Lecture: 9

Design of Real-Time Systems

General Approach

Prof. Kasim M. Al-Aubidy
Computer Engineering Department
Philadelphia University
Summer Semester, 2011
Course Objectives:
The main objective of this unit are:

• To give an outline of a general approach to the design of computer-based real-time systems.

• To consider three approaches in designing the software of real-time systems, these are:
  – Single-program approach.
  – Foreground/background approach.
  – Multi-tasking approach.

• To show how to approach the planning and design of a real-time system.

• To illustrate the basic approaches for the top-level design of real-time software.
Real-Time Systems Design:

- The approach to the design of RTSs is no different in outline from that required for any computer-based system. The work can be divided into two main sections:
  - The planning phase, and
  - The development phase.

Planning Phase:

- It is concerned with interpreting user requirements to produce a detailed specification of the system to be developed and outline plan of the resources, people, time, equipment and costs.
- At this stage preliminary decisions regarding the division of functions between hardware and software will be made.
Development Phase:

- The aim of preliminary design stage is to decompose the system into a set of specific sub-tasks. The inputs to this stage are the HL specifications. During this stage extensive liaison between hardware and software designers is needed.
- The detailed design is usually broken down into two stages;
  - Decomposition into modules, and
  - Module internal design.
Specification Document:
Example: Hot-air blowers.
Preliminary Design:

**Hardware Design:** to be discussed in lecture.

**Software Design:**

- The required software must perform several functions:
  - DDC for temperature control.
  - Operator display.
  - Operator input.
  - Management information.
  - System start-up and shut-down.
  - Clock/calendar function.
- The control module has a hard constraint, it must run every 40 msec.
- The clock/calendar module must run every 20 msec.
- The operator display has a hard constraint in that an update interval of 5 sec is given.
- Soft constraints are adequate for operator i/p and for the management information.
Software Design:

- There are several different activities which can be divided into sub-problems. The sub-problems will have to share information. To achieve this, there are three approaches:

1. **Single-Program Approach:**
   - The modules are treated as procedures or subroutines of a single program.
   - This structure is easy to program, however, it imposes the most severe of the time constraints.

**Example:** for the system to work the clock/calendar module and any one of the other modules must complete their operations within $T$. $t_1$, $t_2$, $t_3$, $t_4$ and $t_5$ are the maximum execution times for the given modules, then a requirement for the system to work is;

$$t_1 + \max(t_2, t_3, t_4, t_5) \leq T$$
2. **Foreground/Background System:**

- There are advantages (less time constraints) if the modules with hard time can be separated from, and handled independently of, the modules with soft time constraints or on constraints.
- The modules with hard time constraints are run in “foreground” and the modules with soft constraints (or no constraints) are run in the “background”.
- The partitioning into foreground and background usually requires the support of a real-time OS.
- A requirement for the foreground part to work is that:
  \[ t_1 + t_2 \leq T \]
- A requirement for the background part to work is that:
  - \(\max(t_3, t_4, t_5) \) is less than 10 sec.
  - Display module runs on average every 5 sec, and
  - Operator input responds in less than 10 sec.
**Multi-Tasking Approach:**

- The design and programming of large RT systems is eased if the foreground/background partitioning can be extended into multiple partitions to allow the concept of many active tasks each can be carried out in parallel.

- The implementation of a multi-tasking system requires the ability to;
  - Create separate tasks.
  - Schedule running of the tasks on a 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.

- The facilities to perform the above actions are typically provided by a RTOS or a combination of RTOS and a real-time programming language.
**Mutual Exclusion:**

- Consider the transfer of information from i/p task to a control task. The i/p task gets the values for the controller i/p parameters (gain, Ti and Td). From these it computes the controller parameters (KP, KI, and KD) and these are transferred to the CONTROL task.
- A simple method is to hold the parameters values in an area of memory (common data area) and hence is accessible to both tasks.
For more information: