ENGG4420 – Real-Time Systems Design

Fall 2025

Instructor: Petros Spachos

Slides: Radu Muresan

References

Main References:

1. R. Muresan, ENGG*4420 F22 Lecture Notes, CourseLink University of Guelph
2. R. Muresan, ENGG*4420 Laboratory Manual, CourseLink University of Guelph
3. G. C. Buttazzo, Hard Real-Time Computing Systems, 3rd Edition, 2011, Springer
4. Micriμm Press, μC/OS-III The Real-Time Kernel, User Manual
5. Richard Barry, Mastering the FreeRTOS Real-Time Kernel, A Hands-On Tutorial Guide

Additional References:

1. G. F. Franklin, J. D. Powell, A. Emami-Naeini. Feedback Control of Dynamic Systems, 7th Edition, 2014, Pearson
2. Warren Gay, Beginning STM32, Apress, 2018

ENGG4420: Course Module Diagram

- The course is organized in 4 main parts:
  - Real-time computer control with examples
  - Real-Time Operating Systems with two examples of commercial RTOS
    - uC/OS-III
    - FreeRTOS
  - Hard real-time computing systems and scheduling algorithms
    - Educational small real-time kernel
  - Reading: Reliability and fault tolerance

Real-Time Computer Control Topics

- Real-time definitions and application examples
- Design of real-time systems
- Examples of plant models for real-time implementation and testing (laboratory example)
- Simple Direct Digital Control (DDC) concepts and control algorithm implementation -- examples

What is Real-Time

- **Real-time** relates to the quantitative notion of time and is measured using a physical clock
  - Within this context, real-time response is the ability to reliably and without fail respond to an event or perform an operation within a guaranteed time period
- **Logical time** (or virtual time) relates to the qualitative notion of time and is expressed using event ordering relations
  - In this context, while dealing with logical time, time readings from a physical clock are not necessary for ordering the events

Real-Time System Definition

- A system is called a real-time system, when we need quantitative expression of time (i.e., real time) to describe the behaviour of the system
  - Both the computer system and the controlled system (environment) use the same time scale.
- More definitions … see lecture notes (courselink)!

Real Time and Logical Time Examples

- Real time example
  - Chemical plant – temperature controlled chemical reaction chamber

- Logical time example
  - Library software – automation of the bookkeeping activities in a college library

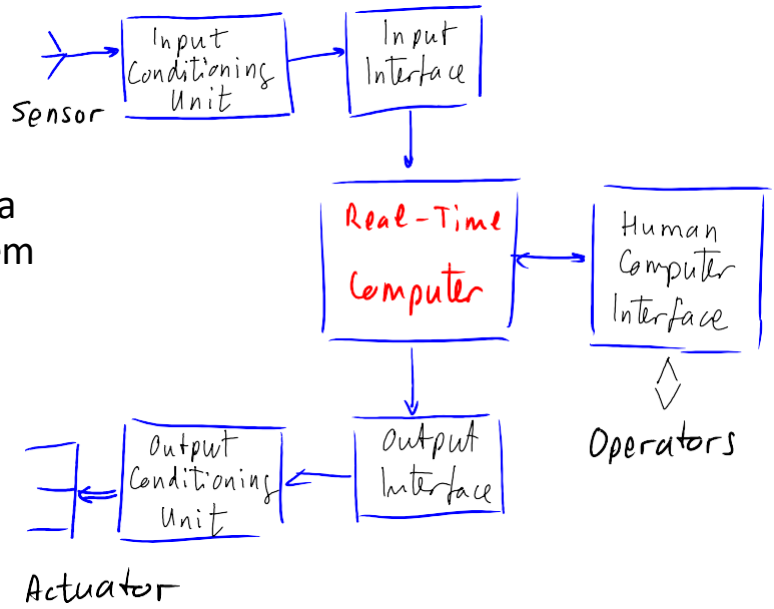
Examples of Real-Time Systems

- Chemical and nuclear plant control …
  - Typical time bounds for chemical reactions control are few microseconds to several milliseconds
- Control of production processes …
  - Typical time of workstations are in the order of 100x of ms
- Supervisory control data acquisition (SCADA) …
  - Sensing of data on distributed events, around few ms
- Robotic type applications …
  - Path planning time bounds, around few ms depending on the system

Examples of Real-Time System

- Telecommunications – cellular systems, video conferencing, routing, etc.
  - Ex. For a cellular system, the hand off calls from a base station to the next is a few ms
- Aerospace – avionics, flight simulation, airline cabin management systems, satellite tracking system, on-board aircraft computer, etc.
  - Ex. For on-board aircraft computer the position adjustment control must be performed in few microseconds
- And other examples … see lecture notes!

Basic Model of a Real-Time System

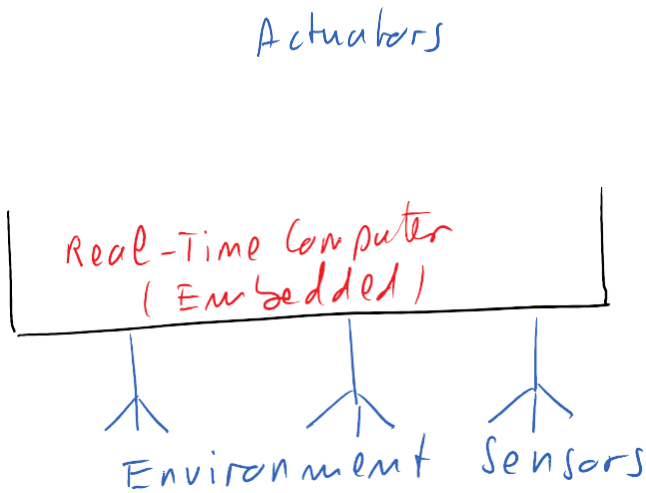


Important Characteristics of Real-Time Systems

- Time constraints: each real-time task is associated with some time constraints: a) Deadline (common constraint); b) Delay; c) Duration.
  - As a result, a Real-Time Operating System (RTOS) must support mechanisms to ensure that all tasks meet their respective time constraints.
- Correctness criterion: correctness in real-time system implies both logical correctness and time correctness of results
  - Time at which the results are produced is important
  - A logically correct result produced after the deadline would be considered incorrect
- Safety-criticality: many real-time systems include safety and reliability features and become safety-critical systems -- a safety-critical system is required to be highly reliable
  - Safe system is one that does not cause any damage even when it fails
  - Reliable system is one that can operate for long durations of time without any failure
- Concurrency: a real-time system usually needs to respond to several independent events within very short and strict time bounds:
- Other characteristics are: stability, task criticality, etc.

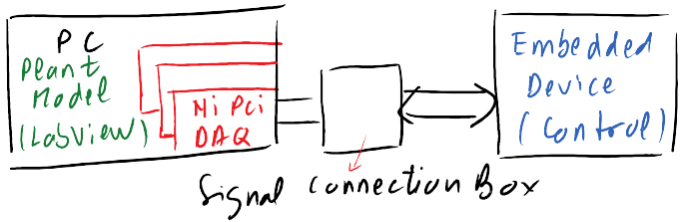
Design of Real-Time Embedded System

- Complex due to its environment interaction
- Difficult to test during the design process
- Need to use plant models and environment models in order to develop a functional design
- LabView is a software tool that is useful in simulating environment, modeling plants, testing and implementing supporting software

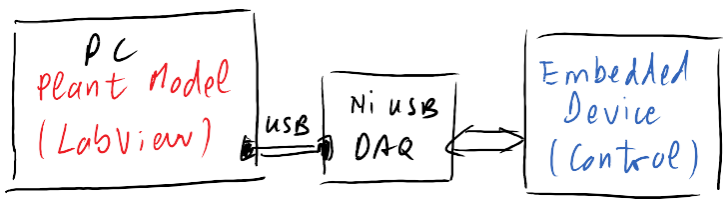


Test Architectures Using LabView Software and NI hardware

a)



b)



Classification of Real-Time Systems Based on Synchronization

- Computer tasks are connected by physical devices to external processes
  - Note that tasks are internal actions carried by the computer
  - External processes have their own time scale
- Computer tasks can operate in real-time if their actions are synchronized to the time-scale of the external processes
- Real-time systems can be classified based on synchronization type used:
  - **Clock based** – use passage of time or actual time of the day
  - **Event based** – the events determine the synchronization
  - **Interactive** – relationship between actions in computer and processes is loosely defined

Importance of Plant Characteristic

- In the context of synchronization between computer and the processes we need to consider the notion of plant characteristic!
- What is plant characteristic?
- There is a tight relationship between the plant characteristic and the sampling rate used by a real-time digital control computer
  - The computer must be able to carry out all the required operations within each sampling time

Features of Real-Time Tasks Based on Synchronization Methods

- Clock based tasks – are referred to cyclic or periodic
  - These type of tasks are to be run once during a time period T, or they must run exactly every T unit intervals
  - Implementation – use a real-time clock and use interrupts from this clock
- Event based tasks – event occurs at non-deterministic intervals; aperiodic tasks
  - Actions are performed in response to some event
  - Implementation of such task will use interrupts generated by events response must be within a maximum time to event issue
- Interactive tasks – the real-time requirements for these types of tasks is expressed in the average response time (order of seconds)
  - Interactive versus event-based tasks
  - Interactive systems respond at a time determined by the internal state of the computer and without reference to the environment
  - Interactive versus clock-based tasks
  - Clock based are tightly synchronized to an external process through a clock while interactive are not

Types of Real-Time Systems and Tasks Based on Deadline Constraints

- Hard real-time tasks
  - Must produce its results within certain predefined time bounds otherwise **failure** results
  - Time bounds for this class ranges from several micro-seconds to a few milliseconds. Ex … robot, path planning
- Firm real-time tasks
  - Must produce the result in a predefined deadline but the **system will not fail** if the result is late, the result gets discarded
  - Time bounds for this class range from few milliseconds to several hundreds of milliseconds. Ex … satellite tracking system
- Soft real-time task
  - Average time bounds for these systems ranges from a fraction of second to a few seconds over a time period
  - Ex … railway seat reservation system

Summary of Task Classification Based on Time Bounds

- Hard real-time tasks – deadline time bounds from several µs to a few ms
- Firm real-time tasks – deadline time bounds from a few ms to 100s of ms
- Soft real-time tasks – deadline time bounds from fractions of a second to a few seconds
- Non-real-time tasks – deadline time bounds are in order of a few minutes, hours or even days

Formal Classification of Real-Time Tasks

Hard		Soft	
Periodic (Cyclic)	*A periodic (Event)*	*Periodic (Cyclic)*	*A periodic (Event)*
$t_c(i) = t_s \pm a$	$t_e(i) \le T_e$	$\frac{1}{n}\sum_{i=1}^{n} t_c(i) = t_s \pm a$	$\frac{1}{n}\sum_{i=1}^{n} t_e(i) \le T_a$
		$n = T / t_s$	$n = T / t_s$

$t_c(i)$	the interval between the i and $i-1$ cycles,
$t_e(i)$	the response time to the ith occurrence of event e,
t_s	the desired periodic (cyclic) interval,
T_e	the maximum permitted response time to event e,
T_a	the average permitted response time to event e measured over some time interval T,
n	the number of occurrences of event e within the time interval T, or the number of cyclic repetitions during the time interval T,
a	a small timing tolerance.

Design of Real-Time Systems, General Introduction

- Design Example
- Single-Program Approach
- Foreground/Background System
- Multi-Tasking Approach

Design Example of Real Time Computer Control

- There is more to designing and implementing computer control systems than simply programming the control algorithms
- The overall design of the system can be divided into two main phases
  - The planning phase: concerned with interpreting user requirements and generating a specification document of the system to be developed and an outline of the resources
  - The development phase: contains a preliminary design stage (software and hardware division) and a detailed design stage
    - The detailed design stage is usually divided into two substages:
      - Decomposition into modules and
      - Module internal design

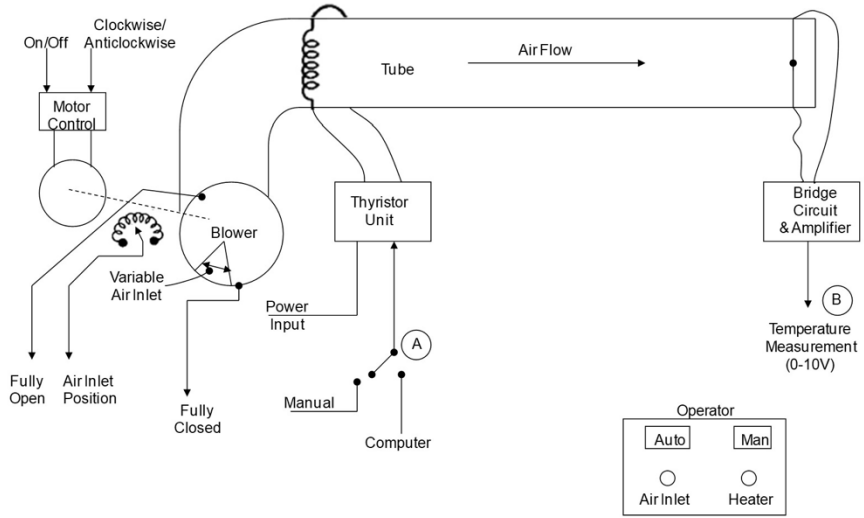
Cont…

- Within the software design for real-time systems additional heuristics are required
- For example, software modules must be divided into following categories:
  - Real-time, hard constraint;
  - Real-time, soft constrain;
  - Interactive
- The general rule is to aim to minimize the amount of software that falls into the hard constraint category …

Example of the Real-Time Control System Procedure: Hot Air Blower System

- Problem:
  - The system comprises a set of hot-air blowers arranged along a conveyer belt
  - Several different configurations may be used with a minimum of 6 blowers and a maximum of 12
- Let's look first to an individual hot air plant unit

Hot Air Blower Plant Unit, Plant Diagram



Hot Air Blower Plant Components Description

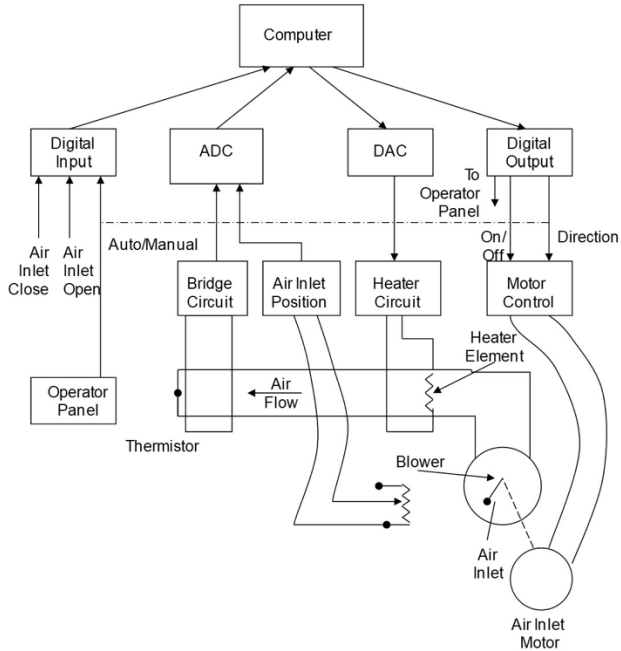
- Centrifugal fan …
- Thermistor …
- Heating element …
- Reversible motor …
- Potentiometer wiper for inlet air …
- Microswitches …
- Slider potentiometer for the temperature reference …
- Operator panel …
- Reference temperature setting …

Real Time Computer Functional Requirements for this Plant

- The operation of this simple plant using a computer requires that that software be provided to support:
  - Monitoring
  - Control
  - Actuation of the plant
  - Communication

Computer Control, Hardware/Software Interfacing of the Hot-Air Blower

- Software interfaces are directly connected to the plant
- In general, the microcontrollers support many types of I/O interfaces

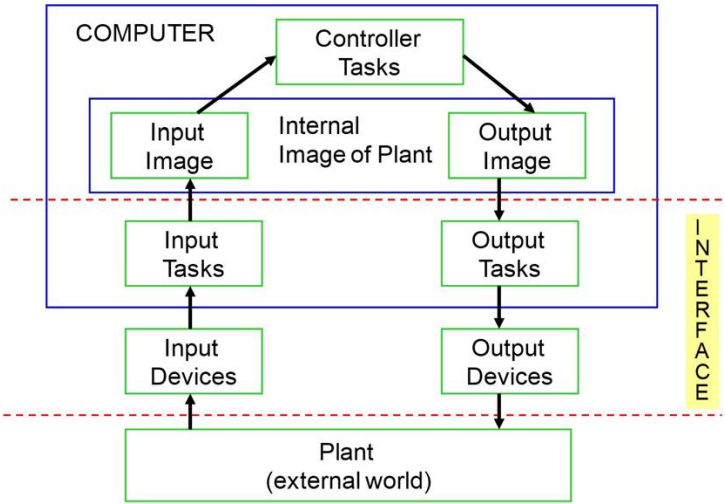


Description of the Plant Operation Functions

- Monitoring – information from the plant
  - Collecting information from the plant instruments require digital and analog signal interfacing
- Control – digital control algorithm
  - Involves the digital equivalent of continuous feedback control for temperature control (direct digital control – DDC) and for the control of the fan-inlet cover position
- Actuation
  - Required the provision of a voltage proportional to the demanded heat output to drive the heater control
  - And, control for the motor driving the fan-inlet cover
- Communication … support for information to and from operator

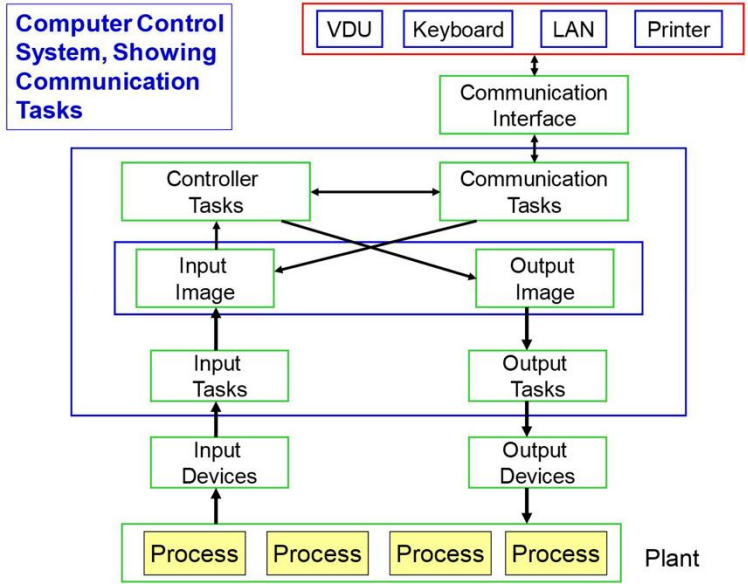
Generalized Computer Control System Showing HW and SW Interfaces

- Input devices are devices which transmit information to the computer
- The software to operate this devices here is represented as input tasks and output tasks
- The input image is a snapshot of the status of the plant, renewed at specified intervals
- The output image represent the current set of outputs generated by the control calculations



Computer Control with Communication Tasks

- Simple systems can treat the communication function as part of the input and output tasks
- Many applications will require dedicated communication tasks



Example of the Real-Time Control System Procedure: Hot Air Blower System

- System Problem:
  - The system comprises a set of hot-air blowers units arranged along a conveyer belt. Several different configurations may be used with a minimum of 6 blowers and a maximum of 12.
- Hot-air blower system, design specifications components:
  - Plant interface
  - Control
  - Operator communication
  - Management information

Plant Interface Design

- Input from the plant
  - Outlet temperature: analog signal, range 0 to 10 V, corresponding to 20^{o} C to 64^{o} C
- Output to plant:
  - Heater control: analog signal 0 V to 10V, corresponding to no heat (0V) to full heat (10V), linear relationship

Plant Control Design

- Digital PID controller with a sampling interval of 40 ms is to be used; the sampling interval may be changed but should not be less than 40 ms
  - The controller parameters should be given to the user in standard analog form such as:
    - Proportional gain, integral action time and derivative action time.
  - The set point is to be entered from the keyboard
  - The controller parameters are to be variable and should be entered from the keyboard

Operator Communication Design

Operator display =>

Display values updated every 5 seconds!

General Settings		Controller settings	
Set temperature	:nn.n C	Proportional gain	:nn.n
Actual temperature	:nn.n C	Integral action	:nn.nn s
Error	:nn.n C	Derivative action	:nn.nn s
Heater output	:nn% FS	Sampling interval	:nn ms
Other settings			
Date	:dd/mm/yyyy	Time	:hh.mm

Operator input =>

1. Set temperature = nn.n
2. Proportional gain = nn.n
3. Integral action = nn.nn
4. Derivative action = nn.nn
5. Sampling interval = nn
6. Management information
7. Accept entries

Select menu number to change

Management Information design

- Selecting item 6 of the operator menu a management summary will be performed of the plant over the previous 24 hours
- The summary will include:
  - Average error in ^{o}C in 24 hours period
  - Average heat demand %F.S. in 24 hours
  - Each 15-minute period report – average demanded temperature, average error, and average heat demand
  - Date and time of output

Preliminary Hardware Design

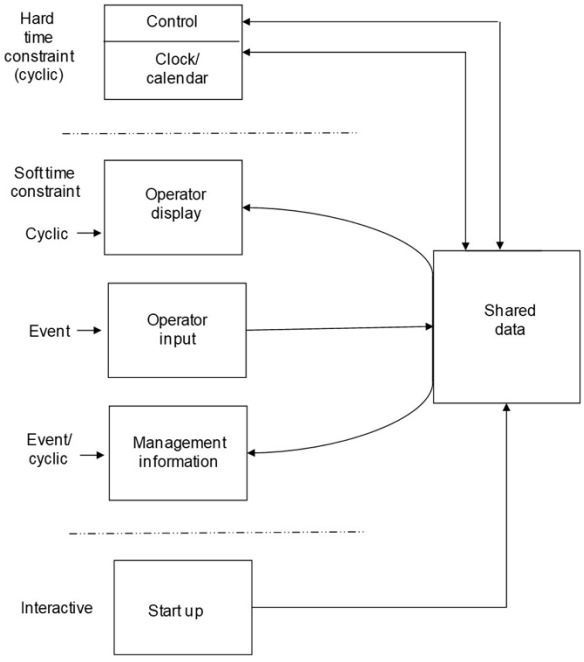
- Hardware design; there are various possibilities for the hardware structure:
  1. Single computer with multi-channel ADC and DAC boards
     - Given that the specification calls for the system to be able to run with a sample interval for the control loop of 40ms, can this be met with 12 units sharing one processor?
  2. Separate general-purpose computers on each unit
     - Is putting a computer that includes a display and keyboard on each unit an expensive solution? Will communication between computers be required?
  3. Separate computer-based microcontrollers on each unit linked to a single general-purpose computer
     - What sort of communication network should be used? A shared high-speed bus? A local-area network? Where should the microcontrollers be located? At each blower unit or together in a central location?

Preliminary Real-Time Software Design

- Based on the air-blower system specification document the software needs to perform the following functions:
  - Direct digital control (DDC) for temperature and blower control
  - Operator display
  - Operator input
  - Provision of management information
  - System start-up and shut-down
  - Clock/calendar function

Block Diagram of Software Modules for the Air-Blower System

* Types of time constraints for the required tasks



Constraints Analysis for the System's Tasks

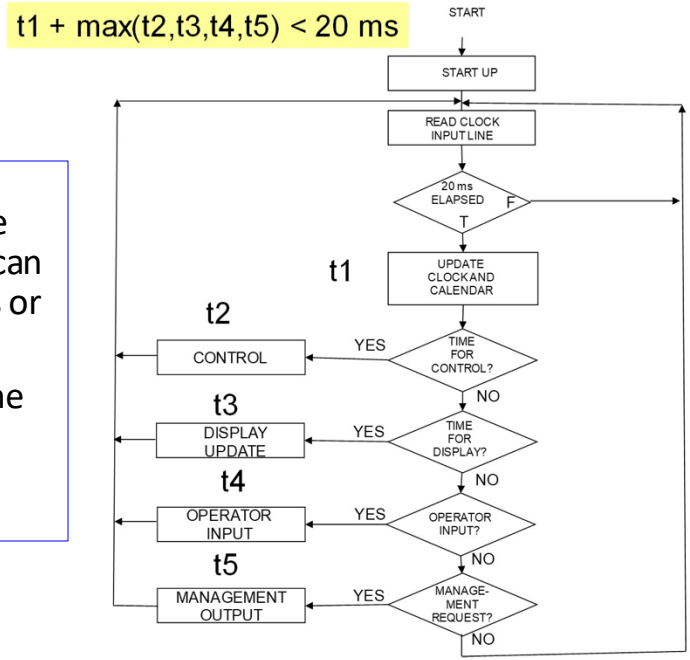
- Control module has a hard constraint – runs every 40 ms
  - …
- Clock/calendar task has a hard constraint – runs every 20 ms
  - …
- Operator display task has a soft constraint – runs every 5 s on average
- Operator input and management information tasks will have soft constraints
- Start-up module can be viewed as an interactive task

System Software Implementation Approach

- There are few different activities which can be divided into subproblems where these subproblems must share some information – this is dependent of the general approach implementation of the system
- Types of general software implementation of the system
  - Single program
  - Foreground/background system
  - Real-time multi-tasking

Single Program Software Implementation

- Using the standard programing approach the modules for this system can be treated as procedures or subroutines
- t1, t2, t3, t4 and t5 are the individual modules



Single Program Structure, Advantages/Disadvantage

- **Advantages**: simple structure, easy to implement
- **Disadvantages**: it imposes severe time constraints
  - For example, the requirement that the clock/calendar module must run every 20 ms on all the modules.
  - Note that time t1 must include the control computations for each of the 12 units plus the times must include testing execution …
- **Conclusion**: the single-program approach can be used for simple systems with minimum of hardware and software
  - In our example, the management output requirement makes our system unsuitable for the single program approach.

Foreground/Background Software Implementation

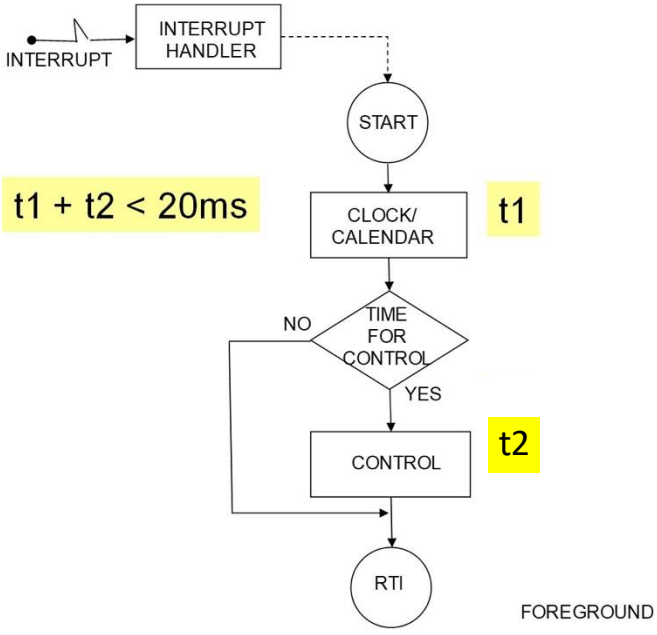
- This technique could provide less module interaction plus less time constraints if:
  - The hard time constraints module can be separated from, and handled independently of the other modules (soft and interactive)
- In this approach the modules with hard constraints are run in "foreground" and the modules with soft or no constraints in "background"
- What is foreground and background???

Implementation Considerations

- The partitioning into foreground and background usually requires the support of a real-time operating system
- However, for this approach it is possible to adopt many standard operating systems as well if the hardware supports interrupts
  - The foreground tasks are written as an interrupt routine
  - The background tasks are written as a standard program

Foreground Module

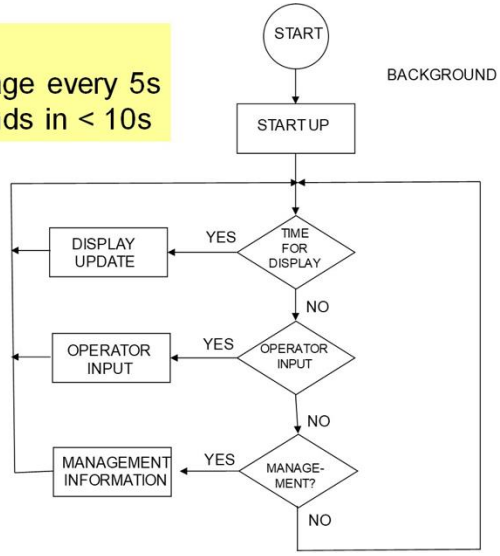
- This is a clear separation of the foreground part
- The time requirement of this part to work is given in the diagram



Background Module

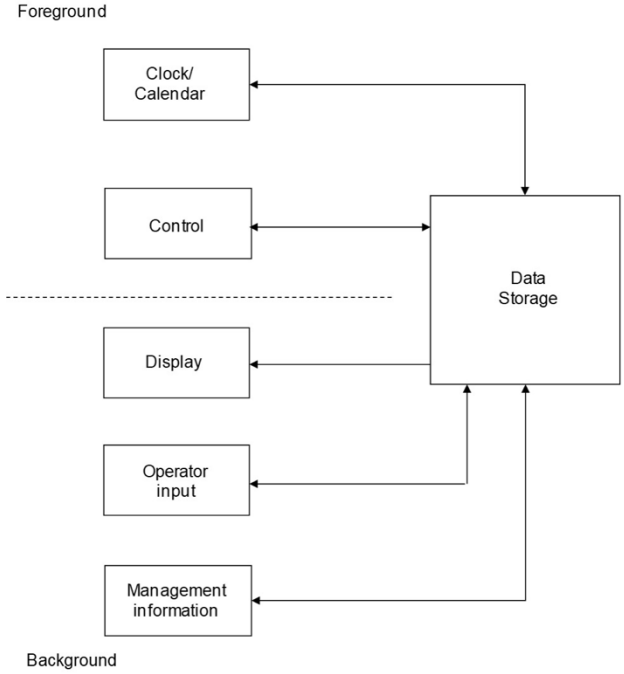
1. max(t3,t4,t5) < 10s;
2. display runs on average every 5s
3. operator input responds in < 10s

- Background module also has some time requirements in order to work
- In this approach we relaxed the time constraints, but we affected the data communication between tasks which will complicate the evaluation of the performance required to report



Foreground/Background Data Structure Considerations

- This approach separates the control structure of the foreground and background modules, the modules are still linked through the data structure
  - They share data variable
  - For example … buffering needed
- Need data access protection and activity synchronization or rendezvous between tasks



Multi-Tasking Software Implementation

- The design and programming of large real-time systems is eased if the operating system supports the existence of many active tasks
- At the preliminary design stage each activity is considered to be a separate task
- The implication of this approach is that each task may be carried in parallel but there is no assumption at this stage as how many processes will be used in the system

Multi-Tasking Approach

- The implementation of multi-tasking systems requires the ability to:
  - Create separate tasks
  - Schedule running of the tasks, usually on priority basis
  - Share data between tasks
  - Synchronize tasks with each other and with external events
  - Prevent tasks corrupting each other
  - Control the starting and stopping of tasks
- To support these actions, we typically use a real-time operating system (RTOS)

Common Problems Arising in the Design of Multi-Tasking and why we Need an RTOS Support

- Example 1: transfer of information from an input task to a control task =>
  - Mutual exclusion support
- Example 2: as part of the maintenance procedure a record is kept for a device – after a specified number of accesses maintenance is required =>
  - Condition flags
  - Counting semaphores
    - Test conditions set conditions must be indivisible operations …

Cont…

- Example 3: each task that requires access to a particular resource must know the details of the semaphore used to protect that resource and to use it – can create some difficulty for a large system =>
  - Monitor construct
- Example 4: Alternative to the use of monitors and signals to ensure mutual exclusion and synchronization in intertask communication =>
  - Rendezvous protocol

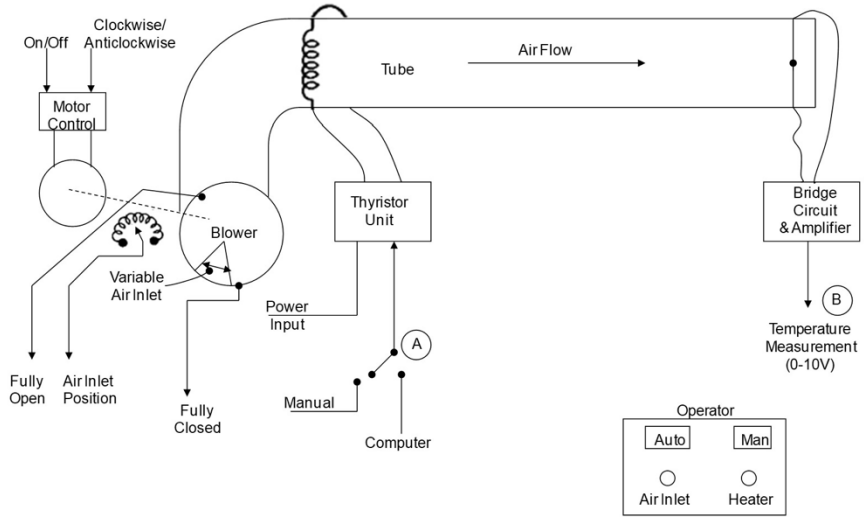
Summary

- Important to divide the operations into tasks
- Definitions
  - Process, plant …
  - Program …
  - Module …
  - Task …
- There are 3 models for implementing real-time software applications – the designer needs to determine the need of an RTOS
  - Single program
  - Foreground/background
  - Real-time multi-tasking

Designing and Testing Real-Time Systems, What are the Issues?

- In the design phase in order to test a real-time system implementation we can use plant models and connect these models through software to our control system
- In our course we present briefly some simple plant modeling results
  - Heat and fluid-flow model: hot-air plant
  - Mechanical system model: cruise control
  - Electrical motor

Hot Air Blower Plant Device, System Diagram



Lecture Summary

- What is Real-Time?
  - Real-time = a system must respond within a guaranteed time limit.
  - Two concepts of time:
    - Real time → measured by a physical clock (strict deadlines).
    - Logical/virtual time → based on event ordering (sequence matters, not clock time)
- Examples:
  - Real time → chemical plant temperature control
  - Logical time → library management software

Lecture Summary

- Real-Time System Characteristics
  - **Time constraints**: deadlines, delays, durations.
  - **Correctness**: results must be both logically correct and produced on time.
  - **Safety-criticality**: many real-time systems are mission/safety critical.
  - **Concurrency**: must handle multiple events at once.

Lecture Summary

- Types of Real-Time Systems
  - **Hard real-time**: missing a deadline = system failure (e.g., robotics, avionics).
  - **Firm real-time**: late results are useless but not catastrophic (e.g., satellite tracking).
  - **Soft real-time**: deadlines are flexible, occasional delays acceptable (e.g., video streaming).
  - **Non-real-time**: no strict deadlines (e.g., batch data processing).

Lecture Summary

- Synchronization Methods
  - **Clock-based (periodic tasks)** → run every T ms (cyclic scheduling).
  - **Event-based (aperiodic tasks)** → triggered by events.
  - **Interactive tasks** → response time is looser (seconds)

- System Design Approaches
- Three models for software implementation:
  1. Single Program → simple but inflexible (all tasks sequential).
  2. Foreground/Background → urgent tasks run in "foreground" (interrupts), others in background.
  3. Multi-tasking (RTOS-based) → tasks run concurrently, managed by scheduling and synchronization (best for complex systems).

Lecture Summary

Case Study: Hot Air Blower System

- A plant with 6–12 hot air blowers along a conveyor belt.

Needs:

- Monitoring (temperature, fan status).
- Control (PID controller for heat and airflow).
- Actuation (heater/motor outputs).
- Operator communication (settings, displays).
- Management information (logging, averages).

Design considerations:

- Hardware: single computer vs. distributed microcontrollers.
- Software: PID control, operator input, display, logging.
- Task classification:
    - Hard constraints: control loop (40 ms), clock/calendar (20 ms).
    - Soft constraints: operator display (5 s).
    - Interactive: start-up.

Lecture Summary

- Real-time systems must meet **both correctness and timing requirements**
- The **design process** includes modeling, simulation, and constraints analysis
- The **implementation model** chosen (single, foreground/background, multitasking) depends on system complexity
- **RTOS** support is often necessary for concurrency, synchronization, and reliability