# KOE-061 REAL TIME SYSTEMS (Unit 1)

Saurabh Mishra

Assistant Professor

Department of Information Technology

# Unit 1 Syllabus

Definition, Typical Real Time Applications: Digital Control, High Level Controls, Signal Processing etc.,

Release Times, Dead-lines, and Timing Constraints,

Hard Real Time Systems and Soft Real Time Systems,

Reference Models for Real Time Systems: Processors and Resources, Temporal Parameters of Real Time Workload, Periodic Task Model, Precedence Constraints and Data Dependency.

# Content

- Introduction
- RTOS vs Other OS
- Characteristics of RTOS
- Types of RTOS, Embedded Systems
- Reference Models
- Temporal Parameters, Precedence and Timing Constraints

# Introduction

- Revision
  - Operating system
  - Drivers
  - Application software
  - Uniprocessor and Multiprocessor system

# Introduction

- What is real time system?

- How real time system is different from traditional OS?
  - Major issues

# Introduction

- **Real Time** is a quantitative notion of time measures using physical clock
- Example:
    - If force >500 newton, airbags inflate in 100 MS
    - OR
    - If Temp>100 Deg Cel. Then coolant shower start in 200 MS
- In traditional OS
    - Not very strict time bound process
    - Before, after, sometimes , eventually etc

# Introduction

- Real time Systems
  - Time constrained
  - Mostly embedded system
    - An embedded system is **a combination of computer hardware and software designed for a specific function.**
    - **Examples:** Toy, UAV, Reactor, Spaceship, satellite, ATC.
  - Mostly real time systems are embedded systems but not all embedded systems are real time systems.

# Introduction

- RTOS
  - Deadline specific
  - Task Scheduling primary mechanism to meet respective deadline
- Why there is surge in embedded system
  - Example: AC, Microwave, TV, Fridge, smoke detection system or any sensor system.
  - Processor and internet get economic
  - Reduces size
  - Reduce power consumption
  - Increase processing power and hardware and software reliability

# Introduction

- Example: An automobile airbag system


Respond in a time-bound fashion or the system fails

# Introduction

- How RTOS different from other OS?
  - An embedded system respond to external inputs:
    - If response is late, then system failure
  - General Purpose
    - Minimize the response time and ensures fairness
  - Help tasks to meet deadline(very hard deadline)

# Introduction

- Characteristics of a RTOS
  - Real time:
    - At least few tasks have real-time constrains
    - Eg. Deadline
  - Correctness Criterion:
    - Result should be logically correct
    - And also within stipulated time

# Introduction

- Why have an OS in an embedded device?
  - Multitasking, scheduling, synchronization
  - Timing aspects
  - Memory management
  - File or data handling
  - Networking/communication
  - Graphic display
  - Interfacing and wide range of IO device
  - Scheduling and buffering of IO operations
  - Security and power management

# Introduction

- Example: A smartphone operating system contains over million lines of code
- Project will hardly have time and funding
- Typical embedded OS license fees are less than a desktop OS
- Some very simple low end devices might not need an OS
  - Devices have some complex tasks to do.

# Introduction

- Types of Real Time Systems:
  - Real Time Systems are different from traditional System
    - Task specific, Deadline specific
  - Classified largely based on the consequence of not meeting deadline
    - Hard real time systems
    - Soft real time systems
    - Firm real time systems

# Introduction

- Hard Real Time Systems
  - If deadline not met, System failed
  - Task deadline are in micro or milliseconds
  - Mostly safety critical
- Firm Real Time Systems
  - If deadline is missed occasionally the system does not fail
  - The result produced by a task after the deadline are ignored
  - Example: Video conference, other sensors

# Introduction

- Soft Real Time Systems
  - Utility of the result decreases with time after deadline
  - If several times deadline missed the system utility decreased
  - Use probabilistic requirements on deadline
  - 99% of the time deadline met
  - Example: All interactive applications, phone calls, reservation system.

# Type of Task

- Periodic
  - Recur according to a timer
  - A vast majority all real-time tasks are periodic
- Aperiodic
  - Recur randomly and are soft real-time tasks
- Sporadic
  - Recur randomly but hard real-time tasks

# Timing Constraints

- A timing Constraint:
  - Defined with respect to some event

- An event:
  - Occurs at an instant of time
  - Generated either by the system or environment

# Real Time Tasks

- Real Time tasks get generated due to certain event occurrences
  - Either internal or external events
- Example
  - A task may get generated due to a temperature sensor sensing high level
- When a task gets generated
  - It is said to be released or arrived

# Real Time Task Scheduling

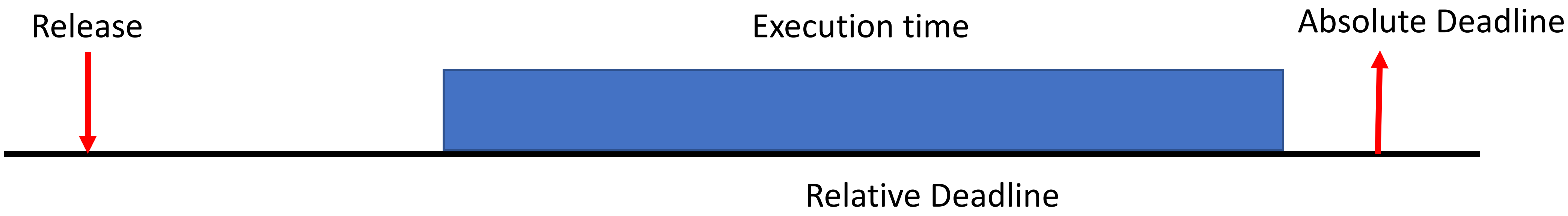
- Essentially refers to the order in which the various tasks are to be executed

- It is the primary means adopted by an operating system to meet task deadlines

- Obviously Scheduler is a very important component of a RTOS

# Real Time Workload

- Job
  - A unit of work
  - A computation , a file read, message passing, etc
  - A task instance
- Task: A sequence of similar jobs



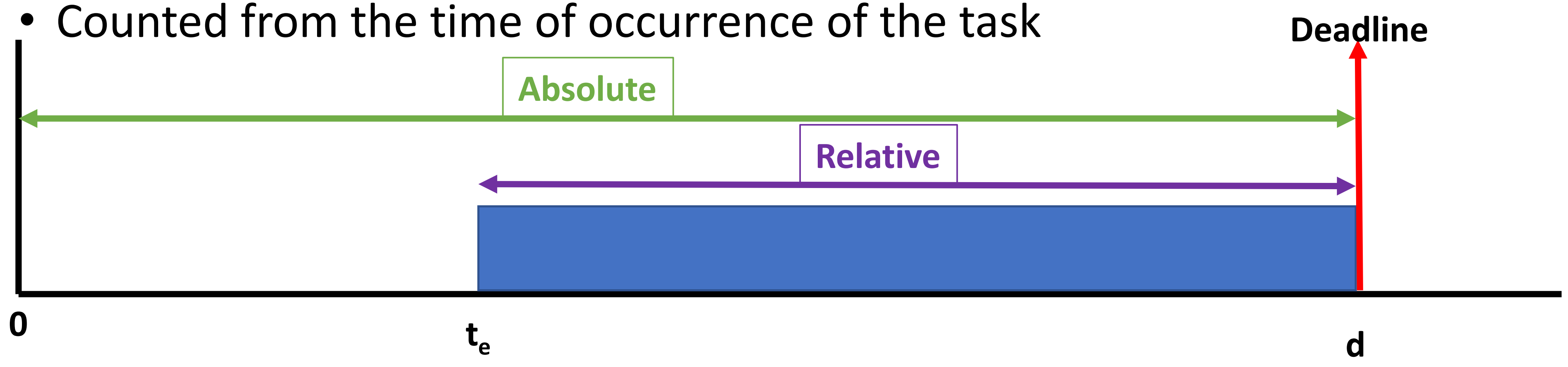
# Task Instance(Job)

- A task typically recurs a large number of times
  - Each time triggered by an event
  - Each time a task recurs, an instance of the task is said to have been generated or released

- The $i^{th}$ time a task T occurs
  - Job or task instance $T_i$ is said to have arrived

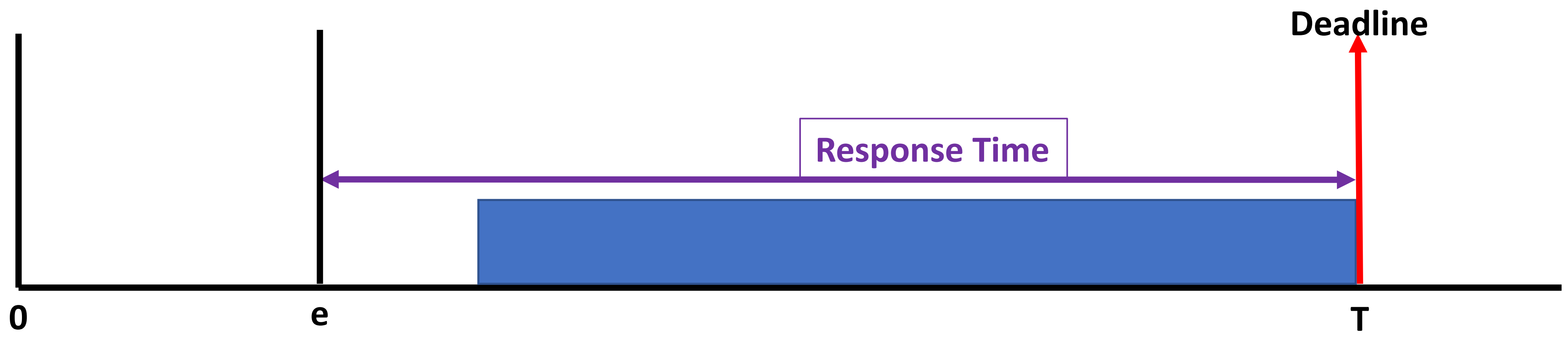
# Relative and Absolute Deadline

- Absolute Deadline
  - Counted from time 0
- Relative Deadline
  - Counted from the time of occurrence of the task



# Response Time

- Duration between task release time and task completion time
- Release time
  - The time of occurrence of the event generating the task
- Completion Time
  - Result produced by the task



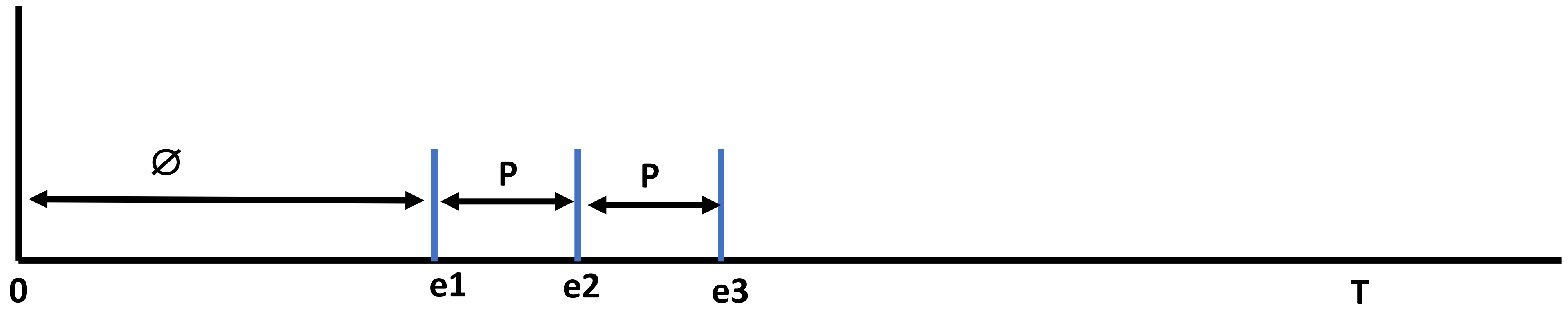
# Response Time

- For soft real time tasks:
  - The response time needs to be minimized

- For Hard Real Time task:
  - As long as the task completed within its deadline
    - No advantage of completing it any early

# Phase of a Periodic Task

- Phase for a periodic task:
  - The time from 0 till the occurrence of the first instance of the task
  - Denoted by $\varnothing$



# Phase Time Example

- A track correction task starts 2000 ms after the launch of the rocket
  - Periodically  recurs every 50 ms then on
  - Each instance of the task requires a processing time of 8 ms and its deadline is 50 ms

# Few Task Scheduling Technologies

- Valid Schedule
  - At most one task assigned to the processor at a time
  - No task is scheduled before its ready
  - Precedence and resource constraints of all tasks are satisfied
- Feasible Schedule
  - Valid schedule in which all tasks meet their respective time constraints

# Scheduling Terminologies

- Proficient schedule
    - A scheduler S2 is more proficient as compared to another scheduler S1
        - If whatever task that S2 can feasibly schedule so can S1, but not vice versa
- Equally Proficient schedule
    - If task set feasibly scheduled by the one can also be scheduled by other and. vice versa
- Optimal Scheduler
    - An optimal scheduler can feasibly schedule any task set that can be scheduled by any other scheduler

# Scheduling Points

- At the points on the time line
  - Scheduler makes decision regarding which task to be run next
- Clock Driven
  - Scheduling points are defined by interrupts from a periodic timer
- Event Driven
  - Scheduling points are defined by task completion and generation events

# Task Scheduling on Uniprocessors

- During 1970-80s

- Real Time schedulers are broadly classified into:
  - Clock Driven
  - Event Driven

# Scheduling Summery

- Endless loop : No Task, Polled
- Simple cyclic executive : Single Frequency
- Multi-rate cyclic executive : Multiple Frequencies
- Priority based preemptive scheduler : Interrupt Driven

# Embedded Systems

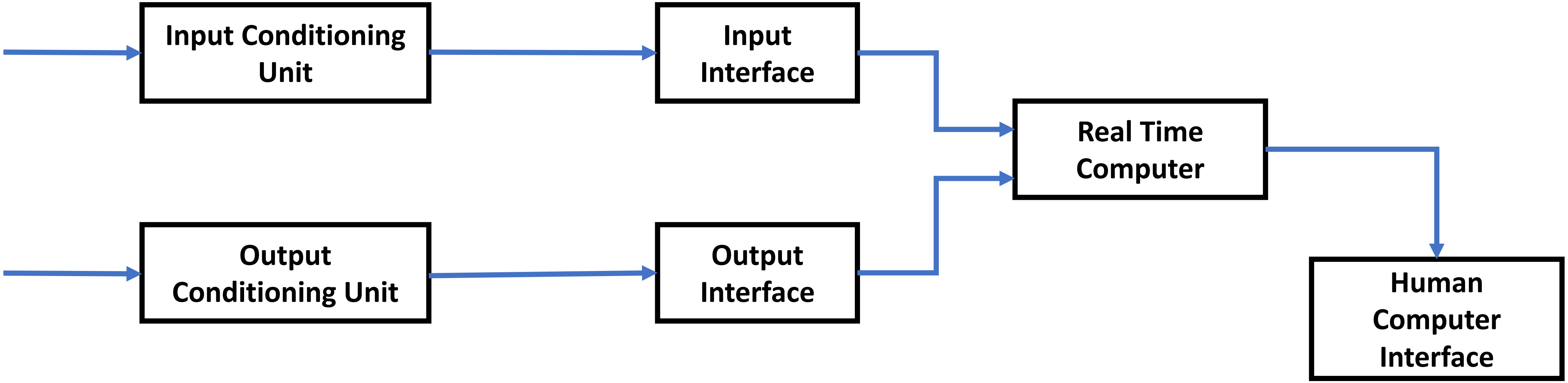
- Home appliances
  - TV, Fridge, Washing Machine, Microwave
- Toys
- Cars
- Vending Machines
- Trains
- Phones
- Radar and other Security equipments
- Rocket, Satellite

# Embedded Systems Applications

- Industry
  - Chemical Plants, Automobile Assembly lines, Robotic hands
- Medical
  - X-Ray, Ultrasound machines
  - Robotic hands
  - Wearable devices
- Peripheral equipment
  - Printer, camera, camcorders, sensors
- Transport
  - UAV, Automobiles

# Embedded Systems

- Why not implement all functionalities in hardware
  - Limited number of functionalities
- Basic Model of Embedded system



# Sensors

- A sensor converts some physical characteristic of its environment into electrical signals.

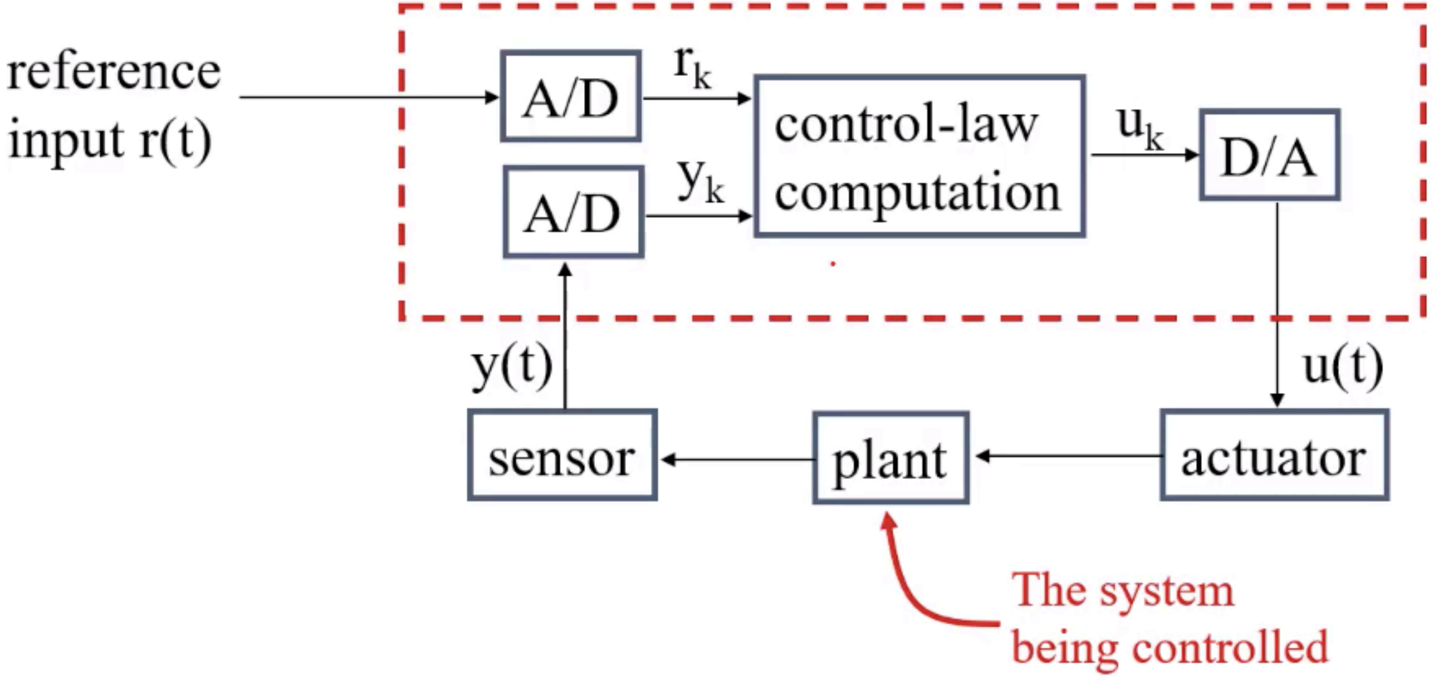
- Example sensors:
  - A photo-voltaic cell converts light energy into electrical energy.
  - A temperature sensor typically operates based on the principle of a thermocouple.
  - A pressure sensor typically operates based on the piezoelectricity principle.

# Actuators

- An actuator converts electrical signals from a computer into some physical actions.
- The physical actions may be:
  - Motion, change of thermal, electrical, pneumatic, or physical characteristics of some objects.
- Example actuators:-
  - Motors
  - Stepper Motor
  - Heaters
  - Hydraulic and pneumatic actuators

# Examples

- Many Embedded(Real Time) Systems are Control Systems
- A simple one sensor, one actuator control system



# Simple Control System

- Pseudo-code for this system:

```
set timer to interrupt periodically with period T;
at each timer interrupt do
      do analog-to-digital conversion to get y;
      compute control output u;
      output u and do digital-to-analog conversion;
end do
```

- T is called the sampling period. T is a key design choice. Typical range for T: seconds to milliseconds

# Multi-rate Control Systems

- More complicated control systems have multiple sensors and actuators and must support control loops of different rates.
- Example: Helicopter flight controller.

**Do the following in *each* 1/180-sec. cycle:**

validate sensor data and select data source;
if failure, reconfigure the system

***Every sixth* cycle do:**

keyboard input and mode selection;
data normalization and coordinate transformation;
tracking reference update
control laws of the outer pitch-control loop;
control laws of the outer roll-control loop;
control laws of the outer yaw- and collective-control loop

**Every *other* cycle do:**

control laws of the inner pitch-control loop;
control laws of the inner roll- and collective-control loop

Compute the control laws of the inner yaw-control loop;

Output commands;

Carry out built-in test;

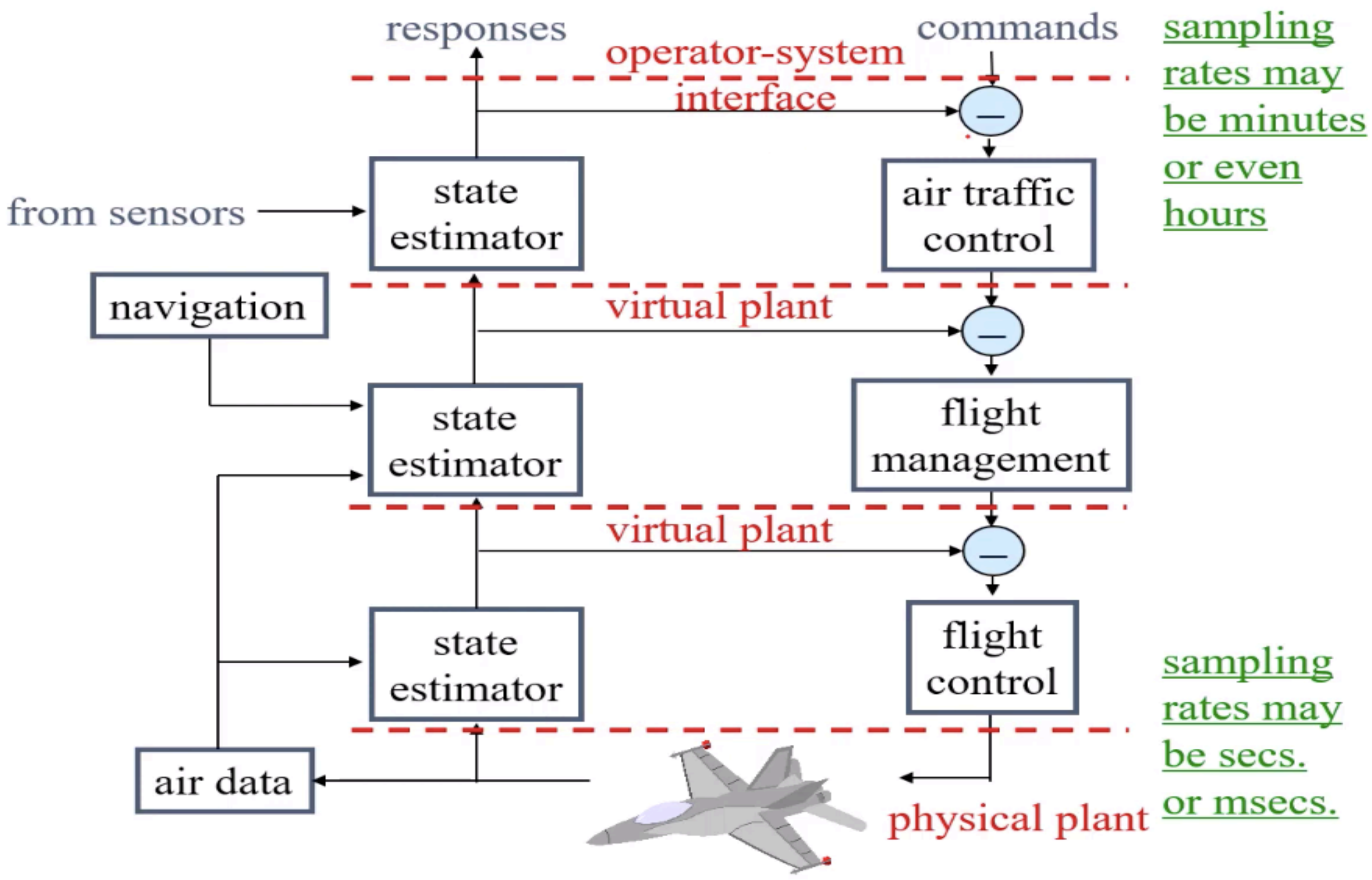
Wait until beginning of the next cycle

Note: Having only harmonic rates simplifies the system.

# Hierarchical Control Systems

**Example:**

**Air Traffic Control System**



# Signal-Processing Systems

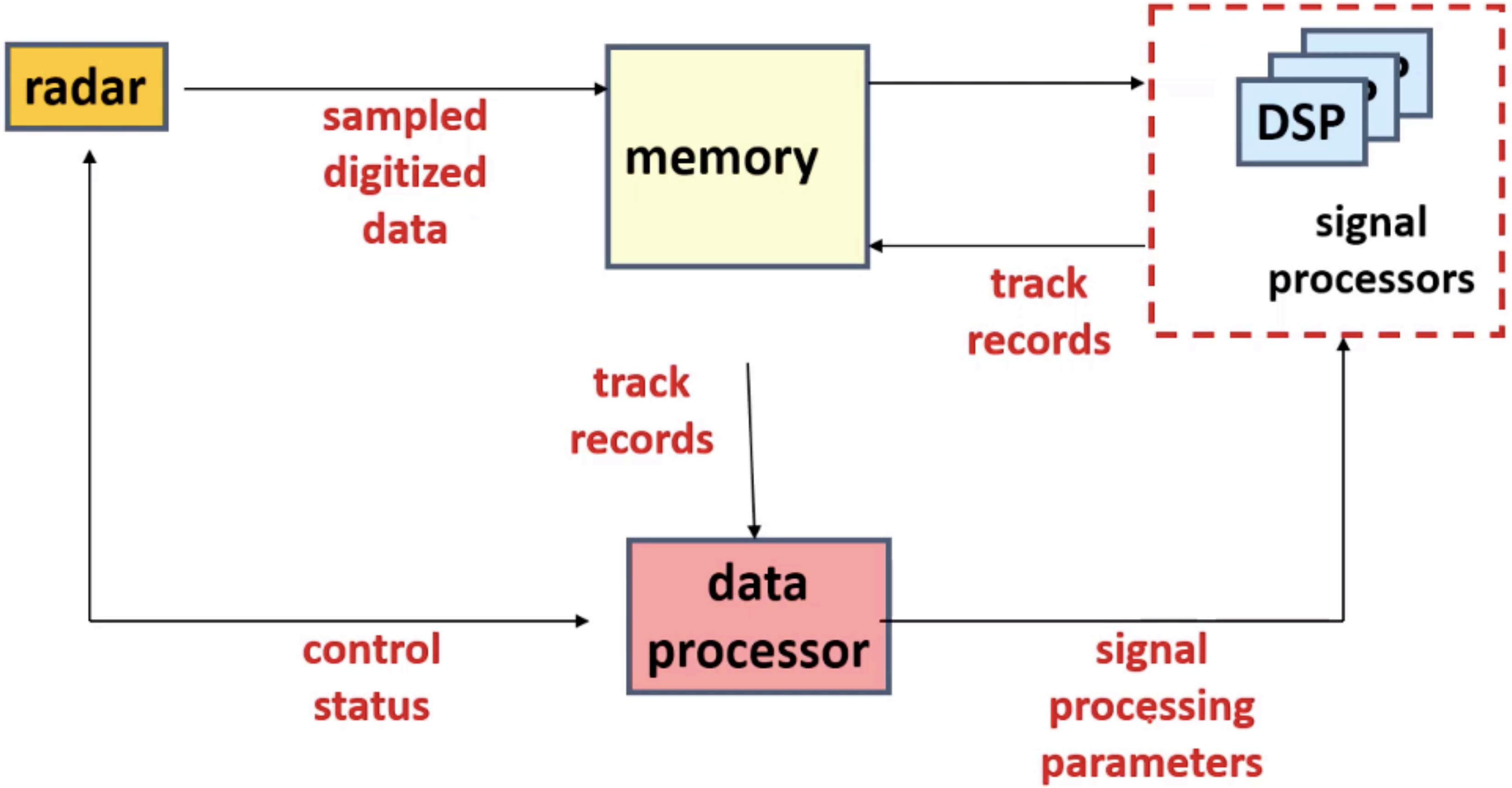
- Signal-processing systems transform data from one form to another.

- Examples:
  - Digital filtering
  - Video and voice compression/decompression.
  - Radar signal processing.

- Response times range from a few milliseconds to a few seconds.

# Signal Processing System

- Example: Radar System



# Signal Conditioning(Characteristics)

- Analog signals are generated by a sensor.
- A photo-voltaic cell normally generates signals in the millivolts range.
- Need to be conditioned before they can be processed by a computer.
- Important types of conditioning:
  - Voltage Amplification
  - Voltage Level Shifting
  - Frequency Range Shifting and Filtering
  - Signal Mode Conversion

# Embedded System(Characteristics)

- Characteristics of An Embedded System
- Real-time:
  - Some tasks are real-time
  - Each real-time task is associated with some time constraints, e.g. a Deadline.
- Correctness Criterion:
  - Results should be logically correct, And within the stipulated time.
- Safety and Task Criticality:
  - A critical task is one whose failure causes system failure (example: obstacle avoidance).

# Embedded System(Characteristics)

- A safe system does not cause damage.
  - A safety-critical real-time system is one where any failure causes severe damage
- Concurrency:
  - A Real Time system needs to respond to several independent events.
  - Typically, separate tasks process each independent event.
  - For the same inputs, the result can be different (Non-determinism)

# Embedded System(Characteristics)

- Distributed and Feedback Structure
- Custom Hardware:
    - An embedded system is often implemented on custom H/W that is specially designed and developed for the purpose.
- Reactive:
    - On-going interaction between computer and environment.
- Stability:
    - Under overload conditions, at least the important tasks should perform acceptably.
- Exception Handling

# Safety and Reliability

- A safe system:
  - Does not cause damage even when it fails.

- A reliable system:
  - Operates for long time without any failure.

- Independent concepts in traditional systems(Not much related).

# Safety and Reliability

- In traditional systems:
  - Safety and reliability are independent concerns.
  - A system can be safe and unreliable and vice versa.
  - Give examples of:
    - A safe and unreliable system
    - A reliable and unsafe system
- Interrelated in safety-critical system.
  - A safety critical system is one for which any failure of the system would result in severe damage.
- Safety can be ensured only through increased reliability.

# Safety and Reliability

- An unreliable system can be made safe upon a failure:
  - By reverting to a fail-safe state.

- A fail-safe state:
  - No damage can result if a system fails in this state.
  - Example: For a traffic light, all lights orange and blinking

# Fail-Safe State

- The fail-safe state of a word processing program:
  - The document being processed has been saved onto the disk.

- Fail-safe states help separate the issues of safety and reliability.
  - Even if a system is unreliable, it can always be made to fail in a fail-safe

# Safety and Reliability

- For a safety-critical system
  - No fail-safe state exists.
- Consider the navigation system on an aircraft:
  - When the navigation system fails:
  - Shutting down the engine can be of little help!
- As a result, for a safety-critical system:
  - The only way to achieve safety is by making it reliable.

# How to Design a Highly Reliable System?

- Error Avoidance
- Error Detection and Removing
- Fault Tolerance

# Fault Tolerance in RT System

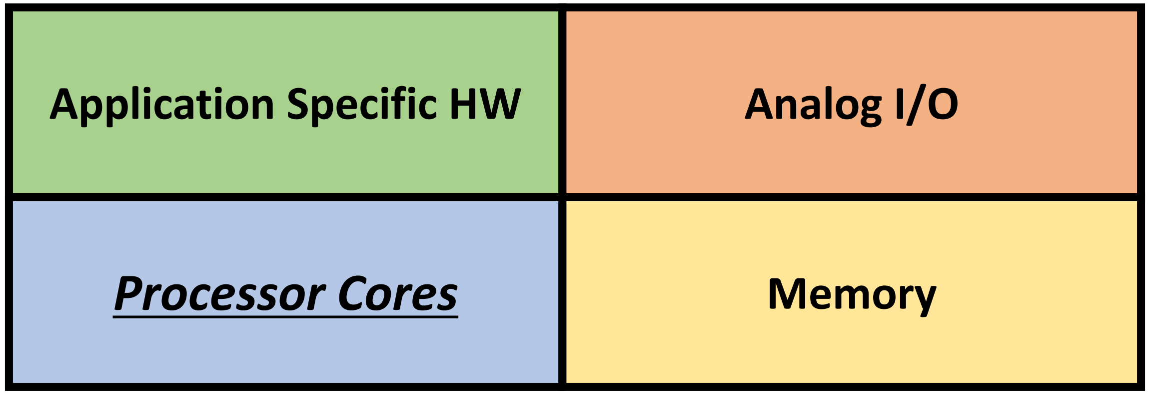
- Hardware Fault Tolerance:
  - Built in self test (BIST)
  - Triple modular redundancy(Polling/Voting)

- Software Fault Tolerance :
  - N-Version programming
  - Recovery Blocks

# N-Version Programming

- N-version Programming
- Software fault tolerance technique inspired by TMR of hardware:
  - Different teams are employed to develop the same software.
  - Unsatisfactory performance in practice.
  - Reason: Faults are correlated in the different versions.
  - All versions fail for similar reasons.

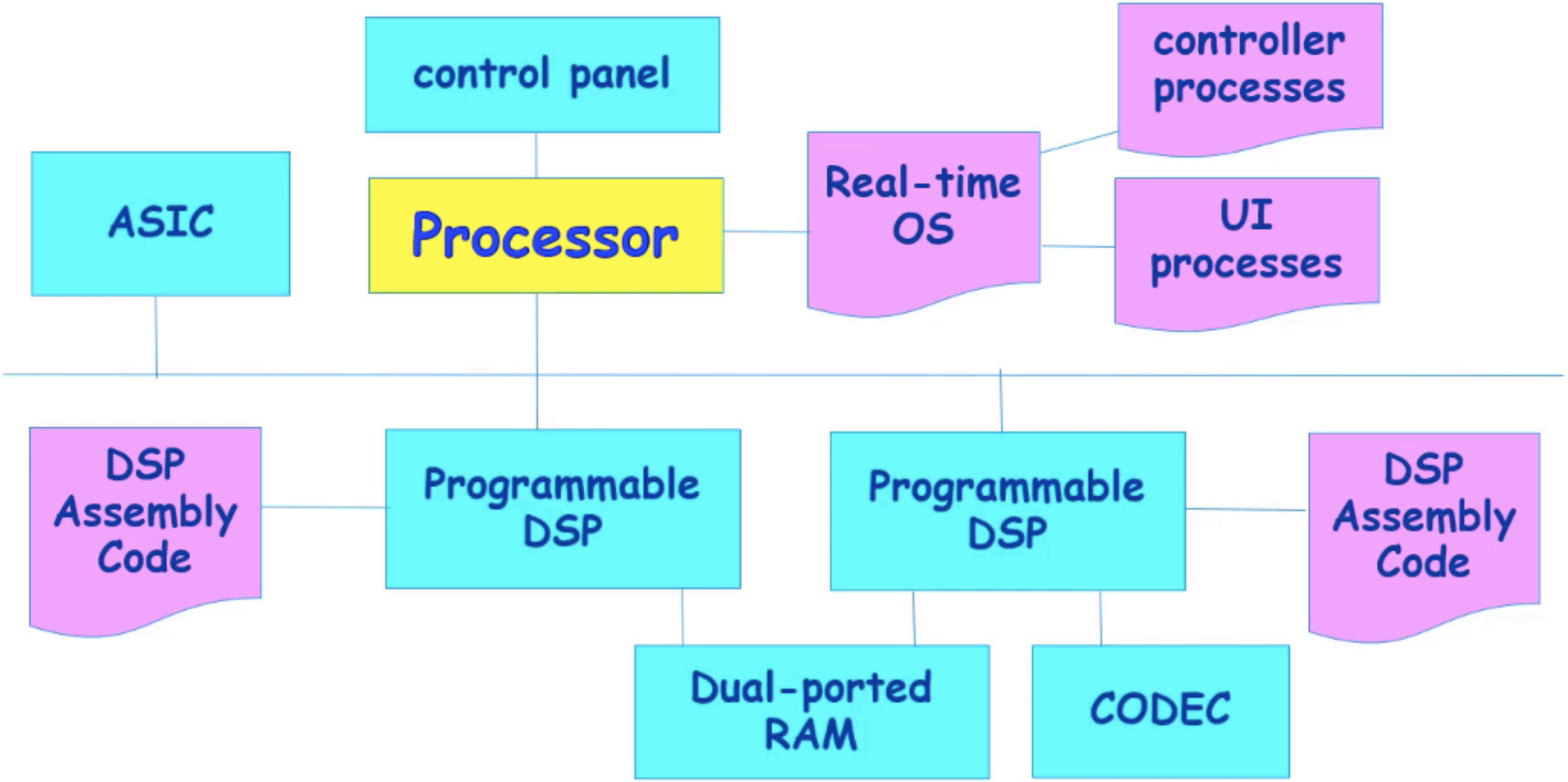
# Modern Embedded Systems



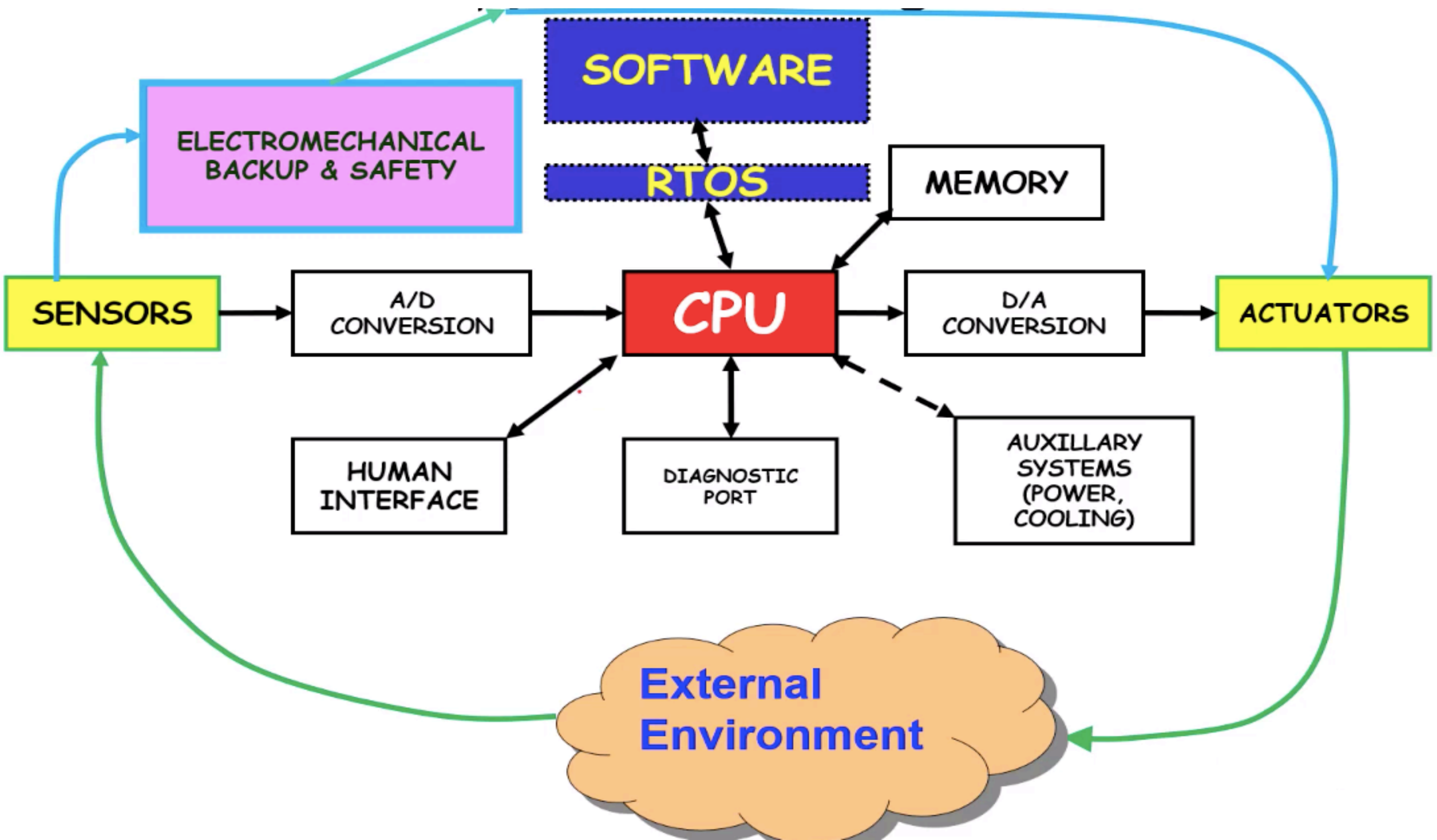

- Embedded systems incorporate:
  - Application-specific hardware (ASICS, FPGAs etc.).
    - performance, low power
  - Programmable processors: DSPs, controllers etc.
  - Mechanical transducers and actuators

# Block Diagram of An Embedded System

AMBA:
Advance
Microcontroller
Bus
Architecture



# Embedded System = Hardware + RTOS + Application Program



# Why OS is important

- Support for:
  - Multitasking, scheduling, and synchronization
  - Timing aspects
  - Memory management
  - File systems
  - Networking
  - Graphics displays
  - Interfacing wide range of I/O devices
  - Scheduling and buffering of I/O operations
  - Security and power Management

# Why Have an OS in an Embedded Device?

- Example: recent cell phone operating system contained over five million lines of code!
- Few, if any projects will have the time and funding:
  - To develop all of this code on their own!
- Typical Embedded OS license fees are a few dollars per device --- less than a desktop OS
- Some very simple low-end devices might not need an OS:
  - But new devices are getting more complex.

# Question to Design an Embedded System

- What is Actually Being Used?
- What Types of Processors are used?
- What Operating Systems are used?
- What Programming Languages are used?

# How is Real-Time OS Different from Traditional OS?

- An embedded system responds to external inputs:
  - If response is late, the system fails.

- General purpose OS:
  - Not designed for real-time use

- Real-time OS:
  - Helps tasks meet their deadline

# Exercise

- Explain, what are RTS.
- Differentiate between RTOS ad traditional OS.
- Explain, what are embedded systems.
- What do you mean by actuators and sensors.

# Types of Real-Time Systems

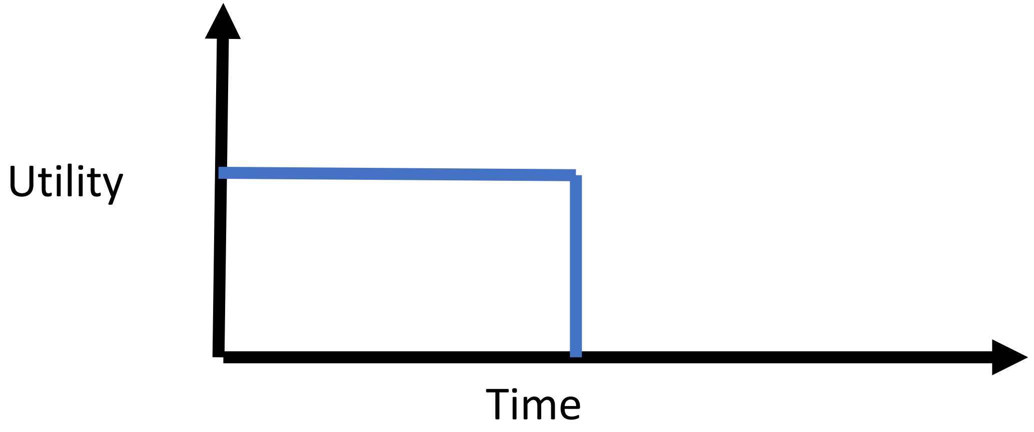
- Real-time systems are different from traditional systems:
  - Tasks have deadlines associated with them.
  - Classified based on the consequence of a failure:
    - Hard real-time systems
    - Soft real-time systems
    - Firm real-time systems

# Hard Real Time Systems

- If a deadline is not met: The system is said to have failed.
- The task deadlines are of the order of micro or milliseconds.
- Many hard real-time systems are safety- critical.
- Examples:
  - Industrial control applications•
  - On-board computers
  - Robots

# Firm Real-Time Systems

- If a deadline is missed occasionally, the system does not fail:
- The results produced by a task after the deadline are rejected.

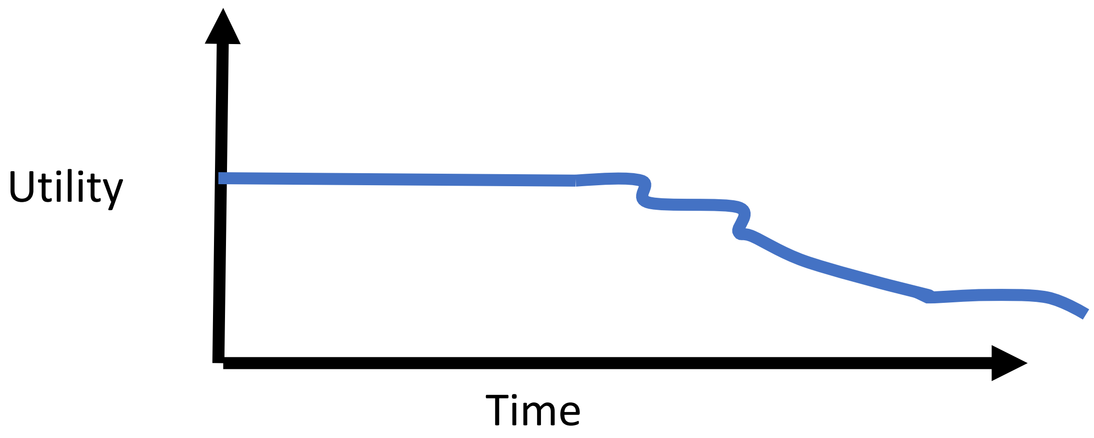


# Firm Real-Time Systems

- Examples:
  - A video conferencing application
  - A telemetry application
  - Satellite-based surveillance applications

# Soft Real-Time Systems

- Ifa deadline is missed, the system does not fail:
- Only the performance of the system is said to have degraded.
- The utility of a result decreases with time after the deadline.



# Soft Real-Time Systems

- Soft Real-Time Systems
- Another definition:
  - Use probabilistic requirements on deadline.
  - For example, 99% of deadlines will be met.

# Types of Real-Time Tasks

- Periodic:
  - Periodic tasks repeat after a certain fixed time interval.
- Sporadic:
  - Sporadic tasks recur at random instants.
- Aperiodic:
  - Same as sporadic except that minimum separation between two instances can be

# Timing Constraints

- A timing constraint:-
  - Defined with respect to some event.
- An event:
  - Can occur at an instant of time
  - May also occur over a duration
  - Generated either by the system or its environment

# Events in a Real-Time System

- Events in a real-time system can be classified into:-
  - Stimulus Events
  - Response Events

# Stimulus Event

- Generated by the environment:
  - Act on the system.
- Typically asynchronous in nature:
  - Aperiodic
  - Can also be periodic
- Example:
  - Aperiodic: Telephone System
  - A user pressing a button on a telephone set
  - Stimulus event acts on the telephone system.
  - Periodic stimulus event example
    - Periodic sensing of temperature in a chemical plant.

# Response Event

- Produced by the system:
  - In response to some stimulus events

- Example:
  - In a chemical plant as soon as the temperature exceeds 100°C,
  - The system responds by switching off the heater.

# Classification of Timing Constraints

- Different timing constraints can broadly be classified into:

- Performance constraints
  - Imposed on the response of the system.

- Behavioral constraints
  - Imposed on the stimuli generated by the environment.

# Types of Timing Constraints

- Both performance and behavioral constraints can be classified into:
  - Delay Constraints
  - Deadline Constraints
  - Duration Constraints

# Delay Constrains

- Expresses minimum time delay:
- Allowed between the occurrence of two arbitrary events e1 and e2

- T(e2) – T(e1) >= d

- If e2 occurs earlier than d then a delay violation would occur

# Deadline Constrains

- Expresses the maximum permissible separation:
  - Between events. any two arbitrary

- T(e2) – T(e1) <= d

# Duration Constrains

- A duration constraint on an event:
  - Specifies the time period over which the event acts.

- A duration constraint can be:-
  - minimum type
  - maximum type.

# SS Deadline Example

- Deadline is defined between two stimuli.
  - A behavioral constraint.
  - Imposed on stimulus.

- Once a user completes dialling a digit,
  - He must dial the next digit within the next 5 seconds.
  - Otherwise an idle tone is produced.

# RS Deadline Example

- Deadline is defined on the stimulus from the respective response event.
  - A behavioral constraint.
  - Imposed on stimulus.

- Once the dial tone appears:
  - The first digit must be dialled within 30 seconds,
  - Otherwise the system enters an idle state and an idle tone is produced

# RR Deadline Example

- Deadline is defined on the response from another response.
  - A performance constraint.
  - Imposed on response.

- Example: PSTN
- Once ring tone is given to the callee,
  - Ring back tone must be given to the caller within two seconds,

# SR Deadline Example

- Deadline is defined on the response from the respective stimulus.
  - A performance constraint.
  - Imposed on response.

- Once the receiver of the hand set is lifted:
  - The dial tone must be produced by the system within 2 seconds,
  - otherwise a beeping sound is produced until the handset is replaced.

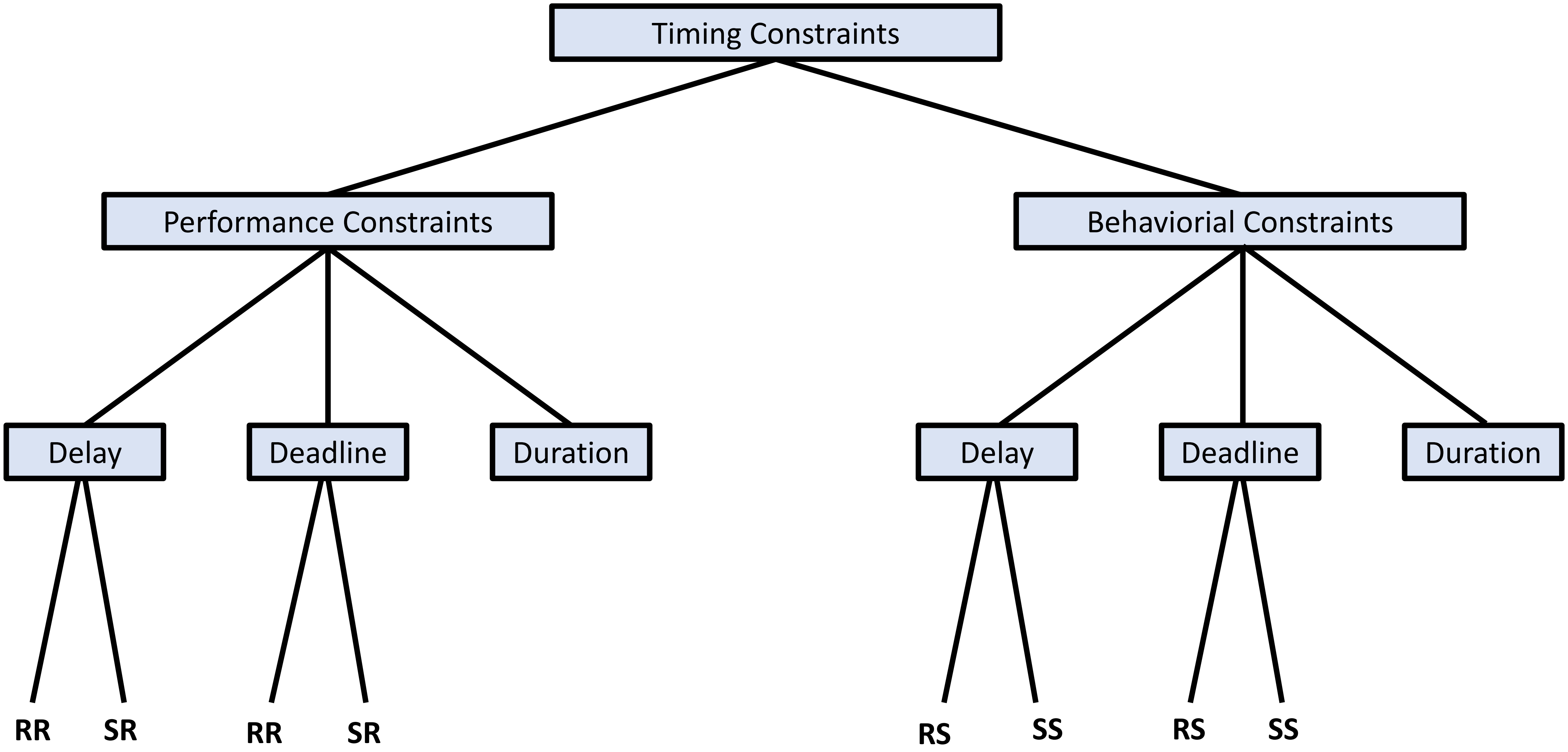
# SS Type Delay Constraint

- A behavioral constraint.
  - Imposed on the environment.

- Once a digit is dialled,
  - The next digit should be dialled after at least 1 second.
  - Otherwise, a beeping sound is produced until the call initiator replaces the handset.

# Duration Constraint

- Specifies the time interval over which the event acts.
- If you press the button of the handset for less than 15 seconds,
  - It connects to the local operator.
- If you press the button for any duration lasting between 15 to 30 seconds,
  - It connects to the international operator.
- If you keep the button pressed for more than 30 seconds,
  - Then on releasing it would produce the dial tone.


Timing Constraints
Performance Constraints
Behaviorial Constraints
Delay
Deadline
Duration
Delay
Deadline
Duration
RR
SR
RR
SR
RS
SS
RS
SS

# Why Model Timing Constraints?

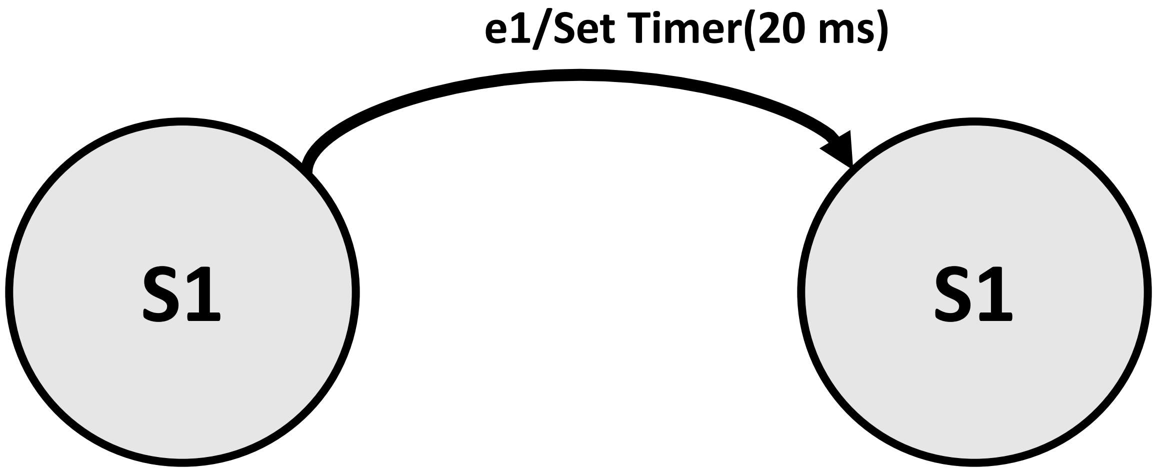
- Modelling time constraints in a system:
  - Can serve as a formal specification of the system.
  - May be used to automatically generate code.
  - Can help to understand real-time behavior.

# Modelling Time Constraints

- Several approaches can be used.
  - We discuss an approach based on FSM proposed by Dasarathy (IEEE TSE, 1985).

- A state is defined in terms of the values assumed by some attributes.
  - The states of an elevator may be denoted in terms of its directions of motion.
  - Values of the attribute direction define the states up, down, and stationery.

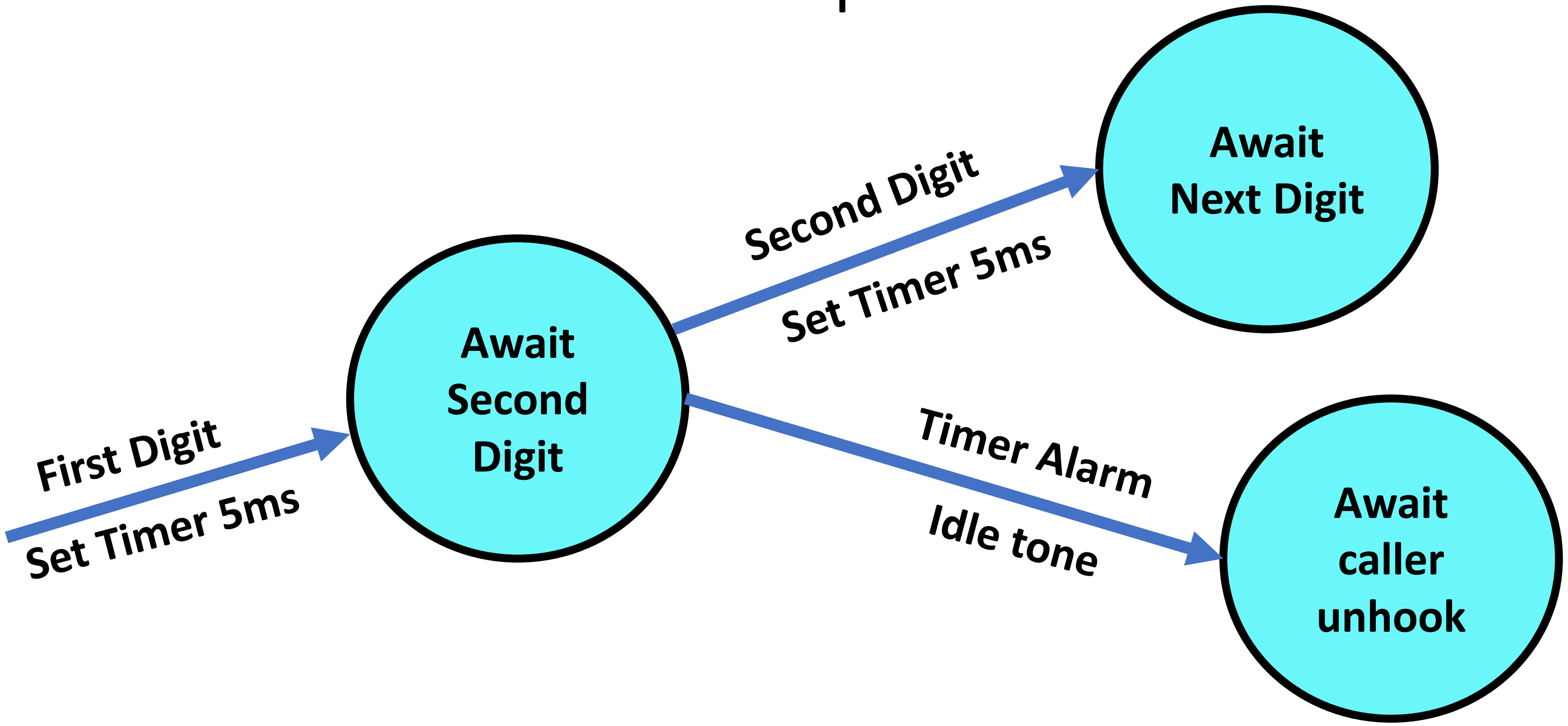
# Finite State Machine(FSM)




- Transition is annotated with:
  - Enabling event
  - Action that would takes place during transition

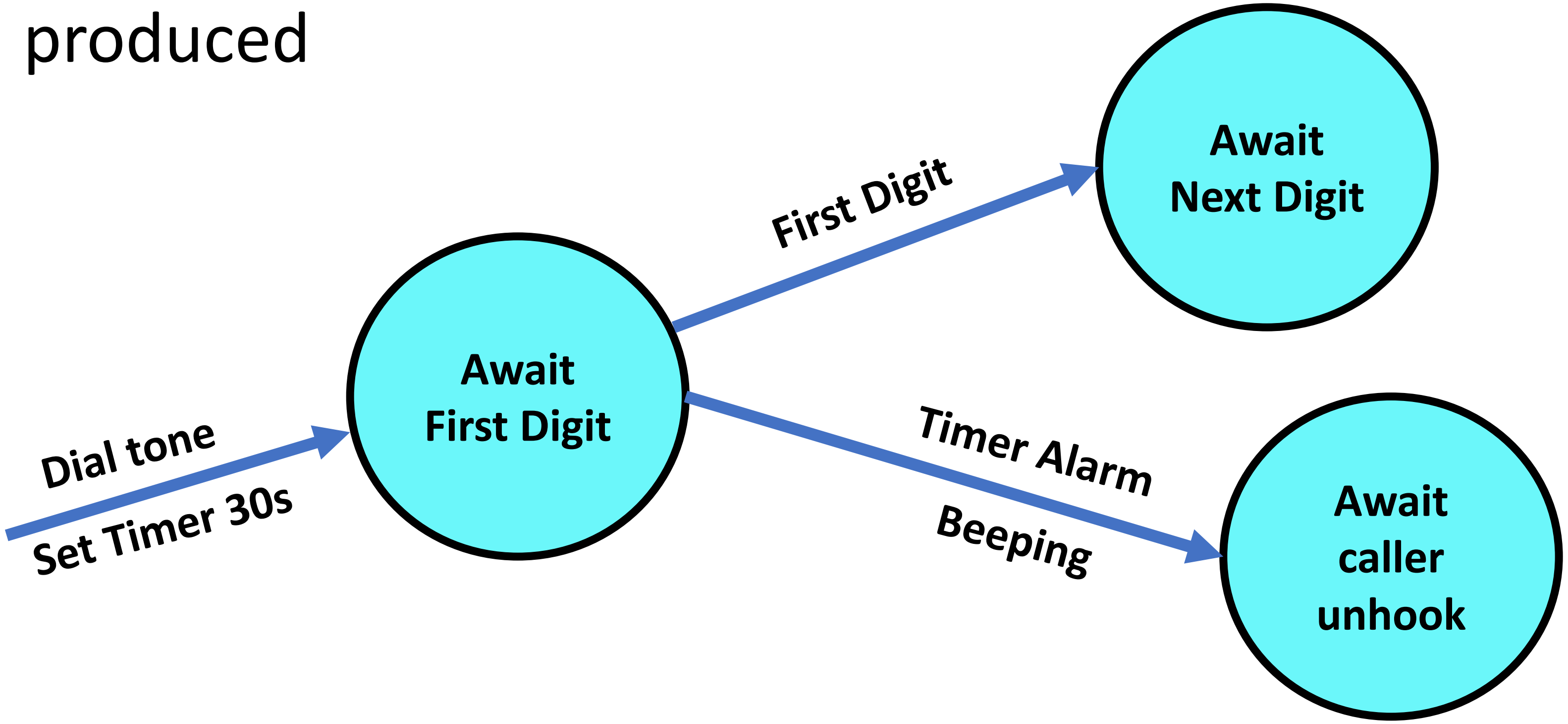
# Model of SS Deadline Constraint

- Once a user completes dialing a digit,
  - He must dial the next digit within the next seconds.
  - Otherwise an idle tone is produced.



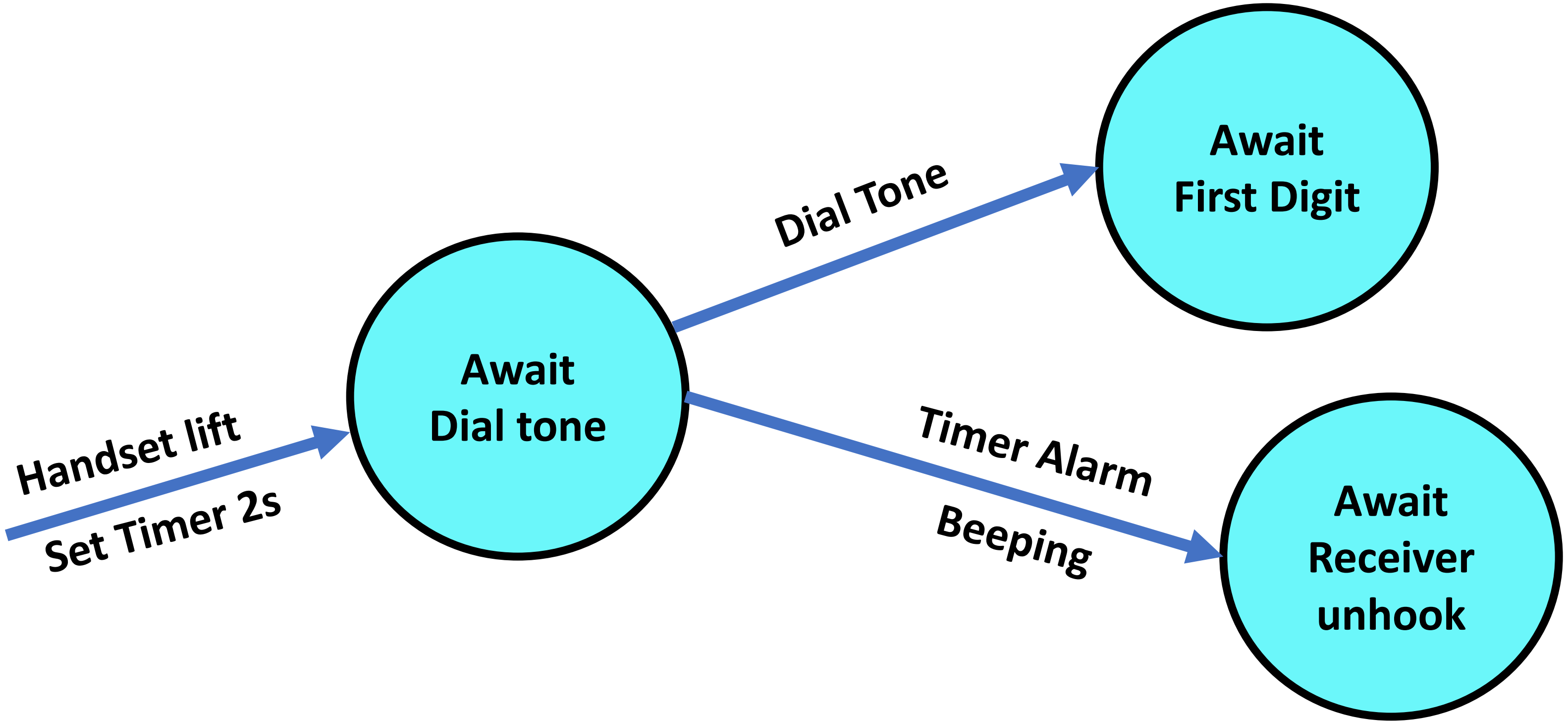
# Model of RS Deadline Constraint

- Once the dial tone appears:
  - The first digit must be dialed within 30 seconds,
  - Otherwise the system enters an idle state and an idle tone is produced



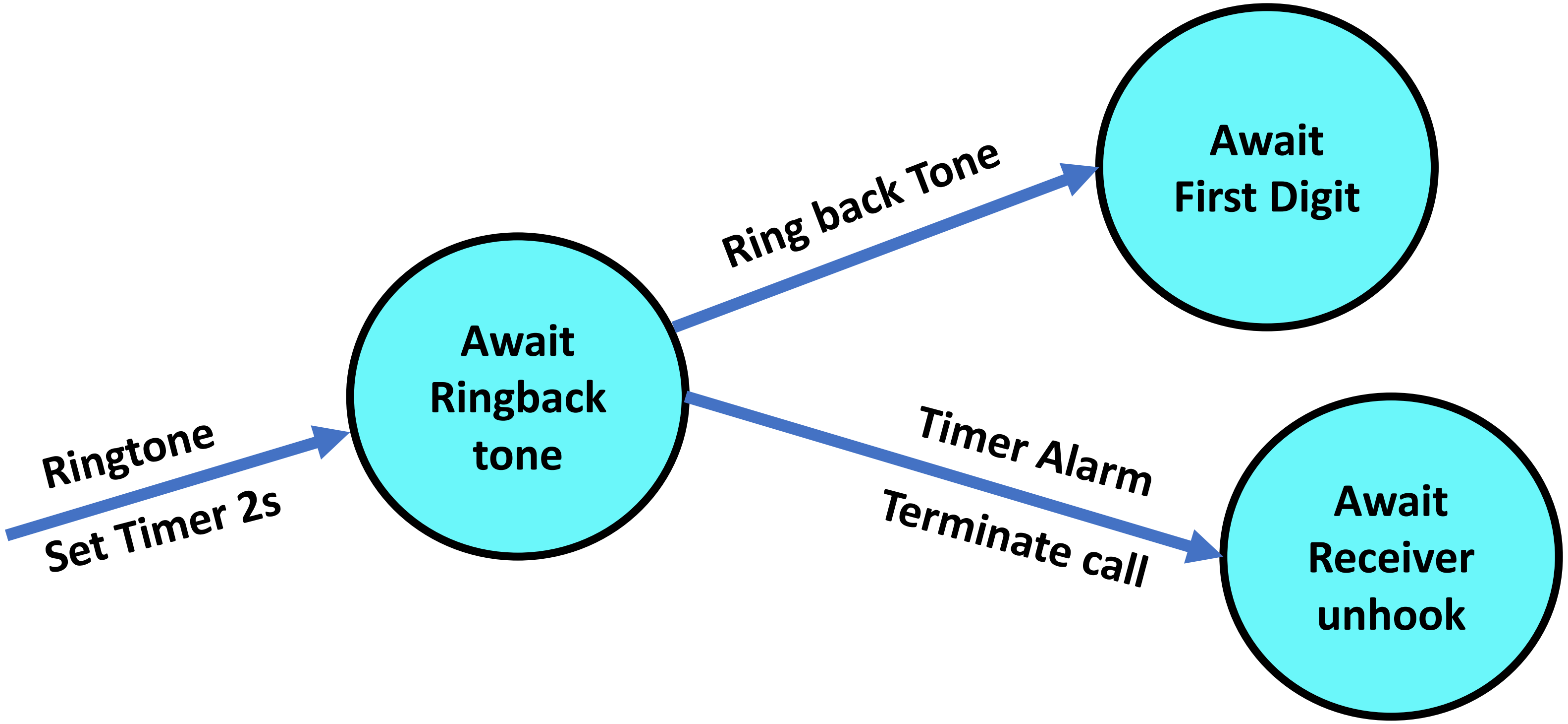
# Model of SR Deadline Constraint

- Once the receiver is lifted from the hand set:
  - The dial tone must be produced by the system within 2 seconds,
  - Otherwise a beeping sound is produced until the handset is replaced. Await



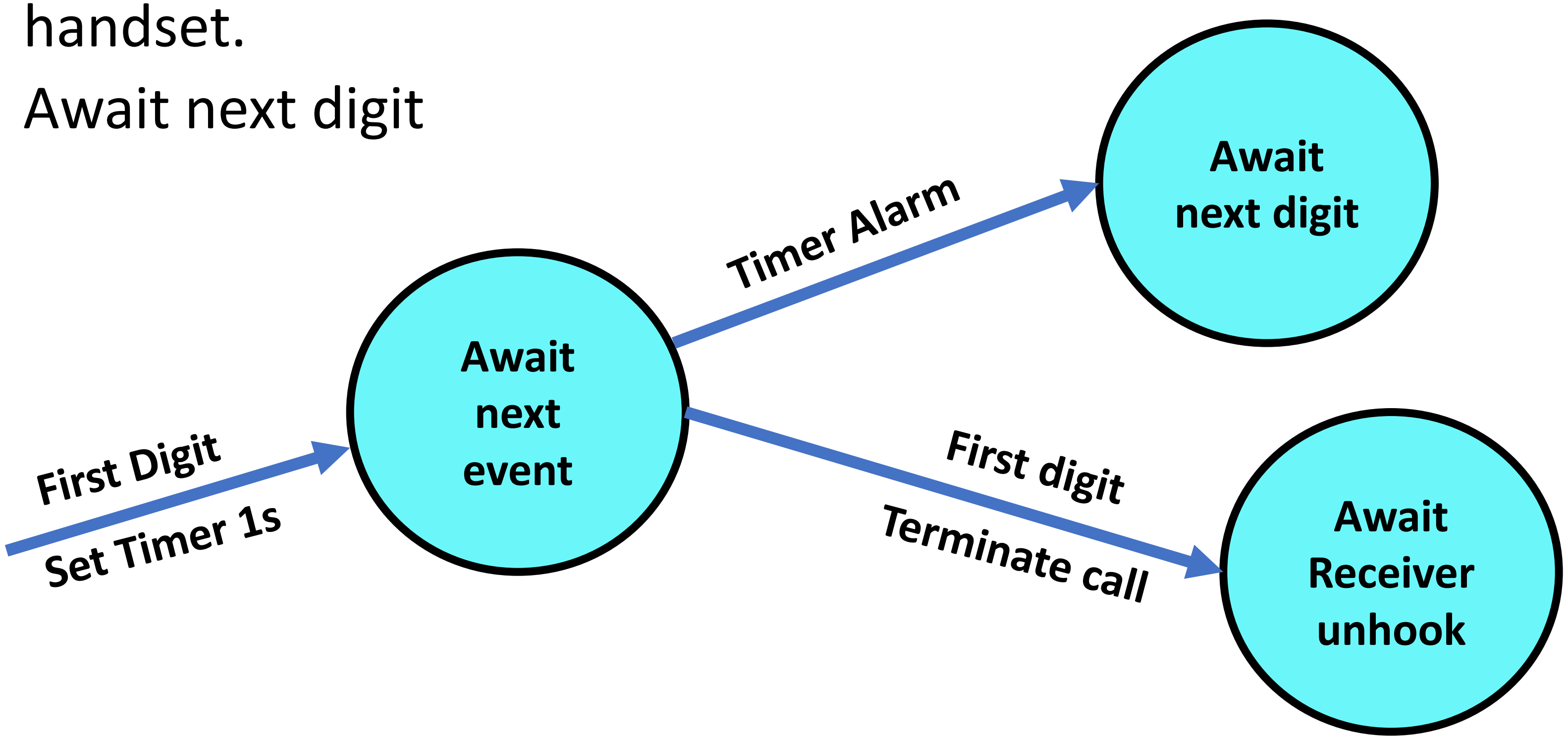
# Model of RR Deadline Constraint

- Once ring tone is given to the callee,
  - Ring back tone must be given to the caller within two seconds,
  - Otherwise the call is terminated.



# Model of Delay Constraint

- Once a digit is dialed,
  - The next digit should be dialed after at least 1 second.
  - Otherwise, a beeping sound is produced until the call initiator replaces the handset.
  - Await next digit



# Duration Constraint Example

- If you press the button of the handset for less than 15 seconds,
  - It connects to the local operator.

- If you press the button for any duration lasting between 15 to 30 seconds,
  - It connects to the international operator.

- If you keep the button pressed for more than 30 seconds,
  - Then on releasing it would produce the dial tone.

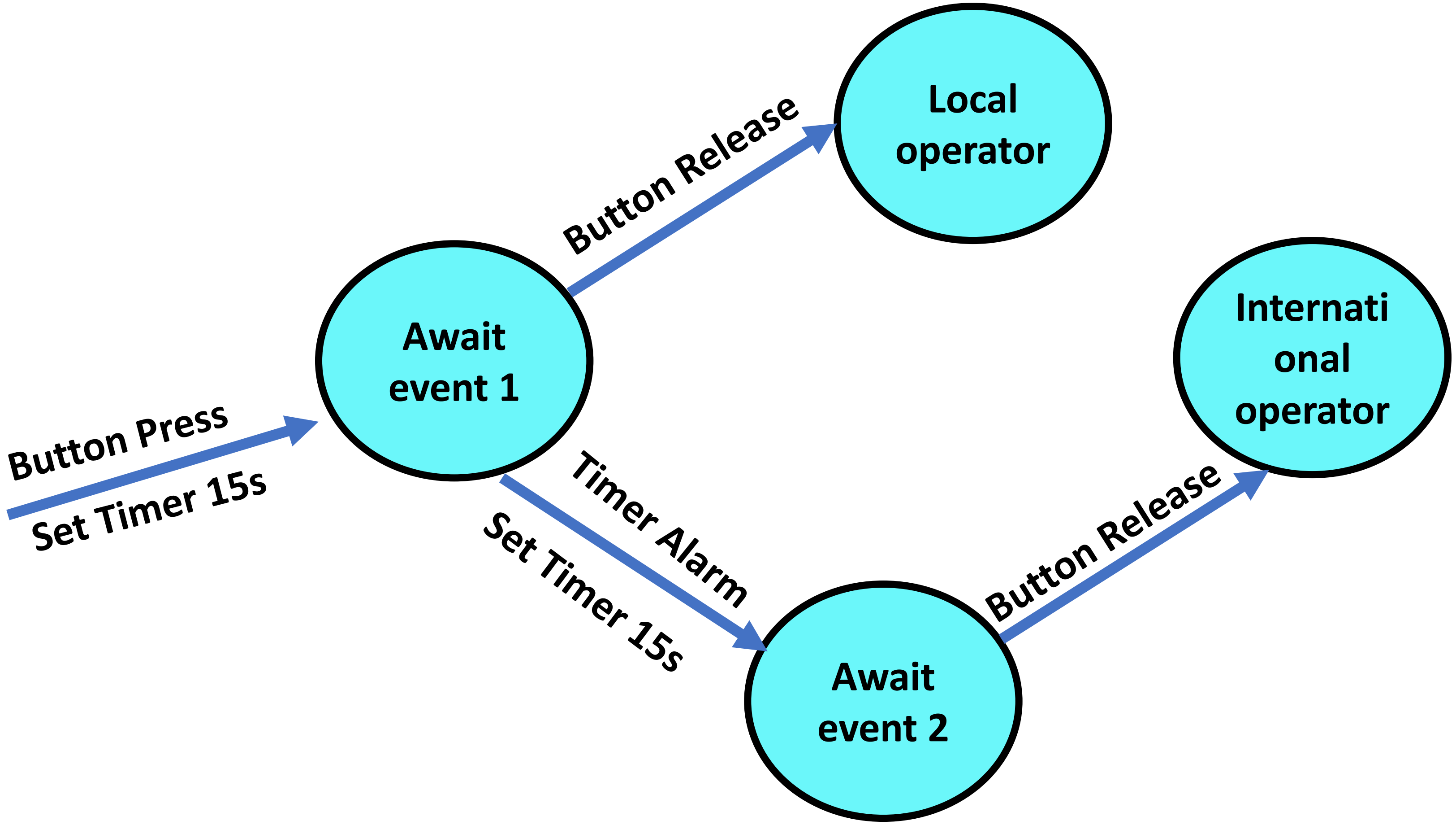
# Model Construction for Duration Example

- To construct the model:

- First identify the deadline and delay constraints.

- The constraints are for which events?

# Model of Duration Constrains Example



# Real Time Tasks

- Real Time tasks get generated due to certain event occurrences
  - Either internal or external events

- Example
  - A task may get generated due to a temperature sensor sensing high level

- When a task gets generated
  - It is said to be released or arrived

# Real Time Task Scheduling

- Essentially refers to the order in which the various tasks are to be executed

- It is the primary means adopted by an operating system to meet task deadlines

- Obviously Scheduler is a very important component of a RTOS

# Few Task Scheduling Technologies

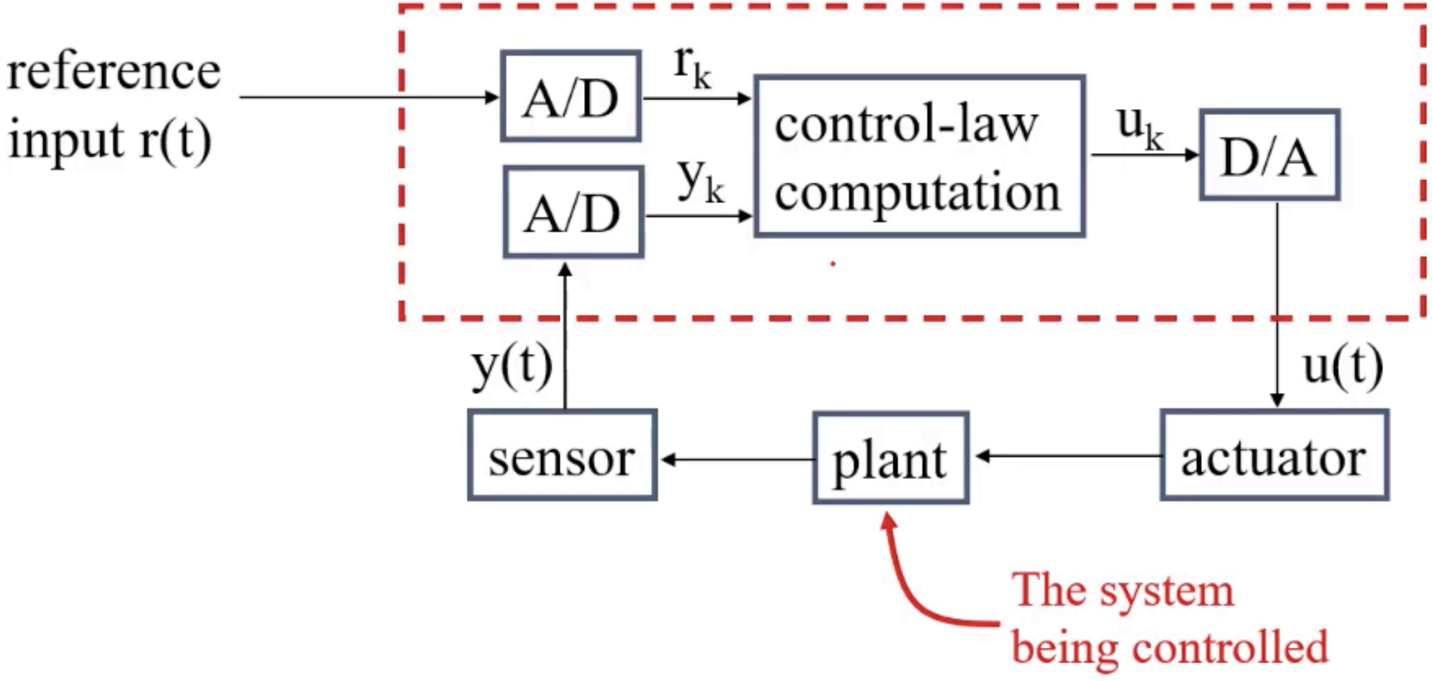
- Valid Schedule
  - At most one task assigned to the processor at a time
  - No task is scheduled before its ready
  - Precedence and resource constraints of all tasks are satisfied
- Feasible Schedule
  - Valid schedule in which all tasks meet their respective time constraints

# Scheduling Terminologies

- Proficient schedule
  - A scheduler S2 is more proficient as compared to another scheduler S1
    - If whatever task that S2 can feasibly schedule so can S1, but not vice versa
- Equally Proficient schedule
  - If task set feasibly scheduled by the one can also be scheduled by other and. vice versa
- Optimal Scheduler
  - An optimal scheduler can feasibly schedule any task set that can be scheduled by any other scheduler

# Examples

- Many Embedded(Real Time) Systems are Control Systems
- A simple one sensor, one actuator control system



# Simple Control System

- Pseudo-code for this system:

```
set timer to interrupt periodically with period T;
at each timer interrupt do
    do analog-to-digital conversion to get y;
    compute control output u;
    output u and do digital-to-analog conversion;
end do
```

- T is called the sampling period. T is a key design choice. Typical range for T: seconds to milliseconds

# Multi-rate Control Systems

- More complicated control systems have multiple sensors and actuators and must support control loops of different rates.
- Example: Helicopter flight controller.

**Do the following in *each* 1/180-sec. cycle:**

validate sensor data and select data source;
if failure, reconfigure the system

***Every sixth* cycle do:**

keyboard input and mode selection;
data normalization and coordinate transformation;
tracking reference update
control laws of the outer pitch-control loop;
control laws of the outer roll-control loop;
control laws of the outer yaw- and collective-control loop

**Every *other* cycle do:**

control laws of the inner pitch-control loop;
control laws of the inner roll- and collective-control loop

Compute the control laws of the inner yaw-control loop;

Output commands;

Carry out built-in test;

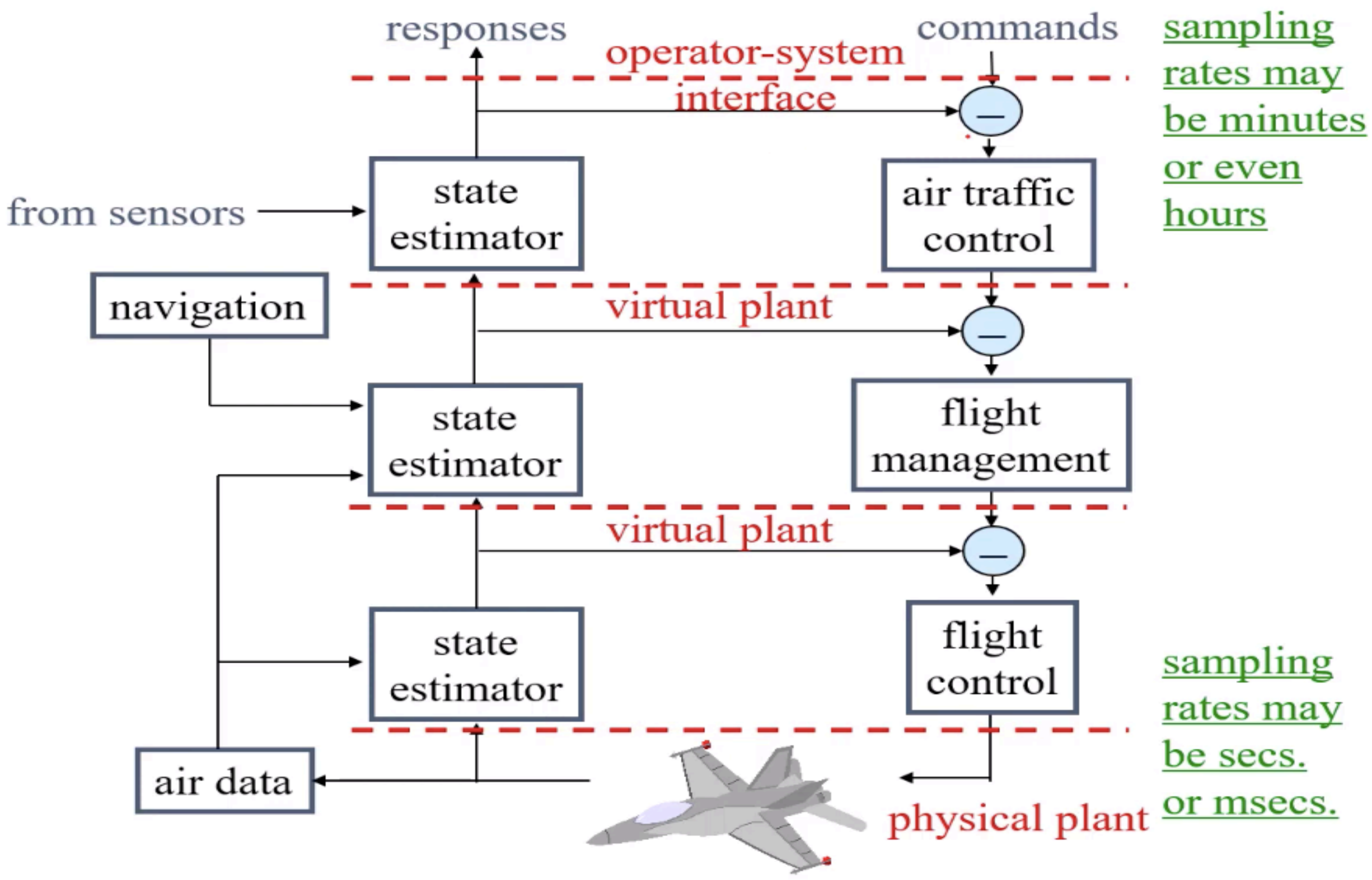
Wait until beginning of the next cycle

Note: Having only harmonic rates simplifies the system.

# Hierarchical Control Systems

**Example:**

**Air Traffic Control System**



# Signal-Processing Systems

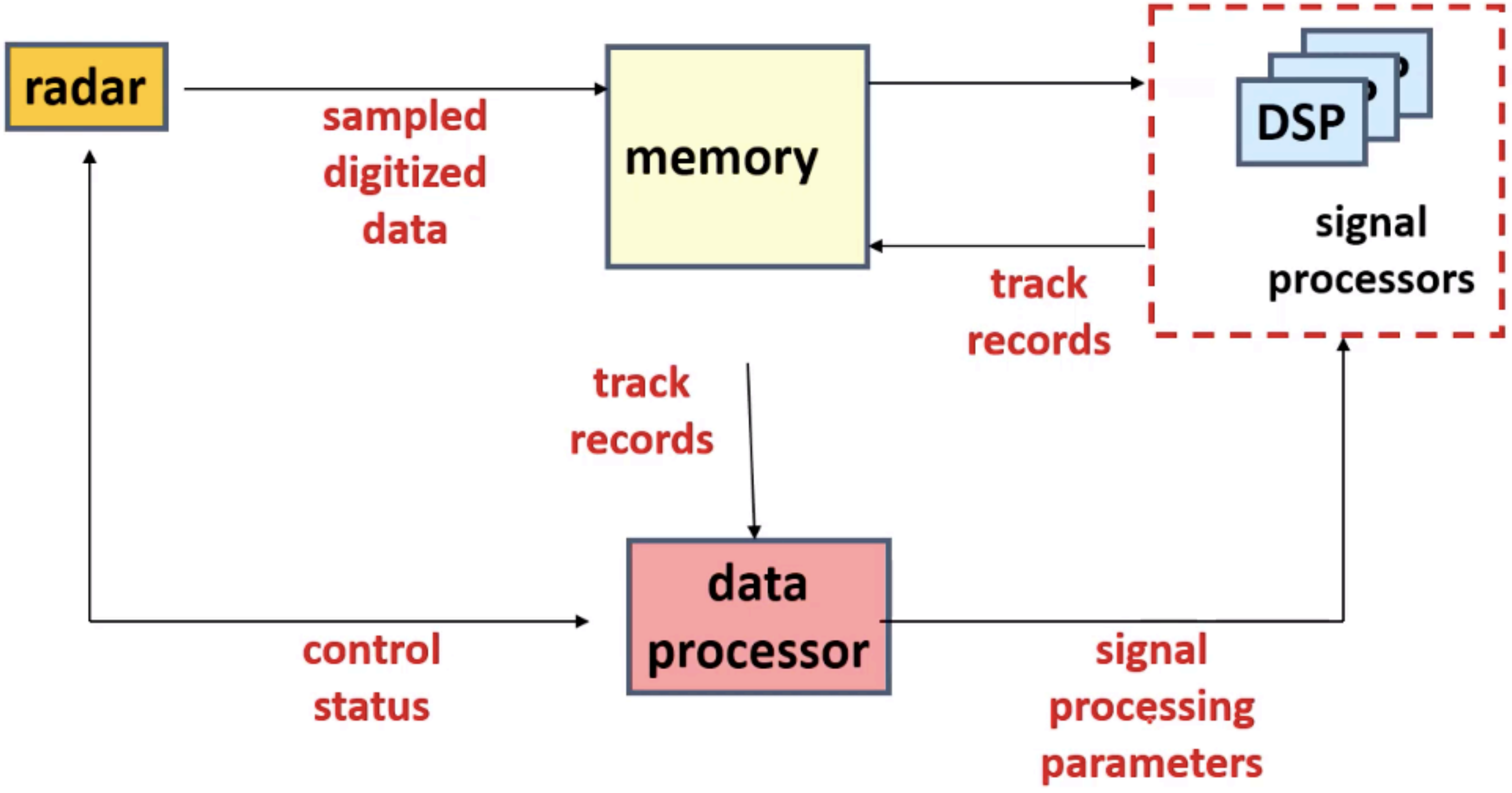
- Signal-processing systems transform data from one form to another.

- Examples:
  - Digital filtering
  - Video and voice compression/decompression.
  - Radar signal processing.

- Response times range from a few milliseconds to a few seconds.

# Signal Processing System

- Example: Radar System



# Reference Model of Embedded Systems

- A reference model focuses on
  - The timing properties and
  - Resource requirements of system components and
  - The way the operating system allocates the available system resources among them.

# Reference Model of Embedded Systems

- According to the reference model, a system is characterized by:
  - A workload model that describes the applications supported by the system
  - A resource model that describes the system resources available to the application
  - Algorithms that define how the application system uses the resources at all times.

# Processors and Resources

- System resources: processors and resources

- Processors – active resources $P_n$
  - Examples: CPUs, transmission lines, disks
- Resources – passive resources $R_m$
  - Examples: memory, sequence number, database locks
  - Examples: computation job shares data with other computations, data guarded by semaphores; communication ACK sequence number

- The elements of a system can be modeled as processors or resources depending on the use of the model.

# Other Types of Dependencies

- Temporal dependency
- AND/OR precedence constraints
- Conditional branches
- Exclusive Access to Resources
- Pipeline relationship of periodic schedules

Unit 1 finished here.
For Queries contact at
saurabh.mishra@kiet.edu