Modellierung von Echtzeitsystemen

Reaktive Systeme

Werkzeuge: SCADE, Esterel Studio

Fragen zur letzten Vorlesung

- Wieso wird Rekursion in Esterel nicht unterstützt?
- Wie funktioniert die Umsetzung des Await-Statements?

```
module Temperature:
input IN1,IN2;
output OUT1,OUT2;
    loop
      await IN1; emit OUT1;
    end loop;
    ||
    loop
      await IN1; emit OUT1;
    end loop;
```

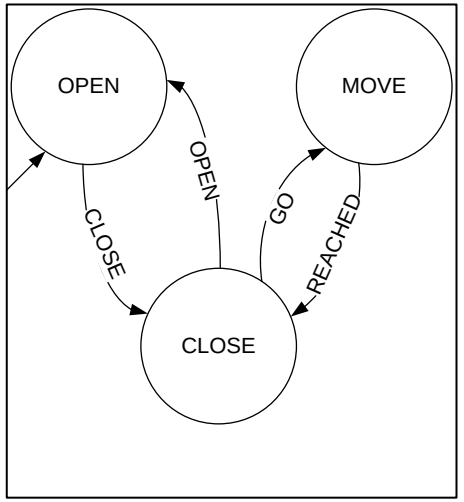
Exkurs: Automaten zur Modellierung von reaktiven Systemen

Automaten im Kontext von reaktiven Systemen

- Durch graphische Darstellung (z.B. Automaten) kann die Verständlichkeit des Codes stark verbessert werden.
- Ein reaktives System kann durch die zyklische Ausführung folgender Schritte beschrieben werden:
  1. Lesen der Eingangssignale
  2. Berechnung der Reaktionen
  3. Auslösen der Ausgangssignale
- Die zyklische Ausführung kann im Automatenmodell wie folgt interpretiert werden:
  1. Lesen der Eingabe im aktuellen Zustand
  2. Berechnen der Zustandsübergangsfunktion und ggfs. Zustandswechsel
  3. Erzeugung von Ausgangssignalen (abhängig von alten Zustand und gelesenem Eingangssignal)
- Die kennen gelernte Synchronitätshypothese bedeutet im Bezug auf den Automaten der im Vergleich zur Zeit für Änderungen der Umgebung eine vernachlässigbare Zeit für die Berechnung der Zustandsübergange und Ausgabefunktion.
- Pro Runde wird im Allgemeinen (1 Ausnahme) genau ein Zustandsübergang berechnet.

Endliche Automaten

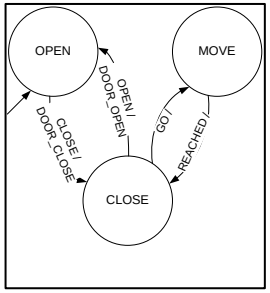
- Ein klassischer endlicher Automat $(Q, \Sigma, \delta, s_0, F)$ ist eine endliche Menge von Zuständen und Zustandsübergängen mit:
  - endlicher Menge von Zuständen Q
  - endliches Eingabealphabet Σ
  - Übergangsfunktion $\delta : Q \times \Sigma \rightarrow Q$
  - Menge von Endzuständen $F \subseteq Q$



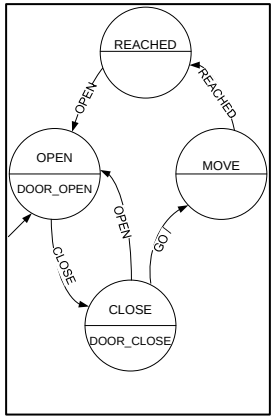
- Um die Verständlichkeit zu verbessern, werden nur deterministische Automaten modelliert, also $|S|=1$ und $\delta(q,\alpha)=q' \wedge \delta(q,\alpha)=q'' \Rightarrow p'=q''$
- Problem bei der klassischen Definition von Automaten: Ausgaben können nicht modelliert werden.

Automaten mit Ausgaben

- Mealy- und Moore-Automaten unterstützen die Ausgabe von Signalen
- Die Ausgaben von Mealy-Automaten ($Q,\Sigma,\Omega, \delta, \lambda, q_0,F$) sind dabei an die Übergänge gebunden mit
  - $Q, \Sigma, \delta, q_0, F$ wie bei klassischem Automat
  - Ausgabealphabet Ω
  - Ausgabefunktion λ: $Q \times \Sigma \rightarrow \Omega$



- Bei Moore-Automaten ($Q,\Sigma,\Omega, \delta, \lambda, q_0,F$) ist die Ausgabe dagegen von den Zuständen abhängig:
  - $Q, \Sigma, \Omega, \delta, q_0, F$ wie bei Mealy-Automat
  - Ausgabefunktion λ: $Q \rightarrow \Omega$
- Moore- und Mealy-Automat sind gleich mächtig: sie können ineinander konvertiert werden.



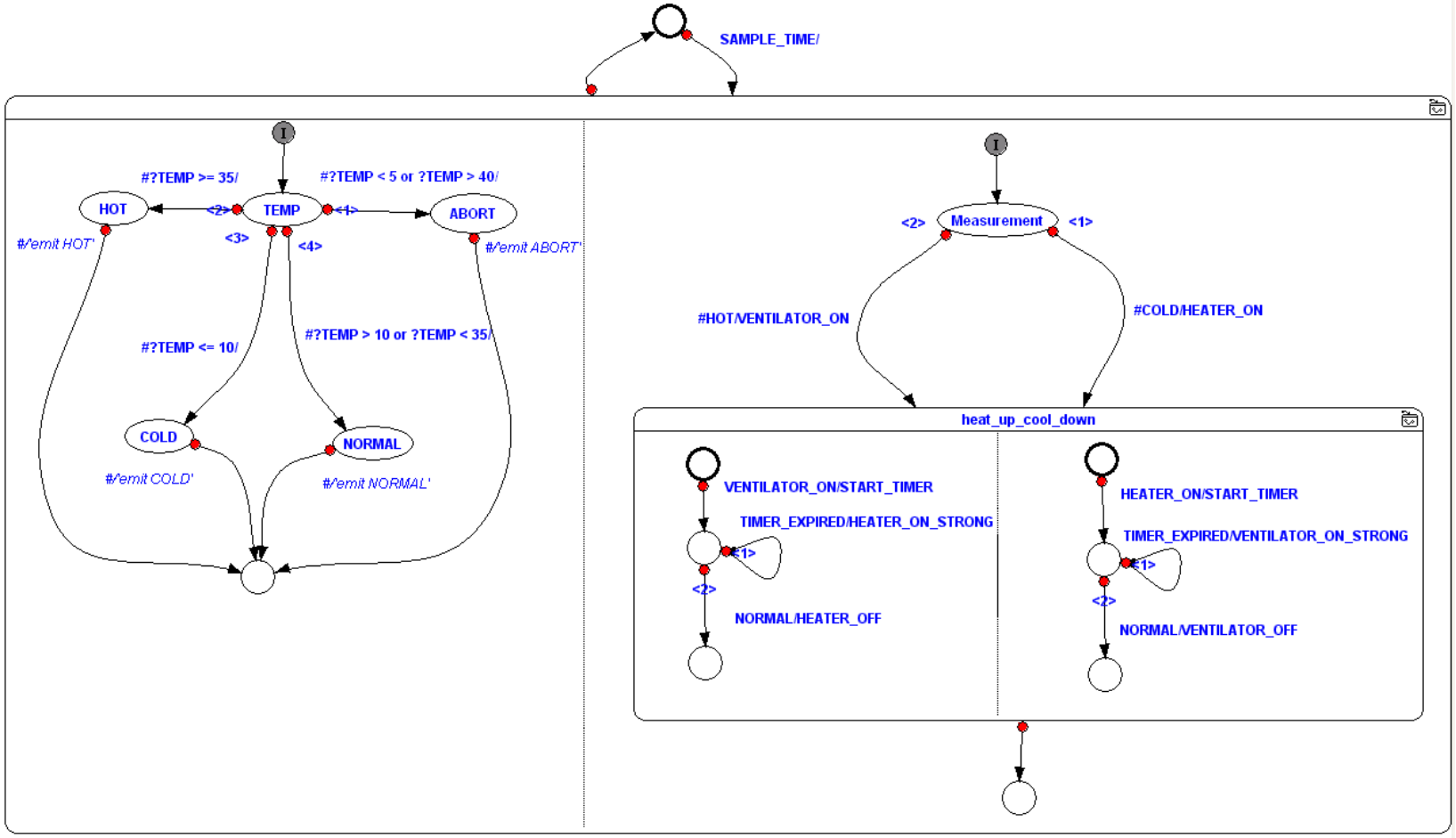
Harel-Automaten / Statecharts

- Heutiger Standard zur Beschreibung von reaktiven Systemen sind die von David Harel 1987 vorgeschlagenen Statecharts.
- Statecharts zeichnen sich durch folgende Eigenschaften aus:
  - Vereinigung der Eigenschaften von Mealy- und Moore-Automaten
    - Ausgaben in Abhängigkeit von Zustandsübergängen möglich
    - Durchführung von Ausgaben beim Erreichen eines Zustands (**`onExit`**) oder Verlassen eines Zustandes (**`onEntry`**)
  - Zur Erhöhung der Lesbarkeit: Hierarchische Strukturierung von Teilautomaten möglich inkl. Gedächtnisfunktionalität
  - Darstellung paralleler Abläufe durch parallele Teilautomaten.
  - Verknüpfung von Zuständen mit Aktionen: Befehle **`do`** (zeitlich begrenzte Aktivität), **`throughout`** (zeitlich unbegrenzte Aktivität)
  - Einführung spontaner bzw. überwachter (guarded) Übergänge

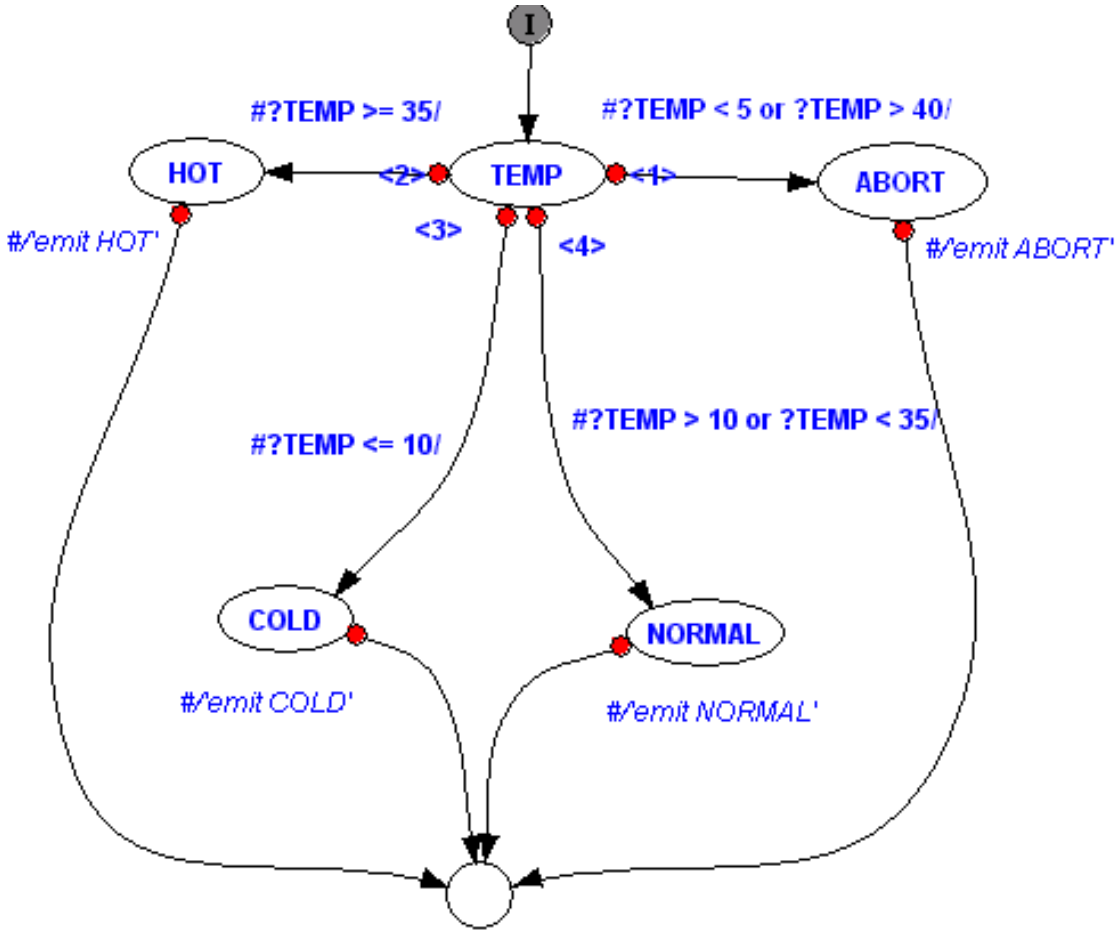
Safe State Machine

- Esterel benutzt eine eigene Klasse von Automaten, die den Statecharts sehr ähnlich sind:
  - Vereinigung der Eigenschaften von Mealy- und Moore-Automaten
    - Ausgaben in Abhängigkeit von Zustandsübergängen möglich
    - Durchführung von Ausgaben beim Erreichen eines Zustands (**`onExit`**) oder Verlassen eines Zustandes (**`onEntry`**)
  - Zur Erhöhung der Lesbarkeit: Hierarchische Strukturierung von Teilautomaten möglich inkl. Gedächtnisfunktionalität
  - Darstellung paralleler Abläufe durch parallele Teilautomaten.
  - Verknüpfung von Zuständen mit Aktionen: Befehle **`do`** (zeitlich begrenzte Aktivität), **`throughout`** (zeitlich unbegrenzte Aktivität)
  - Einführung spontaner bzw. überwachter (guarded) Übergänge
  - Zusätzliche Esterel abhängige Konstrukte (z.B. `pre` Operator)

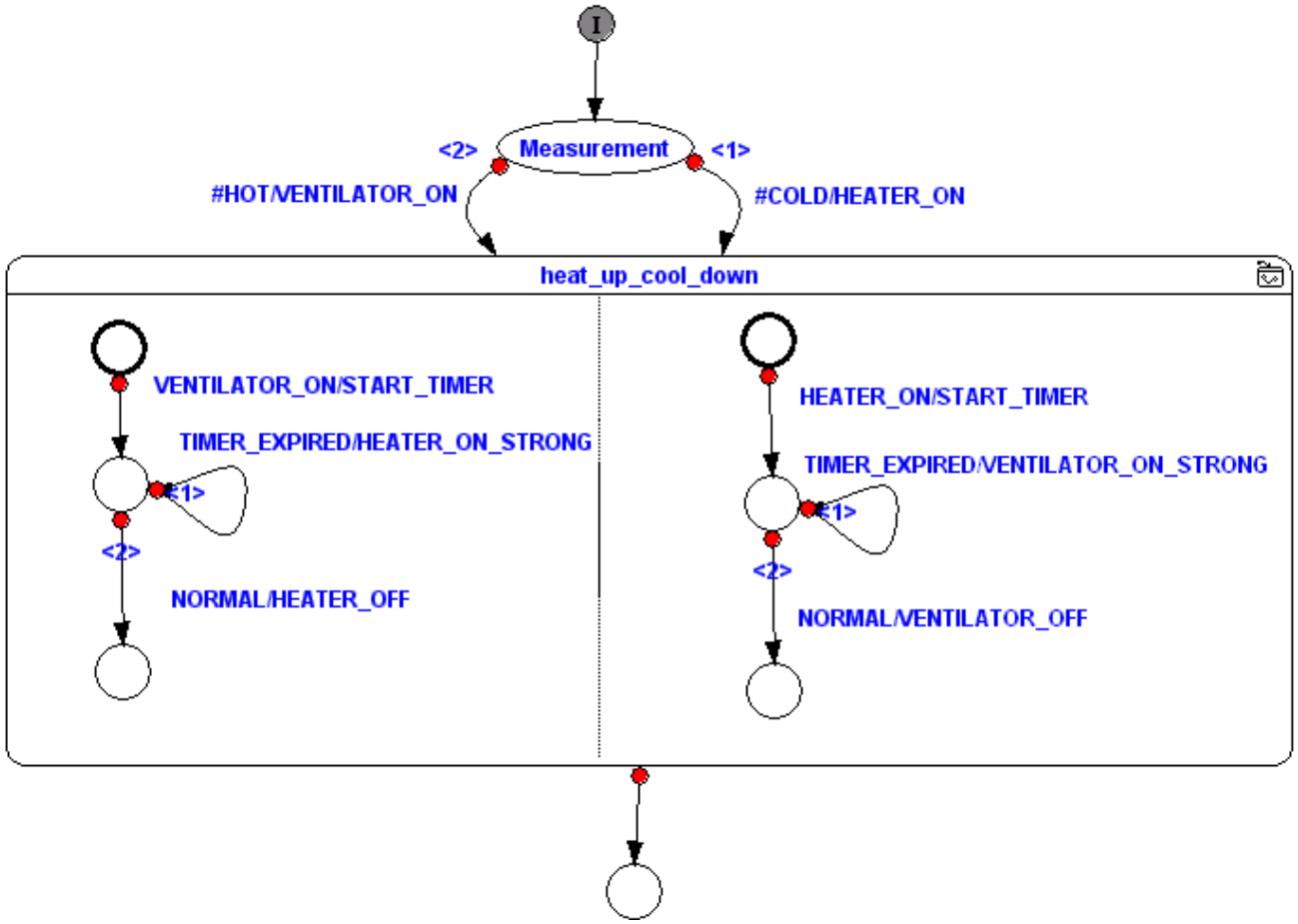
Beispiel als Automat



Beispiel als Automat – Teil 1



Beispiel als Automat – Teil 2



Modellierung von Echtzeitsystemen

Reaktive Systeme - Klausurfragen

Werkzeuge: SCADE, Esterel Studio

Auszug aus Klausur WS 06/07

a) Vervollständigen Sie folgende Testfälle, so dass das Modul xxx diese Testfälle erfüllt:

1. T1=({D},___),(___,{F})
2. T2=({D},___),({D},___})
3. T3=(___,{E}),(___,{F}),(___,{})
4. T4=(_____),(___,{N})
5. T5=(_____),(___,{E}),(___,{E})

Zur Erinnerung: ({A},{B}),({C},{D}) bedeutet: im ersten Moment erfolgt das Ereignis A als Eingabe, die Reaktion des Moduls ist B, im zweiten Moment erfolgt das Ereignis C als Eingabe mit der Reaktion D.

```
module xxx:
    input U, D;
    output E,F,N;
    var V=1;
    loop
        await
          case U do
                    if(?V>0)
                              V:=V+1;
                    else
                              V:=V+1;
                              emit F;
          case D do
                    if(?V>0) then
                              V:=V-1;
                              emit E;
                    else
                              emit N;
        end await;
    end loop;
```

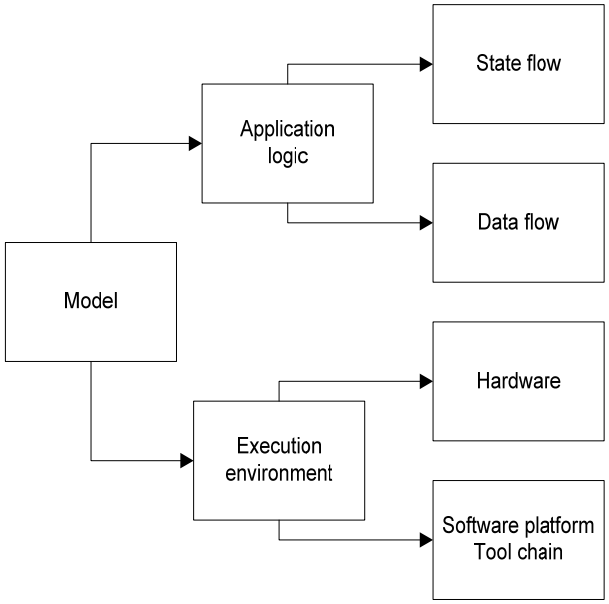
Modellierung von Echtzeitsystemen

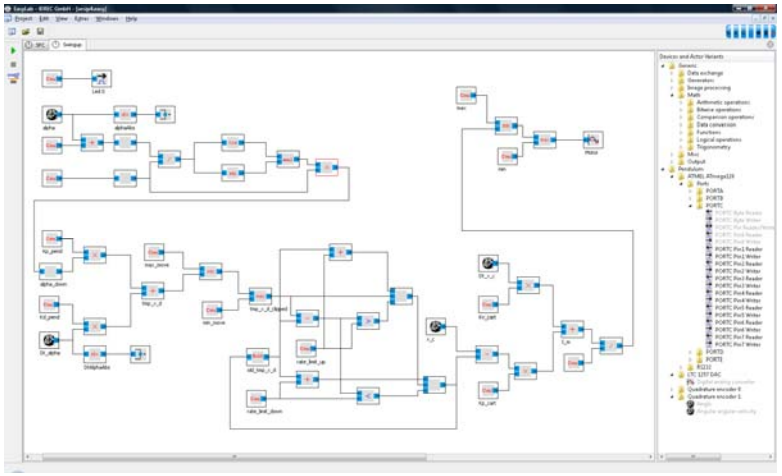
Synchroner Datenfluss

Werkzeug: EasyLab

EasyLab

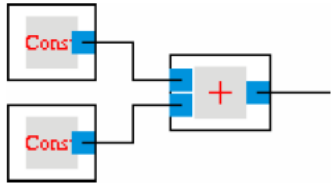
- Unterstützung verschiedener Sichten auf das System:
  - Anwendungslogik
    - Kontrollfluss und Datenfluss
  - Ausführungsumgebung
    - Hardware: Geräte kapseln den Zugriff auf Hardware-komponenten und bieten eine abstrakte Programmierschnittstelle
    - Die passende Werkzeugkette kann gewählt und angestoßen werden (Compilierung, Download)
- Inhalt der Vorlesung: Datenfluss





Synchroner Datenfluss

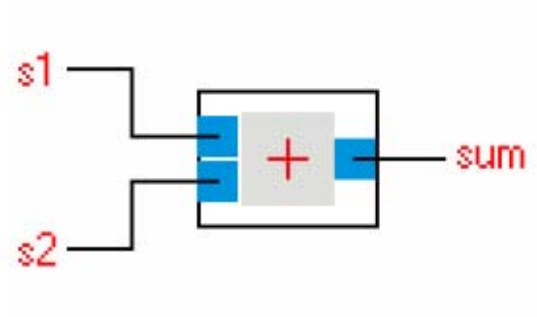
- Geeignet für datenzentrische Anwendungen
- Aktoren sind (z.T. typisierte) Funktionsblöcke,
  - die die einfließenden Daten verarbeiten/konvertieren und
  - das Ergebnis an Ausgängen zur Verfügung stellen

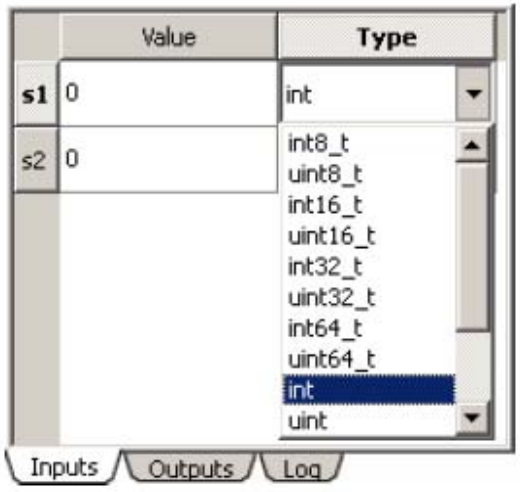


- Ausführungssemantik:
  - ein synchroner Datenfluss wird zyklisch gestartet, die Daten durchlaufen den kompletten Graphen
  - Parallelität ist direkt im Ausführungsmodell sichtbar
  - Zyklen sind erlaubt, allerdings müssen die Daten gepuffert werden

Überladen von Aktoren

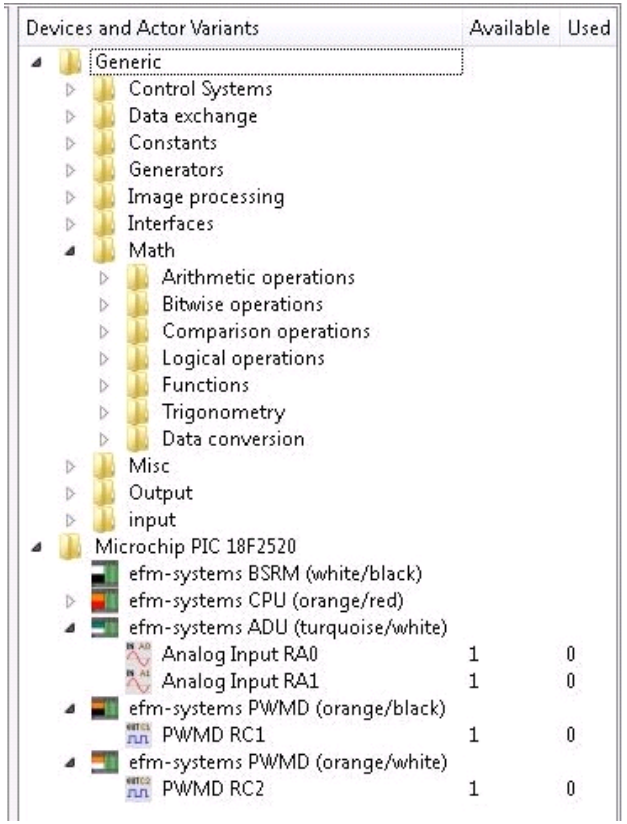
- Aktoren können typunabhängig angeboten werden
- Der Typ eines Aktors ist solange frei wählbar, bis sich durch Anschlussbelegung der Typ automatisch ergibt





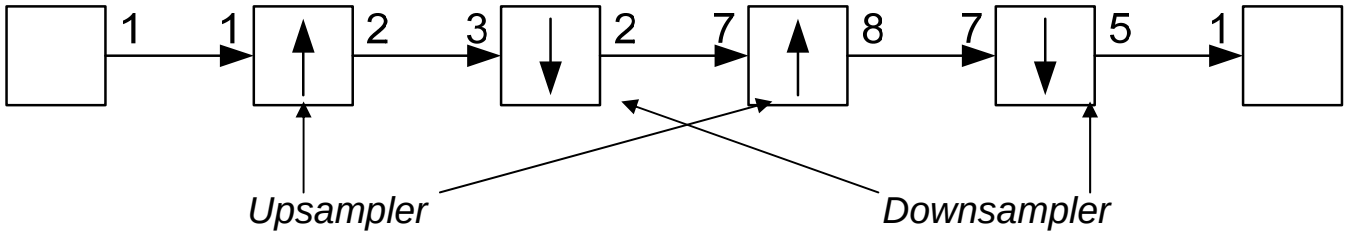
Geräte

- Neben Softwareaktoren sind in Echtzeitsystemen auch Aktoren zur Ansteuerung der Hardware interessant
- Lösung in EasyLab:
  - Geräte / Devices bieten Hardwareaktoren an
  - Hardwareaktoren bieten entweder Eingänge oder Ausgänge
  - Hardwareaktoren benötigen Ressourcen ⇒ die Verwendung der Aktoren ist in Ihrer Anzahl beschränkt



Multiratensysteme

- Häufig erfolgen Empfang und Verarbeitung von Daten in unterschiedlichen Raten:
  - Beispiel: Empfang über serielle Schnittstelle (bitweiser Empfang) und byteweise Verarbeitung
- Im synchronen Datenfluss können über Puffer Daten gesammelt und schließlich verarbeitet werden. Die Frequenzen der Aktoren werden bei den entsprechenden Eingängen angegeben:
  Beispiel: Konvertierung von 44,1 Hz (DAT) nach 48 Hz (CD)



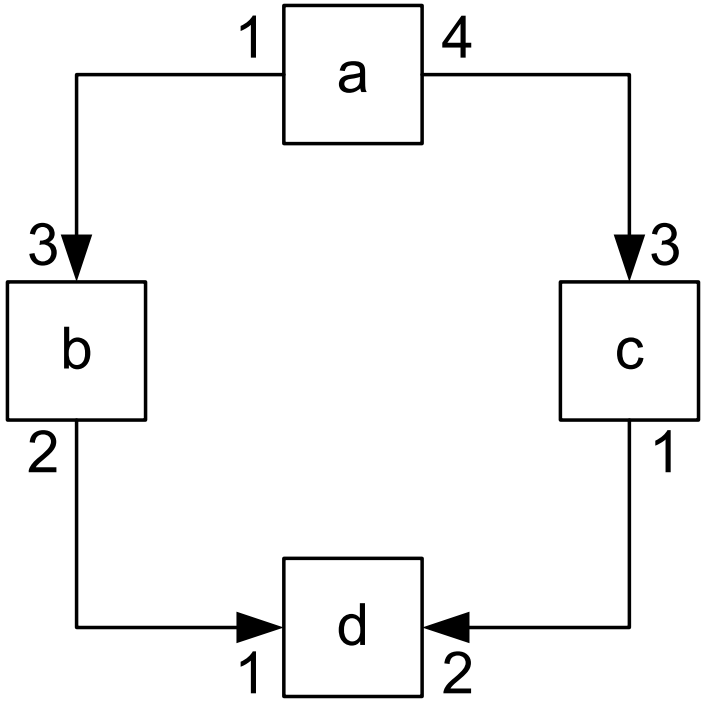
- Es kann notwendig sein, dass einige Felder bereits vorbelegt werden. Dies geschieht durch eine initiale Belegung der Kanten.

Ausführungssemantik von Multiratensystemen

- Ein Knoten kann erst schalten, wenn für alle eingehenden Kanten genügend Marken (Einzeldaten) vorliegen
- Vorteil von synchronen Datenfluss im Allgemeinen: der Ausführungsplan kann vorab berechnet werden
  - Ziel:
    - Sequenz von Aktorausführungen, die folgende Bedingungen erfüllt:
      - Jeder Aktor wird entsprechend seiner Rate ausgeführt
      - Die Aktoren sind an der Position im Ausführungsplan bereit für die Ausführung (Daten liegen an)
      - Kein Prozess darf zu oft laufen (Datenüberlauf)
  - Schritte:
    - Berechnung der relativen Ausführungsraten (Lösung von linearen Gleichungen)
    - Bestimmung eines periodischen Ausführungsplans durch Simulierung einer Runde (so dass die Belegung am Ende der Runde der Initialbelegung entspricht)
  - Typischerweise gibt es mehrere Ausführungspläne, durch entsprechende Wahl kann eine Reduktion der Codegröße, der Puffergröße erreicht werden

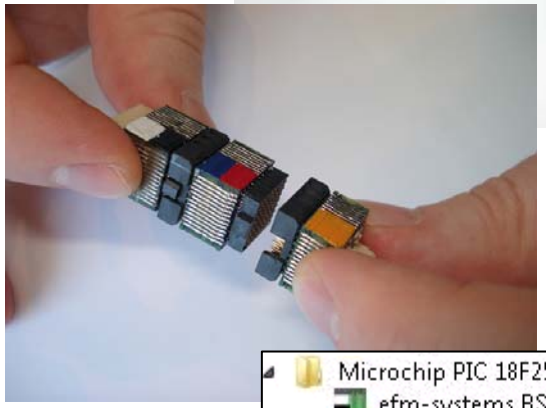
Berechnung eines Ausführungsplans

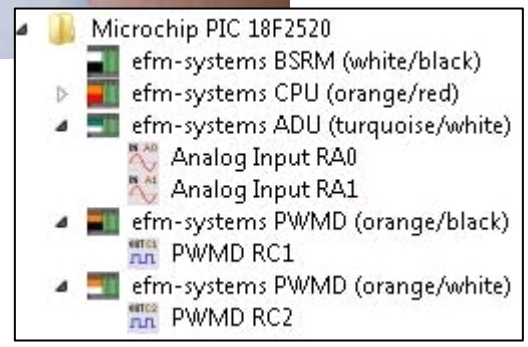
- Beispiel:
  - Raten der Aktoren:
    - a=3
    - b=1
    - c=4
    - d=2
  - Mögliche Ausführungspläne:
    - AAABCCCCDD
    - AAACCCCBDD
    - AAABCCDCCD
    - ...

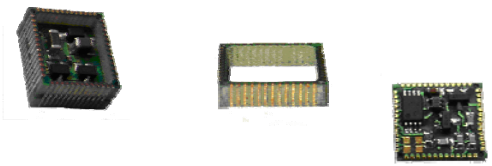


Anwendungsbeispiel

- Hardware
  - Zylinder: Kolbenposition, Einstellregler, 2 magnetische Ventile
  - Regelungsmodule: Match-X Bausteine:
    MCU Modul, ADC Modul,
    2 Treiberbausteine für induktive Lasten
- Ziel: Positionsregelung des Kolbens
- Implementierung:
  - Hardwaremodell basiert auf Bibliothek der Match-X Bausteine (Microchip PIC18F MCU)
  - Anwendungslogik
    - Einfaches Datenflussdiagramm
    - Integration der Hardwarelogik







Pneumatikzylinder