OSEK/VDX

Networked Embedded Systems

Sebastian Koschmieder

Technical University of Berlin
Telecommunication Networks Group

June 5, 2015

Outline

1. Introduction
2. OSEK OS
3. OSEK COM
4. OSEK NM
5. OSEK OIL
6. OSEK TIME

Introduction

- OSEK/VDX is a norm, not only an OS
- means: "**O**ffene **S**ysteme und deren **S**chnittstellen für die **E**lektronik in **K**raftfahrzeugen"
- developed by several German automotive companies
- PSA joined later
- same system basis on every ECU

System Overview

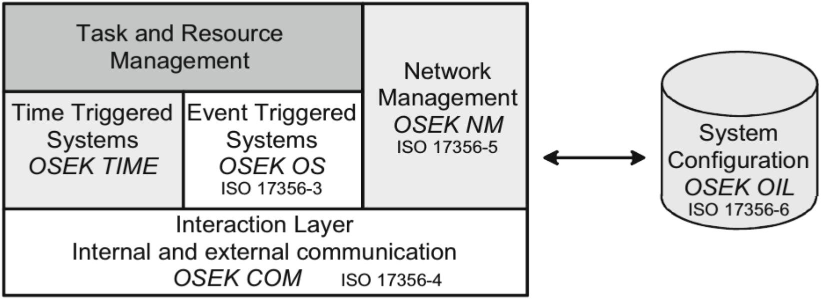


Figure: norm and system overview [ZS14]

Scheduler I

- main component
- event-based
- static priorities
- several scheduling classes (BCC1 to ECC2)
- ISR interrupts tasks

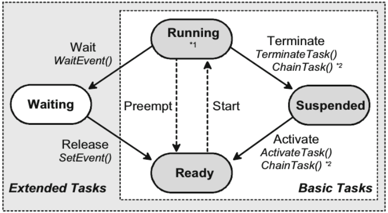


Figure: statechart [ZS14]

Scheduler II

- BCC1
  - most simple class for scheduling
  - unique priorities
  - no task interrupts another task
  - no event handling
  - static scheduling is possible
- ECC2
  - most complex class
  - event handling, preemption, ...
  - to schedule, every priority has a fifo
  - first priority than Round Robin

Events and Ressources

- to synchronize tasks, you need to use binary events
- memory access synchronized by ressources
  - ressources take priorities
  - avoid priority inversion
- shared ressources take the highest task priority
- task which holds a ressource, copy the priority of it

Priority Inversion

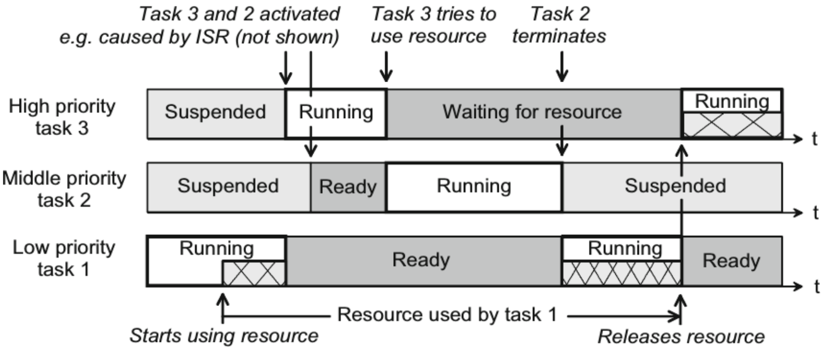


Figure: priority inversion [ZS14]

Priority Inversion

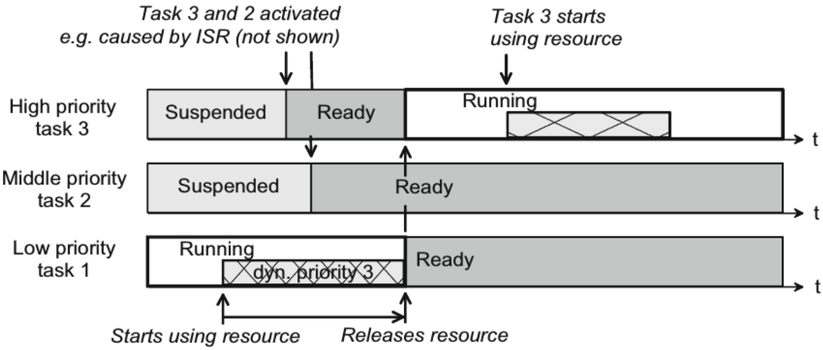


Figure: avoid priority inversion [ZS14]

Counter and Alarm

- leave the CPU tasks have to terminate or to wait
- use alarms for periodic tasks
- alarm may use an action
- trigger the alarm by using a counter

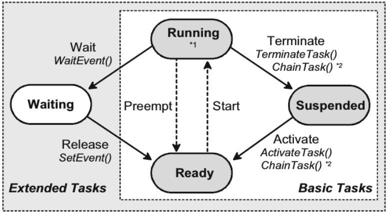


Figure: statechart [ZS14]

System Overview

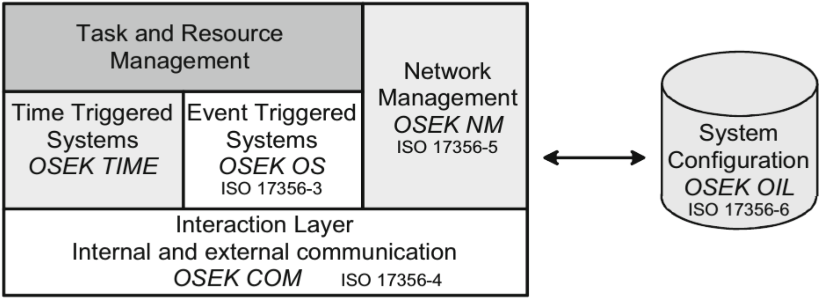


Figure: norm and system overview [ZS14]

Communication I

- transparent communication between tasks and ECUs
- unqueued and queued
- I-PDU for bus transport
- single and periodic mode

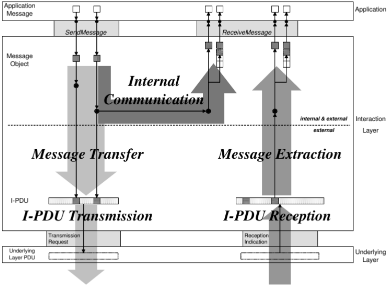


Figure: communication [ose04]

Communication II

- sender calls *sendMessage*
- copy message into a buffer and return to caller
- receiver receives a notification
- receives the message via *receiveMessage*

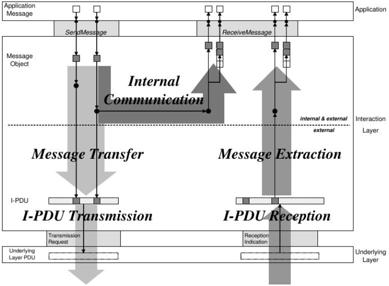


Figure: communication [ose04]

Communication III

- any message has to be defined static in the oil-file
- external communication belogs to a I-PDU message
- I-PDU messages handle more than one message
- it is a bit vector, not byte aligned
  - bounded values
  - sender truncate MSB
  - receiver fills MSB with 0
- byte order of native types predefined per message
- only predefined (compile time) addresses possible

System Overview

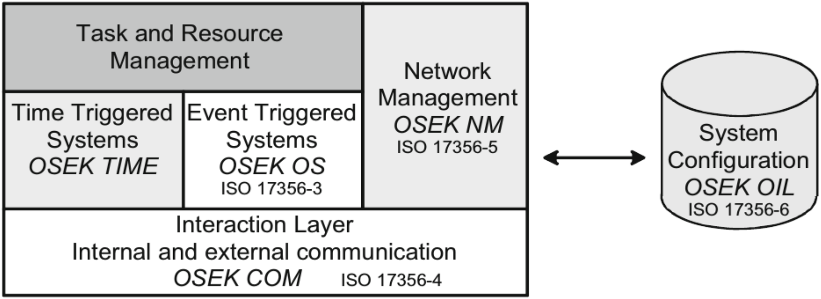


Figure: norm and system overview [ZS14]

Network Management

- only on broadcasting channels
- no master
- all nodes have to be present at development time
- two diffent modes possible
  - indirect network management
  - direct network management
- for indirect NM, every node is a passive channel observer
  - every message, refresh internal storage of active nodes

Direct Network Management

- nodes have an ascending ID
- sending order is predefined
- ring messages for keep alive signal
- if no answer, an alive message is send by all nodes
- reinitialization phase begins and all nodes send alive messages
- now every node recognizes its neighbours → normal operation

System Overview

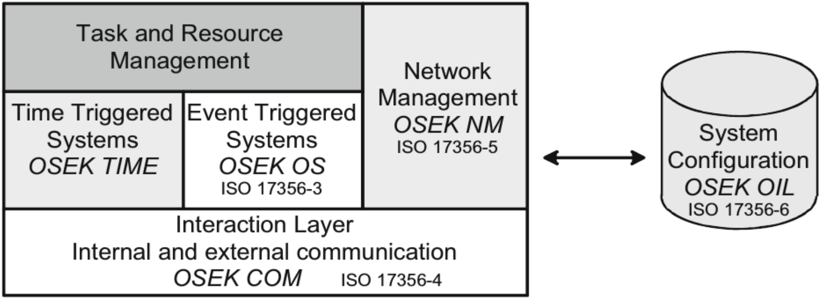


Figure: norm and system overview [ZS14]

OSEK Implementation Language

- used to configure the system
- all tasks, ressources, messages, ... have to defined here
- configuration is statically at compiletime
- no changes allowed

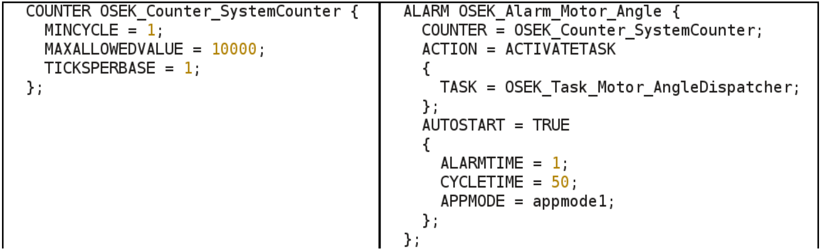
```
TASK OSEK_Task_Motor_AngleDispatcher {
  AUTOSTART = FALSE;
  PRIORITY = 3;
  ACTIVATION = 1;
  SCHEDULE = NON;
  STACKSIZE = 128;
};
```

```
DeclareTask( OSEK_Task_Motor_AngleDispatcher );
TASK( OSEK_Task_Motor_AngleDispatcher ) {

        /**
         * CODE
         */
        ...
        TerminateTask();
}
```

OSEK Implementation Language

- used to configure the system
- all tasks, ressources, messages, ... have to defined here
- configuration is statically at compiletime
- no changes allowed



```
COUNTER OSEK_Counter_SystemCounter {
  MINCYCLE = 1;
  MAXALLOWEDVALUE = 10000;
  TICKSPERBASE = 1;
};
```

```
ALARM OSEK_Alarm_Motor_Angle {
  COUNTER = OSEK_Counter_SystemCounter;
  ACTION = ACTIVATETASK
  {
    TASK = OSEK_Task_Motor_AngleDispatcher;
  };
  AUTOSTART = TRUE
  {
    ALARMTIME = 1;
    CYCLETIME = 50;
    APPMODE = appmode1;
  };
};
```

System Overview

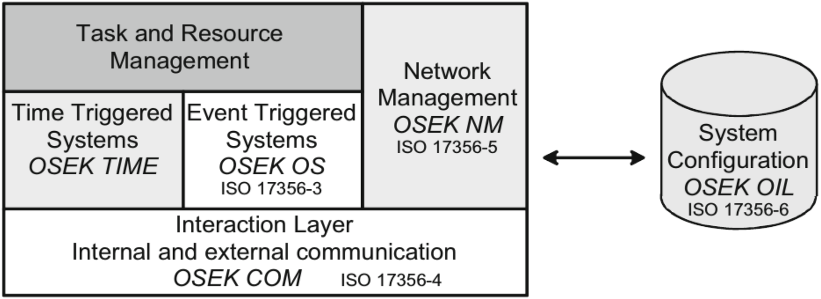


Figure: norm and system overview [ZS14]

Time-triggered OSEK

- completely different OS
- time-triggered scheduler
- OSEK OS is merley a low-prio task
- incompatible with old concepts
- no actual products, only spec
- further development in AUTOSAR

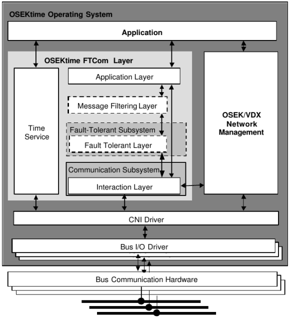


Figure: architecture [ose01]

Q/A

Bibliography

[ose01] OSEK/VDX Time-Triggered Operating System. http://portal.osek-vdx.org/files/pdf/specs/ttos10.pdf, July 2001.

[ose04] OSEK/VDX Communication. http://portal.osek-vdx.org/files/pdf/specs/osekcom303.pdf, July 2004.

[ose05] OSEK/VDX Operating System. http://portal.osek-vdx.org/files/pdf/specs/os223.pdf, February 2005.

[ZS14] Werner Zimmermann and Ralf Schmidgall. Bussysteme in der Fahrzeugtechnik. Springer Vieweg, 5th edition, 2014.