Chapter 13
Real-Time Software Design
Designing embedded software systems whose behavior is subject to hard time constraints

Objectives
- To explain the concept of a real-time system and why these systems are usually implemented as concurrent processes
- To describe a design process for real-time systems
- To explain the role of a real-time executive
- To introduce generic architectures for monitoring and control and data acquisition systems

Topics covered
- Systems design
- Real-time executives
- Monitoring and control systems
- Data acquisition systems

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 period
- 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 the system. Partition functions to either hardware or software.
- Design decisions should 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
Real-time 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

Real-time 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 them by using a real-time executive or operating system

Timing constraints

- May require extensive simulation and experiment 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 modelling

- 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 modelling real-time systems.
- However, FSM models lack structure. Even simple systems can have a complex model.
- The UML includes notations for defining state machine models

Microwave oven state machine

Real-time programming

- Hard-real time systems may have to be programmed in assembly language to ensure that deadlines are met
- Languages such as C allow efficient programs to be written but do not have constructs to support concurrency or shared resource management
- Ada is a language designed to support real-time programming, so it automatically provide each software system with a real-time executive.
Java as a real-time language

- Java supports lightweight concurrency (threads and synchronized methods) and can be used for some soft real-time systems
- Java 2.0 is not suitable for hard RT programming or programming where precise control of timing is required
  - Not possible to specify thread execution time
  - Uncontrollable garbage collection
  - Not possible to discover queue sizes for shared resources
  - Variable virtual machine implementation
  - Not possible to do space or timing analysis

Real-time executives

- Real-time executives are specialized operating systems which manage the processes in a 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 timing 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

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

Process management

- Concerned with managing the set of concurrent processes
- Periodic processes are executed at pre-specified time intervals
- The executive uses the real-time clock to determine when to execute a process
- Process period - time between executions
- Process deadline - the time by which processing must be completed

RTE process management

- Scheduler
  - Choose process for execution
- Resource manager
  - Allocate memory and processor
- Dispatcher
  - Start execution on an available processor

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
- The dispatcher takes the process from ready list, loads it onto a processor and starts execution

Scheduling strategies

- Non pre-emptive scheduling
  - Once the execution of a process has been started, it runs to completion or until it is blocked for some reason (e.g. waiting for I/O)
- Pre-emptive scheduling
  - The execution of an executing processes may be suspended if a higher priority process requires service
- Scheduling algorithms
  - Round-robin
  - Rate monotonic
  - Shortest deadline first
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

Burglar alarm system

- A 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, the system switches on lights around the area and calls police automatically
- The system should include provision for operation without a mains power supply

Burglar alarm system

- Sensors
  - Movement detectors, window sensors, door sensors.
  - 50 window sensors, 30 door sensors and 200 movement detectors
  - Voltage drop sensor
- 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.

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

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

- A burglar alarm system is primarily a monitoring system. It collects data from sensors but 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

- 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

- A 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

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.

Java implementation of a ring buffer 1

```java
class CircularBuffer
{
    int bufsize ;
    SensorRecord [] store ;
    int numberOfEntries = 0 ;
    int front = 0, back = 0 ;
    CircularBuffer (int n) {
        bufsize = n ;
        store = new SensorRecord [bufsize] ;
    } // CircularBuffer
    synchronized void put (SensorRecord rec) throws InterruptedException
    {
        if ( numberOfEntries == bufsize)
            wait () ;
        store [back] = new SensorRecord (rec.sensorId, rec.sensorVal) ;
        back = back + 1 ;
        if (back == bufsize)
            back = 0 ;
        numberOfEntries = numberOfEntries + 1 ;
        notify () ;
    } // put
```

Java implementation of a ring buffer 2

```java
synchronized SensorRecord get () throws InterruptedException
{
    SensorRecord result = new SensorRecord (-1, -1) ;
    if (numberOfEntries == 0)
        wait () ;
    result = store [front] ;
    front = front + 1 ;
    if (front == bufsize)
        front = 0 ;
    numberOfEntries = numberOfEntries - 1 ;
    notify () ;
    return result ;
} // get
```
<table>
<thead>
<tr>
<th>Key points</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Real-time system correctness depends not just on what the system does but also on how fast it reacts</td>
</tr>
<tr>
<td>● A general RT system model involves associating processes with sensors and actuators</td>
</tr>
<tr>
<td>● Real-time systems architectures are usually designed as a number of concurrent processes</td>
</tr>
<tr>
<td>● Real-time executives are responsible for process and resource management.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key points (continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Monitoring and control systems poll sensors and send control signal to actuators</td>
</tr>
<tr>
<td>● Data acquisition systems are usually organized according to a producer-consumer model</td>
</tr>
<tr>
<td>● Java has facilities for supporting concurrency but is not suitable for the development of time-critical systems</td>
</tr>
</tbody>
</table>