

Real-time Systems

- Systems controlled by embedded software whose behaviour is subject to timing constraints

Real-time systems

A real-time system is a 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. Generally, real-time systems monitor and control their environment

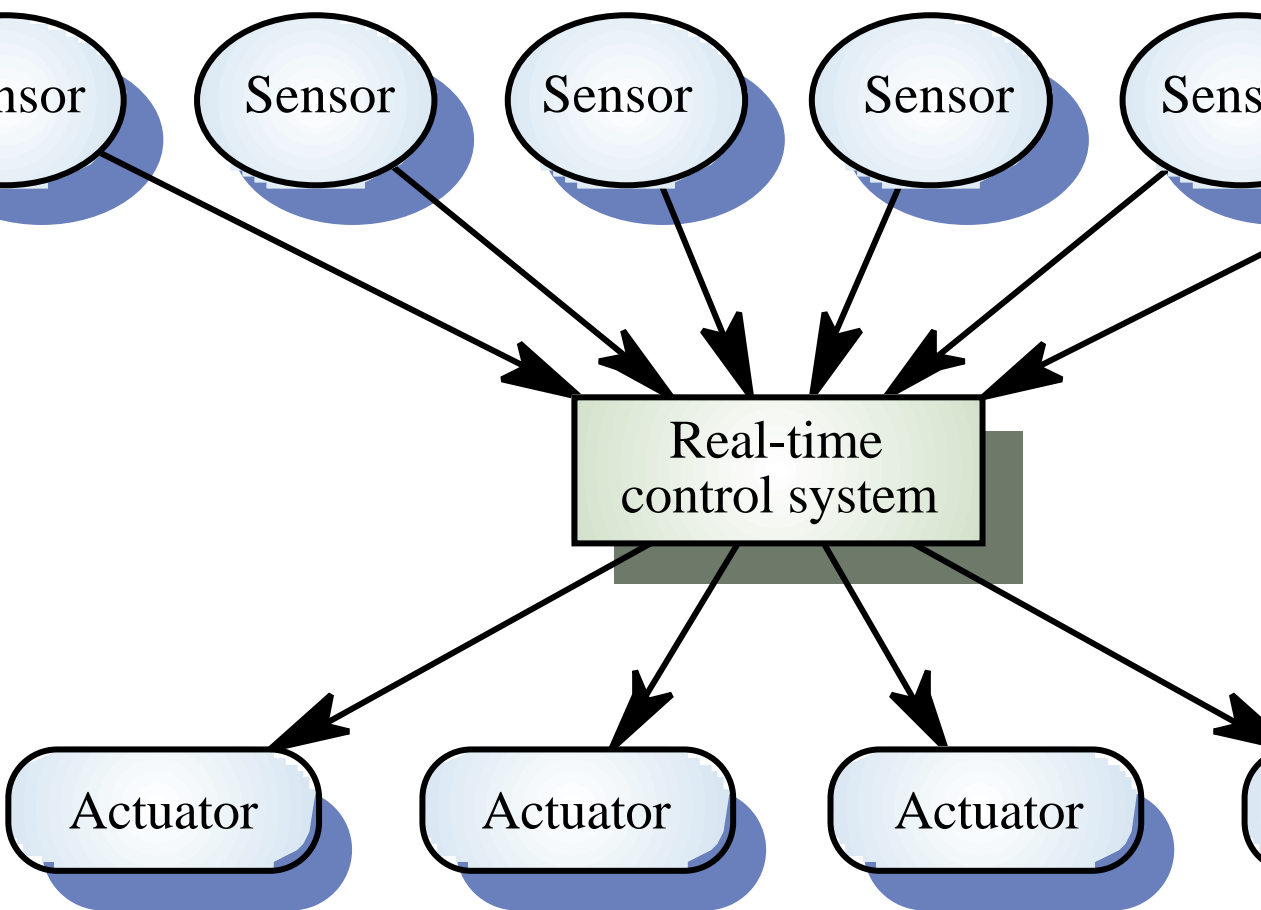
A ‘soft’ real-time system is a system whose operation is degraded if results are not produced according to the specified timing requirements

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

Examples of real-time systems

- Airbag control system in a car
- Speed detector system on a motorway
- Controller for a video recorder
- Fire alarm system
- Fuel control system in an aircraft engine
- Telephone exchange system
- Bank auto-teller machine

A real-time system model



sensors and actuators

- Sensors are sub-systems which provide information about the systems environment
 - Temperature sensor
 - Timer
 - Flow rate sensor
 - Cardiac monitor
- Actuators are sub-systems which affect the systems environment
 - Heater/refrigerator
 - Alarm
 - Motor

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

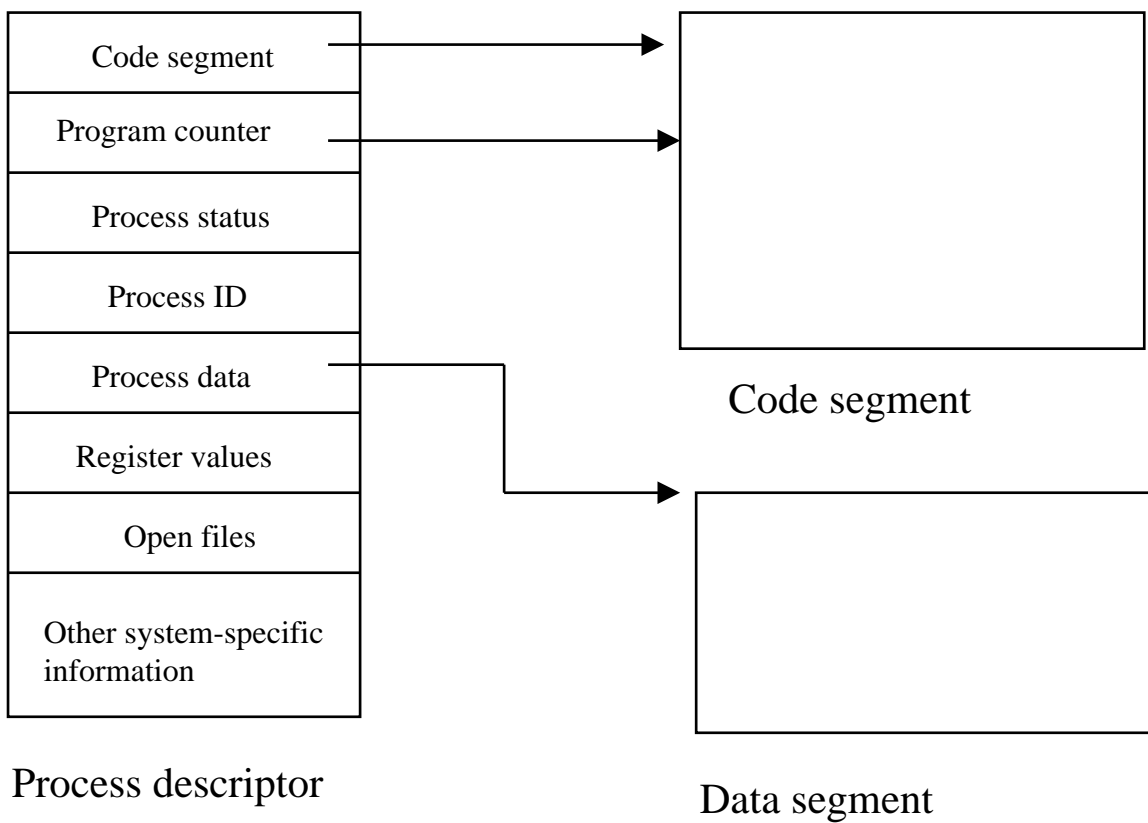
Architectural considerations

- Because of the need to respond to timing demands made by different stimuli/responses, the architecture of a real-time system 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 and implemented as a set of cooperating processes with a real-time executive controlling these processes
- Processes run in parallel in real-time systems

Processes

- Processes are executing programs (or program components)
- They consist of a code segment, possibly one or more data segments plus status information
- The status information allows the processes to be started and stopped by some kind of process control system

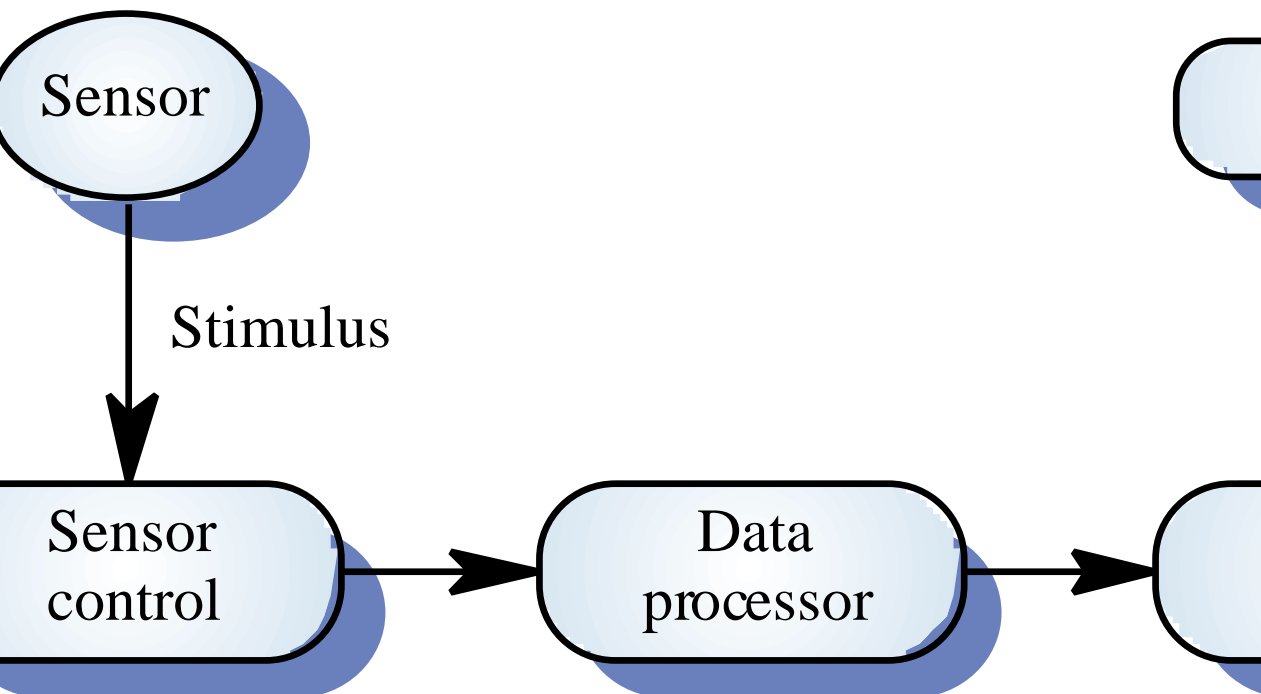
Process descriptors



system elements

- **Sensor control processes**
 - Collect information from sensors. May buffer information collected in response to a sensor stimulus
- **Data processors**
 - Carries out processing of collected information and computes the system response
- **Actuator control processes**
 - Generates control signals for the actuator

Sensor/actuator processes



-T systems design process

- Identify system type
- Identify the stimuli to be processed and the required responses to these stimuli
- For each stimulus and response, identify the timing constraints
- Define the process architecture by aggregating the stimulus and response processing into concurrent processes. A process may be associated with each class of stimulus and response.

-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 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

Types of real-time systems

- **Monitoring systems**
 - Systems which monitor a set of sensors and report on their status. May take alarm actions when problems arise
- **Monitoring and control systems**
 - Systems which monitor a set of sensors and send signals to actuators depending on sensor inputs
- **Data acquisition systems**
 - Systems which collect data from ‘producers’ and deliver data to ‘consumers’

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

- The 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
- Provision for operation without a mains power supply must be supported

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*
- *Define process architecture using 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

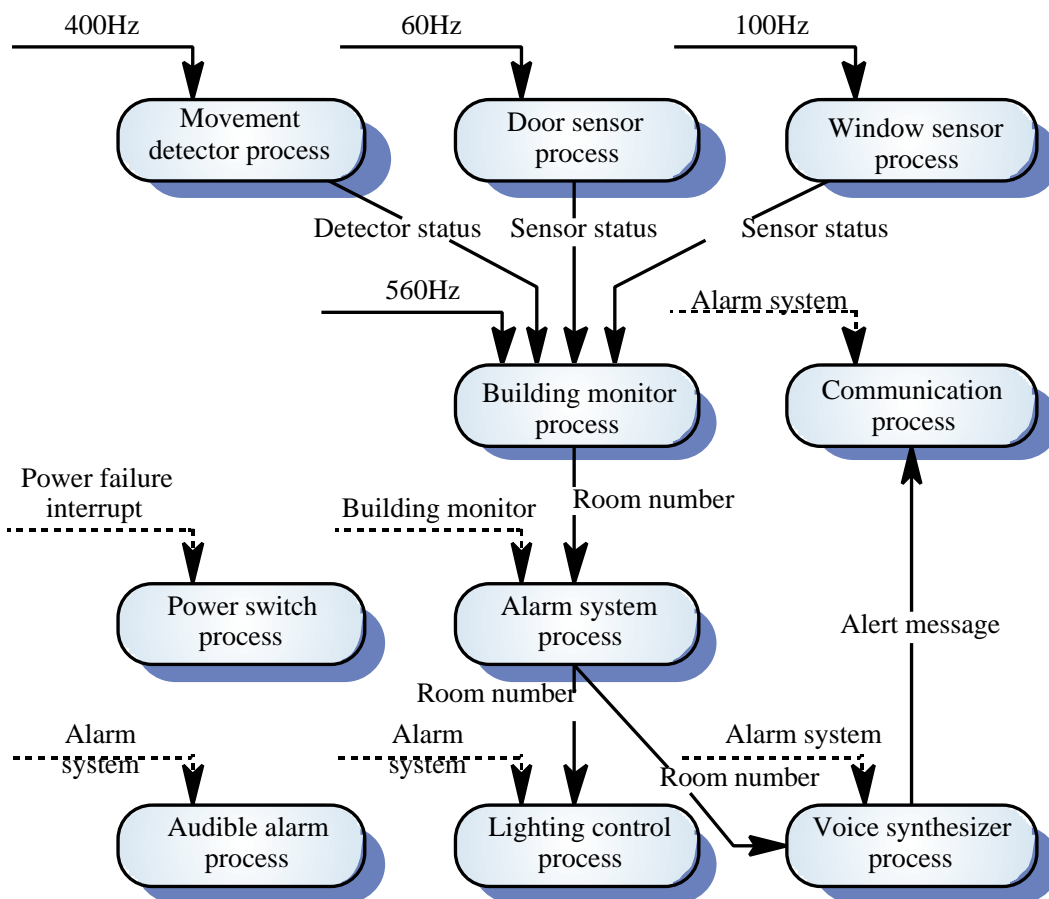
Identify stimuli and responses

- 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
 - A stimulus may be received from each of the different types of sensor that is associated with the system

Define timing constraints

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

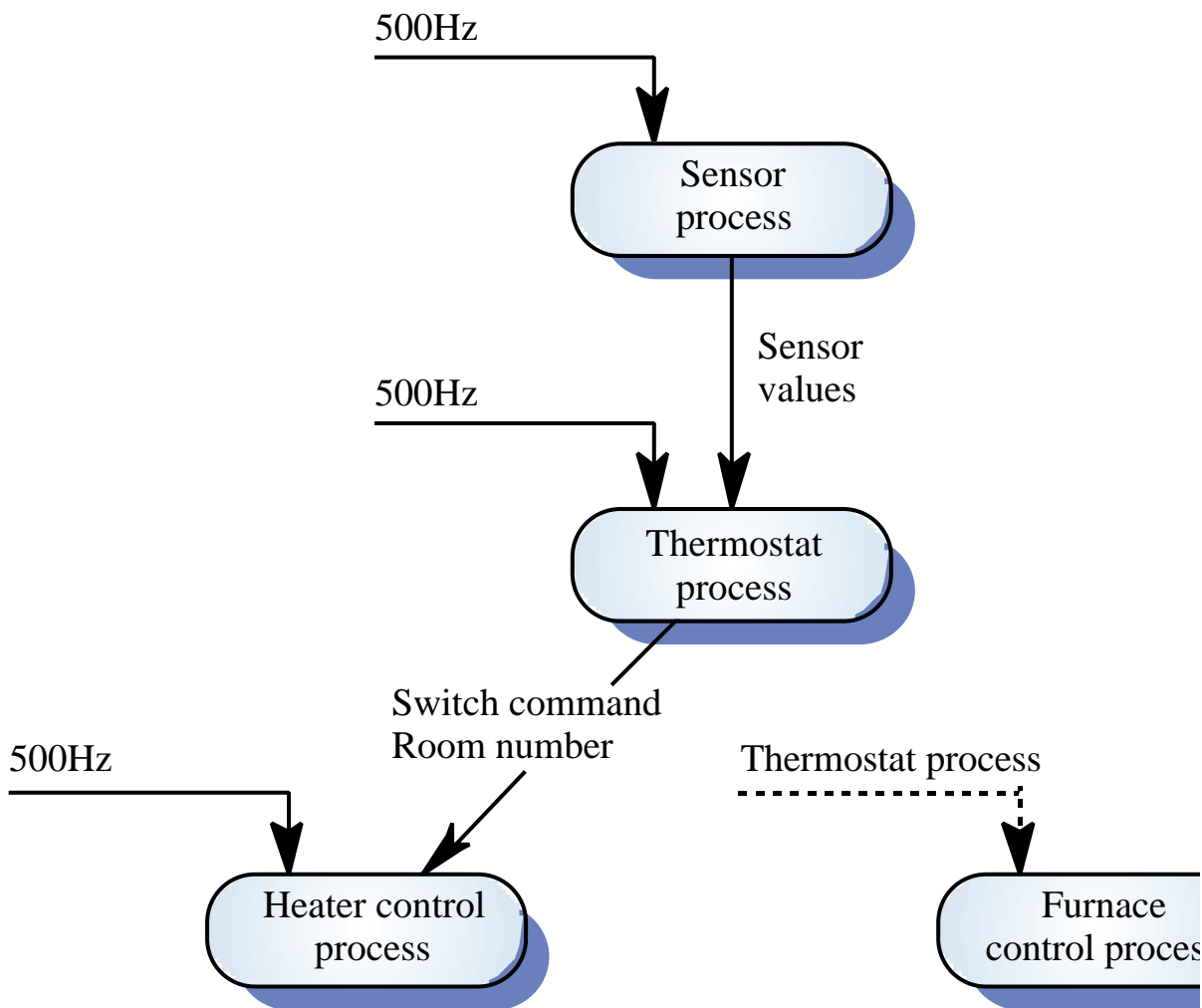
Define process architecture



Control systems

- Burglar alarm system is primarily a monitoring system. It collects data from sensors but has no real-time actuator control
- Control systems are similar but, in response to sensor values, the system sends control signals to actuators
- An 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

- Another important class of real-time systems which are concerned with acquiring data from the environment and processing this in some way
 - Satellite imaging system
 - Destructive testing systems
 - Reaction monitoring systems
- Normally, these systems have producer processes which are interfaced to sensors and produce data and consumer processes which (somehow) process that data

Data acquisition systems

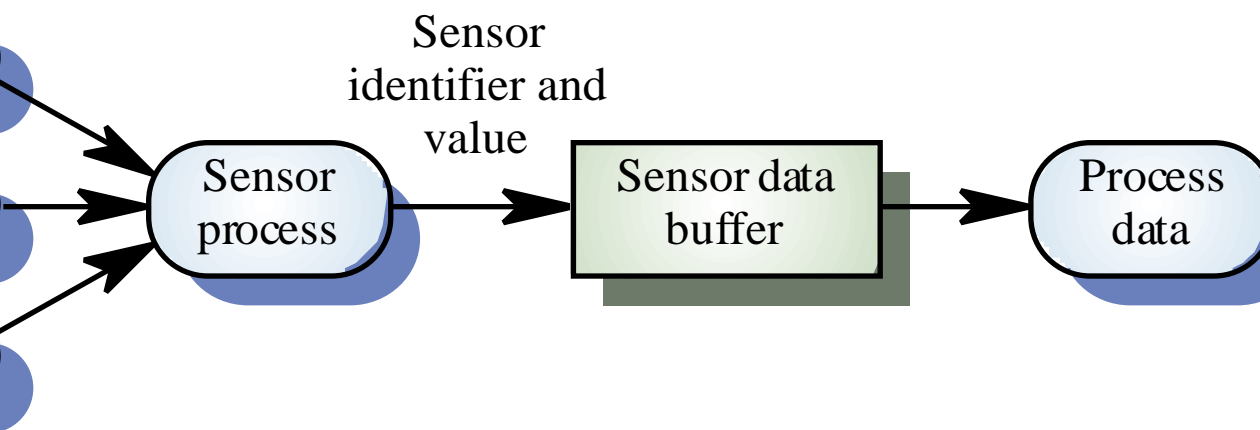
- Collect data from sensors for subsequent processing and analysis.
- Data collection processes (producers) and processing processes (consumers) may have different periods and deadlines.
- Data collection may be faster than processing e.g. collecting information about an explosion.
- Circular or ring buffers are a mechanism for smoothing speed differences.

Reactor data collection

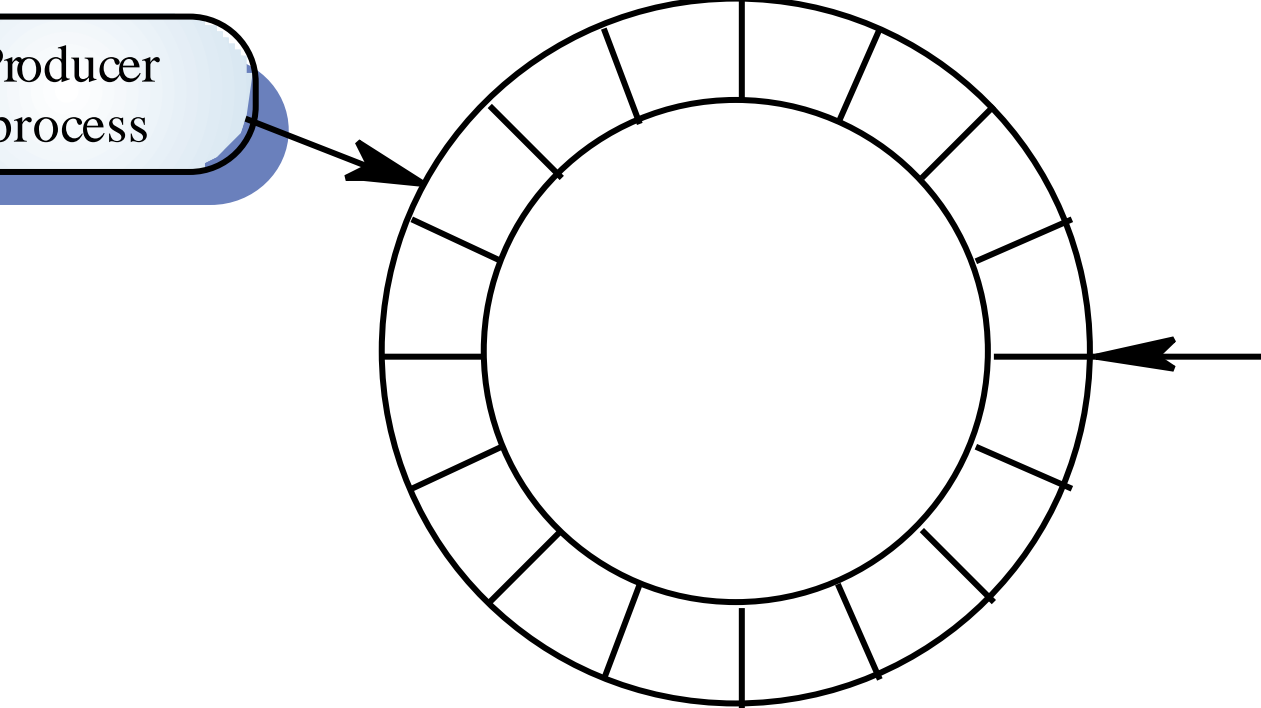
- System collects data from a set of sensors monitoring the neutron flux from a nuclear reactor.
- Flux data is placed in a ring buffer for later processing.
- The ring buffer is itself implemented as a concurrent process so that the collection and processing processes may be synchronized.

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.

Key points

- Real-time system correctness depends not just on what the system does but also on how fast it reacts
- Real-time systems are usually designed as a number of concurrent processes
- A R-T system model may associate processes with each class of sensor and actuator
- Monitoring and control systems poll sensors and send control signal to actuators
- Data acquisition systems are usually organised according to a producer consumer model

Real-time system programming

- Real-time systems can be developed in conventional languages such as C and C++. These have the advantage that they support low-level programming so that efficient code can be produced
- However, these languages lack any specific facilities for real-time systems development

Real-time programming facilities

- Facilities for creating, starting and stopping parallel processes
- Process synchronisation and control
- Access to a real-time clock
- Facilities to protect and manage shared resources
- Interrupt and exception management facilities
- Facilities to access and modify the priorities of processes at run-time to ensure the required timing behaviour

The Ada programming language

- Ada is a programming language that was designed in the late-1970s and came into widespread use in the 1980s, particularly for military applications
- The motivation for the language was to produce a single language for all US military systems development
- The inclusion of language-level facilities to support real-time systems development was a requirement for Ada

Ada tasks

- Tasks are the basic component type in Ada which are used to implement real-time systems
- Tasks are like objects and they have, within them, a set of services (like procedures) which can be called. These services (task entries) can run in parallel with entries from other tasks
- Task declarations declare the task interface, task bodies declare the task implementation
- Task services are defined using the ‘accept’ statement

Ada tasking

- Each entry has an associated queue. When an entry is called, this is queued and entries are serviced in turn
- Conditional entries for tasks are possible where an entry will not be processed unless some condition has been satisfied
- Select statements can be used to program conditional acceptance of task entries

Ada task specification

```
task Draw_person is  
    entry Head (H: Hat-size) ;  
    entry Body (B: Number_of_buttons) ;  
    entry Arms;  
    entry Legs;  
end Draw_person ;
```

Ada tasking example

task Thermocouple **is**

entry Get_temperature (T: in out TEMPERATURE) ;

entry Calibrate (T: TEMPERATURE) ;

entry Disconnect ;

end Thermocouple ;

task Controller **is**

entry Initialize ;

entry Control ;

entry Shutdown ;

end Controller ;

Ada tasking example

```
task body Thermocouple is  
begin  
    accept Get_temperature (T: in out TEMPERATURE) do  
    -- code here to interrogate the hardware  
    end Get_temperature ;  
    accept Calibrate (T: TEMPERATURE) do  
    -- code here to calibrate the thermocouple  
    end Calibrate ;  
    accept Disconnect do  
    -- code to implement a hardware shutdown  
    end Disconnect ;  
end Thermocouple ;
```

Real-time systems and Java

- Java was originally designed for writing simple embedded control systems so has some facilities (threads and monitors) that would appear to make it suitable for real-time systems development
- However, the requirement that Java must be portable (hence you can't predict the timing behaviour of Java programs) means that it cannot be used for real-time systems development where time is a critical factor

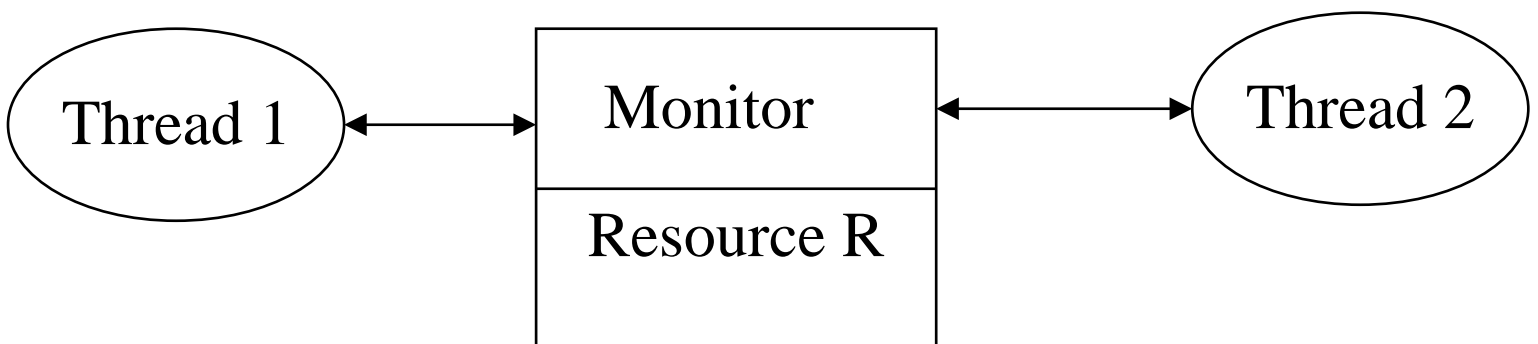
Java threads

- Threads in Java are used to implement concurrently executing objects and these can serve as processes in a real-time system
- Threads may be created simply by extending the Thread class

```
class Thermocouple extends Thread
{
    ... attribute declarations here ...
    public Temperature Get_temperature ( ) ;
    public void Calibrate (Temperature t) ;
    public void Disconnect ( ) ;
} // Thermocouple ;
```

Monitors in Java

- A monitor is a construct which allows a shared resource to be protected against accidental concurrent updates. Only one thread may access monitored resource at any one time.



Monitor declaration

- The `synchronized` keyword is used to identify a monitor. It may be used to create methods that may only be called by one thread at one time:

```
class CircularBuffer
{
    ... private attribute declarations here ...
    public synchronized void put ( BuffEntry b ) ;
    public synchronized BuffEntry get ( ) ;

} // CircularBuffer
```

Data structure synchronization

- The synchronized keyword may also be associated directly with a data structure that is to be protected against concurrent access. This then means that only a single thread can access this data structure at any one time

```
synchronized (windowSensors)
{
    s = windowSensors [n] ;
}
```

Java exception handling

- The exception management facilities in Java allow transfer of control to an exception manager when an exception is detected. This facility is essential for implementing non-stop systems

```
if (cumulativeDose >= maxDailyDose)
    throw new DailyDoseExceededException ( ) ;
....
catch (DailyDoseExceededException e)
{
    Alarm.Activate (“Maximum daily dose exceeded”) ;
    ControlSwitch.switchIt (ControlSwitch.off) ;
}
```

Java problems

- Java is only suitable for use in real-time systems programming where there are no stringent timing requirements
- Problems with Java for RTS development are:
 - Lack of facilities to specify that threads should run at particular times
 - Uncontrollable garbage collection
 - Memory fragmentation from garbage collector
 - Inability to determine queue sizes for monitors
 - Space inefficiency
 - Inability to carry out run-time space or processor time analysis

Key points

Languages for real-time systems programming should have facilities for creating and managing concurrent processes. Access to a real-time clock is essential and the language should allow explicit timing statements to be made

Ada is a language for military systems that has been specifically designed for RTS development

Java has concurrent programming facilities for does not allow enough detailed control over thread timing to be suitable for hard RTS development

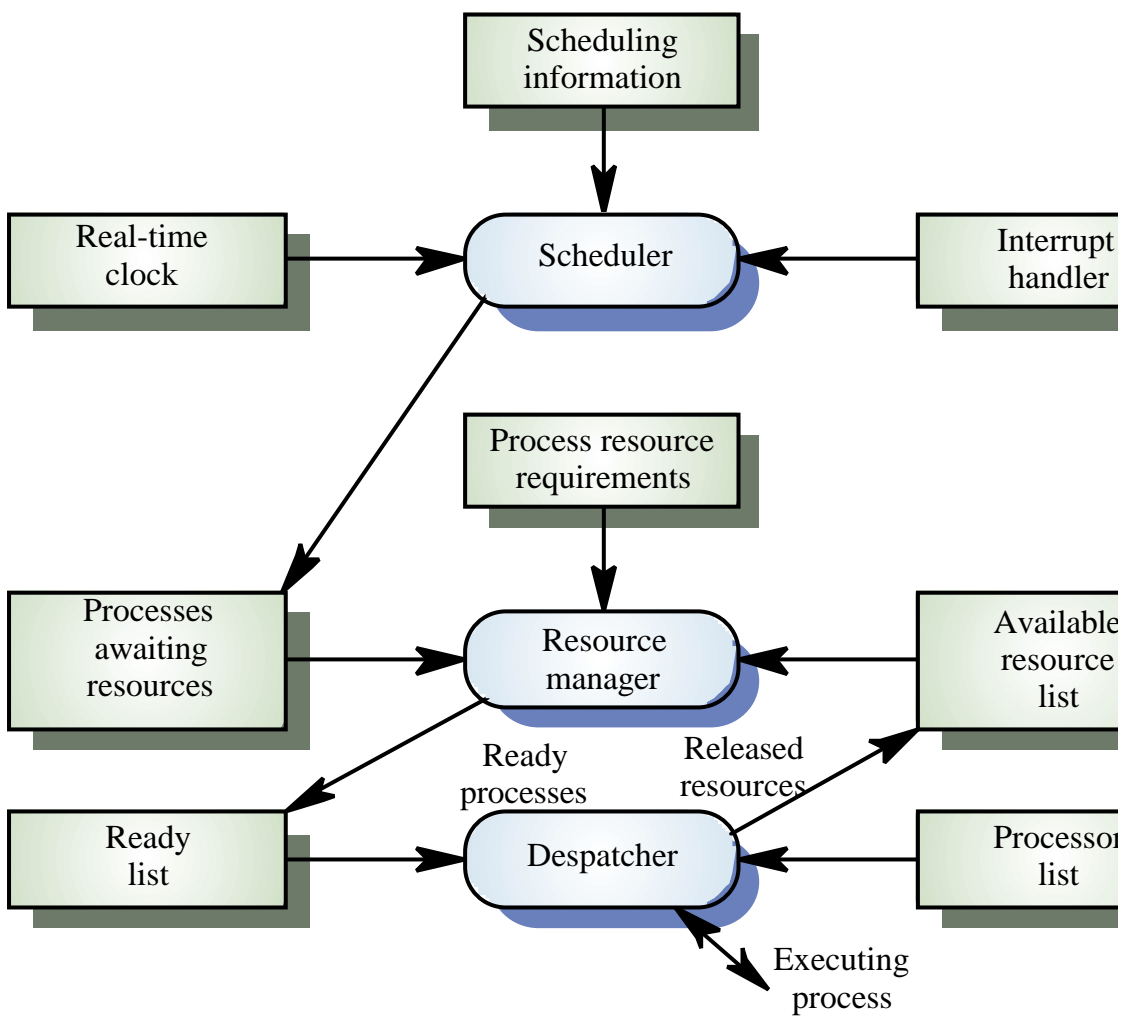
Real-time operating systems

- Real-time systems must often run on minimal hardware with no disks, limited memory, etc. They sometimes run on ‘bare machines’ i.e. machines without any operating system.
- Conventional operating systems are not designed to support real-time systems as they are:
 - Too big
 - Not designed for rapid process switching
 - Include unnecessary functionality
 - Unpredictable in their timing behaviour

Real-time executives

- Real-time executives are specialised 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
- They normally do not include facilities such as file management

Real-time executive components



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

- Non-stop systems are systems such as telephone switches that must run continually. Typical down-time is a few minutes per year
- 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.

Process attributes

- The RTE uses a number of attributes of each process to help make decisions on process management
 - Process priority The priority of the process wrt other processes in the system. The process priority depends on the role of the process in the system and its deadline.
 - Process period The number of times per second that the process is to be scheduled for execution
 - Process deadline The time by which the process must complete its operation
 - Process execution time(s) The average and worst case execution times for the process

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

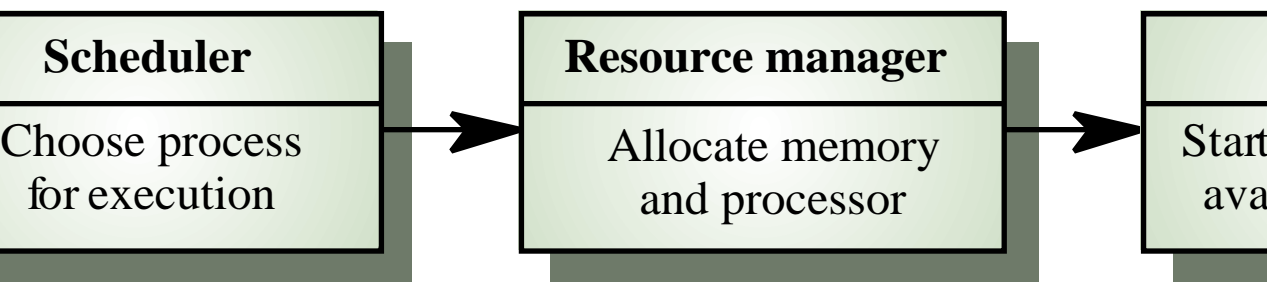
Interrupt servicing

- 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 control returned to the interrupted process
- Interrupt service routines **MUST** be short, simple and fast

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

TE 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

process scheduling

- Effective scheduling of processes is essential if the timing constraints associated with the RTS are to be satisfied
- The tasks in the system must be allocated processor resources in such a way that all tasks meet their specified deadlines
- The role of the scheduler is to choose a task for execution in such a way that scheduling constraints are satisfied

Scheduling strategies

- Non-preemptive scheduling
 - Once a task has been scheduled, it runs to completion
 - Causes problems when a higher priority task becomes available for execution while a lower priority task is running
- Preemptive scheduling
 - A task with a lower priority is halted (execution is preempted) and the processor is assigned to a higher priority task
 - The scheduler must ensure that the lower priority task eventually gets a share of processor resources (it could continually be preempted by higher priority tasks). One way to do this is to increase the priority of a preempted task when it loses control of the processor

Static and dynamic scheduling

- **Static scheduling**
 - All scheduling decisions are made at compile time and a table is created that completely defines the scheduling of processes in the system. All information (maximum execution times, deadlines, etc) for all processes must be known in advance
- **Dynamic scheduling**
 - The scheduler makes scheduling decisions at run-time based on information about the processes that are ready for execution
- **Dynamic scheduling is more flexible and makes better use of processor resource; static scheduling is more predictable and verifiable**

Key points

- Real-time executives are simply operating systems that are responsible for process and resource management.
- Components of an RTE are a clock, interrupt handler, a scheduler, a resource manager and a dispatcher
- Scheduling decisions are very important to ensure that real-time constraints are met by the system