INTERNATIONAL STANDARD

ISO 17356-1

First edition
2005-01-15

Road vehicles — Open interface for embedded automotive applications —

Part 1: General structure and terms, definitions and abbreviated terms

Véhicules routiers — Interface ouverte pour applications automobiles embarquées —

Partie 1: Structure générale et termes, définitions et termes abrégés

Reference number
ISO 17356-1:2005(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 17356-1 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 17356 consists of the following parts, under the general title *Road vehicles — Open interface for embedded automotive applications*:

— *Part 1: General structure and terms, definitions and abbreviated terms*

— *Part 2: OSEK/VDX specifications for binding OS,COM and NM*

— *Part 3: OSEK/VDX operating system (OS)*

— *Part 4: OSEK/VDX communication (COM)*

— *Part 5: OSEK/VDX network management (NM)*

— *Part 6: OSEK/VDX implementation language (OIL)*

Road vehicles — Open interface for embedded automotive applications —

Part 1: General structure and terms, definitions and abbreviated terms

1 Scope

This part of ISO 17356 outlines the general structure of, and defines terms and abbreviations used in relation to, the specification of the software open interface for embedded automotive applications given by the other parts of ISO 17356.

2 Terms, definitions and abbreviated terms

For the purposes of this document, the following terms, definitions and abbreviated terms apply.

2.1
acceptance filtering
mechanism which decides whether each received protocol frame is to be taken into account by the local node or ignored

2.2
activate
action of changing a task from the suspended to the ready state

NOTE The transition is achieved by a system service.

2.3
actual configuration
set of all operable nodes to which communication access is possible

NOTE See **operability of a node** (2.81).

2.4
address-related communication
type of communication between nodes using node addresses where each address-related communication message contains certain data and — either explicitly or implicitly — the node address of the transmitter and the receiver

NOTE 1 See **node addressing** (2.76).

NOTE 2 The communication of the network management is based completely on address-related communication.

2.5
alarm
association between a counter and a task, event or callback such that the task, event or callback occurs when a particular counter value is reached

NOTE 1 The expiry value can be defined relative to the current counter value or can be an absolute value.

NOTE 2 Alarms can be defined to be either single-shot or cyclic.

NOTE 3 An alarm is statically assigned at system generation time to one counter and a task, event or alarm callback routine.

2.6
alarm callback
short function, provided by the application, called when an alarm expires but before any task is activated or event set

2.7
alarm management
manipulation of an alarm’s running/cancelled state, and the counter value at which it next expires

NOTE Alarm management is based on the counter concept. Alarms are a way of linking alarm callbacks, task activation or event setting to counter values.

2.8
alive message
message used to announce an initialized and operable node for integration in the actual configuration

NOTE 1 See **operability of a node** (2.81).

NOTE 2 A dedicated NM message is used for this purpose.

2.9
application program interface
API
description of the application's interface to the operating system, communications and network management functions

2.10
application error
error where the operating system cannot execute the requested service correctly, but assumes the correctness of its internal data, and calls centralized error treatment

2.11
arbitration
mechanism that guarantees that a simultaneous access made by multiple stations results in contention where one frame will survive uncorrupted

2.12
basic conformance class
BCC
conformance class of the operating system in which only basic tasks are permitted

NOTE Two basic conformance classes are distinguished: BCC1 and BCC2.

2.13
basic task
BT
task that can only release the processor when it terminates, when the operating system executes a higher-priority task or when an interrupt occurs

NOTE 1 A basic task can only enter the task states suspended, ready and running.

NOTE 2 It is not possible for a basic task to wait for an event.

2.14
broadcast
case of multicast whereby a single message is addressed to all nodes simultaneously

2.15
busOff
condition of switching off from the bus so that protocol frames can neither be sent nor received

2.16
callout
general mechanism, based upon function calls, allowing the behaviour of the interaction layer to be customized and enhanced

NOTE 1 Callouts are configured statically, are invoked in response to the passage of a message or I-PDU and cannot be changed at run-time.

NOTE 2 The prototype for a callout allows it to return a value that determines further treatment by the IL of the message or I-PDU.

2.17
controller area network
CAN
protocol originally defined for use as a communication network for control application in vehicles

2.18
certification
process of determining whether an implementation is consistent with a given reference model

NOTE The scope of the reference model has to be settled according to the objectives of the project and all constraints necessary to fulfil those objectives incorporated in the reference model.

2.19
COM-callback
short function, provided by the application, which can be called by the interaction layer as a notification mechanism (class 1)

NOTE No parameters are passed to a COM-callback routine and it does not have a return value. A COM-callback routine runs either on interrupt level or on task level.

2.20
communication layer
set of all entities and elements which constitute a communication layer based on the ISO/OSI reference model

NOTE For the basic model, see ISO 7498-1.

2.21
configurability
ability to set the parameters of a system in terms of static values

EXAMPLE Number of tasks, RAM size for stack, size of message buffer.

2.22
confirmation
service primitive via which a service provider informs a service user about the result of a preceding service request

NOTE The confirmation service primitive is defined by the ISO/OSI reference model (ISO 7498).

2.23
conformance class
CC
subset of services chosen by the application

NOTE 1 In each module (operating system, communication, network management), a pool of services is provided, with each of these being divided into a number of defined subsets. Applications can choose to use a particular subset of the services in order to reduce demands on the CPU and memory.

NOTE 2 The subsets are upwardly compatible and are described as conformance classes.

2.24
connection
logical communication channel between a transmitter and a receiver

NOTE A message is sent by exactly one transmitter and is received by exactly one receiver.

2.25
constructional element
definition and declaration services for system objects

2.26
counter
system object that registers recurring events such as time or angle

NOTE A counter is represented by a count and some counter-specific constants.

2.27
critical section
sequence of instructions where mutual exclusion is ensured

NOTE Such a section is called “critical” because shared data is modified within it.

2.28
data consistency
content of a given message correlating unambiguously to the operation performed on the message by the application such that no unforeseen sequence of operations may alter the content and thereby render it inconsistent with respect to its allowed and expected value

2.29
data link layer
communication layer, consisting of the communication hardware and the communication driver software, that provides services for the transfer of I-PDUs

2.30
deadlock
tasks that block one another so that further processing of the tasks concerned is no longer possible

EXAMPLE Each of two tasks waits for the reception of a message to be sent by the other task before sending its own message.

2.31
direct node monitoring
active monitoring of a node by another node in the network

NOTE For this purpose, the monitored node sends an NM message according to a dedicated and uniform algorithm. For the network-wide synchronization of NM messages, a logical ring is used.

2.32
deadline monitoring
informing of the application via the notification mechanism that a message has not been received from another node within a specified interval, or if a request to send an I-PDU has not been completed by the DLL within a specified interval

2.33
error handling
error service provided to handle errors detected by the operating system

NOTE The basic framework is predefined and has to be completed by the user, thus giving the user a choice of efficient centralized or decentralized error handling.

2.34
error hook
routine (ErrorHook) called when a system service returns a StatusType value not equal to E_OK or when an error is detected during task activation or event setting

2.35
event
method of task synchronization peculiar to extended task whereby a task may suspend its execution without terminating

NOTE The task suspends its execution by waiting for an event and continues when an appropriate event is set. Basic tasks cannot use events.

2.36
event mechanism
means of task synchronization using events

2.37
extended conformance class
ECC
conformance class of the operating system in which basic and extended tasks are permitted

NOTE Two extended conformance classes are distinguished: ECC1 and ECC2.

2.38
extended task
ET
task that is allowed to use additional operating system services, which may result in a waiting state

NOTE An extended task can enter the task state *suspended*, *ready*, *running* or *waiting*.

2.39
fatal error
error where the operating system can no longer assume correctness of its internal data

NOTE In this case, the operating system calls the centralized system shutdown.

2.40
frame
data unit determined according to the data link protocol specifying the arrangement and meaning of bits or bit fields in the data transferred across the transfer medium

2.41
full pre-emptive scheduling
scheduling where a task which is presently running may be pre-empted at any instruction by the occurrence of a trigger condition pre-set by the operating system that puts the running task into the ready state as soon as a higher-priority task becomes ready

NOTE The pre-emptee's context is saved so that it can be continued at the location where it was pre-empted.

2.42
group addressing
addressing of several receiver nodes, implemented using multicast connections, in a single address-related NM message

NOTE See **address-related communication** (2.4).

2.43
hook routine
user-defined function which will be called by the operating system only under certain circumstances and in a defined context

NOTE Hook routines may be used for tracing or application-dependent debugging purposes, user-defined extensions to context switches and in error handling. Most operating system services are not allowed in hook routines.

2.44
indication
service primitive where a service provider informs a service user about the occurrence of either an internal event or a service request issued by another service user

NOTE The indication service primitive is defined by the ISO/OSI reference model (ISO 7498).

2.45
indirect node monitoring
monitoring of a node by “listening” to dedicated application communication messages

NOTE Indirect node monitoring is based on monitored state messages which are sent periodically.

2.46
interaction layer
communication layer that implements the interface between the application and other potential communication layers such as the DLL and network layers

NOTE 1 The communication services of the interaction layer are independent of both microcontroller and network protocol.

NOTE 2 The interaction layer enables internal and network-wide communication by means of UnQueued messages and Queued messages.

2.47
internal communication
exchange of messages between tasks belonging to the same node

2.48
internal resource
resource which is not visible to the user and therefore cannot be addressed by the system functions GetResource and ReleaseResource

NOTE Internal resources are managed strictly internally within a clearly defined set of system functions.

2.49
interrupt
enforced suspension of the execution of the current program section

2.50
interrupt latency
time between the moment an interrupt occurs and the execution of the first instruction of the interrupt service routine

2.51
interrupt level
priority level provided by the CPU for ISRs

NOTE To keep the interrupt latency as short as possible, it is preferable that only absolutely indispensable actions be performed at interrupt level.

2.52
interrupt service routine
function that provides the main processing of an interrupt

2.53
intertask communication
mode of information interchange between tasks

NOTE In the course of intertask communication, messages are logically copied from the local area of a task (transmitter) to the local area of another task (receiver).

2.53
I-PDU
collection of messages for simultaneous transfer between nodes in a network

NOTE At the sending node, the interaction layer (IL) is responsible for packing messages into an I-PDU and then sending it to the underlying layer (transport layer or DLL) for transmission; at the receiving node, the DLL (or transport layer) rebuilds the I-PDU and passes it to the IL, which then unpacks the messages.

2.54
ISR category
trade-off between ISR response time and API complexity

NOTE Interrupt processing is subdivided into two categories of ISRs: Category 1 comprises all ISRs which do not use operating system services and are, therefore, typically faster for entry and exit than category 2 ISRs. Category 1 ISRs are only allowed to use a very restricted set of operating system services, whereas category 2 ISRs are allowed to use a less restricted set but are typically inherently slower.

2.55
latency time
time delay between the request of an activity and its execution

2.56
limp home
operating mode in NM which is entered in case of an error which cannot be corrected

2.57
limp home configuration
set of all nodes which cannot participate in direct node monitoring due to failure

2.58
limp home message
dedicated NM message used for notifying a node that the system has entered the limp home state

2.59
logical ring
structure imposed by software rather than physical arrangement that orders the nodes within a network such that every node has exactly one successor and one predecessor and a pathway exists from any node to any other node

NOTE 1 The nodes are arranged in terms of a ring. The logical ring is used for the network-wide synchronization of NM messages. In a logical ring, the communication sequence is defined independently of the network structure; therefore, each node is assigned a logical successor — the first logical node is the successor of the last logical node in the ring.

NOTE 2 A ring message is always sent from a node to its logical successor.

2.60
message
fundamental unit of data transfer between an application and a COM's IL and, therefore, also of intra- and inter-ECU communications

NOTE A message can be 0 or more bits long and could contain some application-specific data ranging from a bit to a large array or structure. Therefore, messages can support event and signal-based communication as well as more complex interfaces.

2.61
mixed pre-emptive scheduling
scheduling policy which enables the use of both full-pre-emptive and non-pre-emptive scheduling policies for the execution of different tasks on the same system

NOTE The distinction is made via a task attribute (pre-emptable/non-pre-emptable).

2.62
multiple task requesting
property of a task that allows it to have more than one activation outstanding

NOTE 1 See **activate** (2.2).

NOTE 2 The operating system receives and records activation. On terminating the task (see *terminate*), the operating system checks whether any activation are outstanding. If there are any, the task immediately re-enters the running state.

2.63
mutual exclusion
prevention by one task of one or more other tasks running for a specified section of code

2.64
one–to–N connection
1:N connection
logical communication channel between a transmitter and N receivers

NOTE A message is sent by exactly one transmitter and is received by N receivers.

2.65
network configuration
set of nodes in the network

NOTE Within network management (NM), two configurations are distinguished: actual configuration and limp home configuration.

2.66
network management
NM
ensuring of the safety and availability of a communications network of autonomous control units

EXAMPLE 1 Initialization of the node and network-related (global) activities.

EXAMPLE 2 Co-ordination of global NM operating modes.

NOTE NM distinguishes between node-related (local) and network-related (global) activities.

2.67
NMBus-sleep
non-participation in NM communication

NOTE This is an NM operating mode. A request for this mode request must be confirmed by all nodes in the network.

2.68
NM callback
short function provided by the application which can be called by the interaction layer as a notification mechanism (class 1) so that NM can be informed of the network's state

NOTE 1 A parameter can be passed to an NM-callback routine and does not have a return value.

NOTE 2 An NM-callback routine runs either on interrupt level or on task level.

2.69
NM infrastructure
order structures and addressing mechanisms (window) which are accessed by the network management, including a communication infrastructure for the exchange of NM messages, so that each node is able to communicate with any other node on the network in a straightforward fashion

EXAMPLE Logical ring.

2.70
NMLimpHome
NM operating mode which is entered in case of an error which cannot be remedied

2.71
NM message
NMPDU exchanged between NM entities

NOTE The NM distinguishes between regular ring messages, alive messages and limp home messages.

2.72
NM operating mode
NM mode
specific behaviour of the NM

NOTE 1 The NM can enter different local operating (e.g. Nmoff) and global operating (e.g. sleep) modes. The transition to a different global operating mode requires a network-wide coordination, i.e. the local NM for each node has to enter the same global mode. Local operating modes only affect the local NM of a node and are transparent for all the other nodes.

NOTE 2 Operating modes of the application are not managed by the NM.

2.73
NM protocol data unit
NMPDU
NM message communicated between the sending and receiving NM entities containing an address field with source and destination address, a control field with an op-code and an optional data field with application-specific ring data

2.74
NMSleep mode
NM operating mode where the node does not participate in NM communication

NOTE The NM distinguishes between a local sleep mode and a global sleep mode. In both cases, the transition into the sleep mode is notified network-wide. The difference between the two is that a local sleep mode request must not be confirmed by the other nodes in the network, whereas a global sleep mode request must be confirmed by all nodes in the network.

2.75
node
network topological entity where one or more physical data links meet

NOTE Each node is separately addressable on the network.

2.76
node addressing
method by which each node on a network is uniquely identified so that it can be referred to by other members of the network

NOTE 1 Addresses are used to transmit address-related NM messages from one node to another.

NOTE 2 Individual node addressing is implemented using 1:1 connections. Several nodes can be addressed using group addressing.

2.77
non-pre-emptive scheduling
scheduling policy in which a task switch is only performed via one of a selection of explicitly defined system services (explicit rescheduling points)

2.78
non-pre-emptable task
task which can not be pre-empted by other tasks

NOTE 1 See **pre-empt** (2.86).

NOTE 2 Such a task only releases the processor at rescheduling points.

2.79
offline
state of the data link layer in which only NM communication is allowed

NOTE This implies that no application communication is possible.

2.80
online
normal state of the data link layer where both application and network management communication are possible

2.81
operability of a node
possibility of a node to participate in the NM's direct or indirect node monitoring

2.82
OSEKtime
operating system especially tailored to the needs of time-triggered architectures

2.83
OS processing level
processing level for the execution of operating system services

2.84
overrun
attempt to store data in memory beyond allocated capacity

EXAMPLE Exceeding the length of the queue in a queued message object.

2.85
PostTaskHook
system hook routine called upon leaving a task either due to pre-emption by another task or by termination

2.86
pre-empt
change in a task from running to ready state, imposed by the scheduler when it decides to switch to another task

NOTE In the case of a non-pre-emptive scheduling policy, pre-emption only occurs at explicit rescheduling points.

2.87
pre-emptable task
task which can be pre-empted by any task of higher priority

NOTE See **pre-empt** (2.86).

2.88
PreTaskHook
system hook routine called before entering or returning to a task

2.89
priority ceiling protocol
mechanism used to prevent deadlocks and priority inversion within the framework of resource management

2.90
protocol
formal set of conventions or rules governing the exchange of information between protocol entities comprising syntax and semantics of the protocol messages as well as the instructions on how to react to them

2.91
protocol entity
task or procedure for handling a protocol

2.92
queue
data storage area organized so that more than one datum can be inserted and data are removed in their order of insertion

2.93
queued message
message that appears in a queue

2.94
ready
state of being prepared to meet all functional prerequisites for a transition into the running state, waiting for allocation of the processor

NOTE The scheduler decides which ready task is executed next, the state is reached via the state transitions activate, release and pre-empt, and is exited by start.

2.95
re-entrant
function that can be called again during an interruption of its execution where both calls are executed correctly

2.96
rescheduling point
operating system calls which cause the activation of the scheduler

EXAMPLE Explicit call of the scheduler, successful termination of a task.

NOTE This exists in full-pre-emptive, mixed pre-emptive system and non-pre-emptive system.

2.97
regular ring message
normal NM message, containing network status information, that is also used to indicate a station logoff or local sleep mode, or to request for global sleep mode

NOTE See **NMSleep mode** (2.74).

2.98
release
change of a task from waiting to ready state caused by at least one event occurring for which the task has previously been waiting

2.99
reply message
dedicated NM message for replying to the reception of a request message

NOTE The reply message can be used by a slave of a logical star.

2.100
request
service primitive where a service user requests a service from a service provider

NOTE The service primitive is defined by the ISO/OSI reference model (ISO 7498).

2.101
request message
dedicated NM message for requesting the transmission of a reply message

NOTE This can be used by the master of a logical star.

2.102
resource
operating system providing abstraction to support task and ISR coordination by mutual exclusion in critical sections

NOTE 1 A task or ISR that locks a resource cannot be pre-empted or interrupted by any other task or ISR that is also declared as one that might also lock that resource.

NOTE 2 The assignment of resources to tasks and ISRs is performed at system generation time and cannot be changed by the application.

2.103
resource management
manipulation at run-time of resources either implicitly (in the case of internal resources) or via the get-and-release calls

2.104
response
service primitive used by a service user in order to reply to a preceding indication from service provider

NOTE The service primitive is defined by the ISO/OSI reference model (ISO 7498).

2.105
ring data
data sent, using an NMPDU's data field, and received by the application, with guaranteed data consistency

2.106
ring message
normal NM message containing the network status information and also used to indicate a node's local sleep mode or to request global sleep mode

NOTE See **NMSleep mode** (2.74).

2.107
running
task state assigned to the CPU so that its instructions can be executed

NOTE Only one task can be in the running state at any point in time. The state is entered by the state transition start and can be exited via the state transition wait, pre-empt or terminate.

2.108
scalability
setting of the scope of capabilities of a system as determined by its functionality

NOTE See **conformance class** (2.23)

2.109
scheduler
OS sub-system that decides whether a task switch should be put into the running state according to the selected scheduling policy

NOTE The scheduler can be considered to occupy a resource which can also be occupied by any task. Thus a task can block the scheduler to achieve arbitrary periods where it is the only task that can run.

2.110
scheduling policy
policy used by the scheduler to determine whether a task may or may not be pre-empted by another task

NOTE Three scheduling policies are distinguished: non-pre-emptive, full-pre-emptive and mixed-pre-emptive scheduling.

2.111
segmented communication
segmented data transfer
transfer of application data that cannot fit into a single protocol frame by the transmitter disassembling the data into segments that are small enough to fit into protocol frames, sending the segments and then the receiver reassembling the data

NOTE See **I-PDU** (2.53)

2.112
semaphore
means for the synchronization of access to shared data

NOTE See **resource management** (2.103)

2.113
severe error
error where the operating system could not achieve the requested service, but assumes the correctness of its internal data and calls centralized error treatment

NOTE Additionally, the operating system returns the error by the status information for decentralized error handling.

2.114
start
change of a task from ready to running state, thereby causing the task to be executed

2.115
startupHook
system hook routine called after the operating system start-up and before the scheduler is running

2.116
suspended
task state of being passive and not occupying any dynamic OS or memory resources

NOTE 1 A task in this state is on before it is entered, or can reach it via the status transition terminate.

NOTE 2 To exit the state, the task is activated.

2.117
system generation service
definition and directive which is necessary to set-up module at compile time

2.118
ShutdownHook
system hook routine called when a system shutdown is requested by the application or by the operating system

2.119
task
entity consisting of code and management information controlled by the scheduler

NOTE 1 The applications running on the ECU are composed of tasks and ISRs. Tasks can be executed concurrently with other tasks under the control of the scheduler according to the task priority assigned to it and the selected scheduling policy.

NOTE 2 From a programmer's point-of-view, a task is a C-like function.

NOTE 3 A distinction is made between basic tasks and extended tasks.

2.120
task level
processor's processing priority where most application software is executed

NOTE 1 Some application software is also executed in ISRs.

NOTE 2 Tasks are executed according to the priority assigned to them and the selected scheduling policy.

NOTE 3 Other processing levels are interrupt level and operating system level.

2.121
task management
manipulation of a task's state

NOTE Task management is achieved by activation [see **activate** (2.2)] and termination [see **terminate** (2.126)] of tasks, as well as by use of the scheduler.

2.122
task priority
measure of the precedence with which the task is to be executed

NOTE 1 Initial priorities are defined statically. However, as the application runs, tasks can change their priority [see **priority ceiling protocol** (2.89)].

NOTE 2 Depending upon the CC, more than one task may be declared at the same priority within a system.

NOTE 3 Tasks of equal priority are started according to the order in which they are activated.

2.123
task state
condition distinguishing a task by which it is either running, ready, waiting or suspended

NOTE 1 See **running** (2.107), **ready** (2.94), **waiting** (2.133) and **suspended** (2.116).

NOTE 2 Basic tasks cannot change to the state waiting.

NOTE 3 A task can only be in one state at any one time.

2.124
task switching time
time between the occurrence of the "task switch event" up to the execution of the first instruction of the "new" task

2.125
task switching mechanism
mechanism, managed by the scheduler, that performs a context switch to a selected task

2.126
terminate
change of a task from running to suspended state

NOTE 1 The running task causes its transition into the suspended state by means of a system service.

NOTE 2 A task can only terminate itself.

2.127
unacknowledged communication
unacknowledged data transfer
status under which the transmitter receives no data from the receiver confirming that the message has been received

2.128
unidirectional communication
data transfer mode characterized by data being exchanged in one direction only

2.129
unqueued message
message that is overwritten upon arrival of a new message, and where the application's read operation is non-destructive, thereby allowing the same message data to be read more than once

2.130
unsegmented communication
unsegmented data transfer
transfer of data that fits within a single bus frame

2.131
validation
ensuring of the correctness of a specification

2.132
wait
change of a task from running to waiting state, thereby causing the running task temporarily to suspend execution but without the loss of its context

2.133
waiting
task state in which the task is unable to continue execution owing to its awaiting at least one event

NOTE 1 Lower priority tasks can run while the task waits for an event to permit it to continue operation.

NOTE 2 Only extended tasks can enter the waiting state.

NOTE 3 Event reception causes the task to be released and therefore to make the transition into the ready state.

2.134
warning
return value that is not E_OK and is not equivalent to an error, giving complementary information related to a system service execution

API	application program interface
BCC	basic conformance class
BNF	Backus-Naur form
BT	basic task
CAN	controller area network
CC	conformance class
CCC	communication conformance class
COM	communication
CPU	central processing unit
DLL	data link layer
ECC	extended conformance class
ECU	electronic control unit
EPROM	erasable programmable read only memory
ET	extended task
FIFO	first in first out
IL	interaction layer
ISR	interrupt service routine
LIFO	last in first out
MSB	most significant bit
NM	network management
NMPDU	NM protocol data unit
OIL	OSEK implementation language
OS	operating system
OSEK	abbreviation of the German *Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug* — “open system and its corresponding interfaces for automotive electronics”
UML	unified modelling language
UUDT	unacknowledged unsegmented data transfer
VDX	vehicle distributed executive

3 Structure of ISO 17356

3.1 General

ISO 17356 consists of a set of specifications comprising six normative documents, including this part of ISO 17356. ISO 17356-3 to ISO 17356-5 define the requirements of an operating system OS (real-time executive for ECUs), communication features COM (data exchange within and between ECUs) and network management strategies NM (configuration determination and monitoring), while ISO 17356-2 and ISO 17356-6 describe how to proceed with implementation of the specification set, specifying the binding of the OS, COM and NM, and the OIL.

Figure 1 shows a interpretation of ISO 17356.

NOTE ISO 17356 does not follow all the OSI layers because they are not really necessary for automotive applications where the size of software must remain as small as possible.

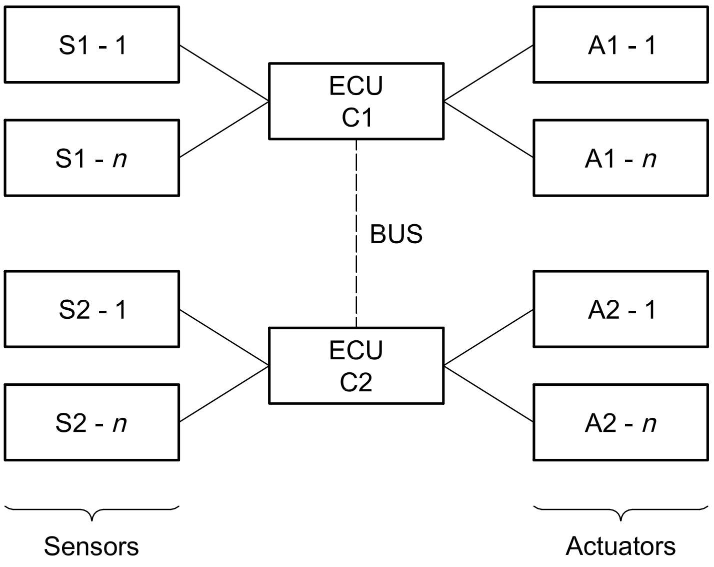

Figure 1 — ISO 17356 model

3.2 ISO 17356-2 — Binding

To avoid replication of requirements applicable to different parts of ISO 17356, Part 2 collates all requirements that are owned by the different specifications and creates a link between them.

3.3 ISO 17356-3 — OS

The specification of ISO 17356-3 provides a pool of services and processing mechanisms. The OS serves as a basis for the controlled, real-time execution of concurrent applications and provides their environment on a processor. The architecture of the ISO 17356-3 specification distinguishes three processing levels:

— interrupt level;

— level for operating systems activities;

— task level.

The interrupt level is assigned the highest priority. The task level on which the application software is executed owns the lowest priority. In addition to the management of the three processing levels, the operating system provides functions such as task management, event management, resource management, counter, alarms and error treatment.

3.4 ISO 17356-4 — COM

ISO 17356-4 specifies a uniform communication environment for automotive control unit application software. The COM specification increases the portability of application software modules by defining common software communication interfaces and behaviours for internal (within an ECU) and external (between networked vehicle nodes) communication, independent of the used communication protocol.

3.5 ISO 17356-5 — NM

NM as specified by ISO 17356-5 provides standardized features which ensure the functionality of inter-networking by standardized interfaces. Its essential task is to ensure the safety and reliability of a communication network for ECUs.

At a basic configuration stage, NM implementations complying with the ISO 17356-5 specifications shall be implemented in all networked nodes. This implies a solution for NM which can be implemented throughout the broad range of available hardware offered in today's ECUs.

ISO 17356-5 NM offers two alternative mechanisms for network monitoring:

— indirect monitoring by monitored application messages;

— direct monitoring by dedicated NM communication.

3.6 ISO 17356-6 — OIL

ISO 17356-6 OIL is a language that describes a system consisting of OS, COM and NM components by supporting the declaration and parameterization of system objects. In order to build the system, the OIL description is translated into data required by a particular implementation.

Annex A
(informative)

History and rationale of 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.

The automotive industry realized that it had to control the drastic increase of embedded ECUs used in road vehicles, especially for safety and security reasons, and that movement from a classical electronic architecture to an integrated one had become essential.

In a classical architecture, a system, S1, is carried out with sensors, S1*i*, one ECU C1 and actuators, A1*i*. Inside the ECU C1, the software drives the function of system S1. One function corresponds to several sensors and actuators and a single ECU with its own software. Whereas, in an integrated automotive electronic architecture, several functions with related actuators and sensors connected to it are to be found in a single ECU, while, owing to multiplex organization, the sensors' data are shared between several ECUs.

See Figure A.1.

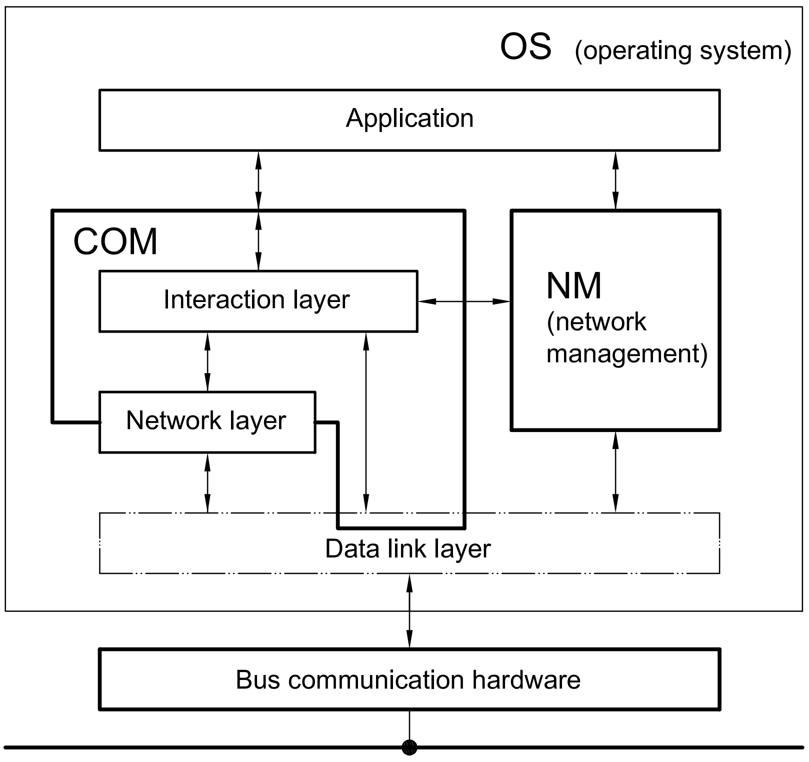

Figure A.1 — Typical automotive architecture

Most important is that because several functions can be found in the one ECU, and the same function located in several ECUs, the software becomes difficult to implement and to debug.

The European OSEK/VDX consortium was charged with the task of overcoming these difficulties. The publication of ISO 17356 is based on the results of the work done by the consortium since 1994. The rationale for its work is the following.

a) **Problems needing to be solved**

   1) High, recurring expenses in the development and variant management of non-application related aspects of control unit software.

   2) Incompatibility of control units made by different manufacturers due to different interfaces and protocols.

b) **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 areas of real-time operating system, communication and network management,

   — specification of a user interface independent of hardware and network,

   — efficient design of architecture, with a functionality that is configurable and scaleable, enabling optimal adjustment of the architecture to the application in question, and

   — verification of functionality and implementation.

c) **Advantages**

   — Clear savings in costs and development time.

   — Enhanced quality of the software of control units of various companies.

   — Standardized interfacing features for control units with different architectural designs.

   — Sequenced utilization 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

Bibliography

[1] ISO 7498 (all parts), *Information processing systems — Open Systems Interconnection*

ICS 43.040.15

Price based on 21 pages