

Chapter 16

Chapter 16 Real-Time Systems Design

Learning Objective

... Designing embedded software systems whose behavior is subject to **timing** constraints.

Frederick T Sheldon
Assistant Professor of Computer Science
Washington State University

Objectives

- To show why real-time systems are usually designed as a set of cooperating concurrent processes
- To show the usefulness of state models in real-time systems design
- To describe the platform support required by real-time systems
- To introduce generic architectures for some types of real-time system

CS-422 Software Engineering Principles
From Software Engineering by J. Sommerville, 1998.

Chapter 16
Slide 2

Topics covered

- Systems design
- State machine modeling
- Real-time executives
- Monitoring and control systems
- Data acquisition systems

CS-422 Software Engineering Principles
From Software Engineering by J. Sommerville, 1998.

Chapter 16
Slide 3

Real-time systems

Systems which monitor and control their environment

Inevitably associated with hardware devices

Sensors: Collect data from the system environment

Actuators: Change (in some way) the system's environment

Time is critical. Real-time systems **MUST** respond within specified times

Definition

A real-time system is a software system where the correct functioning of the system depends on the results produced by the system and the time at which these results are produced

A 'soft' real-time system is a system whose operation is degraded if results are not produced according to the specified timing requirements

A 'hard' real-time system is a system whose operation is incorrect if results are not produced according to the timing specification

Stimulus/Response Systems

Given a stimulus, the system must produce a response within a specified time

Periodic stimuli. Stimuli which occur at predictable time intervals

For example, a temperature sensor may be polled 10 times per second

Aperiodic stimuli. Stimuli which occur at unpredictable times

For example, a system power failure may trigger an interrupt which must be processed by the system

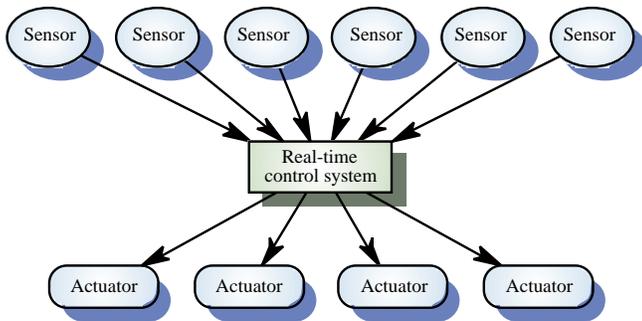
Architectural considerations

Because of the need to respond to timing demands made by different stimuli/responses, the system architecture must allow for fast switching between stimulus handlers

Timing demands of different stimuli are different so a simple sequential loop is not usually adequate

Real-time systems are usually designed as cooperating processes with a real-time executive controlling these processes

A real-time system model



System elements

Sensors control processes

Collect information from sensors. May buffer information collected in response to a sensor stimulus

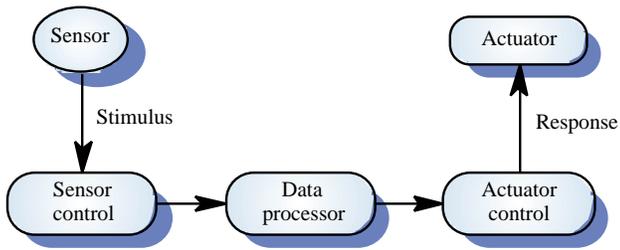
Data processor

Carries out processing of collected information and computes the system response

Actuator control

Generates control signals for the actuator

Sensor/actuator processes



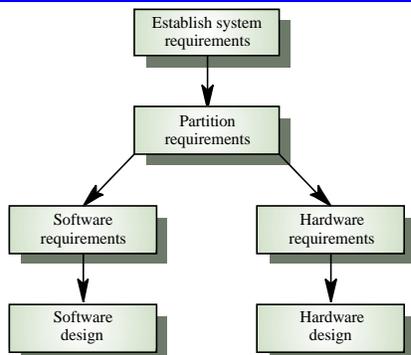
System design

Design both the hardware and the software associated with system. Partition functions to either hardware or software

Design decisions should (normally) be made on the basis of non-functional system requirements

Hardware delivers better performance but potentially longer development and less scope for change

Hardware and software design



R-T systems design process

Identify the stimuli to be processed and the required responses to these stimuli

For each stimulus and response, identify the timing constraints

Aggregate the stimulus and response processing into concurrent processes. A process may be associated with each class of stimulus and response

R-T systems design process

Design algorithms to process each class of stimulus and response. These must meet the given timing requirements

Design a scheduling system which will ensure that processes are started in time to meet their deadlines

Integrate using a real-time executive or operating system

Timing constraints

May require extensive simulation/analysis and experimentation to ensure that these are met by the system

May mean that certain design strategies such as object-oriented design cannot be used because of the additional overhead involved

May mean that low-level programming language features have to be used for performance reasons

State machine modeling

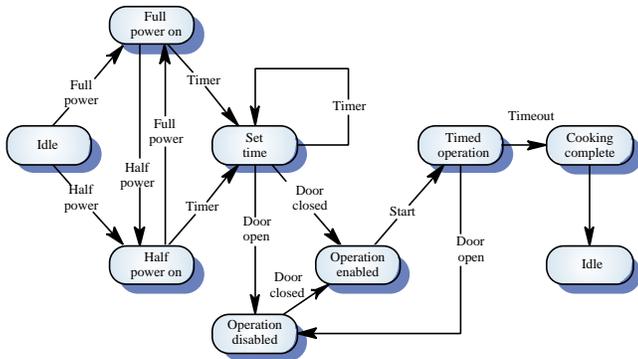
The effect of a stimulus in a real-time system may trigger a transition from one state to another.

Finite state machines can be used for modeling real-time systems.

However, FSM models lack structure. Even simple systems can have a complex model.

Thread diagrams which show an event sequence are a means of managing the complexity in state machine models.

Microwave oven state machine



Microwave oven states

State	Description
Half power on	The oven power is set to 300 watts
Full power on	The oven power is set to 600 watts
Set time	The cooking time is set to the user's input value
Operation disabled	Oven operation is disabled for safety. Interior oven light is on
Operation enabled	Oven operation is enabled. Interior oven light is off
Timed operation	Oven in operation. Interior oven light is on. Timer is counting down.
Cooking complete	Timer has counted down to zero. Audible alarm signal is on. Oven light is off.

Microwave oven stimuli

Stimulus	Description
Half power	The user has pressed the half power button
Full power	The user has pressed the full power button
Timer	The user has pressed one of the timer buttons
Door open	The oven door switch is not closed
Door closed	The oven door switch is closed
Start	The user has pressed the start button
Timeout	Timer signal indicating that set cooking time is finished

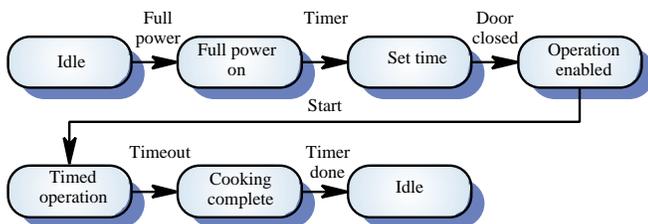
Thread diagrams

Used to structure and present state model information

Show the end-to-end processing of a specific set of stimuli

Thread diagrams should be produced for message combinations. This means for systems with multiple stimuli, this approach may be impractical

Thread diagram - full power



Real-time executives

Real-time executives are specialized operating systems which manage the processes in the RTS
Responsible for process management and resource (processor and memory) allocation
May be based on a standard RTE kernel which is used unchanged or modified for a particular application
Does not include facilities such as file management

Executive components

Real-time clock
Provides information for process scheduling.

Interrupt handler
Manages aperiodic requests for service.

Scheduler
Chooses the next process to be run.

Resource manager
Allocates memory and processor resources.

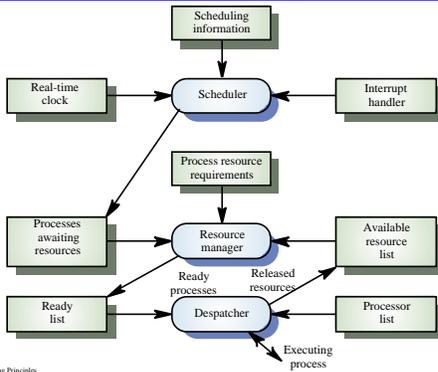
Dispatcher
Starts process execution.

Non-stop system components

Configuration manager
Responsible for the dynamic reconfiguration of the system software and hardware. Hardware modules may be replaced and software upgraded without stopping the systems

Fault manager
Responsible for detecting software and hardware faults and taking appropriate actions (e.g. switching to backup disks) to ensure that the system continues in operation

Real-time executive components



Process priority

The processing of some types of stimuli must sometimes take priority

Interrupt level priority. Highest priority which is allocated to processes requiring a very fast response

Clock level priority. Allocated to periodic processes

Within these, further levels of priority may be assigned

Interrupt servicing

Control is transferred automatically to a pre-determined memory location

This location contains an instruction to jump to an interrupt service routine

Further interrupts are disabled, the interrupt serviced and then control is returned to the interrupted process

Interrupt service routines MUST be short, simple and fast Why?

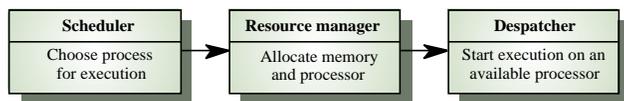
Periodic process servicing

In most real-time systems, there will be several classes of periodic process, each with different periods (the time between executions), execution times and deadlines (the time by which processing must be completed)

The real-time clock ticks periodically and each tick causes an interrupt which schedules the process manager for periodic processes

The process manager selects a process which is ready for execution

RTE process management



Process switching

The scheduler chooses the next process to be executed by the processor. This depends on a scheduling strategy which may take the process priority into account

The resource manager allocates memory and a processor for the process to be executed

Dispatcher takes process from ready list, loads it onto a processor and starts execution

Monitoring and control systems

Important class of real-time systems
Continuously check sensors and take actions depending on sensor values
Monitoring systems examine sensors and report their results
Control systems take sensor values and control hardware actuators

Intruder alarm system

System is required to monitor sensors on doors and windows to detect the presence of intruders in a building
When a sensor indicates a break-in, system switches on lights around the area and calls police automatically
Includes provision for operation without a main (or primary) power supply

Intruder alarm system

Sensors
Movement detectors, window sensors, door sensors.
50 window sensors, 30 door sensors and 200 movement detectors.

Actions
When an intruder is detected, police are called automatically.
Lights are switched on in rooms with active sensors.
An audible alarm is switched on.

The system switches automatically to backup power when a voltage drop is detected.

The R-T system design process

- Identify stimuli and associated responses
- Define the timing constraints associated with each stimulus and response
- Allocate system functions to concurrent processes
- Design algorithms for stimulus processing and response generation
- Design a scheduling system which ensures that processes will always be scheduled to meet their deadlines

Stimuli to be processed

Power failure

Generated aperiodically by a circuit monitor. When received, the system must switch to backup power within 50 ms

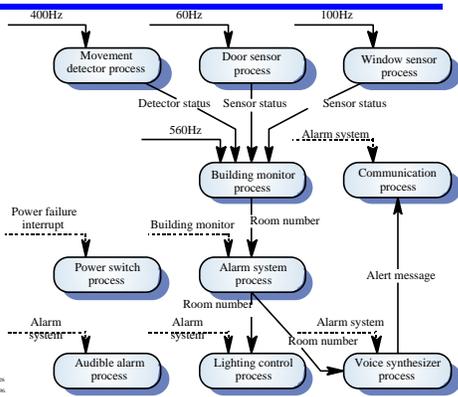
Intruder alarm

Stimulus generated by system sensors. Response is to call the police, switch on building lights and the audible alarm

Timing requirements

Stimulus/Response	Timing requirements
Power fail interrupt	The switch to backup power must be completed within a deadline of 50 ms.
Door alarm	Each door alarm should be polled twice per second.
Window alarm	Each window alarm should be polled twice per second.
Movement detector	Each movement detector should be polled twice per second.
Audible alarm	The audible alarm should be switched on within 1/2 second of an alarm being raised by a sensor.
Lights switch	The lights should be switched on within 1/2 second of an alarm being raised by a sensor.
Communications	The call to the police should be started within 2 seconds of an alarm being raised by a sensor.
Voice synthesiser	A synthesised message should be available within 4 seconds of an alarm being raised by a sensor

Process architecture



Building_monitor process #1

```

task Building_monitor is
  entry Initialize ;
  entry Test ;
  entry Monitor ;
end Building_monitor ;

task body Building_monitor is
  type ROOMS is array (NATURAL range <->) of ROOM_NUMBER ;
  Move_sensor, Window_sensor, Door_sensor : SENSOR ;
  Move_sensor_locations: ROOMS (0..Number_of_move_sensors-1) ;
  Window_sensor_locations: ROOMS (0..Number_of_window_sensors-1) ;
  Corridor_sensor_locations : ROOMS (0..Number_of_corridor_sensors-1) ;
  Next_movement_sensor, Next_window_sensor,
  Next_door_sensor: NATURAL := 0;
begin
  select
  accept Initialize do
    -- code here to read sensor locations from a file and
    -- initialize all location arrays
  end Initialize ;
  or
  accept Test do
    -- code here to activate a sensor test routine
  end Test ;
  or
  accept Monitor do
    -- the main processing loop
  
```

Building_monitor process #2

```

accept Monitor do
  -- the main processing loop
  loop
    -- TIMING: Each movement sensor twice/second
    Next_move_sensor :=
      Next_move_sensor + 1 rem Number_of_move_sensors ;
    -- rendezvous with Movement detector process
    Movement_detector.Interrogate (Move_sensor) ;
    if Move_sensor /= OK then
      Alarm_system.Initiate
      (Move_sensor_locations (Next_move_sensor)) ;
    end if ;
    -- TIMING: Each window sensor twice/second
    -- rendezvous with Window sensor process
    Next_window_sensor :=
      Next_window_sensor + 1 rem Number_of_window_sensors ;
    Window_sensor.Interrogate (Window_sensor) ;
    if Window_sensor /= OK then
      Alarm_system.Initiate (Window_sensor_locations
      (Next_move_sensor)) ;
    end if ;
    -- TIMING: Each door sensor twice/second
    -- rendezvous with Door sensor process
    -- Comparable code to the above here
  end loop ;
end select ;
end Building_monitor ;

```

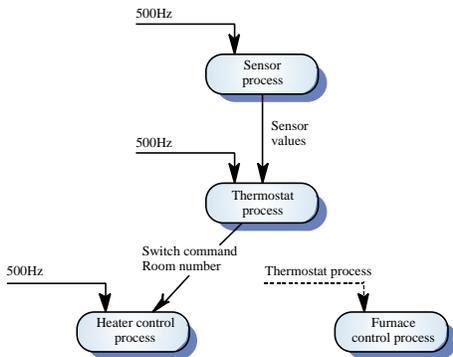
Control systems

Burglar alarm system is primarily a monitoring system. Collects data from sensors but no real-time actuator control

Control systems are similar because, in response to sensor values, system sends control signals to actuators

Example of a monitoring and control system is a system which monitors temperature and switches heaters on and off

A temperature control system



Data acquisition systems

Collect data from sensors for subsequent processing and analysis.

Data collection processes and processing processes may have different periods and deadlines.

Data collection may be faster than processing e.g. collecting information about an explosion.

Circular or ring buffers are a mechanism for smoothing speed differences.

Reactor data collection

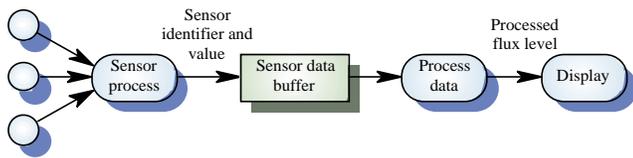
System collects data from a set of sensors monitoring the neutron flux from a nuclear reactor.

Flux data is placed in a ring buffer for later processing.

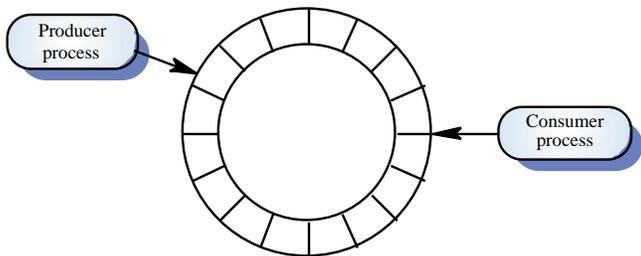
The ring buffer is itself implemented as a concurrent process so that the collection and processing processes may be synchronized.

Reactor flux monitoring

Sensors (each data flow is a sensor value)



A ring buffer



Mutual exclusion

Producer processes collect data and add it to the buffer. Consumer processes take data from the buffer and make elements available

Producer and consumer processes must be mutually excluded from accessing the same element.

The buffer must stop producer processes adding information to a full buffer and consumer processes trying to take information from an empty buffer.

Sensor data buffer - task specification

task Sensor_data_buffer **is**

-- Get and Put are like procedures which may execute in parallel.

-- They are the only interface to the buffer.

entry Put (Val: SENSOR_RECORD);

entry Get (Val: in out SENSOR_RECORD);

end Sensor_data_buffer ;

Sensor data buffer

task body Sensor_data_buffer **is**

Size: **constant** NATURAL := 50000 ;

type BUFSIZE **is range** 0..Size ;

Store: **array** (BUFSIZE) of SENSOR_RECORD ;

Entries: NATURAL := 0 ;

Front, Back: BUFSIZE := 1 ;

begin

loop

-- A call to the Get and Put operations will only be accepted when

-- one of the expressions associated with when is true.

-- Thus, Put can only execute when there is space in the buffer,

-- Get can only take items from the buffer when it is not empty.

select

when Entries < Size =>

accept Put (Val: SENSOR_RECORD) **do**

Store (Back) := Val ;

end Put ;

Back := Back rem BUFSIZE'LAST + 1 ;

Entries := Entries + 1 ;

or

when Entries > 0 =>

accept Get (Val: in out SENSOR_RECORD) **do**

Val := Store (Front) ;

end Get ;

Front := Front rem BUFSIZE'LAST + 1 ;

Entries := Entries - 1 ;

end select ;

end loop ;

end Sensor_data_buffer ;

Key points

Real-time system correctness depends not just on what the system does but also on how fast it reacts

Delay partitioning of functions to hardware and software until as late as possible in the design process

Real-time systems are usually designed as a number of concurrent processes

Key points

A R-T system model may associate processes with each class of sensor and actuator

Real-time executives are responsible for process and resource management

Monitoring and control systems poll sensors and send control signal to actuators

Data acquisition systems are usually organized according to a producer consumer model
