# The Embedded Computing Platform

- CPU bus: Backbone of the H/W system
- Memory.
- I/O devices: Timer/counters, A/D, D/A, Keyboard, Touch-screen, ...
- Design:  Seat-belt controller, alarm clock

Taken/modified from "Computers as Components, W. Wolf"

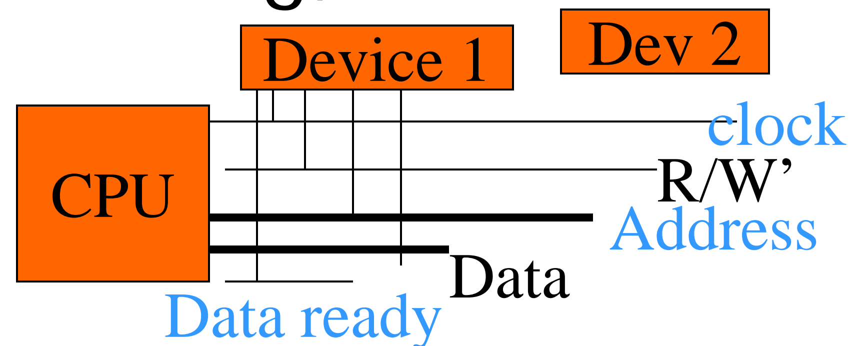
# CPU bus: Wires by which CPU communicates with memory and devices

- Connects CPU to:
  - memory;
  - I/O devices.
- A protocol controls communication between entities.

Taken/modified from "Computers as Components, W. Wolf"

# Typical bus signals

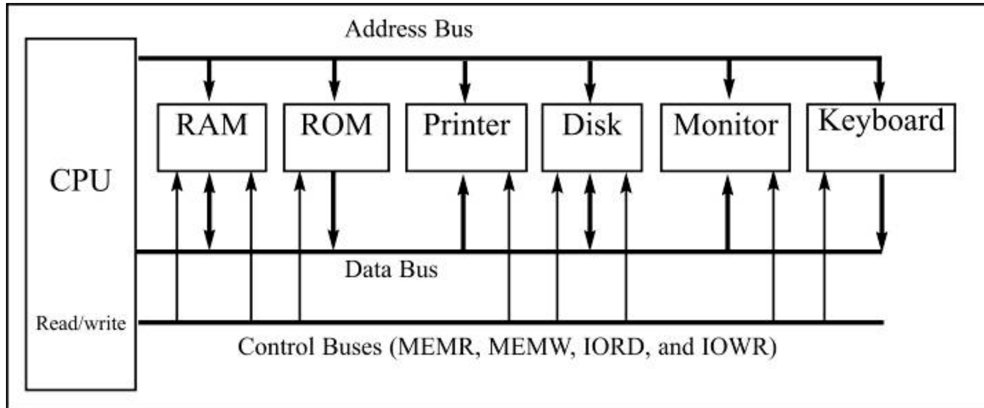
- Clock: Provides synchronization to bus components
- R/W′: true when bus is reading, false when writing
- Address: a-bit bundle.
- Data: n-bit bundle.
- Data ready′: Signals when values on data bus are ready

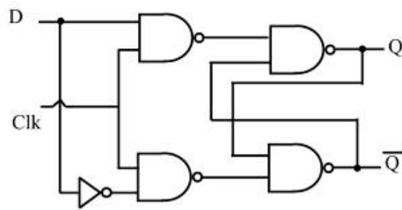



Taken/modified from "Computers as Components, W. Wolf"

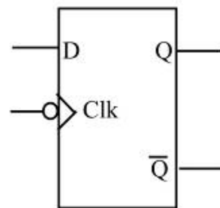
# Computer Buses

**How is the problem of multiple sources of information (inputs) and outputs on a data bus handled?**




(a) Circuit diagram



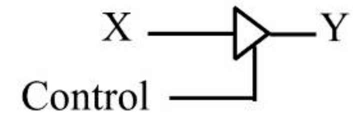

(b) Block diagram

| Clk | D | Q |
|---|---|---|
| No | x | no change |
| ↴ | 0 | 0 |
| ↴ | 1 | 1 |

x = don't care

(c) Truth table

**Buffer**

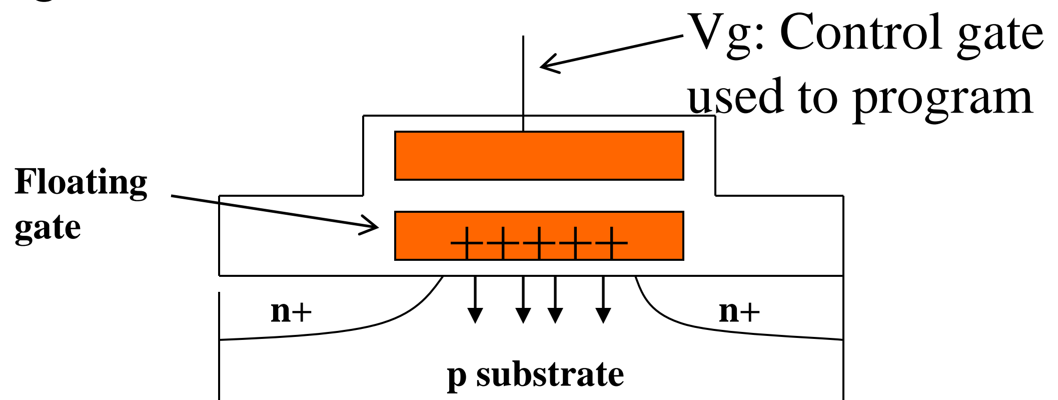



***Tri-state buffer -*** does not change the logic level of the input. It is used to isolate or amplify the signal.

Taken/modified from "HCS12 Microcontroller …, Mazidi/Causey"

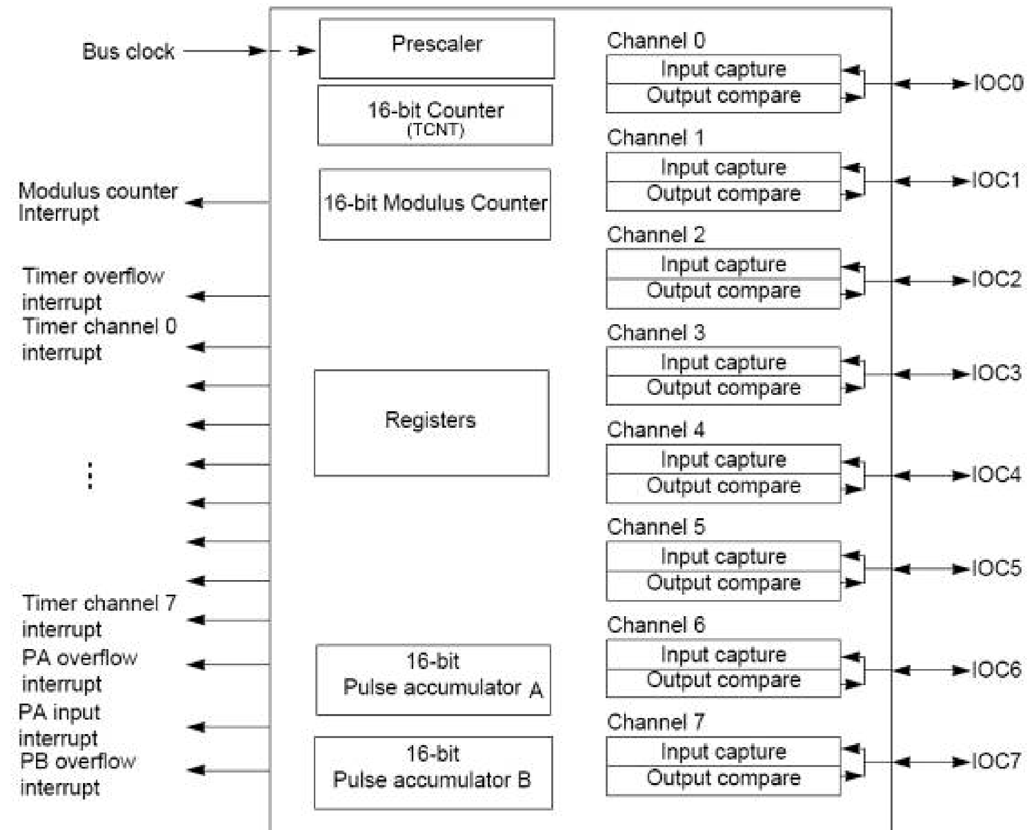
# Memory Devices

- RAM (Random Access Memory): SRAM (feedback cct) & DRAM (switch+capacitor)
- PROM (fuses blown), EPROM (floating-gate transistors charged/ UV to erase)
- Flash Memory: Evolution of Electrically Erasable Memory EEPROM- holds data on a floating transistor gate



# Timers and counters

- Very similar:
  - a timer is incremented by a periodic signal;
  - a counter is incremented by an asynchronous, occasional signal.
- Rollover causes interrupt.

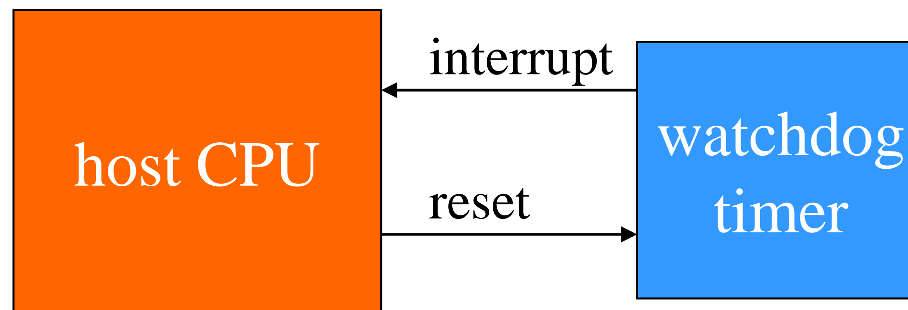



HCS12 Timer/Counter Blocks

Taken/modified from "Computers as Components, W. Wolf"

# Watchdog timer

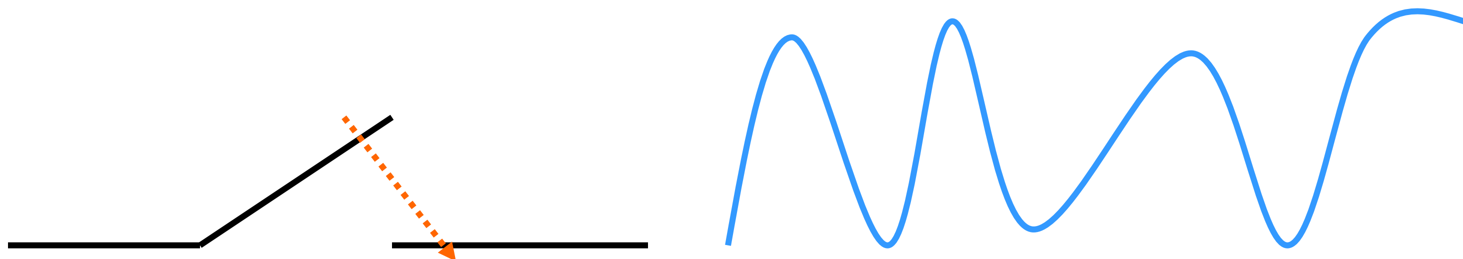
- Watchdog timer is periodically reset by system timer.
- If watchdog is not reset, it generates an interrupt to reset the host.



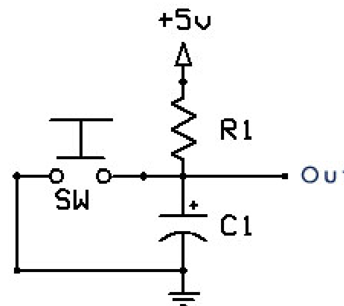

Taken/modified from "Computers as Components, W. Wolf"

# Switch debouncing

- A switch must be debounced to eliminate  multiple contacts caused by mechanical bouncing: Hardware (single-shot), software can be used



Hardware debouncing

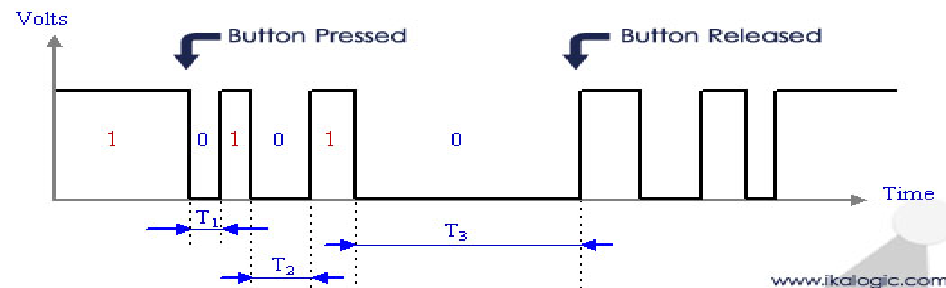



Taken/modified from "Computers as Components, W. Wolf"

# Software de-bouncing

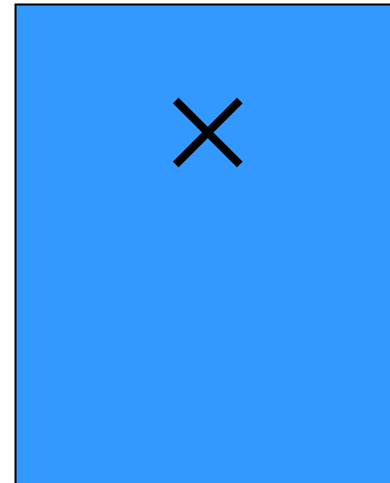
```
unsigned  char  counter;
unsigned  char T_valid;    //minimum duration of a valid pulse

  DDRA = 0x00;             // Initialize port A as input port
  T_valid = 100;     //Number representing how long to wait

  while(counter < 255){
    if (counter  <  255){    //prevent the counter to roll back to 0
                         counter++;
                        }
    if   ( PORTA_BIT0  ==   1 ){
                          counter = 0;    //reset the counter back to 0
                             }
    if (counter  >  T_valid)   break;
            } //while

// code after debouncing ….
```

Switch connected to bit 0 of PORTA



# Touchscreen

- Includes **input** and **output** devices:
- Input device is a two-dimensional voltmeter
- Output device?

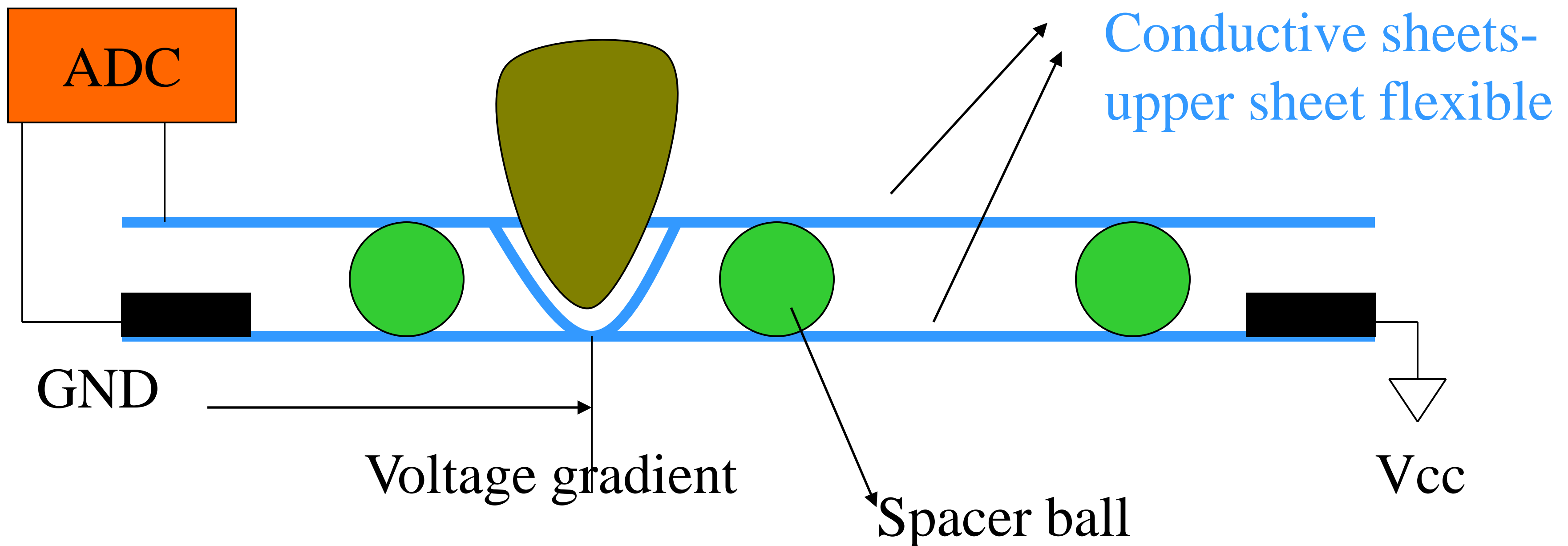


Taken/modified from "Computers as Components, W. Wolf"

# Touchscreen position sensing

-Resistive (or capacitive)

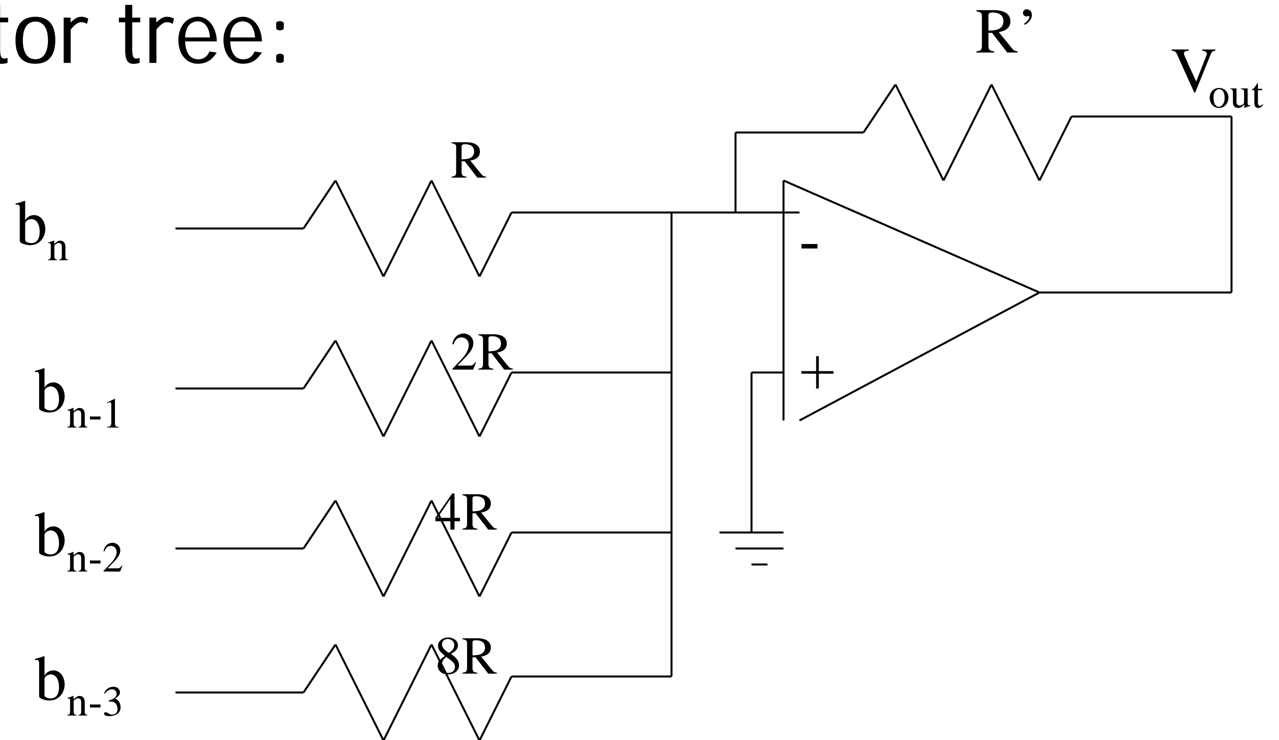
- Alternates between x and y position sensing




Taken/modified from "Computers as Components, W. Wolf"

# Digital-to-analog conversion

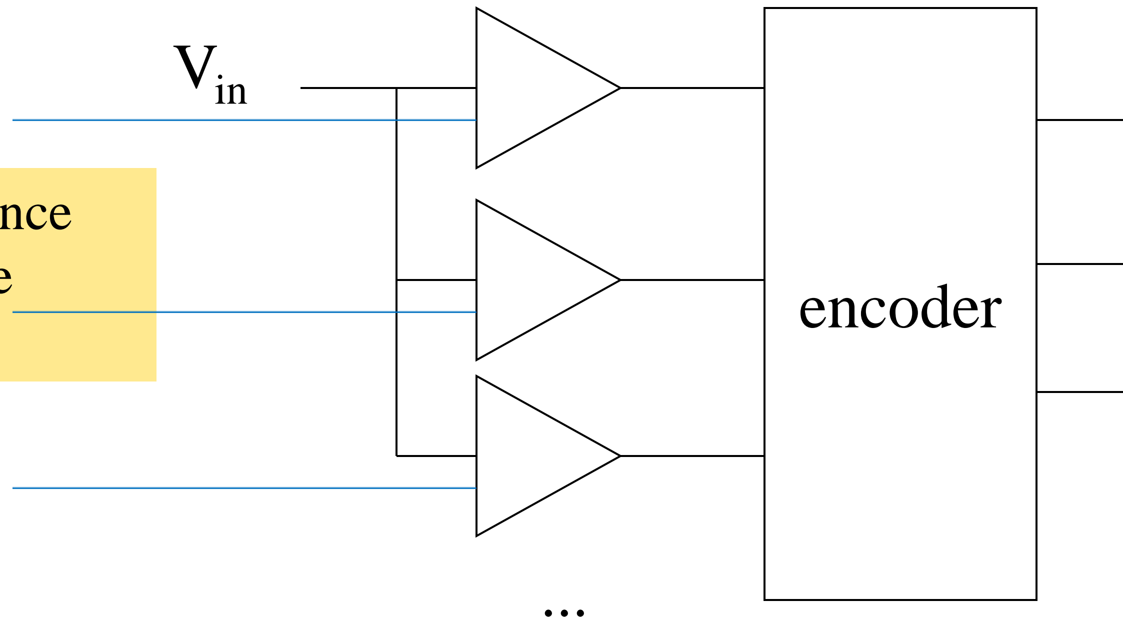
- Use resistor tree:




Taken/modified from "Computers as Components, W. Wolf"

# Flash A/D conversion

- N-bit result requires $2^n$ comparators:




Reference voltage levels

Taken/modified from "Computers as Components, W. Wolf"

... HCS12

➔ Successive approximation ADC

# … Flash ADC

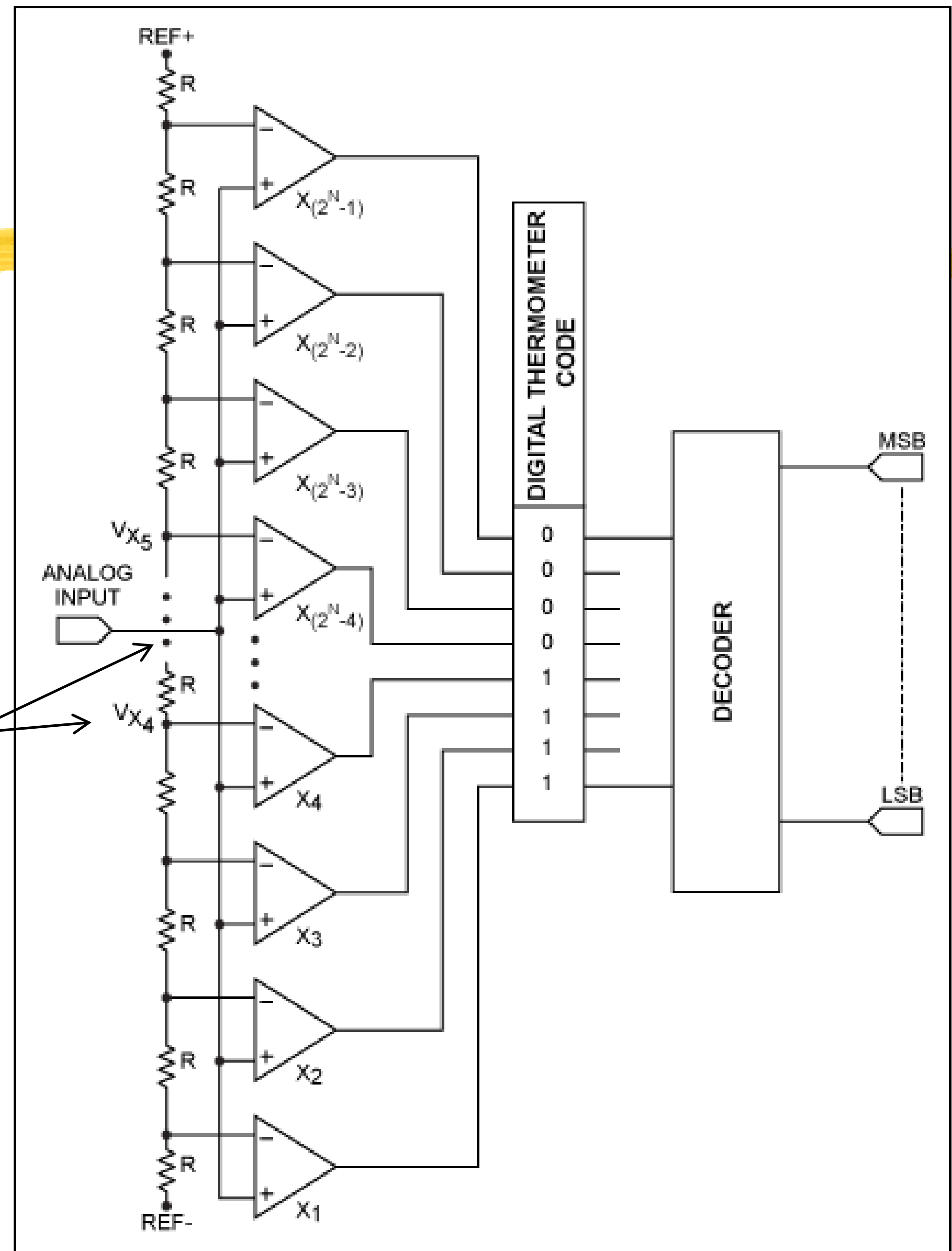
- For an N-bit converter, the circuit employs $2^N$-1 comparators.

- Each comparator produces a 1 when its analog input voltage is higher than the reference voltage applied to it. Otherwise it is zero

- Example: If $V_{X4} < Vin < V_{X5}$

➔comparators $X_1$ through $X_4$ produce 1s and the remaining comparators produce 0s.

- The thermometer code is then decoded to the appropriate digital output code.



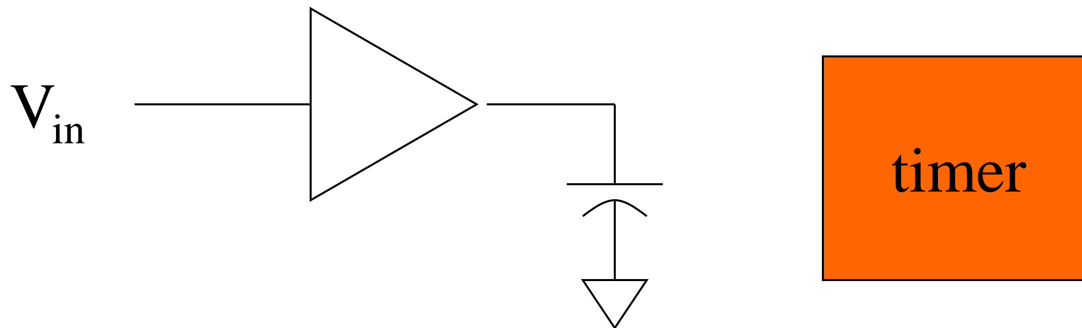

ADC diagram from maxim.com

# Thermometer code vs Binary code

| Binary | | | Thermometer | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | B | C | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Dual-slope conversion

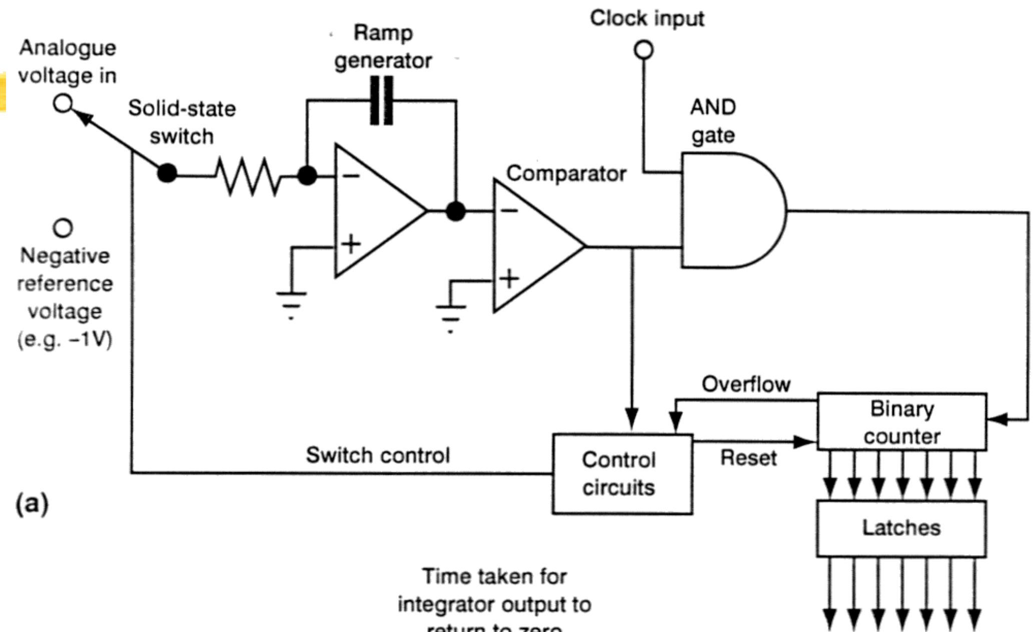
- Use counter to measure the time required to charge/discharge capacitor.
- Charging, then discharging eliminates non-linearities.

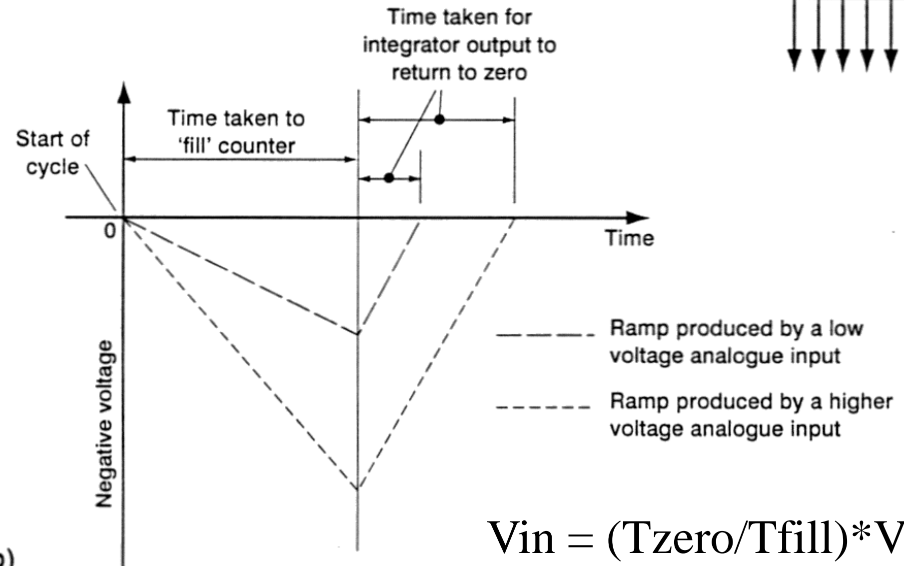



Taken/modified from "Computers as Components, W. Wolf"

# Dual slope ADC: High res. but slow

- The integrator generates a negative ramp whose slope is proportional to the analog input voltage
- The comparator goes HIGH, enabling clock pulses into the counter
- When counter overflows, it resets to zero and the control circuit switches the switch to a reference negative voltage
- Vin can be obtained by using Tzero, Tfill, Vref

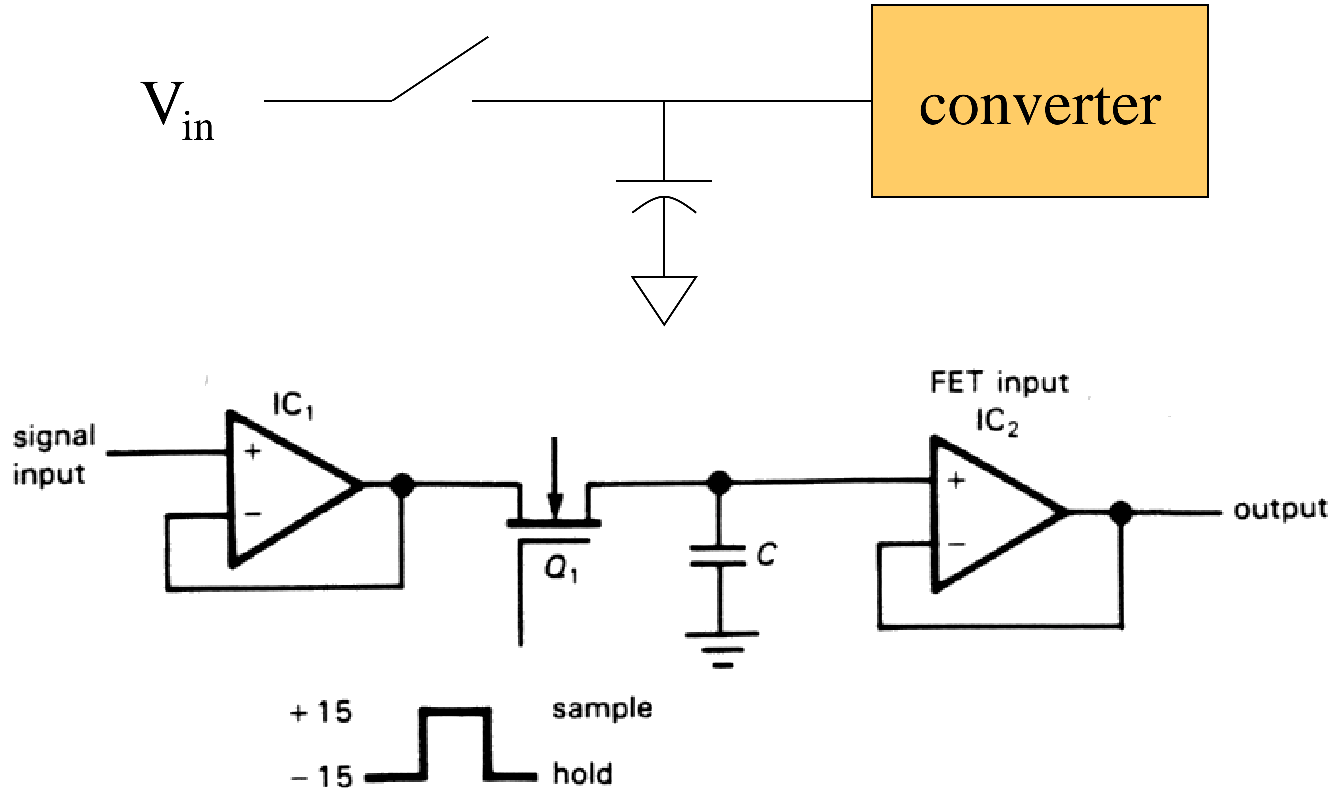



Vin = (Tzero/Tfill)*Vref

# Sample-and-hold

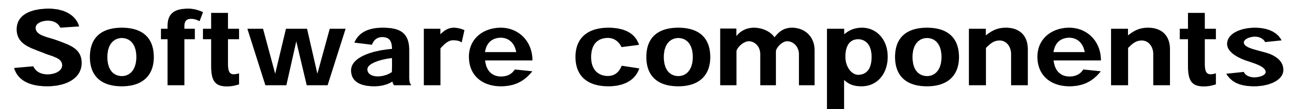
- Required in any A/D:




Taken/modified from "Computers as Components, W. Wolf"

- Architectures and components:
  - software;
  - hardware.
- Debugging.
- Testing.

# Software components



- Need to break the design up into pieces to be able to write the code.
- Some component designs come up often
  → Glue logic
- A design pattern is a generic description of a component that can be customized and used in different circumstances
  → e.g., an object

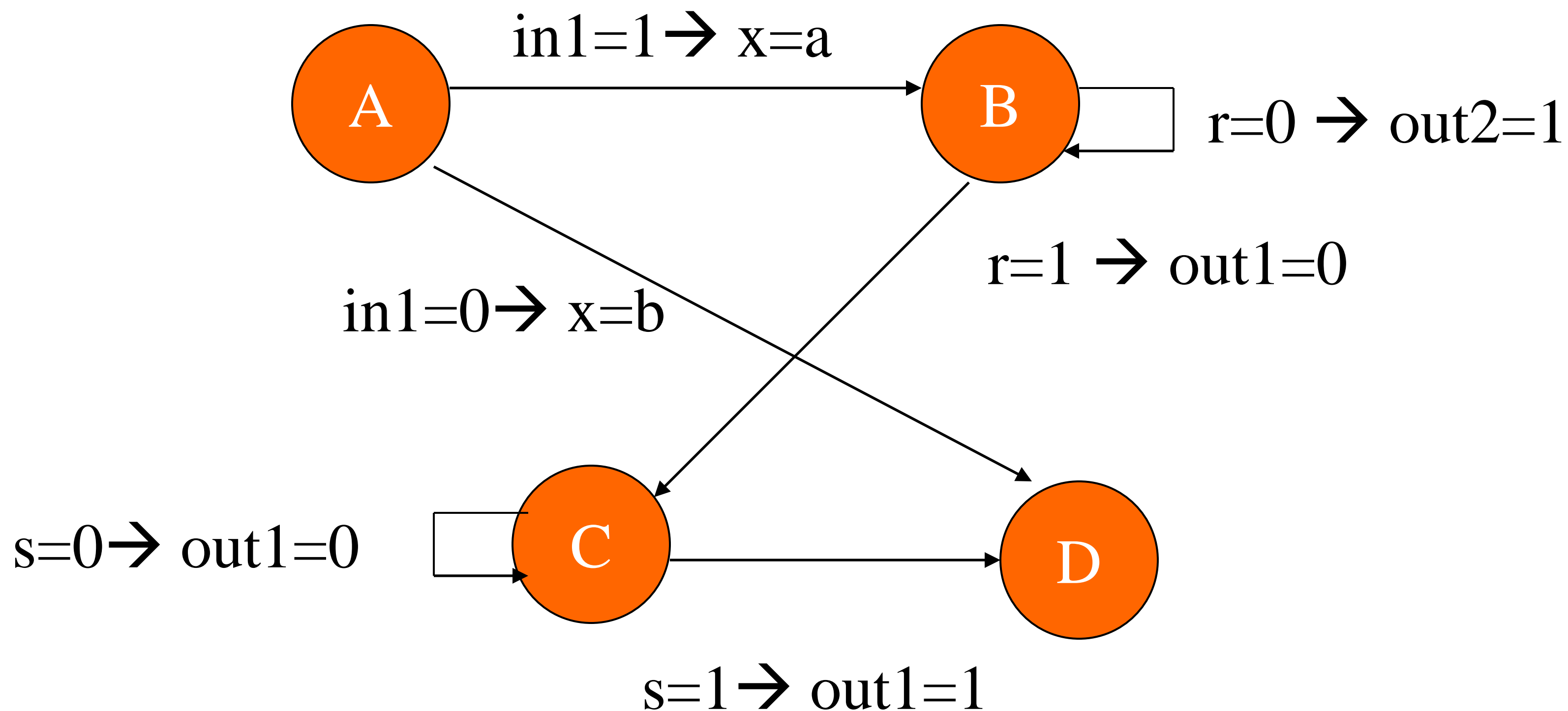
Taken/modified from "Computers as Components, W. Wolf"

# State Machine: A design pattern

- State machine: Output depends on current input and state
- A state machine keeps internal state as a variable, changes state based on inputs.
- Uses:
  - control-dominated code;
  - reactive systems.

Taken/modified from "Computers as Components, W. Wolf"

# State machine specification




Taken/modified from "Computers as Components, W. Wolf"

# C code structure

- Current state is kept in a variable.
- State table is implemented as a switch.
  - Cases define states.
  - States can test inputs.
- Switch is repeatedly evaluated in a while loop.

# C state machine structure

```
while (TRUE) {
  switch (state) {
     case state1: ......
  }
}
```

# C state table

```
switch (state) {
case A: if (in1==1) { x = a; state = B; }
  else { x = b; state = D; }
  break;
case B: if (r==0) { out2 = 1; state = B; }
  else { out1 = 0; state = C; }
  break;
case C: if (s==0) { out1 = 0; state = C; }
  else { out1 = 1; state = D; }
  break;
}
```

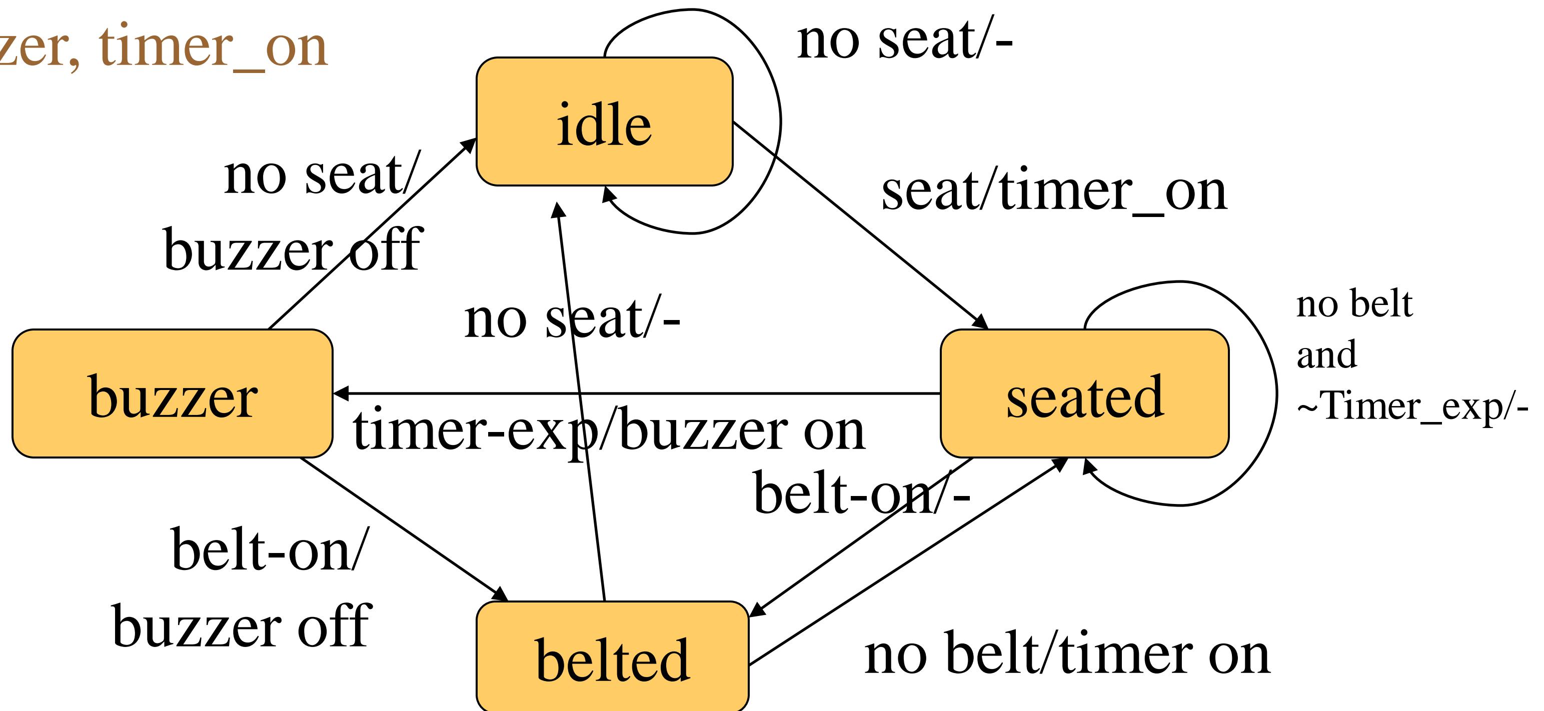
***What if "break" is not written?***

Taken/modified from "Computers as Components, W. Wolf"

# Seat-belt Controller

***3 inputs:*** seat sensor, belt sensor, timer _expired

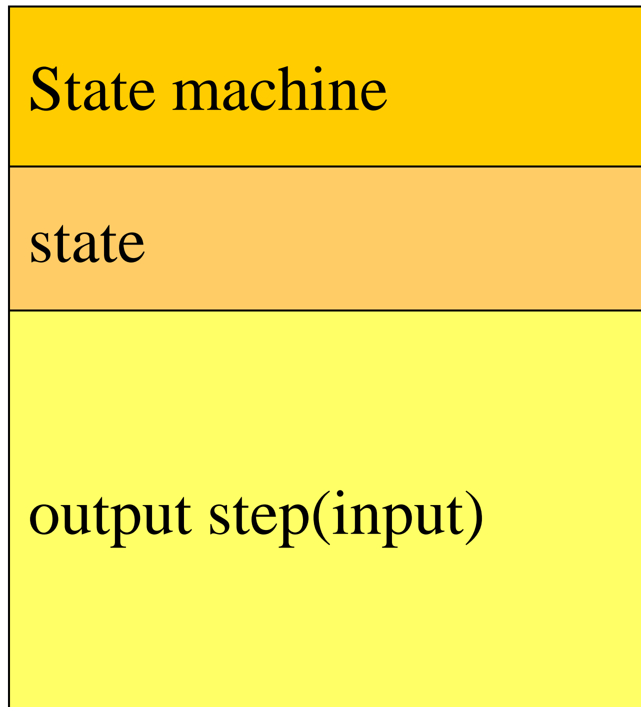
***2 out:*** buzzer, timer_on




➔ Turn buzzer on if a person sits in a seat and does not fasten the seat belt within a fixed time period, etc

Taken/modified from "Computers as Components, W. Wolf"

# State machine pattern




NOTE: State information is private to the class but available to step()

# C implementation

```
#define IDLE 0
#define SEATED 1
#define BELTED 2
#define BUZZER 3
while(TRUE){
  switch (state) {
      case IDLE: if (seat) { state = SEATED; timer_on =
               TRUE; }
               break;
      case SEATED: if (belt) state = BELTED;
               else if (time_off) state = BUZZER;
               break;
      …
  } }
```

Taken/modified from "Computers as Components, W. Wolf"

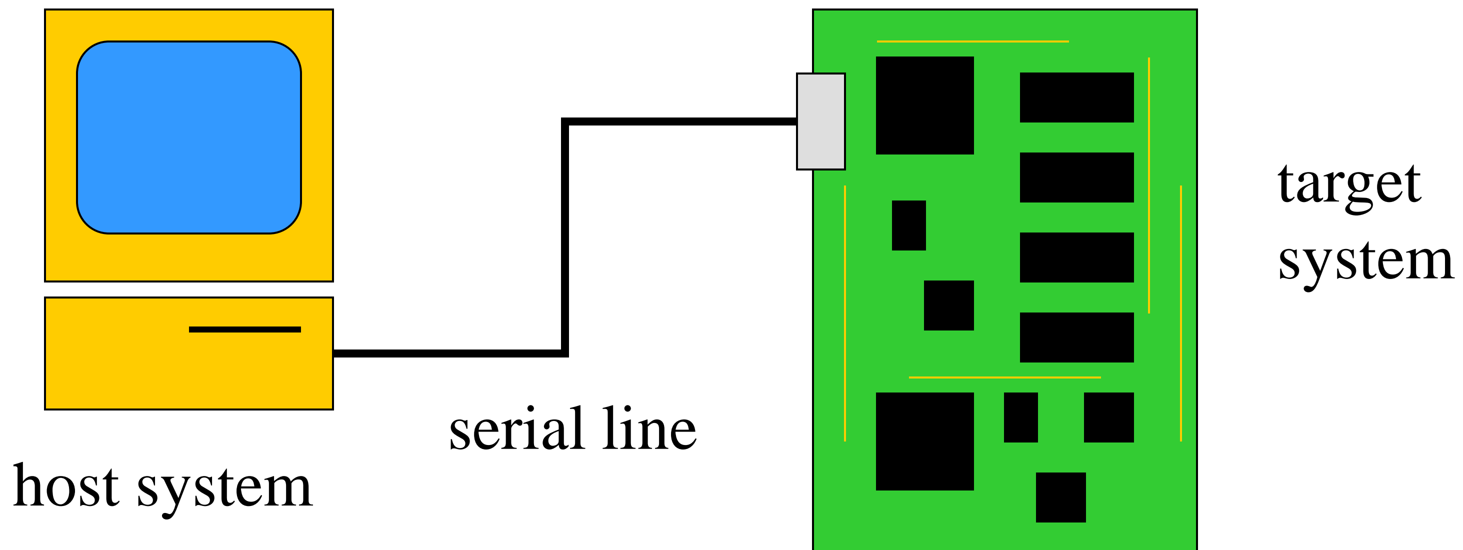
# Embedded software development

- Want to develop as much code as possible on a standard platform:
  - friendlier programming environment;
  - easier debugging.
- Environment should allow testing software elements without the full hardware/software platform → e.g., simulation environment

Taken/modified from "Computers as Components, W. Wolf"

# Host/target system

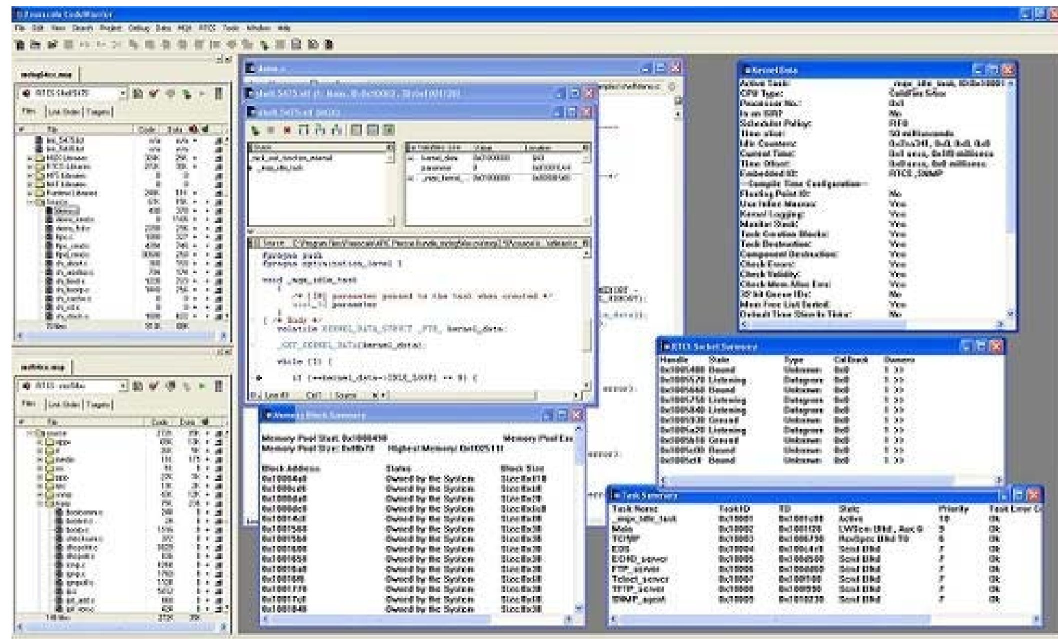
- Use a host system to prepare software for target system:




Taken/modified from "Computers as Components, W. Wolf"

# Host-based tools (e.g., Codewarrior)

- Cross compiler:
  - compiles code on host for target system.
- Cross debugger:
  - displays target state, allows target system to be controlled.



Taken/modified from "Computers as Components, W. Wolf"

# Evaluation boards

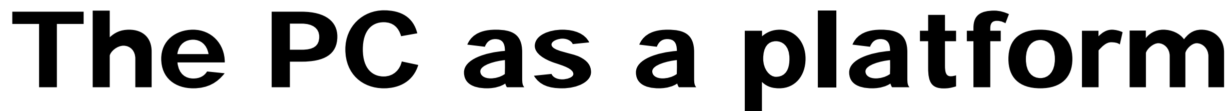
- Designed by CPU manufacturer or others.
- Includes CPU, memory, some I/O devices.
- May include prototyping section.
- Evaluation boards can be used as starting point for your custom board design.

Taken/modified from "Computers as Components, W. Wolf"

# Adding logic to a board: Glue logic

- Programmable logic devices (PLDs) provide low/medium density logic.
- Field-programmable gate arrays (FPGAs) provide more logic and multi-level logic.
- Application-specific integrated circuits (ASICs) are manufactured for a single purpose.

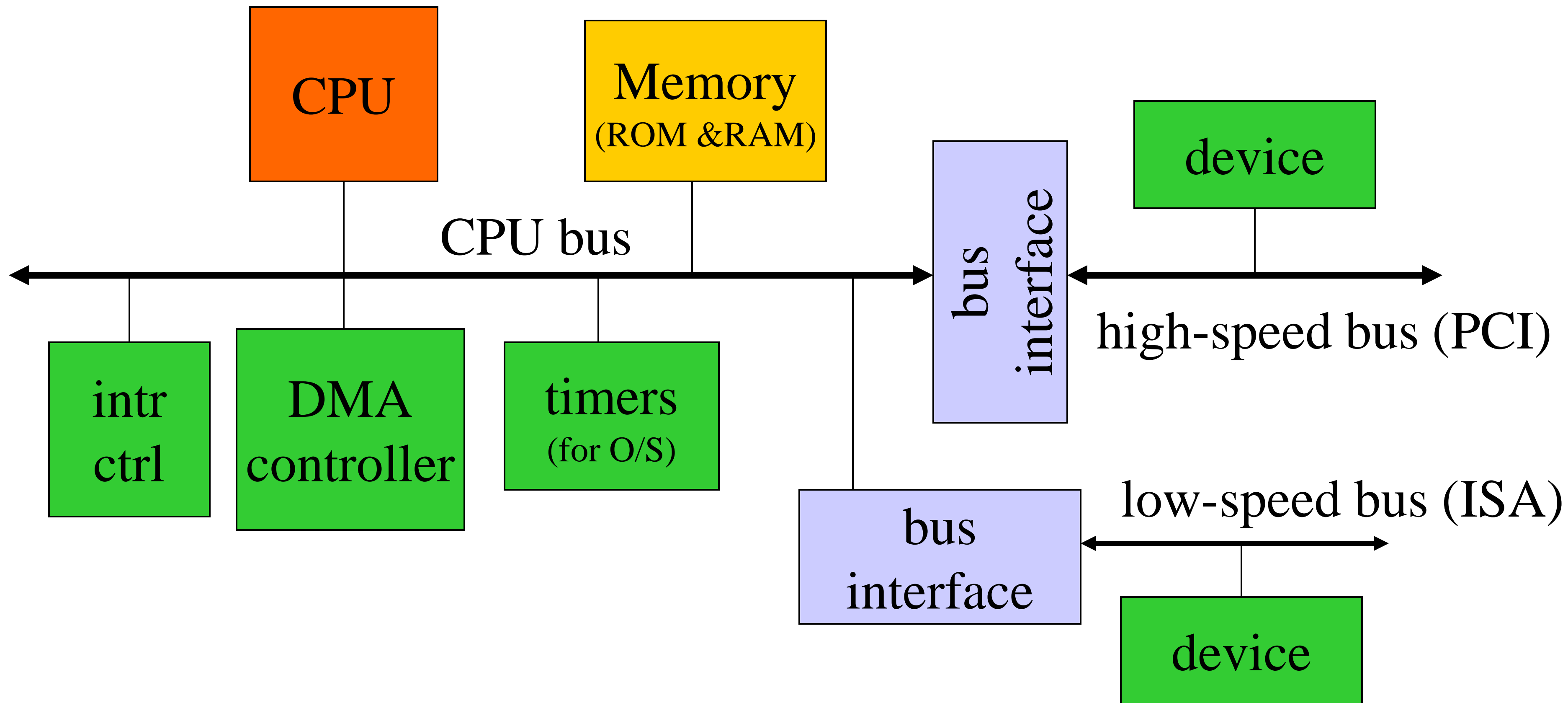
# The PC as a platform



- Advantages:
  - cheap and easy to get;
  - rich and familiar software environment.
- Disadvantages:
  - requires a lot of hardware resources;
  - not well-adapted to real-time (e.g., if non-real-time O/S's are used)

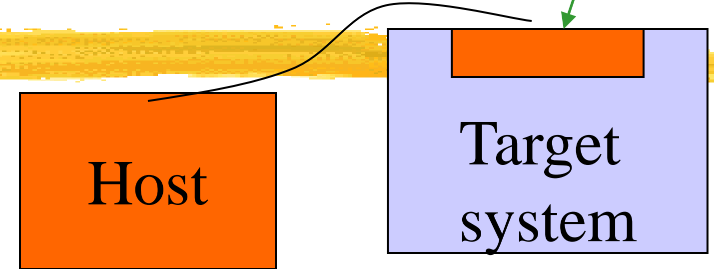
Taken/modified from "Computers as Components, W. Wolf"

# Typical PC hardware platform




Taken/modified from "Computers as Components, W. Wolf"

# Debugging embedded systems




- Challenges:
  - target system may be hard to observe;
  - target may be hard to control;
  - may be hard to generate realistic inputs;
  - setup sequence may be complex.

- Host-Target configurations: Host loads programs into target, start/stop execution, examine memory & CPU registers

# Software debuggers

- A monitor program residing on the target provides basic debugger functions.
- Debugger should have a minimal footprint in memory.
- User program must be careful not to destroy debugger program, but , should be able to recover from some damage caused by user code.

# Breakpoints: An important debugging tool

- A breakpoint allows the user to stop execution, examine system state, and change state.

- Replace the breakpointed instruction with a subroutine call to the monitor program.

Taken/modified from "Computers as Components, W. Wolf"

# ARM breakpoints: Replace the instruction at bkpt with a subroutine call to the monitor program

```
0x400  MUL r4,r6,r6
0x404  ADD r2,r2,r4
0x408  ADD r0,r0,#1
0x40c  B loop
```

uninstrumented code

```
0x400  MUL r4,r6,r6
0x404  ADD r2,r2,r4
0x408  ADD r0,r0,#1
0x40c  BL bkpoint
```

code with breakpoint

Breakpoint handling routine

- Saves all registers.
- Can display CPU state to the user.

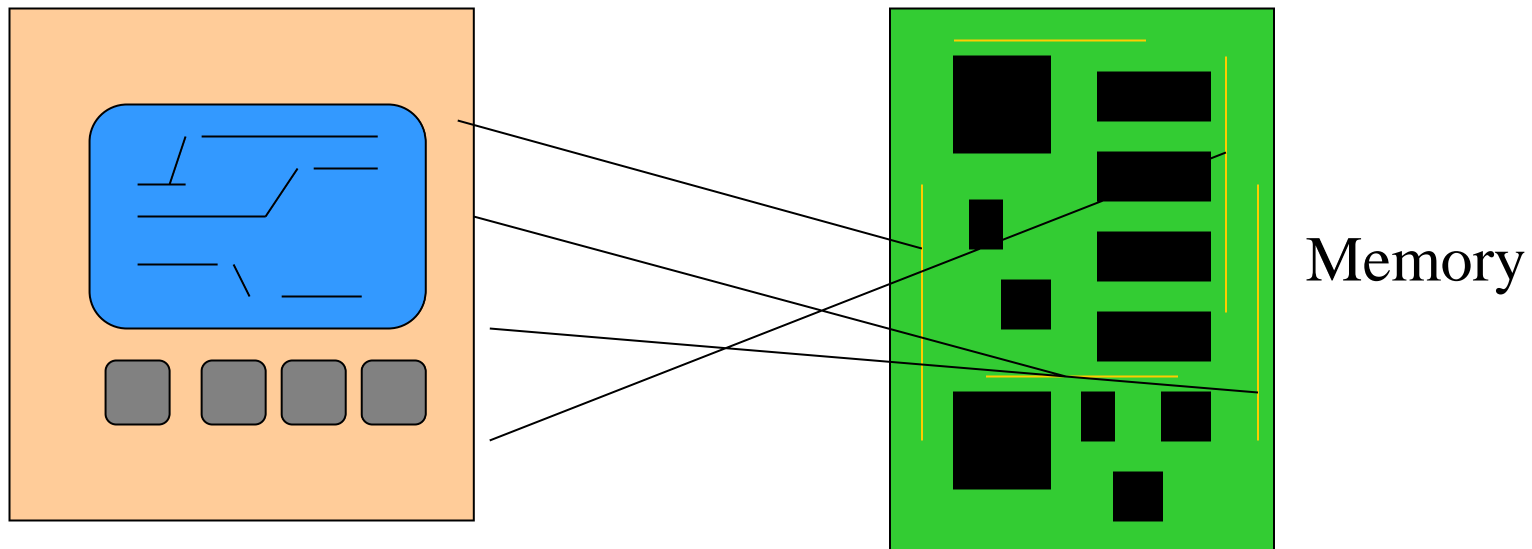
Taken/modified from "Computers as Components, W. Wolf"

# Breakpoint handler actions

- Save registers.
- Allow user to examine machine.
- Before returning, restore system state.
  - Safest way to execute the instruction is to replace it and execute in place.

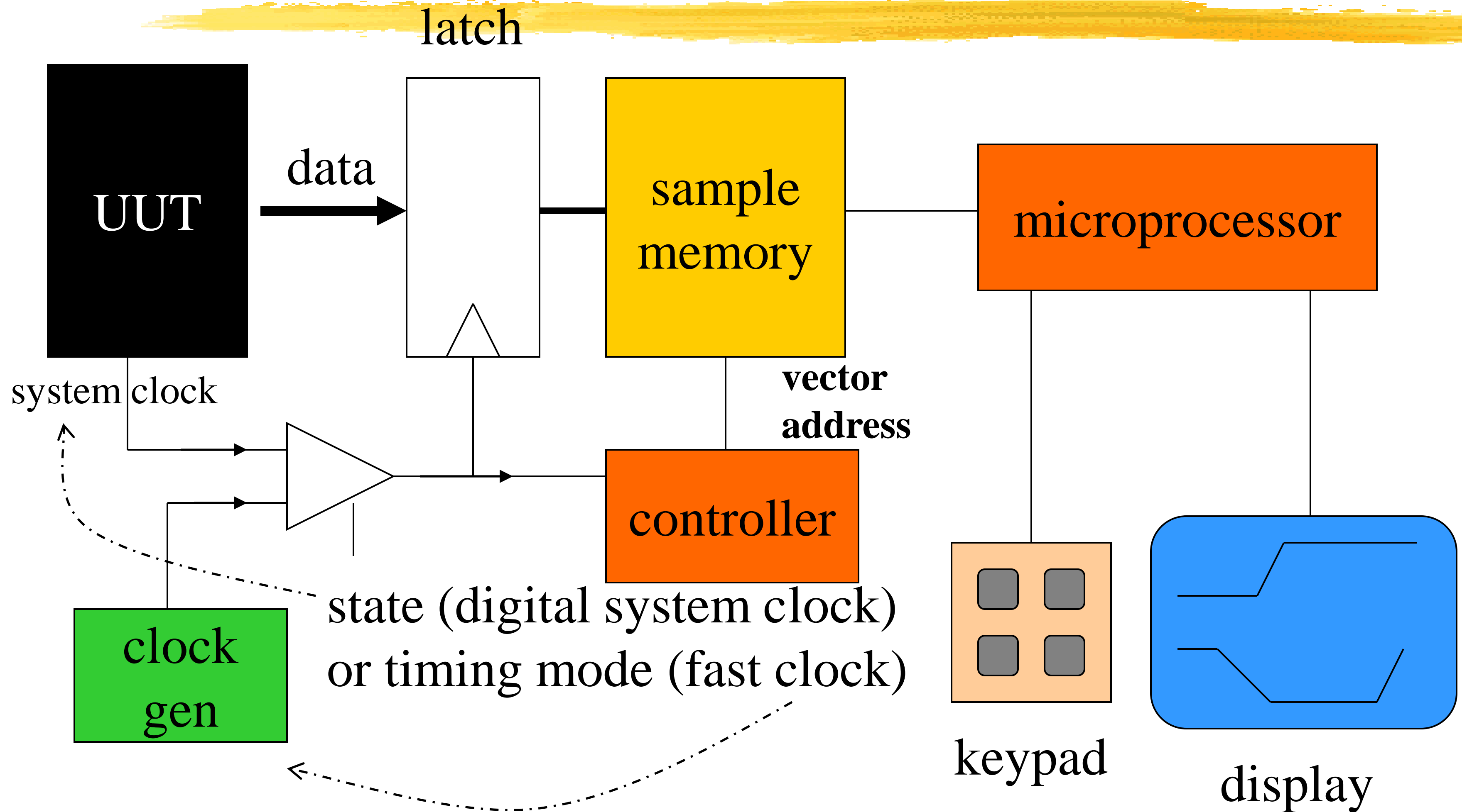
# Logic analyzers

- A logic analyzer is an array of low-grade oscilloscopes → Records values of signals in an internal memory and displays them once the memory is full.




Taken/modified from "Computers as Components, W. Wolf"

# Logic analyzer architecture

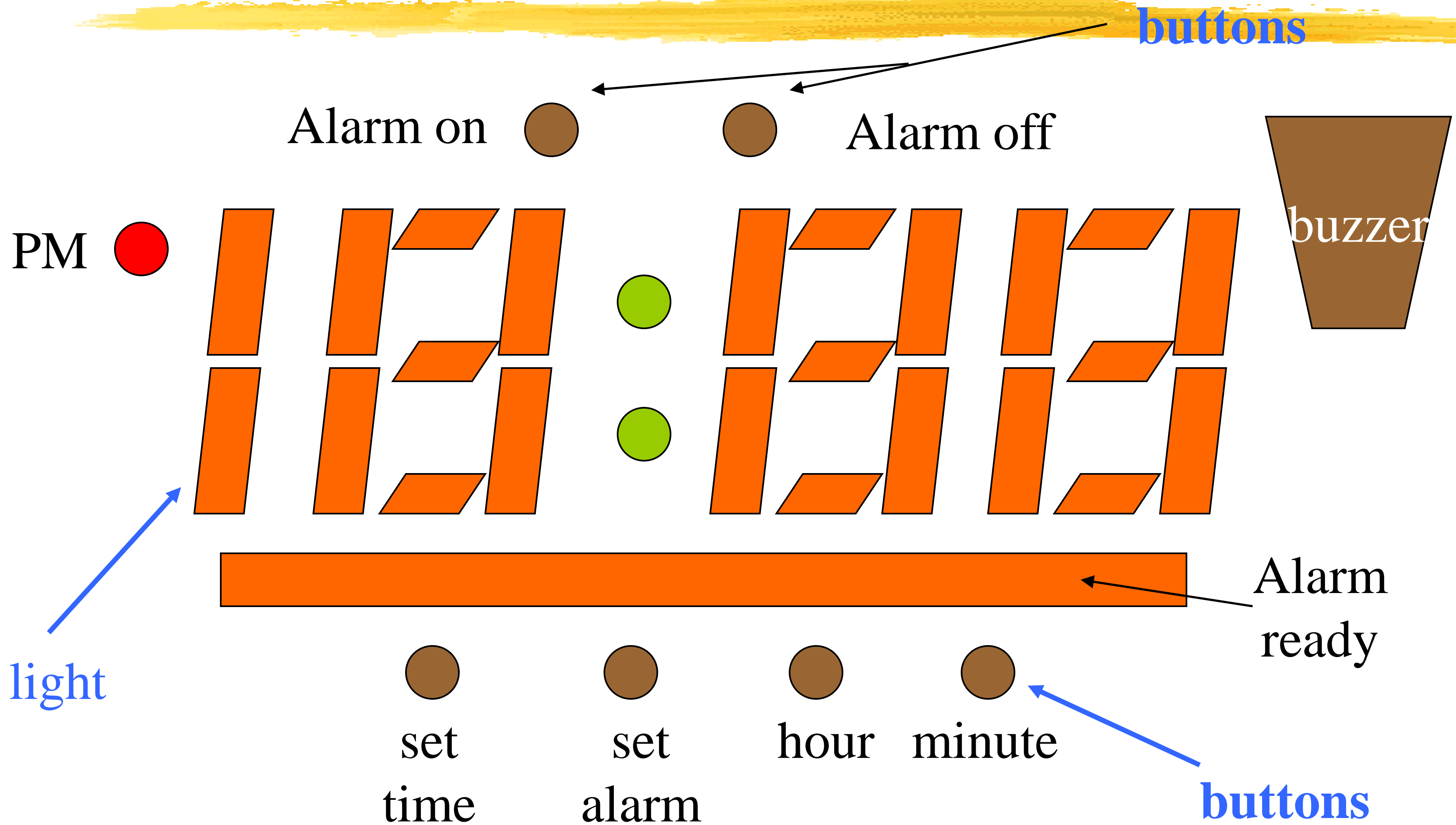



Taken/modified from "Computers as Components, W. Wolf"

# The embedded systems platform

- Example: alarm clock

# Alarm clock interface




Taken/modified from "Computers as Components, W. Wolf"

# Operations

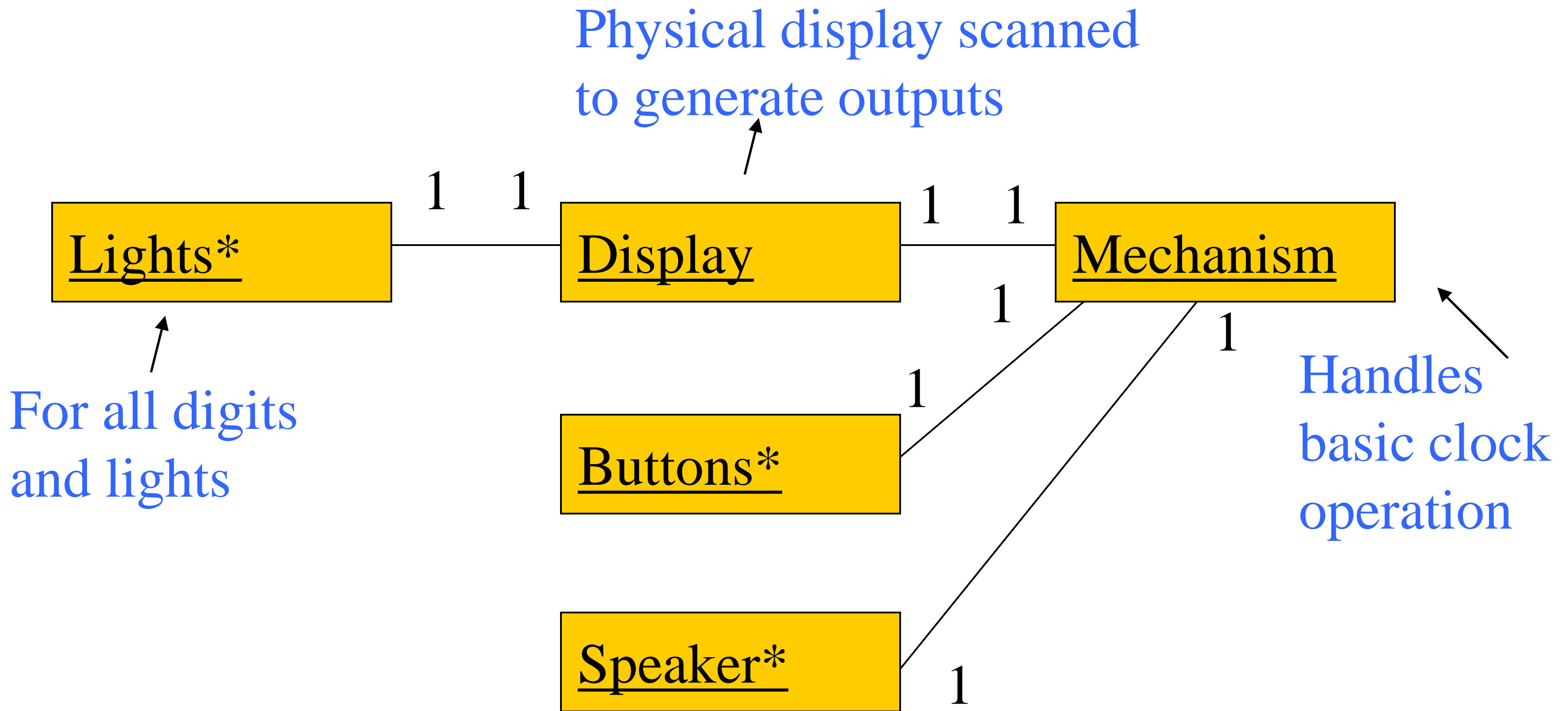
- Set time: hold set time, depress hour, minute.
- Set alarm time: hold set alarm, depress hour, minute.
- Turn alarm on/off: depress alarm on/off.

# Alarm clock requirements

| | |
|---|---|
| name | alarm clock |
| purpose | 12-hour digital clock with one alarm |
| inputs | Push buttons: set time, set alarm, hour, minute, alarm on/off |
| outputs | four-digit display, PM indicator, alarm ready, buzzer |
| functions | keep time, set time, set alarm, turn alarm on/off, activate buzzer by alarm, PM light on after 12 |
| performance | hours and minutes, no seconds; not high precision |
| Manufacturing cost | consumer product |
| power | AC |
| physical size/weight | fits on stand |

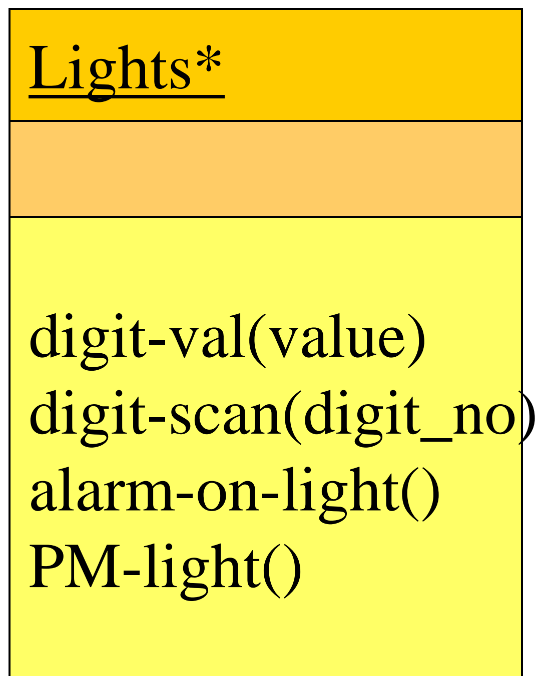
Taken/modified from "Computers as Components, W. Wolf"

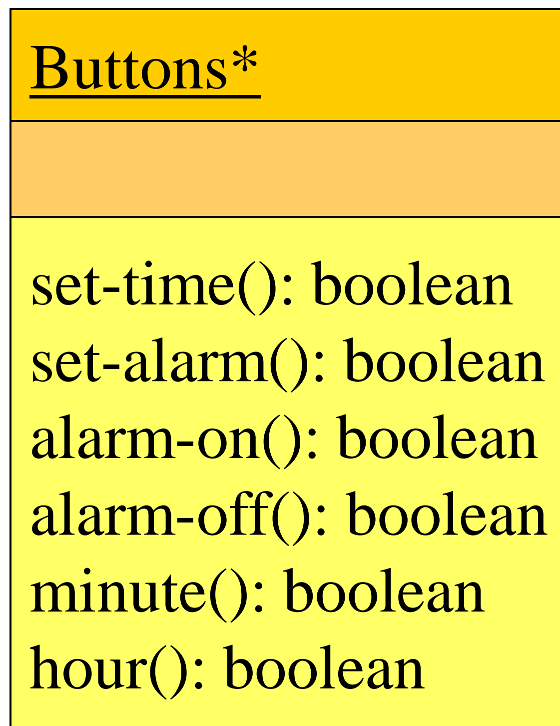
# Alarm clock class diagram

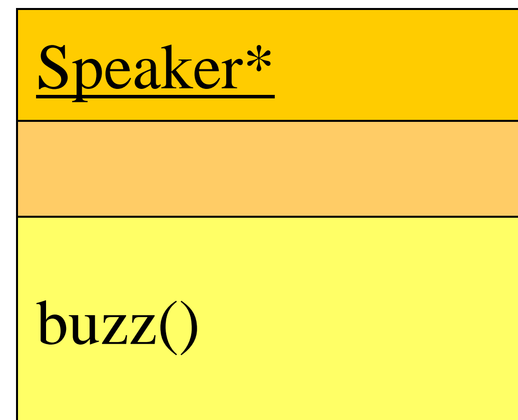



Taken/modified from "Computers as Components, W. Wolf"

# Alarm clock physical classes




To drive the lights

Read-only access to state of buttons

Taken/modified from "Computers as Components, W. Wolf"

# Display class: Generate display by scanning digits periodically

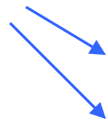
| Display |
| --- |
| time[4]: integer<br>alarm-indicator: boolean<br>PM-indicator: boolean |
| set-time()<br>alarm-light-on()<br>alarm-light-off()<br>PM-light-on()<br>PM-light-off() |

Taken/modified from "Computers as Components, W. Wolf"

# Mechanism class: keeps track of current time, alarm time, etc

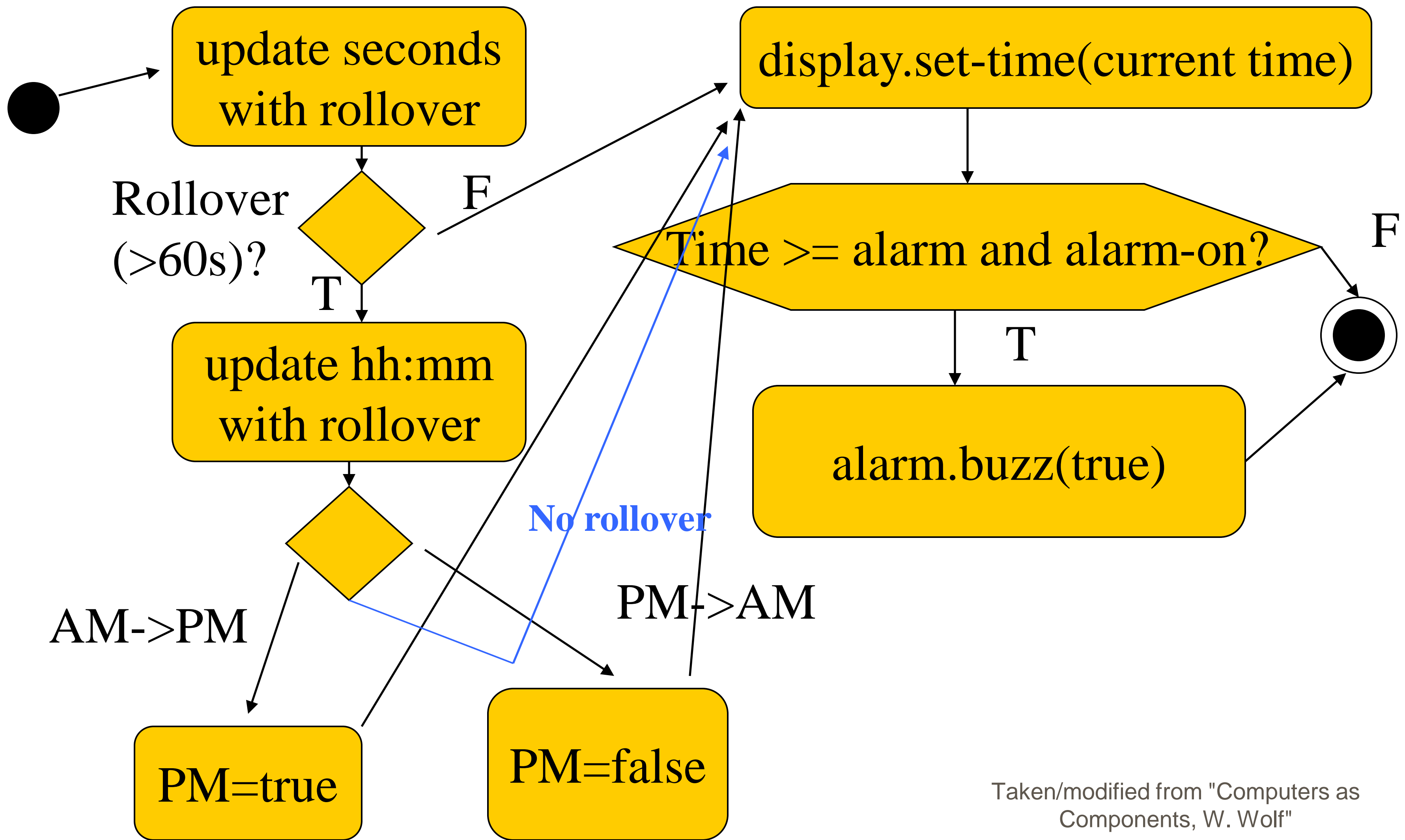
| **<u>Mechanism</u>** |
| --- |
| **Seconds: integer**<br>**PM: boolean**<br>**tens-hours, ones-hours: integer**<br>**tens-minutes, ones-minutes: integer**<br>**alarm-ready: boolean**<br>**alarm-tens-hours, alarm-ones-hours: integer**<br>**alarm-tens-minutes, alarm-ones-minutes: integer** |
| **scan-keyboard()**<br>**update-time()** |

Periodic behaviors



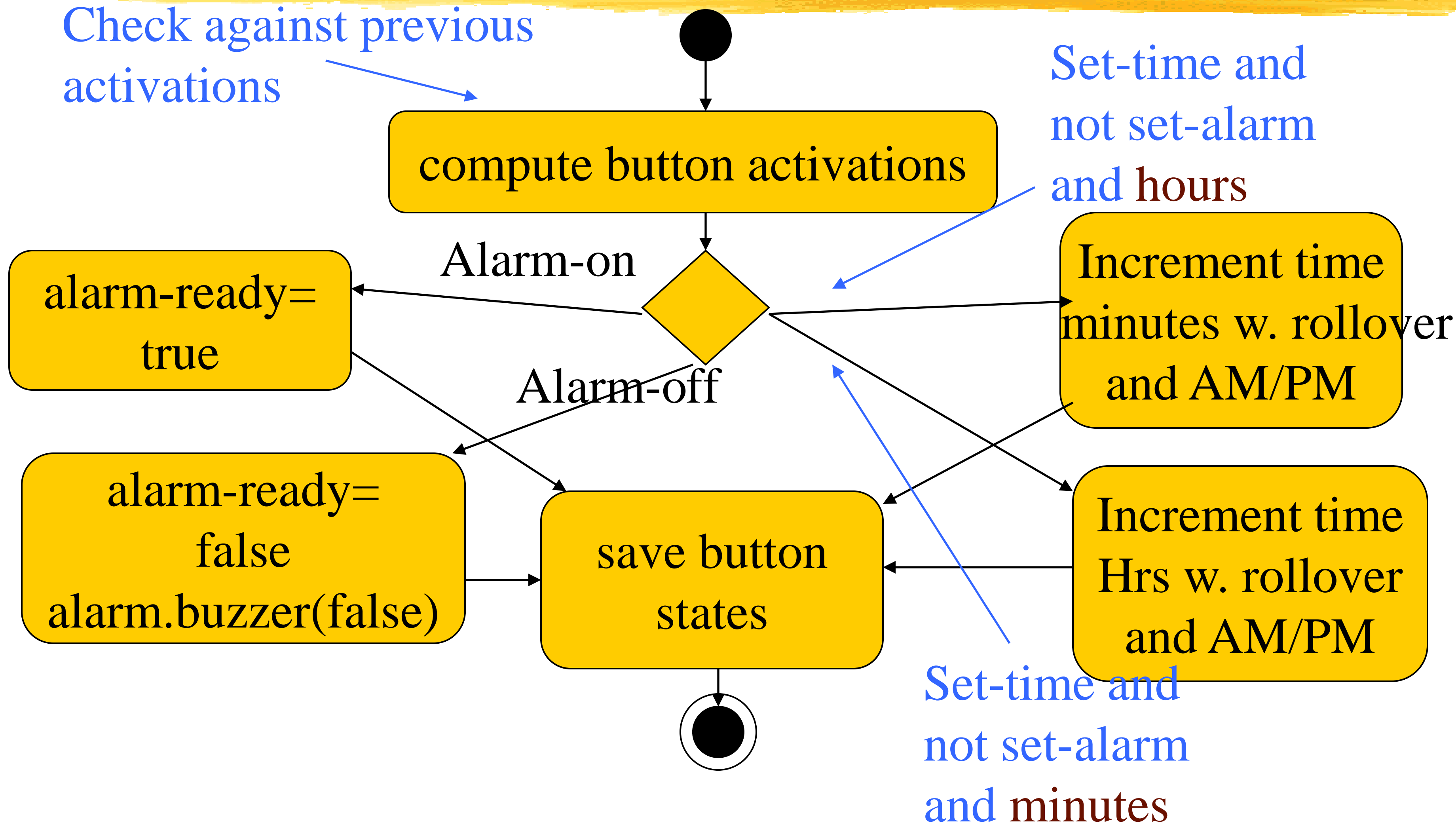
Taken/modified from "Computers as Components, W. Wolf"

# Update-time behavior: once/sec




Taken/modified from "Computers as Components, W. Wolf"

# Scan-keyboard behavior- state diagram




Taken/modified from "Computers as Components, W. Wolf"

# System architecture

- Includes:
  - **periodic behavior (clock);**
  - **aperiodic behavior (buttons, buzzer activation).**
- Two major software components:
  - **interrupt-driven routine updates time→ time stored in a memory variable.**
  - **foreground program deals with buttons, commands → By polling buttons and executing their commands**

Taken/modified from "Computers as Components, W. Wolf"

# Interrupt-driven routine

- Timer probably can't handle one-minute interrupt interval → requires a large number of timer bits- not realistic
- Use software variable to convert interrupt frequency to seconds.

# Foreground program

- Operates as while loop:

```
while (TRUE) {
// read and preprocess button values
      read_buttons(button_values);
      process_command(button_values);
//decide whether to turn alarm on
      check_alarm();
}
```

Taken/modified from "Computers as Components, W. Wolf"

# Testing

- Component testing:
  - **test interrupt code on the platform**
  - **test foreground program**
- System testing:
  - **relatively few components to integrate;**
  - **check clock accuracy;**
  - **check recognition of buttons, buzzer, etc.**