Finite-State Machines

Networks and Embedded Software

Module 5.3.1 (optional)

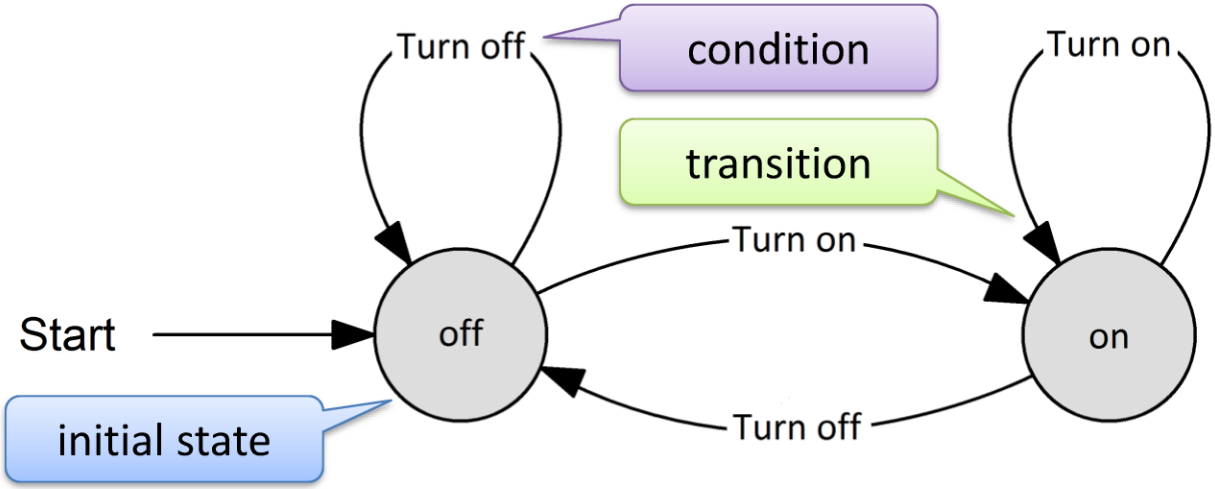
by Wolfgang Neff

Finite State Machines (1)

- Components
  - States
  - Transitions
  - Conditions
  - An initial state
- Basic concept
  - State says what to do
  - Conditions control state switching

Finite State Machines (2)

- Representation
  - State diagram (cf. Module 3.3.2)



Finite State Machines (3)

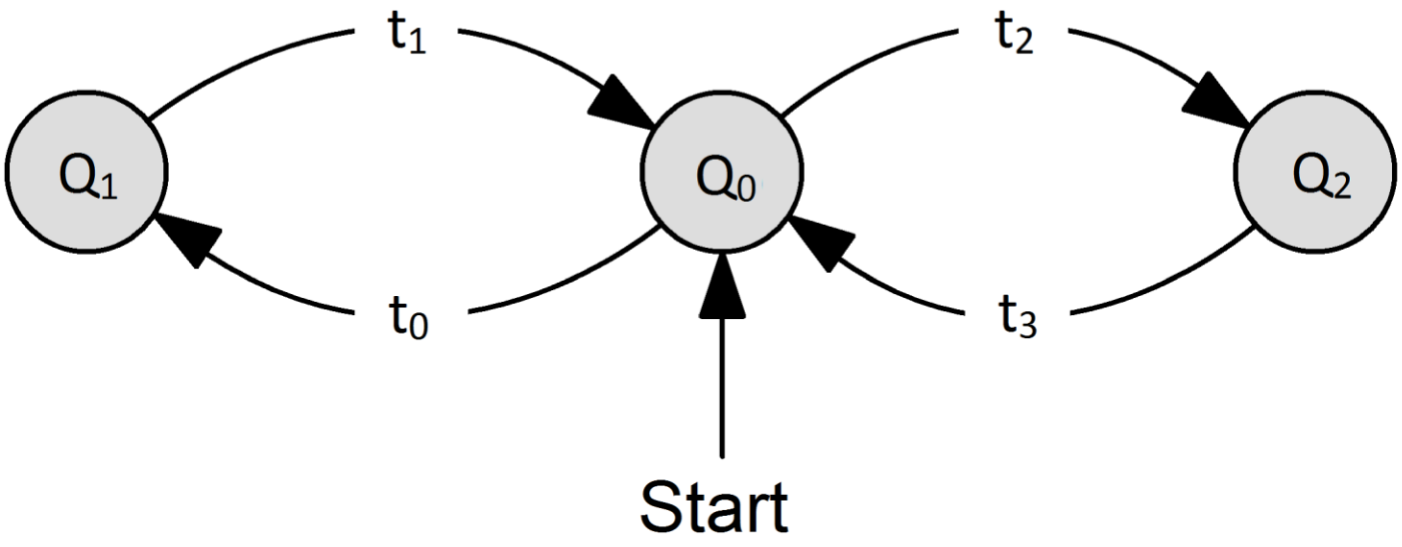
- Example
  - A LED is controlled by three buttons
    - If button 0 is pressed the LED is turned on.
    - If button 1 is pressed the LED is turned off.
    - While button 2 is pressed the LED is turned on.
  - States
    - Q_0: LED is off
    - Q_1: LED is on, waiting for a press on button 1
    - Q_2: LED is on, waiting for the release of button 2

Finite State Machines (4)

- Example (continued)
  - Transitions
    - t_0: State Q_0, button 0 pressed → state Q_1
    - t_1: State Q_1, button 1 pressed → state Q_0
    - t_2: State Q_0, button 2 pressed → state Q_2
    - t_3: State Q_2, button 2 released → state Q_0

Finite State Machines (5)

- Example (continued)
  - State diagram



Finite State Machines (6)

- Example (continued)
  - Program

```
int state=0;
while(1) {
  switch (state) {
    case 0:
      // Handle state
      led = off;
      // Handle transitions
      if (button0) state=1;
      if (button2) state=2;
      break;
```

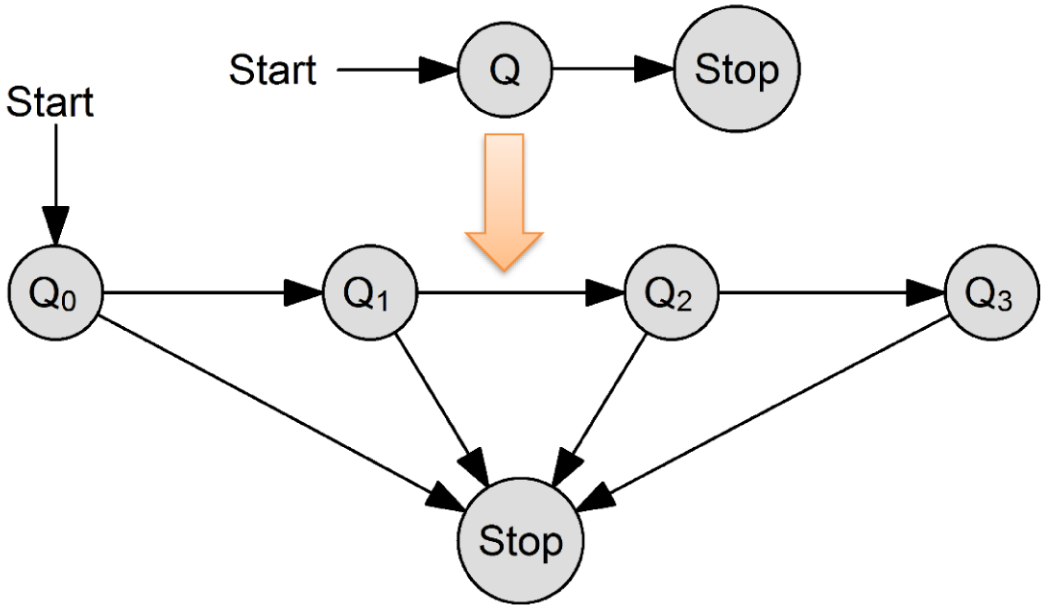
Finite State Machines (7)

- Example (finished)
  - Program (continued)

```
      case 1:
        led = on;
        if (button1) state=0;
        break;
      case 2:
        led = on;
        if (!button2) state=0;
        break;
    } /* switch */
  } /* while */
```

Finite State Machines (8)

- Fine-Grained Control
  - Split state into more states



Finite State Machines (9)

- Example
  - Program

```
int state=0;
while(1) {
  switch (state) {
    case 0:
      doSomething(0);
      if (button) state = STOP;
      else state = 1;
      break;
    case 1:
      doSomething(1);
      …
```

Finite State Machines (10)

- Example (continued)
  - Program (continued)

```
void doSomething(int action) {
  switch (action) {
    case 0:
      action0();
      return;
    case 1:
      action1();
      …
  } /* switch */
} /* function */
```

Finite State Machines (11)

- Scheduling a Chronology
  - Timer controls transitions
    - `ISR(TIMER_OVF_vect) {`
    - `  state++;`
    - `  …`
    - `}`