SFWR ENG 4A03: Design of Real-Time Systems and Computerized Control Systems

Dr. Mark Lawford
Assistant Professor
Dept. of Computing And Software
Faculty of Engineering
McMaster University

lawford@mcmaster.ca

Outline

- Overview of course organization
- What is a real-time system?
  - hard vs. soft real-time systems
  - distinguishing characteristics
  - Real-Time misconceptions
- Example: Reactor Shutdown System

What is a Real-Time System?

- *Real-Time*: Pertaining to the performance of a computation during the actual time that the related physical process transpires in order that the results of the computation can be used in guiding the physical process. [IEEE Standard Dictionary]
- Correctness depends not only on logical result but also time at which results are produced! [Stankovic'88]
- Current system output depends upon "timed behavior" of input.

Types of Real-Time Requirements:

Hard - timing requirement must always be met (e.g. some reactor trips)

Soft - occasionally missing a deadline does not result in system failure (e.g. RT-video, user interface, data logging, even many control loops!)

Real-time systems are frequently classified as hard or soft real-time systems. A hard real-time system has time-critical deadlines that must be met; otherwise a catastrophic system failure can occur.

In a soft real-time system, it is considered undesirable, but not catastrophic, if deadlines are occasionally missed.

This course is decidedly hard. ;-) Though, as we will see, many systems have both hard and soft RT requirements.

Characteristics of Real-Time Systems:

Real-time systems tend to be:

- Embedded Systems: i.e. a component of a larger hardware/software system.
- Interact with an external "environment" (i.e. the "plant" or system under control)
- Reactive Systems: event-driven and must respond to external stimuli ("events"). System response is typically state dependent.
- Concurrent Processing: many events that need to be processed in parallel.
- Safety Critical

Misconceptions about Real-Time Systems

[Stankovic'88]

- faster hardware implies all deadlines will be met
- real-time computing is merely fast computing
- RT systems are low-level coding done using ad-hoc methods.

Fast is relative. More important that system is "fast enough", deterministic and predictable.

Worst-case response times of interest rather than average-case.

Scheduling theory, software design, formal methods and RTOS are changing things.

Example:
Reactor Shutdown System (SDS)

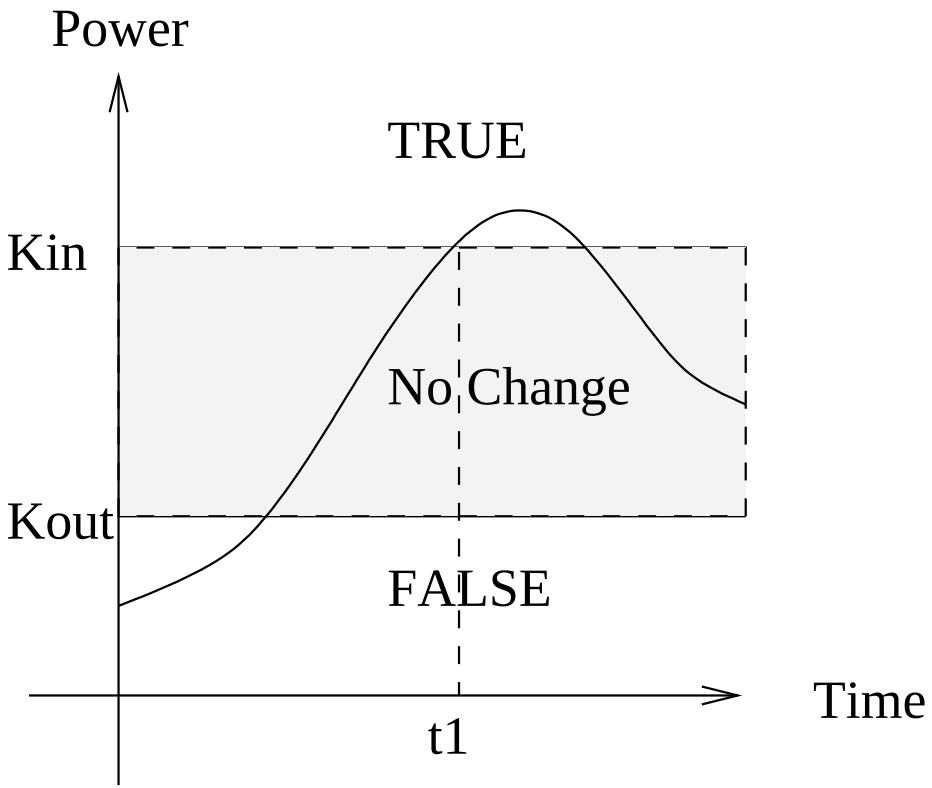
What is an SDS?

- watchdog system that monitors system parameters
- shuts down (trips) reactor if it observes "bad" behavior
- process control is performed a separate Digital Control computer (DCC) - not as critical

Consider simple subsystem: Power Conditioning

- Many sensors have a Power threshold below (or above) which readings are unreliable so it's "conditioned out" for certain Power levels.
- A deadband is used to eliminate sensor "chatter"

General Power Conditioning Function



PwrCond(Prev:bool, Power:posreal):bool =

$Power \leq Kout$	$Kout < Power < Kin$	$Power \geq Kin$
$FALSE$	$Prev$	$TRUE$

When Power:

- drops below $Kout$, sensor is unreliable so it's "conditioned out" ($PwrCond = FALSE$).
- exceeds Kin, the sensor is "conditioned in" and is used to evaluate the system.
- is between $Kout$ and Kin, the value of $PwrCond$ is left unchanged by setting it to its previous value, $Prev$.

E.g. For the graph of $Power$ above, $PwrCond$ would start out FALSE, then become TRUE at time $t1$ and remain TRUE.

Real-time SDS Requirement

An example of a simple reactor trip requirement:

- monitor plant parameters (e.g. Primary Heat Transport Pressure & Reactor Power) using sensors & A/D conversion
- if parameters exceed set-points in particular way, shutdown (trip) the reactor
- old hardware implementation to be replaced by microprocessor based system with 100ms cycle time

Delayed Trip System

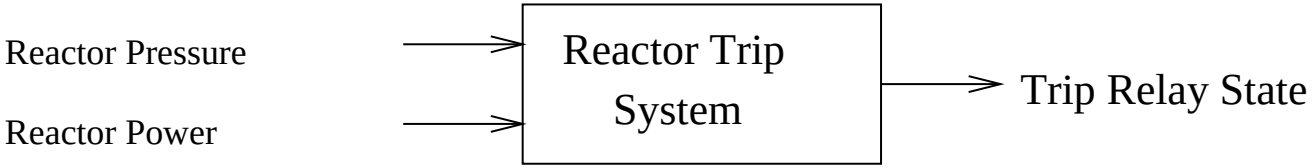


Figure 1: Block diagram for DTS

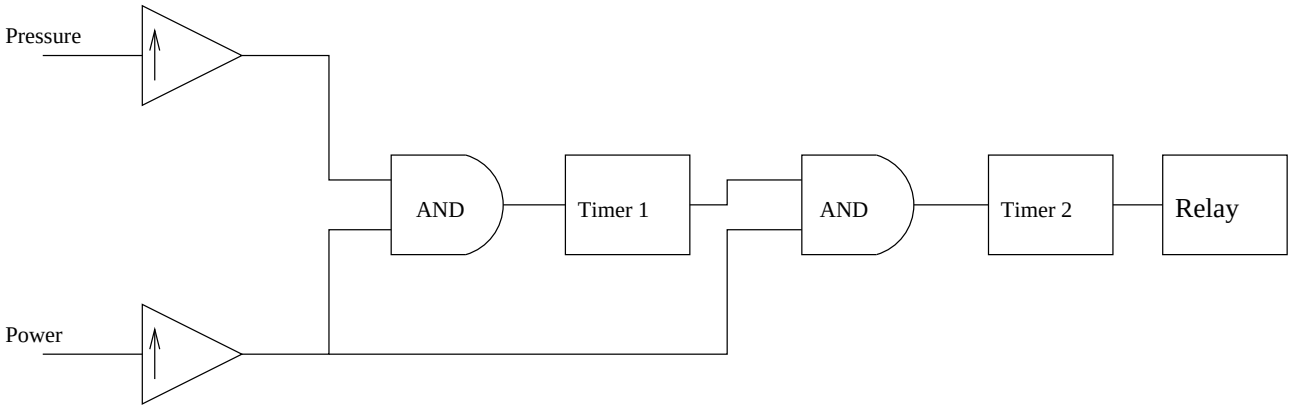
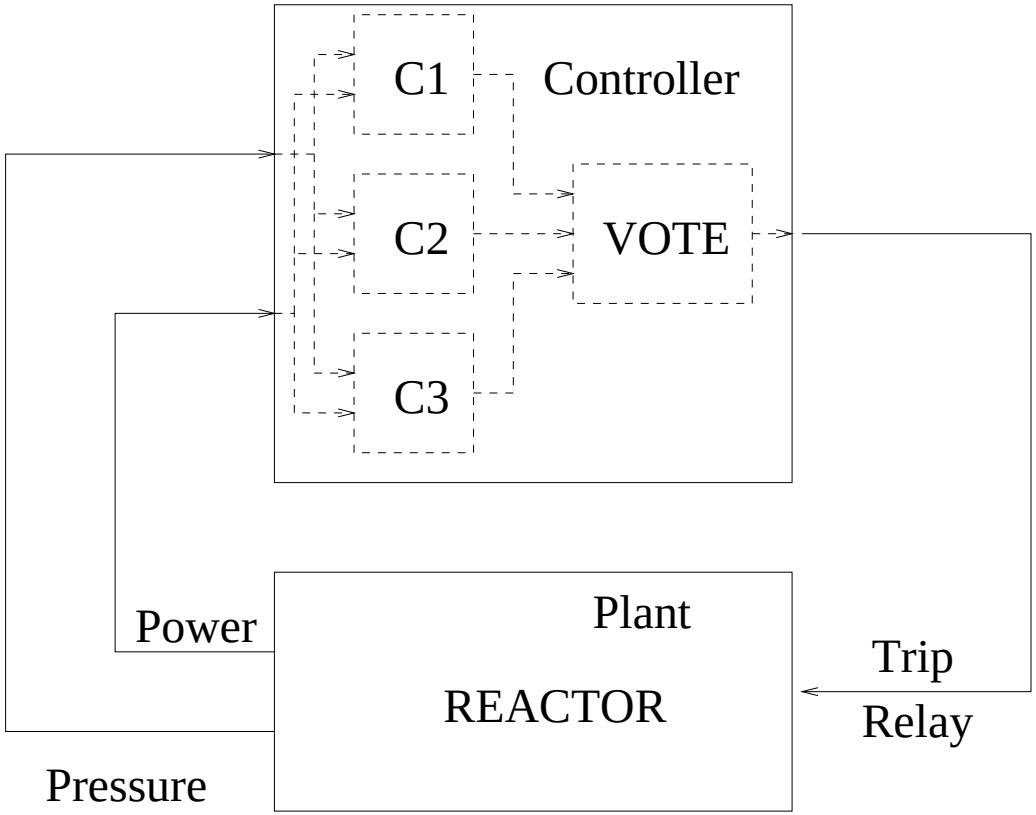


Figure 2: Analog implementation of DTS

Concurrent Real-Time Systems

Redundant systems run concurrently and perform majority vote to decide when to shutdown system.

Does this real-time control system do what we want?



SDS Safety/Performance Considerations:

- Check for short circuits/sensor failures
- Use dead-band to eliminate "chatter"
- Power dependent set points increase operating margin
- "Condition out" sensor in unreliable operating region
- Digital trip output uses "-ve logic" (fail-safe in power loss)

Additional SDS Considerations:

- Use multiple sensors to improve reliability
- Some use of redundant variables to reduce risk of memory errors
- Soft real-time requirements:
  - operator display/interface
  - data-logging to external system via serial communication
- Many other concurrent tasks associate with other trips that must meet hard real-time deadlines.