OSEK/VDX

Binding Specification

Version 1.1

11th September 2000

What is OSEK/VDX?

OSEK/VDX is a joint project of the automotive industry. It aims at an industry standard for an open-ended architecture for distributed control units in vehicles.
A real-time operating system, software interfaces and functions for communication and network management tasks are thus jointly specified.

The term OSEK means ”Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug” (Open systems and the corresponding interfaces for automotive electronics). The term VDX means „Vehicle Distributed eXecutive“. The functionality of OSEK operating system was harmonised with VDX. For simplicity OSEK will be used instead of OSEK/VDX in the document.

OSEK/VDX partners
The following companies attended and contributed to the OSEK/VDX Technical Committee:

Accelerated Technology Inc.,
ACTIA,
Adam Opel AG,
AFT GmbH,
ATM Computer GmbH,
Blaupunkt,
BMW AG,
Borg Instruments GmbH,
Cambridge Consultants,
Continental Teves,
Cummins Engine Company,
DaimlerChrysler AG,
Delco Electronics,
Denso,
Epsilon GmbH,
ETAS GmbH
FIAT - Centro Ricerche,
FZI,
GM Europe GmbH,
Hella KG,
Hewlett Packard France,
Hitachi Micro Systems Europe Ltd.,
Hitex,
IBM Deutschland Entwicklung GmbH,
IIIT - University of Karlsruhe,
Infineon,
INRIA,
Integrated Systems Inc.,
IRISA,
LucasVarity,
Magneti Marelli,
Mecel,
Motorola,
National Semiconductor,
NEC Electronics GmbH,
NRTA,
Philips Car Systems,
Porsche AG,
PSA,
Renault,
Robert Bosch GmbH,
Sagem Electronic Division,
Siemens Automotive,
Softing GmbH,
ST Mircroelectronics,
Stenkil Systems AB,
Sysgo Real-Time Solutions GmbH,
TECSI,
Telelogic GmbH,
TEMIC,
Texas Instruments,
Thomson-CSF Detexis,
Trialog,
TRW Automotive Electronics,
UTA - United Technologies Automotive,
VDO Adolf Schindling GmbH,
Vector Informatik,
Visteon,
Volkswagen AG,
Volvo Car Corporation,
Wind River Systems,
3Soft GmbH.

Motivation

- High, recurring expenses in the development and variant management of non-application related aspects of control unit software.
- Incompatibility of control units made by different manufacturers due to different interfaces and protocols.

Goal

Support of the portability and reusability of the application software by:

- Specification of interfaces which are abstract and as application-independent as possible, in the following areas: real-time operating system, communication and network management.
- Specification of a user interface independent of hardware and network.
- Efficient design of architecture: The functionality shall be configurable and scaleable, to enable optimal adjustment of the architecture to the application in question.
- Verification of functionality and implementation of prototypes in selected pilot projects.

Advantages

- Clear savings in costs and development time.
- Enhanced quality of the software of control units of various companies.
- Standardised interfacing features for control units with different architectural designs.
- Sequenced utilisation of the intelligence (existing resources) distributed in the vehicle, to enhance the performance of the overall system without requiring additional hardware.
- Provides independence with regards to individual implementation, as the specification does not prescribe implementation aspects.

Scope of the document :

As the standardisation of requirements that are applicable to different OSEK/VDX specifications should not be replicated within the different specifications, this document is therefore set-up to collate all requirements that are owned by the different specifications. This document also provides an overall description of the OSEK/VDX specifications set.

This binding specification is therefore a 'normative' document and intention is laid to prevent a divergence of the OSEK/VDX specifications by giving the possibility to refer to a single binding document.

Table of Contents

List of Figures

List of Tables

1 General description (informative)

The OSEK/VDX specification set consists of 4 normative documents that define the requirements of an operating system (real-time executive for ECU's), communication features (data exchange within and between ECU's) and network management strategies (configuration determination and monitoring).

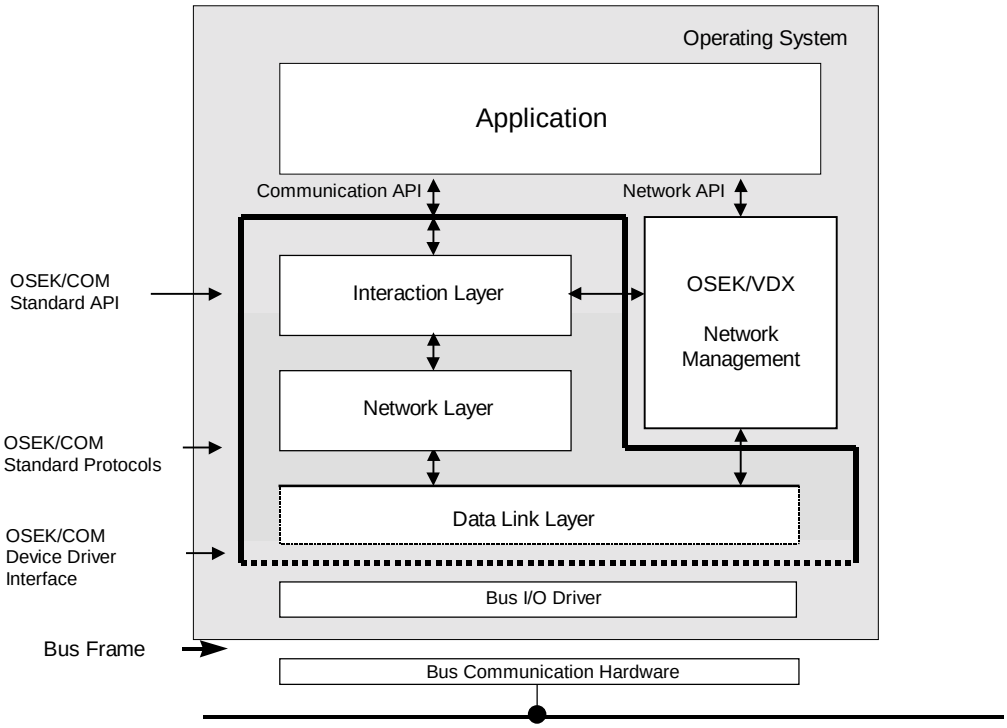

Figure 1-1: Layer model of OSEK/VDX

OSEK/VDX operating system (OS)

The specification of the OSEK/VDX OS provides a pool of services and processing mechanisms. The operating system serves as a basis for the controlled real-time execution of concurrent application and provides their environment on a processor. The architecture of the OSEK/VDX OS distinguishes three processing levels : interrupt level, a logical level for operating systems activities and task level. The interrupt level is assigned higher priorities than the task level. In addition to the management of the processing levels, operating system services are provided for functionality like task management, event management, resource management, counter, alarm and error treatment.

OSEK/VDX communication (COM)

The communication specification provides interfaces and protocols for the transfer of data within vehicle networks systems. This communication takes place between and within network stations (ECU's). The positioning of OSEK/VDX COM within the OSEK/VDX architecture is

represented in Figure 1-1. It shows the interface with the application, OSEK/VDX network management and the hardware communication bus. This specification is organised in layers, including the interaction layer, network layer and data link layer interface. The interaction layer provides the application programming interface (API) of OSEK/VDX COM to support the transfer of messages within and between network stations. For network communication, the interaction layer uses services provided by the lower layers. The network layer provides a protocol for the unacknowledged and segmented (USDT) transfer of application messages. The data link layer defines an open interface so that OSEK/VDX protocols can interface with different type of buses. ECU-internal communication is handled by the interaction layer only.

OSEK/VDX network management (NM)

Network serves as a basis for new distributed control functions that are independent of local ECU platforms. As a consequence of networking, the local station behaviour influences and depends on the global behaviour, and vice versa. The mutual influences and dependencies often require network wide negotiated management. In order to guarantee the reliability and safety of a distributed system, the OSEK/VDX NM gives support for several of such management tasks. The basic concept of OSEK/VDX NM is monitoring network stations. Two alternatives monitoring mechanisms are offered: direct and indirect station monitoring. Direct monitoring is performed by a dedicated communication of the network management. Direct monitoring of the nodes may be impossible or undesirable. This could be the case for example for very simple or time critical applications. The mechanism of indirect monitoring is therefore available. It is based on the use of monitored application messages. This indirect monitoring is limited to nodes that send messages in the course of normal operation.

OSEK/VDX Implementation Language (OIL)

To reach the original goal of the OSEK/VDX project in having portable software, a recommended way of describing an OSEK/VDX system has to be defined. This is the motivation for elaborating a standardised OSEK/VDX Implementation language, called OIL. In parallel, OIL provides a possibility to configure an OSEK/VDX application inside a particular central processing unit (CPU). As a consequence there is exactly one OIL description for each CPU.

2 Bindings configuration (normative)

Multiple bindings of the OSEK/VDX specifications are available, each reflecting development efforts resulting from OSEK/VDX evaluations and introduction of new requirements. Interfaces between the OSEK/VDX specifications are guaranteed within the following binding sets. The following tables list the issues of all OSEK/VDX specifications applicable to each particular binding reference. Two binding references are created to document on one hand the configuration of OSEK/VDX specifications (i.e. referred to as 'SB') and on the other hand the configuration of OSEK/VDX certification plans ((i.e. referred to as 'CB').

2.1 Binding index of OSEK/VDX specifications :

Specification binding identifier:	**SB0**	**SB1**	**SB2**	**SB3**
Binding	-	-	-	**1.0**
Operating System (OS)	**1.0**	**2.0r1**	2.0r1	**2.1**
Communication (COM)	**1.0**	**2.0a**	**2.1r1**	**2.2.1**
Network management (NM)	**1.0**	**2.1**	**2.5**	**2.5.1**
OSEK Implementation Language (OIL)	**-**	**2.0**	**2.1**	**2.2**

Table 2-1 : Binding index OSEK/VDX specifications

2.2 Binding index of OSEK/VDX certification plans :

Certification binding identifier:	**CB0**	**CB1**	**CB2**	**CB3**
Conformance methodology	**-**	**2.0**	2.0	**3.0**
OS test plan	**-**	**2.0**	2.0	**3.0**
COM test plan	**-**	**-**	**2.0**	**3.0**
NM test plan	**-**	**-**	**2.0**	**3.0**

Table 2-2 : Binding index OSEK certification plans

3 Common requirements specification (normative)

3.1 Definition of error codes

The different parts of OSEK/VDX (e.g. OSEK OS, OSEK COM) specify return codes of system functions to indicate different conditions which can arise during performing the system function. These return codes of type *StatusType* are defined as define-variables in the respective documentation. However, because these return codes can not be seen locally (e.g. they are used as input parameter to *ShutdownOS*), unique values have to be defined across the different specifications.

To accommodate this, ranges of error code values have been defined which are assigned to the different parts of the specification. Each range consists of 32 values. Within each range, the first up to 16 values are consecutively defined as standard return values. Starting with the second half of the range, the second 16 values may be defined consecutively to inform about detection of implementation specific additional errors (e.g. stack overflow, corruption of internal lists etc.).

Within the first range, the value ‘0’ (E_OK) has a special meaning. It indicates the successful completion of a system function without any specific return indication.

The ranges are assigned as follows:

0	E_OK
1 to 31	OSEK OS error codes
32 to 63	OSEK COM error codes
64 to 95	OSEK NM error codes
96 to 255	OSEK RESERVED

3.2 Definition of StatusType

The data type *StatusType* is used within all parts of OSEK/VDX. To be able to combine different parts of OSEK/VDX from different supplies (e.g. OSEK COM from supplier A, OSEK NM from supplier B), the definition of this type has to be handled with care to avoid conflicts.

Conflicts can arise if the definitions are different between the different parts of OSEK/VDX. Moreover, even if the definitions are the same, the compiler will have to create an error if the same type is defined more than once in one translation unit.

Therefore, the definition of *StatusType* and of the constant E_OK have to be done as follows in all parts of OSEK:

```
#ifndef  STATUSTYPEDEFINED
#define STATUSTYPEDEFINED
typedef unsigned char StatusType;
#define E_OK 0
#endif
```

These definitions have to be done in the header files supplied by the OSEK suppliers.

Please note that, if *StatusType* is not set to 'unsigned char', there is no guarantee that implementations of different OSEK parts by different suppliers will be able to coexist.

3.3 Support of '*internal communication*'

The definition of messages for *internal*, *external* and *internal-external communication* must be consistent and guaranteed. To cope with the situation that both kernels, i.e. COM and OS, are linked within a system, rules are set up clarifying which kernel handles *internal communication*.

If both COM and OS kernels are present but one of CCCA or CCCB only is to be supported (no application message use *external communication*) then the OSEK OS kernel shall provide the functionality to handle internal messages, i.e. those using *internal communication*.

If both COM and OS kernels are present but one of CCC0 onwards is to be supported to handle *external communication* in addition to internal communication then the OSEK COM kernel shall provide the functionality to handle internal messages, i.e. those using *internal communication*.

Thus, it is guaranteed that definitions of data types used within internal and external message handling are consistent within a system.

To internally assure that the stated rules are followed, a define-variable LOCALMESSAGESONLY is defined. *Internal communication* within OSEK OS must be implemented if this define-variable is set.

The define-variable LOCALMESSAGESONLY shall be defined by the tool which generates a system out of an OIL file. As long as the definition of messages in OIL has not been completed, other means of definition may be used.

4 History

Issue	Description	Date
1.0 c 1	Original issue candidate release 1.	21st July 2000
1.0	Original issue 1.0 from candidate release 1 with no requirement change.	28th July 2000
1.1	Replacement of COM 2.2 with COM 2.2.1 in section 2.1.	11th September 2000