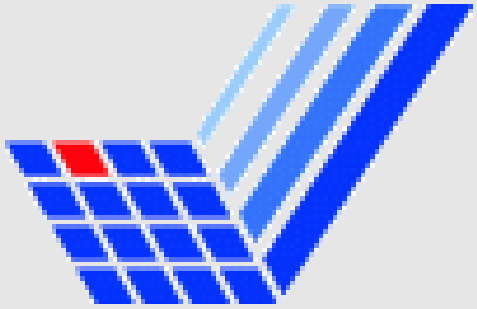
# OSEKtime - Time-Triggered OSEK/OS

**Gregor Kaleta**

gregor.kaleta@udo.edu

PG AutoLab
Seminarwochenende 21.-23. Oktober 2007



# Überblick

- Einleitung
- OSEKtime
  - Task-Zustandsmodell, Scheduling-Verfahren
  - Interrupt-Verarbeitung
  - Deadline-Überwachung / Fehlerbehandlung
  - OSEK-VDX als Subsystem
- OSEKtime FTCom
  - Schichtenmodell
  - Globale Zeit – Synchronisation
- Toolkette

# Notwendigkeit von OSEKtime

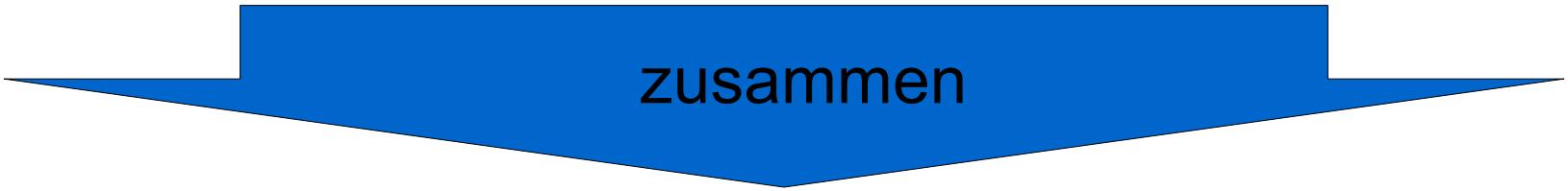
- OSEK\VDX bewährt für Innenraum und Antriebsstrang da autonome Steuergeräte lose Verbunde bilden
- Ungeeignet für X-by-Wire Anwendungen z.B.
  - vollelektronische Bremse (Brake-by-Wire)
  - Lenkung (Drive-by-Wire)
- erfordern zuverlässiges und koordiniertes Handeln in hoher zeitlicher Synchronität
- OSEK\VDX bietet keine globale Zeit für alle Steuergeräte
- Lösung: neuer Standard OSEKtime

# Überblick

- Einleitung
- OSEKtime
  - Task-Zustandsmodell, Scheduling-Verfahren
  - Interrupt-Verarbeitung
  - Deadline-Überwachung / Fehlerbehandlung
  - OSEK-VDX als Subsystem
- OSEKtime FTCom
  - Schichtenmodell
  - Globale Zeit – Synchronisation
- Toolkette

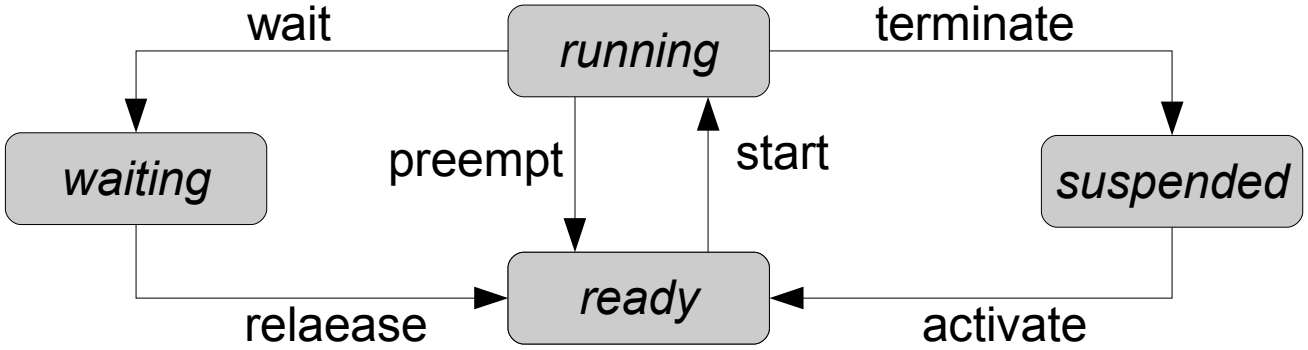
# Was ist OSEKtime

- OSEK/VDX time triggered operating system Version 1.0 verfügbar seit 24. Juli 2001
- zeitgesteuertes Konzept
- besteht im Wesentlichen aus zwei Teilen:
  - Betriebssystem:
    - Tasks
    - Interrupt Service Routinen
    - deren zeitliche Überwachung
  - Kommunikationsschicht FTCom:
    - fehlertolerante Kommunikation
    - globale Zeit für alle Knoten
    - implizite Flusskontrolle

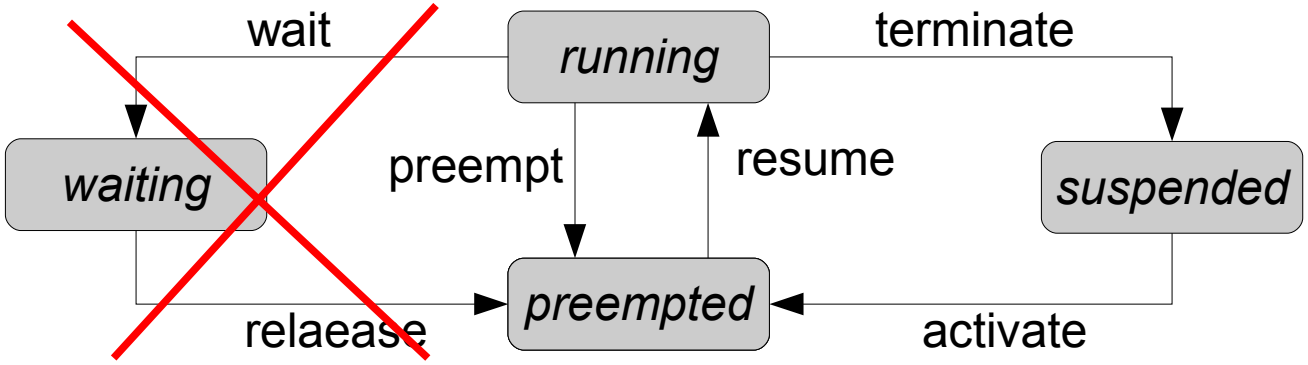



Erfüllung der geforderten Eigenschaften

# Task-Zustandsmodell

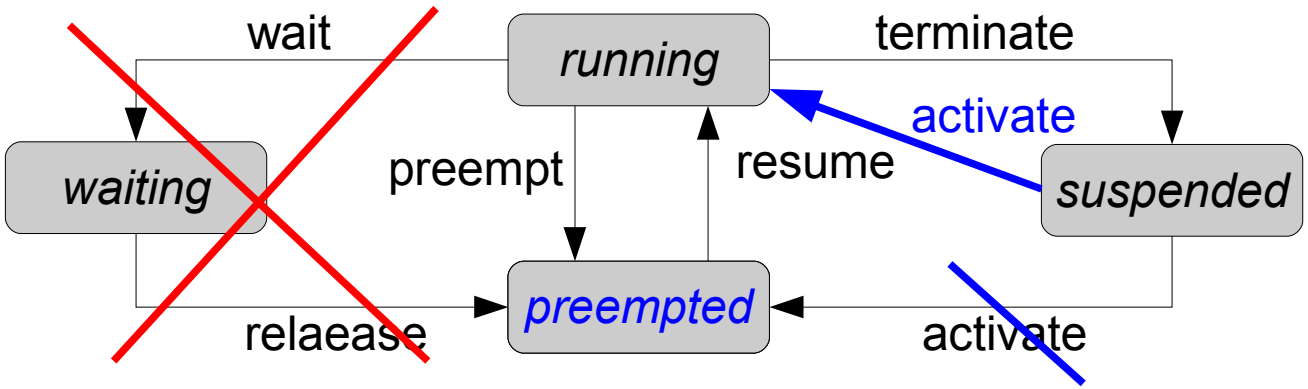


# Task-Zustandsmodell



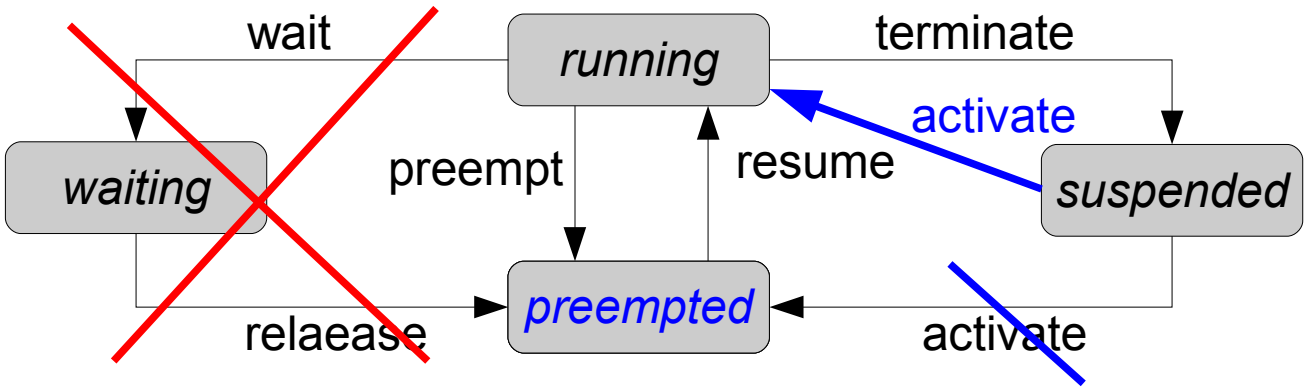

- Keine Events, Counter, Alarme oder Ressourcen => Mechanismus durch Variablen (Task-Attribute) realisierbar

# Task-Zustandsmodell




- Keine Events, Counter, Alarme oder Ressourcen
  => Mechanismus durch Variablen (Task-Attribute) realisierbar
- Keine statischen Prioritäten, fester Startzeitpunkt nach Dispatcher Table: Aktivierung => sofortige Ausführung!

# Task-Zustandsmodell

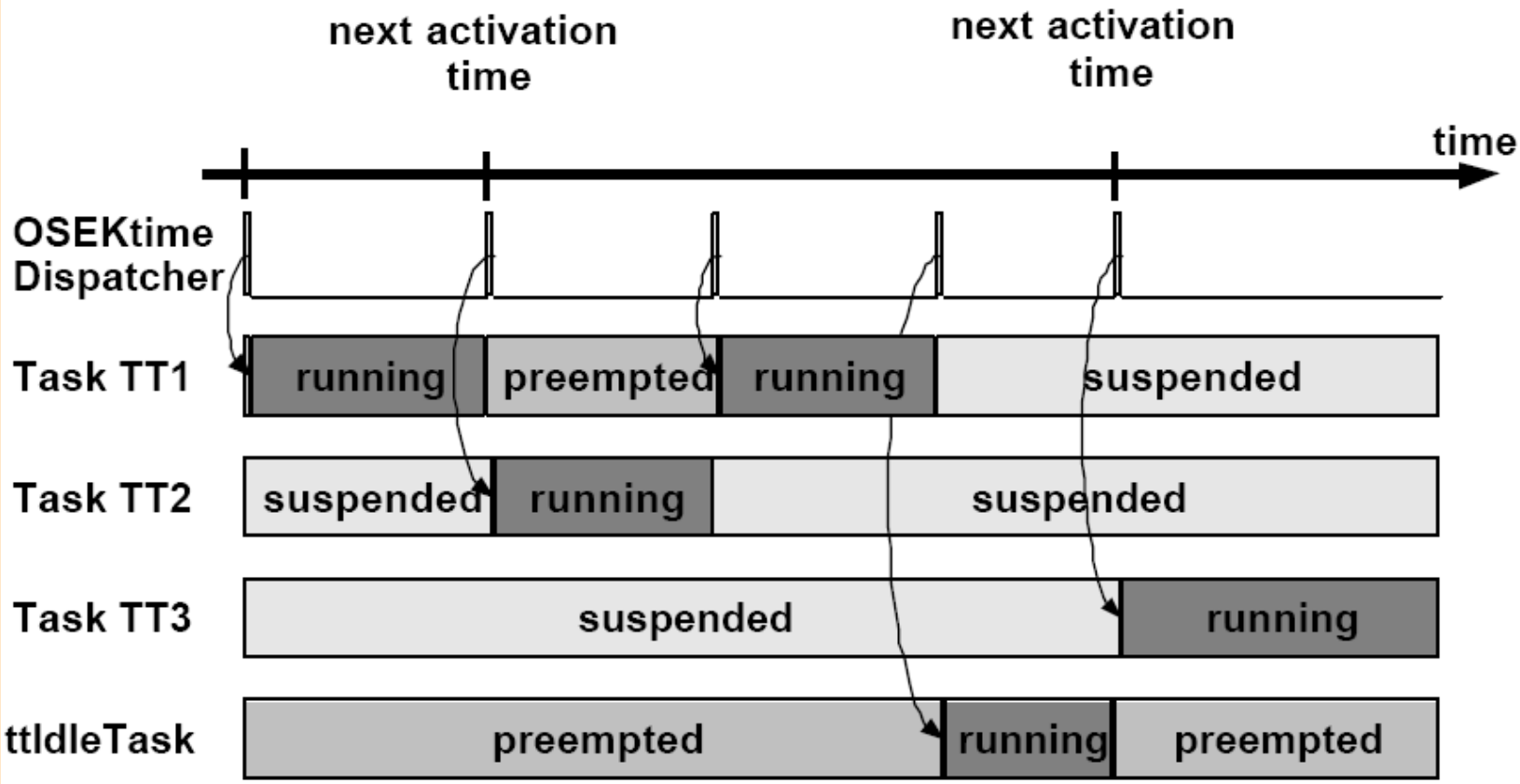



- Keine Events, Counter, Alarme oder Ressourcen
  => Mechanismus durch Variablen (Task-Attribute) realisierbar
- Keine statischen Prioritäten, fester Startzeitpunkt nach Dispatcher Table: Aktivierung => sofortige Ausführung!
- Terminierung => Beendigung der Task
  => Aktivierung einer neuen Task
- Bestimmte Interrupt-Service-Routinen (ISRs) können Tasks nicht unterbrechen

# Scheduling-Verfahren

- Dispatcher Table eine Art Ablauftabelle
- Dispatcher wird durch Timer-Interrupt gestartet
- Nur Dispatcher kann Tasks aktivieren
- Stack Based Scheduling: Starten verdrängter Tasks nach LIFO-Prinzip
- Dispatcher Round: ein kompletter Durchlauf der Tabelle
- Mehrfachaktivierung eines Task in einem Durchlauf mögl.
- Konzept der Anwendungsmodi: mit `SwitchAppMode()` im laufenden Betrieb die Ablauftabelle umschalten
- Leerlauf-Task `ttIdleTask`

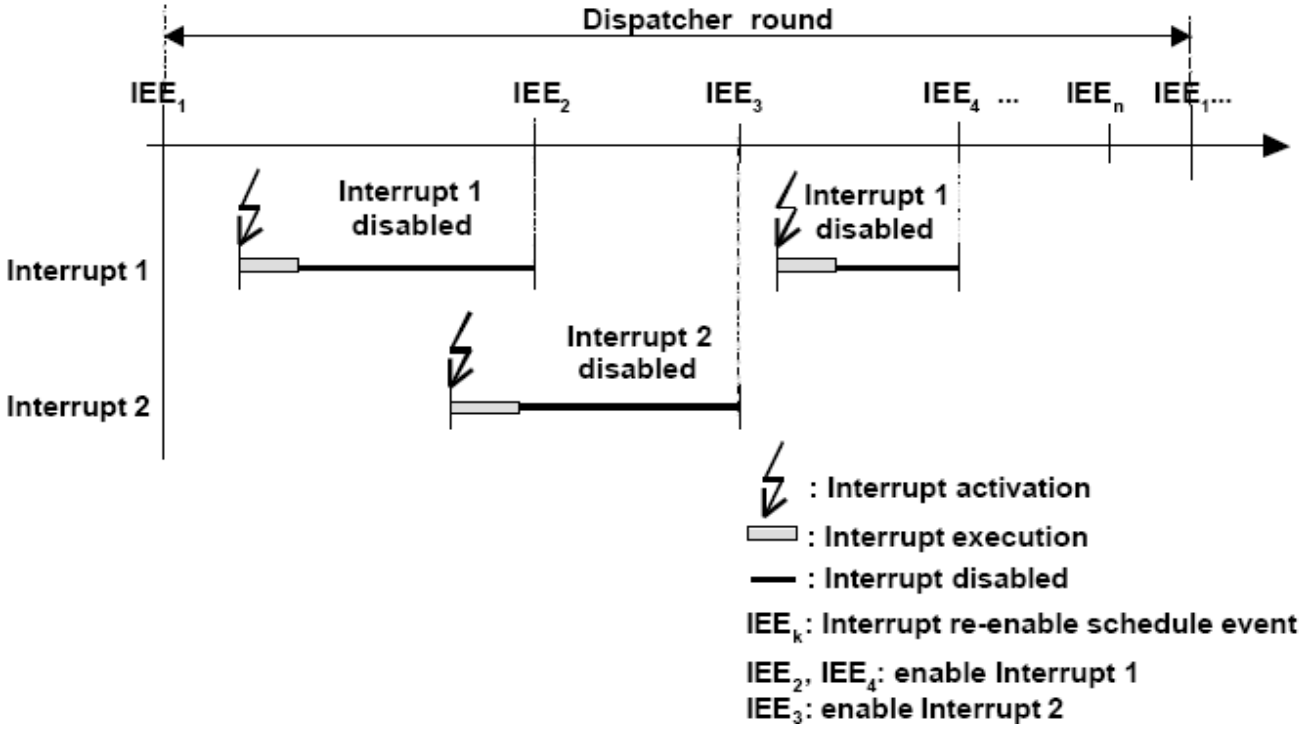
# Beispiel eines Task-Ablaufs



# Interrupt-Verarbeitung

- OSEK/VDX: Interrupts können laufende Tasks jederzeit unterbrechen
- OSEKtime: in der Ablauftabelle werden für jeden Interrupt Zeitfenster definiert, in denen er genau einmal auftreten darf
- Nach Auftreten und Bearbeitung der Interrupt Service Routine bleibt der Interrupt bis zu seinem nächsten Zeitfenster gesperrt
- Definiert werden OSEKtime Interrupts mit dem Makro `ttISR`
- Nicht maskierbare Interrupts sollten vermieden werden, da sie ein Verschieben der OSEKtime-OS-dispatching-Interrupts bewirken

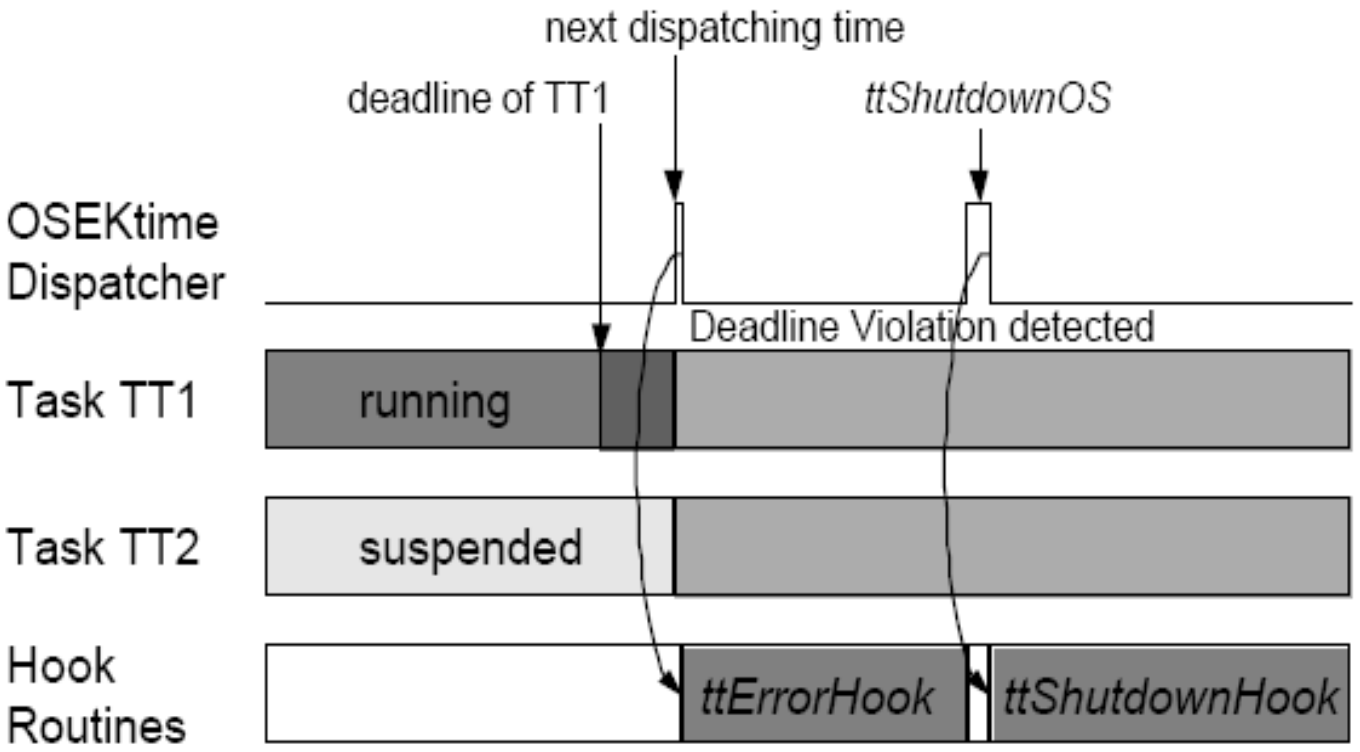
# Interrupt-Verarbeitung



# Deadline-Überwachung/Fehlerbehandl.

- Attribut *Deadline-Monitoring* in Ablauftabelle gespeichert
  - *Strikte Task-Deadline-Überwachung:* exakter Zeitpunkt
  - *Nicht strikte Task-Deadline-Überwachung:* passender Zeitpunkt zw. Deadline und Ende der Dispatcher Round
- Dispatcher überwacht Deadline zur Laufzeit
- Bei Deadlineverletzung Aufruf von `ttErrorHook()`, dass eine eigene Fehlerbehandlung ermöglicht
- Danach beendet sich OSEKtime selbst
- Hier kann verblieben werden oder mit `ttStartOS()` ein Neustart initialisiert werden

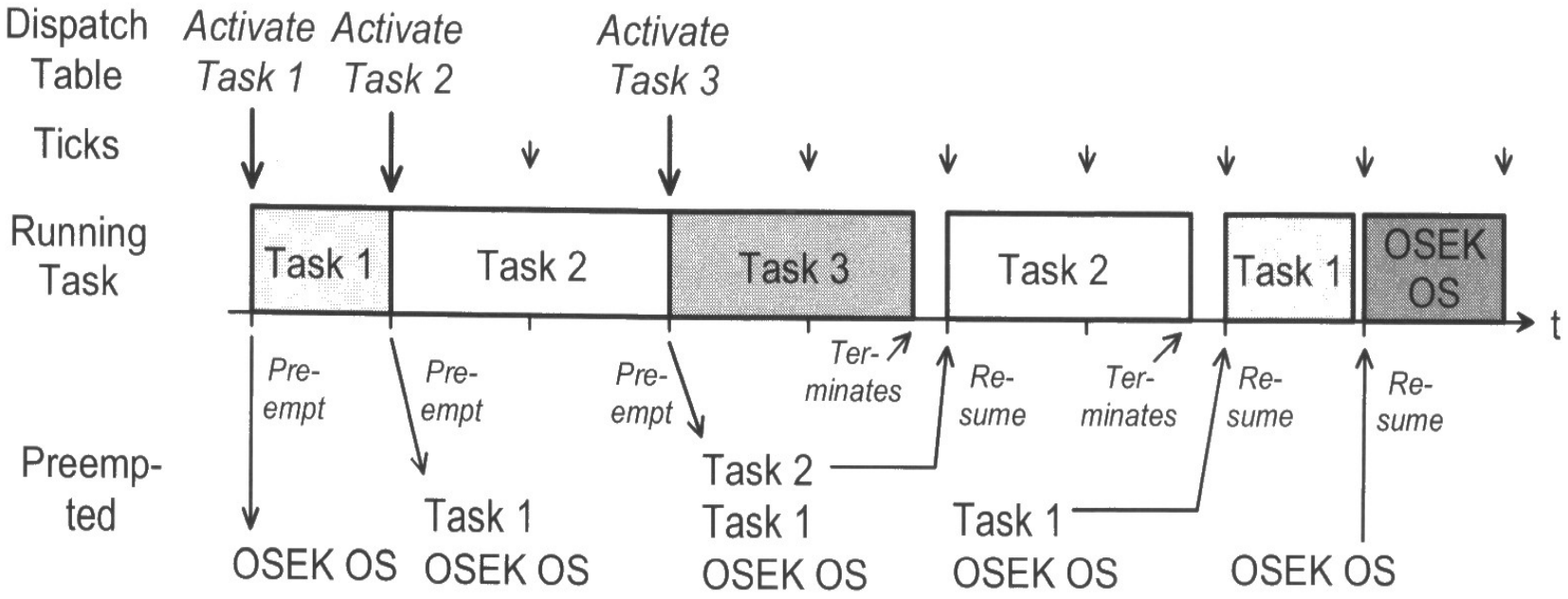
# Deadline-Überwachung/Fehlerbehandl.



# OSEK/VDX als Subsystem

- `ttIdleTask` erster Task, der vom Dispatcher gestartet wird=>läuft also immer, wenn kein andere Task im *running*
- ist nicht in der Dispatcher Tabelle eingetragen
- wird nicht periodisch neu gestartet
- besitzt keine Deadline
- während des `ttIdleTask` dürfen OSEK/VDX Tasks laufen
- er kann jeder Zeit vom OSEKtime Dispatcher (Task, Interrupt) unterbrochen werden
- OSEK/VDX Interrupts nur in „Subsystem-Zeit“

# Beispiel eines Task-Ablaufs



# Überblick

- Einleitung
- OSEKtime
  - Task-Zustandsmodell, Scheduling-Verfahren
  - Interrupt-Verarbeitung
  - Deadline-Überwachung / Fehlerbehandlung
  - OSEK-VDX als Subsystem
- OSEKtime FTCom
  - Schichtenmodell
  - Globale Zeit – Synchronisation
- Toolkette

# OSEKtime FTCom

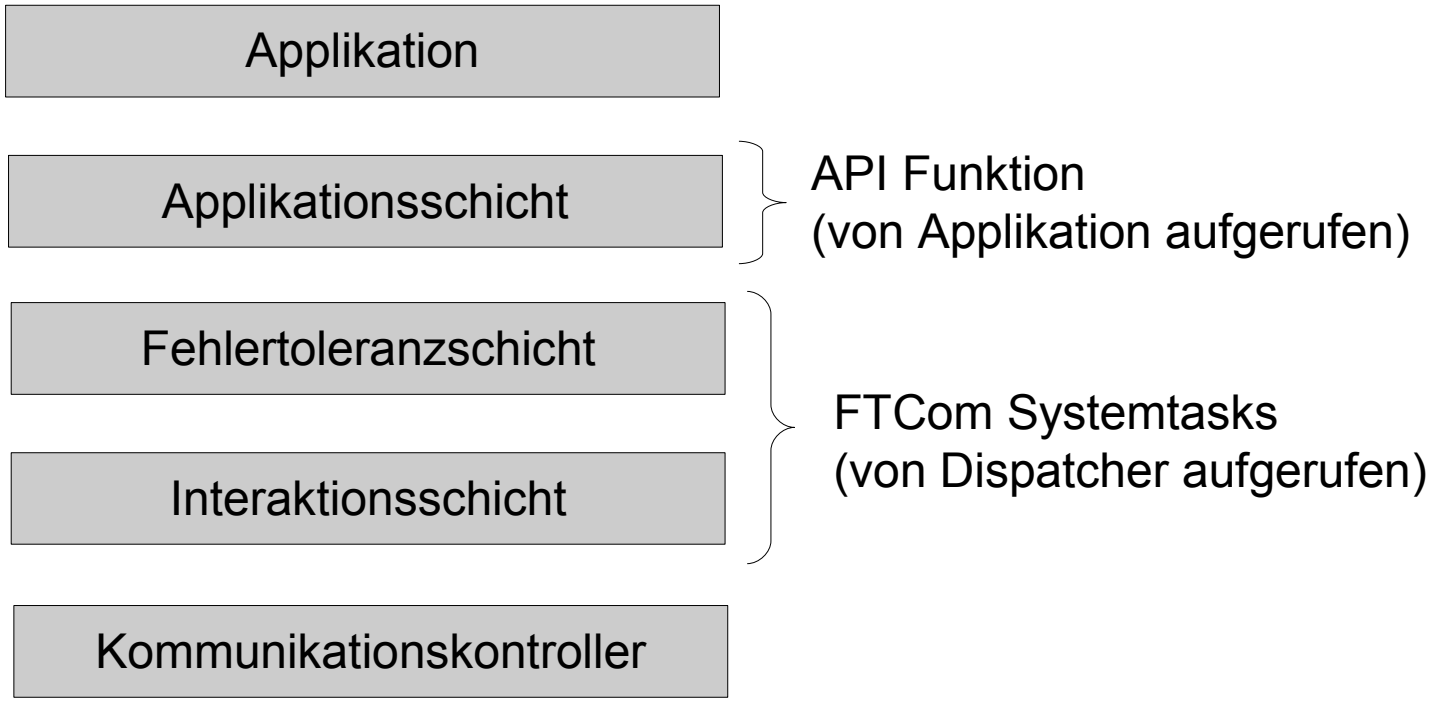
**F**ault-**T**olerant **Com**munication

- Bsp.: Break-by-Wire erfordert
  - deterministisches Zeitverhalten der Kommunikation
  - globale Zeit (garantierte Synchronisation)
  - Robustheit gegen Fehler
- OSEK/VDX Fault-Tolerant Communication Version 1.0 spezifiziert am 24. Juli 2001
- Abstraktion vom konkreten Kommunikationsprotokoll (z.B. FlexRay) => standardisierte Schnittstelle

# Fault-Tolerant Communication

- Babbling Idiot:
  - Buswächter erlaubt Schreibzugriff nur in bestimmten Zeitfenstern
- Fehlertoleranz:
  - Nachrichtenreplikation
  - redundanter Versand
  - Nachrichtenreduktion
  - Bsp. Konsistente Fehler im Wertebereich
    - Reduktionsalgorithmus: Majoritätsvotum
    - Anzahl Nachrichteninstanzen: 3
- Fehlererkennung durch Kommunikationskontroller:
  - korrupte Daten
  - fehlende, verfrühte und verspätete Nachrichten

# Schichtenmodell: OSEKtime FTCom

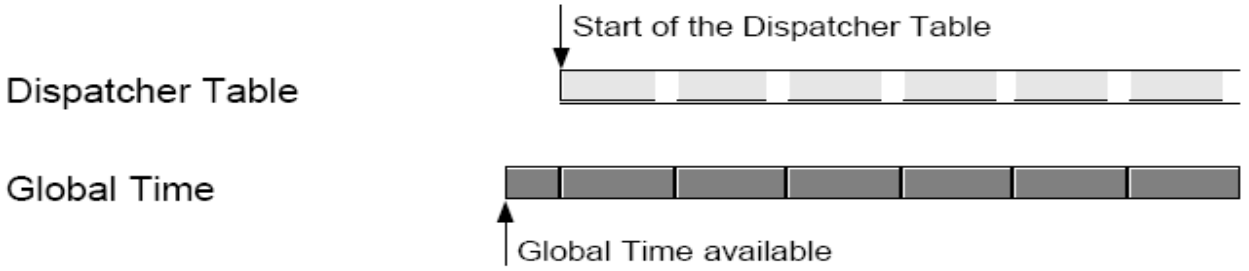


# Globale Zeit - Synchronisation

- Synchronisationslayer wird von FTCom bereitgestellt
- Synchronisation erfolgt beim Systemstart und im Normalbetrieb zyklisch am Ende der Dispatcher Round
- `ttSyncTimes()`: Betriebssystem berechnet Drift zw. lokalen und globalen Zeit, um Ground State zu erweitern oder zu verringern
- Zeitsynchronisation über das Bussystem selbst, z.B. FlexRay-Mechanismen, ist nicht spezifiziert

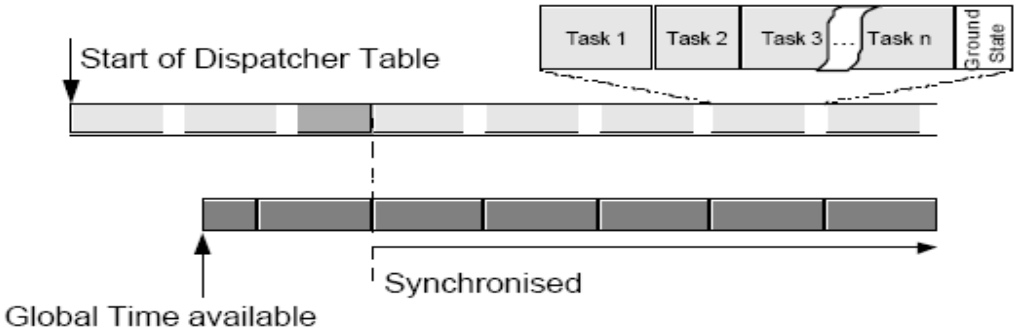
# Synchronisationsverfahren

**Synchronous Start-up**



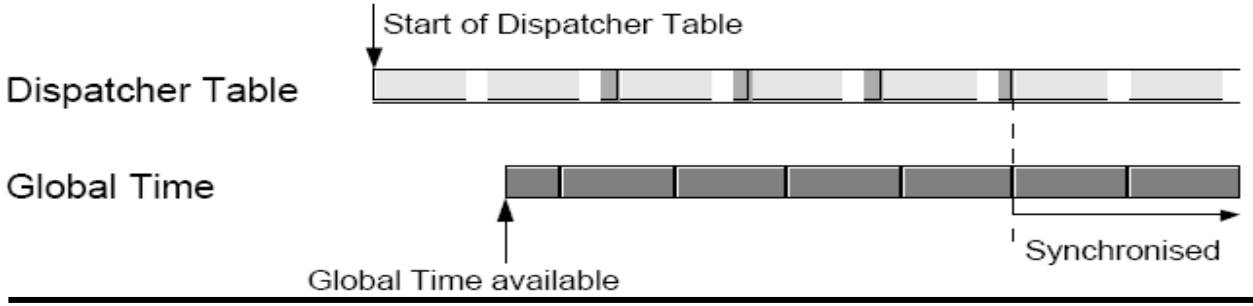

warten auf globale Zeit, dann ausführen

**Asynchronous Start-up - hard**




asynchron ausführen, dann schlagartig anpassen

**Asynchronous Start-up - smooth**

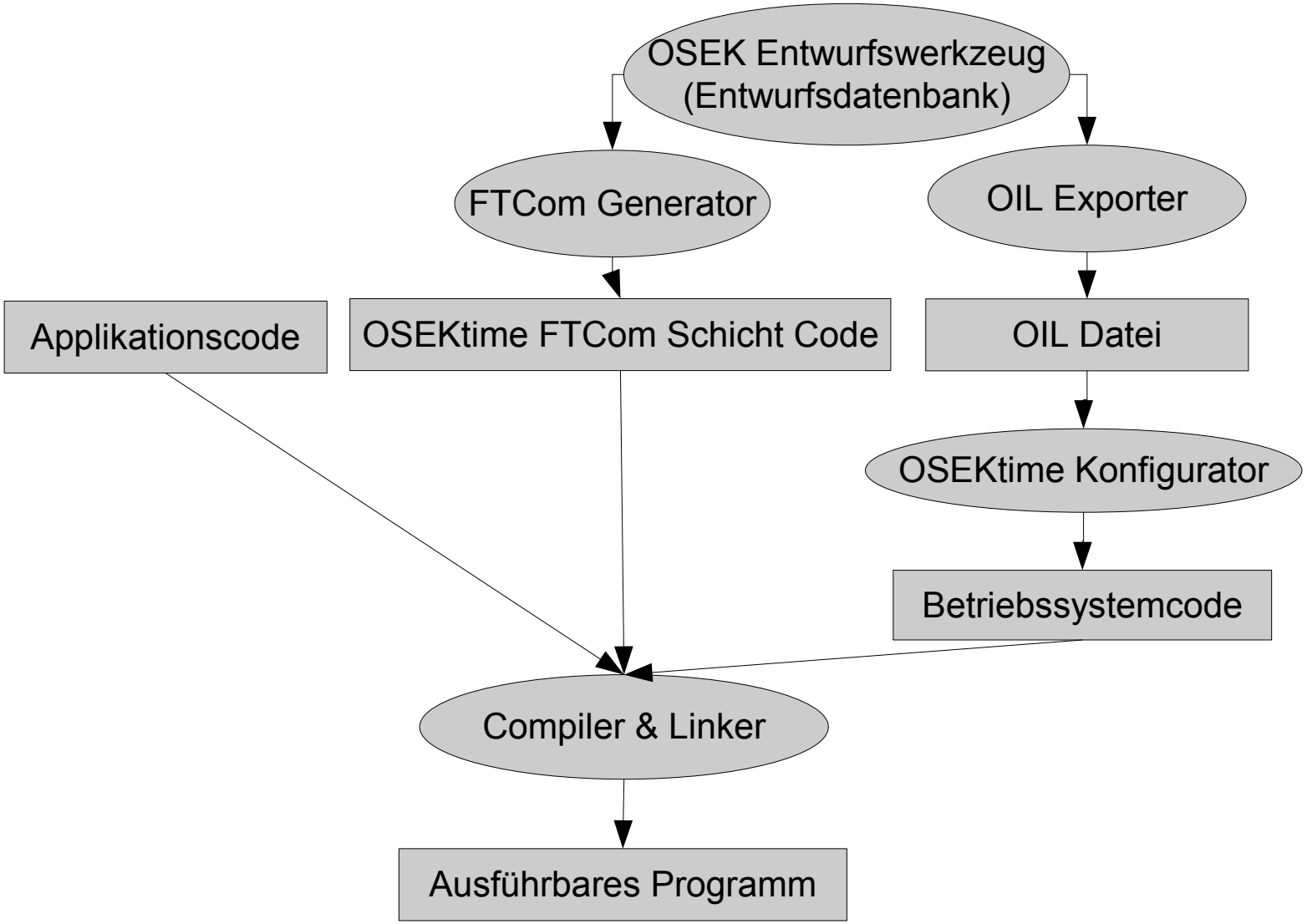



asynchron ausführen, dann schrittweise anpassen

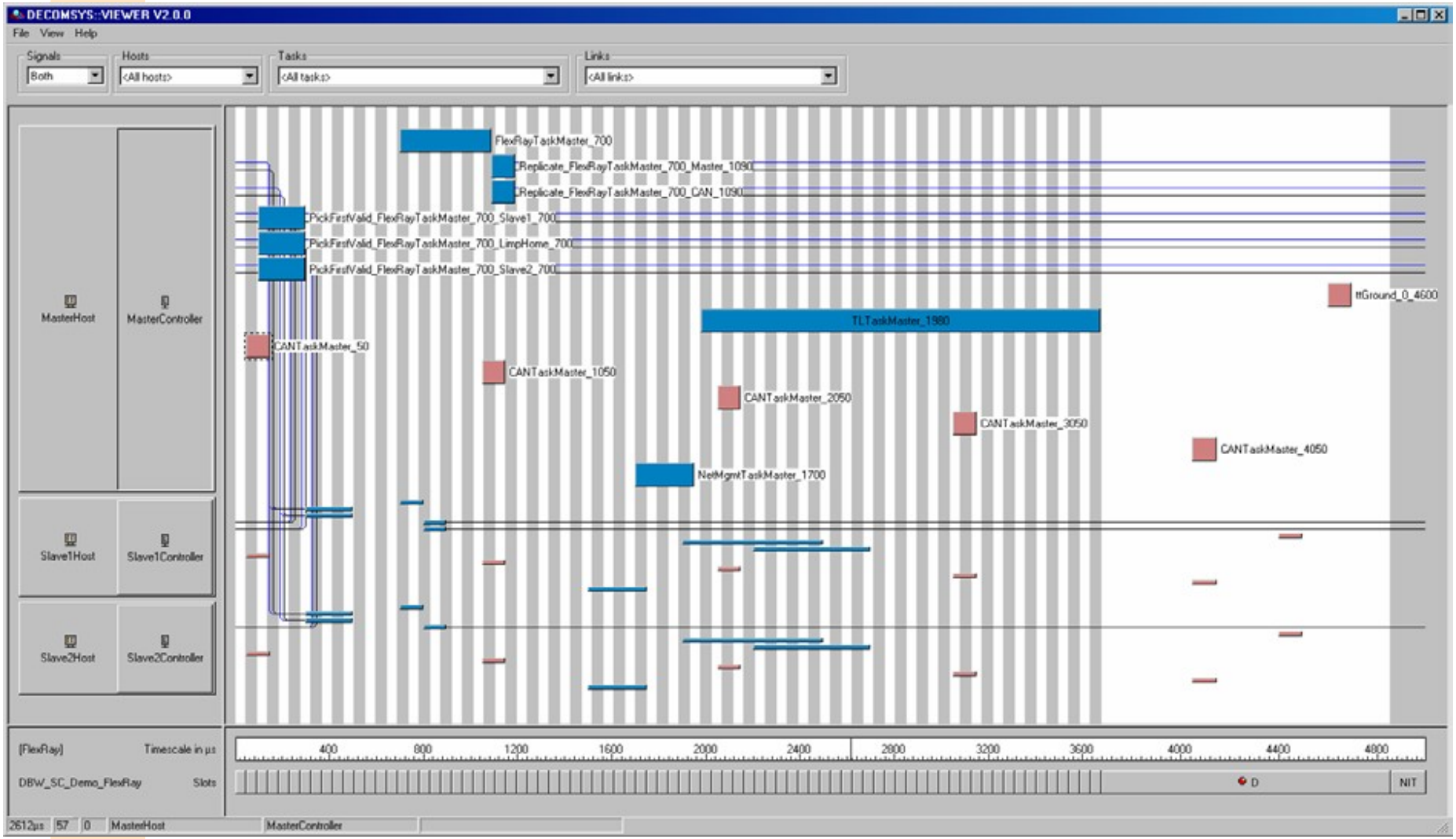
# Überblick

- Einleitung
- OSEKtime
  - Task-Zustandsmodell, Scheduling-Verfahren
  - Interrupt-Verarbeitung
  - Deadline-Überwachung / Fehlerbehandlung
  - OSEK-VDX als Subsystem
- OSEKtime FTCom
  - Schichtenmodell
  - Globale Zeit – Synchronisation
- Toolkette

# Toolkette


OSEK Entwurfswerkzeug
(Entwurfsdatenbank)
FTCom Generator
OIL Exporter
Applikationscode
OSEKtime FTCom Schicht Code
OIL Datei
OSEKtime Konfigurator
Betriebssystemcode
Compiler & Linker
Ausführbares Programm

# Task Schedule in Time Core



# Zusammenfassung

- OSEKtime
  - zeitgesteuertes Konzept (keine Events, ...)
  - gemischtes System (OSEKtime und OSEK/VDX)
- OSEK FTCom
  - deterministisches Zeitverhalten der Kommunikation
  - globale Zeit (garantierte Synchronisation)
  - Fehlertoleranz
  - Allerdings nur Schnittstellenspezifikation, keine konkreten Kommunikationsprotokolle
  - meist Einsatz von FlexRay
- => geeignet für sicherheitsrelevante Applikationen im Automobil

# Literaturverzeichnis


- OSEK/VDX: Time-Triggered Operating System Version 1.0, 2001
- OSEK/VDX: Fault-Tolerant Communikation Version 1.0, 2001
- Homann, Matthias: OSEK. Bonn: mitp-Verlag, 2005
- Zimmermann, Werner; Schidgall, Ralf: Bussysteme in der Fahrzeugtechnik. Wiesbaden: Vieweg, 2007
- Galla, Thomas M.; Olig, Jochen: OSEKtime – Eine Softwareplattform für sicherheitsrelevante verteilte Applikationen im Automobil
- Galla, Thomas M.: TimeCore – Software Komponenten und Entwurfswerkzeuge für sicherheitsrelevante verteilte Applikationen im Automobil

# Fragen

... bei Fragen fragen!