Concepts

Event: Any occurrence that changes the PC non-sequentially – change the flow of control.

***Synchronous* events**: - (synced with code execution)

Occur at predictable times.

***Asynchronous* events**: - (not synced with execution)

Occur at unpredictable times.

***Periodic* events**:

Repeat at precise, regular times.

***Aperiodic* events**:

Repeat, but do not occur at regular periods.

***Sporadic* events**:

Appear infrequently, at irregular times.

Concepts

	Periodic	Aperiodic	Sporadic
Synchronous	Cyclic code Processes scheduled by internal clock	Branch instruction Garbage collection	Error recovery System calls
Asynchronous	Clock-generated interrupt	Regular interrupt with no fixed period	Externally generated interrupt

Determinism:

A system is *deterministic* if for each possible state and input a unique output and next state can be determined.

Concepts

Utilization (time-loading factor – *U*):

The percentage of non-idle processing time.

Utilization (%)	Category	Application
0-25	Excess processing power (CPU wasted)	Various
26-50	Very safe	High-consequence system
51-68	Safe	High-consequence system
69	Theoretical limit *Periodic / independent tasks*	Embedded system
70-82	Questionable	Embedded system
83-99	Dangerous	Embedded system
100+	Overload	Stressed system

Concepts

Real-time system design issues

- Selection of HW and SW “platform”
- Specification and design
- – Including *temporal* behavior
- Analyzing the system design, predicting behavior
- Programming language nuances
- System fault tolerance and reliability
- Design and execution of tests
- Open systems technology/interoperability
- Measuring response times and correcting the design

Example embedded systems

Domain	**Application**
Avionics	Navigation/Displays…
Vehitronics	X-by-wire…
Multimedia	Games, simulators…
Medicine	Implanted devices Robot surgery
Industrial systems	Assembly lines Process plants
Civilian	Elevator control Microwave ovens…

Misconceptions about r/t systems

- Real-time systems are “fast” systems.
- Rate-monotonic (scheduling) analysis has solved the problem.
- We have universal methodologies for r/t systems specification and design.
- There is no need to build a real-time operating system because commercial products exist.
- Real-time systems are about scheduling theory.