Implementierung des OSEK – Standards auf einem Motorola - µController

Gliederung

- Einführung
- OSEK
- MC68HC08
- Ansätze der Arbeit

Einführung I

Einführung II

Gegebenheiten:

- Verschiedene Mikrokontroller
- Unterschiedliche Funktionalität
- Eine Funktion pro Kontroller implementiert

Anforderungen:

- Echtzeitfähigkeit
- Plattformunabhängigkeit
- Kommunikationsfähigkeit

Aufgabe (extern)

Design eines Echtzeitsystems zur
Verwaltung mehrerer verschiedener
Verarbeitungsaufgaben und Bereitstellung
von Kommunikationsfunktionalität

OSEK / VDX

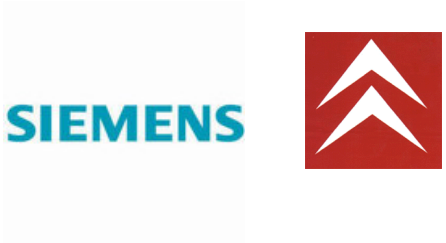
OSEK I

Offene

Systeme und deren Schnittstellen für die

Elektronik im

Kraftfahrzeug



OSEK II

- **1993** – Gründung von OSEK durch deutsche Automobilindustrie unter Leitung des IIIT
- **1994** – Zusammenschluss mit französischer Herstellerinitiative VDX (Vehicle Distributed eXecutive)
- **1995** – erster gemeinsamer Standard in Workshop veröffentlicht
- **heute** – ISO 17356 (Standardisierung teilweise abgeschlossen)

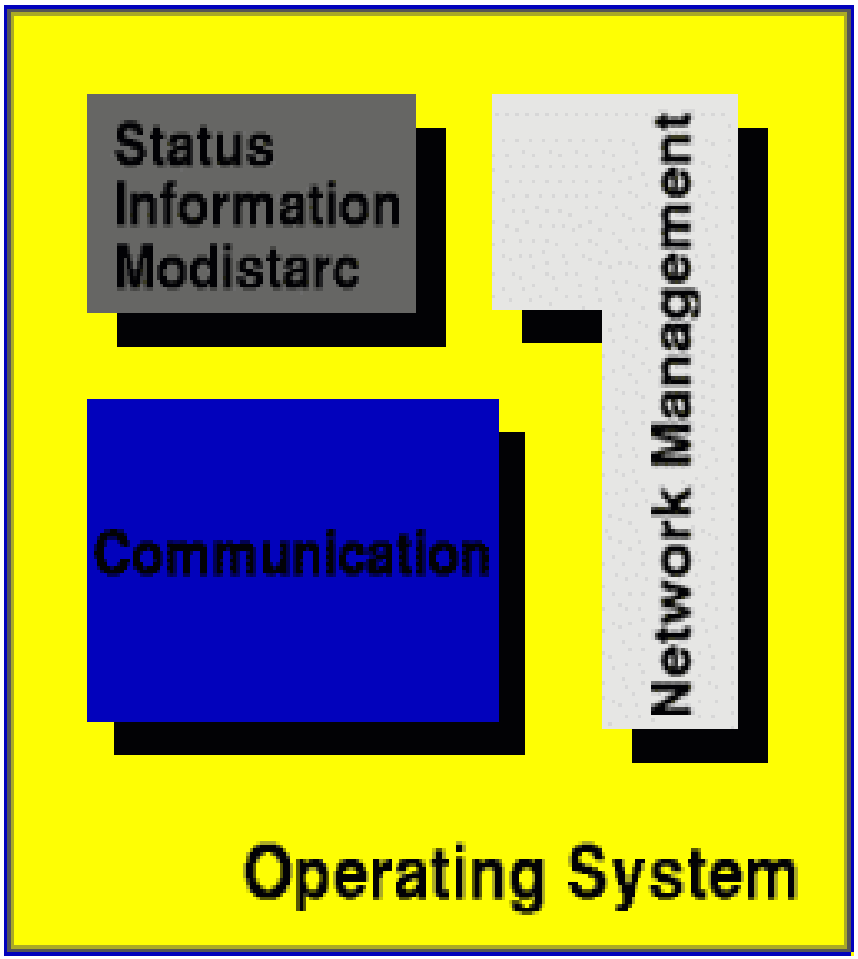
OSEK III

Kernstücke:

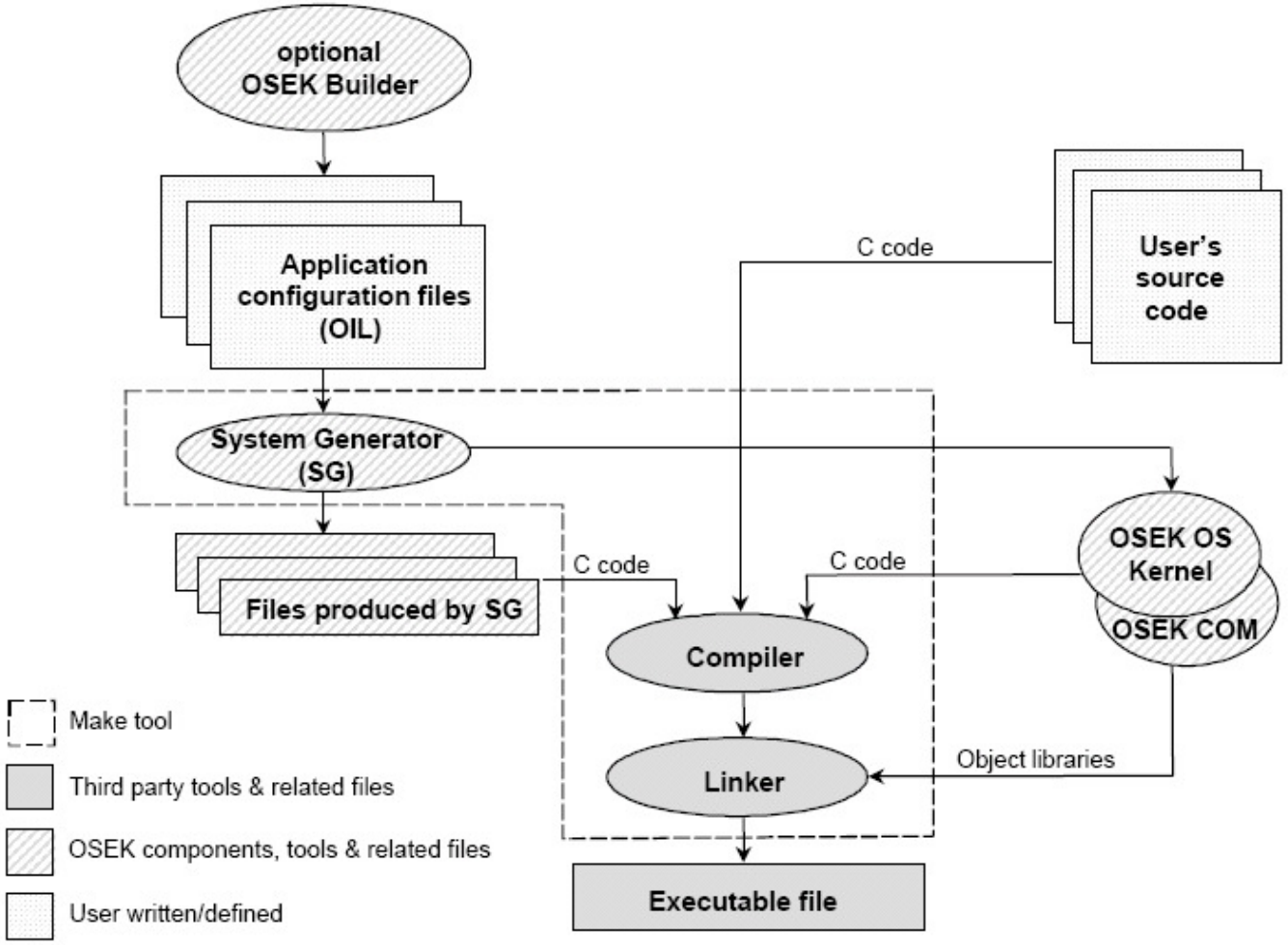
- OSEK – OS
- OSEK – COM
- OSEK – NM

Zusätzlich:

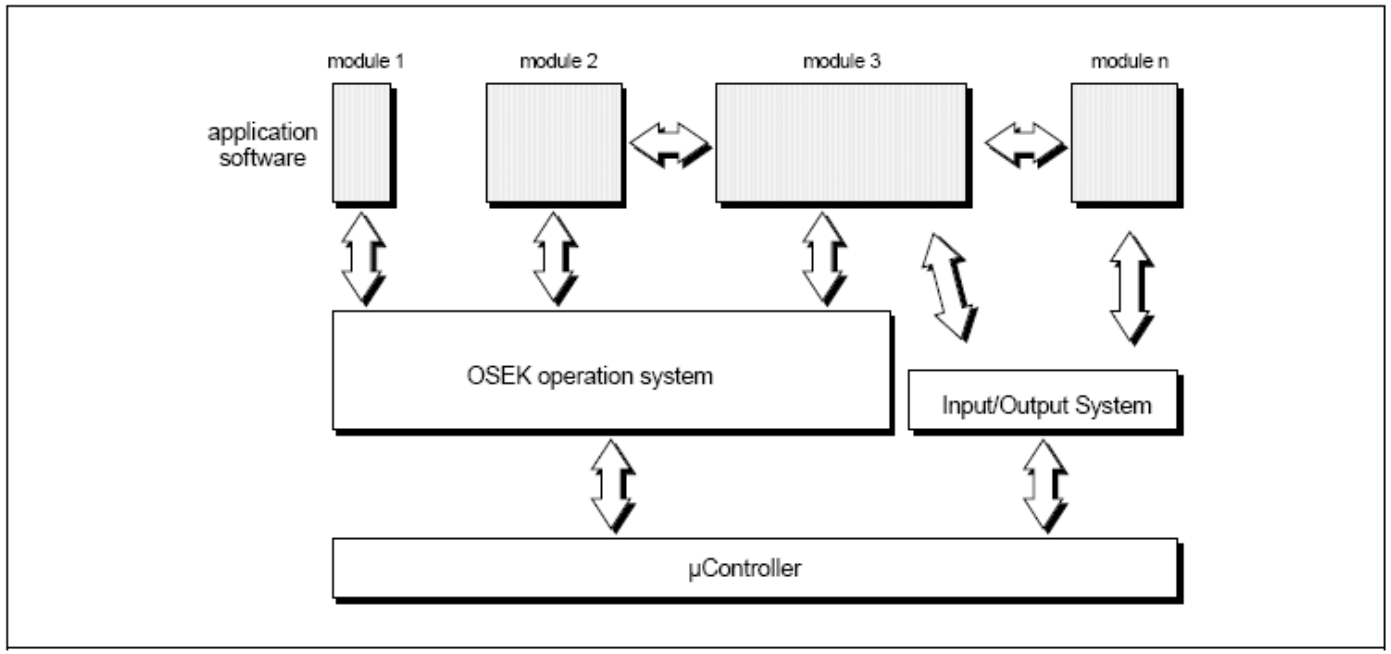
- OIL
- Binding Document
- MODISTARC



OSEK - OIL


optional
OSEK Builder
Application
configuration files
(OIL)
System Generator
(SG)
Files produced by SG
C code
User's
source
code
C code
C code
OSEK OS
Kernel
OSEK COM
Compiler
Linker
Object libraries
Executable file
Make tool
Third party tools & related files
OSEK components, tools & related files
User written/defined

OSEK - OS



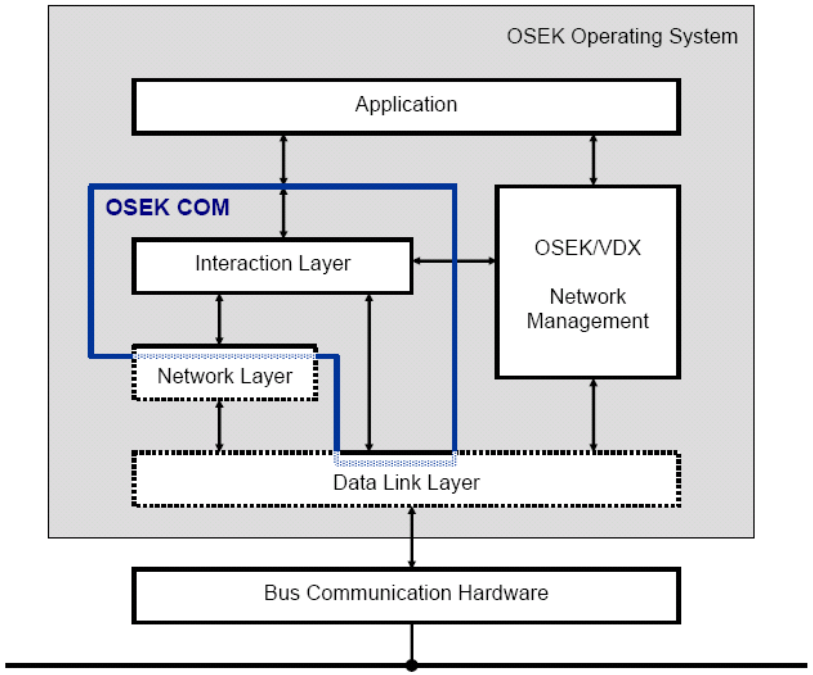
- Statische Skalierung & Konfiguration
- Portabilität von Applikationen
- Echtzeitfähigkeit & Voraussagbarkeit

OSEK - COM

Beschreibung einer Kommunikations-Umgebung für

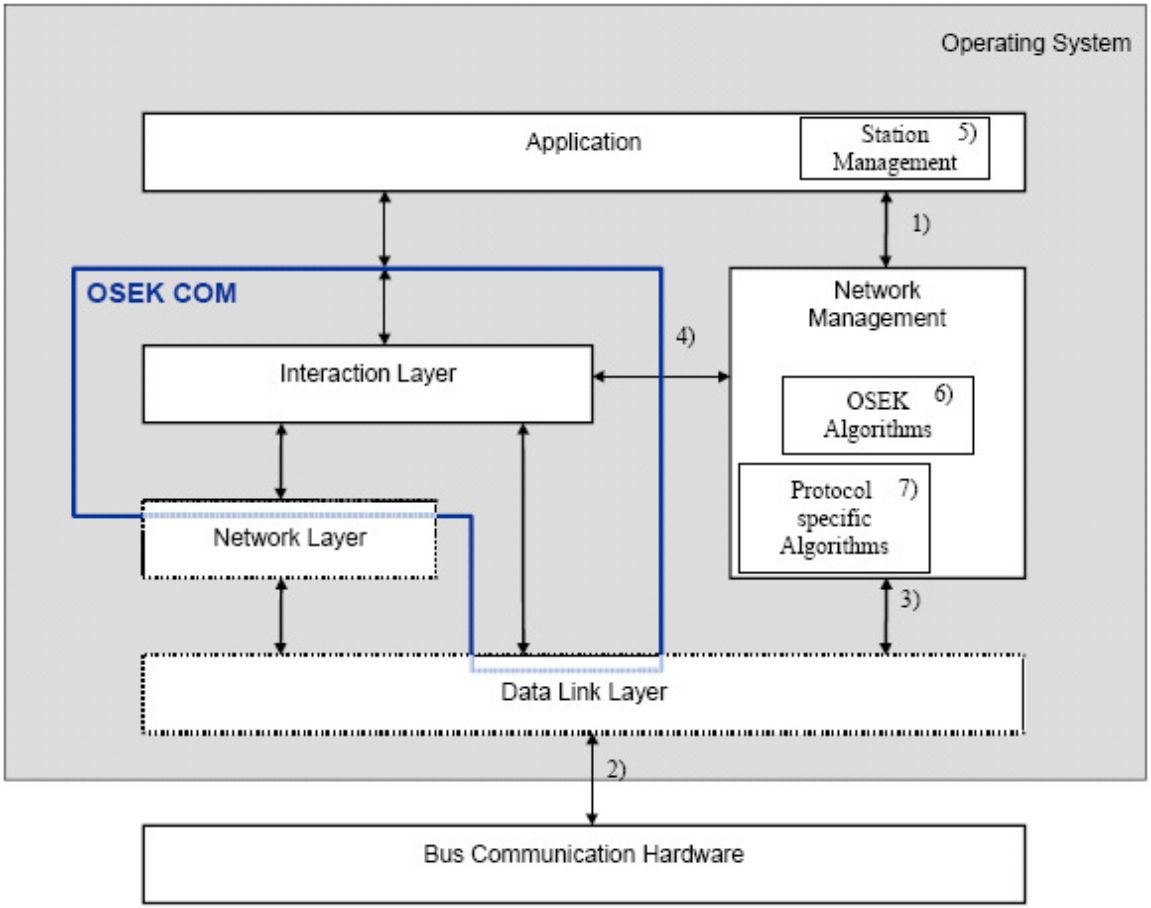
1 - interne Kommunikation (auf einer ECU)

2 - externe Kommunikation (zwischen Prozessen auf verschiedenen ECU)



OSEK - NM

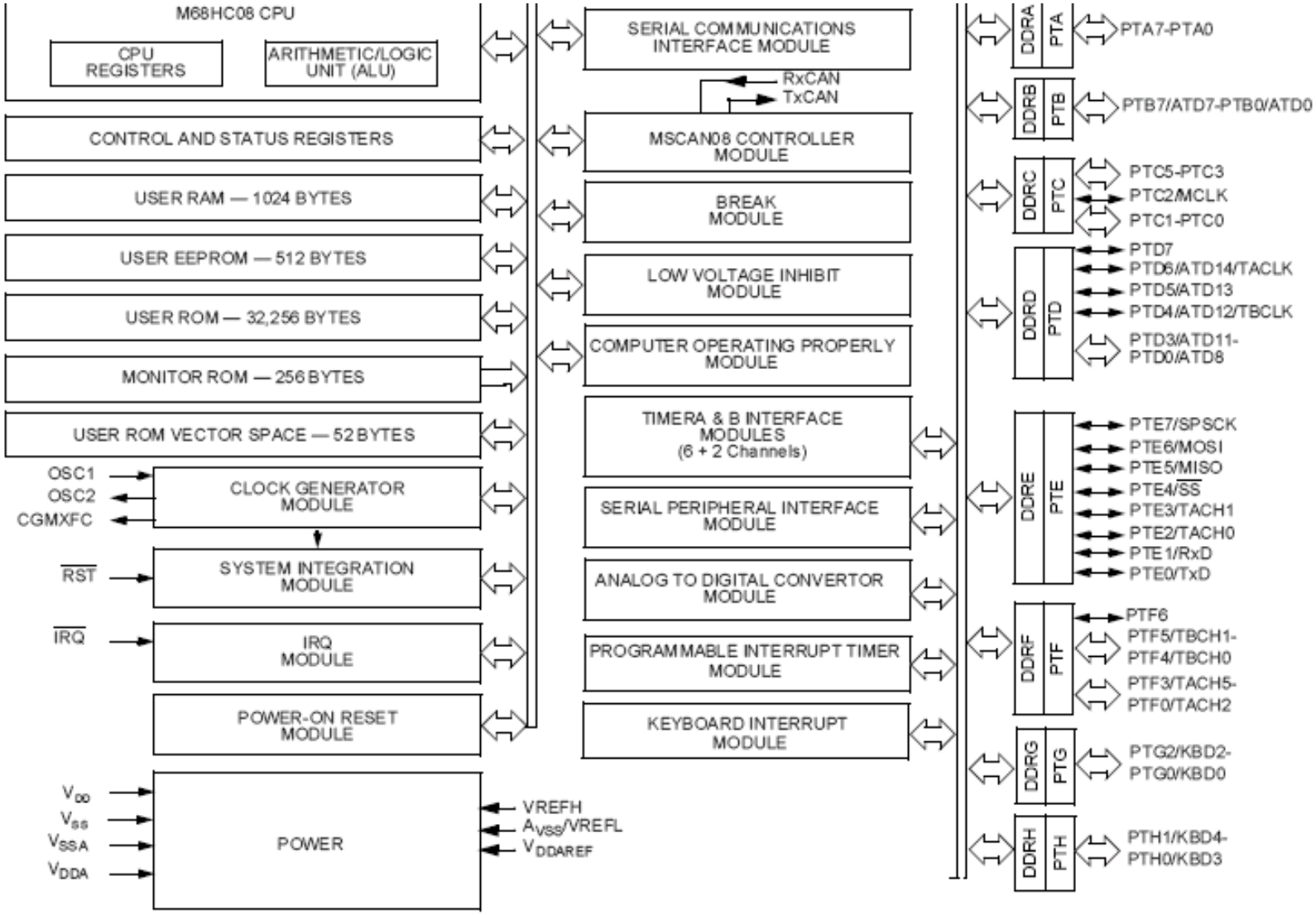
1 – OSEK API
2 – Mehrere Bussverbindungen an µ-Controller
3 – Protokollspezifische Interfaces
4 – Interface für OSEK - COM
5 – Stationsmanagement
6 – OSEK Algorithmen
7 – Protokollspezifische Algorithmen



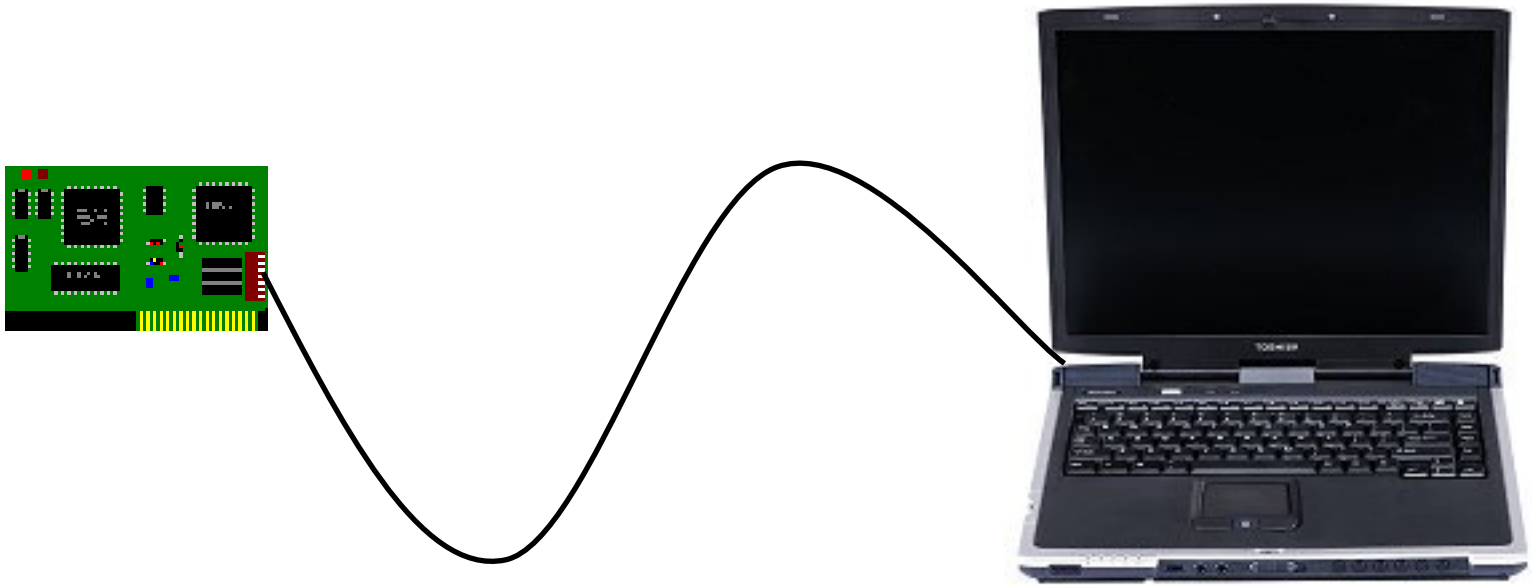
MC68HC08

- 8,4 MHz Busfrequenz
- MSCAN08 – Kontroller (CAN 2.0b)
- Serielle Schnittstelle
- Timer / Clock Generator / AD – Wandler
- 32 KByte User – ROM
- 1KByte On-Chip RAM
- 512 Byte On-Chip EEPROM
- Dezimaloperationen (binär kodiert)
- Speicher zu Speicher Operationen

MC68HC08


M68HC08 CPU
CPU REGISTERS
ARITHMETIC/LOGIC UNIT (ALU)
CONTROL AND STATUS REGISTERS
USER RAM — 1024 BYTES
USER EEPROM — 512 BYTES
USER ROM — 32,256 BYTES
MONITOR ROM — 256 BYTES
USER ROM VECTOR SPACE — 52 BYTES
OSC1
OSC2
CGMXFC
CLOCK GENERATOR MODULE
RST
SYSTEM INTEGRATION MODULE
IRQ
IRQ MODULE
POWER-ON RESET MODULE
VDD
VSS
VSSA
VDDA
POWER
VREFH
AVSS/VREFL
VDDAREF
SERIAL COMMUNICATIONS INTERFACE MODULE
RxCAN
TxCAN
MSCAN08 CONTROLLER MODULE
BREAK MODULE
LOW VOLTAGE INHIBIT MODULE
COMPUTER OPERATING PROPERLY MODULE
TIMERA & B INTERFACE MODULES (6 + 2 Channels)
SERIAL PERIPHERAL INTERFACE MODULE
ANALOG TO DIGITAL CONVERTOR MODULE
PROGRAMMABLE INTERRUPT TIMER MODULE
KEYBOARD INTERRUPT MODULE
DDRA
PTA
PTA7-PTA0
DDRB
PTB
PTB7/ATD7-PTB0/ATD0
DDRC
PTC
PTC5-PTC3
PTC2/MCLK
PTC1-PTC0
DDRD
PTD
PTD7
PTD6/ATD14/TACLK
PTD5/ATD13
PTD4/ATD12/TBCLK
PTD3/ATD11-PTD0/ATD8
DDRE
PTE
PTE7/SPSCK
PTE6/MOSI
PTE5/MISO
PTE4/SS
PTE3/TACH1
PTE2/TACH0
PTE1/RxD
PTE0/TxD
DDRF
PTF
PTF6
PTF5/TBCH1-PTF4/TBCH0
PTF3/TACH5-PTF0/TACH2
DDRG
PTG
PTG2/KBD2-PTG0/KBD0
DDRH
PTH
PTH1/KBD4-PTH0/KBD3

Software - Entwicklung

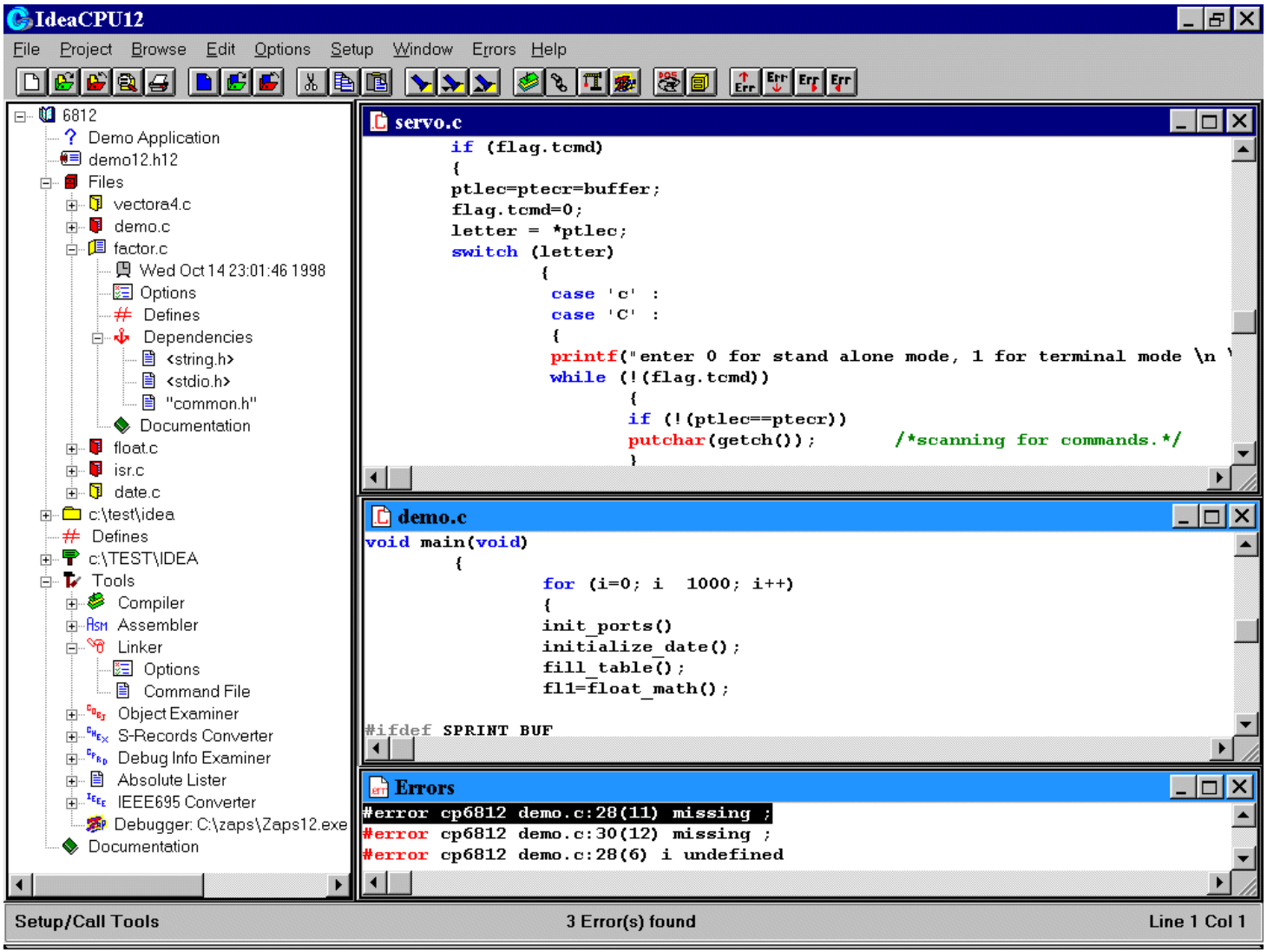


Development Kit mit Kippschaltern, LEDs und einem Beeper

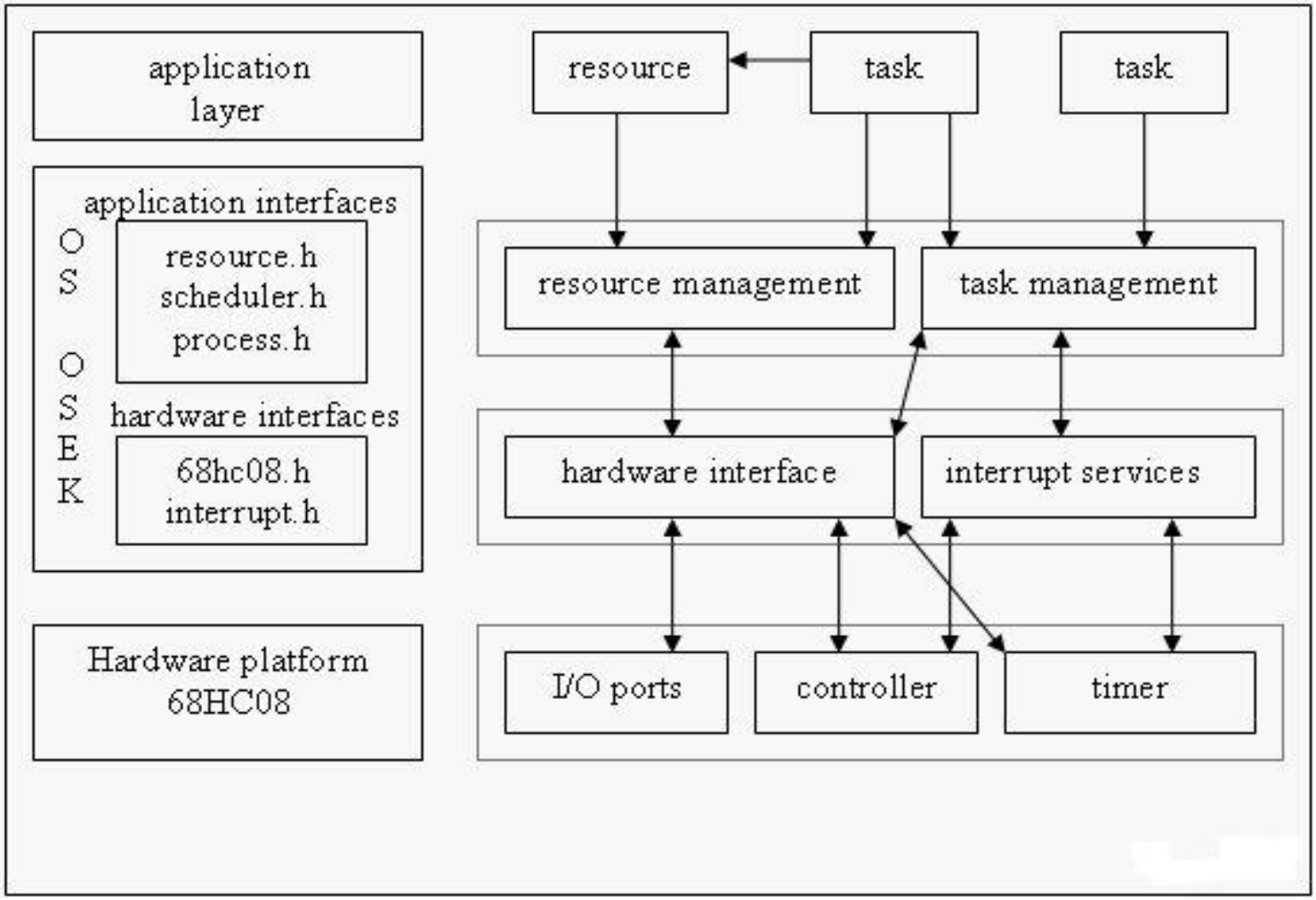
Serielle Verbindung

Toshiba Satellite Notebook (WinXP)

Cosmic Development Tool



Entwurf


application layer
application interfaces
OSEK
resource.h
scheduler.h
process.h
hardware interfaces
68hc08.h
interrupt.h
Hardware platform 68HC08
resource
task
task
resource management
task management
hardware interface
interrupt services
I/O ports
controller
timer

Task / Resource Management

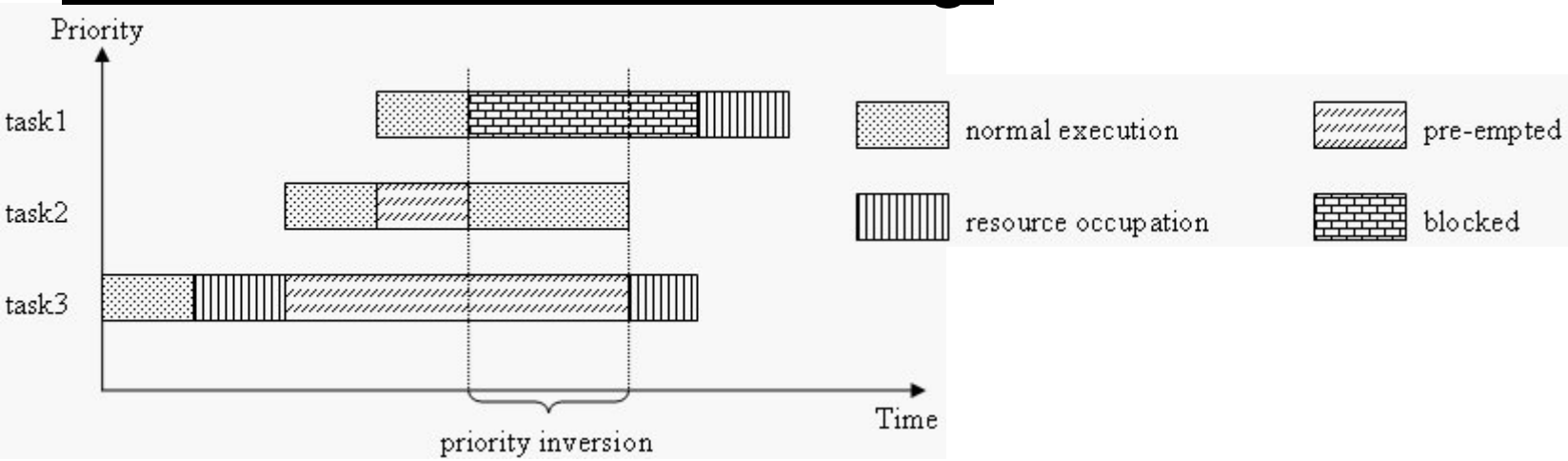
Taskvarianten:

- Unterbrechbar
- Nicht Verdrängbar

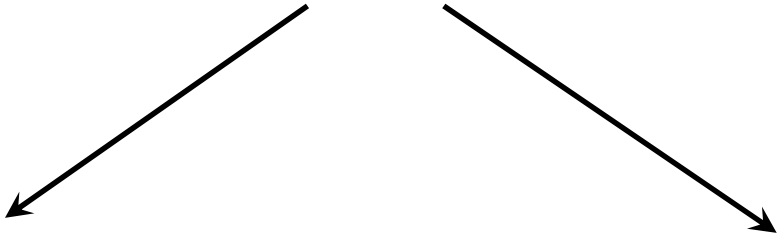
Resourcenschutz:

- Locks
- Semaphore

→ Prioritätsvertauschung:



Lösungen



Prioritätsvererbung:

- blockierender Task übernimmt Priorität des wartenden

Ceiling Protocoll:

- Resource bekommt Priorität zugeordnet
- nutzender Task übernimmt Ressourcen – Priorität
- Weitere Eigenschaft

  → keine Deadlocks

RTOS - Statische Zuordnungen

0xC000	char startStack	(lower byte)	1 byte
0xC001	SRT [0]	ResourceType symbol	1 byte
0xC002		PriorityType ceilingPriority	1 byte
0xC003	SRT [1]	...	2 byte
...			
0xC01F	SRT [15]	...	2 byte
0xC021	STT [0]	* startPC	2 byte
0xC022			
0xC023		PriorityType staticPriority	1 byte
0xC024		TickType interval	1 byte
0xC025	STT [1]		4 byte
...			
0xC045	STT [9]		4 byte
0xC049			

- SRT: Static Resource Table
- STT: Static Task Table

Dynamische Zuordnungen

0x0090	DRT [0]	TaskType owner	1 byte
0x0091	DRT [1]	...	1 byte
...			
0x009F	DRT [15]	...	1 byte
0x00A0	DTT [0]	* actualPC	2 byte
0x00A1			
0x00A2		char actualPriority	1 byte
0x00A3		register X	1 byte
0x00A4		register H	1 byte
0x00A5		ACCU	1 byte
0x00A6		CCR	1 byte
0x00A7		TaskStateType status	1 byte
0x00A8		TickType tickCount	1 byte
0x00A9		char stackCounter	1 byte
0x00AA		char [ ] privateStack	48 byte
...			
0x00D9			
0x0114	DTT [1]	...	58 byte
...			
0x02AA	DTT [9]	...	58 byte
0x02E4			

- DRT: Dynamic Resource Table
- DTT: Dynamic Task Table

northumbria UNIVERSITY

TECHNISCHE UNIVERSITÄT DRESDEN

Fragen?