Embedded Systems / Eingebettete Systeme

BSc-Studiengang Informatik
Campus Minden

Matthias König

FH Bielefeld
University of
Applied Sciences

Beispiel einer bekannten Anwendung: Prozessorchipkarte

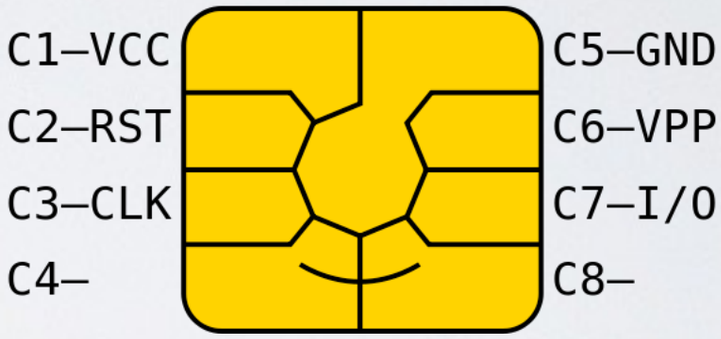
- Verwendung meistens zur Identifikation, Authentifizierung, Datenspeicherung

- Mikroprozessor, RAM, ROM (Betriebssystem), EEPROM, Input/Output

- Kommunikation über Pinout

- Standardisiert: ISO/IEC 7810

Pinout Chipkarte



[Quelle: http://commons.wikimedia.org/wiki/File:SmartCardPinout.svg]

WIEDERHOLUNG

Launchpad - Blink ARM-Assembler

120 Byte

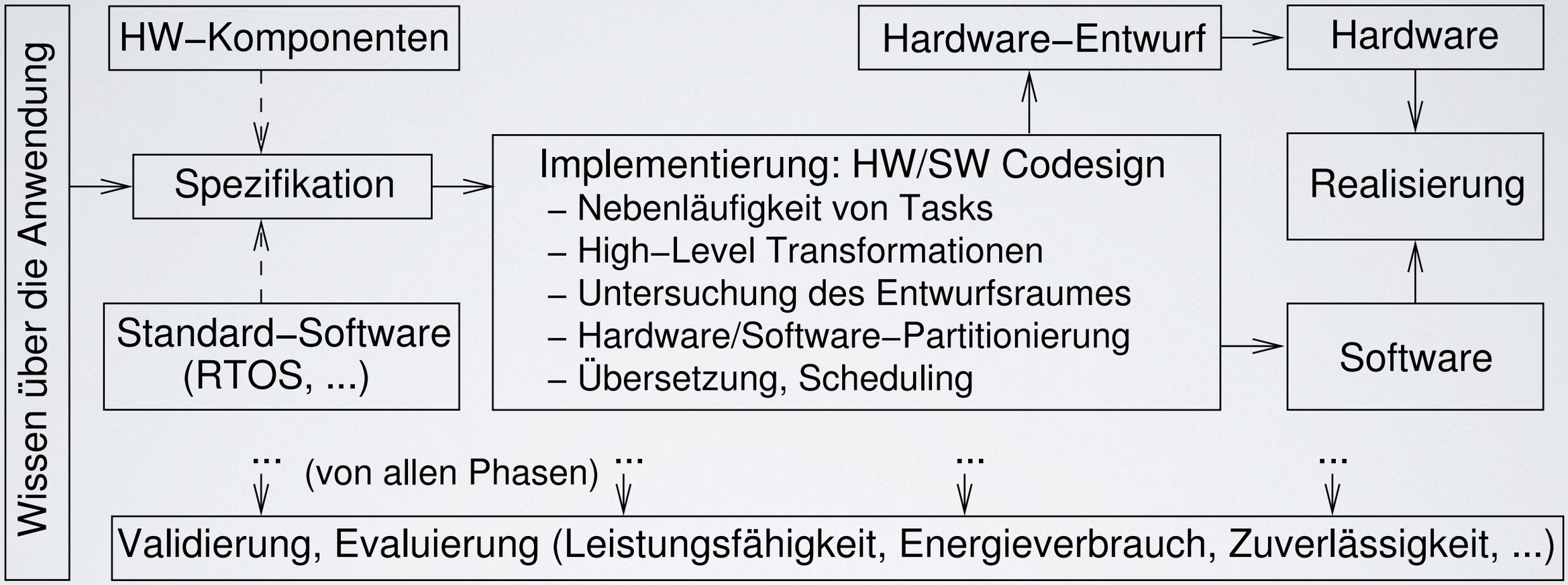
```
@ Blink the LED on the Launchpad ARM Cortex M4 board.
@ Directives
        .thumb
        .syntax unified
@ Equates
        .equ SYSCTL_RCGC2_R, 0x400fe108
        .equ GPIO_PORTF_DIR_R, 0x40025400
        .equ GPIO_PORTF_DEN_R, 0x4002551c
        .equ GPIO_PORTF_DATA_R, 0x400253fc
        .equ STACKINIT,   0x20005000
        .equ LEDDELAY,    800000
.section .text
        .org 0
@ Vectors
vectors:
        .word STACKINIT         @ stack pointer value
        .word _start + 1        @ reset vector
        .word _nmi_handler + 1  @
        .word _hard_fault  + 1  @
        .word _memory_fault + 1 @
        .word _bus_fault + 1    @
        .word _usage_fault + 1  @
_start:
        @ enable  gpio
        ldr r6, = SYSCTL_RCGC2_R
        mov r0, 0x20
        str r0, [r6]

        ldr r6, = GPIO_PORTF_DIR_R
        mov r0, 0x8
        str r0, [r6]
        ldr r6, = GPIO_PORTF_DEN_R
        str r0, [r6]

        ldr r6, = GPIO_PORTF_DATA_R    @ point to port F
loop:
        ldr r2, [r6]
        orr r2, #0x8
        str r2, [r6]            @  turn on LED
        ldr r1, = LEDDELAY
delay1:
        subs r1, 1
        bne delay1

        ldr r2, [r6]
        and r2, #0xfffffff7
        str r2, [r6]            @ turn off LED
        ldr r1, = LEDDELAY
delay2:
        subs r1, 1
        bne delay2

        b loop                  @ continue forever

_dummy:                         @ just hang in a loop
_nmi_handler:
_hard_fault:
_memory_fault:
_bus_fault:
_usage_fault:
        add r0, 1
        add r1, 1
        b _dummy
```

Launchpad - startup_gcc.c in C

```c
#include <stdint.h>
#include "inc/hw_nvic.h"
#include "inc/hw_types.h"

void ResetISR(void);
static void NmiSR(void);
static void FaultISR(void);
static void IntDefaultHandler(void);
extern int main(void);

static uint32_t pui32Stack[64];

// The vector table.  Note that the proper constructs must be placed
on this to
__attribute__ ((section(".isr_vector")))
void (* const g_pfnVectors[])(void) = {
    (void (*)(void))((uint32_t)pui32Stack + sizeof(pui32Stack)),
                                            // The initial stack
pointer
    ResetISR,                               // The reset handler
    NmiSR,                                  // The NMI handler
    FaultISR,                               // The hard fault handler
    IntDefaultHandler,                      // The MPU fault handler
    ...
    IntDefaultHandler                       // PWM 1 Fault
};

// for linker
extern uint32_t _etext;
extern uint32_t _data;
extern uint32_t _edata;
extern uint32_t _bss;
extern uint32_t _ebss;
```

```c
void ResetISR(void) {
    uint32_t *pui32Src, *pui32Dest;

    pui32Src = &_etext;
    for(pui32Dest = &_data; pui32Dest < &_edata; )    {
        *pui32Dest++ = *pui32Src++;
    }

    __asm("    ldr     r0, =_bss\n"
          "    ldr     r1, =_ebss\n"
          "    mov     r2, #0\n"
          "    .thumb_func\n"
          "zero_loop:\n"
          "        cmp     r0, r1\n"
          "        it      lt\n"
          "        strlt   r2, [r0], #4\n"
          "        blt     zero_loop");

    HWREG(NVIC_CPAC) = ((HWREG(NVIC_CPAC) &
                         ~(NVIC_CPAC_CP10_M | NVIC_CPAC_CP11_M)) |
                        NVIC_CPAC_CP10_FULL | NVIC_CPAC_CP11_FULL);
    main();
}

static void NmiSR(void) {
    while(1) {}
}

static void FaultISR(void) {
    while(1){}
}

static void IntDefaultHandler(void) {
    while(1){}
}
```

Hardware/Software Codedesign



Entwurf Eingebetteter Systeme (nach Marwedel)

[Quelle: Marwedel, Eingebettete Systeme]

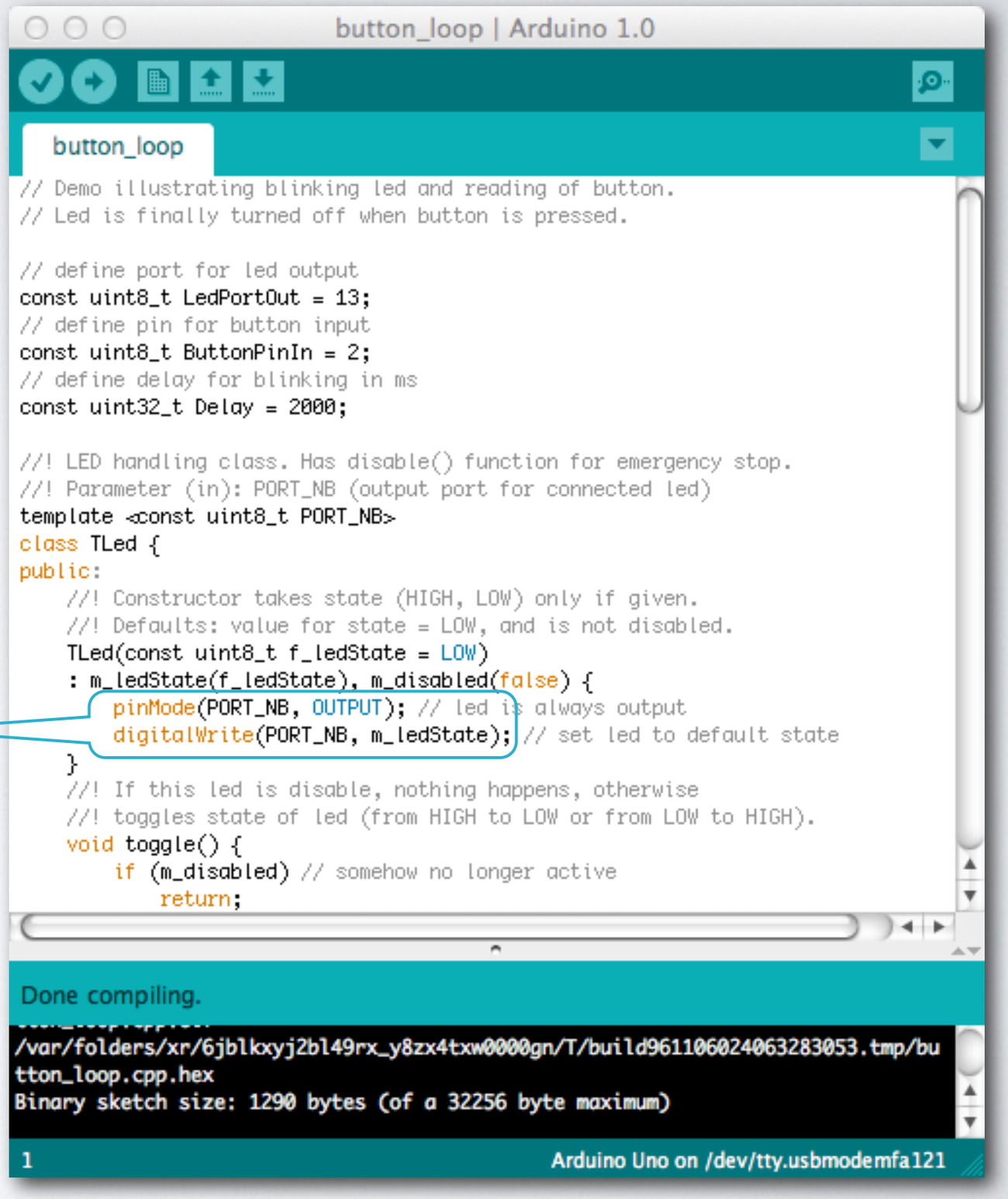
Entwicklung Eingebetteter Systeme

Entwicklungswerkzeuge

Beispiel Arduino/Energia:

Entwicklungsumgebung
und Bibliotheken basierend
auf C++

Bibliotheksfunktionen



Softwareentwicklungsprozess

beinhaltet generell:

- Anforderungen / Requirements
- Entwurf / Design
- Implementierung / Construction
- Test / Testing
- Wartung / Maintenance

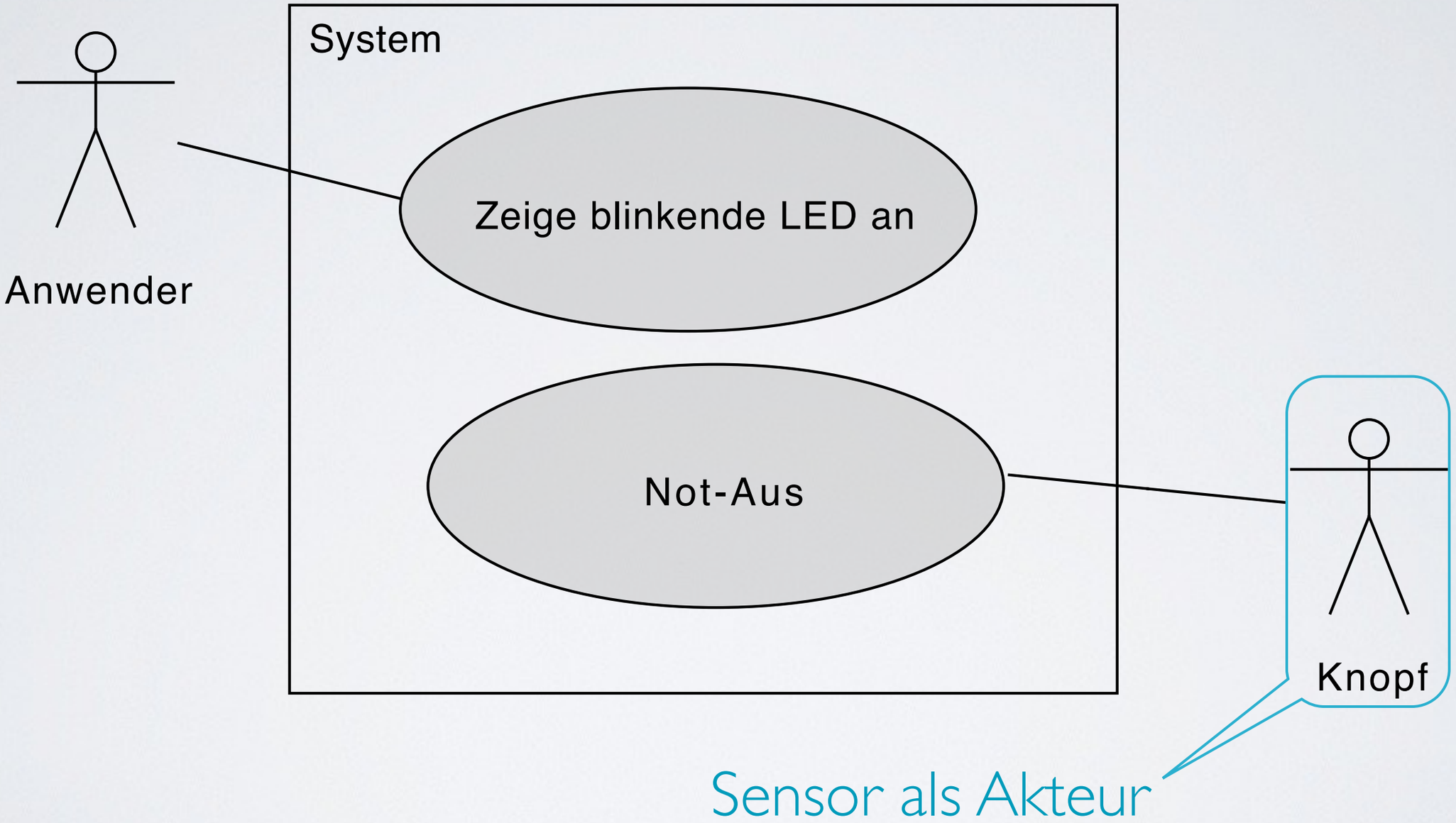
Anforderungen / Requirements

- Feststellung von Anforderungen (Requirements engineering) entsprechend Vorgehen des Software Engineering (vgl. entsprechende Veranstaltung), beispielsweise mit

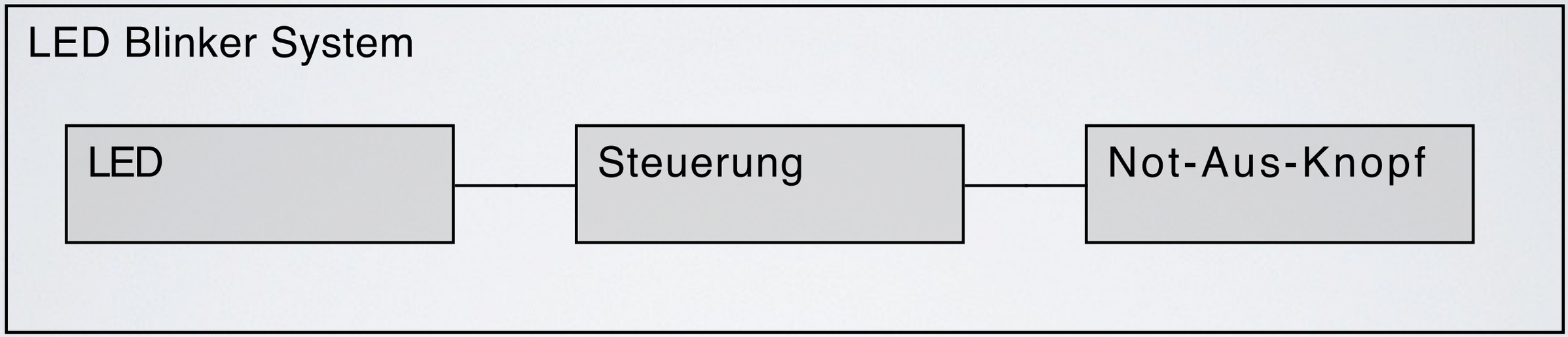
  - Anwendungsfällen

  - Kompositionsstrukturdiagramm

Beispiel: Anwendungsfall



Beispiel: Kompositionsstrukturdiagramm



Entwurf: Spezifikation & Modellierung

- Spezifikation / Modellierung für systematisches Vorgehen bei nachfolgenden Schritten bezüglich
  - Prüfung auf Widerspruchsfreiheit, Vollständigkeit
  - Herleitung der Implementierung
- Maschinenlesbarkeit vorteilhaft

[Quelle: Marwedel, Eingebettete Systeme]

Anforderungen an Spezifikationssprachen

- Abbildung von Hierarchie (Abstraktion zum Herunterbrechen von Komplexität, besser lesbar für Menschen)

  - Verhaltenshierarchie,
    z.B. hierarchische Zustände, Funktionen

  - Strukturelle Hierarchie,
    Zusammensetzung aus physikalische Komponenten

[Quelle: Marwedel, Eingebettete Systeme]

Anforderungen an Spezifikationssprachen

- Unterstützung von reaktiven Systemen hinsichtlich:
  - Zuständen
  - Ereignissen (externe/interne)
  - Ausnahmen

[Quelle: Marwedel, Eingebettete Systeme]

Anforderungen an Spezifikationssprachen

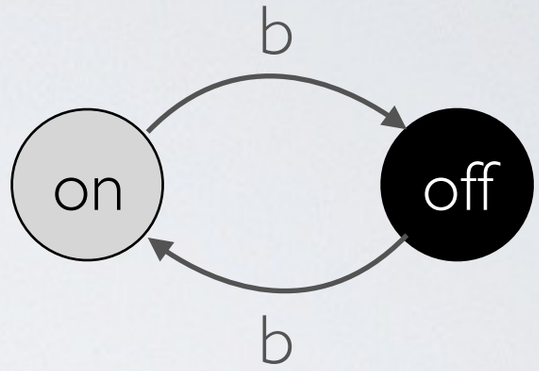
- Darstellung von Zeitverhalten
- Nebenläufigkeit (auch bei verteilten Systemen)
- Synchronization und Kommunikation
- Plattformunabhängig
- Ausführbar, prüfbar
- Unterstützung bei Entwurf komplexer Systeme

[Quelle: Marwedel, Eingebettete Systeme]

Spezifikationssprachen

- stellen aufgrund der vielen Anforderungen immer einen **Kompromiss** dar,
- unterscheiden sich bei Modellen für
  - Ablauf von Berechnungen (sequentiell, Zustand, Zeitstempel)
  - Kommunikation (shared memory, message passing)

Endliche Automaten / Finite State Machines



Systembeschreibung durch

- Zustände / State
- Ereignisse / Event
- Übergänge / Transition
  - wenn Ereignis b im Zustand on ⟶ gehe in Zustand off

Statecharts

[Harel, 1987]: "*Statecharts constitute a visual formalism for describing states and transitions in a modular fashion, enabling clustering, orthogonality (i.e. concurrency) and refinement...*"

Citizen quartz multi-alarm

main

displays

regular

update

mon

hr

sec

date

10 min

1 min

day

year

mode

wait

2 sec in wait

time

date

d(b up)

2 min in date

2 min in update

beep-test

00

10

01

beep

stopwatch

zero

d (in off)

display

reg.

(in on)

lap

run

on

off

out

chime

alarm 2

update 2

alarm 1

update 1

hr

min

2 min in out (no changes)

T hits T1 (P1)

any button pressed

30 sec in alarms-beep

T hits T2 (P2)

T hits T1 (P)

alarms-beep

alarm 1 beeps

alarm 2 beeps

both beep

alarm 1-status

disabled

enabled

(in alarm 1 .off)

(in alarm 1 .on)

alarm 2-status

(in alarm 2 .off)

(in alarm 2 .on)

chime-status

(in chime.on)

(in chime.off)

quiet

T is whole hour

2 sec in beep

light

power

OK

battery weakens

blink

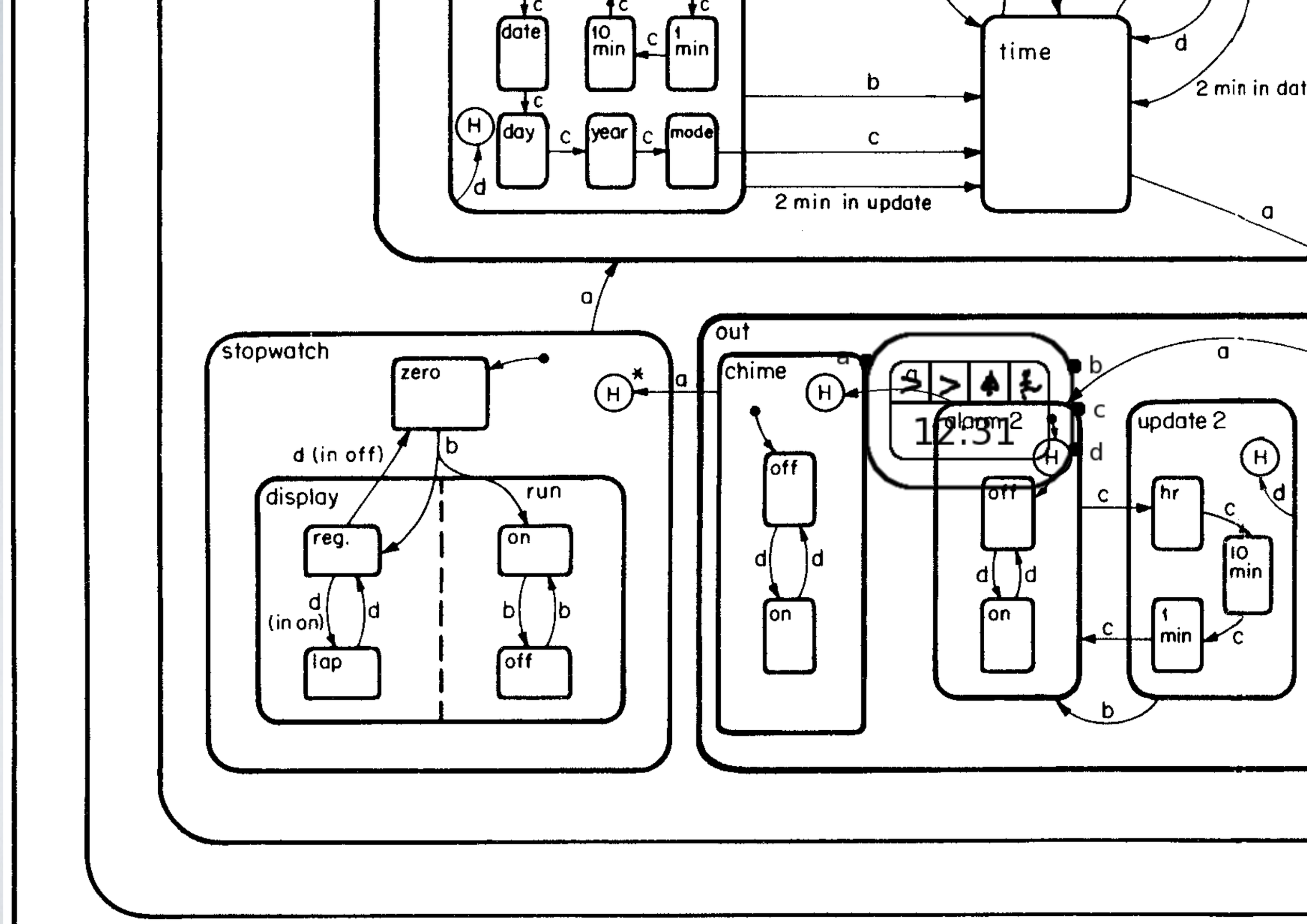
battery inserted

battery removed /clh(main*)

dead

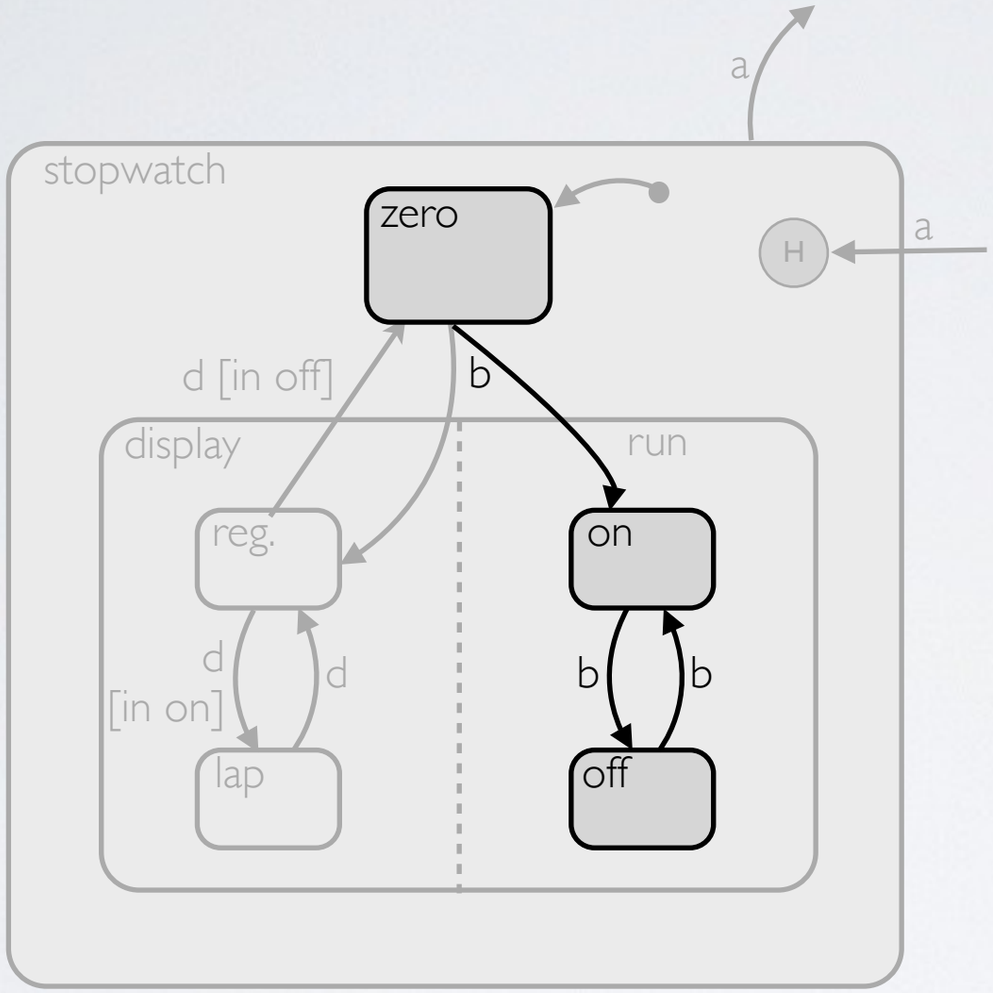
weak battery dies /clh(main*)

[Quelle:Harel, 1987]

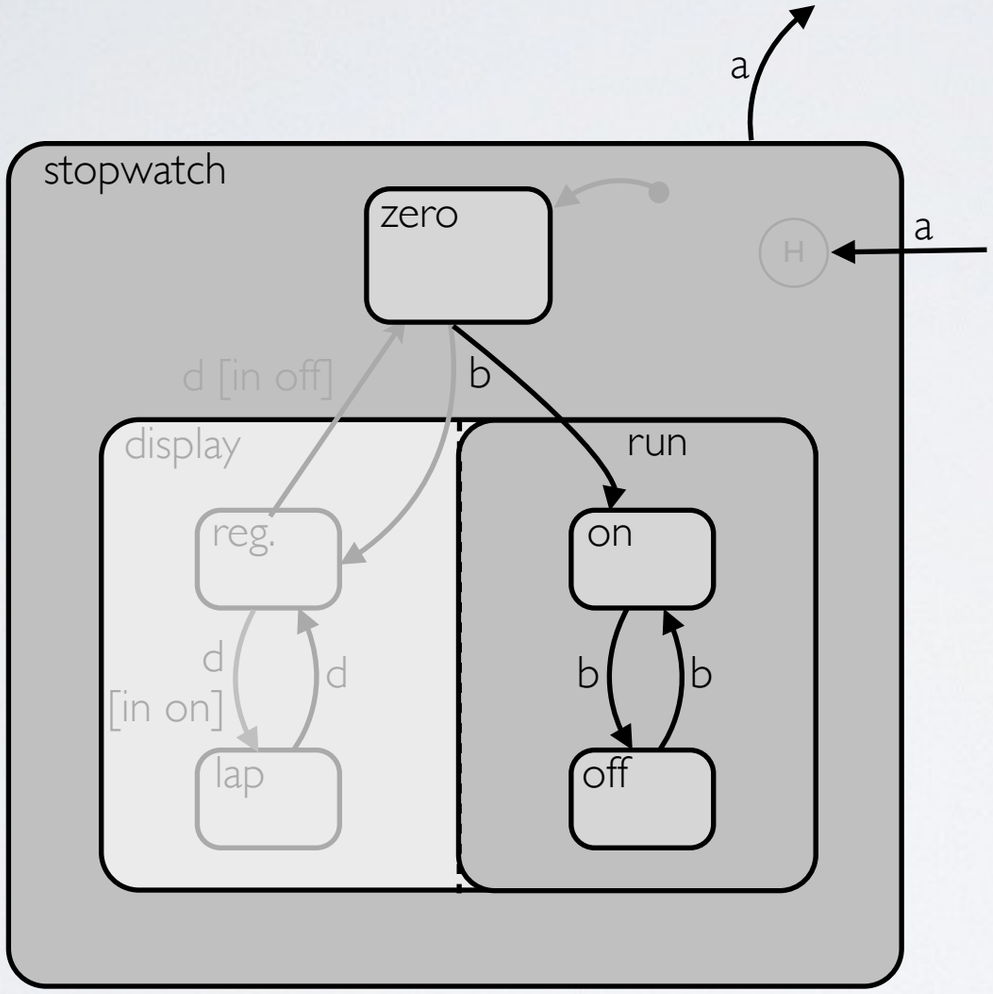


[Quelle:Harel, 1987]

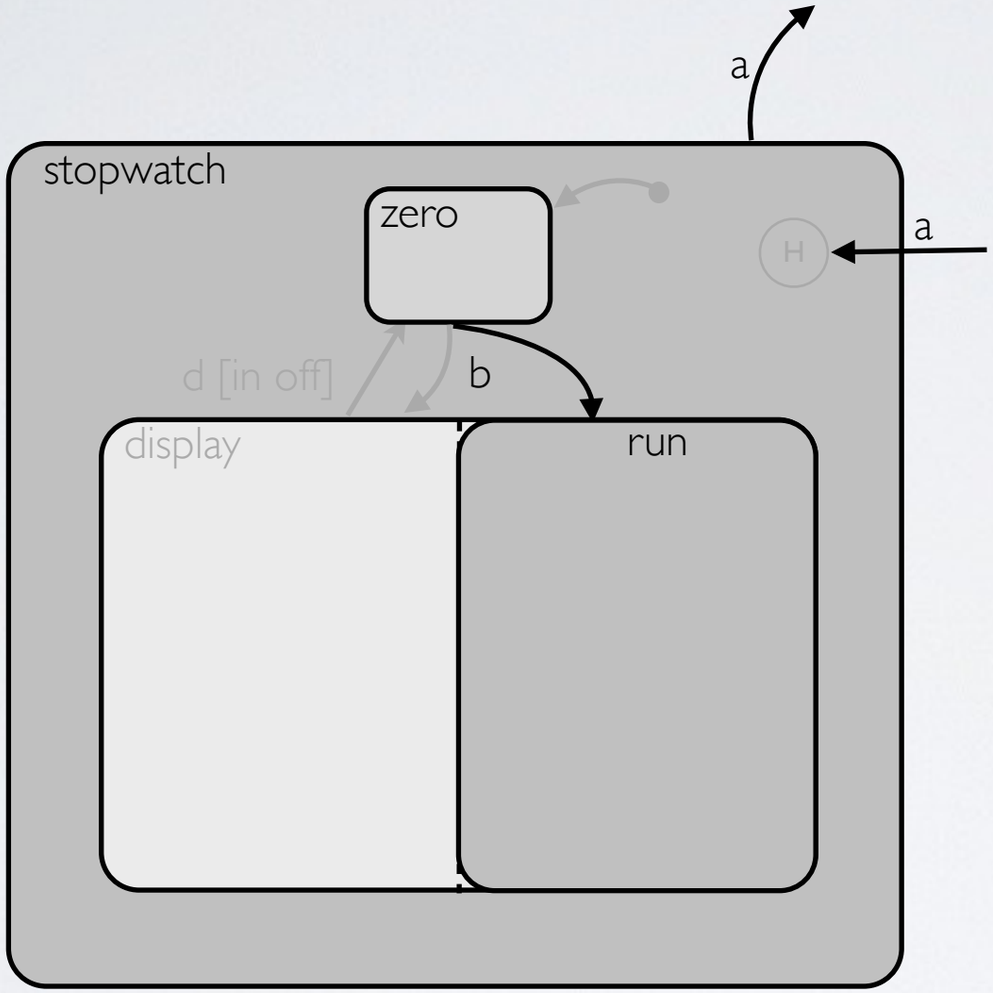
Endliche Automaten / Finite State Machines



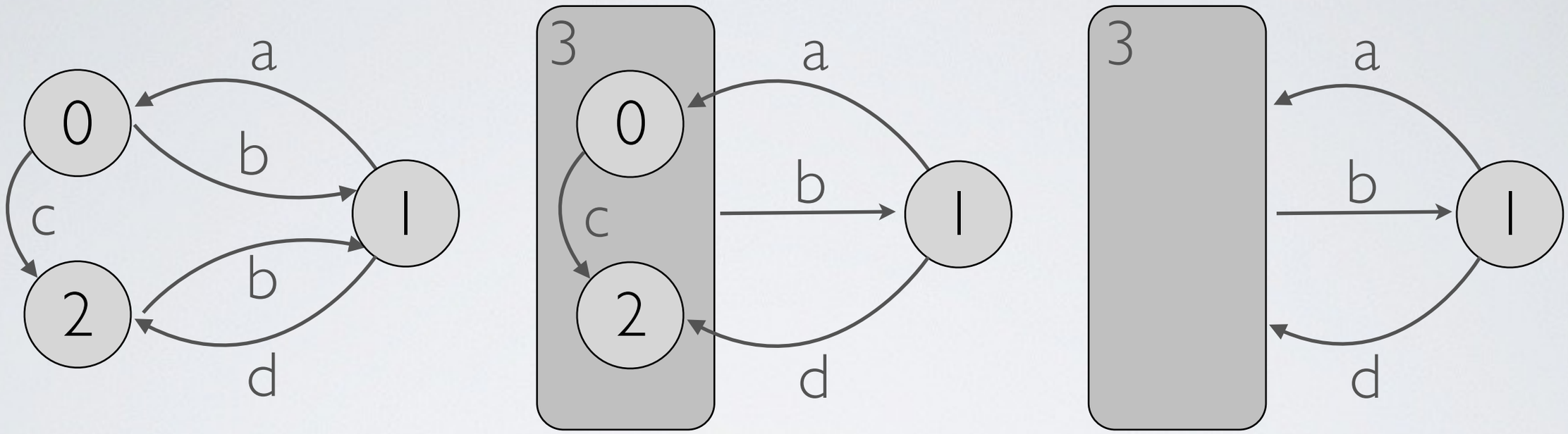
Statecharts: Hierarchie



Statecharts: Modularisierung

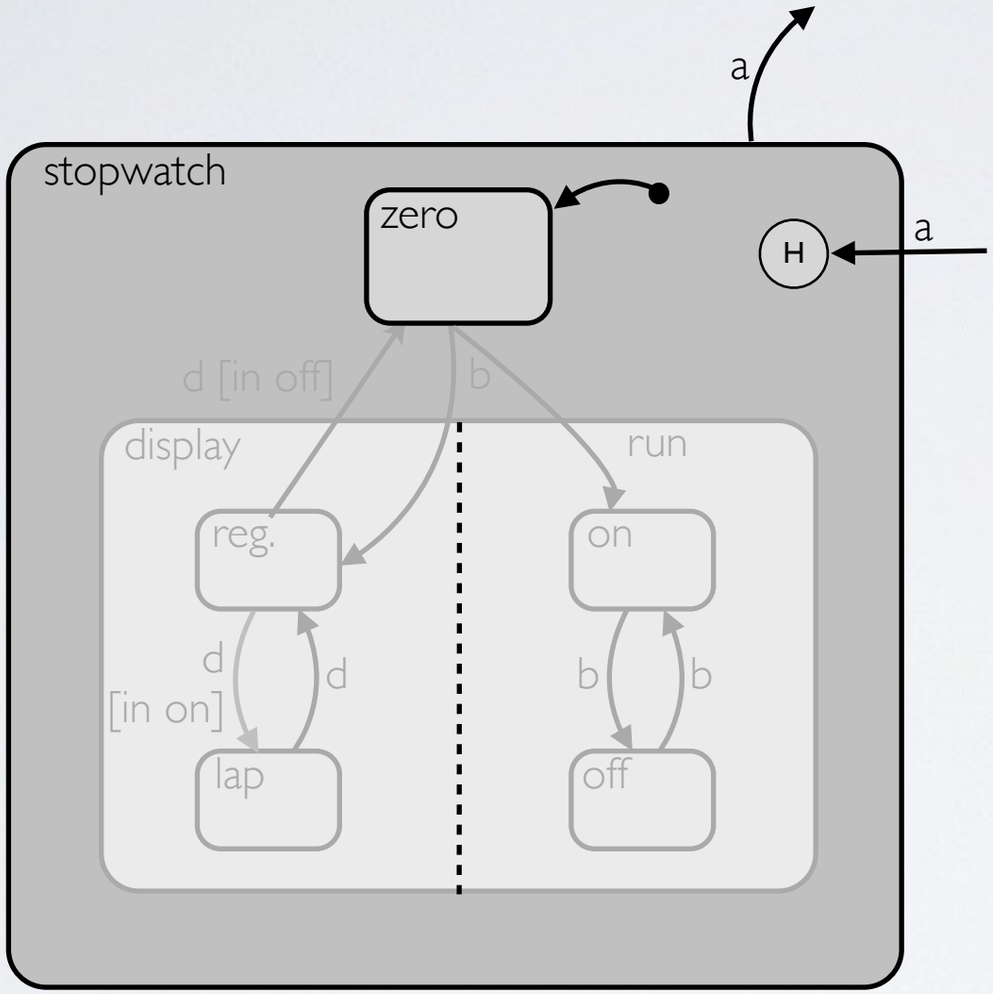


Statecharts: Hierarchie / Modularisierung

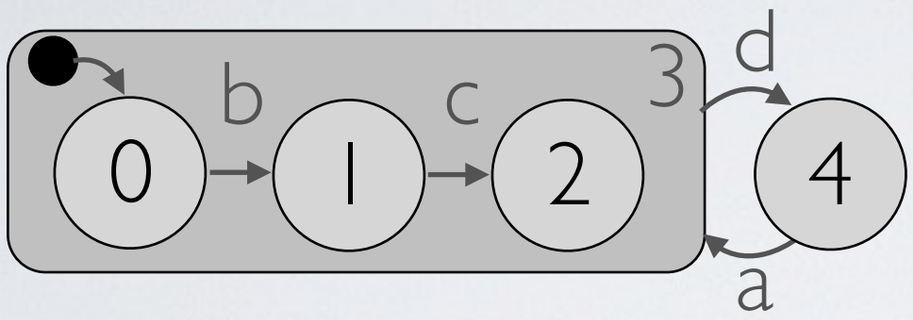


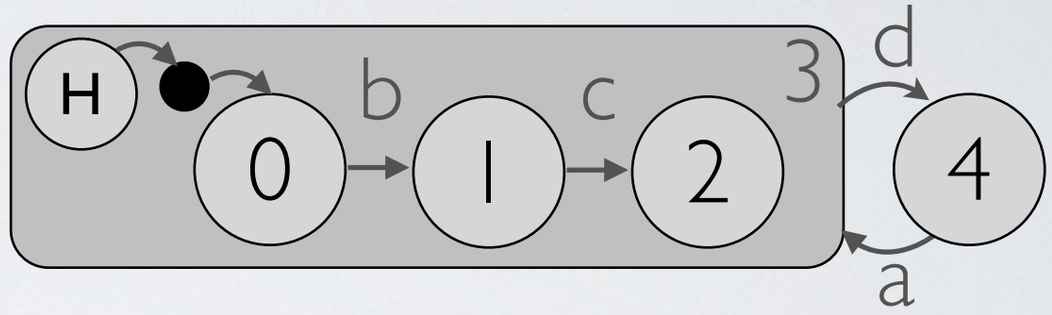
- Superzustände umfassen Unterzustände.
- Ein Oder-Superzustand hat nur einen aktiven Unterzustand.

Statecharts: Defaults und History



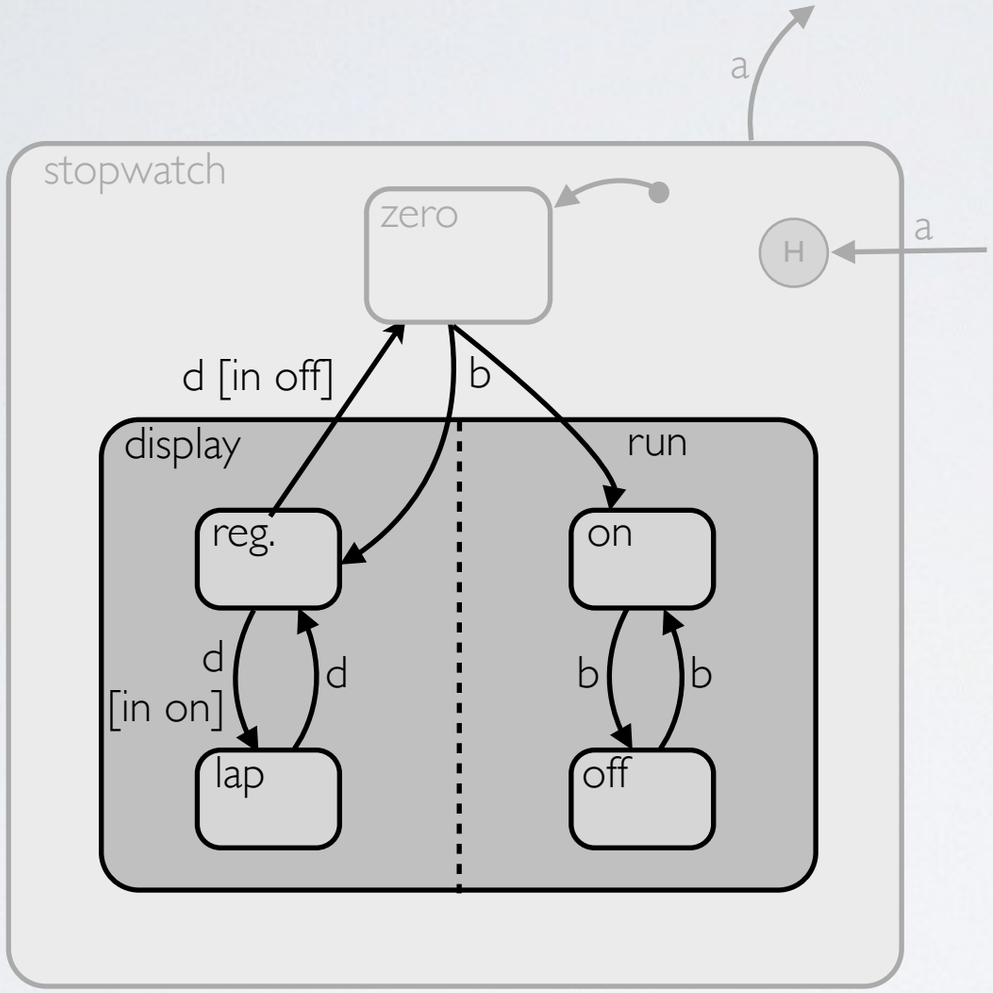
Statecharts: Defaults und History



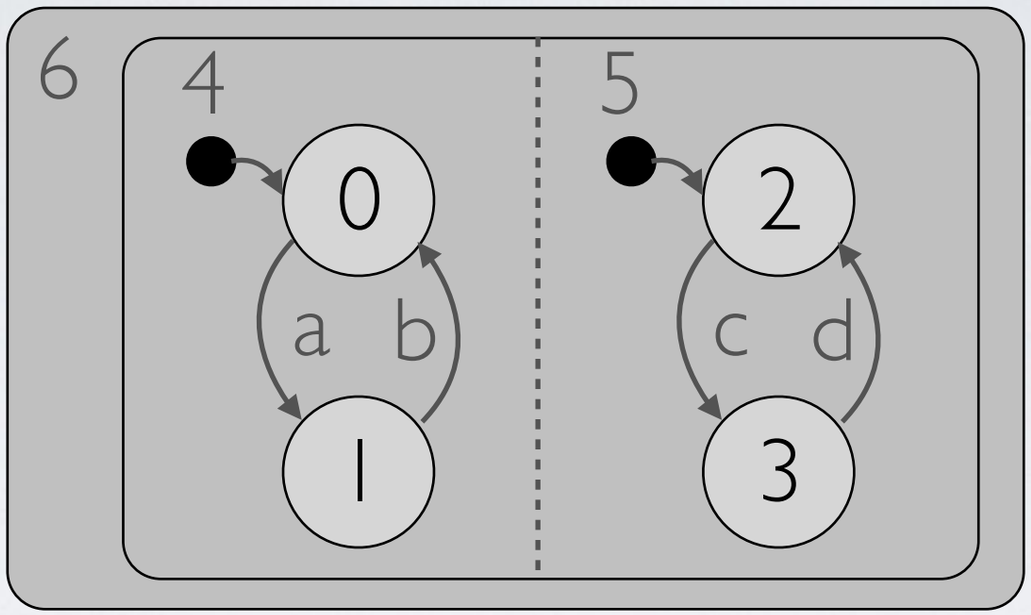


- Default-Zustand ● wird aktiv, wenn Superzustand aktiv wird.
- History-Merkmal (H) ermöglicht Rückkehr zum letzten aktiven Unterzustand, wenn der Superzustand aktiv wird.

Statecharts: Nebenläufigkeit & Kanten



Statecharts: Nebenläufigkeit



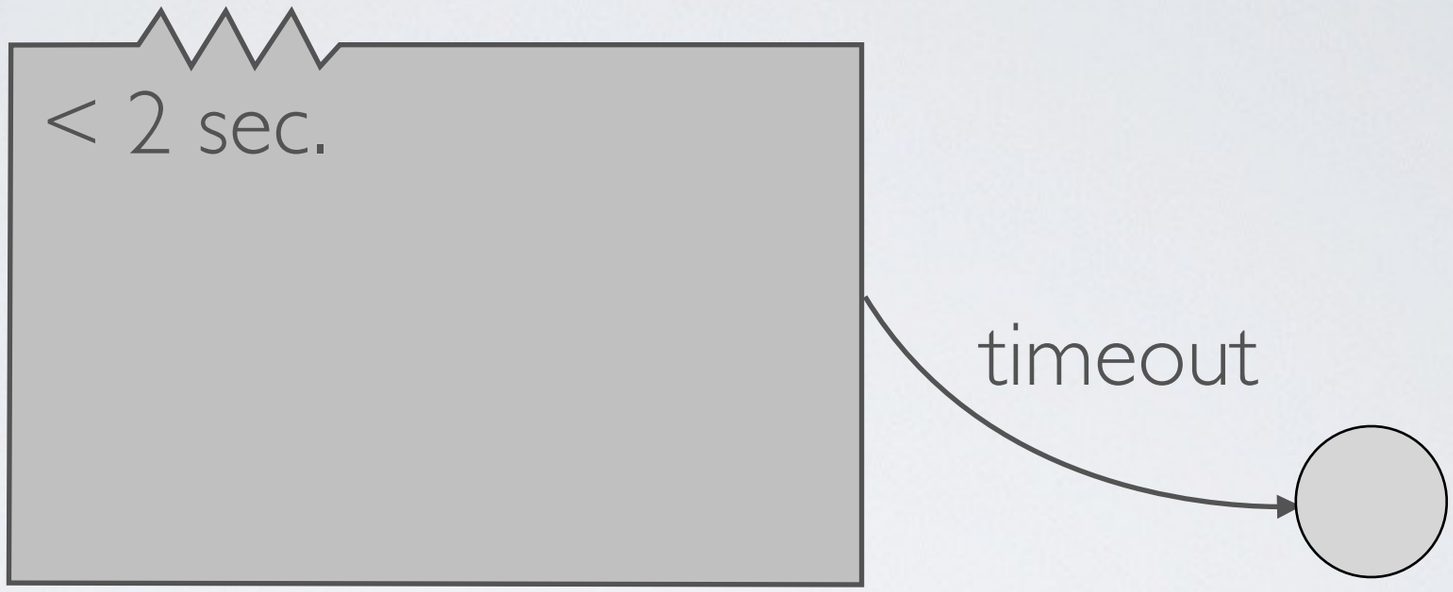
- Ein Und-Superzustand beinhaltet zwei oder mehr unabhängige Komponenten mit aktiven Unterzuständen.

- Und-Superzustände werden mit gestrichelter Line getrennt.

Statecharts: Kantenbeschriftungen

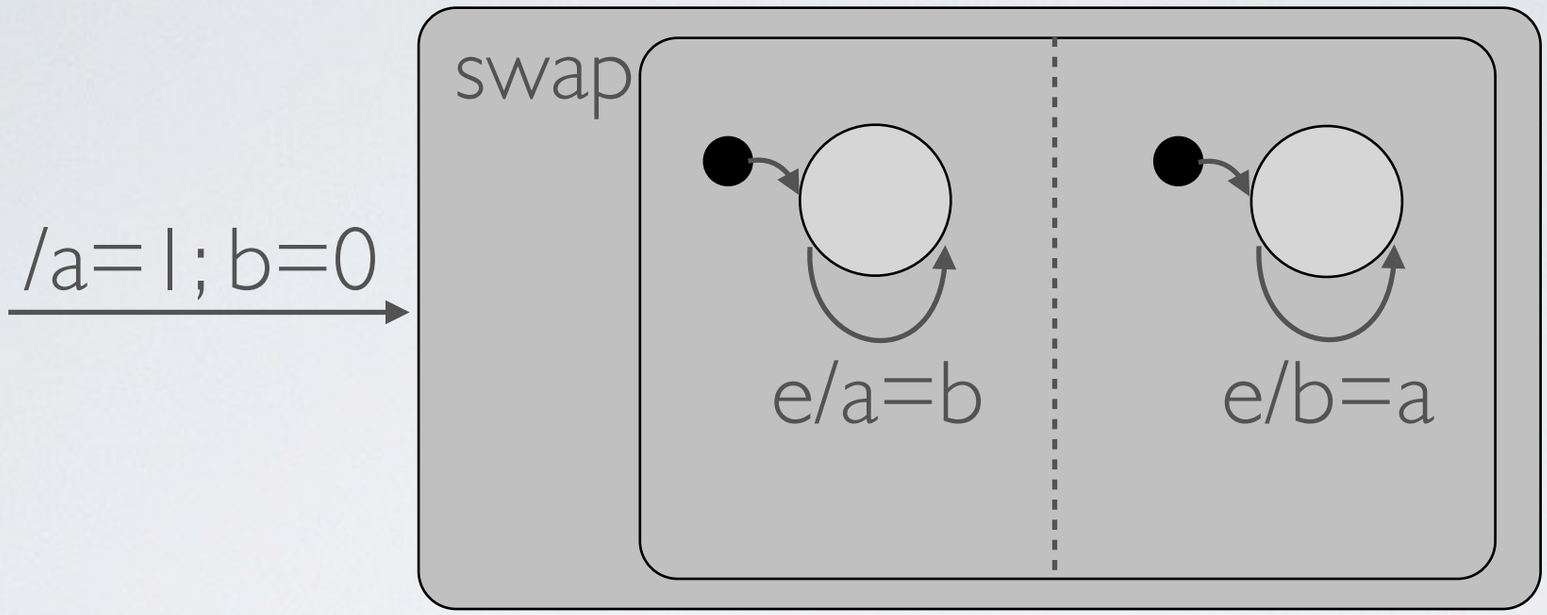
- Ereignisse aktiven/deaktivieren Zustände.
- Bedingungen werden geprüft und ausgewertet.
- Aktionen weisen Variablen zu oder generieren Ereignisse.

Statecharts: Timer



- Wenn kein Ereignis eintritt, geht der Zustand nach Ablauf einer vorgegebenen Zeit in spezifizierten Zustand über.

Statecharts: StateMate Semantics

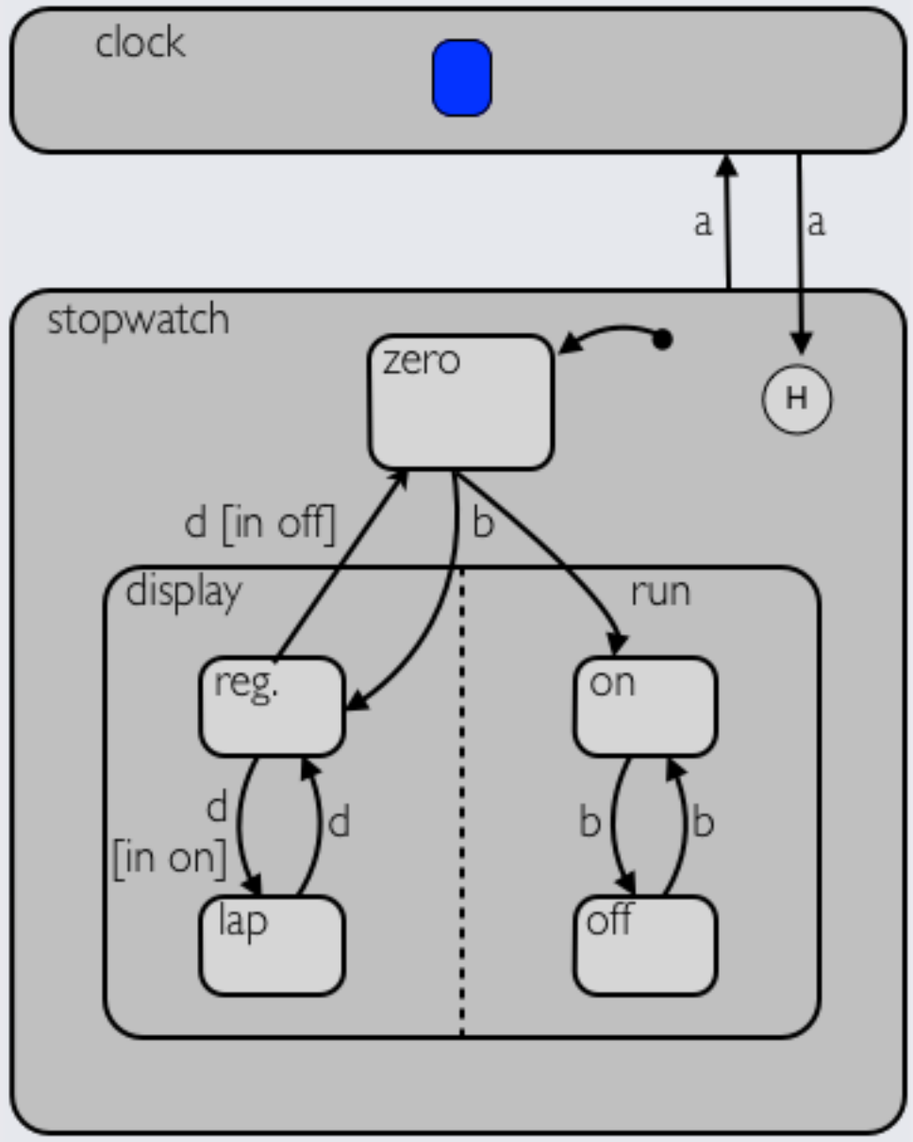


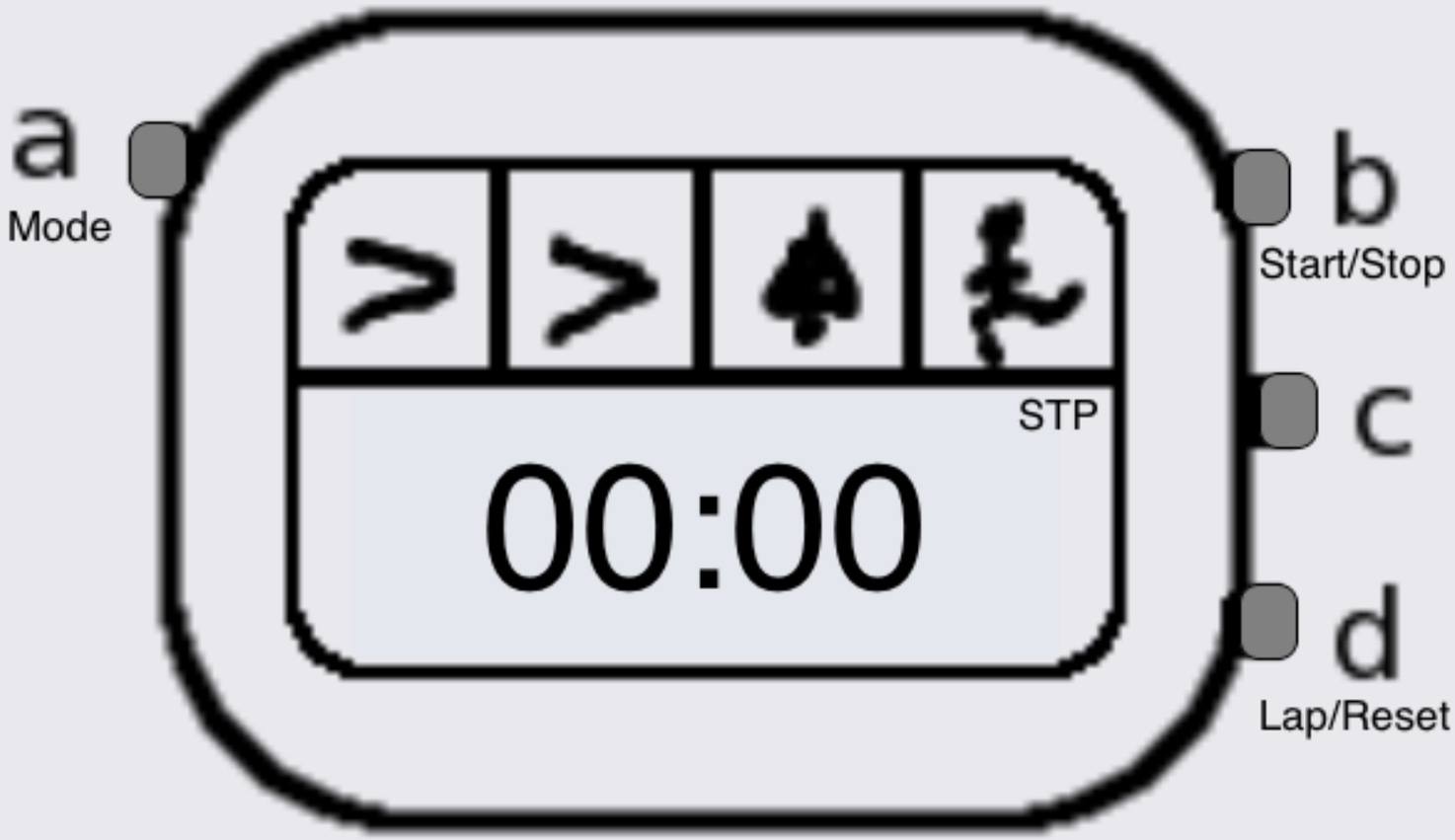
1. Auswertung von Ereignisse und Bedingungen.
2. Berechnung von Zuweisungen (temporär Speicherung).
3. Übergang in neue Zustände und Zuweisung zu Variablen.

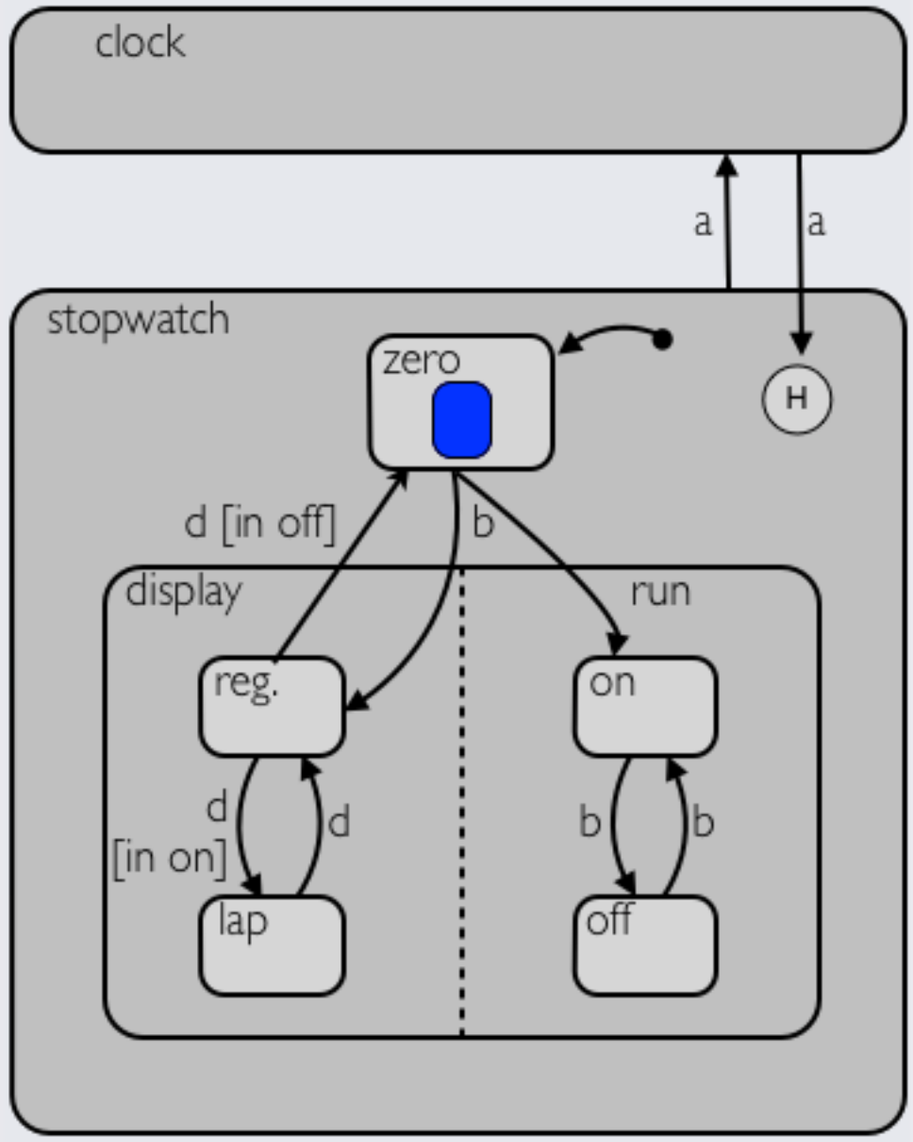
→ Erreicht Broadcast-Mechanismus.

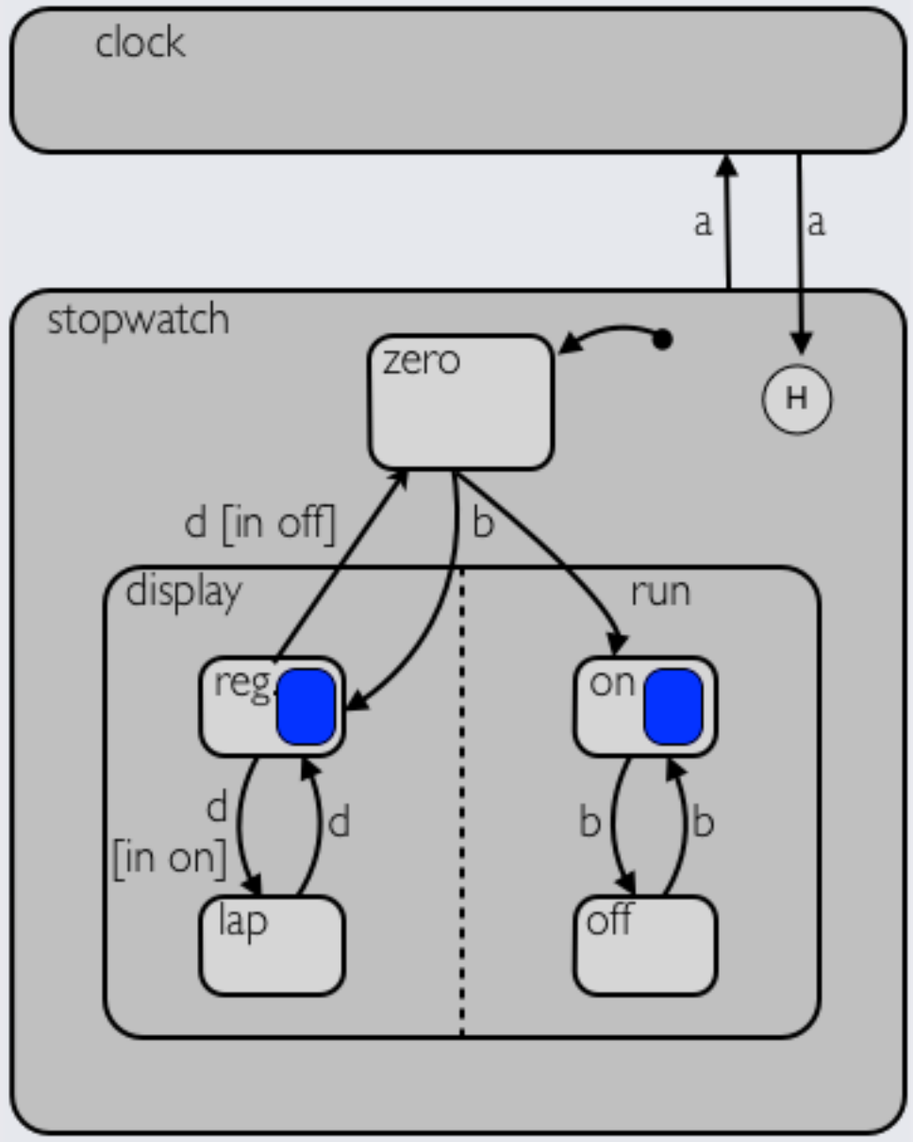
Demo

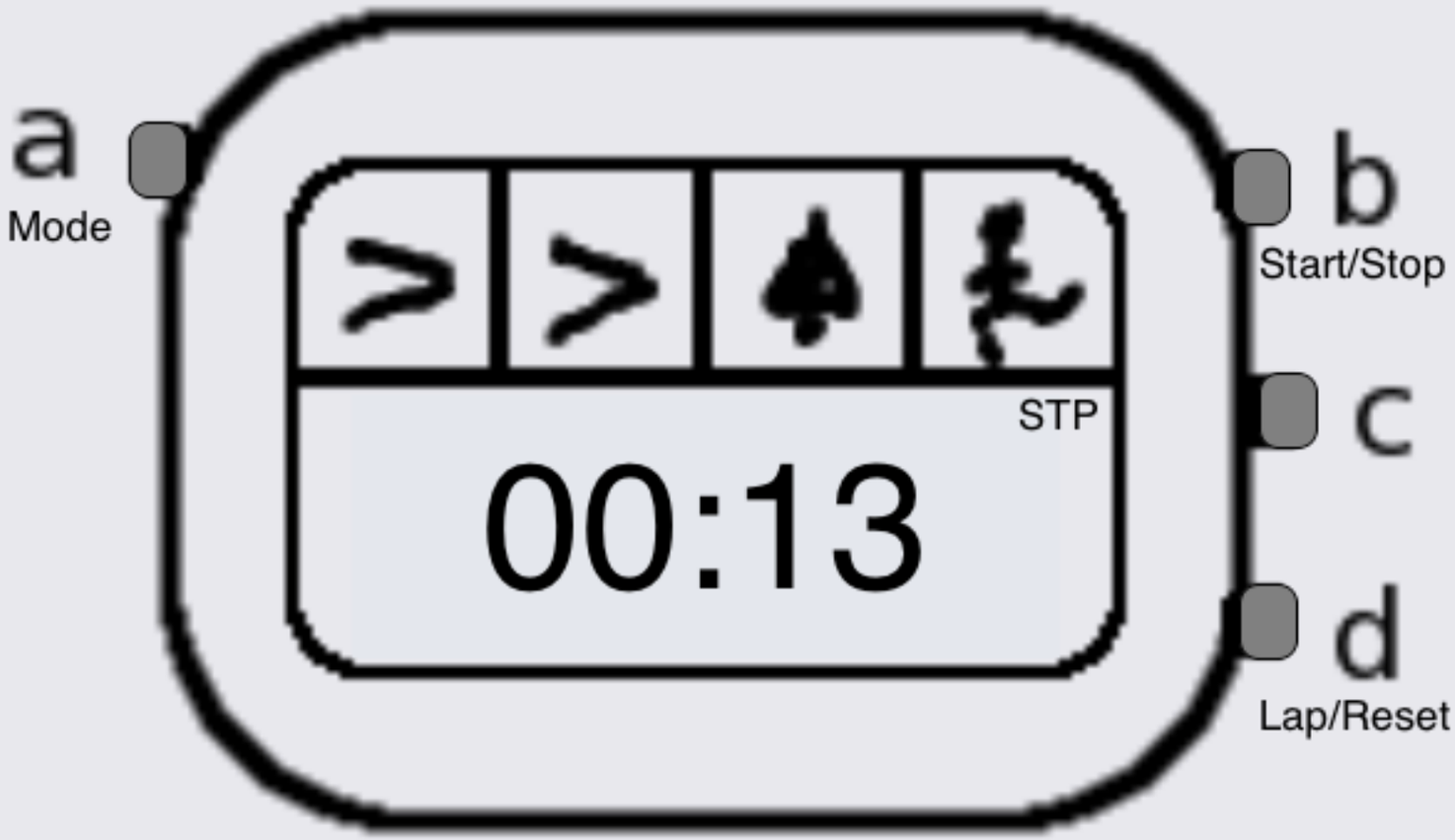

a
Mode
b
Start/Stop
c
d
Lap/Reset
11:17:20

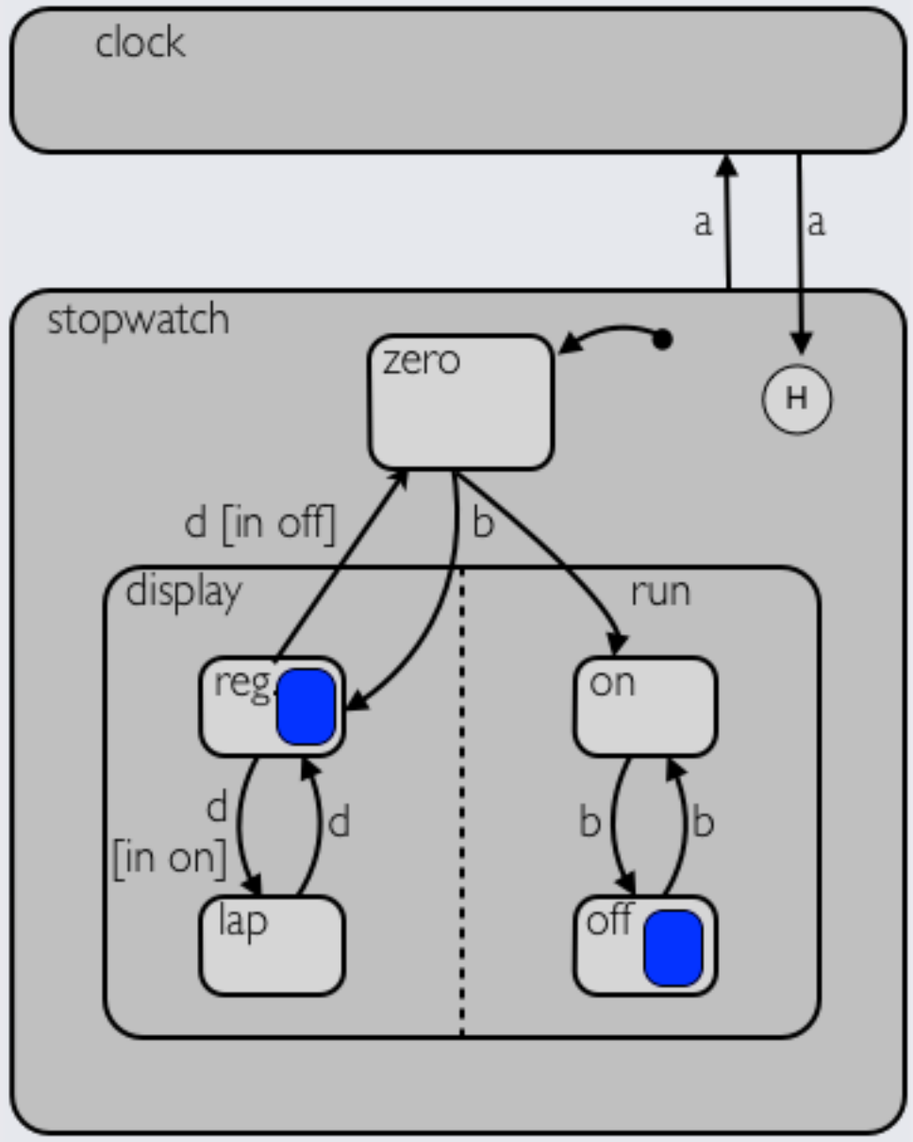

clock
a
a
stopwatch
zero
H
d [in off]
b
display
run
reg.
on
d
[in on]
d
b
b
lap
off

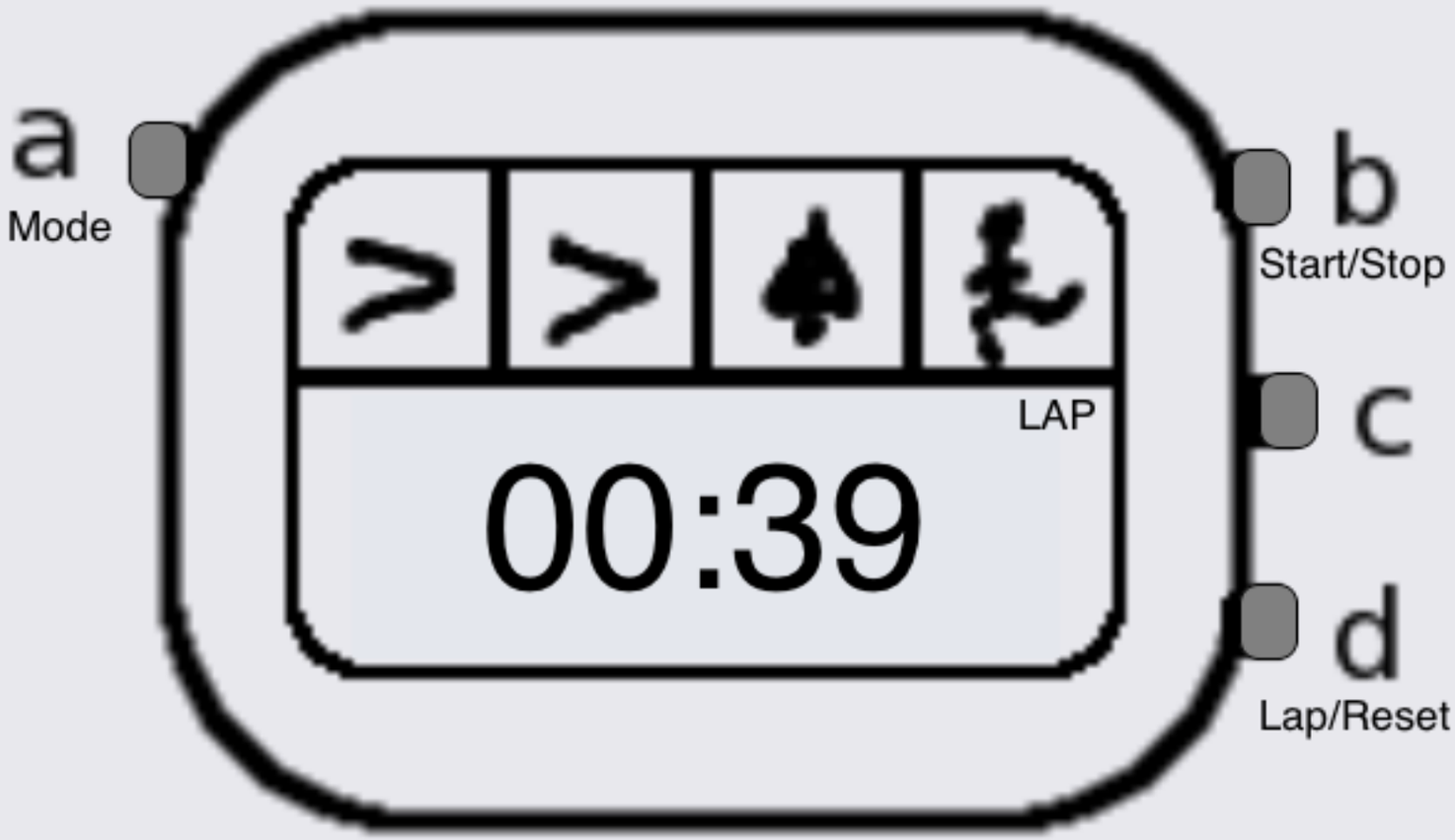


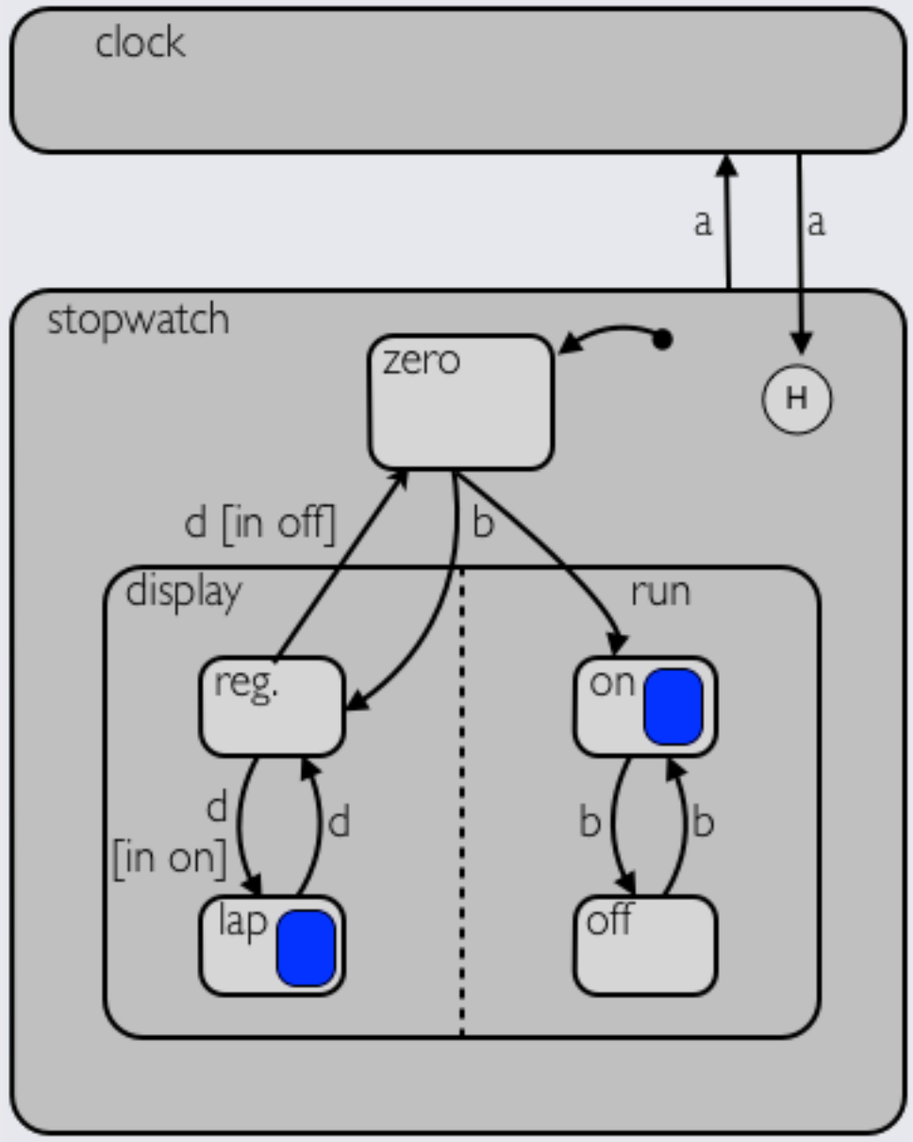


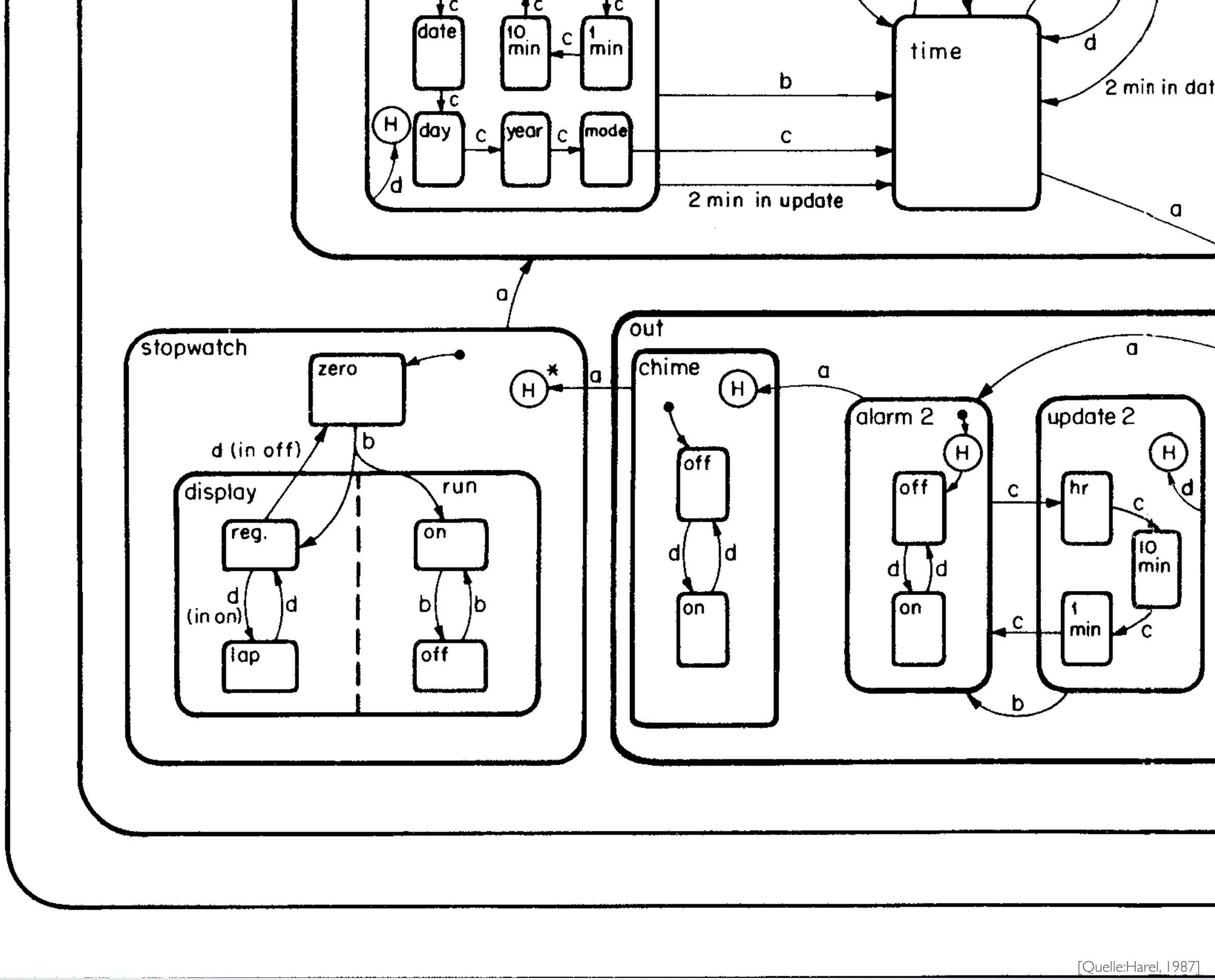












[Quelle:Harel, 1987]

main

displays

regular

update

wait

2 sec
in wait

$\hat{c}$

c

d(b up)

date

time

d

2 min in date

b

c

2 min in update

beep-test

2 min in
(no cha

a

a

stopwatch

a

out

chime

a

a

alarm 2

c

c

b

alarm 1

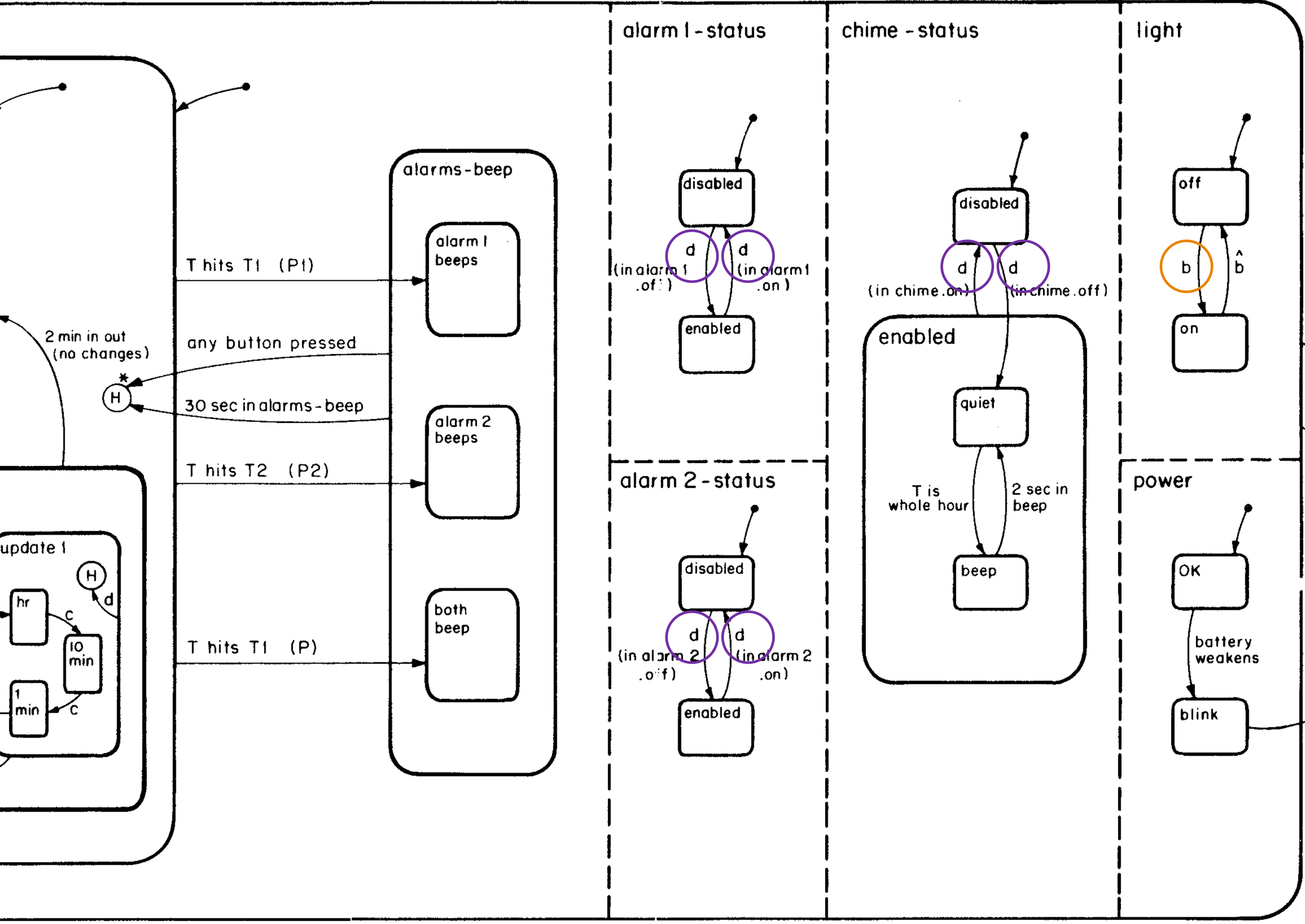
c

c

b

update 1

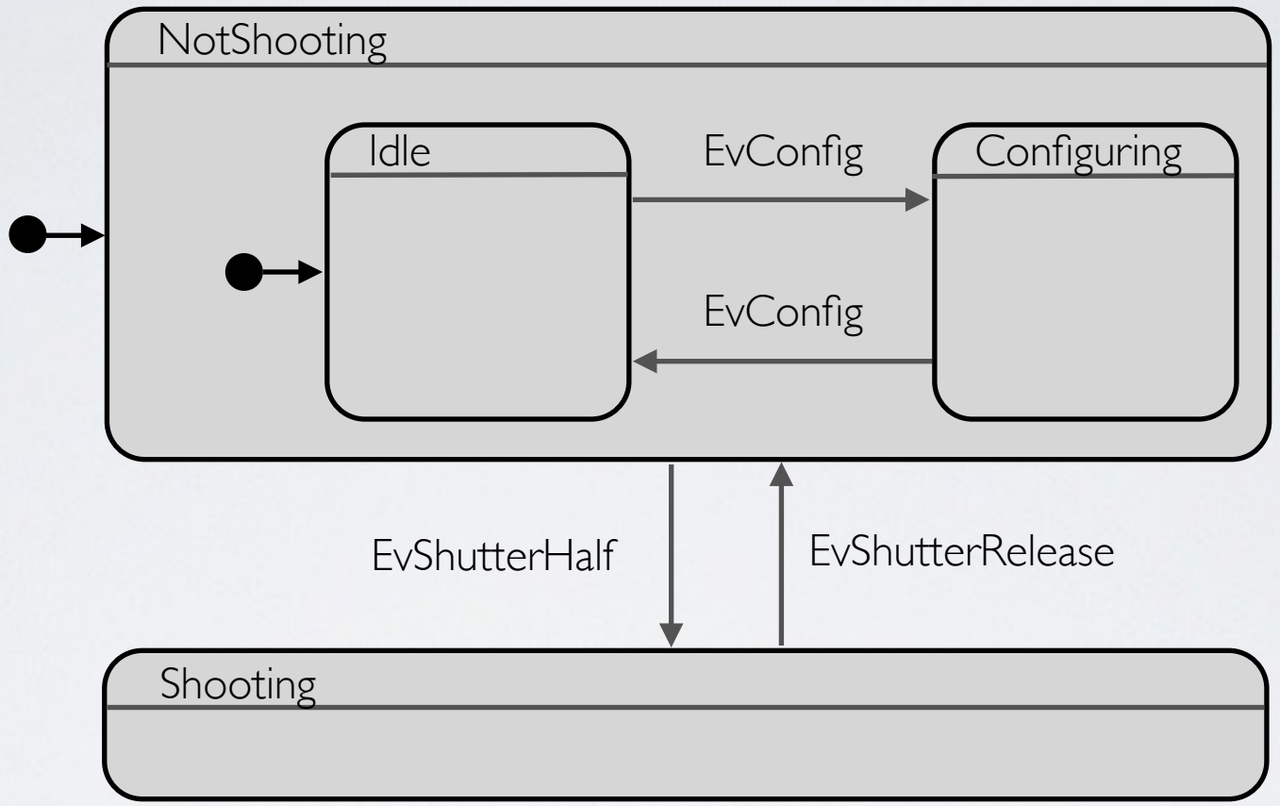
[Quelle:Harel, 1987]



[Quelle:Harel, 1987]

Beispiel - Modell einer Kamera

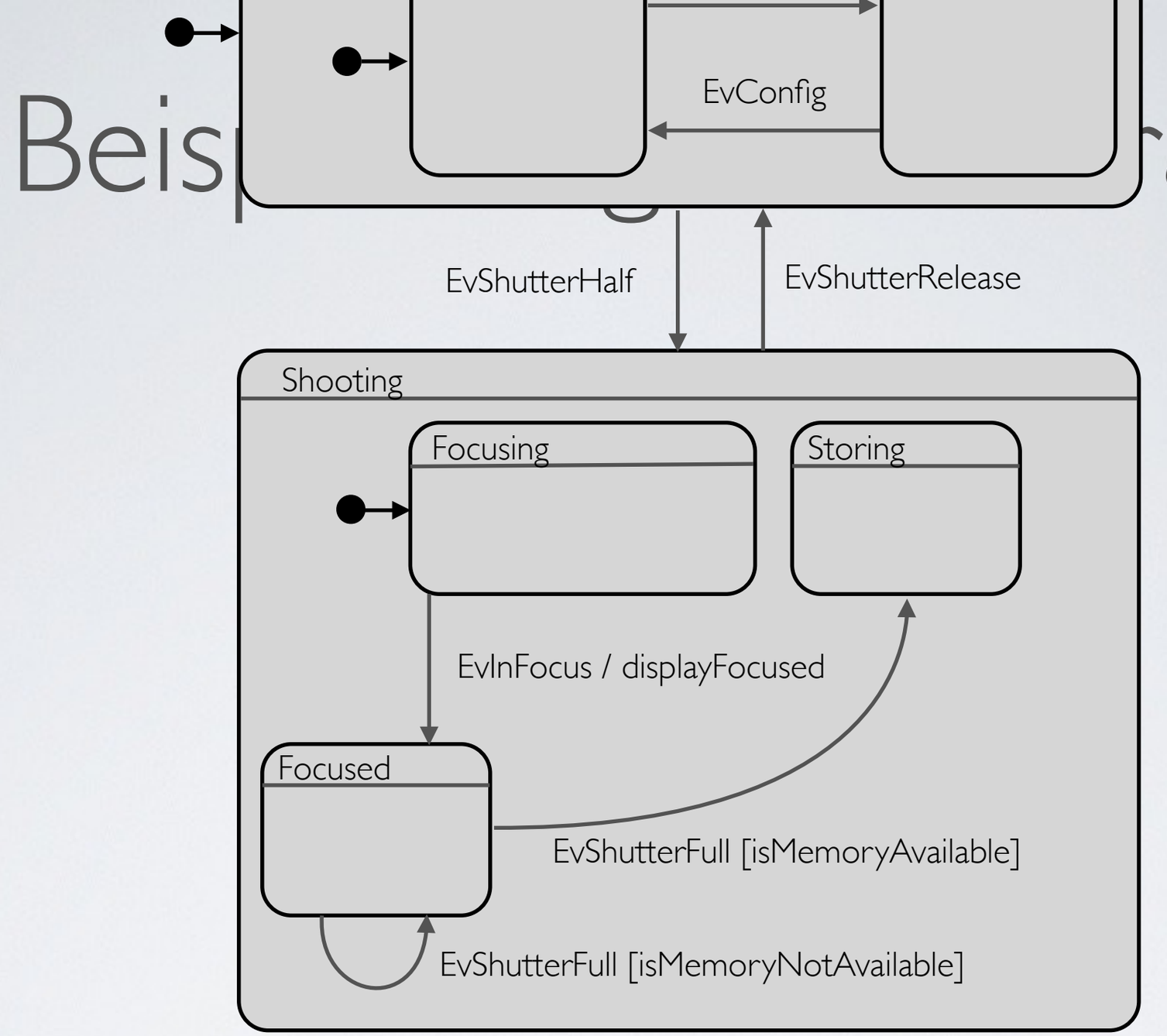
Beispiel: Digitalkamera



- Shutter-Knopf: halb- oder voll gedrückt
  - halbgedrückt aktiviert Aufnahmezustand (*Shooting*)
- Config-Knopf

[Quelle (Beispiel): Boost Statechart Library, 2012]

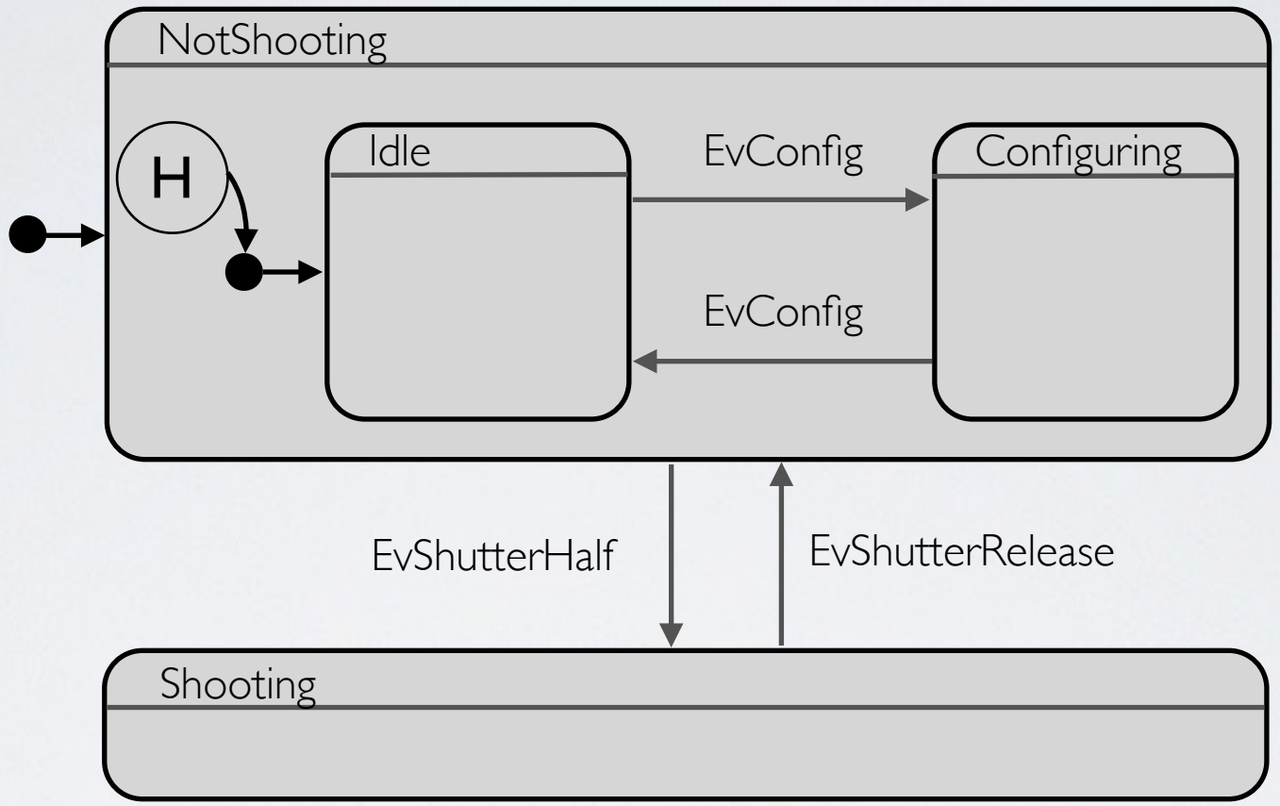
Beispiel



- *Focusing*
  - Fokussieren eines Ziels
- *Focused*
  - Aufnahme eines Bildes bei genug Speicher

[Quelle (Beispiel): Boost Statechart Library, 2012]

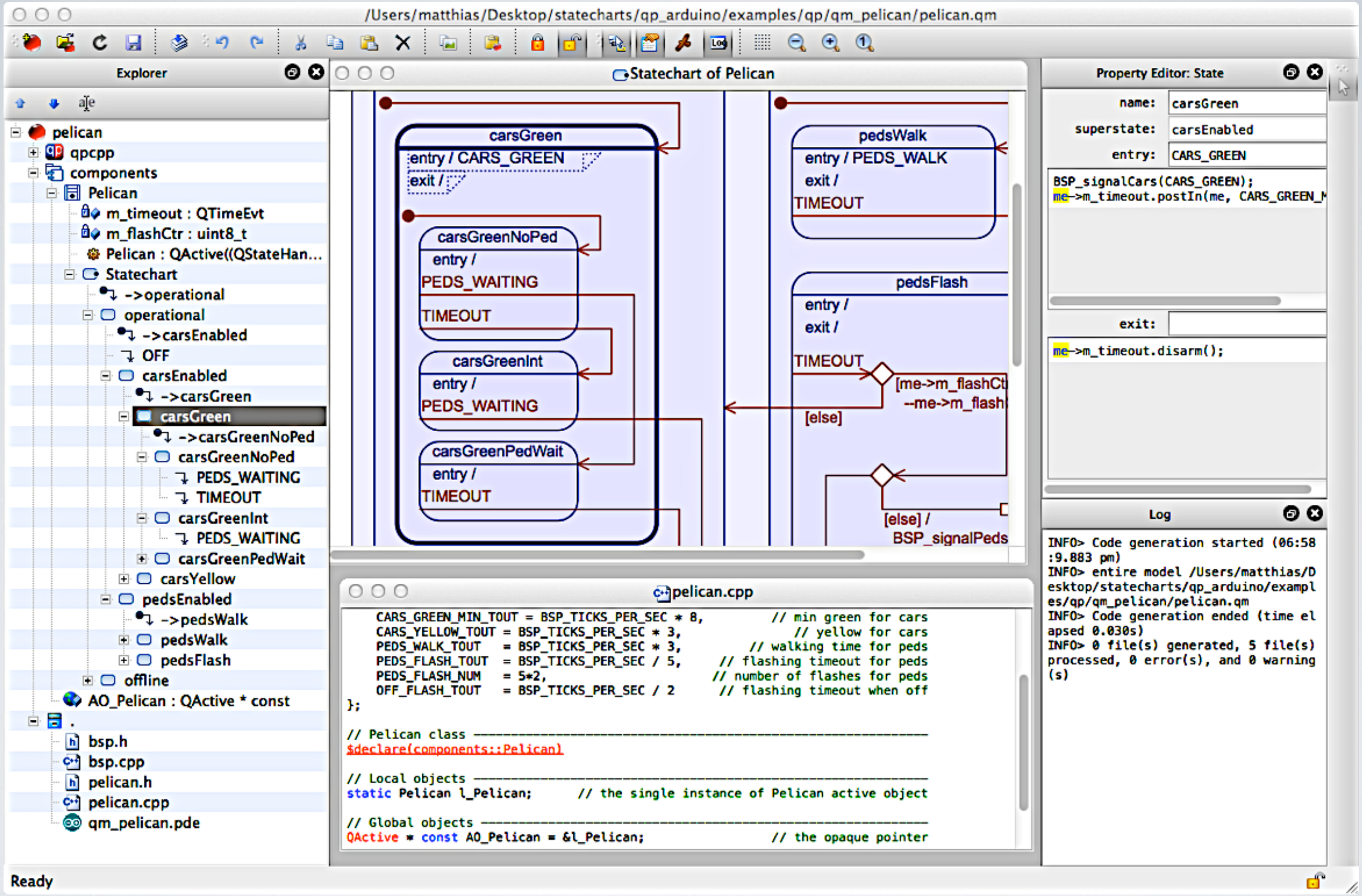
Beispiel: Digitalkamera



- *EvShutterHalf* in *Configure* bzw. *Idle* Zustand ausgelöst?
- Dann Rückkehr in diesen Zustand
- History-Mechanismus

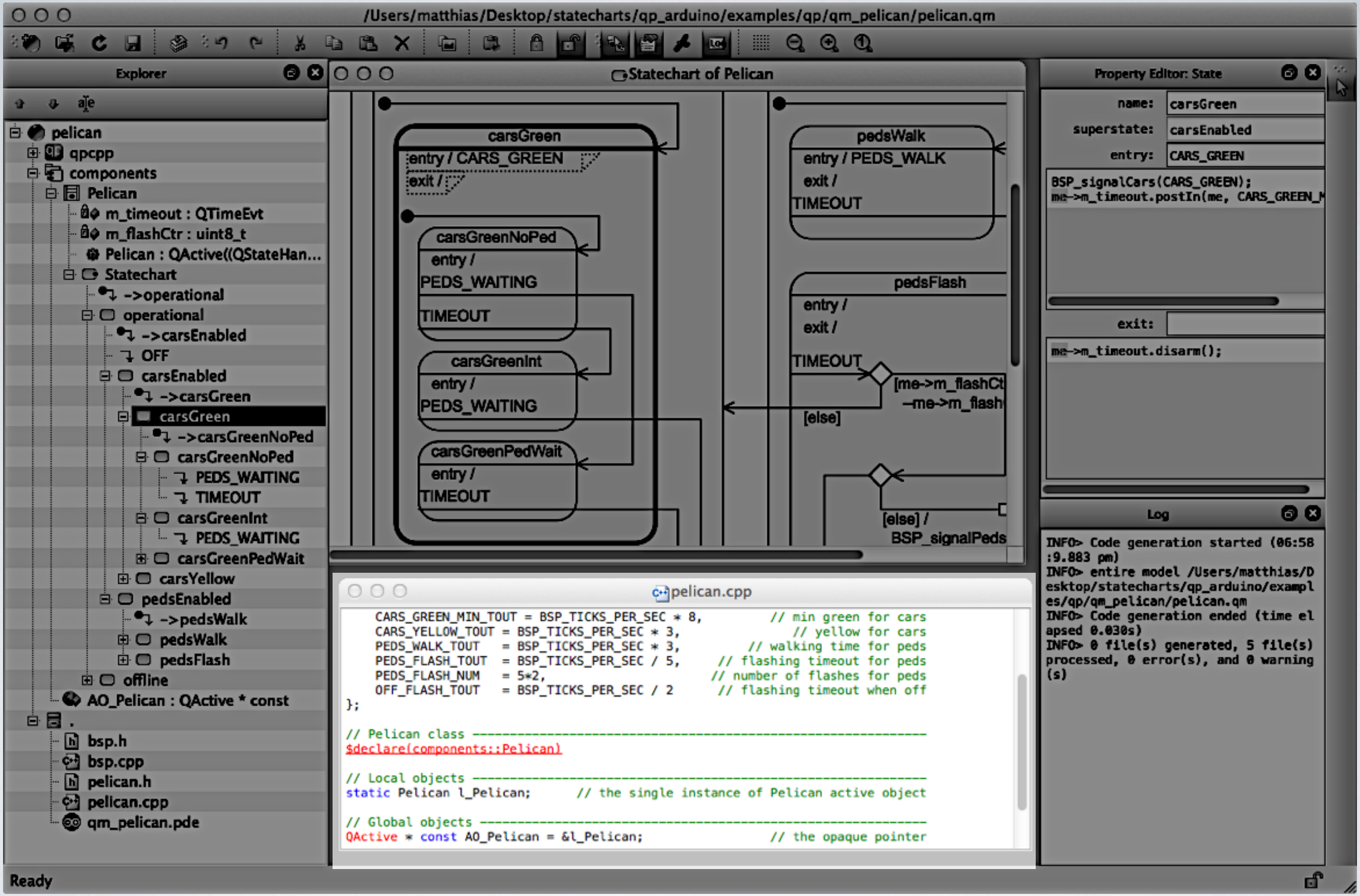
[Quelle (Beispiel): Boost Statechart Library, 2012]

Beispiel - Tool für Statecharts



Beispiel: Modellierung und Codegenerierung mit QM

[Quelle (Software, Beispiel): Quantum Leaps, 2012]



Beispiel: Modellierung und Codegenerierung mit QM

[Quelle (Software, Beispiel): Quantum Leaps, 2012]

pelican.cpp

pelican.cpp 〉 carsGreenPedWait(Pelican *me, QEvent const *e)

```
    CARS_GREEN_MIN_TOUT = BSP_TICKS_PER_SEC * 8,          // min green for cars
    CARS_YELLOW_TOUT = BSP_TICKS_PER_SEC * 3,               // yellow for cars
    PEDS_WALK_TOUT   = BSP_TICKS_PER_SEC * 3,           // walking time for peds
    PEDS_FLASH_TOUT  = BSP_TICKS_PER_SEC / 5,      // flashing timeout for peds
    PEDS_FLASH_NUM   = 5*2,                       // number of flashes for peds
    OFF_FLASH_TOUT   = BSP_TICKS_PER_SEC / 2       // flashing timeout when off
};

// Pelican class -----------------------------------------------------------
// $(components::Pelican) ........................................................
/// PEdestrian LIght CONtrolled (PELICAN) crossing
class Pelican : public QActive {
private:
    QTimeEvt m_timeout;
    uint8_t m_flashCtr;

public:
    /// constructor
    Pelican() : QActive((QStateHandler)&Pelican::initial), m_timeout(TIMEOUT_SIG)
        {
    }

protected:
    static QState initial(Pelican *me, QEvent const *e);
    static QState operational(Pelican *me, QEvent const *e);
    static QState carsEnabled(Pelican *me, QEvent const *e);
    static QState carsGreen(Pelican *me, QEvent const *e);
    static QState carsGreenNoPed(Pelican *me, QEvent const *e);
    static QState carsGreenInt(Pelican *me, QEvent const *e);
    static QState carsGreenPedWait(Pelican *me, QEvent const *e);
    static QState carsYellow(Pelican *me, QEvent const *e);
    static QState pedsEnabled(Pelican *me, QEvent const *e);
    static QState pedsWalk(Pelican *me, QEvent const *e);
    static QState pedsFlash(Pelican *me, QEvent const *e);
    static QState offline(Pelican *me, QEvent const *e);
};

// Local objects -----------------------------------------------------------
static Pelican l_Pelican;      // the single instance of Pelican active object

// Global objects ----------------------------------------------------------
QActive * const AO_Pelican = &l_Pelican;                // the opaque pointer
```

pelican.cpp

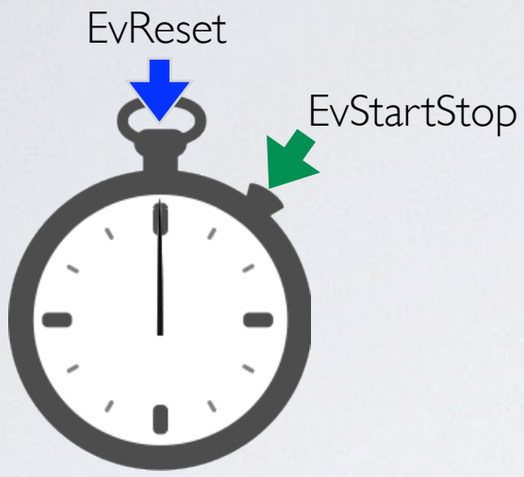
```
    CARS_GREEN_MIN_TOUT = BSP_TICKS_PER_SEC * 8,          // min green fo
    CARS_YELLOW_TOUT = BSP_TICKS_PER_SEC * 3,               // yellow fo
    PEDS_WALK_TOUT   = BSP_TICKS_PER_SEC * 3,           // walking time fo
    PEDS_FLASH_TOUT  = BSP_TICKS_PER_SEC / 5,      // flashing timeout fo
    PEDS_FLASH_NUM   = 5*2,                       // number of flashes fo
    OFF_FLASH_TOUT   = BSP_TICKS_PER_SEC / 2       // flashing timeout wh
};

// Pelican class -----------------------------------------------------------
$declare(components::Pelican)

// Local objects -----------------------------------------------------------
static Pelican l_Pelican;      // the single instance of Pelican active

// Global objects ----------------------------------------------------------
QActive * const AO_Pelican = &l_Pelican;                // the opaque p
```

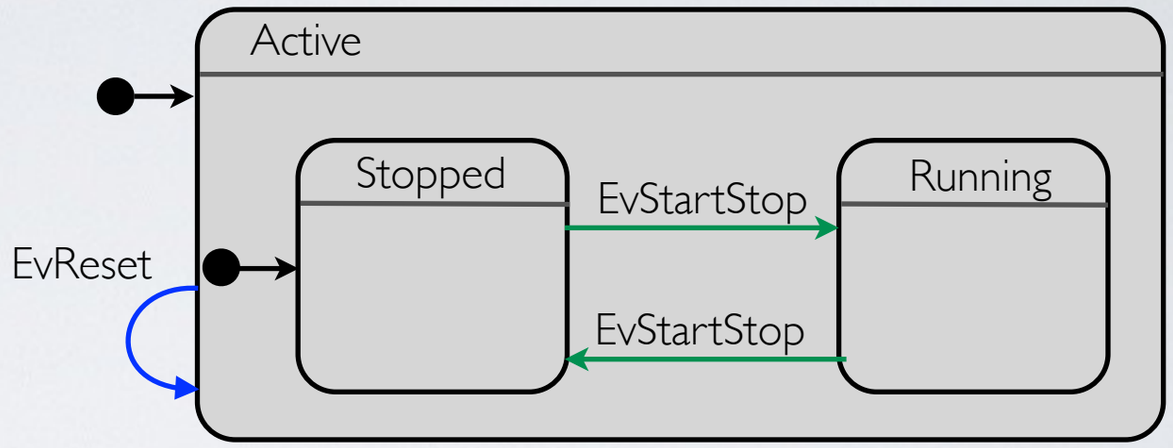
Beispiel: Modellierung und Codegenerierung mit QM

[Quelle (Software, Beispiel): Quantum Leaps, 2012]

Beispiel - Programmierbibliothek

Implementierungbeispiel: *Stopp Uhr*





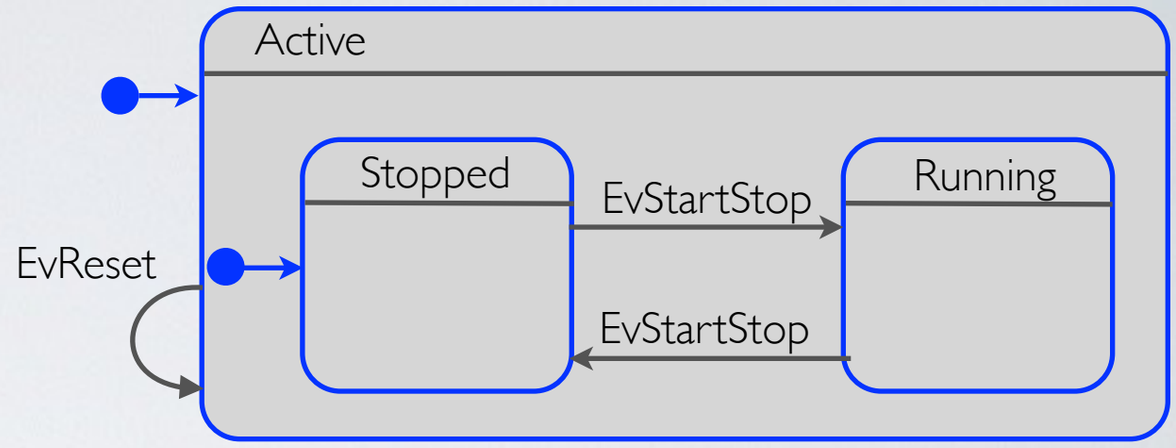
Stopp Uhr modelliert durch die Zustände:

- *Stopped*: Anhalten der Zeit.
  - Reset: Uhr zurück in 0 Position. Bleibt im *Stopped* Zustand.
  - Start/Stopp: Übergang in *Running* Zustand.

- *Running:* Zeit läuft und wird gezeigt.
  - Reset: Uhr zurück in 0 Position. Übergang in *Stopped* Zustand.
  - Start/Stopp: Übergang in *Stopped* Zustand.

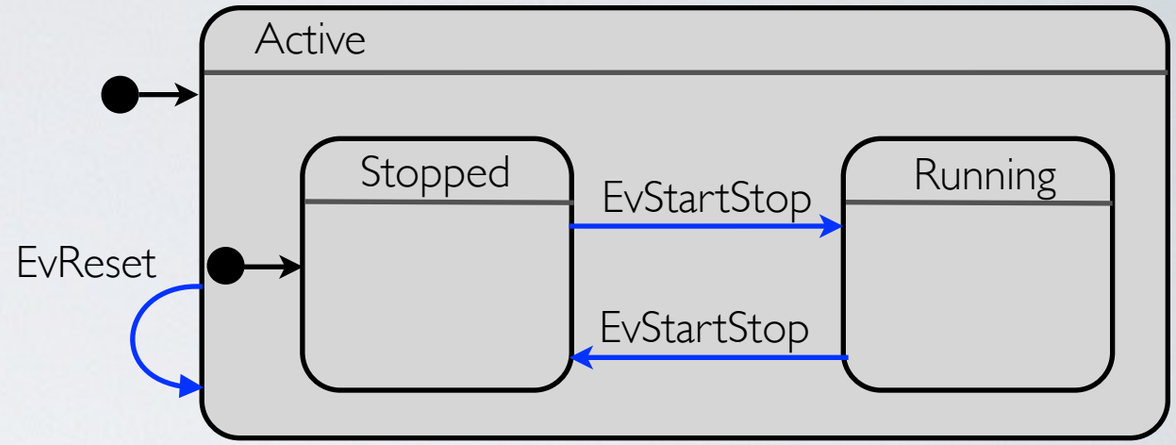
Beispiel: *Stopp Uhr* mit C++ & Boost Bibliothek

```
// setup
#include <boost/statechart/event.hpp>
#include <boost/statechart/state_machine.hpp>
#include <boost/statechart/simple_state.hpp>

using namespace boost::statechart;

// event<most_derived_class>
struct EvStartStop : event< EvStartStop > {};
struct EvReset : event< EvReset > {};

// forward declaration
struct Active;
struct Stopped;

// state_machine<most_derived_class, initial_state>
struct StopWatch : state_machine< StopWatch, Active > {};

// States
// simple_state<most_derived_class, context, initial_state>
struct Active : simple_state< Active, StopWatch, Stopped > {};
struct Running : simple_state< Running, Active > {};
struct Stopped : simple_state< Stopped, Active > {};


int main() {
    StopWatch myWatch;
    myWatch.initiate();
    return 0;
}
```



Beispiel: *Stopp Uhr* mit C++ & Boost Bibliothek

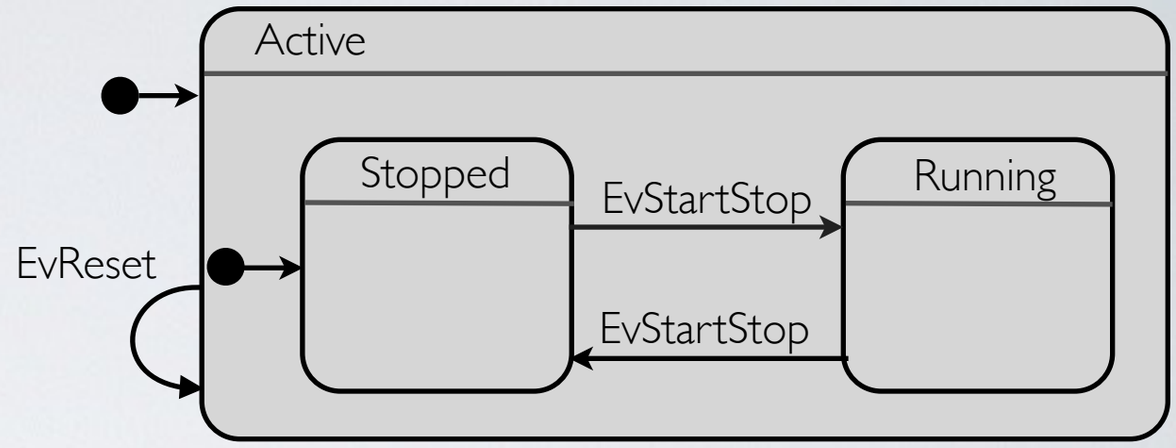
```
#include <boost/statechart/transition.hpp>
...

// State active
struct Active : simple_state< Active, StopWatch, Stopped > {
    typedef transition< EvReset, Active > reactions;

};

// State running
struct Running : simple_state< Running, Active > {
    typedef transition< EvStartStop, Stopped > reactions;

};

// State stopped
struct Stopped : simple_state< Stopped, Active > {
    typedef transition< EvStartStop, Running > reactions;
};


int main() {
    StopWatch myWatch;
    myWatch.initiate();
    myWatch.process_event( EvStartStop() );
    myWatch.process_event( EvStartStop() );
    myWatch.process_event( EvReset() );
    return 0;
}
```



Beispiel: *Stopp Uhr* mit C++ & Boost Bibliothek

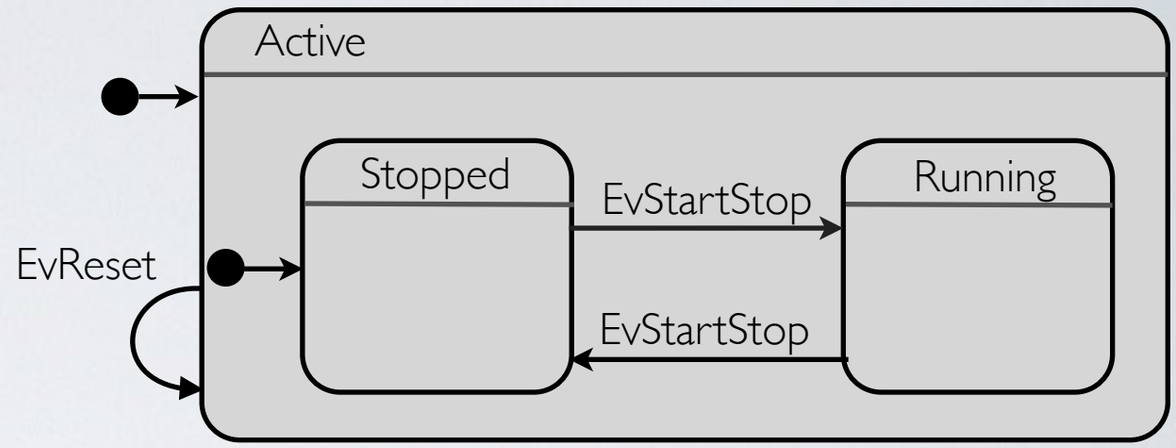
...

```
// State active
struct Active : simple_state< Active, StopWatch, Stopped > {
    typedef transition< EvReset, Active > reactions;

    Active() : elapsedTime_( 0.0 ) {}
    double & elapsedTime() { return elapsedTime_; }
private:
    double elapsedTime_;
};

// State running
struct Running : simple_state< Running, Active > {
    typedef transition< EvStartStop, Stopped > reactions;

    Running() : startTime_( std::time( 0 ) ) {}
    ~Running() { context< Active >().elapsedTime() += std::difftime( std::time( 0 ), startTime_ ); }
private:
    std::time_t startTime_;
};

// State stopped
struct Stopped : simple_state< Stopped, Active > {
    typedef transition< EvStartStop, Running > reactions;
};


int main() {
    StopWatch myWatch;
    myWatch.initiate();
    myWatch.process_event( EvStartStop() );
    myWatch.process_event( EvStartStop() );
    myWatch.process_event( EvReset() );
    return 0;
}
```



Beispiel: *Stopp Uhr* mit C++ & Boost Bibliothek

```
struct IElapsedTime {
    virtual double elapsedTime() const = 0;
};
// State machine
struct StopWatch : state_machine< StopWatch, Active > {
    double elapsedTime() const {
      return state_cast< const IElapsedTime & >().elapsedTime(); }
};
// State active
struct Active : simple_state< Active, StopWatch, Stopped > {
    typedef transition< EvReset, Active > reactions;
    Active() : elapsedTime_( 0.0 ) {}
    double elapsedTime() const { return elapsedTime_; }
    double & elapsedTime() { return elapsedTime_; }
private:
    double elapsedTime_;
};
// State running
struct Running : IElapsedTime, simple_state< Running, Active > {
    typedef transition< EvStartStop, Stopped > reactions;
    Running() : startTime_( std::time( 0 ) ) {}
    ~Running() { context< Active >().elapsedTime() = elapsedTime(); }
    virtual double elapsedTime() const { return context< Active >().elapsedTime()
                                          +  std::difftime( std::time( 0 ), startTime_ ); }
private:
    std::time_t startTime_;
};
// State stopped
struct Stopped : IElapsedTime, simple_state< Stopped, Active > {
    typedef transition< EvStartStop, Running > reactions;
    virtual double elapsedTime() const { return context< Active >().elapsedTime(); }
};

int main() {
  StopWatch watch;
  watch.initiate();
  watch.process_event(EvStartStop());
  sleep(1);
  watch.process_event(EvStartStop()); //watch.elapsedTime() = 1
  watch.process_event(EvStartStop());
  sleep(3);                           //watch.elapsedTime() = 4
  watch.process_event(EvReset());     //watch.elapsedTime() = 0
  return 0;
}
```



Statecharts

- Erweiterung von endlichen Automaten durch
  - Hierarchie / Modularität
  - Nebenläufigkeit

- Ähnliche/verwandte Ansätze vorhanden
  - UML Statechart Diagramme
  - Rhapsody

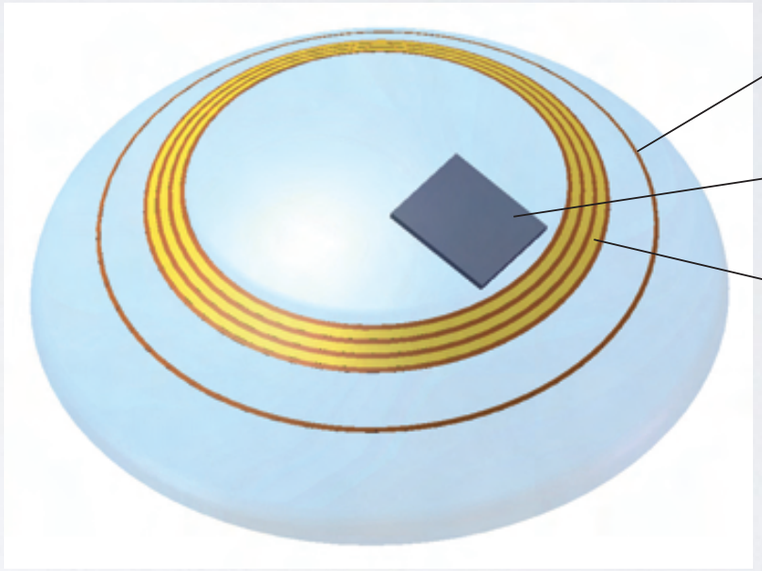
- Begrenzt einsetzbar für
  - komplexe Berechnungsverfahren
  - verteilte Systeme

Statecharts: Vor- und Nachteile

+ Hierarchische Beschreibung endlicher Automaten
+ Nebenläufigkeit
+ Werkzeugunterstützung
- Unübersichtlich für komplexe Modelle
- Grenzen bei der Wiederverwendung
- Nicht ausgelegt für verteilte Systeme

Beispiel einer wenig bekannten Anwendung: Kontaktlinse

- 2010
  - Kleines Embedded System
  - Messung von Augeninnendruck
  - Mikroprozessor, Dehnungssensor, Antenne
  - Energie über Radiowellen
- 2014
  - Google schützt Kontaktlinse mit eingebauter Kamera



[Quelle: Sensimed AG, SENSIMED Triggerfish® brochure, 2010]

Literatur / Quellen

- Arduino, URL: http://www.arduino.cc, 2012
- Boost Statechart Library, URL, http://www.boost.org/doc/libs/1_35_0/libs/statechart/doc/tutorial.htm, 2012
- Harel, Statecharts: A Visual Formalism For Complex Systems, Science of Computer Programming 8, 1987
- Marwedel, Eingebettete Systeme, Springer-Verlag, 2008
- Sensimed AG, SENSIMED Triggerfish® brochure, URL: http://www.sensimed.ch/images/pdf/sensimed_brochure_for_office_use.pdf, 2010
- Quantum Leaps, QP™ Modeler, URL: http://www.state-machine.com/qm/index.php, 2012
- **Stand aller Internetquellen: 10.04.2012**