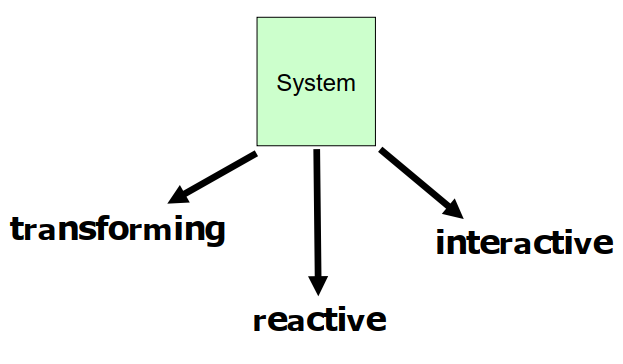
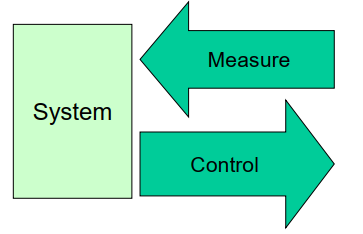
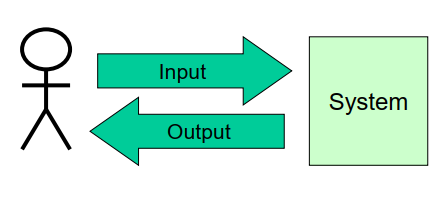
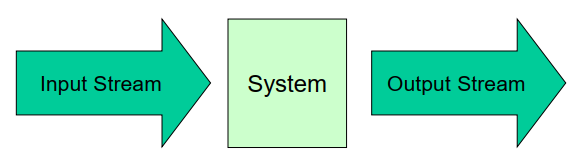
### Systeme







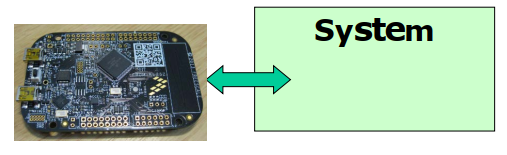
|  |  |  |
| --- | --- | --- |
| **Transforming** | **Reactive** | **Interactive** |
| Möglichst keine Latenz | Möglichst kurze Latenz | Kurze Latenz |
| mehr Speed 🡪 mehr Speicher | **x** | Hohe Systemlast |
|  |  |  |
| **Anwendungsbeispiele** | | |
| AD-Wandler | Echtzeitregelungssysteme | Billettautomat |
| Com.-Schnittstellen | Airbag | HMI Schnittstellen allg. |
| Encoder |  |  |

**Ein System tritt meistens als Kombination von transitiven, reaktiven resp. interaktiven Teilsystemen auf**

### Echtzeit für Computer Systeme

**Definition Echtzeitsysteme:**

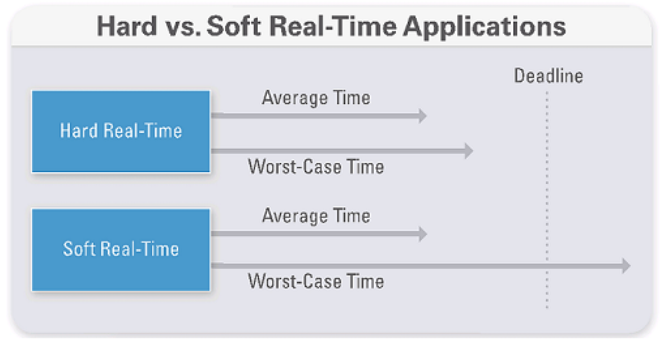
Unabhängig der Systemlast resultiert nach einer Systeminteraktion (event) innerhalb einer festgelegten Zeit aus einem definierten Zustand ein erwarteter, korrekter Folgezustand



* Computer / MCU muss sich an der «richtigen» Zeit orientieren
  + In ms anstatt Anzahl Ticks
* Der Folgezustand darf weder zu früh noch zu spät eintreten
* Die Systemlast wird unter anderem durch die Anzahl Nebenläufigkeiten, Intervall der Events, Reaktionszeit auf die Events und der Abarbeitungszeit der Events bestimmt
* Je kürzer die Reaktionszeit sein soll, desto grösser wird die Systemlast

**Claims**

|  |  |
| --- | --- |
| Rechtzeitigkeit (Timeliness) | Nebenläufigkeit (Concurrency) |
| System auf die Bereiche **Input**, **Process** und **Output** aufteilen und jeweils die Realtime-Anforderung einzeln betrachten/beweisen  *Unterscheidung:*  Absolut 🡪 zu einem best. Zeitpunkt  Relative 🡪 innerhalb einer best. Zeit | Bei mehreren Aufgaben 🡪 Multitasking, Nesting  **Aber**: nicht für viele/ schnelle tasks geeignet |



Degradation

Quelle: <http://www.ni.com/>

**Hard Real-Time:** garantiert, dass eine Deadline eingehalten wird

**Soft Real-Time:** periodisches Überschreiten der Deadline möglich. 🡪 Degradierung

Beispiele

**Hard Real-Time :** Airbag

**Soft Real-Time:** Videoplayer (Degradation: verpixelung/verzerrung der Darstellung)

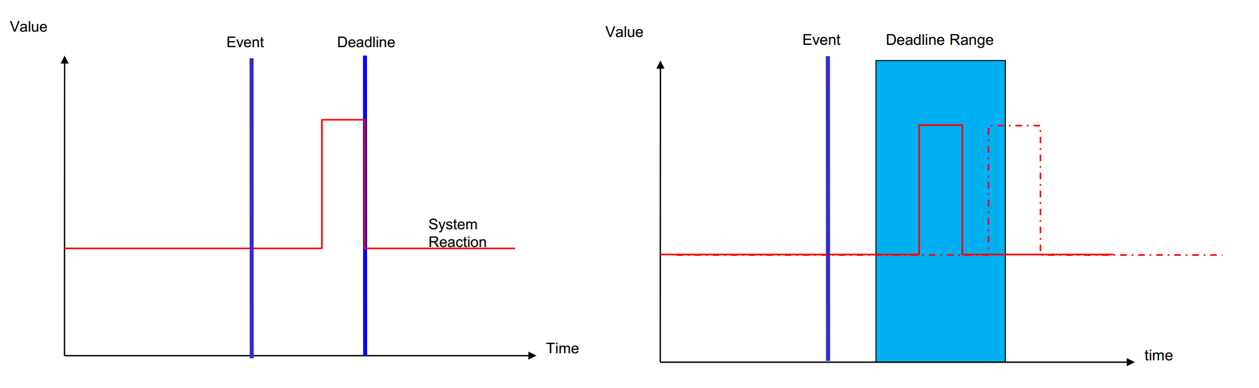
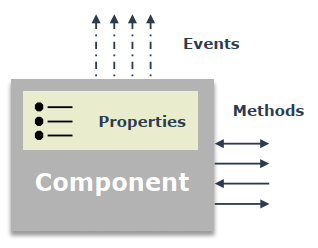
****

Abbildung Hard & Soft Real-Time

### Processor Expert

* Softwarekomponente, Middleware
* Virtueller Programmierer
* Kann auch Dokumentation generieren
* Nachteil: besondere Behandlung bei der Nutzung von PE in einem VCS
  + Projekt speichern, Projekt schliessen, Pull, Projekt öffnen

# Embedded Components



Methoden:

* Enabled/disabled usw.

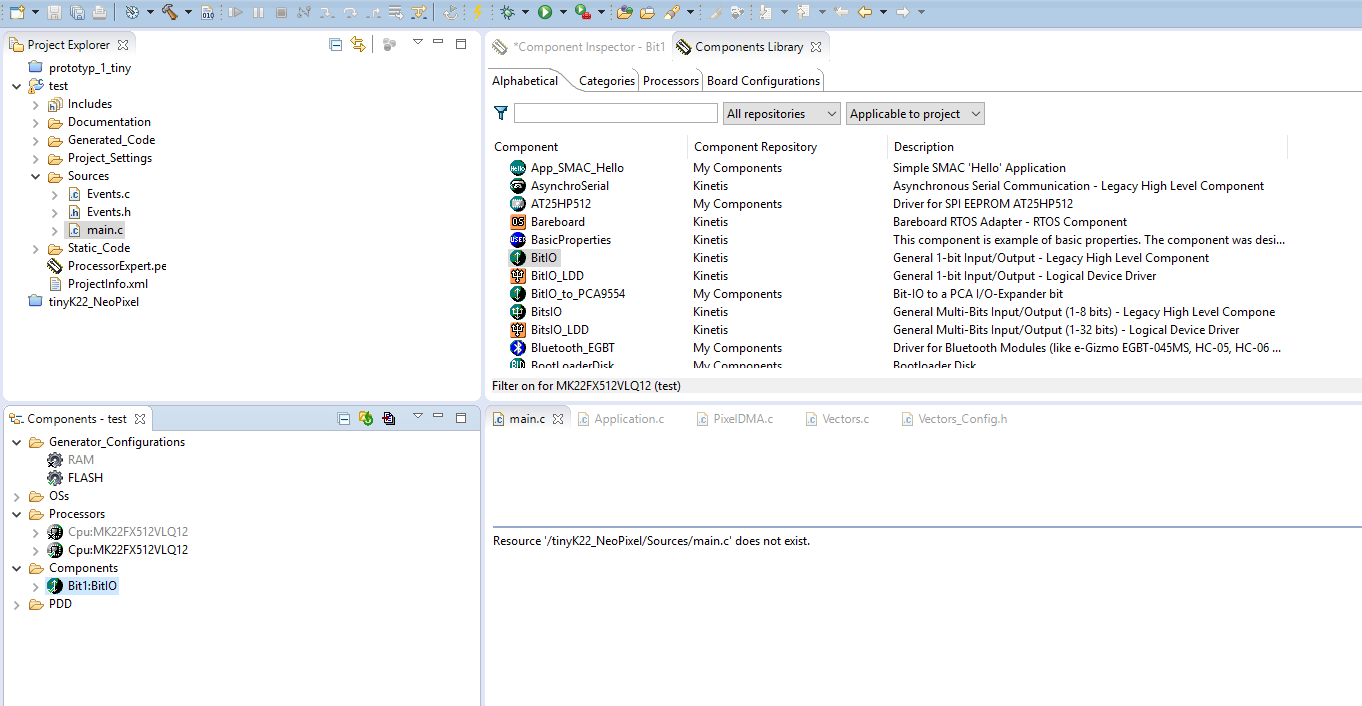
Events:

* Implementation der ISR
* *Events.c* resp *Events.h*

Properties:

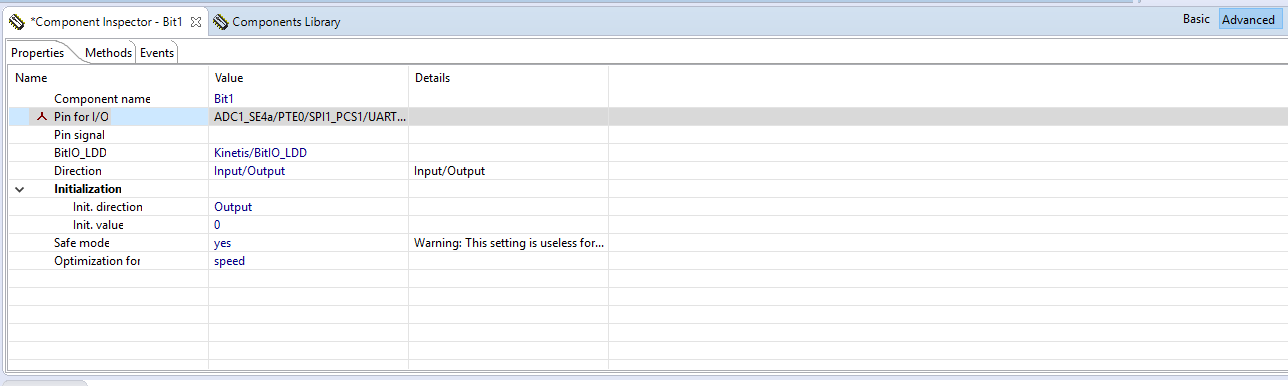
* Zb pulse-width, duty-cycle bei einem PWM

# Betty Bossi



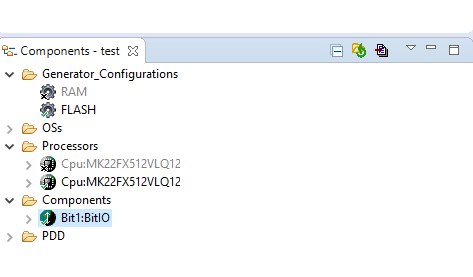
Doppelklick oder

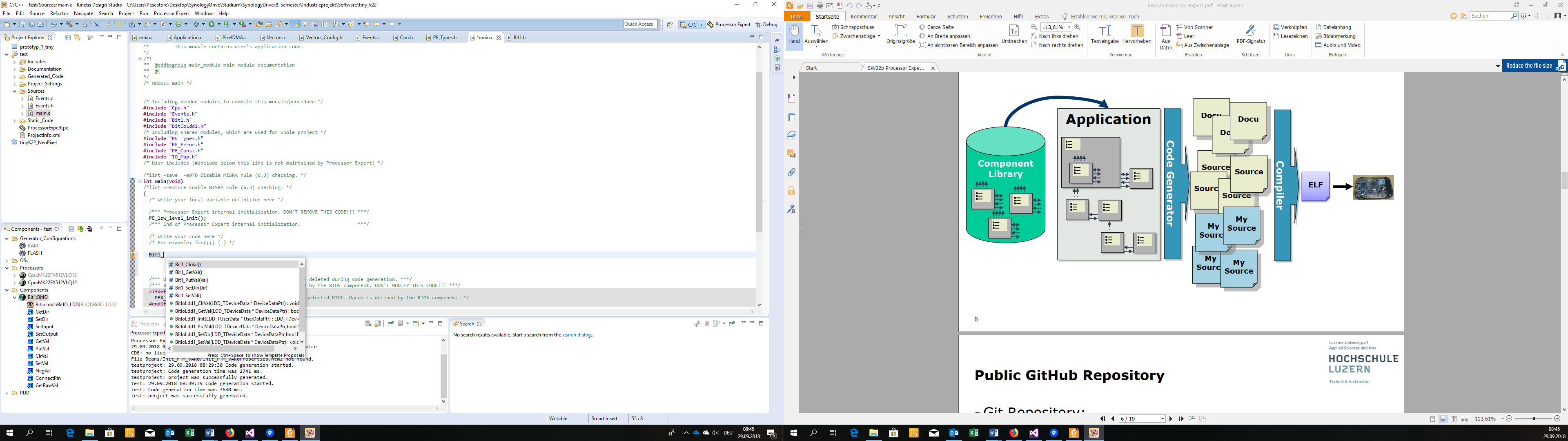
drag and drop



Einstellungen vornehmen im Component Inspector

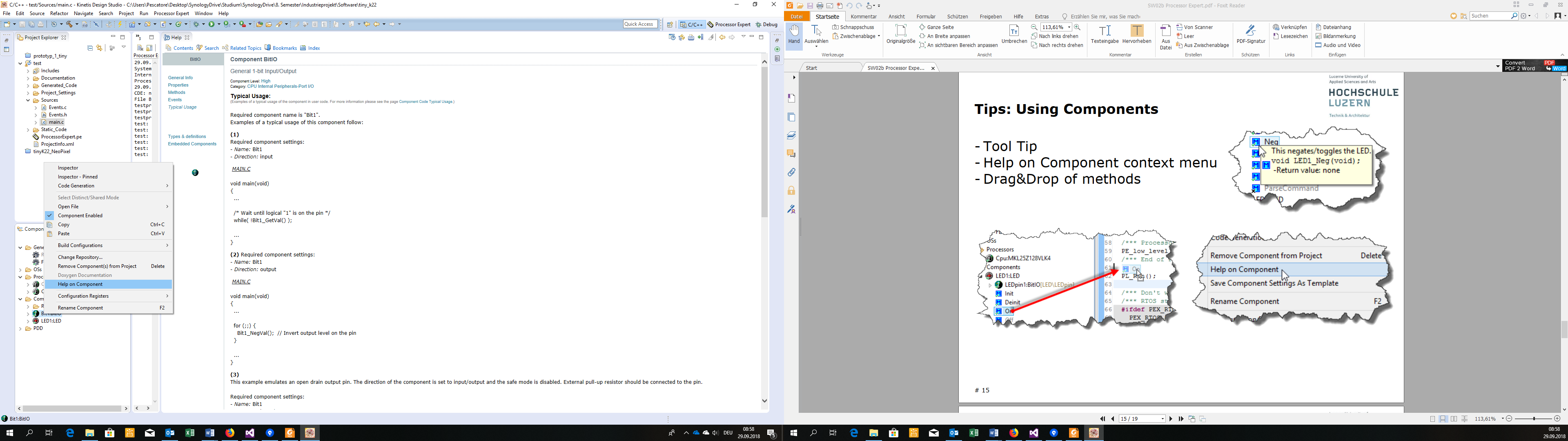
Generate Code





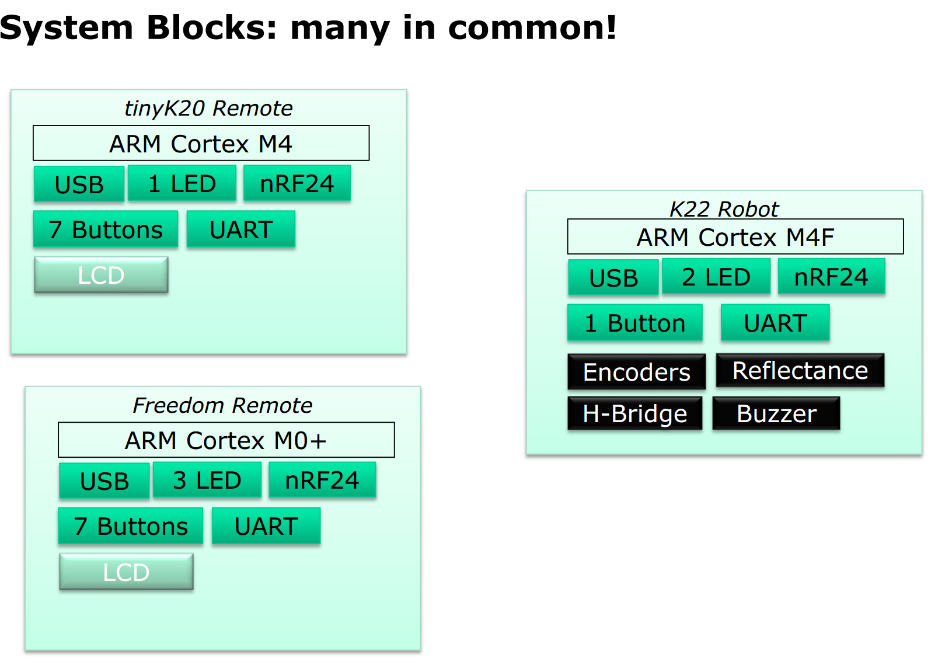
Mit ctrl + Leertaste sichtbar

Gut zu wissen - Help on Component



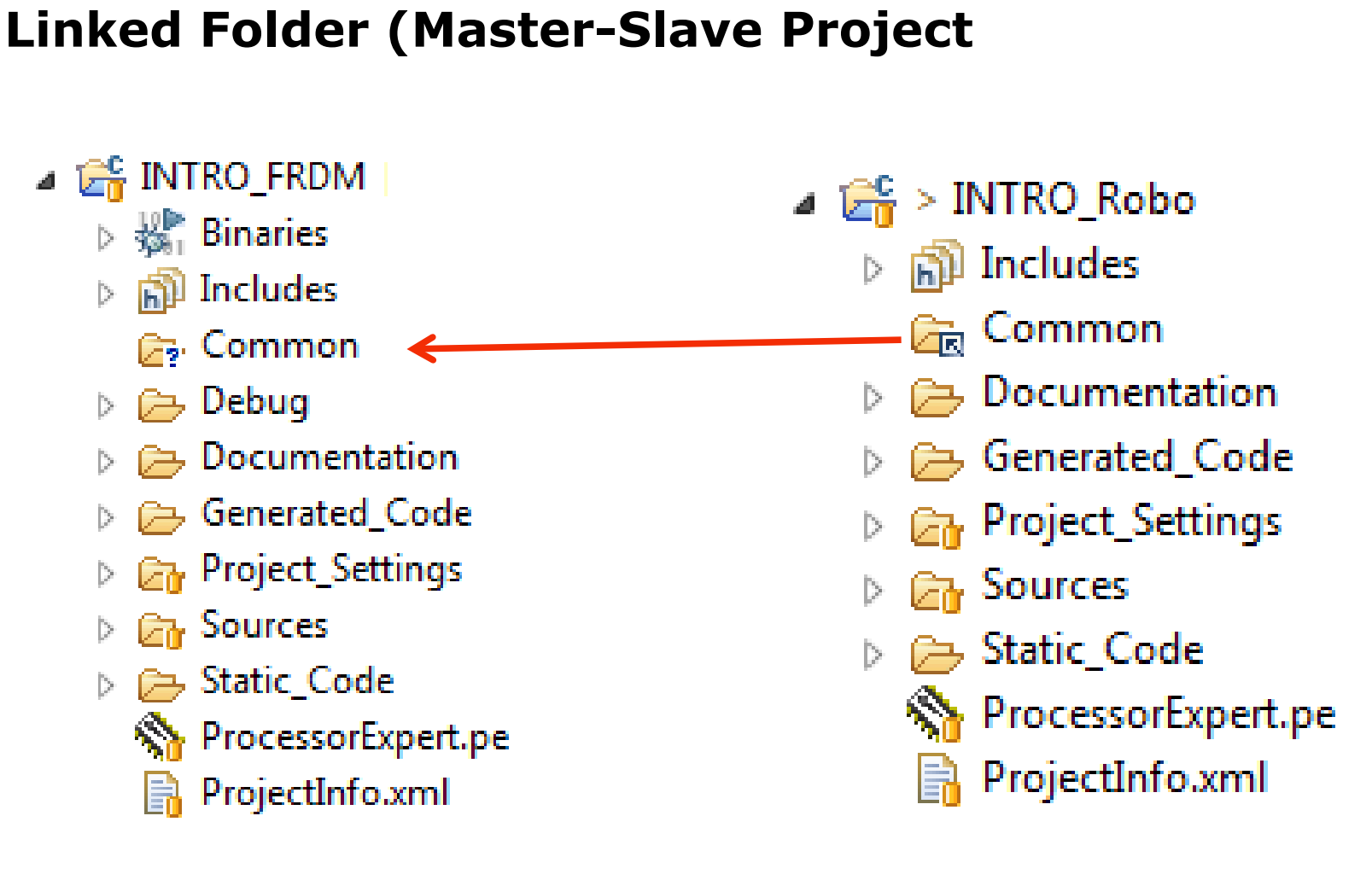
### Project Structure

Ziel: - System Blöcke, welche bei einzelnen Systemteile gemeinsam sind auch gemeinsam nutzen:

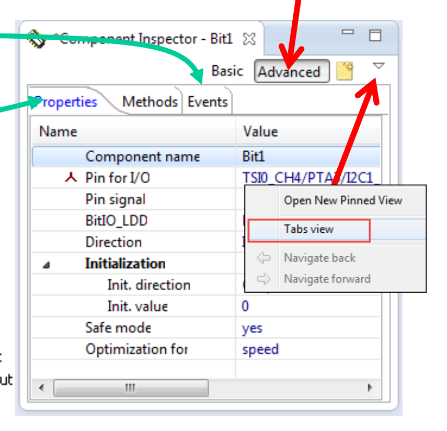


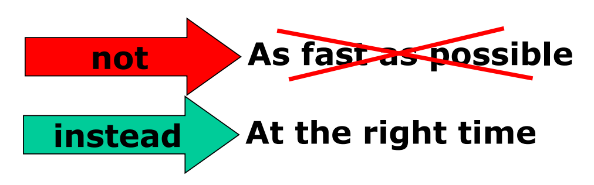
Vorteil: - Weniger Entwicklungsaufwand

Umsetzung: - Beide Teilprojekte verlinken auf den gemeinsamen Ordner:



# TIPPS

* Component Inspector “richtig” einstellen: «Tabs view» & «Advanced».
* Real-Time heisst nicht «so schnell wie möglich» sondern «zum

****Richtigen Zeitpunkt».

* Nutzt Tool Tipp und Help on Component.

# Für ein sorgenfreies Arbeiten im Team

* Definieren wer welche Änderungen macht
* Regeln aufstellen:
  + A ist fertig mit der Änderung 🡪 commit 🡪 push
  + B löscht .pe file, schliess sein Projekt und macht dann ein pull
  + Bei B wird dann das .pe file von A generiert

# Quizfragen

* Zwischen was für drei Systemtypen kann unterschieden werden?
* Was ist versteht man unter Hard-Real-Time?
* Auf was muss geachtet werden, wenn man Processor Expert und ein VCS benutzt?
* Was ist die logische Konsequenz bezüglich Systemlast, wenn man eine kürzere Reaktionszeit erreichen will?
* Was für Vorteile bringen «Shared Resources»?