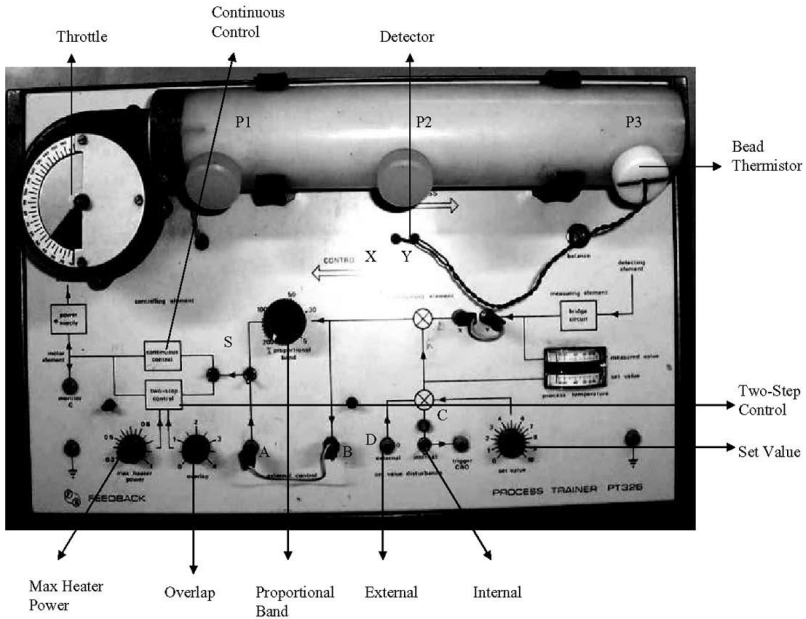
Real-Time Systems Testing Methods: Plant Models

- What is a plant model?
- A plant model is a dynamic model expressed mathematically with a set of differential equations that describe the dynamic behaviour of the process
- How do we develop a dynamic model?
  1. By using principles of underlying physics
  2. By testing a prototype of the device and performing measurements of response to various inputs and using the collected data to construct an analytical model

Modeling the PT326 Process Trainer: plant used in our labs

Solution: the PT326 unit falls in the category of **thermal processes**.
The principles that governs the behavior of the thermal process in PT326 apparatus is the **balance of heat energy**!!



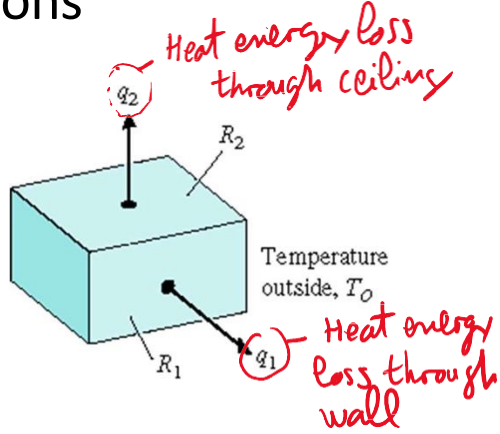
ENGG4420: Developed by Radu Muresan, F22

Heat and Fluid-Flow Models

- The dynamic models of temperature control systems involve the flow and storage of heat energy
- There are 3 different ways of heat flow from one object to another:
  - Conduction, convection and radiation
- Heat energy flows through a substance due to transfer of heat:
  - q = …
- The temperature of the substance is affected by the heat flow as:
  - dT/dt = …

Example of Heat Flow Calculations

- We consider a room which has two sides un-insulated.
- Derive the system model that produces the temperature in the room

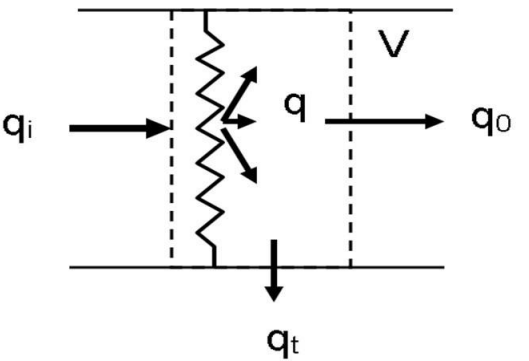


Specific Heat and Thermal Conductivity

- These are material properties that are given in tables and can be used to calculate thermal capacity C and thermal resistance R for a substance of interest
- c_v: specific heat is the heat capacity per unit mass of material. This can be used to calculate C as:
  - C = …
- k: thermal conductivity is the property of the material to conduct heat. R is related to k by the following relation:
  - 1/R = …

PT326 System Model

- We characterize a small volume around the heating element – the heated air from around the element will flow in the tube.
- What are the heat flow sources in and out of this volume?
- The rate of heat accumulated in volume V is:
  - q_a = …



PT326 Apparatus, Model Equations

- We use two expressions for heat accumulation in volume V in order to get the model for the temperature rise ΔT as a function of the heat flow generated by the apparatus heat element resistor R
- Assumptions: the only heat accumulated in volume V is due to the heat transferred from the heating element, the other components are negligible
- We follow up with equations (8), (9) and (10) -- See Lecture Notes -- to derive the model equation (11) for a small rise of temperature ΔT in volume V as:
- (11):

Transfer Function of the Process Trainer Model

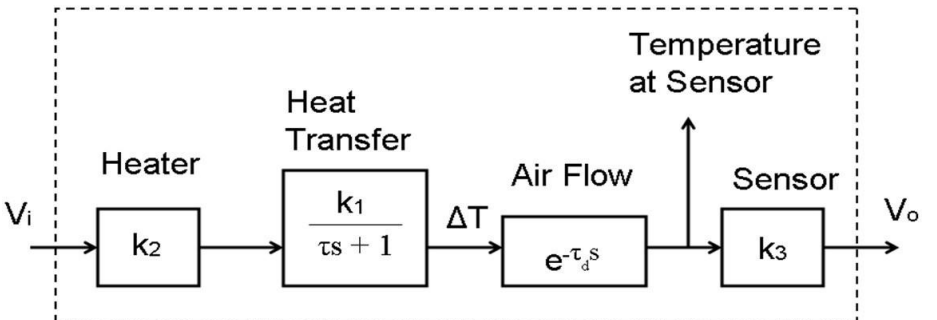
- Taking the Laplace transform of our derived equation (11) we get the transfer function of the system in the frequency domain as: transfer function = output/input □
- Equations (12), (13):
  - (12):

Transfer Function of the Process Trainer Model with Transport Delay

- Note that the temperature sensor is placed at a distance down the tube from the heat source.
- This will introduce transport delay for our model if we consider the transfer function as:
  (sensor voltage)/(heat source voltage) → equation (14):

- (14):

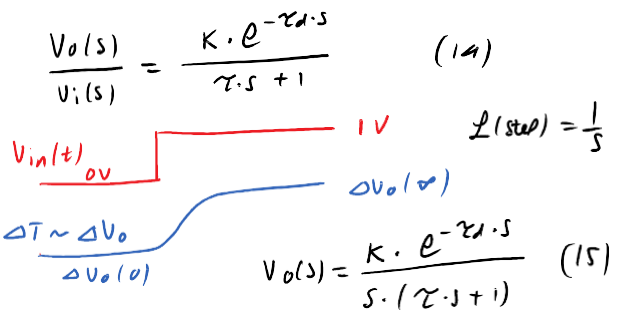
System Model Block Diagram of the PT326 Apparatus



$$\frac{ke^{-\tau_d s}}{\tau s + 1}$$

How Do We Determine τ_d and τ

- Constants τd and τ that are part of our P326 apparatus model and are important for a real-time system designer for the purpose of defining correct sampling intervals
- By using system step response we can measure these parameters
- In the laboratory manual we provide these parameters for different openings of the blower cover



Dynamic Models of Mechanical Systems

- The Newton's law equation of motion is key for developing a mathematical model of a mechanical system
  - **$F = m \cdot a$**
- Application of Newton's law involves:
  - Define convenient coordinates to account for the body's motion
  - Determine the forces on the body using the free-body diagram
  - Write the equations of motion

Equation Model Example: Cruise Control

- The transfer function for the cruise control system is:
  TF = (velocity of the car v)/(engine force u)
- V(s)/U(s) = ?

Models of Electrical Circuits

- The basic equations of electric circuits are based on Kirchoff's laws:
  - Kirchoff's current law (KCL) states that the sum of currents leaving a junction or node equals the algebraic sum of currents entering the node.
  - Kirchoff's voltage law (KVL) states that the algebraic sum of all voltages taken around a closed path in a circuit is zero.

Models of Electromechanical Systems

- Electric current and magnetic fields interact in two ways that are important in the operation of electromechanical actuators and sensors:
  a. LAW OF MOTORS: If a current of i amperes [A] in a conductor of length l meters is arranged at right angles in a magnetic field of B tesla [T], then there is a force on the conductor at right angles to the plane formed by i and B, with magnitude of:
     - $F = B \cdot l \cdot i$ [Newton].
  b. LAW OF GENERATORS: If a conductor of length l meters is moving in a magnetic field of B Tesla at a velocity of v [m/s] at mutually right angles, an electric voltage is established across the conductor with magnitude:
     - $e(t) = B \cdot l \cdot v$ [Volts].

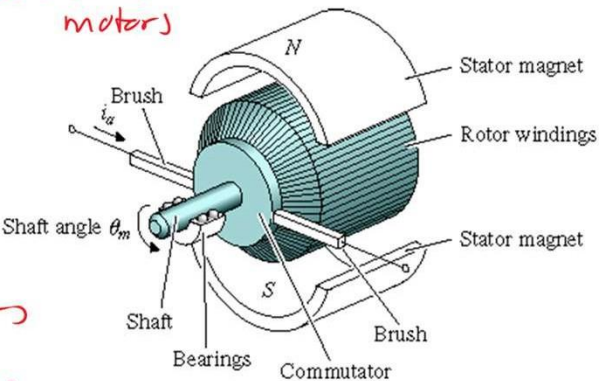
Modeling Example: DC Motor

$T = K_t \cdot i_a \quad (1)$ → Law of motors

$e = K_e \cdot \dot{\theta}_m \quad (2)$

Law of generators

K_t, K_e – motor constants



The Transfer Function of the Motor

- TF = (motor angular rotation θ_m)/(armature voltage v_a)
- $\Theta_m(s)/V_a(s)$ = …

Assignments

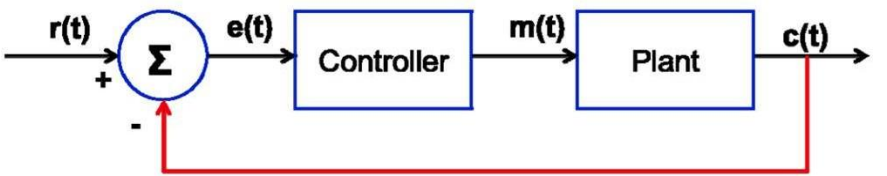
- Review all examples in lecture notes and check the homework given in lecture notes.
- Review the model for mechanical and electromechanical systems (only as presented in the power points presentations during the class lectures)

Hard Real-Time Constraints Tasks, Implementation Methods

- In our hot-air blower system example the control task and the clock/calendar task are hard-real time tasks due to their tight time constraints (*ms* order)
- In this section, we introduce general considerations related to the implementation of hard-real time control task
- We shall consider:
  - The digital form of a PID control algorithm, implementation details
  - Timing requirements and choice of sampling interval

The Diagram of a Continuous Control System with Feedback

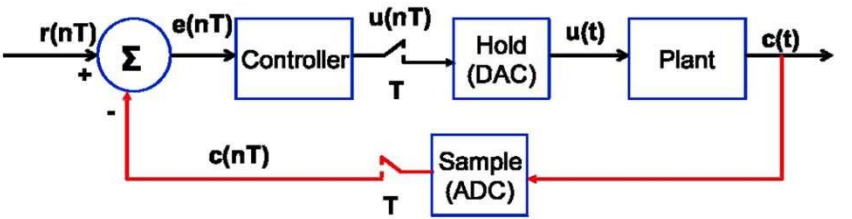
- In this diagram here we show a simplified block diagram of the feedback part of a system such as the temperature control or air blower control applicable to our hot air unit.



r(t) – set point; c(t) – controlled variable; e(t) = r(t) – c(t) –> error; m(t) – manipulated variable.

Sampled Feedback Control System, Block Diagram

- A digitization process is applied in order to interface to the computer controlling a plant.
- The digital computer system will sample the plant to generate the input data image and will generate an output data image to interface with the plant.



The symbols c(nT), r(nT), e(nT), and u(nT) are the sampled values of c(t), r(t), e(t), and u(t), respectively, at sample times nT where n is an integer and T is the sampling interval.

PID Controller

- For a wide range of industrial processes, the PID controller achieves a good performance in controlling the system.
- For this reason, the PID and PI algorithms are widely used.

- $m(t) =$

- The PID controller contains three main terms:
  - proportional – characterized by the proportional gain K_p
  - Integral – characterized by the integral action time T_i
  - Differential – characterized by the derivative action time T_d
- See Lecture Notes for further reading on PID control.

Direct Digital Control, The PID Equations

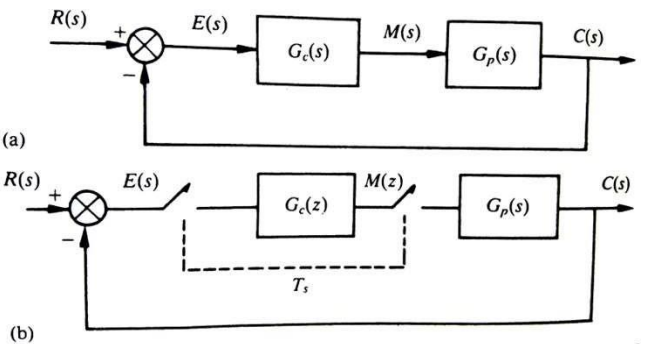
- The differential equation presented is the time domain representation of the controller.
- Applying Laplace transform we can derive the equivalent frequency domain representation as:

- $G_c(s) = M(s)/E(s) =$

- Both the time domain t and frequency domain s representations are continuous representations

Frequency Domain Control System, Block Diagram

- In the frequency domain the overall system of controller and plant can be represented by a continuous form and discrete form.



The Discrete Representation of the PID Controller

- Both the time domain t and frequency domain s representations are continuous representations
- To implement the controller as a real-time task we need a digital algorithm.
- For this we need to convert from the continuous form into the discrete form of the controller representation
- There are several methods to do this but for the purpose of this course we present only the use of first-order finite differences.

First Order Finite Difference Method for PID Algorithm Conversion

- In the time domain m(t) version of the controller we replace the differential and integral terms by their first order discrete equivalents => the discrete equation for m(n)
- Discrete equivalents …

- m(n) =

- Where m(n) represents the value of m at some time interval $n\Delta t$

Discrete PID Algorithm For the Controller: Hard Real Time Periodic Algorithm

- The discrete equation m(n) for the PID controller can be easily transformed into a digital program algorithm as follows:

$$s(n) = s(n-1) + e(n);$$
$$m(n) = K_p \cdot e(n) + K_i \cdot s(n) + K_d \cdot [e(n) - e(n-1)];$$

Where: m(n) =m(nΔt); e(n) = e(nΔt); and Δt = T_s is the sample period
K_i and K_d are new parameters defined as:

- Ki = Kp(Ts/Ti); and Kd = Kp(Td/Ts);

Basic Programming Statements: Hard Time Periodic PID Digital Algorithm

- In implementing this algorithm, we could assume that:
  - the plant output is obtained by using an ADC to sample and convert the output signal,
  - the actuator control signal is outputted through a DAC to the actuator; and
  - procedures ADC and DAC are available to read the ADC and to send values to the DAC.

Summary of DDC Implementation of Controllers

- Can be implemented as real-time embedded applications
- A continuous time representation of a controller can be translated into a discrete equivalent using Z transform analysis
- Digital controllers use sampled and quantized signals
- A controller that operates on signals that are sampled but not quantized is called DISCRETE
- A controller that operates on signals that are sampled and then quantized is called DIGITAL
- The digital computer function is to calculate the next control value
- The next control value will be converted by a DAC into analog signal

Summary of DDC Implementation of Controllers

- The use of detailed plant models allows a wide variety of methods to be used in the design of the controller
- Specifically, two types of representation are common:
  - State-space representation of the difference equations, and
  - Transfer function in z^{-1}
- If controller is represented in difference equation form it can be programmed directly as a digital algorithm
- If the controller is given as a transfer function in z domain, then we can use some techniques to convert it into difference equations
  - Direct method 1 and method 2 – See Lecture Notes.
  - Cascade implementation and parallel implementation – See Lecture Notes.

Practical Considerations for the Implementation of the Real-Time PID Controller Task

The basic PID algorithm

$$s(n) = s(n-1) + e(n);$$
$$m(n) = K_p \cdot e(n) + K_i \cdot s(n) + K_d \cdot [e(n) - e(n-1)];$$

1. Coefficients K_i and K_d are dependent of T_s (sampling period)
   - For correct operation we should consider the sampling period T_s as a variable
2. The control task is a cyclic hard real-time task -- what synchronization method between the program and the plant can be used?
   - Polling; external interrupts; ballast coding; and real-time clock signals.

Practical Considerations for the Implementation of the Real-Time PID Controller Task

3. Bumpless transfer – this relates to the requirement that manual/automatic modes change over be made smoothly
   - Common methods are:
     - Tracking of operator settings
     - Velocity algorithm (most common) – the velocity algorithm gives the change in the value of the manipulated variable at each sample time rather than the absolute value of the variable:
       - $\Delta m = m(n) - m(n-1)$ …

Practical Considerations for the Implementation of the Real-Time PID Controller Task

4. Saturation and integral action wind-up – relates to the fact that the manipulated variable m(n) is limited by physical constraints and the program should take these constraints in consideration to avoid integral term build up due to error that cannot be compensated.
   - Common solutions:
     - Use the velocity algorithm
     - Introduce limits on the summation term when the actuator's physical limits are encountered.

Practical Considerations for the Implementation of the Real-Time PID Controller Task

5. Tuning – relates to the design of the parameters K_p, T_i and T_d that are part of the PID controller task (See lab manual)
   - Tuning methods:
     - All tuning methods are based on simple plant measurements and on the assumption that the plant to be controlled can be model by a first order transfer function
     - Ziegler-Nichols rules based on open loop step: tuning gives some values for Kp, Ti and Td as function of transfer characteristic parameters of the plant; specifically, R (the slope of the transfer function), L (plant lag)
     - Ziegler-Nichols rules based on the closed feedback loop: tuning gives some values for Kp, Ti and Td in terms of the close loop measurements; specifically, loop gain Km at steady state oscillations, and period of steady state oscillations Tp

Practical Considerations for the Implementation of the Real-Time PID Controller Task

6. Choice of sampling interval – the upper and lower boundaries for T_s relate to the feedback loop destabilization and computer resolution, respectively.
   - We give some common empirical rules that can be applied:

   a. Dominant plant time constant, Tp: …

   b. Assumption of Ziegler-Nichols plant model – the plant model is given as first order plant model: …

   c. Closed-loop performance requirements– the closed-loop system is required to have a settling time of Tss: …

Practical Considerations for the Implementation of the Real-Time PID Controller Task

7. Plant input and output – we need to consider that the sampled signals used to calculate our error values are noisy signals and might also need scaling and conditioning.
   - Divide the software into segments where:
     - The plant input is responsible for filtering, scaling and calibration operations.
     - The plant output module converts the output image into the appropriate form required to drive the actuator.

Practical Considerations for the Implementation of the Real-Time PID Controller Task

8. Computational delay -- is the time difference between sampling the plant output and changing the value of the actuator
   - It is a function of: (input/output speed of the processor) + (type and amount of computation) + (controller operation ordering)
   - Controller ordering 1:
     1. read plant input data;
     2. calculate control output;
     3. send output to actuator
   - Controller ordering 2:
     1. send to plant the control value for sample interval (k-1);
     2. read input data for sample interval k;
     3. calculate control output for k^{th} sample interval

Practical Considerations for the Implementation of the Real-Time PID Controller Task

9. For positional and velocity algorithms we can use the trapezoid rule instead of the rectangular rule to approximate the integral summation term.
10. Differentiation normally will accentuate noise and data errors => some form of smoothing (filtering) is needed
    - Average over several samples – four-point central difference method.
    - m(n) = …

Real-Time Control Task Implementation

- The pseudo code algorithm for the real time task is =>
- HOMEWORK!
  - Improve this pseudo code by including all practical consideration elements presented!!

```
TASK RealTimeControl;
  [* Declarations Block];
BEGIN
  [* Initialization Block];
  LOOP
    [* Synchronization Block];
    [* Get Plant Data Block];
    [* Control Calculation Block];
    [* Exit Condition check];
    [* Put Control Data to Plant];
  END LOOP
END TASK
```

RTOS Topics

- So far, we argued the need for a RTOS support when implementing real-time control applications. However, the real-time applications are common in many areas of computing – see our examples in lecture 1.
- RTOS Topics
  - Benchmarks for RTOS
  - uC/OS-III Basics
  - FreeRTOS
  - DIdactic C Kernel (DICK) – example small kernel design

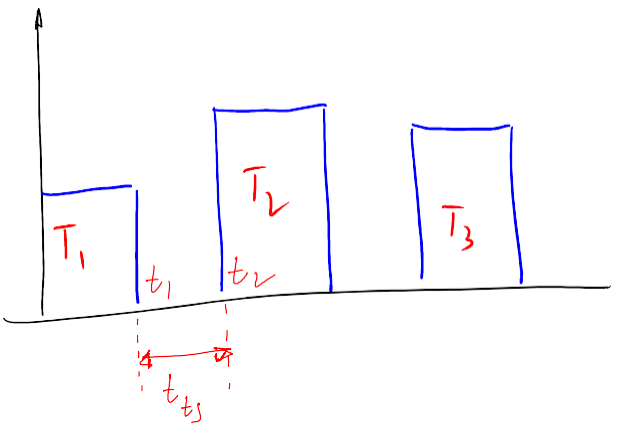
Benchmarking Real-Time Systems

- Synthetic benchmarks are created based on statistical distribution of various instructions in an average program
- Older real-time systems benchmark are Rhealstone metric and Threadmetric. We look to Realstone metric to understand some parameters used to compare the performance of RTOSs
- In Realstone metric, six parameters of real-time systems are considered:
  - 1) task switching time; 2) task preemption time; 3) interrupt latency time; 4) semaphore shuffling time; 5) unbounded priority inversion time; 6) datagram throughput time

Task Switching Time: t_{ts}

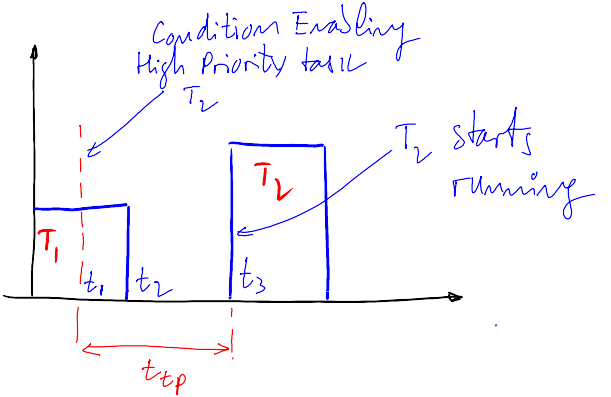
- Task switching time is defined as the time it takes for one context switch among equal priority tasks
- $t_{ts} = t_2 - t_1$

- Task switching time is determined by the efficiency of the kernel data structure



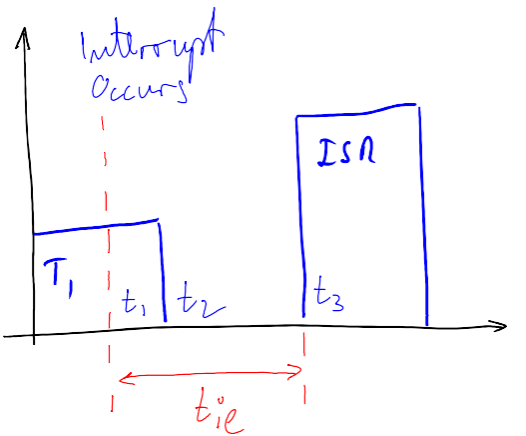
Task Preemption Time: t_{tp}

- Task preemption time is defined as the time it takes to start the execution of a higher priority task (compared to the currently running task) after the condition enabling the task occurred.
- $t_{tp} = t_3 - t_1$
- t_{tp} has 3 parameters …



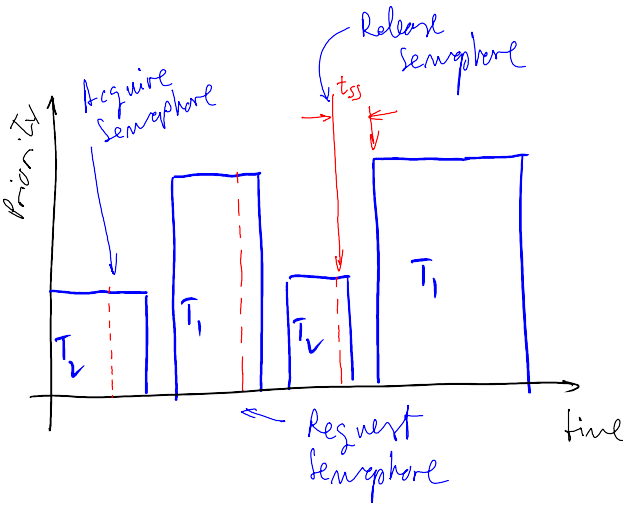
Interrupt Latency Time: t_{il}

- Interrupt latency time is defined as the time it takes to start the execution of the required ISR after an interrupt occurs
- $t_{il} = t_3 - t_1$
- t_{il} consists of 4 components: …



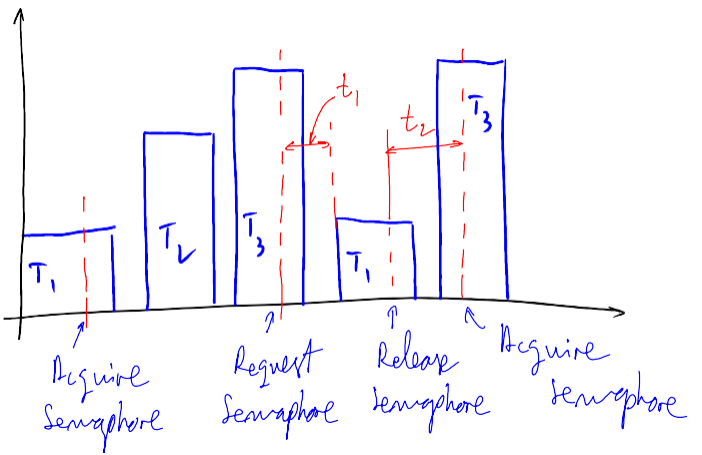
Semaphore Shuffling Time: t_{ss}

- Semaphore shuffling time is defined as the time that elapses between a lower priority task releasing a semaphore and a higher priority task to start running
- t_{ss} =



Unbounded Priority Inversion Time: t_{up}

- Unbounded priority inversion time is computed as the time it takes for the OS to recognize priority inversion (t_1) and run the task holding the resource and start T3 after T1 completes (t_2)
- $t_{up} = t_1 + t_2$



Datagram Throughput Time: t_{dt}

- Datagram throughput time indicates the number of kilobytes of data that can be transferred between two tasks without using shared memory or pointers.
  - This parameter is designed to measure the efficiency of data structures handling message passing primitives that are part of the RTOS

The Rhealstone Metric

- The Rhealstone metric is computed as a weighted average of the identified parameters:
- Rhealstone metric = $a1*t_{ts} + a2*t_{tp} + a3*t_{il} + a4*t_{ss} + a5*t_{up} + a6*t_{dp}$;
- Where the parameters a1 to a6 are empirically determined
- The Rhealstone benchmark's figure of merit is that it characterizes the kernel-hardware combination of a system …
- Drawbacks of Rhealstone metrics are:
  - The ability of a system to meat deadlines of application is not measured
  - The six measurements are somehow ad hoc, no practical justification

Other Metrics

- Interrupt processing overhead – for this we calculate $t_{20,000}$ which represents the time to complete a task when 20,000 interrupts occur per second
  - $I_{20,000} = (t_{20,000} – t_0)/t_0$
  - Where t0 is the time to complete a certain task when there are no interrupts
- Tridimensional measure TM considers 3 factors that affect the performance of a real-time system:
  - $TM = (MIPS1*MIPS2*NIOPS)^{1/3}$
  - Where MIPS1 …

Real Time Embedded Systems

- Real-time systems are systems whereby the correctness of the computed values and their timeliness are at the forefront.
- There are two types of real-time systems, hard and soft real time.
- Examples …

Aerospace

- Flight management systems
- Jet engine controls
- Weapons systems

Audio

- MP3 players
- Amplifiers and tuners

Automotive

- Antilock braking systems
- Climate control
- Engine controls
- Navigation systems (GPS)

Communications

- Routers
- Switches
- Cell phones

Computger peripherals

- Printers
- Scanners

Domestic

- Air conditioning units
- Thermostats
- White goods

Office automation

- FAX machines / copiers

Process control

- Chemical plants
- Factory automation
- Food processing

Robots

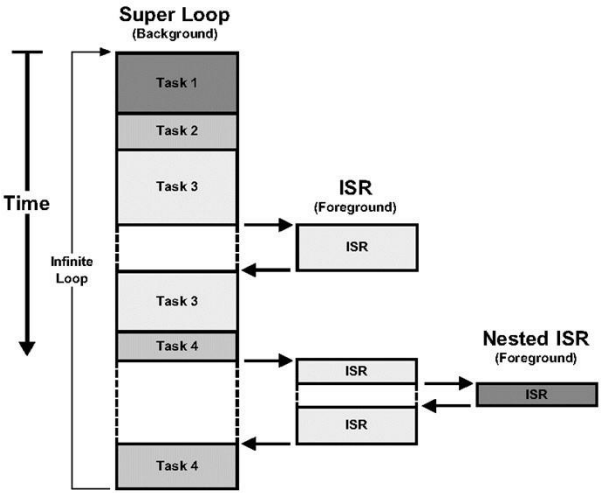
Video

- Broadcasting equipment
- HD Televisions

And many more

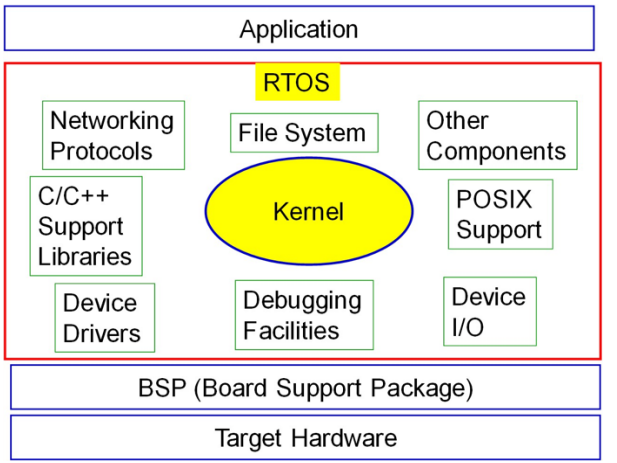
Foreground/Background System or Super Loop

- Applies to small system with low complexity
- Foreground is called the interrupt level
  - ISR handles asynchronous events
- Background is called the task level
  - Note, critical task operations must be handled by ISRs now.



Real-Time Operating System (RTOS) General Concept

- An RTOS is a program that schedules execution in a timely manner, manages system resources, and provides consistent foundation for developing application code
- An RTOS contains …
- An RTOS should be scalable …



Real-Time Kernel

- The kernel is the program that manages the time and resources of a microprocessor, microcontroller or DSP
- What is work or job? … What is a task? …
- A kernel is responsible to manage all tasks of an application in a multitasking system
- Multitasking is the process of scheduling and switching the CPU between several tasks
- General kernel components:
  - Scheduler – implements a set of scheduling algorithms
  - Objects – special kernel constructs such as semaphores, etc.
  - Services – operations that are performed on an objects, or in general

Preemptive Kernel

- uC/OS-III is a preemptive kernel
- Here, the kernel always runs the highest priority task that is ready to run
- Ex … See diagram …
- A kernel introduces time and memory overhead