Chapter I: INTRODUCTION TO REAL-TIME SYSTEMS

- 1.1 Characteristics of Real-Time Systems
- 1.2 Engineering Applications of Real-Time Systems
- 1.3 Examples of Real-Time Systems.

Chapter 1 starts by defining real-time systems and discussing their characteristics and attributes. Engineering applications of real-time systems are discussed in section 1.2. Several examples of the functional requirements of real-time systems are then presented in section 1.3. These examples are complex enough to motivate the use of an industrial software development process. Section 1.3 discusses typical functional and timing requirements for the application examples presented. These requirements are used in later Chapters to provide running examples of software requirements analysis, design, and implementation.

1.1 Characteristics of real-time systems

The term real-time systems has been used extensively in many applications of computing and control systems. Although the term has been well defined in the literature, it is still being misused and misrepresented. We devote the first part of this section to the definitions of terms and the rest of the section will present the well known characteristics of real-time systems.

Definitions

What is a real-time system?

A Real-Time System (RTS) is defined as a system in which the time where the outputs are produced is significant. The outputs must be produced within specified time bounds referred to as deadlines. The correctness of a RTS depends not only on the logical results produced, but also on the times at which such results were produced. The system may enter an incorrect state if a correct result is produced too early or too late with respect to the specified time bounds or deadlines.

Figure 1.1 shows a block diagram representation of an example of a RTS. Inputs may come from sources such as sensors, switches, communication links, or a keyboard and outputs may go to actuators, alarms, motors, lamps, communication links, or displays. Outputs of the system have associated time bounds within which they must be produced.

FIGURE 1.1 An example of a Real-Time system

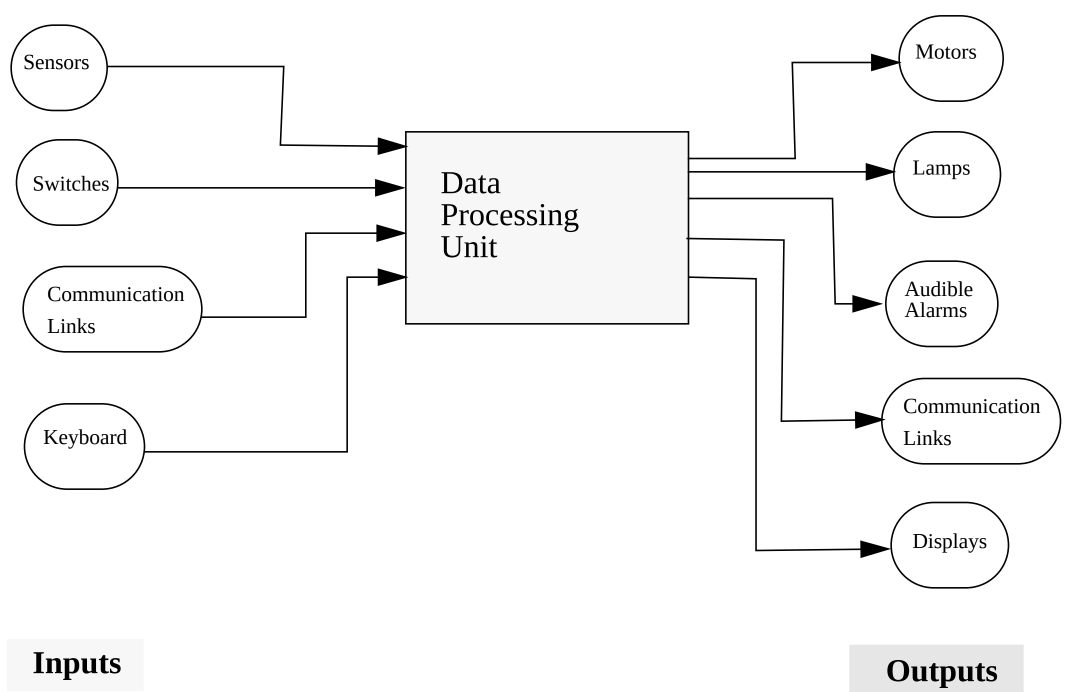
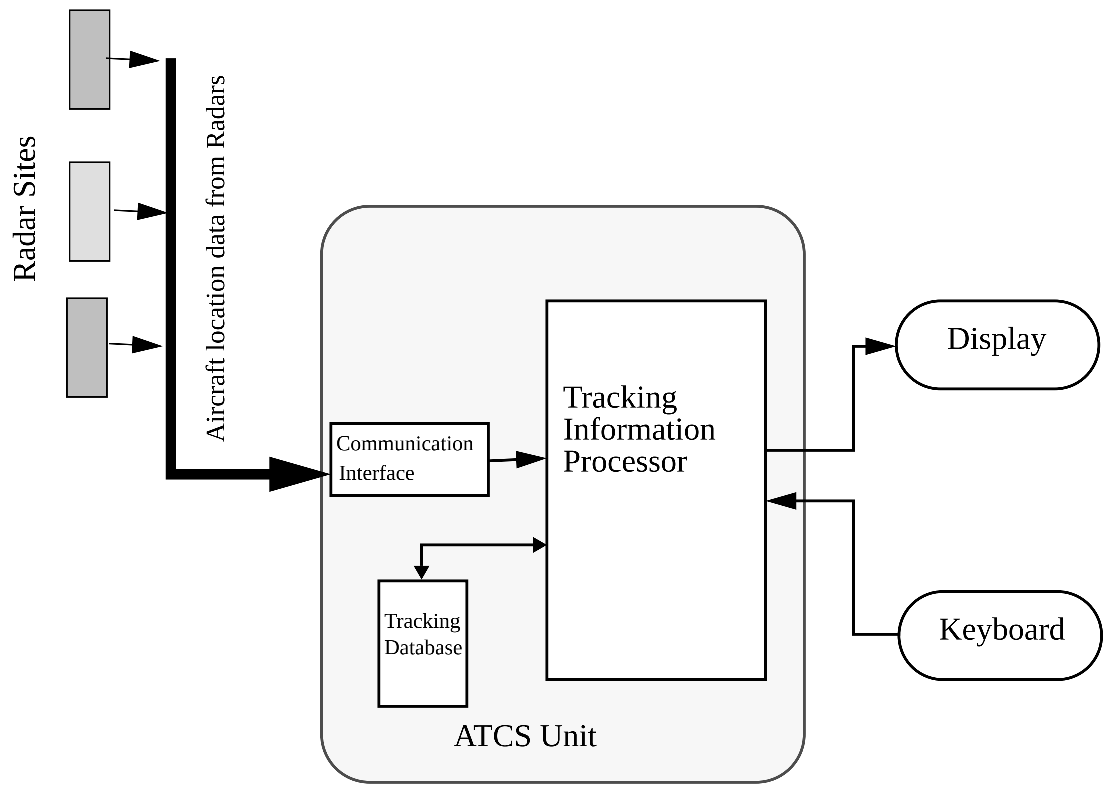


Figure 1.2 shows a block diagram of an air traffic control system. The system displays aircraft tracks obtained from radar sites. It accepts radar data on tracks locations, and operator inputs to establish tracks or to change the location of a specified track. The system periodically extrapolates all aircraft locations and produces an output to a display containing all the current tracks. The

system updates the aircraft tracks (maintained in a track database) from radar input data every 100 ms and updates the output display every 500 ms.

FIGURE 1.2 An Overview of ATCS



Real-time systems are usually categorized as follows:

- Hard RTSs are those systems where it is absolutely imperative that responses occur within the specified deadlines. (Examples are aircraft control, air traffic control, process control applications).
- Soft RTSs are those systems where response times are important but the system will still function correctly if deadlines are occasionally missed. (Examples are communication systems using a time-out protocol. If an acknowledgment for a message was not received before the deadline a time-out occurs and the same message is resent again. Missing the deadlines here can be tolerated occasionally, however if deadlines are missed frequently resent messages will jam the bandwidth of the communication channel and the system will cease to perform).

Characteristics

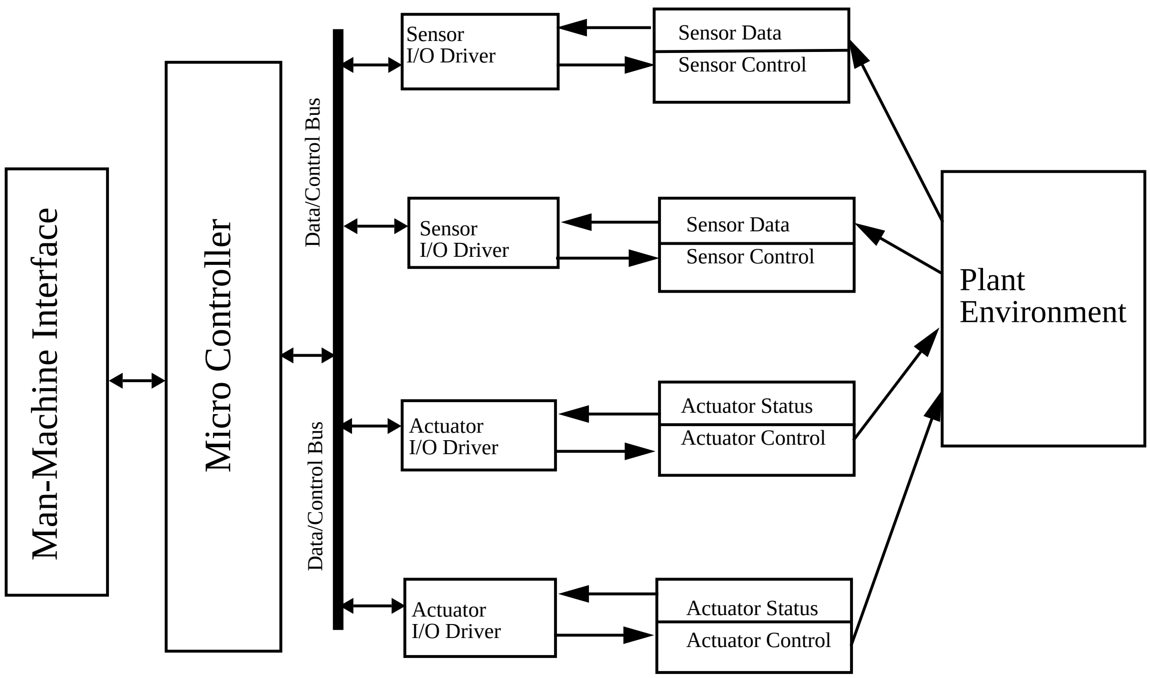
What are the characteristics which distinguish real-time systems from others?

The following characteristics are often found in many RTSs, however, a system does not need to have all these characteristics to be a RTS.

1. An RTS is used within a larger system to provide monitoring, control, and computation functions. Such systems are called “embedded computer systems”. They often contain devices

that act as the senses (e.g., heat sensors, or light sensors), and devices that act as the effect of physical changes (e.g., mechanical, electromechanical, and electronic actuators). In figure 1.3 a block diagram of a typical embedded system is shown. The system is usually built around a micro-controller used for executing various system monitoring and control functions. The system reads the input of various sensors, applies various filtering, calibration, and processing algorithms on the input data, and produces output data to various actuators. An operator interface is provided to allow for manual instructions. Testing and maintenance of the such systems can be performed by special devices such as In Circuit Emulators (ICE). An ICE is a device which monitors the real-time operations of a hardware micro-controller while executing the system software functions.

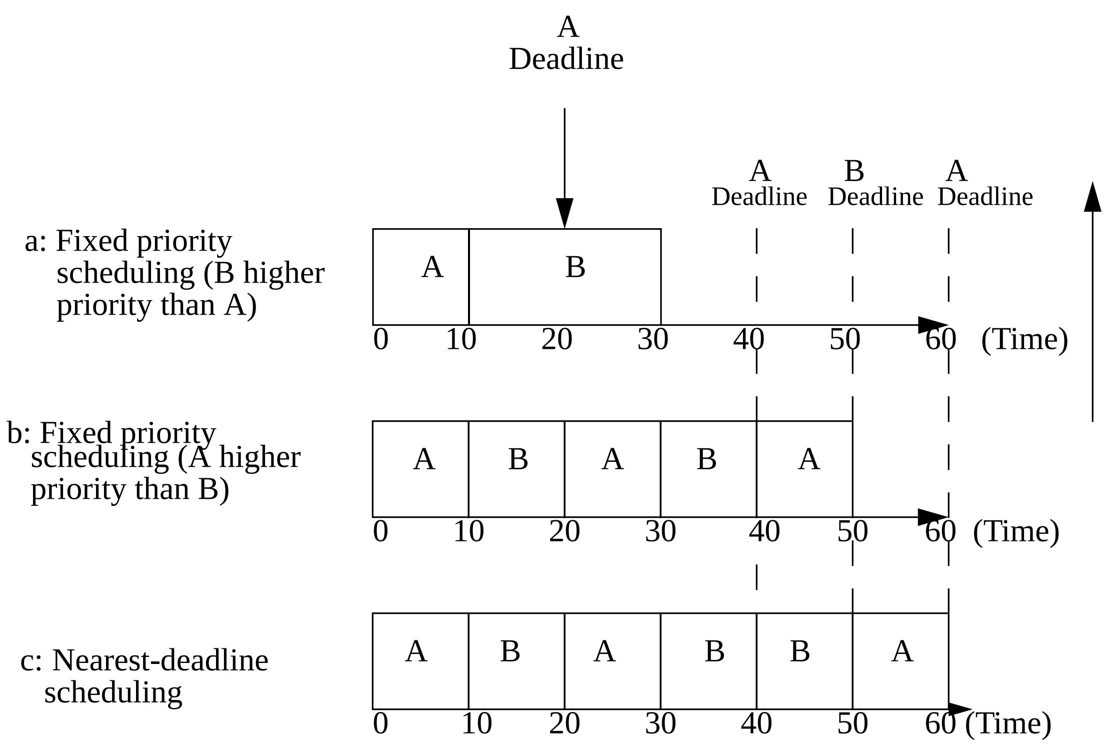
FIGURE 1.3 An Embedded Computer System



2. RTS systems often require concurrent processing of multiple inputs. This involves correlated processing of multiple inputs over the same time interval (e.g. an industrial process control system might be required to correlate values of temperature, pressure, and a concentration of a chemical reaction to perform simultaneous adjustments of heaters and valves to maintain a reaction in the desired state). Concurrent tasks must be created and managed in order to fulfil the functions of a RTS. Task scheduling is one of the important aspects of managing concurrency. Since tasks will compete for the same resources such as a CPU (in a uni-processor system), it is necessary to use priority scheduling in which tasks with more stringent deadlines will be given a higher priority to be scheduled on the CPU first before other tasks of less urgency. A simple example to illustrate this concept is given as follows:

Consider a RTS which collects and processes data from two sensors A and B. Two tasks are created one for each sensor. The task collecting data from sensor A has a deadline of 10 ms (i.e., data must be collected every 10 ms), and the task collecting data from sensor B has a deadline of 20 ms. Task A needs 5 ms to process the data from sensor A and task B needs 10 ms to process sensor B data. In a single microprocessor system if task B was allowed to run first, task A will surely miss the deadline. A high priority given to task A would make both tasks meet the deadlines. However a fixed priority scheme might be problematic since task A will be executed every 5 ms and will preempt B and force it to miss its deadline. A Nearest deadline scheduling scheme gives higher priority to the task whose deadline will be missed. The figure 1.4 below shows the two cases of fixed priority and nearest deadline scheduling.

FIGURE 1.4 Concurrency in Real-Time systems



3. The time scales of many real-time systems are fast by human standards. The complex Devices monitored or controlled often operate in fast time scales (e.g., for an automobile cruise control system to maintain a smooth ride with only small variations from the desired speed, the actual speed must be monitored many times per second). Table 1 shows the response time or deadlines for several real-time applications.

4. The precision of response required for RTSs is greater than that required by other systems. An early or a late response may constitute erroneous behavior. A premature shutdown of a chemical plant could cause extensive damage to equipment or environmental harm.

5. RTS systems have higher reliability and safety requirements than that required by other systems. The failure of a system involved in automatic fund transfer between banks can lead to millions of dollars being lost, failure in an embedded system could result in the failure of a vital life-support system.

6. RTS systems have special environmental, interfacing, and fault-tolerance requirements. These requirements are described further as follows:

a) Environmental requirements. Environmental factors such as temperature (e.g., in space exploration applications systems must operate in a temperature range of -55 to 200 degree centigrade), shock and vibration, size limits, weight limits, usually have an impact on the software requirements for a RTS. Although these factors primarily affect hardware design and implementation as well as the operator interactions with the system, they impact greatly the overall system design requirements. These requirements in turn greatly affect the software since the hardware components used must be adaptable to these environmental factors.

b) Fault-tolerant requirements. Exception handling in RTS has a special consideration due to the high reliability and critical timing requirements these systems usually have. Exceptions here are due to errors which produce program malfunction or system failure. They might originate from faults in the software or the hardware in the system itself or from external sources. Fault-tolerance requirements greatly impact and usually complicate the design of software and hardware components of the system.

c) Interfacing requirements. The devices which are typically interfaced to a RTS are many. Examples include sensors, actuators, switches, displays, communication links, D/A and A/D converters, and pulse-width-modulated controllers. These devices introduce a range of problems for RTS systems. These problems are related to exception handling strategies for interfacing devices failures. Exceptions such as sensor failure, illegal operator actions, and problem faults induced by external electrical interference are common in RTS applications. Fault detection, isolation, and exception handling algorithms greatly complicate the design and implementation of software for such systems.

1.2 Engineering Applications of real-time systems

RTS systems play a vital role in a wide range of engineering applications. Examples of RTSs are systems used for process control applications in which a process is monitored and controlled (e.g., Industrial process control, manufacturing process control, etc.). Communications, command and control applications are also examples of RTSs (e.g. real-time audio/video communication, airline reservation systems, medical centers for automatic patient care, air traffic control systems, remote bank accounting, etc.).

The following six examples illustrate the wide spectrum of engineering applications of RTSs and their specific role [Borko 91].

1. Metal industry applications (Mechanical and Manufacturing Engineering):

In this application computers are typically used in controlling processes such as casting, hot rolling, cold rolling, finishing, annealing, soaking, and other metal processing functions. The requirements of a RTS are:

a) real-time monitoring and control,

b) high availability and reliability which requires redundancy at all levels of monitoring, processing, and control

c) Communications with a broad range of distributed control systems, programmable logic devices and corporate host computers.

2. Electric utility monitoring and control applications (Electrical Power Engineering):

a) Computers are needed in this application to monitor and control plant equipment, to ensure optimal operation and safety, and to prevent costly unscheduled outages. Large quantities of coal or oil are typically consumed by the boilers in the utilities plants. A slight deviation from the optimal efficient performance of these boilers even for a short period of time can seriously impact the cost of electrical energy.

b) A supervisory control and data acquisition (SCADA) system is also used by electric utilities as an energy management system (EMS). In a SCADA system real-time tasks are utilized to collect and process raw data on the status of energy distribution networks. The processed data is saved in a real-time data base suitable for EMS applications. EMS functions range from automatic energy generation control to distribution networks state estimation techniques. These estimation techniques help predict the demand for energy and hence can shape the planning process for energy generations to overcome networks overloading problems. Gas utilities also use SCADA/EMS systems but they have fewer points to monitor, slower scanning rates and slower response time.

c) The use of EMS systems in buildings is another important applications of RTSs, where environmental variables such as temperature and air flows are monitored and controlled to optimize comfort and operating costs in large buildings.

3. Water plants applications (Civil and Environmental Engineering):

In fresh water plants monitoring and control applications, unprocessed incoming water from a reservoir undergoes a number of treatment processes such as aeration, chemical injection, flocculation, and filtration. Water is transmitted through a network of reservoirs and laterals laced with pumping stations. The water system is monitored and controlled by dual redundant computers which also serve as hosts to a large number of distributed satellite computers.

4. Aviation and space applications (Aeronautical Engineering):

a) In Aviation and Space transportation systems, computers are used monitor and control military and civilian aeroplane and space shuttle missions. RTSs also provide structural and aerodynamic testing and simulation during the development of aeroplane and space shuttles.

b) In planetary exploration applications, RTSs collect and analyze data from space exploration missions, used as real-time trainers and simulators. Unmanned flights have been used extensively for space explorations. These flights would not be possible without the complex command and control provided by the RTS consisting of a network of computers on the ground and on the flight instrumentation system in space.

5. Data Communication Applications (Electrical and Computer Engineering):

Domestic and international data communication networks use RTSs as communication processors to provide key features of packet switched networks such as high speed real-time communication, performance capabilities that can handle both on-line and background communication tasks concurrently, as well as extensive line handling for communication protocols. The advances in communication technologies provide an ever increasing complexity for data communication systems.

6. Petrochemical applications (Chemical Engineering, Petroleum Engineering):

The production of high commodity chemicals such as ethylene and prophylene is supervised and controlled by computers. These systems require high performance real-time features and should provide interfaces to regulatory control instrumentation systems and Programmable logic controllers. In petrochemical applications in general, safety related requirements are very important since a system failure may cause an environmental catastrophe.

The following table shows several applications of RTSs and the response time range for each application.

Application	Response Time Range
Speech and Audio Systems	100 ns - 10 ms
Flight Simulation	1- 10 micro sec
Robot controllers	1 ms - 10 ms
Process control systems	100 micro sec - 10 ms industrial automation
Medical Diagnosis and lab	1 ms - 100 ms automation

Table 1: Response times or deadlines of several real-time applications.

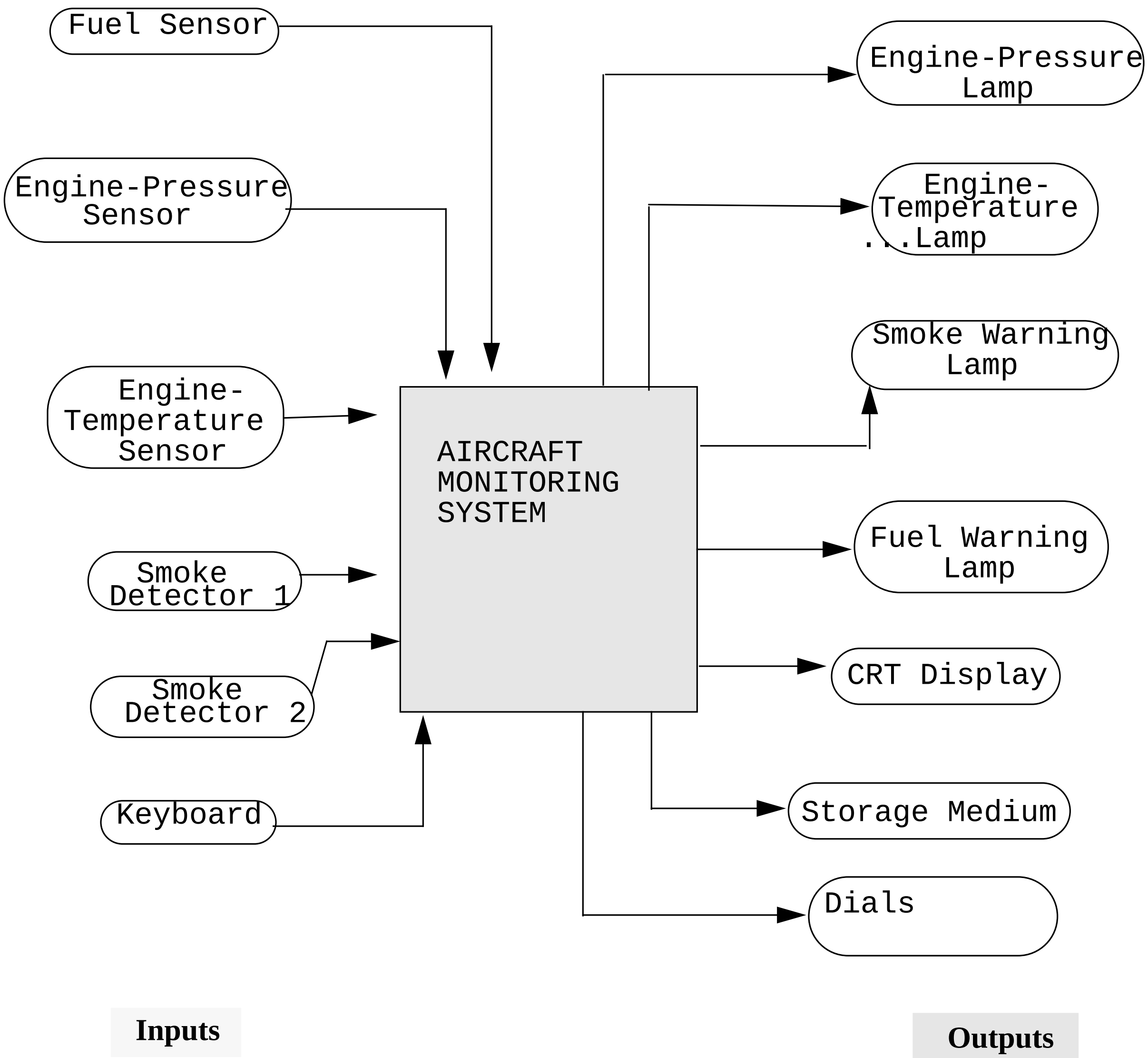
1.3 Examples of Real-Time Systems

In this section, the functional requirements of several examples of RTSs are presented. Some of these examples are used as case studies throughout the text to illustrate the various development and verification techniques.

1.3.1 The functional requirements of an Aircraft Monitoring System

The functional requirements presented in this example describes a software to be developed to monitor the flying condition of a small single engine aircraft. The software polls sensors and checks the validity of readings obtained from these sensors. It then drives a set of dials to show the readings of key parameters such as the engine temperature and pressure. The software also monitors a set of smoke detectors on board the aircraft. Warning lamps are illuminated with red when consecutive non valid readings are received from sensors or when smoke is detected.

FIGURE 1.5 Aircraft Monitoring System



The software also drives a CRT display and a keyboard. Pilot requests for sensor readings or smoke detection simulation are entered through the keyboard. Messages are displayed on the CRT to reflect warning conditions or reading requested by the pilot. The following statements represent more precise functional requirements of the aircraft monitoring system:

1. The system shall perform various aircraft monitoring and recording functions.

2. The aircraft has one engine. the engine is fitted with pressure and temperature sensors.

3. The sensors are polled by the system ar regular 1 second intervals.

4. All sensors readings shall be sent to dials, one for each sensor.

5. All sensor readings shall be tested to be within a safe working range.

6. When three consecutive readings from a sensor were found to be out of range, a lamp corresponding to the sensor is changed from green to red.

7. When a sensor fails to respond to a poll sequence, a time out signal is generated.

8. A timed out sensor shall be treated as if it had supplied an out of range reading

9. Three consecutive time outs shall cause a warning lamp to switch from green to red.

10. A number of smoke detectors are installed in the aircraft. When smoke is first detected, an interrupt is generated by the smoke detectors.

11. When smoke is subsequently no longer detected, an interrupt is generated by the smoke detectors.

12 The system shall switch a smoke warning lamp from green to red when a smoke detection interrupt occurs.

13. A sensor is installed in the fuel tank of the aircraft to provide information on the quantity of fuel remaining.

14. The fuel sensor shall be polled and read by the system at 1 second intervals.

15. The fuel sensor reading shall be passed to a dial.

16. A warning lamp shall be switched from green to red when a 10% or less fuel reading is obtained from the fuel sensor.

17. The system shall have the capability to support a CRT display and a keyboard

18. Measures calculated from the sensor data (such as rate of change of pressure, rate of fuel consumption, etc.) can be requested by the pilot using the keyboard.

19. These measures shall be displayed on the CRT display.

20. Out of limit readings from the sensors or smoke detectors which cause the warning red light to be set shall cause a warning message to be displayed on the CRT.

21 The warnings message shall be of a higher precedence over the measures requested to be displayed By the pilot.

22. The warning messages persist until acknowledged by the pilot via the keyboard.

23. When all warning messages have been acknowledged, the last request shall be displayed.

24. The key board shall be used by the pilot to request the system to simulate the smoke detection.

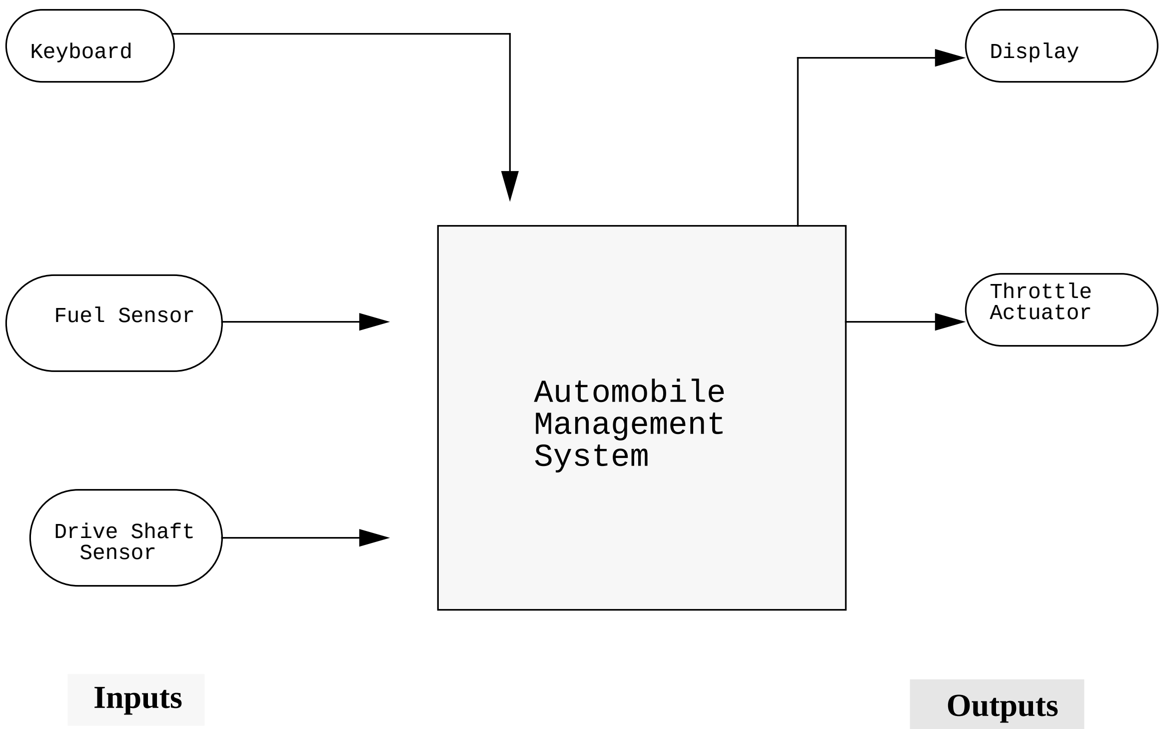
25. All smoke or no smoke interrupts shall be recorded on a magnetic medium.

26 All readings recorded on the magnetic medium shall be tagged with time at which they were received.

1.3.2 The functional requirements of an automobile management system

The following example of an automobile management system (AMS) appeared in several sources in the literature. The version presented here is similar to an example from a book on requirements analysis for real-time systems by Hatley and Pirbahi [Hatley-Pirbahi]. The system monitors and control the speed of the automobile using a cruise control function. It also monitors the mileage, average speed, and fuel consumption.

FIGURE 1.6 Automobile Management System



Messages on fuel consumption, maintenance requirement, and the average speed in a distance traveled in a certain trip. The following statements represent the functional requirements of the system:

1. The system shall consist of the following functions: cruise control; calibrate; average speed monitoring; fuel consumption monitoring; and maintenance monitoring.

2. The driver shall be allowed to enter commands for various functions of the system.

3. The cruise control commands consists of two sets. The first set consists of three commands ACTIVATE, DEACTIVATE, and RESUME which are used to start, stop, or resume the cruise control activities, respectively. The second set of commands consists of START_ACCELERATE and STOP_ACCELERATE to start and stop the acceleration function when the cruise control activities are active.

4. The cruise control function shall be activated at any time when the engine is running and the transmission is in top gear. This event is dependent on having the RUNNING and the TOP_GEAR input control signals asserted.

5. When the ACTIVATE command is selected, the software shall select the current speed and shall produce the necessary output to control the throttle position in order to hold the car at that speed.

6. The Deactivate command shall be serviced by returning the control to the driver regardless of any other commands.

7. The driver must be able to increase the speed at any time by depressing the accelerator pedal (even if the cruise function is active by simply setting the accelerator pedal far enough). When the accelerator pedal is released the system shall gain control and maintain the previously set speed.

8. The driver must be able to reduce the speed by depressing the brake pedal. Any time the brake pedal is depressed or the transmission shifts out of top gear, the cruising action becomes in-active. When the brake is released and the transmission is back in full gear and the RESUME command is pressed, the cruise action shall be reactivated and the speed shall be maintained at the previously selected speed.

9. START_ACCELERATE command shall cause the system at the cruising mode to start acceleration until a STOP_ACCELERATE command is received.

10. The system shall measure the speed and distance traveled by counting the number of pulses it receives from a sensor on the drive shaft.

11. The system shall have a calibrate capability to calibrate the speed and distance measurements since measurements based on drive shaft rotation are affected by tire size and wear. The calibration is to take place by driving a pre-measured mile and counting the number of shaft rotations from the sensor. The calibration function is activated by a START_CALIBRATE command and is terminated by a STOP_CALIBRATE command.

12. The system shall provide the driver with an average speed indication on its display. When the driver enters a SART_TRIP command, and later, whenever he enters an AVERAGE_SPEED request, the system shall display the average speed of the car for all the time the engine has been running since the START_TRIP command. This shall continue until the next SART_TRIP.

13. The driver shall be able to enter the quantity of fuel added using a full numeric keyboard. The system shall calculate and display the fuel consumption over the period since the last fill up.

14. The system shall monitor the Car’s mileage and notify the driver of required maintenance according to the following schedule: oil and oil filter change every 5,000 miles; air filter change every 10,000 miles; major service every 15,000 miles.

15. The appropriate maintenance message shall be displayed intermittently 250 miles before each required service. The message shall then appear continuously 50 miles before the required service and will remain until the driver enters a SERVICE_COMPLETED command.

The above set of requirements were the original set presented by the customer initiating a request for proposals in this example. The following set represents a refinement of the above requirements through meetings or correspondents between prospective developers responding to the request for proposals and customer.

16. The system shall control the car through an actuator attached to the throttle. This actuator is mechanically in parallel with the accelerator pedal mechanism such that whichever demanding greater speed controls the throttle. The system shall drive the actuator by means of an electrical signal having a linear relationship with the throttle deflection. A signal level of 0 volts corresponds to setting the throttle closed (this corresponds in manual control to releasing the acceleration pedal), and a signal level of 8 volts will set the throttle to fully open (this corresponds to depressing the acceleration pedal all the way).

17. The system shall serve as the control part of a servo loop where the engine is the feed forward part. For a smooth servo operation, the system shall update the output to the throttle at least once every second. When the current speed is more than 2 m.p.h. above the selected the speed, the system shall completely release the throttle (this situation would happen when driving down hill). At any speed below this, the system shall drive the throttle to a deflection proportional to the error or until the throttle is fully opened (when a steep uphill is encountered).

18. The system shall avoid rapid increases in acceleration, therefore, the actuator must never open faster than to traverse its full range in 10 seconds (it may close at any rate since the car just coasts when the throttle is closed). When accelerating in normal gradients, the system shall measure the acceleration and hold it at 1 m.p.h./sec. In abnormal gradients, when the acceleration reaches 1.2 m.p.h./sec (down hill), the throttle should be closed, and when it reaches 0.8 m.p.h./sec (up hill), the throttle should be fully opened.

19. The system shall have a full numeric keyboard, and a CRT display.

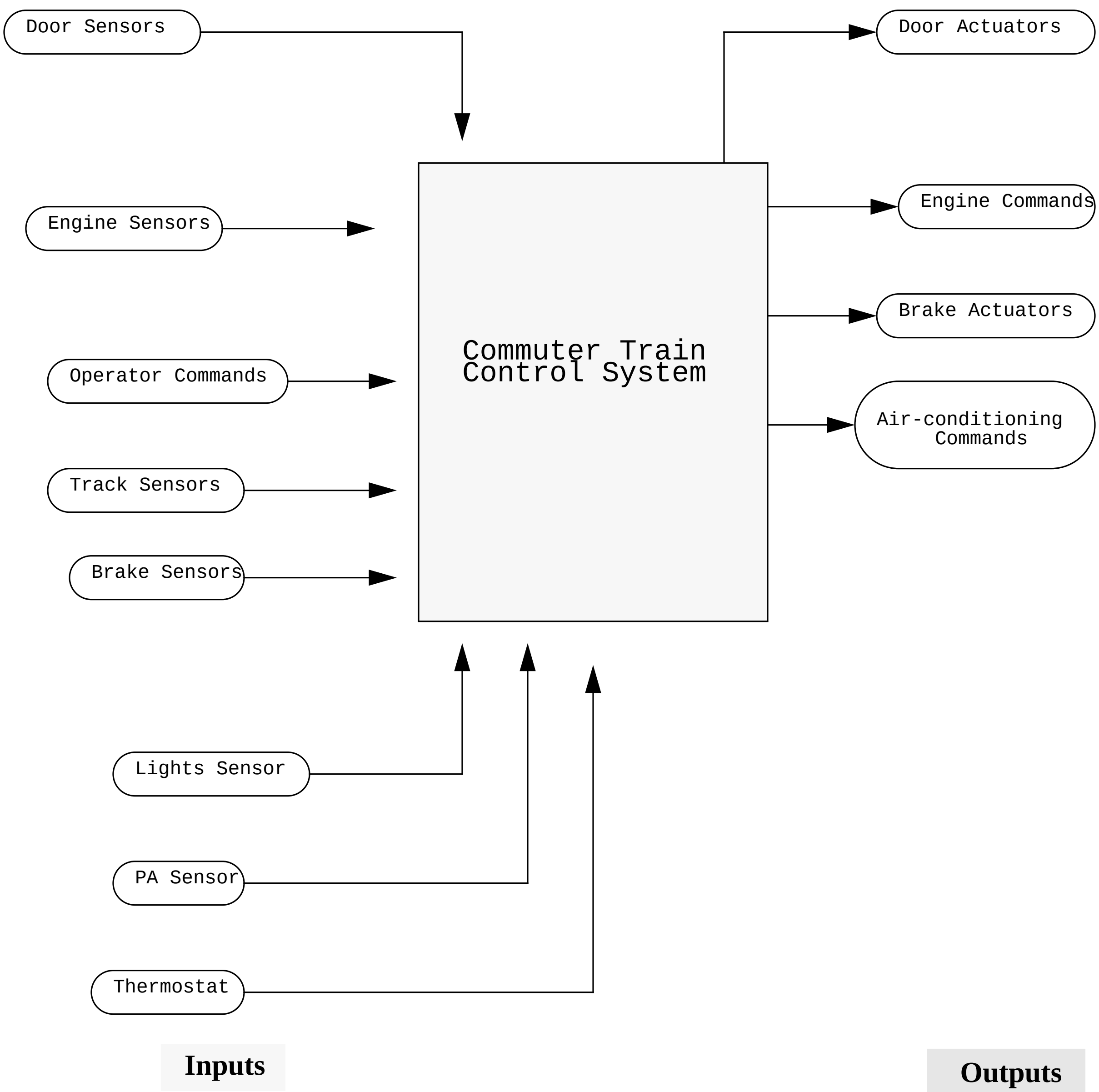
1.3.3 Automated Commuter Train Control System

The following paragraphs describe the functional requirements of an on-train control software component for an Automated Commuter Train System. The system is to operate throughout the metropolitan area of a large city.The system is managed using a central computer and a maintenance computer. Each train is fully automated and contains an on-board computer which communicates with the central computer and the maintainence computer. The trains in this system will be powered via a third rail. At no point will pedestrians or automobiles have access to the tracks. A communications network running parallel to the track will link the Central computer to each train. Messages will pass from train to network (or network to train) via track-side devices located on each track section. The Central computer system will coordinate scheduling data for all trains. It will send an appropriate itinerary to each train in the automated commuter train network. The Central Computer system will be capable of changing the schedule at any time, and will disseminate these changes throughout the system, via the communications network.The Central Computer system will provide information to the on-train computer about obstacles in the train's path,

such as a service vehicle or another train. this system will also warn the train about traffic signals and track switches, as well as turns and gradient changes in each section of track. The Central

Computer system. will monitor the positions of all vehicles on all track sections. The Maintenance Computer system will keep complete and up-to-date records on all trains, communications hardware, and other components of the automated commuter train network. The Maintenance System Database will store service and repair histories for all these components. Figure 1.7 shows the inputs and outputs of the communter train on board control software, and the following paragraphs lists the specific functional requirements for the software.

FIGURE 1.7 Automated Commuter Train



1. The on-train control system shall direct and monitor train acceleration, deceleration, stopping patterns, door opening and closing, lighting, climate control, and announcements to passengers. The operator's only charge will be to override the control system in the case of an emergency.

2. Train runs shall be conducted by the software according to a precise itinerary received from the central computer. Trains will stop and start at the appropriate locations and times.The system shall record the information for every run. This information shall be sent in a report to the Maintenance Database at the end of each nun.

3. The on-train computer software will receive (from the central computer) and react to information on obstacles in the train's path, such as a service vehicle or another train. It will also recieve and react to information on traffic signals and track switches, as well as turns and gradient changes in each section of track.

4. The automated commuter train shall operate in three modes: a Local mode in which it will stop at every stop, an Express mode in which it will go directly to a designated stop and bypass all stops in between, and a Request-Only mode in which the train will stop only if prompted by a passenger.

5. In Request-Only mode, passengers on the train will prompt the train to stop using the Next Stop cord. Passengers waiting at the next stop will be able to stop and approaching train using a button provided at the station.

6. Each train will be equipped with a self-monitoring engine and with the following automated on-train systems: doors, brakes, lights, public address, and climate control. In the event of a malfunction, manual control will be enabled and monitored by the on-train control system.

7. Each train will have four separate brake systems: phase I brakes for general speed regulation, phase II brakes for stopping the train, emergency brakes for stopping the train in the event of failure by either phase I or II brakes, and safety brakes for extra security when the train stops to load and unload passengers.

8. Each train will have two separate door systems, left and right. Itineraries will indicate which side to open at each stop. When doors open at a stop, an on train timer will be activated. The timer will tell the system when it is appropriate to close the doors, given the stop and the time of day. Doors will not close fully until they are unobstructed.

9. In the event of equipment failure or accident, the on-train control system shall notify the Maintenance system automatically, and shall provide precise data about the problem. Appropriate data shall be made available to maintenance crews before they make service calls..

1.3.4 REQUIREMENTS FOR A TRAFFIC LIGHT CONTROL SYSTEM

The traffic light control example is the simplest example of a RTS presented in this Chapter. It exhibits all the chrarcterstics of a RTS discussed in section 1.2

This example is typical of many traffic intersection control systems used in any intersection involving pedestrian crossing. The system can be used with simple intersections involving one way street interrupted by crosswalk. It can also be used with typical more complex intersections involving two two-way streets with left turn lanes. The following paragraphs lists the specific functional requirements for this system.

1. The system shalll control all the pedestrian and traffic lights at a given intersection. The intersection could be as simple as a one-way street interupted by a crosswalk. The intersection could be as complex as two two-way streets with left-turn lanes.

2. A given traffic light will normally be green for G seconds. yellow lights will last Y seconds. A red light will remain red for R seconds before the traffic is allowed to change direction. For example, R seconds after the north-southstreet turns red the east-west light will turn green.

3. Traffic sensors may be present to detect car arrivals and departures.To better service heavy traffic, if M cars arrive while the light is red, the subsequent green light will persist P seconds or until all M cars have departed, whichever comes first.

4. Pedestrian crossing request buttons may be present. After a button is pressed the pedestrian should wait no longer than C seconds for permission to cross. Only crossing request received while the light is red need be serviced.

5. Any conflicts between pedestrian requests and traffic volume should be decided in favor of the pedestrian.

6. At some time during the day, the intersection will automatically suspend normal service and its lights will flash red or yellow.

7. The above variables G, Y, R, and C are specified and along with other specific intersection and lane information during the initialization process of the control system.

1.4 Exercises

1. Specify whether or not the following are soft or hard real-time systems:

a) A police search and retrieval database system which provides information on stolen automobiles.

b) An Automatic Teller Machine

c) A Postal Service mail-processing system which provides scanning and distribution of mail items into appropriate bins.

d) An FBI automated fingerprint identification system serving the whole nation.

f) A An automated gasoline station system which controls the pumps, cash receipts, sales figures, deliveries, etc.

g) A microcontroller-based automobile ignition and injection system.

H) An automated airline seat-reservation system.

2. Give examples of specific real-time systems in Medical applications or in the communication systems and discuss whether they have hard or soft deadlines.

3. From the functional requirements of an aircraft monitoring system presented in section 1.3.1,

a) develop a table containing the timing deadlines for this system. The table should have the following columns:

INPUT EVENT OUTPUT EVENT RESPONSE TIME,

where the input column will list an input to the system such as pilot request, the event will specify a particular value for this input (E.g.,a particular request) to be true, the output column. Specifies an output produced by the system for the given input request, and the event is a particular condition for this output.

b) Specify the synchronous and asynchronous external input events which the software need to respond to as given in the above table in a).

c) Discuss the real-time characteristics of this example in terms of embeddedness, concurrency, and reliability.

4. Repeat the steps in problem 3 above for the automobile management system example presented in section 1.3.2.

5. Repeat the steps in problem 3 above for the automated train example presented in section 1.3.3.

6. Discuss why real-time systems are more difficult to verify than non-real-time systems.

8. Discuss the function of In-Circuit Emulator (ICE) devices in testing embedded real- time systems.

1.5 REFERENCES

[Borko 91] Borko Furht et al, Real-Time Unix Systems: Design and Application guide, Kluwer academic, 1991.

[BURNS&WELLINGS 90] A. Burns, and A. Wellings, Real-Time Systems and their programming languages, Addison Wesley, 1990.

[Bennet 94] S. Bennett, Real Time Computer Control, An Introduction, Prentice Hall, 1994.

[Cooling 95] J.E. Cooling, Software Design for Real Time system, Chapman and Hall, 1995.

[Levi 90] Shem-Tor Levi, A.K. Agrawala, Real-Time System Design, McGraw Hill, 1990.

[Hatley-Pirbahi].