

## **Open Realtime Framework**

**Standardisierte Open Source**

**Umgebung für individuelle**

**Steuerungsarchitekturen**

Vortrag zur Linux Automation Konferenz 2005  
von Hermann Betz, Yellowstone Soft

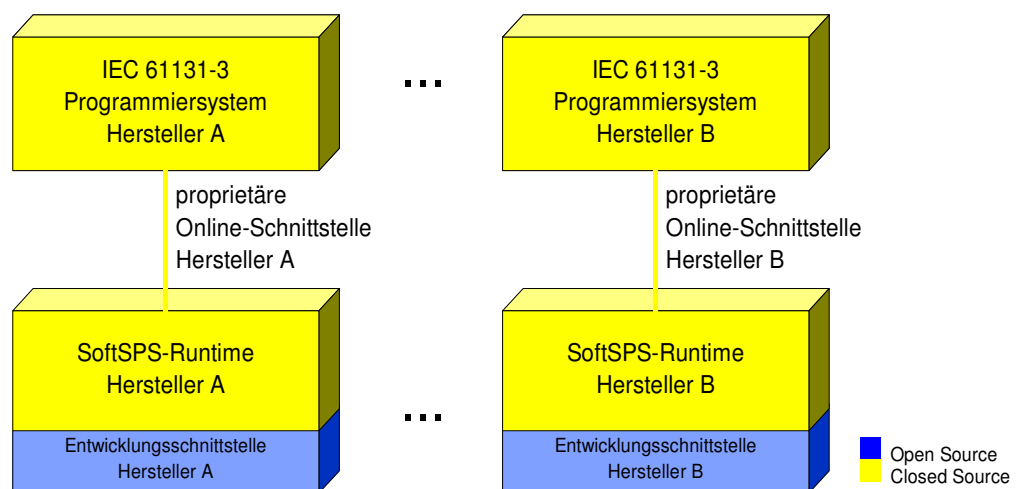
## **Vision**

- **Betriebssystemplattform Linux hat festen Platz in der Automatisierungswelt erobert**
- **auf Applikationsebene dominieren proprietäre Closed Source Lösungen**
- **Schaffung eines einfachen, offenen, skalierbaren Frameworks mit Focus auf die Steuerungstechnik**

## Motivation

- zahlreiche existierende proprietäre Lösungsansätze
- andere Lösungsansätze komplex oder Ressourcen intensiv (z.B. OCEAN basiert auf Corba)
- derzeit keine offene, skalierbare Lösung verfügbar

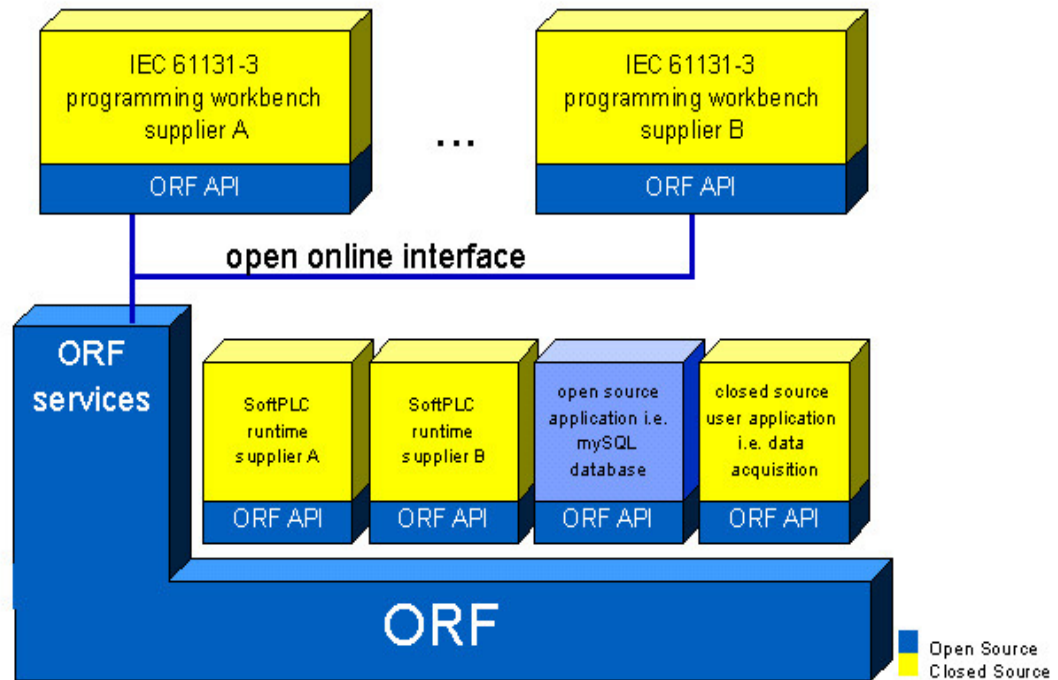
## klassische Architektur:



## Merkmale:

- monolithische Architektur
- festgelegter Funktionsumfang
- beschränkte Erweiterungsmöglichkeiten

## ORF-Architektur:



## Merkmale:

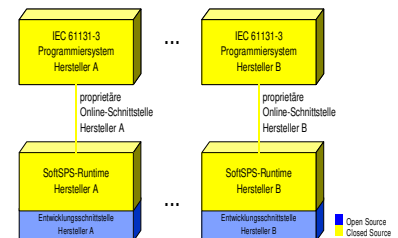
- alle wesentlichen Schnittstellen als Open Source
- Integrationsmöglichkeit für Closed Source Plugins
- offenes Methoden Interface
- Methoden lokal oder über Netzwerk aufrufbar
- Interaktion von SoftSPS und IEC 61131-3 Workbenches unterschiedlicher Hersteller in einheitlicher Umgebung

## Entwicklungskriterien:

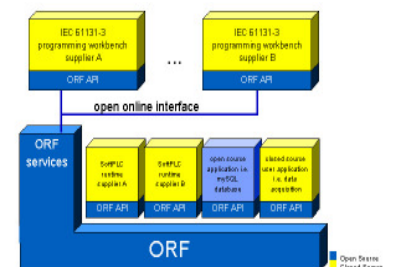
- axiomatisches Design
- plattformübergreifende Spezifikation
- minimale Anforderungen an Betriebssysteme
- lauffähig auf 16 Bit und 32 Bit Architekturen
- hohe Skalierbarkeit
- spezieller Focus auf die Steuerungstechnik
- keep it smart and simple

## Paradigmenwechsel:

Von der Closed Source Steuerung  
mit Erweiterungsschnittstelle



zum Open Source Steuerungssystem  
mit Closed Source Extensions.



## **Strategie:**

- **Spezifikation**
- **Referenzimplementierung (LGPL)**
- **funktionsfähige, verfügbare Plugins**

## **Aufbau der Spezifikation:**

- **Begriffsdefinitionen, prinzipielle Architektur**
- **Speicherlayout**
- **Methoden-API**
- **Systemverhalten**
- **Architekturbeispiele**

## Begriffsdefinitionen:

### PLC:

Komplettes Steuerungssystem, das auf einer Rechnerhardware läuft

### DEVICE:

Reale oder virtuelle CPU innerhalb einer PLC. Eine PLC kann aus einem oder mehreren DEVICES bestehen. Jedem Device ist eine definierte PAGE zugeordnet. Jedes DEVICE wird periodisch aktiviert.

### PAGE:

Shared Memory Speicherbereich, der für jedes DEVICE existiert. Enthält lokale Speicherbereiche, einen Debuggingbereich und Speicher für anwenderspezifische Programme wie z.B. eine SoftSPS.

## Begriffsdefinitionen:

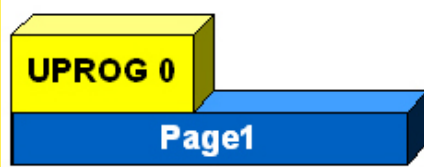
### THREAD:

Kontrollfluß innerhalb eines DEVICE. Jedes DEVICE hat einen Thread 0 der einzelne Programme nacheinander bearbeitet. Weitere Threads mit eingeschränktem Zugriff auf die PAGE sind möglich.

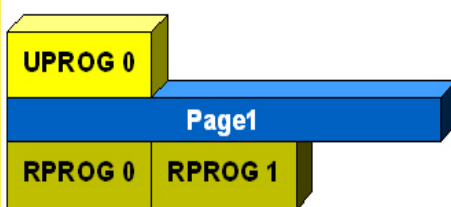
### PROGRAM:

Ablauffähiges Steuerungsprogramm. Es wird zwischen Realtime-Programmen (RPROGs) und User-Programmen (UPROGs) unterschieden. RPROGs werden unter Echtzeitbedingungen ausgeführt. UPROGs laufen ohne Echtzeitbedigungen ab und unterliegen dem Scheduling des Betriebssystems.

## Beispielarchitektur:

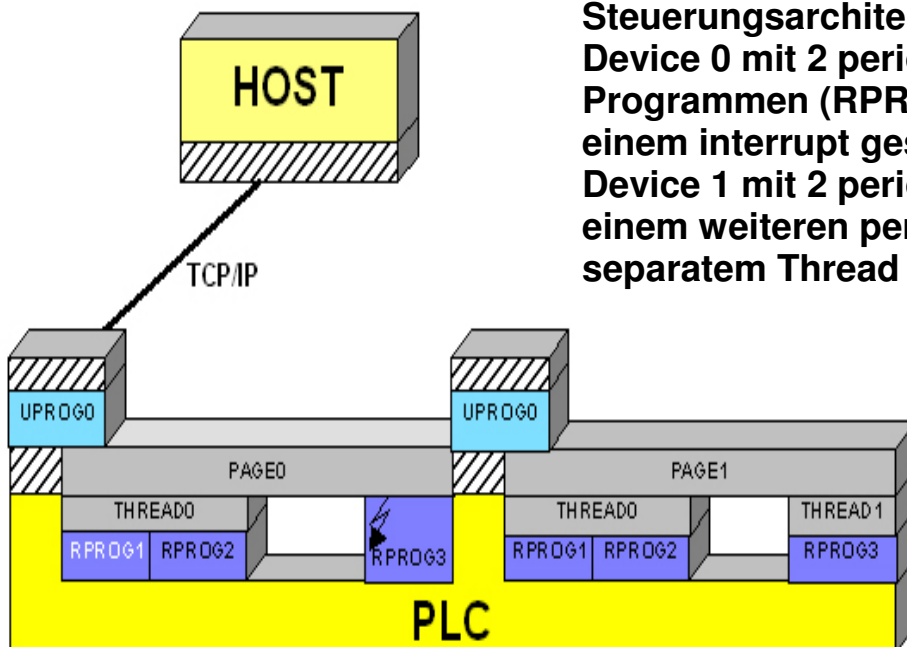


einfachste Steuerungsarchitektur bestehend aus einem Device und einem nicht echtzeitfähigen Programm



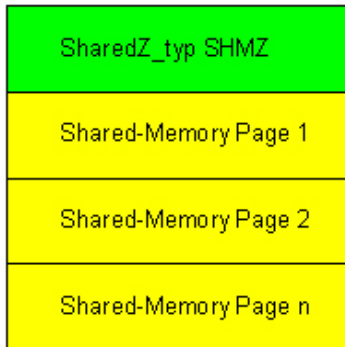
einfache Steuerungsarchitektur bestehend aus einem Device und einem UPROG zur Online Anbindung und zwei echtzeitfähigen RPROGs, die periodisch bearbeitet werden

## Beispielarchitektur:



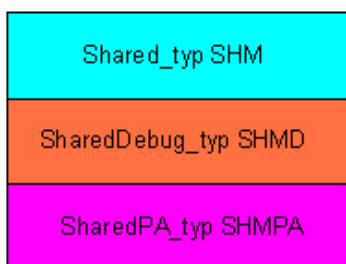
Steuerungsarchitektur mit 2 Devices  
Device 0 mit 2 periodischen Realtime Programmen (RPROG1,RPROG2) und einem interrupt gesteuerten RPROG  
Device 1 mit 2 periodischen RPROGs und einem weiteren periodischen RPROG in separatem Thread 1

## Speicherlayout:



prinzipielles Speicherlayout einer  
PLC mit mehreren Pages

## Speicherlayout:



Aufbau einer Page

## Speicherlayout:

**Aufbau des universellen Teils  
des Shared Memorys**

int inUse
STACKS
HEAP
CODE
EHEAP
EHEAP_IND
CPU Register
Transferliste
BCB

## API-Funktionen:

- derzeit ca. 120 Funktionen
- Unterteilung in unterschiedliche Gruppen
- Aufruf lokal oder remote über Telegramminterface
  
- remote Aufruf über TCP/IP oder jedes Kommunikationssystem, das verbindungsorientiert, transparent und sicher arbeitet

## API-Funktionen:

### 1. elementare Systemfunktionen

- `int orf_shm_open()`
- `SharedPA_typ *orf_get_pa_ptr()`
- `int orf_tcp_open()`
- `int orf_create_rprog()`
- `int orf_start_plc()`
- `int orf_start_device()`

## API-Funktionen:

### 2. Zugriff auf die Shared-Memory Pages

- `int read_device_config()`
- `int read_stack_bytes()`
- `int write_heap()`
- `int read_userdata_bytes()`
- `int write_userdata_bytes()`

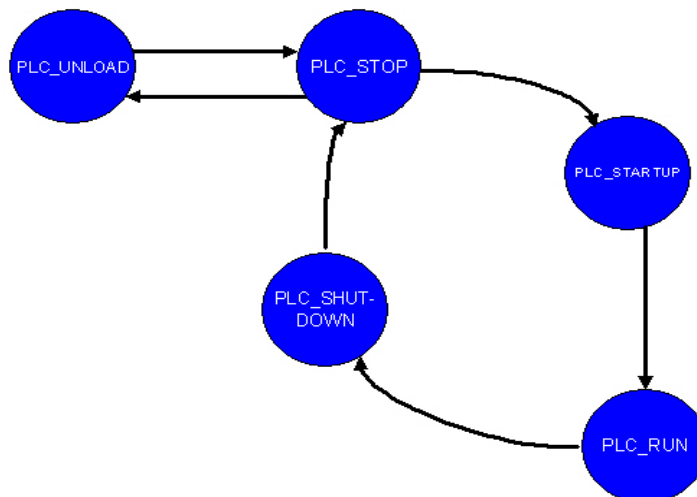
## API-Funktionen:

### 3. Funktionen für Online Debugging

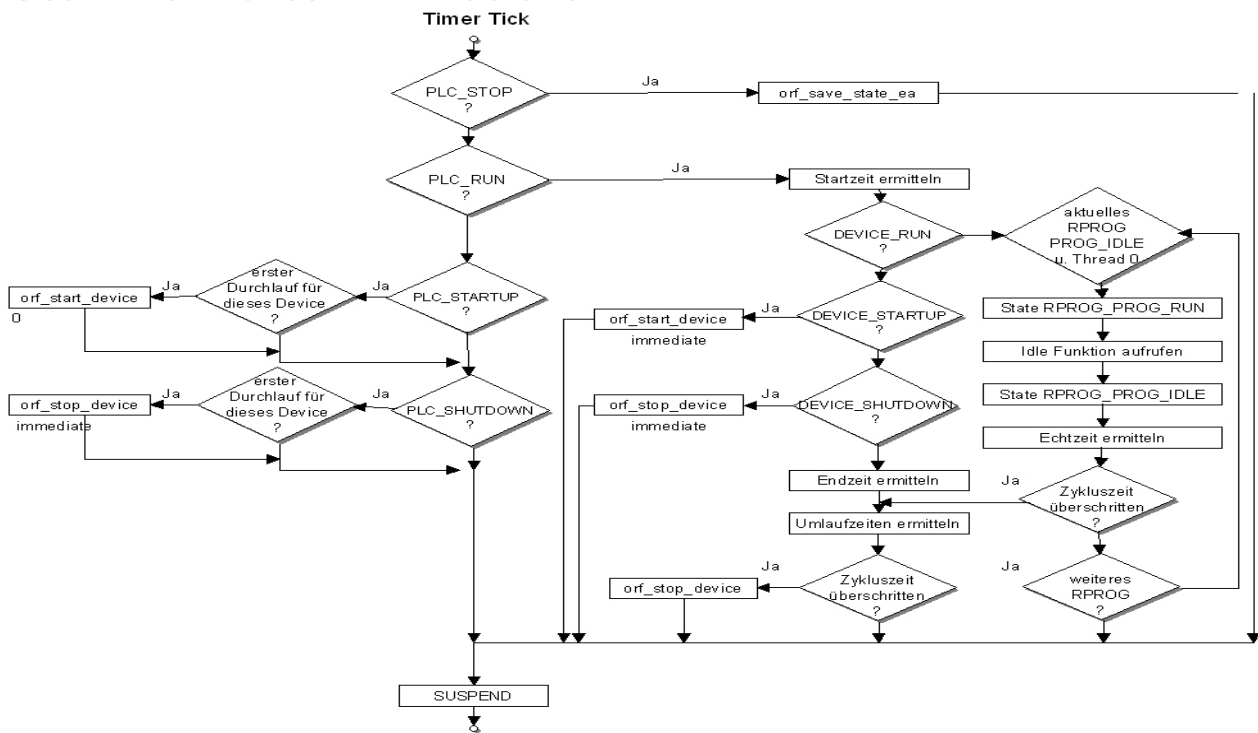
- int write\_breakpoint()
- int activate\_watch\_once()
- Int read\_watch\_all()
- int read\_codetrace\_buffer()

## Systemverhalten:

Definiertes Systemverhalten der einzelnen ORF-Komponenten



## Systemverhalten Thread 0:



## ToDo's für nächste Version der ORF- Spezifikation:

- Ausnahmeereignisse und Reaktion
- standardisiertes I/O Handling
- API für RPROGs
- API für UPROGs
- to be continued ....

# ORF - Open Realtime Framework



Yellowstone Soft  
Brunnenstr. 32  
89584 Ehingen

[www.yellowstone-soft.de](http://www.yellowstone-soft.de)



- Gründung 1988
- strategische Entscheidung für Linux 1998
- Produkte für embedded Linux seit 1999
- Initiierung Open Realtime Framework 2005

# ORF - Open Realtime Framework



**Vielen Dank  
für Ihre  
Aufmerksamkeit**

**Ihre Fragen?**