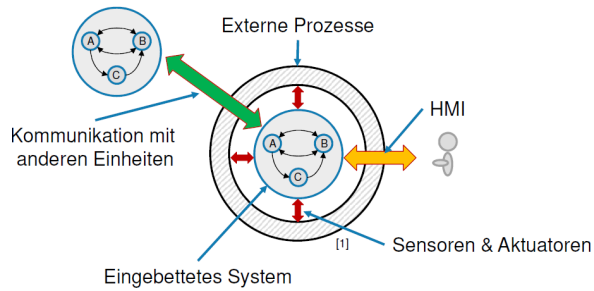


# Embedded Systems

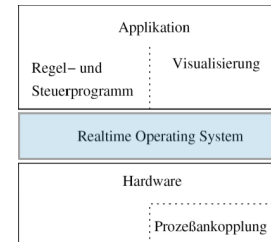
## Einführung

### Eingebettetes System



- integrierte, elektronische Schaltung mit spezifischer Aufgabe, mit der ein Benutzer nur indirekt in Verbindung kommt
- Teil eines Gesamtsystems mit stark beschränkten Ressourcen, bestehend aus Hard- und Software (teilweise ohne Betriebssystem)
- HW/SW-Codedesign z.B. mit *VHDL*, allgemein *tool-* bzw. *modellbasierter* Entwurf
- 75 % verwenden ein Betriebssystem (Tendenz steigend)
  - 25 % mit *Main-Loop*

- Ist eine Kombination aus Hard- und Softwarekomponenten, die in einen technischen Kontext zur *Steuerung, Regelung* und *Überwachung* eines Systems eingebunden sind
- Es verrichtet vordefinierte Aufgaben, oftmals mit *Echtzeitberechnungs*-Anforderungen
- **Speicherprogrammierbare Steuerung (SPS)**: Verwendet zur Fabrikautomatisierung, Verkehrsleitung
- **Standardarchitektur** auf einem PC: Preiswerte Hardware (allerdings oft nicht *industrietauglich*) preiswerte Software, häufig ohne *Echtzeitfähigkeit*
- **Industrie-PC**: Unterstützt Echtzeitbetriebssysteme



### Definition: *Technischer Prozess*

- Prozess, in dem Zustandsgrößen durch *technische* Hilfsmittel festgestellt und beeinflusst werden
  - *Prozess* definiert als Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Information verändert wird
- *Sensoren* (z.B. Thermometer, Kamera, Mikrophon) erfassen Zustandsgrößen, *Aktoren* (z.B. Motoren, Relais, Ventile) beeinflussen sie

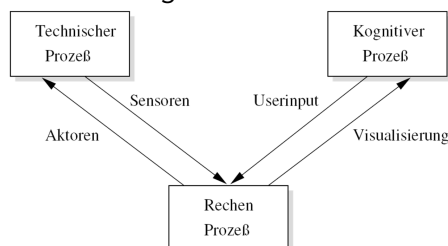
### Klassifikation: *Technischer Prozess*

- **Fließprozess (Regler)**: physikalische Größe mit stückweise kontinuierlichem Wertebereich, ablaufende Vorgänge sind zeit- und ortsabhängig, z.B. chemische Reaktoren, Energieerzeugung in Kraftwerken
- **Folgeprozess (State Machine)**: Binäre, diskrete Informationselemente werden gemeldet oder ausgelöst, z.B. Ampel- oder Aufzugsteuerung
- **Stückprozess (Datenbank)**: Informationselemente werden einzeln identifizierbaren Objekten (Stücken) zugeordnet z.B. Transport- oder Ladevorgänge, Fertigung

### Definition: *Rechenprozess*

- *Task* als Instanz zur dynamischen Abarbeitung eines Programms zur Berechnung von Ausgabewerten aus Eingabewerten über Umformen, Transportieren oder Speichern von Information

### Definition: *Kognitiver Prozess*



- Umformen, transportieren oder verarbeiten von Information im menschlichen Bediener
- Einflussnahme des Bedieners auf den Rechenprozess über *Man Machine Interface (MMI)*

### Definition: *Steuerungssystem*

- Umfasst zur Steuerung erforderliche Rechenprozesse sowie deren Hard- bzw. Software
- Aufgaben:
  - Erfassen von Zustandsgrößen
  - Koordinaten & Überwachung der Prozessabläufe

### Definition: *Steuerung und Regelung*

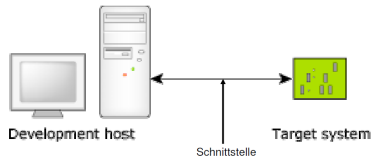
- **Steuerung**: Kein geschlossener *Regelkreis*, Rechenprozess reagiert nicht auf sich ändernde Sensorwerte im technischen Prozess
- **Regelung**: Geschlossener *Regelkreis*, Sensor- bzw. Messwerte werden verwendet, um Stellgrößen daraus zu berechnen

## Self-Hosted-Entwicklung

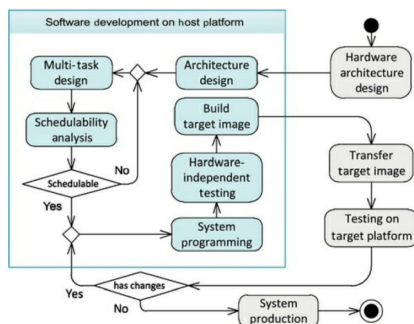
- Entwicklungsumgebung und Zielsystem sind identisch (so wie wir es alle kennen)

## Host-Target-Entwicklung

- Self-Hosted-Entwicklung* oft nicht möglich da Hardware proprietär oder zu leistungsschwach für Entwicklungsumgebung
  - Host:** Entwicklungsrechner, enthält *Cross-Compiler*, *Remote-Debugger*, *Target-Libraries* und -Betriebssystem
    - Cross-Compiler:* Erzeugt *Image*, dass eigentliche Applikation sowie Betriebssystem- und Laufzeitkomponenten + *Startupcode* enthält
    - Remote-Debugger:* Auf dem Host läuft GUI mit *Debug-Info*, über *JTAG* etc. sieht man den Systemzustand des Targets (*Stack*, *Variablenbelegung*) an gewählten *Breakpoints*
    - Ohne Remote-Debugger:* Konsolenausgaben per *printf*, *LEDs* blinken lassen
  - Schnittstelle:** Zum *Downloaden* der Applikation auf das *Target* oder fürs *Debugging*, verschiedenste Variationen möglich (z.B. *Ethernet*, *USB*, *JTAG*, *Flash*, ...)
  - Target:** System, für das entwickelt wird
    - Boot-Monitor:* Programm auf dem Target, über das Software geladen und gestartet werden kann, erfolgt über ähnliche Schnittstellen wie die *Host-Target-Entwicklung* an sich



## Softwareentwicklung in einem Host-Target System



- Object File:** Symboltabelle
- Linker:** Symbol-Auflösung und *Relocation*
- Executable File:** Code & Daten zur Ausführung, Umsetzung auf virtuellen Speicher
- Shared object file:** Code und Daten zum (dynamischen) linken mit anderen *object files*
- Relocatable file:** Code und Daten zum linken mit anderen *object files* um Executable zu erstellen
- Dynamic Linker:** Laden von *shared Libraries*

Tools: *Kemel-Tracer*

- Zeigt *Signale*, *Task-Zustände*, *Semaphoren*, *Interrupts*

Tools: *Stack-Monitor*

- Zeigt maximal verfügbarer *Stack* pro *Task*, aktuelle Auslastung und maximale je gemessene Auslastung (*Hochwassermarken*) des Stacks

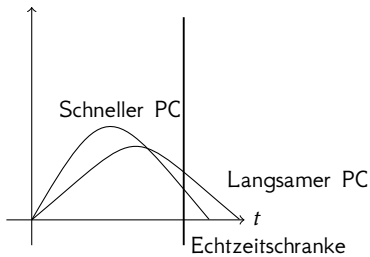
Weitere Tools

- Anzeige der Speicherbelegung und *Auslastung* der *CPU*, *Memory-Leak-Detection*, *Code-Coverage*

## Echtzeitbetrieb

### Kriterien für Echtzeitsysteme

Auslastung

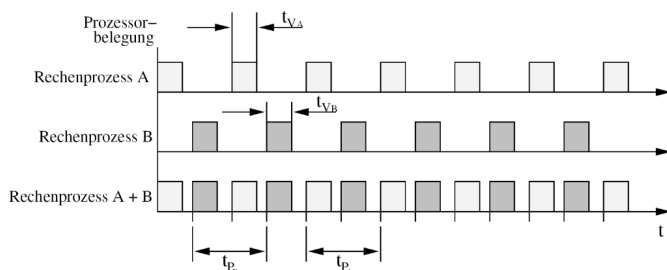


- Schnelligkeit bzw. Geschwindigkeit ist **nicht** wichtig im Kontext einer harten Echtzeitschranke ein schnellerer PC ist zwar häufiger vor der Schranke fertig, aber eben auch nicht zu 100%
- Wichtig dagegen sind:
  - Pünktlichkeit (*Ober- und Untergrenze*) bzw. Rechtzeitigkeit (*Nur Obergrenze*) (*timeliness*)
  - Verfügbarkeit
  - Determinismus (bei gleicher Eingabe im gleichen Zustand liefert das System immer die gleiche Ausgabe)
- Verletzungen von Zeitbedingungen ggf. katastrophal (fristgerechte Bearbeitung von Anforderungen aus einem technischen Prozess)

### Vorbedingungen für Echtzeitbetrieb

- Verarbeitungszeit von Aufgaben berücksichtigen, bei mehreren Aufgaben Reihenfolge der Abarbeitung planen
- Reihenfolge entscheidend für fristgerechte Ergebnisse
- Priorität von Aufgaben gemäß ihrer Wichtigkeit als Planungsgrundlage
- Unterbrechung einer Aufgabe muss durch einen Prozess zur Bearbeitung höher-priorer Aufgaben möglich sein  
⇒ Formaler Rahmen zum Nachweis schritthaltender Verarbeitung

### Echtzeitbedingung: Auslastung



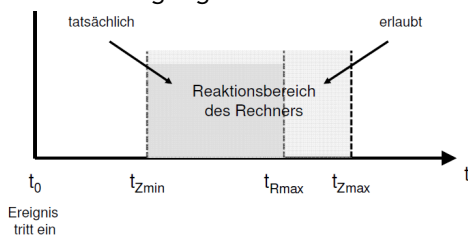
- $t_V$  := Verarbeitungszeit
- $t_P$  := Prozesszeit, Abstand zwischen zwei Anforderungen (*Jobs*) desselben Typs, wenn  $t_P$  konstant handelt es sich um einen *zyklischen* bzw. *periodischen* Prozess
- $\rho = \frac{t_V}{t_P}$  := Auslastung
  - $\rho_A = \frac{t_{VA}}{t_{PA}}$
  - $\rho_B = \frac{t_{VB}}{t_{PB}}$
  - $\rho_{A+B} = \frac{t_{VA}}{t_{PA}} + \frac{t_{VB}}{t_{PB}}$
- $\rho = \sum_{i=0}^n \frac{t_{Vi}}{t_{Pi}}$  := Gesamtauslastung bei  $n$  Prozessen

- **1. Echtzeitbedingung:** Gesamtauslastung aller Prozesse  $\leq 1 \Leftrightarrow \rho = \sum_{i=0}^n \frac{t_{Vi}}{t_{Pi}} \leq 1$

### Art von Rechenprozessen

- *zyklisch*: konstanter Abstand zwischen zwei Anforderungen
- *azyklisch*: Keine Untergrenze zwischen zwei Nachrichten, kommen beliebig (*gefährlich*)
- *sporadisch*: ähnlich wie *azyklisch* aber mit Untergrenze zwischen zwei Nachrichten (z.B. *Netzwerktreiber*)

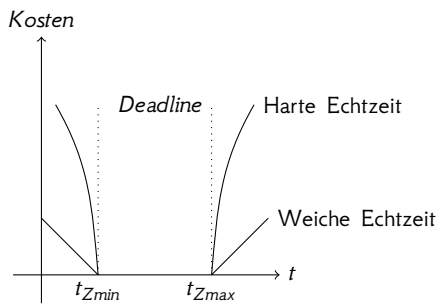
### Echtzeitbedingung: Pünktlichkeit



- Aufgabe darf nicht vor spezifizierten Zeitpunkt  $t_{Zmin}$  erledigt sein (meist unwichtig oder *trivial*)
- Aufgabe muss spätestens bis Zeitpunkt  $t_{Zmax}$  erledigt sein (Rechtzeitigkeit)
- Verbleibende Reaktionszeit:  $t_R = t_V + t_W$  (Verarbeitungszeit + Wartezeit)  
*Wartezeit* := Zeit, bis Rechenkern frei ist

- **2. Echtzeitbedingung:** Um Aufgaben rechtzeitig zu erledigen, muss die Reaktionszeit zwischen der minimal und maximal zulässigen Reaktionszeit liegen:  $t_{Zmin} \leq t_{Rmin} \leq t_R \leq t_{Rmax} \leq t_{Zmax}$

## Harte und weiche Echtzeit



- **Harte Echtzeit:** Verletzung der Rechtzeitigkeit hat *katastrophale* Folgen (z.B. Airbag, Herzschrittmacher)
- **Weiche Echtzeit:** Schlechteres Ergebnis (z.B. *ruckelnde* Videowiedergabe, GPS-Latenz) ⇒ Häufig *Graubereich*
- **Kostenfunktion:** Kosten *explodieren* bei Überschreitung der Echtzeitschranke (**Deadline**) im Falle von Harter Echtzeit, bei Weicher Echtzeit steigen Kosten nur *linear* an

- **Nutzenfunktion:** Ergebnisse außerhalb des Intervalls  $[t_{Zmin}, t_{Zmax}]$  haben bei Harter Echtzeit nahezu *keinen* Nutzen mehr, bei Weicher Echtzeit ungefähr *lineare* Ab- bzw. Zunahme

