples for SGML instances (fragments) 13
Graphical conventions used in DTD diagrams 13
V Credits **15**
1 General **16**
2 Arrangement of network and control units **17**
2.1 Requirements 17
2.2 Description of network connections in MSRSYS DTD 18
Example for an entry in the MSRSYS instance 19
3 Structuring of MSRNET DTD **23**
3.1 General description 24
3.2 Network architecture 24
3.2.1 Connection components 25
3.2.2 Topology 25
3.2.2.1 Network lines 26
3.2.2.2 Description of network nodes 27
3.2.2.3 Description of segmentation 27
3.2.2.4 Example of a description for the network topology 27
3.2.3 Description of network interfaces 29
3.3 Network operation 31
3.3.1 General network management 32
3.3.2 Initialization 32
3.3.3 Error handling 32
3.3.4 Diagnostics 32
3.3.5 Block transmission modes 32
3.3.6 Network signals 33
3.3.7 Messages 34
4 Basic Structures of the MSR Application Profile **39**
4.1 Not Content Orientated Information (ncoi) 39
4.1.1 Chapter 39
4.1.2 Topic 40
4.1.3 Paragraph Level Elements 40
4.1.3.1 Labeled List 41
4.1.3.2 Figure 43
4.1.3.3 Formula 44
4.1.3.4 Note 44

4.1.4 Character Level Elements 45
4.1.4.1 Rendition Oriented Character Level Elements 45
4.1.4.2 Semantically Oriented Character Level Elements 45
4.1.5 Table 48
4.1.6 Parameter tables 49
4.2 Predefined Document Structure 50
4.3 Project Data 50
4.4 Administrative Data 52
4.5 Variant Concept 53
4.6 Multilinguality 53
A Glossary 55
B Bibliography 58
Literature-List 59
Technical Terms 61
Index 66

List of Figures

Figure 1: convention in DTD diagrams 13
Figure 2: MSRSYS DTD and MSRNET DTD 16
Figure 3: Linking network and control unit 17
Figure 4: Structure of Network Specification in MSRSYS 19
Figure 5: Structure of Network Port in MSRSYS 19
Figure 6: Exemplarily scenario 20
Figure 7: Basic structure of MSRNET DTD 24
Figure 8: Structure for connection components 25
Figure 9: Structure of the topology 26
Figure 10: Structure of the network interface 30
Figure 11: Structuring the network operation 32
Figure 12: Network signals 33
Figure 13: Structure of messages 34
Figure 14: Structure of a single net message 35
Figure 15: Layer model for times/transmission times 37
Figure 16: Structure of signals in a message 38
Figure 17: Structure of <ncoi-1> 39
Figure 18: chapter content model 40
Figure 19: Structure of <topic-1> 40
Figure 20: Structure of <labeled list> 42
Figure 21: Structure of <figure> 44
Figure 22: Structure of prms 49
Figure 23: Principles of information acquisition 50
Figure 24: Structure of <project-data> 51
Figure 25: Support of DTD fragmentation through administrative data 52
Figure 26: Multilingual Paragraph 54

List of Tables

Table 1: **Summary of net-port, net-line, and signal name in the MSRSYS instance according to the given example 20**
Table 2: **semantically oriented character level elements 45**
Table 3: **usage of technical terms 46**
Table 4: **sub-elements for xdoc and xfile 48**
Table 5: **Parameter structure 49**

Document Revisions

Documentversion

Company: MSR MEDOC:
Version: 2 State: RD

Author: Dipl.-Inf. H. Gengenbach Date: 18.2.99

Change	Reason	RELATED-TO
Final release		Part

I Abstract

This document describes the principles for the structure of the MSR development documentation MEDOC for *Networks*. The description can also be used as a guideline for preparing structured development documentation.

II Introduction

This document describes the principles for the structure of for the MSR development documentation MEDOC for *Networks*. The description can also be used as guideline for preparing structured development documentation.

The description is given in, amongst others, the structures of graphics that have been acquired from implementation of the *MSRNET DTD* prototype. The meanings for the symbols used in the graphics are given in convention in DTD diagrams (p. 13) .

The objective of the working group *MEDOC Networks* is to define one format for all possible bus descriptions and descriptions for networks. The data modelling is performed initially on the basis of the CAN (Full CAN and Basic CAN). The transfer to further network formats is made after the data modelling has been completed.

III The role of MSR

It is pointed out here that MSR does not conduct any standardization of the systems or their features that are described with MEDOC. MEDOC supports the use of (inter)national standards and in-house norms, as well as non-standardized norms, for the description of systems of data relevant to the documentation of development processes.

IV How to read this document

The general concepts of the MSR SGML application profile are not replicated in this document. Details can be seen in *[External Document: Structure principles of the MSR application profile / Relevant Position: all]*

The following conventions apply

This document is written using *MSRREP DTD*. The following conventions apply to this document:

<**msrsw**>	SGML elements are noted as technical term **[type]**=SGMLTAG.
[type]	SGML attributes are noted as technical term **[type]**=SGML-attribute.
sgml-attribute	Values of SGML attributes or discrete values for elements are noted as technical term **[type]**=code
ASAP2	The considered languages resp. DTDs are marked as technical term **[type]**=product.
ASAP	The committees are noted as **[type]**=organization
ECU	Objects in general are marked as technical terms **[type]**=other. This might be automotive equipment general objects such as variables etc.

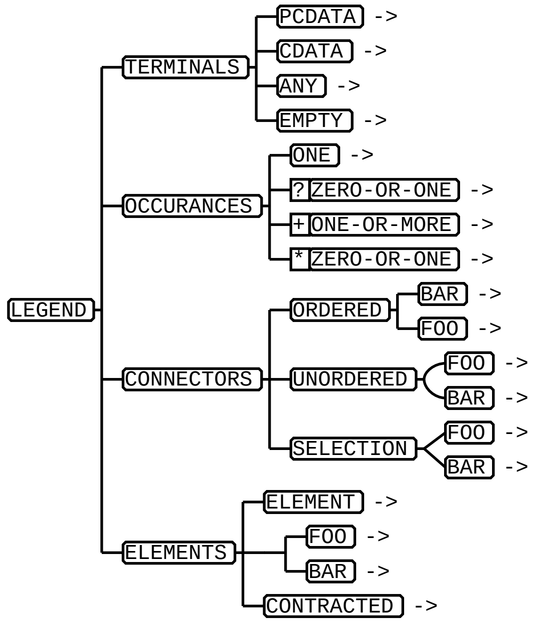
Examples for SGML instances (fragments)

Examples for SGML instances use natural addressing only. For details see *[External Document: Structure principles of the MSR application profile / Relevant Position: Linking]*.

Graphical conventions used in DTD diagrams

The structure of DTDs is shown in the MSR document as DTD diagrams (see convention in DTD diagrams (p. 13)).

Figure 1: convention in DTD diagrams



The meaning of the symbols is:

PCDATA	**Processable Character Data (PCDATA)** Data that consists of zero or more characters of both text and markup. PCDATA is a declared content keyword. PCDATA is used to indicate that all markup delimiters defined in the SGML declaration will be recognized by the parser as markup in the given element rather than data characters.
RCDATA	**Replaceable Character Data (RCDATA)** is data that consists of zero or more characters, in which references to substitutions are not recognized (i.e. RCDATA may contain text and entity references, but no sub-elements). See also: CDATA PCDATA.
CDATA	**Character Data (CDATA)** consists of zero or more text characters, where no markup of any kind is recognized. CDATA is an SGML term. Note that character references are allowed in a CDATA entity (substitution) but not in CDATA content.
ANY	a terminal type indicating that the object may contain text or any element defined in the model.
EMPTY	a terminal type keyword used to indicate that there is no data (i.e. no content, sub-elements or end-tags) for the object allowed in the document instance. This keyword is often used to describe elements that are placeholders or are pointers to external or system-generated data.
One	indicates that the element or the element group occurs exactly once
ZERO-OR-ONE	indicates that the element or the element group is optional
ONE-OR-MORE	indicates that the element or the element group occurs multiple times but at least once
ZERO-OR-MORE	indicates that the element or the element group occurs multiple times but also can be missed (optional)
ORDERED	a connector used to specify that the sibling objects must appear in the document in the order shown in the model
UNORDERED	a connector used to specify that the sibling objects can appear in any order in the document.
SELECTION	a connector used to specify that only one of the sibling objects can appear in the document.
ELEMENT	indicates a single SGML structure element
COLLAPSED	indicates, that the content of the element is not displayed here

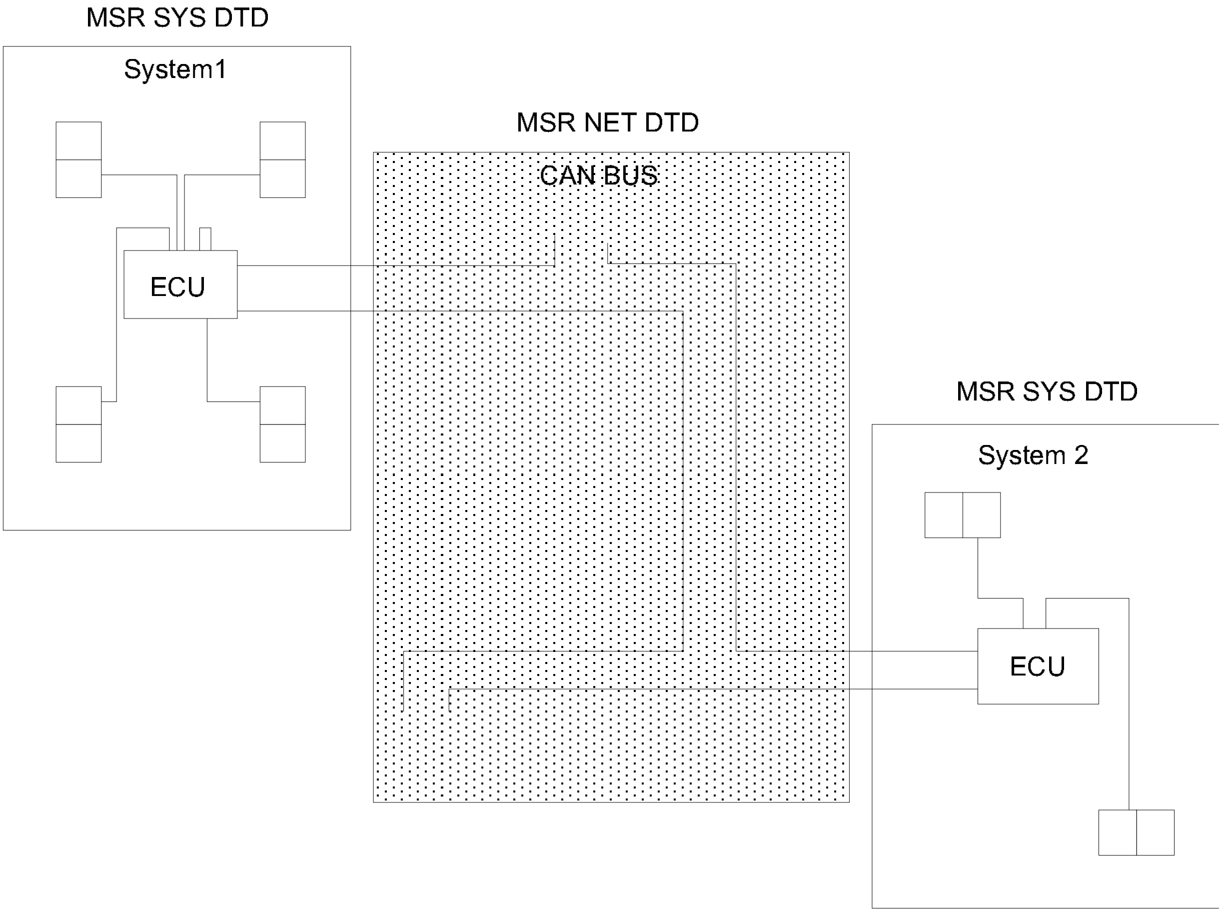
V Credits

The working group thanks for the contributions to the *MSRNET DTD* given by: Thomas Riegraf (Vector Informatik), Detlef Aufdermauer (Volkswagen), Martin Krause (Price Waterhouse Coopers)

1 General

The *MSRSYS DTD* modelled up to now always represented one system including the associated *control unit*s or the associated *control unit* (e.g. an ABS system). Since a network generally links several systems (e.g. the *engine control* and the *ABS control*), it is necessary to describe networks in an SGML instance following the *MSRNET DTD*.

Figure 2: MSRSYS DTD and MSRNET DTD



If a *vehicle system* is connected several network systems in parallel, then these shall be accordingly described in several instances of the *MSRNET DTD*. A clear distinction (engineering as well as organization) between the differing networks is attained by this[1].

1. Gateways initially propose a compilation of several networks in one instance. The gateway functionality is however realized in a specific system or unit, and is therefore to be described in this performance or software.

2 Arrangement of network and control units

2.1 Requirements

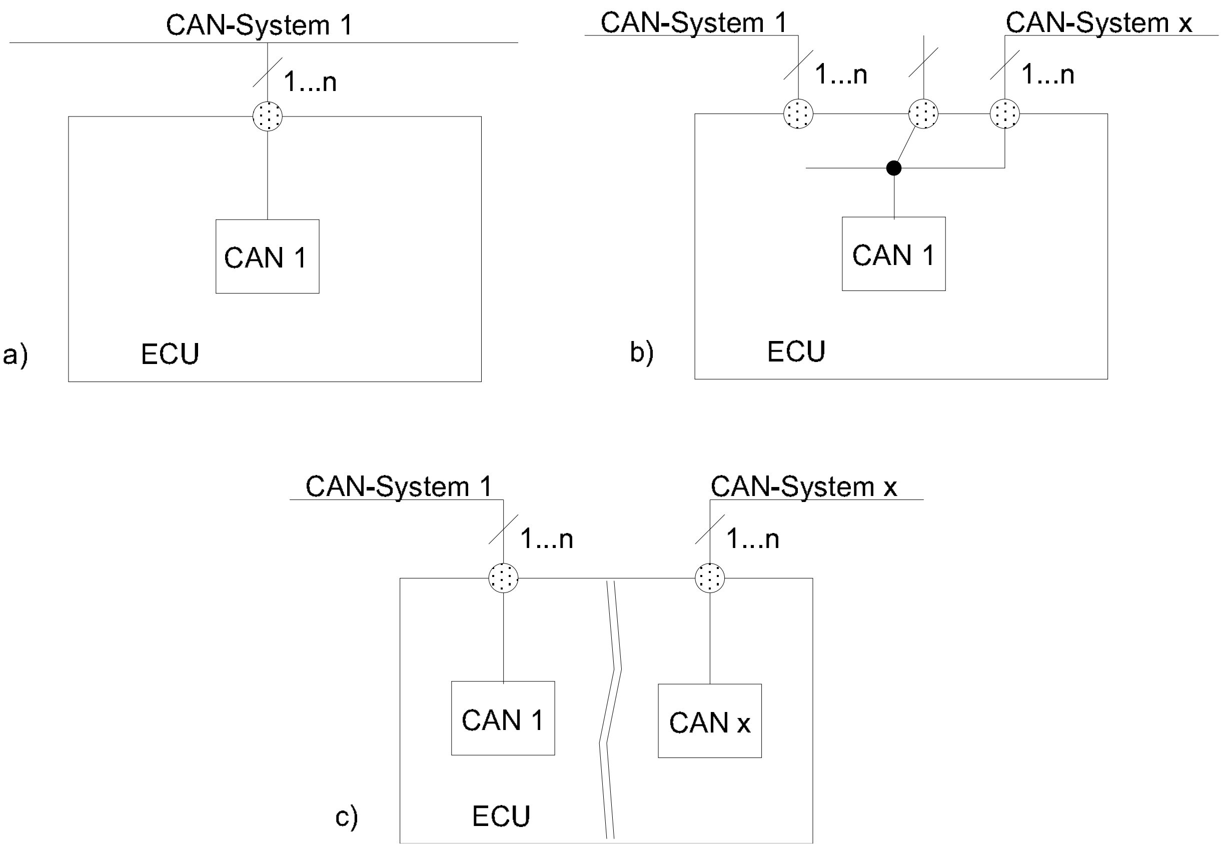
Coupling a *control unit* with one or more networks can be realized by differing ways and means. Three of these possibilities are shown in Linking network and control unit (p. 17) :

a) Single feeder and network system

b) Branched feeder or star point in the *control unit* (multiple n-conductor lines)

c) Connections to several networks (e.g. gateway)

Figure 3: Linking network and control unit



The following cases are covered in Linking network and control unit (p. 17) :

a) The *control unit* is linked to the network via an n-conductor single feeder (e.g. CAN_High, CAN_Low, CAN_Shield).

b) The *control unit* is linked to the network via an n-conductor multiple feeder. This illustrates a branched feeder or star point of the network in the *control unit*. This case is supported by allowing multiple <**net-port**>s within <**network**>.

c) A *control unit* can furthermore be connected to different networks (e.g. gateway). Versions of the interfaces to each of the networks according to case a) and b) are possible here. This is supported by allowing multiple <**network**>s within <**network-spec**>.

b) and c) These cases can occur at the same time. They are supported by allowing multiple <**network**>s each allowing <**net-port**>s.

2.2 Description of network connections in MSRSYS DTD

The definition of network related <**port**>s, <**signal**>s and <**connection**>s in an MSRSYS instance is done in the same way as with any other.

An element <**network-spec**> (refer to Structure of Network Specification in MSRSYS (p. 19)) is introduced in <**architecture**> in the *MSRSYS DTD* in order to be able to specify the network related information for the component types. This modelling was chosen for the following reasons:

- The targets for referencing from *MSRNET DTD* instances are concentrated in <**network-spec**>.
- Network-related characteristics of signals can be concentrated here without a general expansion of the contents model of <**signal**>. This allows to describe network related signals[2] just as any other signal and avoids an oversized content model for <**signal**>.

A possible semantic overlapping with <**signal-class**> is intentionally accepted since the latter has been designed for company and process-specific classifications.

This models the component-specific network characteristics. This implies the explicit arrangement of <**signal**> (which is the component signal) and <**net-line**> (which reflects the sense of the wire with respect to the network, e.g. CAN_High)[3].

Multiple <**net-port**>s are possible in order to be able to cover Linking network and control unit (p. 17) case b). In this case, the network is connected to a component more than once by using several ports. These ports are connected within the component. It can thereby be assumed that the involved ports are named differently. This is done by introducing specific relation of <**net-line**> and <**signal**> per <**net-port**> as shown in Summary of net-port, net-line, and signal name in the MSRSYS instance according to the given example (p. 20) .

2. this is not to mix up with net-signals which represent the signals transmitted on the network by packing them into messages.
3. The net-line identifiers are not explicitly normalized. It is the task of a semantic check to ensure consistency of the net-line identifiers for the differing net-ports.

Figure 4: Structure of Network Specification in MSRSYS

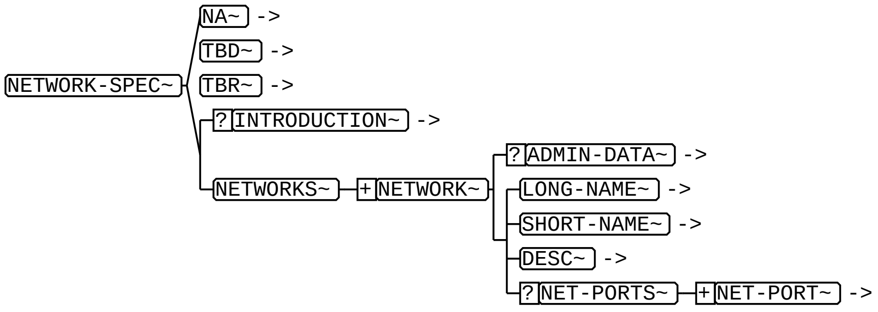
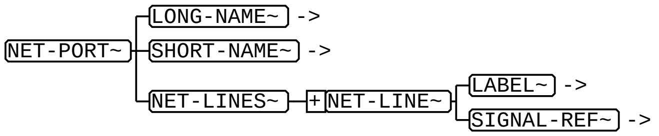


Figure 5: Structure of Network Port in MSRSYS



Example for an entry in the MSRSYS instance

A *MSRSYS DTD* instance (for control unit 2 in Exemplarily scenario (p. 20)) looks in principle like the following (the counterpart in the *MSRNET DTD* instance is given in Example of a description for the network topology (p. 27)):

Figure 6: Exemplarily scenario

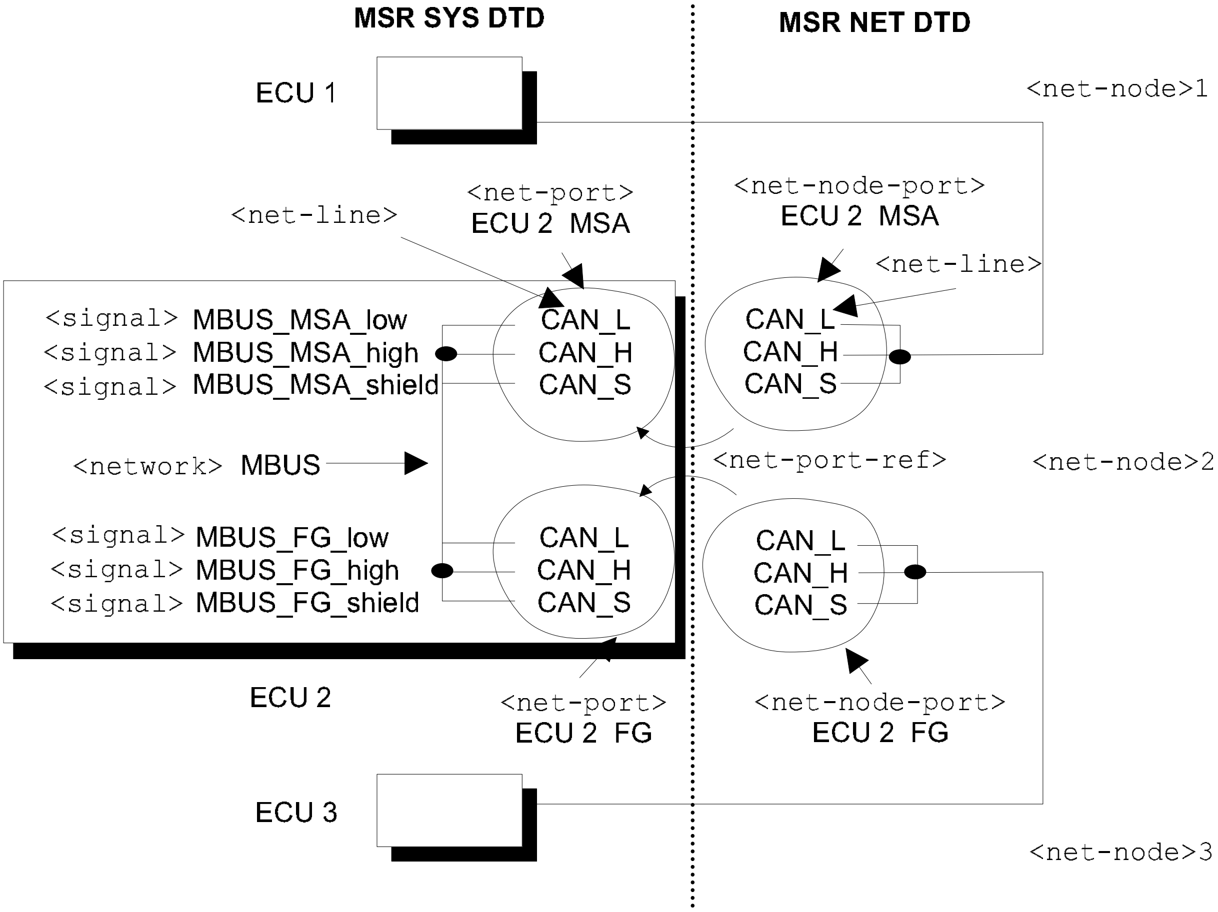


Table 1: Summary of net-port, net-line, and signal name in the MSRSYS instance according to the given example

Net-Port	**Net-Line**	**Signal**
MSA	CAN_LOW	MBUS_MSA_LOW
	CAN_HIGH	MBUS_MSA_HIGH
	CAN_SHIELD	MBUS_MSA_SHIELD
FG	CAN_LOW	MBUS_FG_LOW
	CAN_HIGH	MBUS_FG_HIGH
	CAN_SHIELD	MBUS_FG_SHIELD

```
<part-type>
   <long-name>control unit 2</long-name>
   <short-name>SG2</short-name>
...

 <architecture>
   <scheme-diagrams> ...
   <interface-spec>  ...
   <signal-spec>     ...
   <network-spec>
    <networks>
      <network><long-name>CAN bus for engine management</>
        <short-name>MBUS</>
        <desc>This bus primarily handles engine management</>
        <net-ports>
           <net-port>
              <long-name>engine-side connection</>
              <short-name>MSA</>
              <net-lines>
                <net-line>
                   <net-line-name>CAN_LOW</>
                   <signal-ref sref="MBUS_MSA_LOW">
                </>
                <net-line>
                   <net-line-name>CAN_High</>
                   <signal-ref sref="MBUS_MSA_HIGH">
                </>
                <net-line>
                   <net-line-name>CAN_shield</>
                   <signal-ref sref="MBUS_MSA_SHIELD">
                </>
           </net-port>
           <net-port>
              <long-name>connection on passenger-compartment side</>
              <short-name>FG</>
              <net-lines>
                <net-line>
                   <net-line-name>CAN_LOW</>
                   <signal-ref sref="MBUS_FG_LOW">
                </>
                <net-line>
                   <net-line-name>CAN_HIGH</>
                   <signal-ref sref="MBUS_FG_HIGH">
                </>
                <net-line>
                   <net-line-name>CAN_SHIELD</>
                   <signal-ref sref="MBUS_FG_SHIELD">
                </>
```

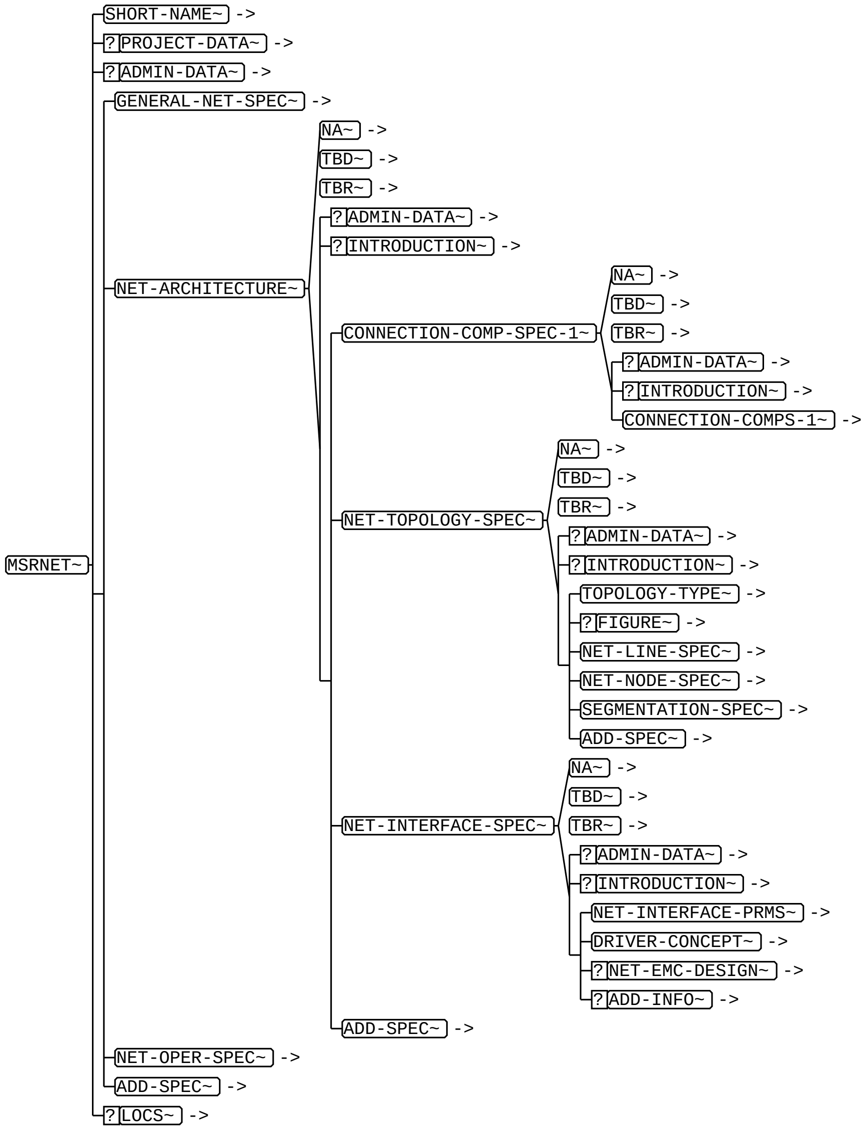
```
            </net-port>
        </net-ports>
     </network>
    </networks>
   </network-spec>
  ...
</part-type>
```

3 Structuring of MSRNET DTD

The description of a CAN network is made up of the following parts:

- Long and short name of the instance (<**long-name**> <**short-name**>)
- Project information <**project-data**>
- Administration information <**admin-data**>
- General global description <**general-net-spec**> General description (p. 24)
- Network architecture (<**net-architecture**>) covering topology, cables, network parameters, driver concept and network EMC design.
- Network operation (<**net-oper-spec**>) covering data traffic, signals, and messages.
- Additional specification (<**add-spec**>)

Figure 7: Basic structure of MSRNET DTD



3.1 General description

General information regarding the network can be given in <**general-net-spec**>. Described in this are for example, the application for the network (power train, comfort area).

3.2 Network architecture

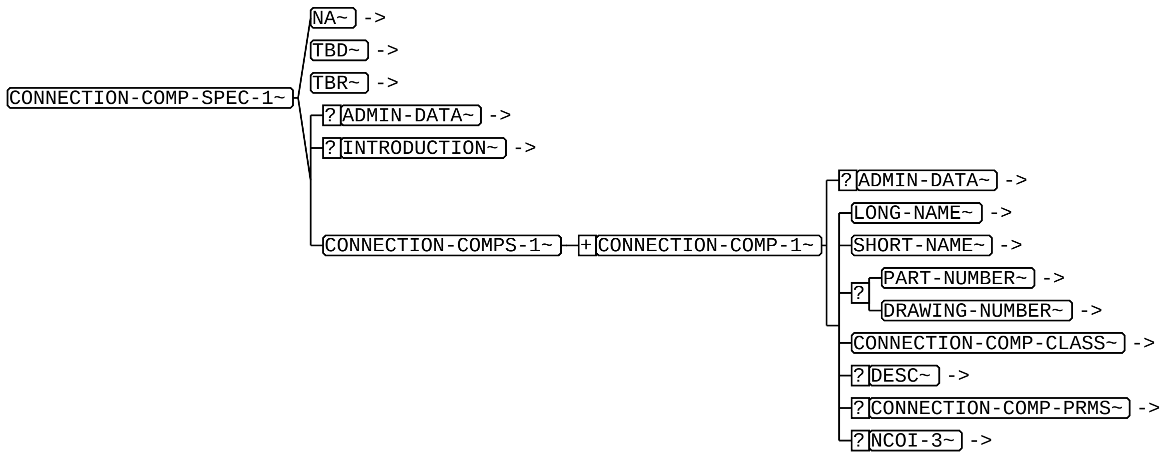
The description of the network architecture (<**net-architecture**>) comprises the topics ”Connection component” <**connection-comp-spec-1**>, ”Topology” (<**net-topology-spec**>) and ”Network interface” <**net-interface-spec**>. Both predefined as well as arbitrary parameters can be used to specify the network and the network hardware.

<**add-spec**> serves to define additional specifications for which no specific structure is given in the DTD.

3.2.1 Connection components

The network is physically set up by connection components such as cables. These components are described in <**connection-comp-spec-1**>[4]. A connection (refer to Structure for connection components (p. 25)), similar as in *MSRSYS DTD*.

Figure 8: Structure for connection components



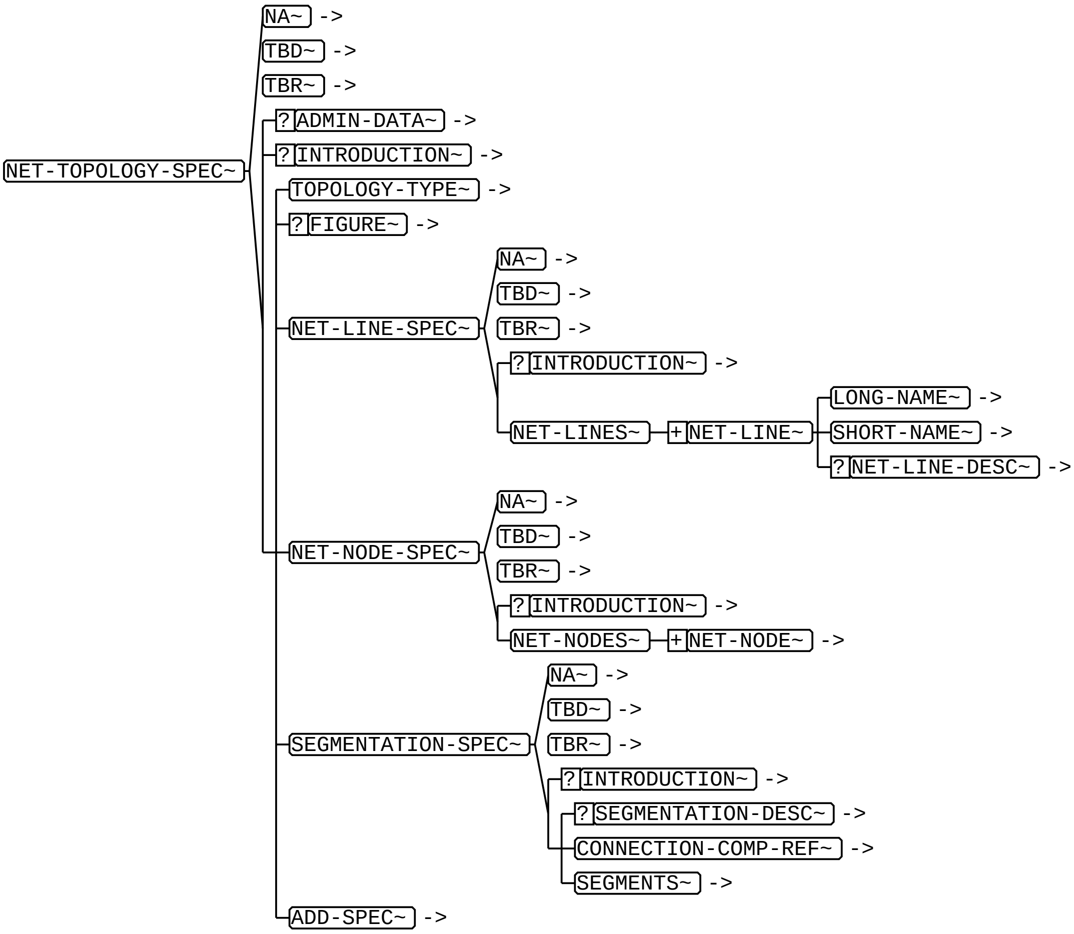
3.2.2 Topology

The topology type, the nodes as well as the segmentation of the network can be documented within <**net-topology-spec**> (refer to Structure of the topology (p. 26)). The type (<**topology-type**>) can express characteristics such as one of ”ring”, ”star”, ”bus”, ”bus with single feeder” or even ”mixed form” and can be supplemented by a figure.

In addition to this, network lines (<**net-line**>), nodes <**net-nodes**> (Network lines (p. 26)) and segmentation <**segmentation**> (Description of segmentation (p. 27)) can be described.

4. A restricted model is used here which does not permit use of a part-type reference.

Figure 9: Structure of the topology



3.2.2.1 Network lines

The physical layer of the network is established by multiple logical[5] signals (called <**net-line**> here). An example for this is *CAN_LOW*, *CAN_HIGH*, *CAN_SHIELD*. These wires can be described in topology (<**net-lines**>). The description comprises as a minimum, a <**short-name**> and a <**long-name**>. A detailed presentation (<**net-line-desc**>) can be given with tables, graphics, etc.

In the *MSRSYS DTD* instances, these <**net-line**>s are assigned to the component signals (compare Description of network connections in MSRSYS DTD (p. 18) , <**net-line**>). Thus it is possible by this to check the consistency[6] (semantics).

---

5. The physical specification of the connection is described in connection-comp-spec-1.
6. Formal referencing of net-lines between msrsys and msrnet was dispensed with for pragmatic reasons since the same designations are used as a rule in all instances.

An explicit assignment of these signals to the lines in the connection-component (e.g. by color code) is not made as the structure of <**connection-comp-1**> is not detailed enough.

3.2.2.2 Description of network nodes

The description of the node comprises:

<**short-name**> Short name for the node in the network. This can also be a number.

<**long-name**> Long name for the node in the network.

<**node-type**> The node type serves to differentiate between *participants* and *auxiliary nodes*. Auxiliary nodes do not participate in the network traffic but rather only serve the physical structure or presentation of the topology.

<**net-node-port**>s The ports for the node are defined here (compare Description of network connections in MSRSYS DTD (p. 18)). This is the only point where the referential consistency to *MSRSYS DTD* instances must be assured. The <**net-node-port**> references the component (<**part-ref**>) as well as the net-port (<**net-port-ref**>) associated with the component type from the *MSRSYS DTD*. [7]

<**interface-circuit**> Information on the terminal circuitry. <**interface-circuit**> can be given here. This information is redundant to *MSRSYS DTD*. The structure is retained in order to be able to give an autonomous description for a MSRNET instances. The information is optional.

3.2.2.3 Description of segmentation

The segmentation can be described by text and graphics (in <**segmentation-desc**>) as well as by a formal specification (in <**segments**>).

The type of cable used as standard for the network can be given in <**connection-comp-ref**>.

The formal specification describes all segments (<**segment**>):

- Segments can be given in variant-specific manner (specified by <**variant-def-refs**>).
- Details on the start-nodes and end-nodes (within <**segment-end-nodes**>), segment lengths (in <**segment-length**>) as well as cable types for the individual segments. The start and end nodes are defined by referencing one of the <**net-node-port**>s of the node in question (<**net-node-port-ref**>).
- The <**segment-length**> is to be given in the SI unit of meters[8].
- A segment-specific type of cable can be given (in <**connection-comp-ref**>) if this deviates from the type of cable used as standard for the network.

3.2.2.4 Example of a description for the network topology

The following example corresponds to the example given in Exemplarily scenario (p. 20) :

7. All node-ports of a node must reference the same part. The reference to the part has not be ”taken out” of the node-ports in spite of this. This would have meant a distribution for the definition of the the semantics for the reference. This should be avoided for handling reasons. It is the task of a semantic check to determine any inconsistencies.
8. The application of the standard parameters model was dispensed with here in order to enable the presentation of segmentation as a closed table.

```
<NET-TOPOLOGY-SPEC>
 <TOPOLOGY-TYPE>bus</TOPOLOGY-TYPE>
 <FIGURE ID="ID-of-topology-figure">
  <LONG-NAME></LONG-NAME>
  <GRAPHIC FILENAME="bus05.eps" NOTATION="EPS"></GRAPHIC>
 </FIGURE>
 <NET-LINE-SPEC><NET-LINES>
  <NET-LINE ID="ID-of-CAN1L">
   <LONG-NAME>CAN_L</LONG-NAME><SHORT-NAME>CAN_L</SHORT-NAME></NET-LINE>
  <NET-LINE ID="ID-of-CAN2H">
   <LONG-NAME>CAN_H</LONG-NAME><SHORT-NAME>CAN_H</SHORT-NAME></NET-LINE>
  <NET-LINE ID="ID-of-CAN1S">
   <LONG-NAME>CAN_S</LONG-NAME><SHORT-NAME>CAN_S</SHORT-NAME></NET-LINE>
  </NET-LINES>
</NET-LINE-SPEC>
<NET-NODE-SPEC>
 <NET-NODES>
   <NET-NODE ID="ID-of-ECU1">
    <LONG-NAME>control unit 1</LONG-NAME>
    <SHORT-NAME>ECU1</SHORT-NAME>
    <NET-NODE-VARIANTS><NODE-VARIANT>
     <NET-NODE-PORTS>
      <NET-NODE-PORT ID="ID-of-ECUCAN1">
       <LONG-NAME>control unit 1 - CAN</LONG-NAME>
       <SHORT-NAME>ECUCAN1</SHORT-NAME>
       <LABEL>CAN</LABEL>
       <NET-PORT-REF NET-PORT="ID-of-nameloc-ECU1CAN">/engine-mgnt/ECU1/CAN
        </NET-PORT-REF></NET-NODE-PORT></NET-NODE-PORTS>
      <NODE-TYPE>participator</NODE-TYPE></NODE-VARIANT>
      </NET-NODE-VARIANTS></NET-NODE>
   <NET-NODE ID="ID-of-ECU2">
    <LONG-NAME>control unit 2</LONG-NAME>
    <SHORT-NAME>ECU2</SHORT-NAME>
    <NET-NODE-VARIANTS><NODE-VARIANT>
     <NET-NODE-PORTS>
      <NET-NODE-PORT ID="ID-of-ECUFG">
       <LONG-NAME>control unit 2 - FG</LONG-NAME>
       <SHORT-NAME>ECU2FG</SHORT-NAME>
       <LABEL>FG</LABEL>
       <NET-PORT-REF NET-PORT="ID-of-nameloc-ECU2-MSA">/engine-mgnt/ECU2/MBUS/MSA
        </NET-PORT-REF></NET-NODE-PORT>
      <NET-NODE-PORT ID="ID-of-ECUMSA">
       <LONG-NAME>control unit 2 - MSA</LONG-NAME>
       <SHORT-NAME>ECU2MSA</SHORT-NAME>
       <LABEL>MSA</LABEL>
       <NET-PORT-REF NET-PORT="ID-of-nameloc-ECU2-FG">/engine-mgnt/ECU2/MBUS/FG
        </NET-PORT-REF></NET-NODE-PORT></NET-NODE-PORTS>
     <NODE-TYPE>participator</NODE-TYPE></NODE-VARIANT>
```

```
     </NET-NODE-VARIANTS></NET-NODE>
   <NET-NODE ID="ID-of-ECU3">
    <LONG-NAME>control unit 3</LONG-NAME>
    <SHORT-NAME>ECU3</SHORT-NAME>
    <NET-NODE-VARIANTS><NODE-VARIANT>
     <NET-NODE-PORTS>
      <NET-NODE-PORT ID="ID-of-ECUCAN3">
       <LONG-NAME>control unit 3 - CAN</LONG-NAME>
       <SHORT-NAME>ECUCAN3</SHORT-NAME>
       <LABEL>CAN</LABEL>
       <NET-PORT-REF NET-PORT="ID-of-nameloc-ECUCAN3">/engine-mgnt/ECU3/CAN
        </NET-PORT-REF></NET-NODE-PORT></NET-NODE-PORTS>
     <NODE-TYPE>auxiliary</NODE-TYPE></NODE-VARIANT></NET-NODE-VARIANTS>
   </NET-NODE></NET-NODES></NET-NODE-SPEC>
<SEGMENTATION-SPEC>
 <CONNECTION-COMP-REF></CONNECTION-COMP-REF>
 <SEGMENTS>
  &lt;SEGMENT&gt;
   <SEGMENT-END-NODES>
    <NET-NODE-PORT-REF NET-NODE-PORT="ID-of-ECU1"></NET-NODE-PORT-REF>
    <NET-NODE-PORT-REF NET-NODE-PORT="ID-of-ECUMSA"></NET-NODE-PORT-REF>
    </SEGMENT-END-NODES>
   <SEGMENT-LENGTH>
    <SHORT-NAME>lseg1</SHORT-NAME>
    <PRM-CHAR>
     <ABS>1.5</ABS>
     <TOL>-</TOL>
     <UNIT>m</UNIT></PRM-CHAR></SEGMENT-LENGTH>
   &lt;/SEGMENT&gt;
  &lt;SEGMENT&gt;
   <SEGMENT-END-NODES>
    <NET-NODE-PORT-REF NET-NODE-PORT="ID-of-ECUFG"></NET-NODE-PORT-REF>
    <NET-NODE-PORT-REF NET-NODE-PORT="ID-of-ECUCAN3"></NET-NODE-PORT-REF>
    </SEGMENT-END-NODES>
   <SEGMENT-LENGTH>
    <SHORT-NAME>lseg2</SHORT-NAME>
    <PRM-CHAR>
     <ABS>2</ABS>
     <TOL>-</TOL>
     <UNIT>m</UNIT></PRM-CHAR></SEGMENT-LENGTH>
   &lt;/SEGMENT&gt;
  </SEGMENTS>
 </SEGMENTATION-SPEC>
</NET-TOPOLOGY-SPEC>
```

3.2.3 Description of network interfaces

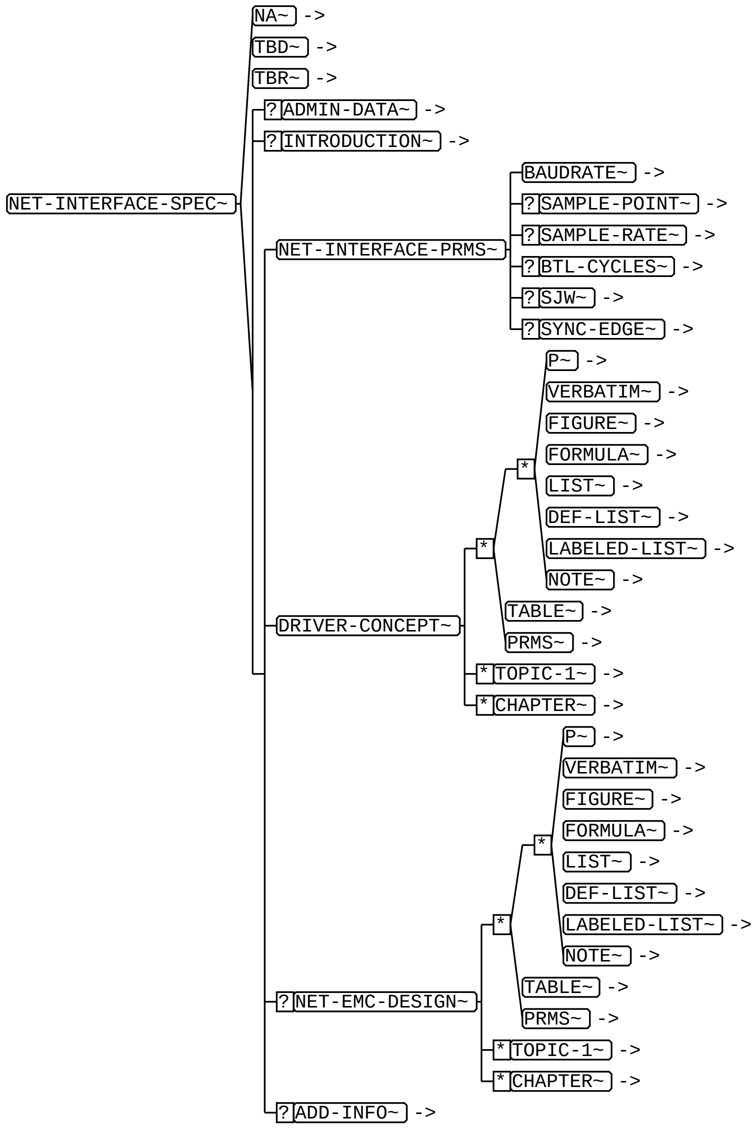
The network interface (<**net-interface-spec**>) constitutes the second part of the hardware description. This describes the driver concept (<**driver-concept**>), the network parameters globally defined for all participants (<**net-interface-**

prms>), notes on the EMC design for the network (<**net-emc-design**>) as well as supplementary information (<**add-info**>).

The description of the bus interface, or bus link, consists of a bus-global physical data sheet (requirements); either according to ISO Norm CAN High-Speed (p. **??**) resp. ISO Norm CAN Low-Speed (p. **??**) or individually characterized.

The detailed description of the CAN-controller hardware occurs within <**part-type-spec**> in the *MSRSYS DTD*.

Figure 10: Structure of the network interface



The network parameters defined globally for all participants are summarized as a parameters table (<**net-interface-prms**>)[9]:

<baudrate> The baud rate (Baud rate (p. **??**)) in Hz. The <**abs**>-<**tol**> model must be used. The tolerance shall be given in %.

The baud-rate programming for the *CAN-controller* can be derived from the above parameter, is not however unique (e.g. *Register occupation*). It concerns here a description of the bus link at a higher level (higher than the physical level). This can also be termed parameterization of the CAN controller and is to be documented for the CAN controller or for the *control unit*.

<sample-point> The sample point for a single bit in a message. The information is given as a percentage. (<**abs**>-<**tol**> model without completed tolerance information)

This information is optional.

This parameter indicates the point in time when a bit will be sampled. It is given as a percentage of the bit time (transmission time for one bit). A typical value is 75 %.

<sample-rate> Number of sampling operations per bit

This information is optional.

<btl-cycles> Number of BTL cycles (p. **??**) BTL cycles as a whole-number value, i.e. the resolution of one bit (<**abs**>-<**tol**> model without completed tolerance information).

This information is optional.

<sjw> SJW (Synchronization Jump Width) as whole-number value (SJW (p. **??**)). The parameter ”SJW” (Syncro Jump Width) is given as a percentage of the bit time as well.

This information is optional.

<sync-edge> Synchronization flank, to be given as a <**text**> parameter

This information is optional.

The <**baudrate**> is global to the bus. The parameters <**sample-point**>, <**sjw**> and <**btl-cycles**> are generally global, can however be participant-specific in exceptional cases. These special circumstances are not covered in *MSR NET DTD*.

The driver modules used or to be used can be described within the scope of the optional driver concept (<**driver-concept**>).

Just as for the driver concept, an structure (<**net-emc-design**>) is available for describing the network EMC design.

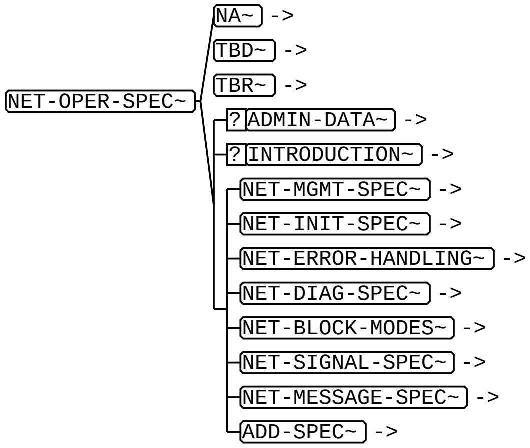
Additional physical parameters such as flank steepness, input resistance and signal level shall be documented as required for the additional information (<**add-info**>).

3.3 Network operation

This section (<**net-oper-spec**>) describes the data traffic for the differing network services (General network management (p. 32) as well as the behavior when faults occur (Error handling (p. 32)) and similar. The structures for the block-wise transmission of data are illustrated in Block transmission modes (p. 32) .

9. A specific parameter model should really be used for each of these parameters. The MSR standard parameters model has however been used in the interests of generalization. If this is executed in future as the architectural form, then more specific model models could be used. The user must however first select the correct model, or the author environment should propose the correct model.

Figure 11: Structuring the network operation



A message that can be transmitted via CAN contains several signals as a rule, whereby it is possible that one signal is being used in more than one message. It is for this reason that the signals (<**net-signal-spec**>, refer to Network signals (p. 33)) are specified first, and then the messages (<**net-message-spec**> refer to Messages (p. 34)).

3.3.1 General network management

Information can be provided in this section (<**net-mgmt-spec**>) on, amongst others, sleep/wake-up mechanisms. If the network can run in different operational modes, then these (and the transition mechanisms) are to be described here as well.

All net-signals and messages used for network management must be specified as net-signals (<**net-signal**>) or net-messages <**net-message**> (compare Network signals (p. 33)). Grouping single messages into groups (using <**net-message-set**>, compare Messages (p. 34)) serves this purpose in particular. Network management signals can be summarized manually as a separate table (CALS table) with <**xref**>s when preparing the documentation[10].

The provsion of standby values is not included in the network description (e.g. for *provisionnet-signal group*) because this is to be executed as a rule by the recipient of the signal.

3.3.2 Initialization

The initialization (<**net-init-spec**>) describes services and protocols that serve to bring a network participant to a state capable of communication following power-up or after a reset.

3.3.3 Error handling

The section <**net-error-handling**> documents the mechanisms for handling errors within the network.

3.3.4 Diagnostics

The section <**net-diag-spec**> documents the mechanisms for processing diagnostic routines in the network.

3.3.5 Block transmission modes

The block transmission modes (<**net-block-modes**>) describe application-specific protocols (initialization, diagnostics and similar) for the transfer of data packages.

---

10. A future revision of the DTD could provide the structural basis for automation of this.

It is recommended to link to (with <**xref**>) the messages used for the block transmission modes. Signals are also used to set up these messages. These signals are identified as such by the net-signal class (<**net-signal-class**>).

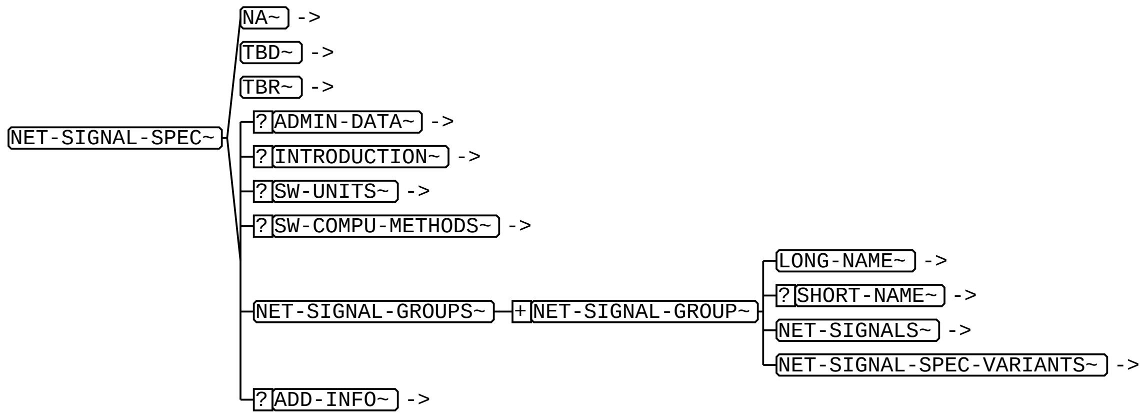
3.3.6 Network signals

<**net-signal-spec**> [11] (refer to Network signals (p. 33)) specifies the signals transmitted in the network.

Network signals are related to the variables in control-unit software. Therefore optional parts of the software data dictionary (<**sw-units**> and <**sw-compu-methods**>) are included in the DTD.

The <**net-signal-group**>s contain those signals having exactly the same characteristics. Therefore duplicated descriptions are not necessary.

Figure 12: Network signals



The following signal characteristics (<**net-signal-spec-variant**>) exist[12]:

<**bitsize**> Length of the signal in bits

<**init-value**> Optional initialization value - this value is transmitted if the associated physical signal has not yet been measured.

<**net-signal-class**> The optional net-signal class (<**net-signal-class**>) can be an "*Application signal*" or "*Network management signal*".

<**error-values**> This value is transmitted if the associated physical signal is not available because of an error situation (e.g. sensor fallen off). Several <**error-value**>s can be given in order to differ between several error situations. This information is optional.

<**sw-limits**> Optional information on the range of values for the signals. The information can be provided both as a network-internal (<**coded**>) as well as a physical (<**phys**>) presentation.

11. The term network signal is used here for differentiation in MSRSYS from the term signal.
12. Some of this information is optional, in order to be able for instance, to apply name and size to these block mode signals as well.

<**sw-compu-method-ref**> The referenced conversion method specifies how the network-internal presentation (coded or internal values) is converted into the corresponding physical values (external values). Refer also to: *[External Document: Structures Principles for MSR Software DTD / State: 1.1.0 / Publisher: MSR AG-MEDOC / Relevant Position: Content model software]*

This information is optional.

3.3.7 Messages

The signals in the network are transmitted as sets of messages. These messages can be specified in <**net-message-spec**> (refer to Structure of messages (p. 34)).

Network messages can be grouped in <**net-message-set**>s. The grouping is **not** intended for messages that have identical specifications. It rather serves for logical grouping of messages.

Figure 13: Structure of messages

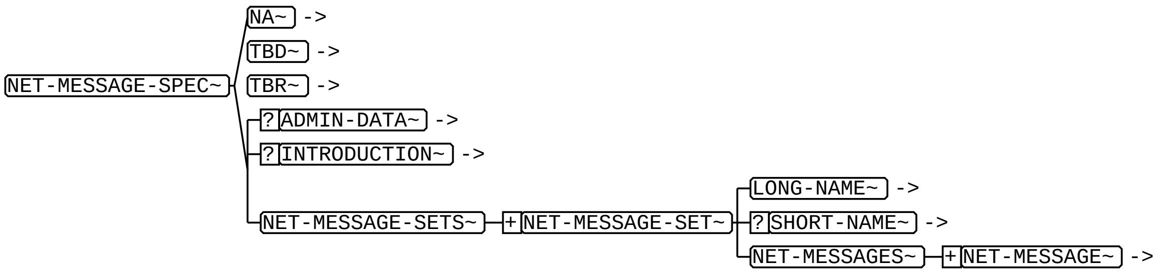
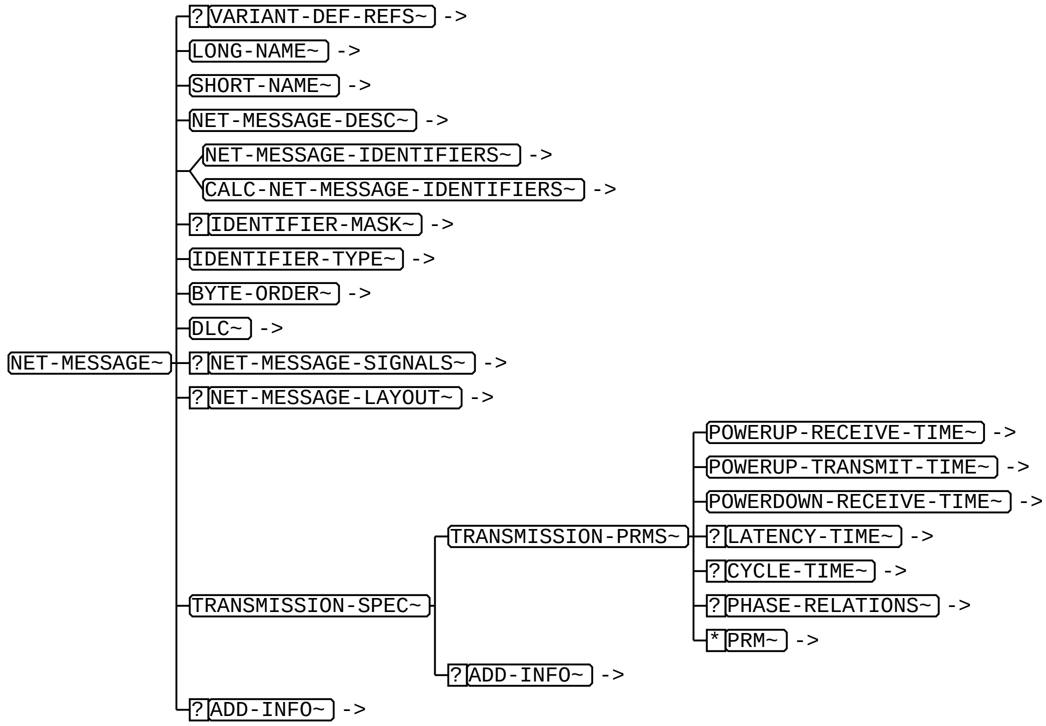


Figure 14: Structure of a single net message



Each message <**net-message**> can be specified variant-specific. The description consists of:

<**net-message-desc**> The message can be described here in prose.

<**net-message-identifiers**> or <**calc-net-message-identifiers**> Each message is identified by an identifier which is either specified either by <**identifier**> within <**net-message-identifier**> or which can be calculated (see.<**calc-net-message-identifiers**>).

Each message can have several identifiers depending on their sender identified by <**net-node-ref**>. In order to enable variant specific sender/id-combinations, it is possible to specify more than one sender together with the according variants (<**variant-def-ref**>).

The identifier is a hex value for identification of the message in the network.

<**calc-net-message-identifiers**> Two mechanisms are available for determining the identifiers of a specific <**net-message**> (for a network node):

- Discrete information for the identifiers <**net-message-identifiers**>.
- A computation of the identifier (<**calc-net-message-identifiers**>) as the sum of a basis and <**identifier-base-address**> and an offset(<**net-identifier-offset**>) given in <**net-node-variant**>. The <**msg-identifier-offset**> is given by means of the respective <**sender**>.

This allows to describe structurally identical (abstract) messages. Such messages can only then be transmitted on the bus when they have received a unique assignment to an identifier.[13]

<**address-mode**> The address mode can be ”*STD*” (Standard) or ”*XTD*” (Extended).

A more appropriate label is ”CAN data transmission format”. This is a very global definition for a bus. In the case of ”*Extended-Format*”, it is possible to define for each *control unit*/CAN controller, whether it is ”*active*” or ”*passive*”.

<**byte-order**> Specifies the byte order in the message. Possible values are: *motorola*, *motorola forwards*, *motorola backwards* or *intel*.

<**dlc**> The information DLC (Data Length Code) (refer to DLC (p. **??**)) is given in bytes and indicates the length of the message. A message can include gaps and hence the length cannot be determined from the sum of the signal lengths.

<**net-message-signals**> The signals of a message are listed in <**net-message-signals**> as <**net-message-signal**>. This information is optional. This is useful for the definition of block modes without <**net-signal**> specification. The usage of <**net-signal**>s within <**net-message**>s must be specified by linking the net-signal with <**net-signal-ref**> and giving an offset. Optional receivers can be linked by <**net-node-ref**>s.

Multiplexed signals can also be supported. These are specified as <**multiplex-signal-set**> which comprises of a <**multiplexor**> and the associated signals (<**multiplex-signal-list**>). The multiplexor is specified by an <**offset**> [14] and its length (<**bitsize**>). The <**multiplex-signal-list**> contains <**multiplex-entry**> which is used to associate a <**multiplex-value**> with a signal list (<**net-message-signals**>). Any number of interleaved multiplex signals can be established in this way[15].

The associated receivers <**receivers**> shall also be documented for each signal of a message (direct or multiplexed) since these can also depend on the message.

<**net-message-layout**> The information <**net-message-layout**> is foreseen as an optional, redundant supplement to the signal list (<**net-message-signals**>). Preprocessed information can be stored here that cannot be derived by the *SGML formatter* from the <**net-message-signals**>. Valid is always the information given in <**net-message-signals**>.

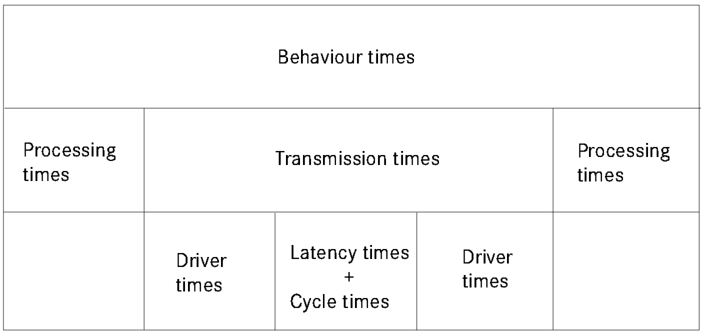
<**sender**> Designates the node which submits the signal by <**net-node-ref**>.

<**transmission-spec**> Describes the transmission procedure for the message.

An rough image in the form of a layer model (similar to the ISO/OSI reference model for communication) is provided by the following illustration:

13. Such messages are e.g. command messages by which a certain function within a control unit can be triggered from different nodes. Further examples are network management and diagnostic messages.
14. The offsets refer principally to the beginning of the message and are summed (i.e. beginning at 0)
15. This hierarchy is possible as a variant-dependent structure at all levels. Variant-dependencies should however only be introduced at the highest level.

Figure 15: Layer model for times/transmission times



The following parameters apply

<powerup-receive-time> Readiness to receive following power-up (p. **??**) Readiness to receive following power-up

Transmission parameter for a message that describes the point in time following *power-up* as of which the message can be received.

<powerup-transmit-time> Readiness to transmit following power-up (p. **??**) Readiness to receive following power-down

Transmission parameter for a message that describes the point in time following *power-up* as of which a message can be sent.

<powerdown-receive-time> Readiness to receive following power-down (p. **??**) Readiness to receive following power-down.

Transmission parameter for a message that describes the time after a *power-down message* during which a reaction to incoming messages is possible even though sending is not possible. It can be prevented by this that a component goes into the sleep mode just when a message is en route to the component.

<latency-time> Latency time (p. **??**) Latency time

Optional transmission parameter for a message that describes the transmission time in the network. The latency time is the time by which the sending process can be delayed by messages having a higher priority. In technical terms, the latency time is period of time between setting the TransmitRequestBit and receiving the AckMessage. The maximum latency time for this message can thus be specified for this message[16].

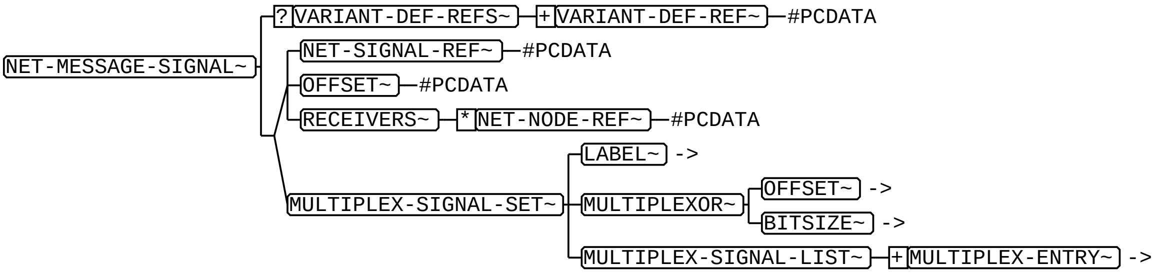
<cycle-time> Cycle time (p. **??**) Cycle time (also cycle tolerance time), optional transmission time for a message that describes the tolerance time for a message for a cyclic control system (cycle sending of messages)[17].

<phase-relations> Phase relation to other messages (p. **??**) Phase relations to other messages.

16. Only of relevance for cyclic control systems, hence optional.
17. Only of relevance for cyclic control systems, hence optional.

Transmission parameter for a message that describes whether there is a phase relation with other messages present.

Figure 16: Structure of signals in a message



4 Basic Structures of the MSR Application Profile

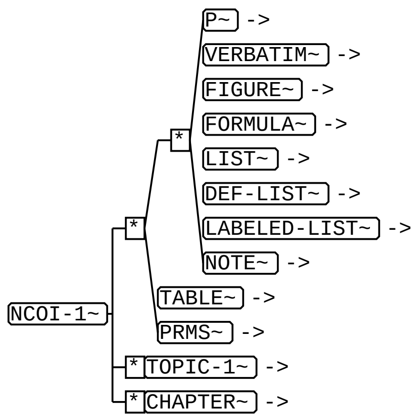
All MSR DTDs are using some common data structures. These operating models are described in this chapter.

4.1 Not Content Orientated Information (ncoi)

<**ncoi-1**> contains all basic descriptive elements. There are also elements like <**chapter**> or <**fail-save-concept**> in the *MSRSYS DTD* which have the same content model as <**ncoi-1**>.

The figure below illustrates the structure of <**ncoi-1**>.

Figure 17: Structure of <ncoi-1>



There also are two weaker ncoi models (*ncoi-2* and *ncoi-3*) with lesser elements than <**ncoi-1**>. *ncoi-2* has no <**chapters**>. *ncoi-3* has also no chapters and futhermore another "topic" model without <**prms**>.

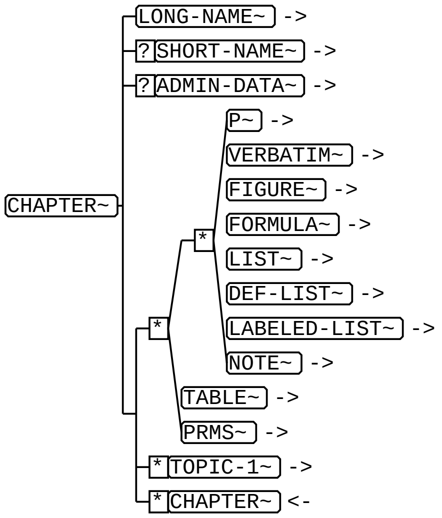
The components of ncoi [18] are interchangeable between all MSR DTDs[19] without any changes.

4.1.1 Chapter

<**chapter**> is a sequence of paragraph level elements mixed with <**chapter**>. <**chapter**>s can be nested as deeply as required. It is up to the author to make sure, that the nesting of the chapters can be handled by the processing system[20].

18. not content orientated information
19. DTDs with the same version of the MSR application profile
20. In that respect, it is recommended to nest not more than four levels.

Figure 18: chapter content model

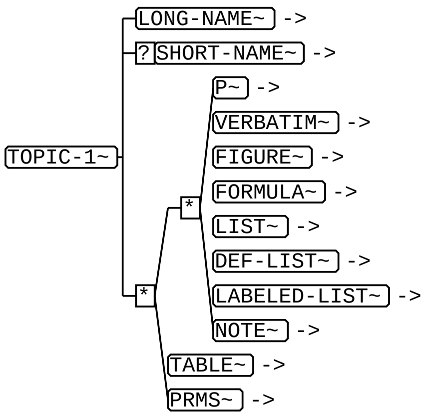


One advantage of using <**chapter**> for all levels[21] is the option to move a chapter using *cut & paste* to any place in the document at any level.

4.1.2 Topic

Use <**topic-1**> or <**topic-2**> to create bridge titles instead of one line paragraphs with entirely emphasized contents. Note that these elements can be referenced by <**xref**>. In difference to <**topic-1**>, <**topic-2**> has no <**prms**>.

Figure 19: Structure of <topic-1>



21. the other option would be to have an extra element for each chapter level

4.1.3 Paragraph Level Elements

"Paragraph level elements" are elements which occur on the same level as <**p**>.

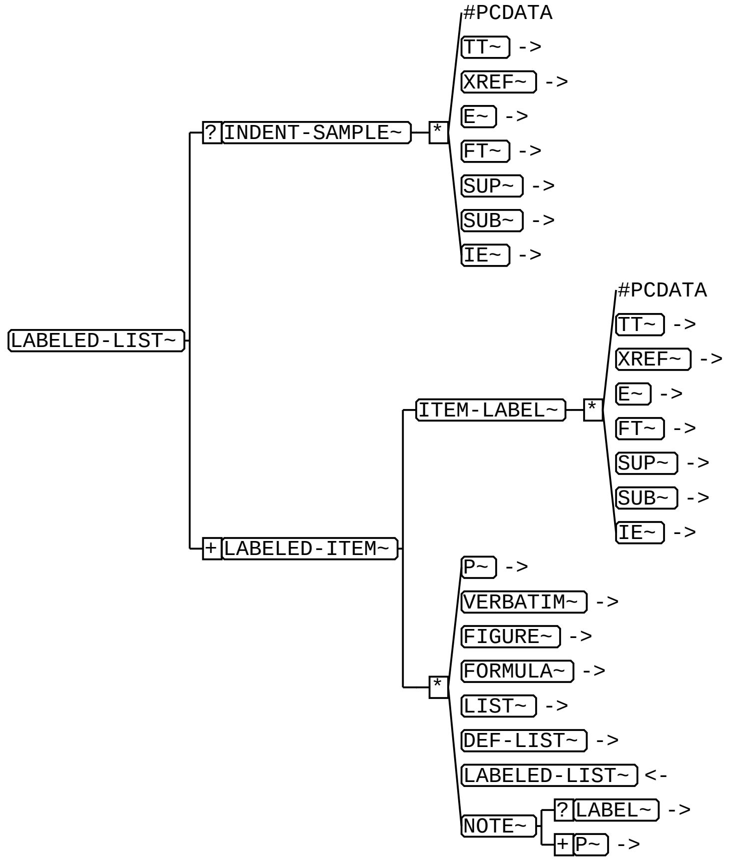
The user should first look for an appropriate one among the available elements before trying to simulate things by using inadequate elements. In that respect the following hints are given:

<**p**>	Paragraph
<**verbatim**>	Preformatted text which is usually set in monospaced font. Tabs, line spaces and carrige returns are considered. Use <**verbatim**> to print program listings etc. It can even be used to show simple diagrams.
<**figure**>	See chapter Figure (p. 43) .
<**formula**>	See chapter Formula (p. 44) .
<**list**>	A ordered or unordered list of items. For an unordered set of items, use <**list type**="**unnumbered**">. For a ordered list of items use <**list type**="**numbered**">[22].
<**def-list**>	Use <**def-list**> to create definition lists which might be collected into an overall definition list or a glossary. In this case <**labeled-list**> might lead to the same rendition but has no information about the fact that terms are defined[23].
<**labeled-list**>	Use<**labeled-list**> to create explanations or even bridge titles for very short topics instead of bulleted lists with emphasized initial words. See also Labeled List (p. 41) Use <**labeled-list**> instead of two column tables if the first column cells almost contain one word.
<**note**>	See chapter Note (p. 44)

4.1.3.1 Labeled List

22. Actually it is up to the rendition system if the sequence is expressed as numbers or as letters.
23. it is not really easy to distinguish between these two elements. As a rule of thumb, one might say that the semantics of def-list is stronger than the one of labeled-list which is more layout oriented. The items of def-list can be referenced by xref which is not possible with the items of a labeled-list.

Figure 20: Structure of <labeled list>



<**labeled-list**> is one of the most powerful elements. If possible it is rendered as a label followed by the item body:

```
XX       XXXXXXXXXXXXXX
         XXXXXXXXXXXXXX

XXXXXX   XXXXXXXXXXXXXX
         XXXXXXXXXXXXXX

XXXXX    XXXXXXXXXXXXXX
         XXXXXXXXXXXXXX
```

The indentation is determined by the *rendition system* which should take into account the biggest <**item-label**>. Sometimes the author wants some influence to the indentation. For this respect <**indent-sample**> can receive any content which is used by the *rendition system* as a sample which must be rendered and measured to determine the indentation. The attribute **[item-label-pos]** defines how the <**item-label**> should be handled. The default value of the attribute is **[item-label-pos]**=”no-newline”. If an <**item-label**> is wider than <**ident-sample**> the most general case is to start the item body in a new line if necessary(**[item-label-pos]**=”newline-if-necessary”):

XXXXXXXXXX

XXXXXXXXXXXXXX
XXXXXXXXXXXXXX

XXXXXX XXXXXXXXXXXXXX
XXXXXXXXXXXXXX

XXXXXX XXXXXXXXXXXXXX
XXXXXXXXXXXXXX

If the attribute has the value **[item-label-pos]**=”newline” the item-body starts generally in a new line.
Note that <**indent-sample**> can be used to adjust the indentation if there are multiple <**labeled-list**>s which should have the same indentation.

4.1.3.2 Figure

<**figure**> is used to insert graphics into the document. A figure can be defined in three different ways.

1. as a real <**graphic**>
2. as an ASCII graphic (<**verbatim**>)
3. as a pure textual description (<**desc**>) of the graphic [24]

The treatment of the graphic is determined by the attributes of <**graphic**>:

Do not enter annotating text to <**long-name**>in<**figure**> or <**table**>(like *Figure 1: ...*). This embellishment is the task of the processing system, not of the author. If the author adds these things, they will be there twice since the *rendition system* will add it again.

[category] Denotes the category of the graphic. This information can be used to generate more specific list of figures

[filename] Denotes the system filename where the *rendition system* can find the graphic. This is not necessarily the final format. It is up to the *rendition system* to locate the graphic in the company specific environment, to change the file extension to get the appropriate graphic representation.

The type of this attribute can be turned from SDATA toENTITY in the DTD file in order to allow *SGML tool*s access to the file using its *entity manager*. In this case, the entity name should be chosen in the style of a filename (e.g. crpctmt.wmf)[25].

[fit]

0 figure is placed in original size. If it does not fit on the page or the available space, it is scaled down.

1 the figure is scaled up or down to fit the page as possible. This value will be ignored if **[width]** or **[height]** is specified in addition.

2 the figure is rotated counterclockwise by 90° if it is landscape and is wider than the actual text area. It is scaled down to the page size if it does not fit otherwise. This value will be ignored if **[width]** or **[height]** is specified in addition.

3 the figure is always rotated counterclockwise by 90°. If it does not fit on the page it will be scaled down. If **[width]** or **[height]** is specified in addition, the figure will be rotated and then scaled to the specified values.

24. This can be used for systems which can’t display graphics.
25. This is the way how this document is prepared. It is visible in the sgml source.

4 the figure is always rotated counterclockwise by 90° and scaled up or down for best fit on the page. This value will be ignored if **[width]** or **[height]** is specified in addition.

[height] If this attribute has a value, the figure will be scaled to the defined height which is a real value with dimensions (e.g. ”10cm”, ”150mm”, ”12.5in”). If also **[width]**is specified the figure will be distorted. This value always specifies the width of the ”figure box” on the page after possible scaling/rotating.

[notation] This attribute specifies the format of the graphic file if used by an *SGML Application* supporting notations.

[scale] If this attribute receives a value, the figure will be scaled by the given factor which must be a signed real number. Numbers greater 1 increase the size of the figure, values less than 1 make the figure smaller. For example with scale=”0.5” the a figure of the size 10x10 cm will appear as 5×5cm.

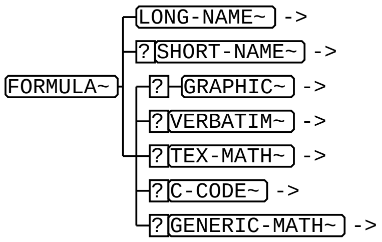
[width] If this attribute has a value, the figure will be scaled to the defined width which is a real value with dimensions (e.g. ”10cm”, ”150mm”, ”12.5in”). If also **[height]**is specified the figure will be distorted. This value always specifies the width of the ”figure box” on the page after possible scaling/rotating.

The scaling attribute precedence is:

- **[scale]** has precedence over all
- **[fit]**has precedence over **[width]**and/or **[height]**

4.1.3.3 Formula

Figure 21: Structure of <figure>



A formula can be described in five different ways which can exist parallel. These are:

<**graphic**> A formula prerendered as a figure.

<**verbatim**> A simple ASCII formula.

<**tex-math**> A *TeX* math formula which can be processed by a *TeX* or *LaTeX* processor.

<**c-code**> A formula which is defined as c-code.

<**generic-math**> This element is intended for the definition of semantic math descriptions which can be processed by math processors. Actually there is no recommentation for the language of the formula specification or usage of a special rendering system.

It is up to the rendering system which of the available representations is used.

4.1.3.4 Note

A note is an object to express a combination of an icon with descriptive text and an additional label. This is useful for things like cautions, hints etc..

The attribute **[notetype]** defines the note category. The following values are available:

- caution
- hint
- tip
- instruction
- exercise
- other

If the attribute **[notetype]** has a value of "other" the user has to specify a own type within the attribute **[user-defined-type]**.

A formatter has to place the right icon before the descriptive text according to the value of **[notetype]** or **[user-defined-type]**. The optional <**label** > can be used to define a title of the note.

4.1.4 Character Level Elements

Character level elements can occur within element like <**p**>, <**item-label**>. There are rendition oriented elements like <**e**> (emphasis), <**sub**> as well as semantically oriented Elements as <**tt**> (technical term) or <**std**>(referring to an external standard). It is highly recommended to use rather semantically oriented elements than rendition oriented ones.

4.1.4.1 Rendition Oriented Character Level Elements

The rendition oriented character level elements are:

<**e**> Emphasizes the text. The attribute **[type]**determines the rendition style.

<**sub**> Subscript - places the contents with smaller font below the base line.

<**sup**> Superscript - places the contents with smaller font above the base line.

4.1.4.2 Semantically Oriented Character Level Elements

Table 2: semantically oriented character level elements

Element	use for	example
<**tt**>	Use for any technical term. The type of that term is determined by the attribute **[type]**[26]. This element could be treated as a back-door to markup information which is not totally semantic. The *SGML processing system* can generate list of technical terms which makes it easier to find misspellings and other errors.	This is an SGML tag <**tt type**=**sgmltag**> we can collect all <**tt**>s
<**xref**>	Used to create links in the document. The role of the target is determined by the attribute **[id-class]** receiving the value of the target's fixed attribute **[f-id-class]**. The attributes of <**xref**> should be maintained by the *authoring system*.	
<**xdoc**>	Used to refer to an external document which usually is not available electronically. <**xdoc**> receives a set of elements characterizing the external document	Details to architectural forms can be found in *[External Document: / Relevant Position:]*.
<**ft**>	Is used to create footnotes	Footnotes seem to be small and unimportant[27].
<**ie**>	creates index entries	It is not necessary to put SGML tags into the Index, since the processing for *MSRREP.DTD* recommends to create a list of SGML tags automatically.
<**xfile**>	Is used to create pointers to external files which are not to be processed by the native *SGML processing system*. The contents of <**xfile**> can be used to connect to appropriate systems in later steps of the processing chain.	The schematic is found in *[External FILE: MOTRONIC wiring diagram / Filename: motronic.asc]*
<**std**>	Is used to refer to a standard.	SGML is defined in *[Standard: Information Processing - Text and Office Information Systems / Subtitle: Standard Generalized Markup Language / State: standard / Date: 1986 / Relevant Position: entire document]*

Table 3: usage of technical terms

type	use for	example
<**tt type**=**sgmltag**>	Used to describe SGML tags including attributes	To describe SGML tags use <**tt type**=**sgmltag**>.
<**tt type**=**sgml-attribute**>	Used to describe SGML attributes outside of tags	The sgmltag is denoted by the attribute **[type]**
<**tt type**=**tool**>	Used to mention tools used for example in a process. This can be software, as well as mechanical tools. The tool should be specified by its nature not by the specific product name.	SGML files are processed using an *SGML processing system*.
<**tt type**=**product**>	Used to mention specific products.	This document is processed using *MetaMorphosis*.
<**tt type**=**variable**>	Used to mention a variable informally. This is used to control the rendition as well as for generating variable lists. This is mainly for informal reports[28]. It is also possible to use this to mention a variable in the ECU software if no <**sw-data-dictionary**> is part of the document. In a later process step, this can be turned over to a formal <**xref**>	The initialization is controlled by the environment variable *MMRC*. The initial advanced angle is calculated based on *N*and*TL*.
<**tt type**=**state**>	Used to mention a state for example of a process.	The documents must at least be *revised* if they are submitted to the customer.
<**tt type**=**prm**>	Used to mention a state for example of a process. It is also possible to use this to mention a calibration parameter in the ECU software if no <**sw-data-dictionary**> is part of the document. In a later process step, this can be turned over to a formal <**xref**>	The initial advanced angle is calculated using a lookup table *KFZW*.
<**tt type**=**material**>	Used to mention material.	Furniture is usually made of *wood* and *plastic*
<**tt type**=**control-element**>	Used to mention control elements of tools like push-buttons, menu items, switches etc. as well as keyboard keys.	To finish the dialog push the *OK* button.
<**tt type**=**code**>	Used to markup program in line code sequences	*MetaMorphosis* is invoked with mm crp.sgm
<**tt type**=**organisation**>	Used to markup the name of an organization.	SGML is standardized by *ISO*

type	use for	example
<**tt type**=**other**>	Used to mention a special term which does not fit to the other types. This is a back-door for the definition of user defined types. They have to be specified within the attribute **[user-defined-type]**. A formatter uses this user defined type only if **[type**=**other]**.	This is a *thing* not covered by <**tt**>.

Table 4: sub-elements for xdoc and xfile

Element	use for	example
<**number**>	Used to markup the document ISBN resp. the standard number	ISBN 0-7923-9432-1
<**state**>	Used to markup the state of the referred document resp. standard.	released
<**date**>	Used to markup the release date of the referred document resp. standard. This could be expressed as year only, if the exact date is not known.	1994
<**publisher**>	Markup the publisher of the document or the standard. This can be the author as well as the publishing organization.	Steven J. DeRose and David G. Durand / Kluwer Academic Publishers
<**position**>	Markup the relevant position in the referenced document resp. standard.	Chapter 5.2 - Architectural forms
<**subtitle**>	Used to markup the subtitle of the referenced document or standard if there is one.	HyTime
<**short-name**>	Used to markup the document identifier	SGML
<**long-name**>	Used to markup the main title of the referenced object.	Making Hypermedia work
<**file**>	Used to markup the file access information. This is intended to be processed by external systems.	*[External FILE: MOTRONIC wiring diagram / Filename: motronic.asc]*

4.1.5 Table

<**table**> is implemented as *CALS table* (see *[External Document: CALS table spec / Relevant Position: all]* at www.oasis.org). Capturing these kind of tables must be supported by the *SGML editor*, so only some hints are given here:

- *CALS table*s consist of mainly three parts within <**tgroup**>: <**thead**>, <**tbody**>, <**tfoot**>.
- Each part is made of <**row**>s of <**entry**>s. Each of these elements have attributes to control the layout of the table.
- <**tgroup**> also receives a set of <**colspec**>s having information about the table columns.
- One of the major problems if *CALS table*s do not work is, that the amount of <**colspec**> elements and <**entry**> does not match the value of the attribute **[columns]** in <**tgroup**>.
- Within <**entry**> most of the paragraph level elements are allowed.

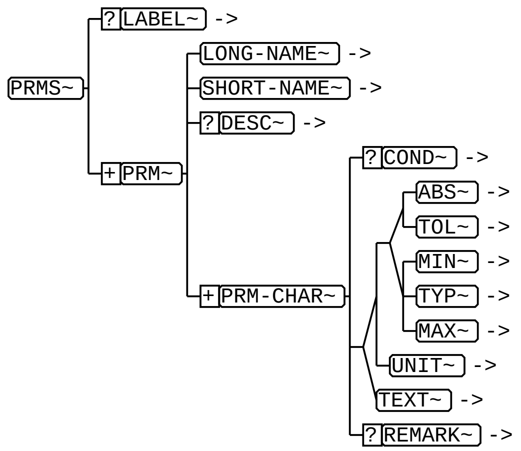
Note It is highly recommended to insert <**thead**>. This creates a table heading which is repeated on each page, if a pagebreak falls into the table.

4.1.6 Parameter tables

User Definable Parameters

For structured documentation of individual numerical and/or alpha-numerical requirements, so-called parameters are available. They have the following structure:

Figure 22: Structure of prms



× parameter × long-name

× short-name

× description

× parameter characteristics

× condition

((× absolute value and tolerance[29] or

× minimum, typical, maximum value[30])

× unit) or

× text[31]

The following representation example can be drawn from this structure:

<**short-name**> U_B

29. For definition of an exact setting or measurement value.
30. For definition of a typical value or value range.
31. For definition of an alpha-numerical value.

Table 5: Parameter structure

		<prm-char>						
Element: <**long-name**>	Element: <**short-name**>	Element: <**min**>	Element: <**typ**>	Element: <**max**>	Element: <**abs**>	Element: <**unit**>	Element: <**tol**>	
Operating voltage	U_B	10,8		14,2		V		
					13,5	V	5 %	
Colour of housing		red, green and blue						
Function state		active						

- Defined Parameters

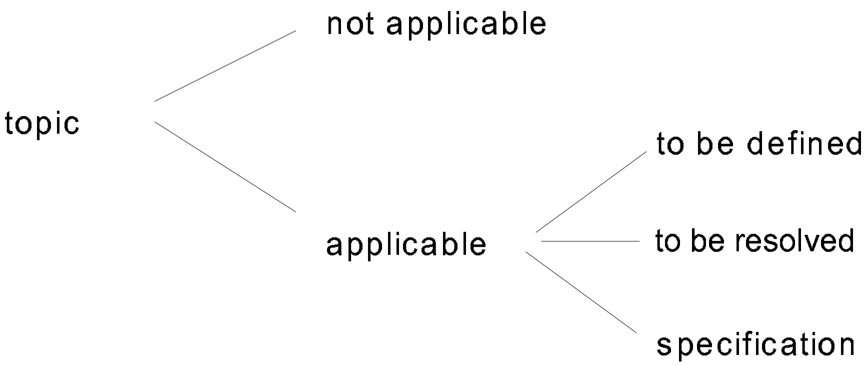
  There are many pre-defined parameters in the MSR DOC DTD. The only difference between them and user defined parameters is that the designation (long-name element) of the parameter is pre-defined.

4.2 Predefined Document Structure

The automotive systems to be described with the help of this DTD possess very different specifications. Because of this, the specification of a particular topic, e.g. ”acoustic characteristics” might not make sense or might only become necessary later on, depending on the project.

This situation was also taken into account in the DTD through the elements ”<**na**>” (not applicable), ”<**tbd**>” (to be defined) and ”<**tbr**>” (to be resolved) as shown in Principles of information acquisition (p. 50) . This is a mechanism is located at each element on chapter level and works like a check list. A user has to make a statement for each topic.

Figure 23: Principles of information acquisition



If a certain topic is not applicable it has to be marked with <**na**>. If it is applicable it can be marked with either with <**tbd**> which indicates that someone has to do a job, or it can be marked with <**tbr**> which indicates that a specification already exists but it hasn’t yet been included, or a detail specification can be defined.

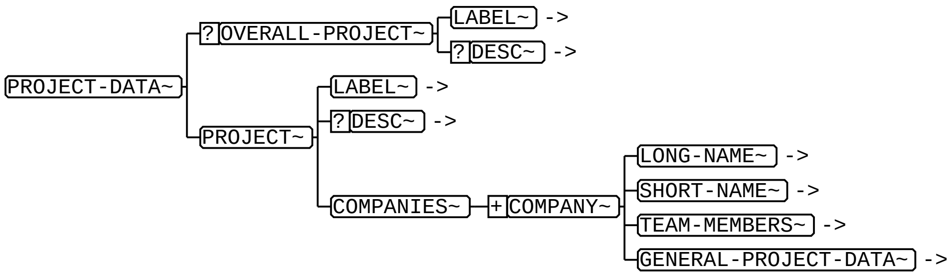
The elements <**na**>and<**tbr**> can be described with a short description. Within the element <**tbd**> the persons responsible for the definitions that have to been made can be specified with <**team-member-ref**>s. The schedule for the definitions can be defined within <**schedule**>.

4.3 Project Data

Registering and documenting development of a MSR system is project-oriented, whereby there may be several versions of the product data of a project. The projects can be combined with the help of main projects. This can be defined within <**overall-project**> by a <**label**> an a short description in <**desc**>. Each project is assigned to a maximum of one main project.

The documentation and continuation of project phases occurs in versions. We differentiate between active versions, the data of which can still be modified, and fixed versions, the data of which can no longer be modified. New versions can be designed on the basis of a fixed version. New versions can reuse complete fixed versions of a document or even parts of such a document. This is illustrated by the following figure:

Figure 24: Structure of <project-data>



Project data can be described by a PDM system in an integrated SGML-Editor and PDM environment. This is information on the current project and possibly the main project. Company-specific details about the project can be specified in <**general-project-data**> on the following items:

System overview <**system-overview**> This chapter can be used to define information about a global system, e.g. a certain car model.

Order justification <**reason-order**> This may be used to specify information about the reasons for the order of the described component resp. for making the specification of such a component.

Objectives <**objectives**> This chapter can be used to specify information about the project objectives. E.g. ”Development and system release of the engine-managment-system for the model NEW-BEETLE”

Models <**sample-spec**> This structure is used to define development samples like A-,B-,C-,D-sample. These samples represent the results of the different development phases.

Variant specification <**variant-spec**> This section is used to specify all variant definitions and their corresponding variant characteristics. See also Variant Concept (p. 53) .

Limits to other projects <**demarcation-other-projects**> This chapter is used to describe the demarcation to other projects.

Parallel developments <**parallel-design**> This can be used to give an overview of the work in parallel projects.

Integration capability <**integration-capability**> In this chapter requirements on the capabilities of integration in other systems can be described.

Acceptance conditions<**acceptance-cond**> This chapter is used to define the general conditions for the acceptance of the described components.

Schedule and plans <**project-schedule**> This chapter is used to define the project-schedule, e.g. project milestones, dates, time limits etc.

Purchasing conditions <**purchasing-cond**> This is used to define purchasing conditions like amount of devices per year, delivery times, storage quantities, etc. .

Protocols, minutes of meeting <**protocols**> This is the place where project minutes and other arrangements can be mentioned.

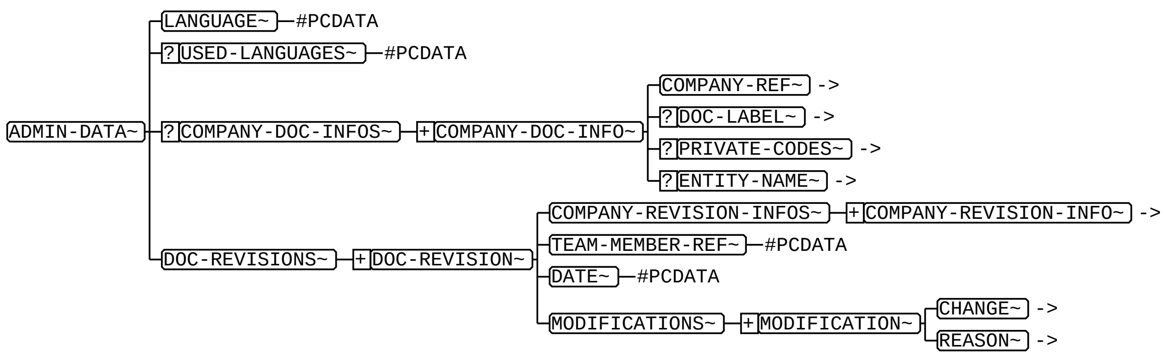
Handed over documents and data<**dir-hand-over-doc-data**> This is the directory of the handed over documents and data.

Additional project specifications<**add-spec**> Any kind of additional project description which can't be described with the chapters mentioned above.

4.4 Administrative Data

Since the respective companies explode the interchange DTD into fragments and use it for the respective acquisition DTDs (perhaps in different departments), the administrative dataappears in many places in the DTD. Each of these places can be used as such a fragment(see below).

Figure 25: Support of DTD fragmentation through administrative data



The operating model is

- The document respectively the fragment is written in a certain language which can be defined in the element <**language**>. This element can be used to control a SGML system, e.g. to set the correct prefix strings for elements.
- The DTD can be configured for the multilingual operation. In this case <**language**> contains the language of the origin document. All languages used in a document have to be defined within <**used-languages**>, that is each language is defined with a <**l-10**>-element which contains the full language name and in the Attribute **[l]** the short language name (see Multilinguality (p. 53)).
- The document (or the fragment) is handled in all companies participating in the project.

- The data management in the various companies is different. For that reason, each participant can enter information about their document management facilities in <**company-doc-info**>:

  <**doc-label**> this is the label under which the document is managed in the company denoted by <**company-ref**>

  <**private-code**> allows to transport company specific information in a private notation. This is the place, where for example *PDMS* (*Product Data Management System*s) can place pointers and document ids required to resynchronize after a document exchange.

  <**entity-name**> It might be the case that each participating company uses a different fragmentation strategy. In order to support this, <**entity-name**> can receive information useable by a *split utility* which creates the desired fragments out of the entire document.

- If a new release of the document or the fragment is given, each participating site may use a specific scheme for revision numbers. For that reason, each <**doc-revision**> can receive <**company-revision-info**> which holds the participant specific information for the actual document revision.

  It is up to a *semantical check utility* to keep sure that there is only one entry per company.

- nevertheless, the actual revision is initiated by one individual denoted by <**team-member-ref**> at one certain point of time denoted by <**date**>.

- Finally the modifications made in that revision are stored in <**modifcations**> where the actual <**change**> as well as the <**reason**> for that change is notified. If possible, the change can be located by <**xref**>.

- For each <**modifcation**> the attribute **[type]** determines, if the change is made to the document only (doc-related) or to the subject of the document (content-related).

4.5 Variant Concept

Especially in the automotive sector there is a multiplicity of different variants of a part type. Normally there is not only one variant documented in the system requirements respectively the product specification of such part types.

To understand the implementation of the variant concept in the MSR DTDs, first some definitions have to be made:

Variant Characteristic Characteristics that lead to a new variant e.g. engine, product line, country, etc. Characteristics are defined in <**variant-char**>. The characteristics have to be subdivided in three classes. These are:

- characteristics which lead to a new subject number(<variant-char **[type="new-part-number"]**>). For this only the existence of such a characteristic is enough to establish a new subject number for this variant!
- characteristics which don't lead to a new subject number (<variant-char **[type="no-new-part-number"]**>).
- characteristics which lead to a new subject number according to shaping.

Variant Definition: Definition of several variants with their variant characteristics for a part type.

Variant: A variant of a part type is defined through the values of it's variant characteristics.

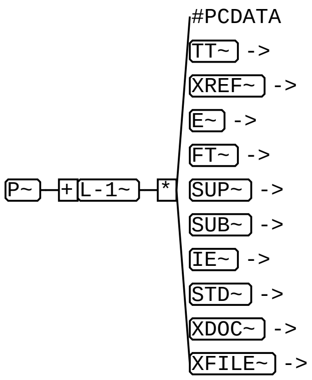
Variant Coding: Allocation of all variant definitions to their corresponding subject- and drawing- numbers and the respective development versions.

4.6 Multilinguality

The MSR DTDs can be configured for multilingual operation. To use the multilingual DTD configuration the DTD switch "multilinguality : YES or NO" have to be set.

The description of multilingual texts is made through multiple terminal elements that is multiple elements with content of #PCDATA. Multilingual elements get one of the additional language elements <**l1**>,<**l2**>, <**l3**>,<**l4**>,<**l10**> to build an aggregate of terminal elements. These language elements provide an attribute **[l]** where the language of this element can be specified. The content of the attribute **[l]** have to be defined as two-letter lower-case symbols according to the *[Standard: Code for the representation of names of languages / Relevant Position: Part1]*

Figure 26: Multilingual Paragraph



A Glossary

Sampling point Determines as a network parameter the point in time when the bit is sampled.

Address mode The address mode for a message can be "Standard" or "Extended".

ASAP The ASAP interfaces have been agreed by the "Study Group for the Standardization of Application Systems (ASAP)". Members of this study group are the German automobile manufacturers and companies in the supplier industry.

Basic CAN CAN Chip; possesses only one sender and one reception memory.

Baud rate Speed of the network.

Operating mode Transmission parameter for a message that describes for instance, whether remote operation is present or not.

Bit stuffing A technique that is used for resynchronization in bit-oriented network protocols. A change in the flank is caused by the inclusion of an additional complementary bit according to a defined number of bits of the same level. The additional bit is removed again by the receiver.

BRT Abbreviation for baud rate prescaler, defined as the number of BTL cycles.

BTL cycles Abbreviation for Bit Timing Logic cycle; a sampling frame is formed via the baud rate prescaler from the oscillator frequency of the chip. This auxiliary cycle (BTL cycle) serves to determine the time segments for the bit timing.

Byte order Network parameter that defines the sequence for the higher-value and the lower-value byte within a word. Since the processor manufacturers "Intel" and "Motorola" have made contrary definitions for the byte order, the values "Intel", "Motorola forwards" and "Motorola backwards" are possible for this parameter.

CAN Abbreviation for Controller Area Network

CAN High Speed Network link and network medium according to

Designation:	ISO Norm CAN High-Speed
State:	DIS
Relevant Position:	Entire document

, amongst others, characterized by a data rate of between 125 Kbit/s and 1 Mbit/s.

CAN Low Speed Network link and network medium according to

Designation:	ISO Norm CAN Low-Speed
State:	DIS
Relevant Position:	Entire document

, amongst others, characterized by a data rate of up to 125 Kbit/s

CAN conformance 2.0A CAN with an identifier length of 11 bits (standard).

CAN conformance 2.0B CAN with an identifier length of 29 bits (Extended). All CAN controllers that can process

CAN 2.0B telegrams are also capable of processing standard frame, i.e. to both send as well as to receive.

CAN conformance 2.0B passive CAN controllers that only process standard frames and that are passive in their response to CAN version 2.0B. These ignore extended telegrams according to version 2.0B.

CANdb CAN data base from Vector Informatik

CRC Abbreviation for Cyclic Redundancy Check; included the error code over all previous positions. The CRC checksum is used for fault detection yet not for fault corrections.

CSMA/CD Abbreviation for Carrier Sense Multiple Access with Collision Detect

DLC Abbreviation for Data Length Code; gives the length of the message in bytes.

Readiness to receive following power-down Transmission parameter for a message that describes the time following a *power- down message*, during which a reaction to incoming messages is possible even though sending is not possible. It is prevented by this that a component goes into the sleep mode when a message is en route to the component.

Readiness to receive following power-up Transmission parameter for a message that describes the point in time following power-up, as of when the message can be received.

Error value Value for a signal if a fault is detected and the actual information can no longer be transmitted.

Full CAN CAN chip that, unlike the basic CAN, includes several sending and receiving storage devices.

Initialization value Value of a signal before the first reception in the receiver.

Node A network node that includes any participant or branch of the network.

Latency time Transmission parameter for a message that describes the transmission time in the network. The latency time is the time by which the the sending process for a message can be delayed by messages having a higher priority. In technical terms, the latency time is the period of time between setting the TransmitRequestBit and receiving the AckMessage. The maximum latency time for this message can thus be specified in the message.

Multiplex-signal group Several signal groups can be grouped together in a multiplex-signal group that can be identified by a respective (group) value. A receiver of the message can filter out the signal group intended for the receiver from this value.

Network EMC design Description of the designs or set-up of the network with regard to electromagnetic compatibility.

net-signal group <**net-signal-group**>s contain those signals having exactly the same characteristics.

net-signal class The net-signal class classifies the net-signals according to application signals and network management signals.

OSEK To be defined

Phase relation to other messages Transmission parameter for a message that describes whether there is a phase relation with other messages present.

Physical range Permissible range for the physical (external) values for a signal that are given on account of the conversion from the network-internal presentation.

Quartz frequency The frequency of the quartz or quartz oscillator connected to the CAN controller.

RTR Abbreviation for Remote Transmission Request; a flag for identification of a data request. A sender is requested to transmit a corresponding data telegram as the response.

Transmission conditions Transmission parameter for a message that describes the transmission conditions.

Readiness to transmit following power-up Transmission parameter for a message that describes the point in time following *power-up* as of which the message can be sent.

SJW Abbreviation for Synchronization Jump Width that is given in BTL cycles. Defines the maximum amount by which the CAN controller may extend or shorten both major segments of the bit time.

Sleep/wake-up mechanism A mechanism that puts the network into a "*dormant state*" with low power consumption (sleep) and re-activates this as required (wake-up); e.g. for a central-locking system.

SLIO Abbreviation for Serial Linked I/O.

Memory layout For a message, the memory layout is a (graphical) supplement to the description of signal lists and multiplex-signal groups that describes the signal distribution of a message in the memory.

Conversion Procedure, on how to convert the network-internal presentation (internal values) of a signal in the corresponding physical values (external values).

Prescaler Or baud-rate prescaler; for generation of the BTL-cycle frequency from the quartz frequency.

Cycle time (Also cycle tolerance time), transmission parameter for a message that describes the tolerance time for a message for a cyclic control system (cycle sending of the message).

B Bibliography

Designation: CAN-Controller Area Network, Principles and Practice
Date: 1984
Publisher: Hüthing Publishers
Relevant Position: Entire book

Designation: CAN Controller Area Network, Principles, Protocols, Modules, Applications
Date: 1994
Publisher: Carl Hanser Publishers Munich Vienna, Munich
Relevant Position: Entire book

Literature-List

Standards

Designation: Code for the representation of names of languages
Relevant Position: Part1

Designation: ISO Norm CAN High-Speed
State: DIS
Relevant Position: Entire document

Designation: ISO Norm CAN Low-Speed
State: DIS
Relevant Position: Entire document

Designation: Information Processing - Text and Office Information Systems
Subtitle: Standard Generalized Markup Language
State: standard
Date: 1986
Relevant Position: entire document

External Documents

Designation: EMPTY title of external document
Relevant Position:

Designation: Structures Principles for MSR Software DTD
State: 1.1.0
Publisher: MSR AG-MEDOC
Relevant Position: Content model software

Designation: CALS table spec
Relevant Position: all

Designation: CAN Controller Area Network, Principles, Protocols, Modules, Applications
Date: 1994
Publisher: Carl Hanser Publishers Munich Vienna, Munich
Relevant Position: Entire book

Designation:	CAN-Controller Area Network, Principles and Practice
Date:	1984
Publisher:	Hüthing Publishers
Relevant Position:	Entire book

Designation:	Structure principles of the MSR application profile
Relevant Position:	all

Designation:	Structure principles of the MSR application profile
Relevant Position:	Linking

External FILES

Designation:	MOTRONIC wiring diagram
File:	motronic.asc

Designation:	MOTRONIC wiring diagram
File:	motronic.asc

Technical Terms

Code

ENTITY 43
SDATA 43
SGML-attribute 13
SGMLTAG 13
code 13
content-related 53
crpctmt.wmf 43
doc-related 53
mm crp.sgm 47
organization 13
other 13
product 13
scale=”0.5” 44
sgml-attribute 13

Control elements

OK 47

Material

plastic 47
wood 47

Organisations

ASAP 13
MEDOC Networks 11

OTHER

Application signal 33
CALS table 48, 48, 48
CAN_HIGH 26
CAN_LOW 26
CAN_SHIELD 26
ECU 13
Extended-Format 36
ISO 47
Network management signal 33
ncoi-2 39, 39
ncoi-3 39, 39
power- down message 56
power-down message 37
provisionnet-signal group 32
thing 48

Parameter

KFZW 47

Products

ABS control 16
ASAP2 13
CAN-controller 31
MSR NET DTD 31
MSRNET DTD 11, 15, 16, 18
MSRREP DTD 13
MSRREP.DTD 46
MSRSYS DTD 16, 18, 27, 27, 30, 39
MetaMorphosis 47, 47
Networks 10, 11
control unit 16, 16, 17, 17, 17, 17, 18, 18, 31, 36
engine control 16
vehicle system 16

SGML Attributes

[category] 43
[columns] 48
[f-id-class] 46
[filename] 43
[fit] 43, 44
[height] 43, 43, 43, 44, 44, 44, 44
[id-class] 46

[item-label-pos] 42, 42, 42, 43
[l] 52, 54, 54
[notation] 44
[notetype] 45, 45, 45
[scale] 44, 44
[type] 13, 13, 13, 13, 13, 13, 13, 45, 46, 47, 53
[type="new-part-number"] 53
[type="no-new-part-number"] 53
[type=other] 48
[user-defined-type] 45, 45, 48
[width] 43, 43, 43, 44, 44, 44, 44

SGML Elements

<**abs**> 31, 31, 31, 50
<**acceptance-cond**> 52
<**add-info**> 30, 31
<**add-spec**> 23, 25, 52
<**address-mode**> 36
<**admin-data**> 23
<**architecture**> 18
<**baudrate**> 31, 31
<**bitsize**> 33, 36
<**btl-cycles**> 31, 31
<**byte-order**> 36
<**c-code**> 44
<**calc-net-message-identifiers**> 35, 35, 35, 35
<**change**> 53
<**chapter**> 39, 39, 39, 39, 40
<**chapters**> 39
<**coded**> 33
<**colspec**> 48, 48
<**company-doc-info**> 53
<**company-ref**> 53
<**company-revision-info**> 53
<**connection**> 18
<**connection-comp-1**> 27
<**connection-comp-ref**> 27, 27
<**connection-comp-spec-1**> 25, 25
<**cycle-time**> 37
<**date**> 48, 53
<**def-list**> 41, 41
<**demarcation-other-projects**> 51
<**desc**> 43, 51
<**dir-hand-over-doc-data**> 52
<**dlc**> 36
<**doc-label**> 53
<**doc-revision**> 53
<**driver-concept**> 29, 31
<**e**> 45, 45
<**entity-name**> 53, 53
<**entry**> 48, 48, 48
<**error-value**> 33
<**error-values**> 33
<**fail-save-concept**> 39
<**figure**> 41, 43, 43
<**file**> 48
<**formula**> 41
<**ft**> 46
<**general-net-spec**> 23, 24
<**general-project-data**> 51
<**generic-math**> 44
<**graphic**> 43, 43, 44
<**ident-sample**> 42
<**identifier**> 35
<**identifier-base-address**> 35
<**ie**> 46
<**indent-sample**> 42, 43
<**init-value**> 33
<**integration-capability**> 51
<**interface-circuit**> 27, 27
<**item-label**> 42, 42, 42, 45
<**l-10**> 52
<**l1**> 54

<**l10**> 54
<**l2**> 54
<**l3**> 54
<**l4**> 54
<**label**> 51
<**label** > 45
<**labeled-list**> 41, 41, 41, 41, 42, 43
<**language**> 52, 52
<**latency-time**> 37
<**list**> 41
<**list type=”numbered”**> 41
<**list type=”unnumbered”**> 41
<**long-name**> 23, 26, 27, 43, 48, 50
<**max**> 50
<**min**> 50
<**modifcation**> 53
<**modifcations**> 53
<**msg-identifier-offset**> 35
<**msrsw**> 13
<**multiplex-entry**> 36
<**multiplex-signal-list**> 36, 36
<**multiplex-signal-set**> 36
<**multiplex-value**> 36
<**multiplexor**> 36
<**na**> 50, 50, 50
<**ncoi-1**> 39, 39, 39, 39
<**net-architecture**> 23, 25
<**net-block-modes**> 32
<**net-diag-spec**> 32
<**net-emc-design**> 30, 31
<**net-error-handling**> 32
<**net-identifier-offset**> 35
<**net-init-spec**> 32
<**net-interface-prms**> 29, 30
<**net-interface-spec**> 25, 29
<**net-line**> 18, 18, 25, 26, 26, 26
<**net-line-desc**> 26
<**net-lines**> 26
<**net-message**> 32, 35, 35, 36
<**net-message-desc**> 35
<**net-message-identifier**> 35
<**net-message-identifiers**> 35, 35
<**net-message-layout**> 36, 36
<**net-message-set**> 32, 34
<**net-message-signal**> 36
<**net-message-signals**> 36, 36, 36, 36, 36, 36
<**net-message-spec**> 32, 34
<**net-mgmt-spec**> 32
<**net-node-port**> 27, 27, 27
<**net-node-port-ref**> 27
<**net-node-ref**> 35, 36, 36
<**net-node-variant**> 35
<**net-nodes**> 25
<**net-oper-spec**> 23, 31
<**net-port**> 18, 18, 18, 18
<**net-port-ref**> 27
<**net-signal**> 32, 36, 36
<**net-signal-class**> 33, 33, 33
<**net-signal-group**> 33, 56
<**net-signal-ref**> 36
<**net-signal-spec**> 32, 33
<**net-signal-spec-variant**> 33
<**net-topology-spec**> 25, 25
<**network**> 18, 18, 18
<**network-spec**> 18, 18, 18
<**node-type**> 27
<**note**> 41
<**number**> 48
<**objectives**> 51
<**offset**> 36
<**overall-project**> 51
<**p**> 41, 41, 45
<**parallel-design**> 51

<**part-ref**> 27
<**part-type-spec**> 30
<**phase-relations**> 37
<**phys**> 33
<**port**> 18
<**position**> 48
<**powerdown-receive-time**> 37
<**powerup-receive-time**> 37
<**powerup-transmit-time**> 37
<**private-code**> 53
<**prm-char**> 50
<**prms**> 39, 40
<**project-data**> 23
<**project-schedule**> 52
<**protocols**> 52
<**publisher**> 48
<**purchasing-cond**> 52
<**reason**> 53
<**reason-order**> 51
<**receivers**> 36
<**row**> 48
<**sample-point**> 31, 31
<**sample-rate**> 31
<**sample-spec**> 51
<**schedule**> 50
<**segment**> 27
<**segment-end-nodes**> 27
<**segment-length**> 27, 27
<**segmentation**> 25
<**segmentation-desc**> 27
<**segments**> 27
<**sender**> 35, 36
<**short-name**> 23, 26, 27, 48, 49, 50
<**signal**> 18, 18, 18, 18, 18
<**signal-class**> 18
<**sjw**> 31, 31
<**state**> 48
<**std**> 45, 46
<**sub**> 45, 45
<**subtitle**> 48
<**sup**> 45
<**sw-compu-method-ref**> 34
<**sw-compu-methods**> 33
<**sw-data-dictionary**> 47, 47
<**sw-limits**> 33
<**sw-units**> 33
<**sync-edge**> 31
<**system-overview**> 51
<**table**> 43, 48
<**tbd**> 50, 50, 50
<**tbody**> 48
<**tbr**> 50, 50, 50
<**team-member-ref**> 50, 53
<**tex-math**> 44
<**text**> 31
<**tfoot**> 48
<**tgroup**> 48, 48, 48
<**thead**> 48, 49
<**tol**> 31, 31, 31, 50
<**topic-1**> 40, 40
<**topic-2**> 40, 40
<**topology-type**> 25
<**transmission-spec**> 36
<**tt**> 45, 46, 46, 48
<**tt type**=**code**> 47
<**tt type**=**control-element**> 47
<**tt type**=**material**> 47
<**tt type**=**organisation**> 47
<**tt type**=**other**> 48
<**tt type**=**prm**> 47
<**tt type**=**product**> 47
<**tt type**=**sgml-attribute**> 47
<**tt type**=**sgmltag**> 46, 47, 47
<**tt type**=**state**> 47

<tt type=tool> 47
<tt type=variable> 47
<typ> 50
<unit> 50
<used-languages> 52
<variant-char> 53
<variant-def-ref> 35
<variant-def-refs> 27
<variant-spec> 51
<verbatim> 41, 41, 43, 44
<xdoc> 46, 46
<xfile> 46, 46
<xref> 32, 33, 40, 46, 46, 47, 47, 53

State

STD 36
XTD 36
active 36
auxiliary nodes 27
dormant state 57
intel 36
motorola 36
motorola backwards 36
motorola forwards 36
participants 27
passive 36
power-up 37, 37, 57
revised 47

Tools

LaTeX 44
MSRNET DTD 16, 19
MSRSYS DTD 19, 25, 26, 27
PDMS 53
Product Data Management System 53
SGML Application 44
SGML editor 48
SGML formatter 36
SGML processing system 46, 46, 47
SGML tool 43
TeX 44, 44
authoring system 46
cut & paste 40
entity manager 43
rendition system 42, 42, 43, 43, 43
semantical check utility 53
split utility 53

Variables

MMRC 47
N 47
Register occupation 31
TL 47

Index

”

”Information Acquisition” 50

A

administrative data 52

C

CALS table 48

D

Definable Parameters 49
Defined Parameters 50

I

index 46

M

multilinguality 53

N

node type 27

P

project data 51, 51
Project structure 51