EE66C Real Time Systems

updated: 7th July 2006– CLR

Course Code:	EE66C
Course Title:	Real Time Systems
Course Dept:	Department of Electrical and Computer Engineering
Course Instructor:	Cathy Radix (cradix@eng.uwi.tt),

Aims Provide an understanding of real time systems (RTS) theory, and the practical issues involved when applying RTS principles to digital control systems.

Learning Objectives

- Given a description of an applied real-time system, identify the real-time characteristics of the application, and explain how they have been addressed.
- For control-related scenarios, utilise available techniques for
  1. translating application-based specifications into (a)periodic/deadline specifications;
  2. mapping ideal priority levels onto available OS priorities;
  3. alleviating the constraints placed on the real time systems by hardware and inter-process communications.
- Given the (a)periodic/deadline specifications for a real time control system, produce a schedule for a cyclic executive, ideal DM, RM or EDF systems (if possible), and demonstrate it's validity by applying utilization and response-time based criteria.
- Given a real time operating system specification/standard, identify the real-time specific features, and contrast the OS with an ideal "open" real-time OS.
- Given a real time system scenario, produce a graphical representation of the system/solution using UML and related techniques.
- Analyse an existing real time system in terms of its reliability and fault tolerance, and recommend changes which would improve performance in either of these areas.
- Propose a solution for a real-time scenario involving hardware control, and implement the solution utilising a real-time operating system kernel.

Course Overview Developments in real time systems theory could be applied to improve the performance of computer-based hardware controllers; in practice they rarely are. This is primarily due to the number of compromises which need to be made to the "ideal" real time system in order to address the needs of hardware control. In this course, the characteristic requirements of real-time applications, and the constraints these (and other) factors place on the application of real-time systems theory are examined. Typical compromises and design techniques are identified. The intent is to provide the student with the ability to apply real-time system theory, and assess whether it will be of benefit in a particular digital-control application.

Prerequisites None: however a basic understanding of statistics, computer architecture, operating systems, and digital control theory is required.
Programming experience, particularly using Assembler/C/C++ would be an asset.
Relevant undergraduate courses include: EE25M, EE26A, EE33A, CS31A, EE39B
Course content will complement: EE66U, EE66V, EE66L.

Weighting 3 credits; 3 lecture hours/week for 12 weeks; additional 9-12 lab hours.

Evaluation & Ungraded Assignments :

Type of Evaluation	Description	%
End of Semester exam	Three hours long. Answer all questions.	50%
Mini-Project	RT Scenario: design & implement [Rcv Wk 8; Present Wk 12]	30%
Presentations	Academic paper review [Rcv Wk 3/5; Present Wk 7]	10%
Problem Sets	Five assignments will be provided [Due Wk 5,7,9,11,13]	10%

Lecture Topics

Section	# Weeks	Topic
1	2	Real Time Overview
2	1	RT Theory and Digital Control
3	2	Operating System Support for Concurrency
4	2	RT Specification, Design, Modelling
5	2	Performance, Reliability and Fault Tolerance
6	2	RT Scenario/System Development
	1	Paper Presentations

References

[BW01] Alan Burns and Wellings. *Real-Time Systems and Programming Languages*. 3rd edition, 2001.

[Dou99] Bruce Powell Douglass. *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns*. Object Technology Series. Addision-Wesley, 1999.

[KS97] C.M. Krishna and Kang G. Shin. *Real-Time Systems*. McGraw-Hill, 1997.

[Liu00] Jane W.S. Liu. *Real-Time Systems*. Prentice-Hall, 2000.

[RIoD02] KTH Royal Institute of Design. Real-time computer control systems course. http://www.md.kth.se/RTC/RTCC, Spring 2002.

[Son95] Sang H. Son, editor. *Advances in Real-Time Systems*. Prentice-Hall, 1995.

Laboratory Tools

PC/PC compatible system(s) with 1 terminal/PC per student running: a real-time operating system(eCos, μCOS-II, IRMX 2.2) with an appropriately configured C compiler; the Cheddar real time simulator; MATLAB 6 with the Real Time Kernel toolbox installed; a UML CASE tool capable of supporting all 9 Case diagrams, as well as timing diagrams.

updated: 7th July 2006– CLR

Course Outline

1 Real Time Overview

Learning objectives:

- Given a description of an applied real-time system, identify the real-time characteristics of the application, and explain how they have been addressed.
  - Identifiable characteristics of a Real Time system
    - Large and Complex
    - Concurrent control of separate system components
    - Facilities to interact with special purpose hardware
    - Guaranteed response times
    - Extreme reliability
    - *Efficient implementation*
  - Appropriate use/comprehension of real-time scheduling terminology
- Given the (a)periodic/deadline specifications for a real time control system, produce a schedule for a cyclic executive, ideal DM, RM or EDF systems (if possible), and demonstrate it's validity by applying utilization and response-time based criteria.

Readings

- Slides from [BW01, www.cs.york.ac.uk/rts/RTSBOOKThirdEdition.html].
- Case studies from [Liu00, Chapters 1,2]
- Graphs from: [Eng02]

Review

Read [Bur95], paying particular attention to the response time prediction models which may be used to determine system feasibility in a variety of cases.

References

[Bur95] Alan Burns. *Pre-emptive Priority Based Scheduling: An Appropriate Engineering Approach*, chapter 10, pages 225–248. Prentice-Hall, 1995.

[BW01] Alan Burns and Wellings. *Real-Time Systems and Programming Languages*. 3rd edition, 2001.

[DoAC02] Lund Institute of Technology Dept. of Automatic Control. Real-time systems course – FRT031. http://www.control.lth.se/~kurstr/, Spring 2002.

[Eng02] Jakob Engblom. Effects of branch predictors on execution time. Technical Report 2002-013, Dept. of Information Technology, Uppsala University, P.O. Box 337, SE-751 05 Uppsala, Sweden, April 2002. Author: jakob@it.uu.se / http://www.docs.uu.se/~jakob Document: http://www.it.uu.se/research/reports/2002-013/2002-013-nc.pdf.

[KS97] C.M. Krishna and Kang G. Shin. *Real-Time Systems*. McGraw-Hill, 1997.

[LaP93] Phillip A. LaPlante. *Real-Time Systems: Design and Analysis, An Engineer's Handbook*. IEEE Press, 1993.

[Liu00] Jane W.S. Liu. *Real-Time Systems*. Prentice-Hall, 2000.

[oY96] University of York. Cs final exam: Real time systems and their programming languages. http://www.cs.york.ac.uk/rts/RTSbookThirdEdition/rtse96.ps, 1996.

[oY99] University of York. Cs final exam: Real time systems and their programming languages. http://www.cs.york.ac.uk/rts/RTSbookThirdEdition/rtse99.pdf, 1999.

[oY00] University of York. Cs final exam: Real time systems and their programming languages. http://www.cs.york.ac.uk/rts/RTSbookThirdEdition/rtse00.pdf, 2000.

[oY01] University of York. Cs final exam: Real time systems and their programming languages. http://www.cs.york.ac.uk/rts/RTSbookThirdEdition/rtse01.ps, 2001.

[oY02] University of York. Cs final exam: Real time systems and their programming languages. http://www.cs.york.ac.uk/rts/RTSbookThirdEdition/rtse02.pdf, 2002.

[Sea04] Frank Singhoff and Jerome Legrand et al. The cheddar project. http://beru.univ-brest.fr/ singhoff/cheddar/, September 2004. Release 1.3p4 patched version.

Review Questions

1. [oY96, Question 1] The following formulae are used to analyse a fixed priority system of tasks to determine if they will meet their completion time requirements:

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil . C_j$$

$$R_i \leq D_i$$

   (a) Define all the terms used in the above equations.

   (b) How would the C_i term be determined?

   (c) How would the B_i term be determined?

   (d) How could the first equation be solved?

   (e) What is release jitter, and how could the first equation be modified to include this behaviour ?

   (f) How can the following task set be optimally scheduled (using fixed priority scheduling)? Is this task set schedulable?

	T	C	B	D
Task_1	8	4	2	8
Task_2	10	2	2	5
Task_3	30	5	2	30

2. (a) [oY00, Question 6(i)] Briefly compare and contrast the use of recovery blocks and exception handlers for programming fault tolerance.

   (b) [oY00, Question 6(iii)] Illustrate how the response time equation is used by calculating the worst-case response times (R) of the following process set (assuming priorities have been assigned in rate monotonic order).

Process	Period	Computation Time	Deadline	Response Time
	T	C	D	R
P1	10	3	10	?
P2	20	8	20	?
P3	25	4	25	?

   (c) [oY00, Question 6(iv)] If a set of processes is subject to a single fault in one of the processes (that is tolerated by either the execution of a recovery block or an exception handler), consider how the response time equation can be modified to accommodate the extra computation. Would the above process set remain schedulable if there were a single fault and each of the processes had an exception handler? Assume the worst-case execution times for the handlers are 2, 3, and 1 respectively for the three process P1, P2 and P3.

3. [oY02, Question 5 (iii)] The following table gives some of the parameters for three processes [scheduled using a fixed priority pre-emptive scheduling]. All time values are in milliseconds.

Process	Period	Deadline	Computation Time	Response Time
PA	20	20	C_A	16
PB	25	12	C_B	10
PC	50	50	C_C	38

   (a) Have these processes been scheduled in deadline monotonic priority ordering?

   (b) Calculate what the missed computation time parameters are (C_A, C_B, C_C).

4. (a) [oY99, Question 2(i)] Define what is meant by blocking time. Compare and contrast the original ceiling priority protocol with the immediate ceiling priority protocol for single processor systems.

   (b) [oY99, Question 2(ii)] A program consists of five tasks, A,B,C,D,E (these are listed in priority order with A having the highest priority), and six resources R1, ... R6 (protected by semaphores implementing the Original Priority Ceiling Protocol). The resource accesses have worst-case execution times given:

R1	R2	R3	R4	R5	R6
10ms	30ms	20ms	60ms	50ms	40ms

   Resources are used by the tasks according to the following table:

Task	Uses
A	R3
B	R1,R2
C	R3,R4,R5
D	R1,R5,R6
E	R2,R6

   Calculate the blocking time for each task in the above table.

5. [oY02, Question 1 (ii)] A POSIX application consists of three threads with the following characteristics:

Thread	Period (ms)	Computation Time (ms)	Priority
A	4	1	5
B	10	2	3
C	20	2	2

   All three tasks share a critical instant at time 0. To handle aperiodic activity, the program has a sporadic server thread with the following characteristics:

   - Replenishment period = 5 milliseconds
   - Budget = 2 milliseconds
   - High priority = 4
   - Low priority = 1

   Assuming that

   - aperiodic events arrive at times, 3, 4, 5 and 6 milliseconds after a critical instant, and
   - to handle each event requires 2 milliseconds of execution time from the sporadic server,

   illustrate the execution of the threads between time 0 and time 20 milliseconds. When showing the sporadic server executing, indicate at what priority level it is running.

6. [oY01, Question 4] Three periodic tasks (A, B and C) monitor environmental inputs and log these readings in a central store. A further periodic task (D) reads the store and updates an operator's screen. The final task (E) is sporadic, it is released for execution if two of the input readings are too high. It then sounds a warning bell for 1 minute. Task E will then wait 1 further minute before it calls back into the store. Hence, the minimum interarrival time for E is 120 seconds. The table gives the temporal characteristics of the five tasks. The units are tenths of a second. One unit of time is spent in the store by each task (but this time has already been added into the computation time for each task).

Task	Period	Deadline	Computation Time
A	15	15	1
B	18	18	2
C	12	12	3
D	35	35	10
E	1200	5	3

   (a) What would the optimal priority assignment be for these five tasks if preemptive priority-based scheduling is used.

   (b) Using response time analysis, decide if this system of tasks would meet all its deadlines.

7. (a) [KS97, Question 3.1] Construct a set of periodic tasks (with release times, [deadlines], execution times, and periods), which can be scheduled feasibly by the EDF algorithm but not by the RM [or DM] algorithms.

   (b) Describe two or more techniques (apart from period transformation) which may be used to make a digital system schedulable.

   (c) [KS97, Question 3.5] Consider a set of five tasks with the following characteristics:

Task	Worst Case Execution Time	Average Execution Time	Period
1	30	5	100
2	10	5	130
3	20	15	140
4	80	10	140
5	10	10	200

   Tasks 1, 3, 5 are critical but Tasks 2,4 are not. Carry out a period transformation for this task set to ensure that the critical tasks always meet their deadlines.

8. (a) [DoAC02, Question 2, Dec 2001 Exam] Automatic garbage collection is a nice feature in Java. However, for realtime applications garbage collection may cause problems. What are the problems?

   (b) [LaP93, Based on Questions 9.7/8] A 16 bit computer has instructions that require two bus cycles, one to fetch the instruction from memory and one to fetch the data from memory. The bus is 16 bits wide and each bus cycle takes 250 nanoseconds. Basic data processing instructions take a total of 500 nanoseconds to execute (i.e. the internal processing time is negligible). The processor also has special instructions which support fixed and floating point manipulation. Fixed point instructions take 6 microseconds, and floating point instructions take 60 microseconds.

   A program to be implemented on this computer requires the multiplication of a 32 bit A/D result by a real constant. What are the relative merits of a lookup table, a fixed point implementation and a floating point implementation respectively?

(c) List (at least) 4 enhanced features of modern processors/computers which can adversely affect the prediction of worst case execution times.

(d) Comment on the potential effects of cache memory and DMA on the speed of the program, and the predictability of worst case execution times.

9. Discuss the following topics:

(a) the differences between RM scheduling and another scheduling algorithm of your choice, (i.e. priority assignment, maximum utilisation, optimality, stability)

(b) how blocking time can be minimised through priority inheritance,

(c) how asynchronous events can be accommodated in scheduled systems,

(d) contributing factors to the overestimation of worst case execution time

(e) common features of real-time systems which do not appear in the "list"

(f) additional issues involved in multiprocessor scheduling

10. Cheddar is a real time scheduling simulator mainly designed for educational purpose. This program provides services to automatically check temporal constraints of real time tasks. Cheddar is developed and maintained by the LIMI/EA 2215 Team, University of Brest. [Sea04]

This version of Cheddar will allow you to explore the schedulability of tasks on processors running preemptive or non-preemptive RM, DM, HPF, EDF or LLF scheduling.

(a) Install/run the Cheddar simulator on your system.

(b) Open the file "sample" (do not type in an extension). This opens the project files sample.proc (definition of the processor) and sample.tsk (definition of the tasks)

(c) Use the list functions (View/List tasks and View/List Processors) to examine the project settings. The "sample" files are for a single processor named "cpu" which is running a pre-emptive RM scheduler. The tasks are as follows:

Task	Execution Time	Period	Deadline	Manually Assigned Priority
taska	3	10	10	1
taskb	5	15	15	2
taskc	10	35	35	3

(Note: the priority and deadline information may be ignored by the RM scheduler)

(d) Check the system schedulability (Tools/Check scheduling). The software will report the cpu utilization and task response times in the lower window.

(e) Draw the timing diagram (Tools/Draw scheduling). The software will generate the chart in the upper window, and report the number of preemptions, and any missed deadlines in the lower window.

(f) Clear both windows (Tools/Clear Workspace) and investigate the effects of changing the scheduling algorithm used by the processor (Edit/Update processors). i.e. Repeat steps 10d,10e for non-preemptive RM, and preemptive/non-preemptive versions of all other scheduling algorithms. Do the results concur with your understanding of the algorithms?

2 RT Theory and Digital Control

Learning objectives:

- For control-related scenarios, utilise available techniques for
  1. translating application-based specifications into (a)periodic/deadline specifications;

Readings

- Slides based on:
  - [WsÅ] Computer Control: An overview
  - [Lei92, Chapters 5–7] Digital Control algorithms, Elements in the control loop, Tutorial case histories and Review Questions.
  - Slides from: Timing Problems in Control Applications [RIoD02, /Material01/Lectures/RTCC_timing_properties.pdf]
  - Slides from: [DoAC02, Lecture 5: http://www.control.lth.se/k̃urstr/L5_02_slides4.pdf]
  - [ÅBE+99] Integrated Control and Scheduling
- [SLSS96, Section 3] On Task Schedulability in Real-Time Control Systems

Assignment A– 2%

1. Identify at least 2 assumptions that are made in conventional discretisation of an analogue PID controller. *2 marks*
2. There are several rules-of thumb used to select sampling time for a controller. For ONE of these rules, describe how it is applied, explain the rationale for the rule, and identify possible consequences/implications for real-time performance. *3 marks*
3. Given a motor control program, and the length of time it takes to operate on a particular processor, explain how you would determine the maximum number of motor controllers that could safely run in a preemptive manner on a single processor. State any assumptions that you make about the processor/motors/program. *5 marks*

Review

Read/summarize at least one of the following papers:

- [RHS97, Streamlining Real-Time Controller Design: From Performance Specifications to end-to-end timing constraints] Pay particular attention to the way in which the end-to-end timing constraints are related to the control description parameters for step/ramp response.
- [LSA+00, Performance Specifications and Metrics for Adaptive Real-Time Systems] Pay attention to the way in which control theory is applied to scheduling admission/rejection by specifying the transient behaviour of the target utilization and the miss-ratio.
- [MFRF01, Jitter Compensation in Real-Time Control systems] Pay attention to the strategies for adjusting control parameters to accommodate timing irregularities.

References

[ÅBE+99] K.-E. Årzén, B. Bernhardsson, J. Eker, A. Cervin, K. Nilsson, P. Persson, and L. Sha. Integrated control and scheduling. Technical Report TFRT-7582, Department of Automatic Control, Lund Institute of Technology, August 1999.

[Ben88] Stuart Bennett. *Real-Time Computer Control: An Introduction*. Prentice-Hall, 1988.

[Cer00a] A. Cervin. The real-time control systems simulator. http://www.control.lth.se/ anton/rtkernel/, April 2000. Release 1.04.

[Cer00b] A. Cervin. The real-time control systems simulator reference manual. Technical Report TFRT-7592, Department of Automatic Control, Lund Institute of Technology, April 2000.

[DoAC02] Lund Institute of Technology Dept. of Automatic Control. Real-time systems course – FRT031. http://www.control.lth.se/k̃urstr/, Spring 2002.

[Lei92] J.R. Leigh. *Applied Digital control: Theory, Design & Implementation*. Prentice Hall, second edition, 1992.

[LSA+00] Chenyang Lu, John A. Stankovic, Tarek F. Abdelzaher, Gang Tao, Sang H. Son, and Micheal Marley. Performance specifications and metrics for adaptive real-time systems. In *Proceedings of the IEEE Real Time Systems Symposium*, 2000.

[MFRF01] Pau Marti, Gerhard Fohler, Krithi Ramamritham, and Josep M. Fuertes. Jitter compensation in real-time control systems. In *Real-Time Systems Symposium*, London, UK, December 2001.

[MT98] William C. Messner and Dawn M. Tilbury. *Control Tutorials For MATLAB And Simulink: A Web-Based Approach*. Addison-Wesley, 1998.

[RHS97] Minsoo Ryu, Seongsoo Hong, and Manas Saksena. Streamlining real-time controller design: From performance specifications to end-to-end timing constraints. In *Proceedings of the IEEE Real Time Systems Symposium*, 1997.

[RIoD02] KTH Royal Institute of Design. Real-time computer control systems course. http://www.md.kth.se/RTC/RTCC, Spring 2002.

[SLSS96] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control systems. In *Proceedings of the IEEE Real Time Systems Symposium*, 1996.

[UC97] Universiy of Michigan and Carnegie Mellon. Control tutorials for matlab. http://www.engin.umich.edu/group/ctm/, August 1997.

[WsÅ] Björn Wittenmark, Karl Johan ström, and Karl Erik Årzén. Computer control: an overview. – http://www.control.lth.se/ kursdr/ifac.pdf.

Review Questions

1. [Lei92, Question 5.28] A designer of industrial control systems must frequently convert requirements posed in general commercial terms (e.g. improve product consistency) into forms that are directly applicable in control design (e.g. achieve a particular closed-loop bandwidth)

   For the following applications suggest usable control criteria equivalent to the original requirement:

   (a) A mass of inertia J has to be rotated from an arbitrary rest position to any other angular rest position within 180° in not longer than T_1 seconds.

   (b) A steerable aerial must not be moved out of alignment by more than e° by wind forces

   (c) Cans are to be filled with 1kg of product to ensure negligible possibility of underweight but with minimum 'give-away'

   (d) The thickness correction scheme for a flat product that is made in batches must satisfy the requirements:

   - Product to be within thickness specification within l meters of the start of each batch
   - Product to be kept within thickness specification during the batch despite the effect of random disturbances

   (e) An endless strip of plate glass of width w meters moves forward at a speed of s meters/second. It is to be cut into lengths of $l \pm e$ meters by a diamond cutter that must be arranged to traverse the strip at right angles using two servomechanisms.

2. [Lei92, Based on Question 6.10] What are the principal parameters, from a control systems viewpoint, that characterize each of the following: a sensor; an actuator; a D/A convertor; an A/D converter; a sample and hold;

3. Based on [Lei92, Question 5.15]

   (a) Explain why, in the digital control of a continuous process, it is important to choose a sampling interval that matches the application.

   (b) Describe briefly some of the approaches that are available for determination of a suitable sampling interval.

   (c) Discuss how the actual sampling interval is influenced by(implemented using) code running on a real-time system.

   (d) Identify (at least two) techniques appropriate for period/task transformation in a real-time control system.

4. A pre-emptive priority based scheduler, may be used to implement an engineering control system. The choice of task period, presence of computation delay, and the development of timing jitter all degrade the performance of an implemented control system relative to the "ideal" designed control system.

   (a) Using an example, explain how delay jitter and period jitter arise in a pre-emptive priority based control system. Highlight at least **one** strategy which may be used to minimise jitter in such a system.

(b) A continuous-time motor speed controller was derived for a motor. The step response of the "ideal" controlled system is shown in Figure 1. Choose an appropriate value of sampling time for a discrete implementation of the controller, and justify your choice.

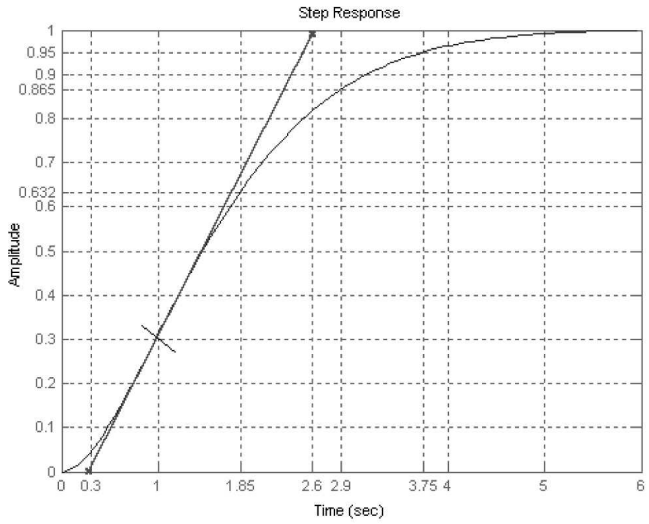


Figure 1: Controlled system step response for Section A 4b

(c) Multiple digital controllers (each for a different motor) were implemented for a suitably selected sampling interval, on an embedded system running a RTOS. The actual and ideal responses for a single motor are shown in Figure 2. Suggest at least two possible reasons for the observed differences in the control response.

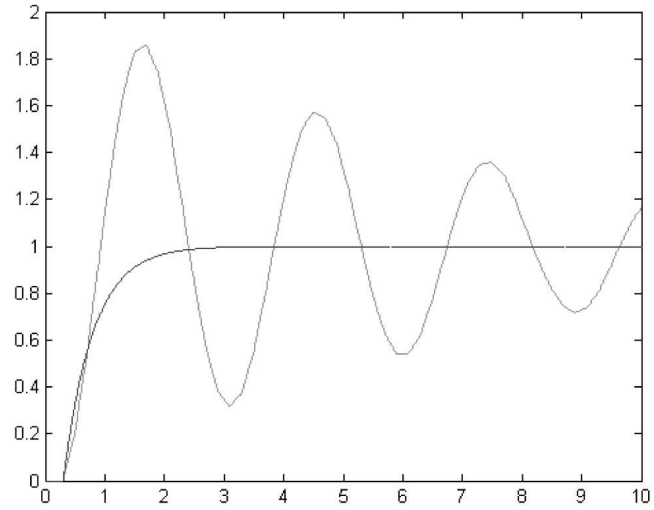


Figure 2: Controlled system step response for Section A 4c

(d) You have been asked to implement a controller for a temperature control task. The analogue controller should be designed using the Ziegler Nichols open loop method. The system step response displays deadtime $\mathbf{a} = 5$ seconds, rise time $\mathbf{b} = 10$ seconds for a process gain $\mathbf{G} = \frac{(0.8-0.9)}{(25-35)} = 0.01$.
The equation for the Ziegler Nichols controller follows:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int^{t} e(s).ds + T_d\frac{de(t)}{dt}\right]$$

	K	T_i	T_d
P	$\mathbf{\frac{b}{Ga}}$		
PI	$\mathbf{\frac{0.9b}{Ga}}$	**3a**	
PID	$\mathbf{\frac{1.2b}{Ga}}$	**2a**	**0.5a**

Justify your choice of P, PI, or PID controller, and write down the final controller equation.

(e) Your task is to implement the controller from 4d using a C program on a microprocessor, whose kernel supports time and delay functions with resolutions of 0.1 seconds, and which is running other tasks simultaneously with the temperature control task. Choose from the code in Figure 3: will you use `R` or `Q` to update the control signal? will you use `W` or `Z` to time the task? Briefly justify your choices.

```
void update() // Version R
{
    e=y-oldy;
    esum=esum+p*e;
    u=K*(e+esum/Ti+Td*e/p);
    oldy=y;
}//-- WCET 0.040 seconds
```

```
void update() // Version Q
{
    e=y-oldy;
    u=u+K*((e-olde)+p*e/Ti+Td*(e-olde-oldde)/p);
    oldy=y;
    olde=e;
    oldde=e-olde;
}// -- WCET 0.044 seconds
```

```
void task()// Version W
{
    while(true)
    {
        update();
        delay(p);
    }
}
```

```
void task() // Version Z
{
    while(true)
    {
        start=timenow();
        update();
        if (timenow()<start+p) delay(p-start);
    }
}
```

Figure 3: Code alternatives for 4d

5. (a) An additional task is added to the system in example 3.1 in [SLSS96]:
   unit 6 $\beta_6 = 0.8$ $C_6 = 35$ $f_{m6} = 2$ $w_6 = 6$
   Using the methods proposed, determine the optimal frequencies for a feasible EDF schedule. Test to see if the tasks can also be scheduled using RM scheduling.

   (b) For the example 3.1 in [SLSS96]:

   i. Construct a cyclic executive schedule based on the specifications in Table 2.

   ii. Determine the % CPU time the cyclic executive utilizes and the performance index ΔJ for the cyclic executive.

   iii. Comment on the % utilization and performance of the cyclic executive relative to RM/EDF scheduler.

6. (a) [Lei92, Question 6.7] State the two common digital versions ((1) absolute and (2) incremental) of the PID algorithm. Comment briefly on the relative advantages and disadvantages of each.

   (b) [DoAC02, Question 7, April 1998]

   i. There are numerous ways of implementing discretetime PI algorithms. One is the following:

$$\Delta u_t = K\left((e_t - e_{t-1}) + \tfrac{h}{T_i}e_{t-1}\right)$$
$$u_t = u_{t-1} + \Delta u_t$$

   What is this implementation form called?

   ii. Antiwindup for the system described above, simply consists of limiting the control signal, i.e. replacing the second equation by

$$u_t = sat(u_{t-1} + \Delta u_t, u_{\min}, u_{\max})$$

   Explain why this anti-windup scheme works.

   iii. The traditional PI algorithm with antiwindup is given by:

$$v_t = Ke_t + I_t$$
$$u_t = sat(v_t, u_{\min}, u_{\max})$$
$$I_{t+1} = I_t + \tfrac{Kh}{T_i}e_t + \tfrac{h}{T_r}(u_t - v_t)$$

   For which choice of T_r are the two antiwindup schemes identical? Hint: First rewrite the second PI algorithm in the form $v_t = v_{t-1} + \Delta u_t$.

   (c) [DoAC02, Question 2, Dec 1999 Exam] A young engineer who had not taken the Real Time Systems course implemented a PD controller in the following way:

```
y := ADIn(ychan)
e := yref - y;
u := K*(beta*yref - y) + (Td/h)*(e - eold);
u := sat(u,umax,umin);
DAOut(u,uchan);
eold := e;
```

   The PD controller was used in a control system where the setpoint changes were introduced to the controllers as step changes.

i. When the controller was commissioned two types of problems occurred that reduced the control performance. Describe the problems.

ii. Give a better implementation of a PD controller.

(d) [DoAC02, Question 3, Dec 1999 Exam] What is the problem with the following implementation of the integral part of a PID controller? How should it be changed?

```
...
v := K * (P + I + D);
...
I := I + (h / Ti)* error; (* plus tracking *)
...
```

(e) [DoAC02, based on Question 4, Dec 2001 Exam] PID controllers are often implemented based on the following continuous representation

$$u(t) = K\left(\beta y_{sp}(t) - y(t) + \frac{1}{T_i}\int (y_{sp}(\tau) - y(\tau)) + T_d\frac{d\left(\gamma y_{sp}(t) - y(t)\right)}{dt}\right)$$

i. What is the motivation for the introduction of the parameters β and γ.

ii. This can be re-stated as:

$$\begin{aligned} u_{ff}(t) &= K(\beta - 1)y_{sp}(t) + KT_d(\gamma - 1)\tfrac{dy_{sp}}{dt} \\ u_{fb}(t) &= K\left(y_{sp}(t) - y(t) + \tfrac{1}{T_i}\int (y_{sp}(\tau) - y(\tau)) + T_d\tfrac{d(y_{sp}(t)-y(t))}{dt}\right) \\ u(t) &= u_{ff}(t) + u_{fb}(t) \end{aligned}$$

What benefit is obtained by re-stating in this form?

7. From [DoAC02, Question 7, Dec 2001 Exam] Working as a control engineer in a factory you came across a control loop whose step response is shown. y is the process output and u is the control signal to the process measured after the actuator. The controller is given by the following state feedback control law

$$v(k) = -Lx(k) + l_i x_i(k)$$

where $v(k)$ is the control signal sent to the actuator, $x(k)$ contains the measured states of the process and L and l_i are controller parameters (L is a vector). x_i is an extra state that is supposed to provide integral action to the controller. It is given by

$$x_i(k+1) = x_i(k) + h(y_r(k) - y(k))$$

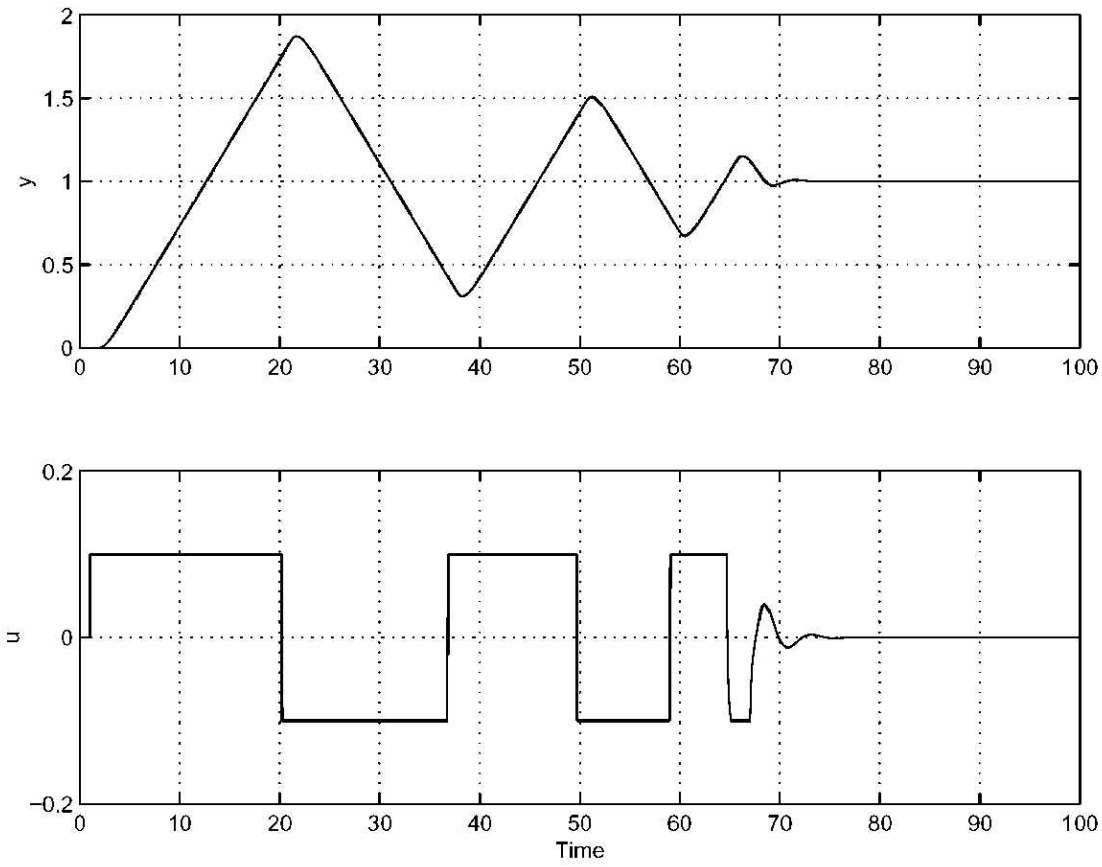
where $y_r(k)$ is the reference value.

(a) What is the likely cause of the behaviour shown in the figure?

(b) Suggest a modification of the control law that should fix the problem.

(c) Given that the process is in steady state, how can you guarantee bumpless transfer when parameter l_i is changed?

8. The department has MATLAB 6 (Release 12), which can be used for modelling Control Systems. If you are unfamiliar with the operation of MATLAB and Simulink for modelling Control Systems, review the help files and tutorials supplied with MATLAB as well as the Control Tutorials for MATLAB[UC97, http://www.engin.umich.edu/group/ctm/], [MT98]. Check your understanding of MATLAB and Simulink by performing the following exercises



(a) [Ben88, based on Questions 4.10,4.11].
The analog system described by the function $\frac{k}{s(s+a)}$ with unity feedback, can be discretized using the z-transform plus zero order hold method. The resulting algorithm is:

$$
\begin{aligned}
e_n &= r - c_n \\
c_n &= \left(\tfrac{k}{a^2}\right)(Ae_{n-1} + Be_{n-2}) + (Cc_{n-1} - Dc_{n-2}) \\
A &= T_s a - 1 + e^{-aT_s} \\
B &= 1 - e^{-aT_s} - aT_s e^{-aT_s} \\
C &= 1 + e^{-aT_s} \\
D &= e^{-aT_s} \\
T_s &= \text{samplinginterval}
\end{aligned}
$$

Write a program which will enable you to calculate the change in output of the system (c_n) with time. It is suggested that 50 values are calculated. The program should enable different values of k, a,T_s, and r to be entered. Using this program, set $k = 2$, $a = 1$,$r = 1$ and investigate the response of the system for different values of T_s. It is suggested that $T_s = 0.02, 0.05, 0.1, 0.2, 0.5$ Compare the results (e.g. in terms of maximum overshoot) with the exact solution for the continuous system (maximum overshoot = 30.5 %).

(b) [Ben88, Question 4.9]The results of an open loop response to a unit step input for a plant are:

Time(seconds)	Output
0.1	0.01
0.2	0.02
0.3	0.06
0.4	0.14
0.5	0.24
0.6	0.34
0.7	0.44
0.8	0.54
0.9	0.64
1.0	0.71
1.1	0.76
1.2	0.79
1.3	0.80

Find

i. the approximate plant model,

ii. a suitable sampling interval for a digital PID controller and,

iii. estimates of the optimum controller settings for PI and PID control.

9. The Real-Time Control Systems Simulator[Cer00a] contains a set of Simulink models and MATLAB files, which will allow you to explore the effects of changing scheduling parameters on the performance of discrete controllers. The simulator supports HPF, RM, DM, and EDF scheduling.

   - In MATLAB, add the path to the real time kernel software (File/Set Path/Add Folder; choose the "kernel" directory on disk/CD; Save)
   - Three examples are included in the simulator archive. The first example concerns PID control of DC servos, the second describes subtask scheduling of control tasks, and the third describes the use of mutexes and events. [Cer00b]

   Carry out the exercises listed in Section 6 of [Cer00b]. Comment on your observations.

10. You have been asked to implement a digital PID controller based on a analogue controller design. The analogue controller was designed using the Ziegler Nichols open loop method where the system step response displays dead time $\mathbf{a} = 9$ seconds, rise time $\mathbf{b} = 20$ seconds. Your task is to implement this using a C program on a microprocessor, which is running other **independent** tasks simultaneously with the motor control task. The system kernel supports time and delay functions with a resolution of 2 seconds.

    (a) Select an appropriate value for the controller sampling period, and justify your choice.

    (b) Derive the constants for the controller equation, and write a C routine to implement the controller.

    (c) Identify and explain ways in which control performance can be affected by the C program implementation of the digital controller.

    (d) Identify and explain the ways in which control performance can been affected by the other tasks running on the microprocessor. Please state any assumptions you make about the task/scheduler/microprocessor.

3 Operating System Support for Concurrency

Learning Objective:

- Given a real time operating system specification/standard, identify the real-time specific features, and contrast the OS with an ideal "open" real-time OS.
  - Characteristics of an "open" real-time OS.
  - Overview of features of current standards(POSIX, EL/IX, μITRON), real time specific (eCos, iRMX, QNX, VxWorks) and general (Linux, Windows) operating systems.
  - Compromises:
    - Mapping design priorities to OS priority levels for tasks/messages.
    - Implementation of periodic tasks using a) sleep b) wait-timer functions
    - Hardware interrupt handling: Split approach
    - Thread scheduling: TCB's, queue structures, pre-emption locks, usage monitor, networking/messaging/signalling, memory protection.
- For control-related scenarios, utilise available techniques for

  2. mapping ideal priority levels onto available OS priorities;
  3. alleviating the constraints placed on the real time systems by hardware and inter-process communications.

Readings

- Slides based on/from:
  - "Operating Systems"[Liu00, Ch. 12]
  - "POSIX thread and Real Time extensions"[Liu00, Appendix]
  - "Real-Time scheduling"[But97, 5.5, pp. 172–188]
- μITRON for Small-Scale Embedded Systems [TS95]

Review Read/summarize at least one of the following papers:

- [JM01, High precision timing within Microsoft Windows: threads, scheduling and system interrupts]. Pay attention to features of Windows which may be used for RT response and the limitations of non-priority message passing.
- [HLSL96, BASEMENT: A Distributed Real-Time Architecture for Vehicle Applications]. Pay attention to how safety-critical and non-safety-critical tasks are handled.
- [LID+03, A Comparison of the RTU Hardware RTOS with a Hardware/Software RTOS]. Pay attention to how hardware can contribute to or replace real-time OS/kernel functions

References

[But97] David R. Butenhof. *Programming with POSIX threads*. Addison-Wesley, 1997.

[BW01] Alan Burns and Wellings. *Real-Time Systems and Programming Languages*. 3rd edition, 2001.

[Gal94] Bill Gallmeister. *POSIX.4: Programming for the Real World*. O'Reilly, 1994.

[HLSL96] Hans Hansson, H. Lawson, M. Strömberg, and Sven Larsson. Basement a distributed real-time architecture for vehicle applications. *Real-Time Systems*, 11(3), 11 1996.

[JM01] A.P. Johnson and M.W.S. Macauley. High precision timing within Microsoft Windows: threads, scheduling and system interrupts. *Microprocessors and Microsystems*, 25:297–307, 2001.

[KS97] C.M. Krishna and Kang G. Shin. *Real-Time Systems*. McGraw-Hill, 1997.

[LID+03] Jaehwan Lee, Karl Ingström, Anders Daleby, Tommy Klevin, Vincent John Mooney III, and Lennart Lindh. A comparison of the rtu hardware rtos with a hardware/software rtos. In *ASP-DAC 2003 (Asia and South Pacific Design Automation Conference 2003)*, 1 2003.

[Liu00] Jane W.S. Liu. *Real-Time Systems*. Prentice-Hall, 2000.

[TS95] Hiroaki Takada and Ken Sakamura. μITRON for small-scale embedded systems. *IEEE Micro*, pages 46–54, December 1995.

Assignment B – 2%

1. [Liu00, Q 12.2] If the *timer_sleep()* function allows only the specification of a time interval for the calling thread to sleep, a periodic task can be implemented as follows:

```
timer_sleep(firstReleaseTime-clock);
nextReleaseTime=firstReleaseTime
do forever
        nextReleaseTime=nextReleaseTime + period;
        statements in the program of the thread
        timer_sleep(nextReleaseTime-clock);
enddo;
```

   where the value of *clock* is equal to the current time and firstReleaseTime is the delay to the first release time of the thread.

   (a) Discuss the factor(s) that may introduce release-time jitters and how large the release time jitter can be. *4 marks*

   (b) If the thread overruns, nextReleaseTime may be earlier than the current time and the argument of the timer_sleep function is negative. What should be the semantic of the function? *4 marks*

2. Given a control system consisting of a set of independent tasks and periods, which have been shown to be RM-schedulable, explain how you could adjust the system if the operating system did not support

   (a) sufficient priority levels *2 marks*

   (b) priority-based pre-emption *2 marks*

3. The specifications for two real-time systems, and three OS's are summarised in Figures 4 and 5. Select an appropriate OS for **each** system; you should justify your choices, making reference to any assumptions you make about hardware/cost. *8 marks*

Ink-Jet Printer Tasks:

1) Dedicated Serial Communication: status and data from PC at 9600 bits/s
2) Print-head control: pulse stepper motor (max. 100 steps across page)
3) Print-head control: pulse stepper motor (1 step/second).
4) Ink control: triggers ink-jet mechanisms before next print-head step.

Factory Tasks:

1) Network Communication: polling independent controllers; send setpoints / receive logs
2) Database: store/retrieve controller logs: guaranteed store; discard queries after 1 minute
3) Monitors: calculate new set-points every 10 seconds

Figure 4: Real-time System Specifications for Q3

	OSA	OSB	OSC
Interrupt Latency (milliseconds)	0.02	0.02	0.002
Context Switching Time (milliseconds)	0.2	0.2	0.002
Pre-emption	None	Priority-based	Round-robin + priority
Priority assignment	None	Priority	EDF
Priority Inheritance	None	Yes	No
# priority levels	None	4	2
max # tasks	2	4	10
timers supported?	Yes	No	Yes
semaphores/mutexes supported?	Yes	Yes	No

Figure 5: OS specifications for Q3

Review Questions

1. (a) What is the thread control block, and how is it used in a real-time operating system?
   (b) Some authors differentiate between the kernel and the operating system. What is the motivation for making this differentiation?
   (c) What advantage is gained by using "split" interrupt handlers in a real-time operating system?

2. [KS97, Q 2.4] There are two ways of implementing a time limit specified for executing a loop. The first is for a compiler to set a maximum number of iterations that may be carried out. The second is to maintain during execution a timer that determines if allowing another iteration would cause the limit to be exceeded. Discuss the advantages and disadvantages of each approach.

3. ([Liu00, Q12.4]) In an operating system, the resolution of the system clock and timers is x. The processor time the scheduler takes to service each timer event is e, and a context switch to the kernel takes no more than CS units of time. Suppose the kernel executes and services timer events once every y units of time.
   (a) In this part, we want to know how the actual timer resolution depends on these system parameters.
      i. Suppose that x and y are large (i.e. order of milliseconds) compared with e and CS (e.g., in order of microseconds). What is the actual timer resolution?
      ii. Suppose that x and y are comparable with e and CS. What is the actual timer resolution?
   (b) We expect that the lengths of intervals returned to a user thread which repeatedly calls *timer_sleep(z)* to deviate from the nominal interval length z. Measurements of many real-time operating systems have found that the maximum deviation can be as large as 20 percent of the nominal value even when there is only one user thread. It was also observed that an abnormally long interval is typically followed by an abnormally short interval, which may in turn be followed by a long interval. Give an explanation of why the lengths of consecutive intervals are correlated.
   (c) Suppose that the number of timer events serviced by the kernel at each clock tick is never greater then l. Derive a formula expressing the error in the time intervals returned by the kernel. You may assume that there are no external interrupts or exceptions of higher priorities while the kernel is servicing the timer calls.

4. ([Liu00, Q12.8]) An application consisting of 11 periodic threads with relative deadlines 1,2,5,6,12,19,21, 27,45,75 and 150 is scheduled on an EDF basis. Suppose the operating system uses the queue structure described in Figure 12–6 (see handout) and supports four distinct relative deadlines, that is, $\Omega' = 4$. Find a mapping from the relative deadlines supported by the system. What is schedulable utilization of the system?

5. [Liu00, Q. 12.9] An operating system provides 256 priorities to threads in the system, but only 32 priorities levels for their messages exchanged through message queues. Suppose that each sending thread chooses the priority of its messages by mapping the 256 thread priority levels to 32 message priority levels. Compare the uniform mapping and constant ratio mapping

schemes. Which one is better and why? Here we measure the performance of a mapping scheme by the average number of messages in each message queue found to have an identical priority; the fewer, the better.

6. [Liu00, Q. 12.7]On a K-bit CPU, the scheduler makes at most $\Omega/K - 1 + log_2 K$ comparisons to find the highest priority nonempty queue among Ω fixed-priority queues. Describe in detail how this operation is done. (Hint: Consider the use of a bit vector containing 1 bit per priority queue to keep the empty/non-empty status of the queues).

7. [Liu00, Q. 12.15]To deliver a per-process signal to a process containing a large number of threads, some of which may have blocked the signal, the operating system must scan the signal masks of all threads to find one that does not block the signal. If it finds none, it queues the signal until some thread in the process unblocks the signal. The overhead of this search is undesirable. How can this overhead be minimized?

8. Based on [BW01, Q 16.4]. A periodic process of period 40ms is controlled by a system with a clock interrupt that has a granularity of 30ms. How can the worst case response time of this process be calculated? Differentiate between clock resolution and clock granularity.

9. (a) Contrast the facilities offered by the POSIX[Gal94] and μITRON[TS95] standards, making reference to their different origins.
   (b) Discuss the pros and cons of implementing real-time functionality at the language level.
   (c) The ideal "open" real-time operating system, would allow the user to submit any job to be scheduled under a known scheduling algorithm, requiring a specific amount of CPU time, and guarantee that the job did not exceed it's specified requirements. Outline, a scheme for implementing such a system.
   (d) Use the Internet to determine how many of the operating systems compared in [Liu00] are still commercially available. Highlight the common features amongst the obsolete and/or remaining systems.
   (e) On the following page you will find descriptions of the eCos and iRMX operating systems. If you had to choose one for a simple embedded control system, which would you choose and why?
   (f) Explore the task creation, communication, scheduling, priority manipulation, blocking handling, and timing facilities available within eCos, Linux, μCOS-II, iRMX, or any popularly used RTOS. Is the chosen OS POSIX/μITRON compliant?

10. For the motor speed controller described in Section 2 Review Question 4b on page 10; you have been asked to select either a dedicated PLC, an embedded system running a real time operating system, or a PC running Windows NT. Presume that cost is not an issue. Identify which system you would select, and give at least **two** supporting arguments.

– from eCos web site

> EL/IX is an application-programming interface (API) based on a subset of the POSIX.1 and ISO C standards plus extensions from Linux/GNU and BSD sockets that are applicable to embedded applications. The result is an API that is more concise than the simple union of those standards because unnecessary or duplicated functionality is eliminated. ... eCos, the embedded Configurable operating system, is an open source real-time operating system for deeply embedded applications. It meets the requirements of the embedded space that Linux cannot yet reach. Linux currently scales from a minimal size of around 500 kilobytes of kernel and 1.5MB of RAM, all before taking into consideration application and service requirements. eCos provides the basic runtime infrastructure necessary to support devices with memory footprints in the 10's to 100's of kilobytes, or with real-time requirements. eCos supports EL/IX Level I, a Linux compatibility interface, for embedded applications in devices that are too small for even stripped down versions of Linux or that require real-time capabilities. eCos provides engineers with maximum control, flexibility and understanding over all aspects of their embedded solution. eCos is highly customizable and adaptable, and can be easily configured using the eCos graphical configuration tool to meet application-specific requirements. eCos also has an ITRON compatibility layer.

– From Introducing the iRMX Operating Systems

> The iRMX(Intel's Real-time Multitasking Executive) OS offers a broad range of real-time, object-based functions and features:
>
> - Real-time processing to monitor and control events
> - Multitasking
> - Preemptive priority-based scheduling
> - Interrupt processing
> - Predictable response time (determinism)
> - Multiprogramming
> - Objects to simplify application design and programming and to control resources
> - Intertask coordination and communication
> - Shared memory and dynamic memory allocation
> - System calls that manipulate objects and control the computer
> - Configurable layers of the OS, each with its own system calls
> - Industry-standard bus support
>
> The C library supports hundreds of C functions and macros for applications that run in the multi-tasking iRMX OS environment. This includes many standard C functions that enable applications to perform common I/O operations without making direct iRMX system calls (OS-independent). There is also support for iRMX OSdependent operations such as multitasking, time-of-day, signal management, and environment management; this enables you to create portable code using standard ANSI and POSIX programming practices. You can mix C library calls with direct iRMX system calls.

– from μITRON specification

> ITRON (Industrial - The Real-time Operating system Nucleus) is a real-time, multitasking OS specification intended for use in industrial embedded systems. Work on the ITRON specification began in 1984 with the start of the TRON Project, and the first specification was released in 1987. That specification is called the ITRON1 specification and was standardized for the major 16-bit microprocessors of that time. After this, work was conducted on the μITRON specification (Ver. 2.0), for use mainly on smaller 8-bit MCUs, and the ITRON2 specification, for use on higher performance 32-bit microprocessors. Both specifications were made public in 1989. Since then the two specifications have been merged. The μITRON4.0 Specification is the latest version of the μITRON real-time kernel specification, a de-facto industry standard in the embedded systems field. The μITRON4.0 specification offers many improvements over previous versions. The most important among them is the definition of the Standard Profile which strictly specifies a standard set of kernel functions for improving application portability.

4 RT Specification, Design, Modelling

- Given a real time system scenario, produce a graphical representation of the system/solution using UML and related techniques.
  - Identify, and correctly interpret the different UML diagrams
  - Given a system description, identify requirements, stakeholders, objects, and messages.
  - Choose/use appropriate UML diagrams to illustrate systems/solutions

Readings

- Slides based on/Extracts from "UML for Systems Engineering" [Hol01]
  - "Modelling" Ch. 2,
  - "The UML diagrams" Ch. 5,
  - "Modelling requirements" Ch. 7
- Slides based on/Extracts from "Doing Hard Time" [Dou99]:
  - "Requirements Analysis of Real Time Systems", Ch 5,
  - "Key Strategies for Object Identification", 6.3,
  - "UML Statecharts", 7.3,
  - "The Role of scenarios in the definition of behaviour", 7.4,
  - "Mechanistic Design", 9
- In class examples: Q.7.11.3 [Dou99]

Assignment C – 2%

1. Construct appropriate sets of UML diagrams for each system specified in Figure 4 on page 18. In each case, you should explain why you have choosen/omitted certain diagrams. *20 marks*

Review Read/summarize at least one of the following papers:

- [GVKT00, "Efficient System Modeling of Complex Real-Time Industrial Networks using the ACCORD/UML methodology"]. Contrast the approach outlined in this paper to the Douglass approach of [Dou99].
- [SR04, "Using UML-Based Rate Monotonic Analysis to Predict Schedulability"]. Pay attention to the description of the UML Profile for Real-Time, then check out the current state of the UML standard (http://www.omg.org/uml/).
- [PEL99, "Experimenting with Real-time Specification Methods: The Model Multiplicity Problem"]. Compare an alternate methodology involving single diagram, to UML with multiple diagrams. Try drawing your own UML diagrams for the case studies.

References

[Ben88] Stuart Bennett. *Real-Time Computer Control: An Introduction*. Prentice-Hall, 1988.

[Dou99] Bruce Powell Douglass. *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns*. Object Technology Series. Addision-Wesley, 1999.

[GVKT00] Sebastien Gerard, Nikos S. Voros, Christios Koulamas, and Fancois Terrier. Efficient system modeling of complex real-time industrial networks using the accord/uml methodology. In *DIPES 2000, International IFIP WG 10.3/WG10.4/WG10.5 Workshop on Distributed and Parallel Embedded Systems*, Paderborn, Germany, October 2000.

[Hol01] Jon Holt. *UML for Systems Engineering: watching the wheels*, volume 2 of *IEE Professional Applications of Computing Series II*. The Institution of Electrical Engineers, 2001.

[Mar] Robert C. Martin. UML tutorials. – http://www.objectmentor.com/resources.

[PEL99] Experimenting with real-time specification methods: The model multiplicity problem. In *Proceedings of the Fourth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in System Analysis and Design (EMMSAD99)*, Heidelberg, Germany, June 1999.

[SR04] H. Saiedian and S. Raghuraman. Using uml-based rate monotonic analysis to predict schedulability. *Computer*, 37(10):56–63, 2004.

Review Questions

1. (a) Describe each of the different types of UML diagram, and illustrate how they may be used.
   (b) [Dou99, Q 5.12.2]What is an actor and what does it contribute to the modeling of a system?
   (c) [Dou99, Q 5.12.4,5,6] What is the relation between:
      i. a use case and a scenario?
      ii. a use case and a statechart?
      iii. a scenario and a statechart?
   (d) Based on [Dou99, Q 7.11.2][With reference to UML diagrams] what is the definition of:
      i. a state?
      ii. a transition?
      iii. an action?
      iv. an activity?
      v. a guard?
   (e) [Dou99, Q 7.11.8] When would you use a timing diagram over a sequence diagram? How about the reverse?
   (f) [Dou99, Q 7.11.10] When would you use an activity diagram over a statechart? How about the reverse?

(g) [Dou99, Q 7.11.9] What is the primary difference between normal states and activity states?

2. (a) Identify the initial state, and final action in the state diagram of Figure 6.

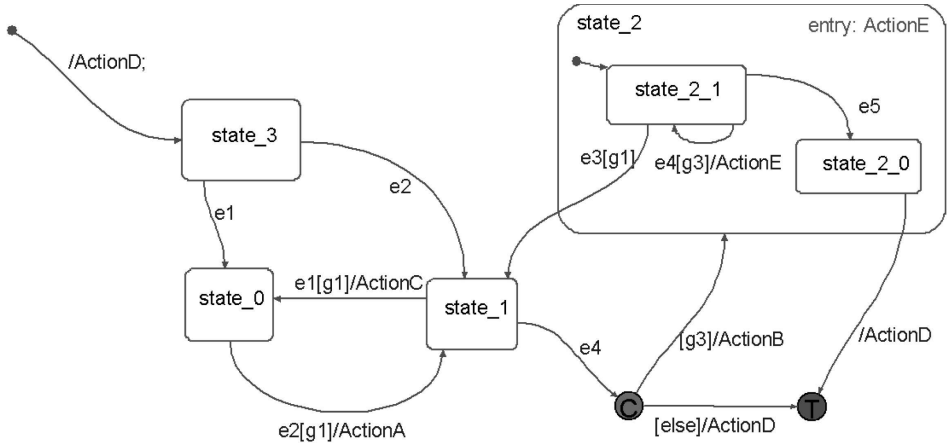


Figure 6: State diagram for Questions 2a, 2b, 2c

(b) A sequence of events occurred in the system described by Figure 6. The sequence is partially shown in Table 1. Fill in the blank spaces of the table. If the space is to be left blank indicate with N/A.

OldState	Event	Guard	Action	NewState
state_1	e1	g1		
	e2		ActionA	
	e4			
			ActionE	
			ActionD	

Table 1: Table for Question 2b

(c) The following timing constraints are associated with the state diagram of Figure 6. Construct an appropriate UML diagram which expresses these timing requirements.

- Actions A and C are mutually exclusive routines. Guard g1 is used to ensure that neither is running. Action A has a duration of 30s, and Action C has a duration of 80s.
- Event e1 is a sporadic event which has a minimum inter-arrival time of 5 ms.
- Event e2 is a periodic event with a period of 120 s.

(d) Construct a statechart for a traffic light system described below:

- If the light is green, it goes red, 2 minutes after going green.
- If the light is red, it goes green, when the car sensor is triggered.
- On initialisation or fatal error the light flashes yellow.

- On error (only detected when the light is red), the controller will attempt to recover, only if the failure is not due to hardware.

(e) A design pattern may be used to convert a statechart into code. Identify at least **one** advantage of using UML design patterns in constructing such a real time system.

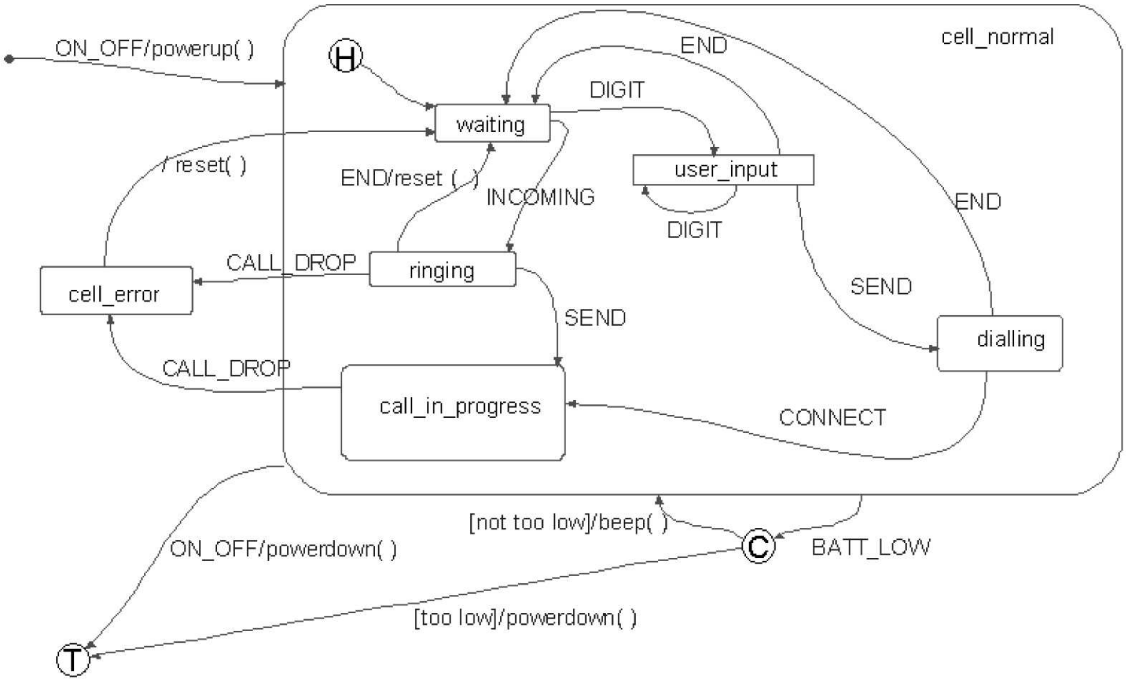


Figure 7: Statechart for Question 3

3. (a) Figure 7 shows a statechart associated with placing a cell-phone call. Identify at least one guard, transition event, transition action, and state.

   (b) Making reference to the figure, identify and explain the features of statecharts which make it possible to model exception handling.

   (c) Describe/illustrate *two* methods by which timing, jitter or other real-time constraints can be expressed using UML notation/diagrams.

   (d) UML may be used for automatic code generation. What implications does this have for system validation/testing?

4. Question 5.12.9 from [Dou99]. “

   Define the use cases for a drive-by-wire automobile control computer, which monitors and controls braking, turning, acceleration control and engine timing. What are the actors?

   ”

5. Check out (http://www.objectsbydesign.com/tools/umltools_byCompany.html) for a list of available UML tools. What are the common features available? [Object Engineering download]

6. A hand-held device for electricity meter readers, consists of a short-range wireless link (for communication with meter(s) and a flash disk for data storage. Figure 8 contains a state-chart for this device.

   In automatic mode the device repeatedly runs three separate tasks:

   - a **whosthere** task (period 30s; timeout 5s) which requests the ID of all meters within range,
   - a **query** task (period 0.5s; timeout 0.1s) which requests a reading from an individual meter whose ID was previously received, and
   - an **interface** task (period 1s; duration 0.01s) which polls the device buttons, and changes mode.

   In manual mode, the **interface** task runs repeatedly, but broadcast and query functions are only executed once on request from the user.

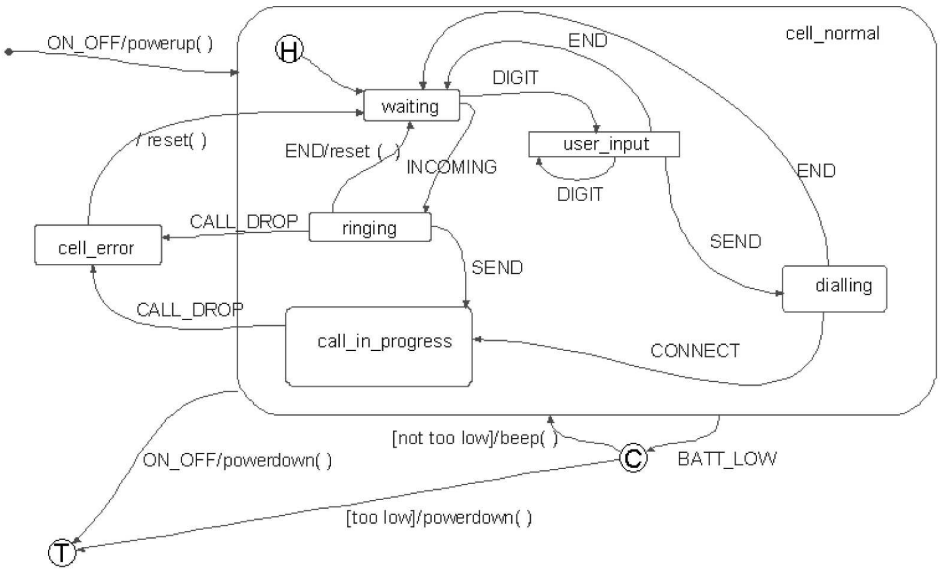


Figure 8: Statechart for Question 6

(a) If possible, identify the initial system state, and a transition action in Figure 8.

(b) Describe "priority inversion". For this application, identify at least one possible cause of priority inversion, and explain one operating system/UML feature which can be used to accommodate/prevent it.

(c) For this application, identify at least one "precedence constraint", and explain one operating system/UML feature which can be used to accommodate it. Your answer should differentiate precedence constraints from priority inversion.

7. From 2002 Past Paper:

   This question refers to the state diagram in Figure 9 which describes a system for the operation of a two joint robotic arm, with a manipulator which can be opened or closed. The control loops for each joint, and the manipulator are run concurrently. The robot must be refitted to meet certain safety regulations:

   (a) Draw a simplified state diagram, showing the two robot states, and the events which transition between states. Identify the initial robot state.

   (b) Identify a guarded event in the diagram.

   (c) Redraw the full state diagram so that:

   - the initial robot state will depend on the initial state of the proximity sensor "p". If the proximity sensor is on, the initial state should be a new state called "safetyStop".
   - the event "evProxClear" will cause the robot to move from the "safetyStop"state to the idle state, only if the proximity sensor is not still on.
   - wherever the proximity sensor trigger event "evProxSensTrig" is received, transition to "safetyStop".

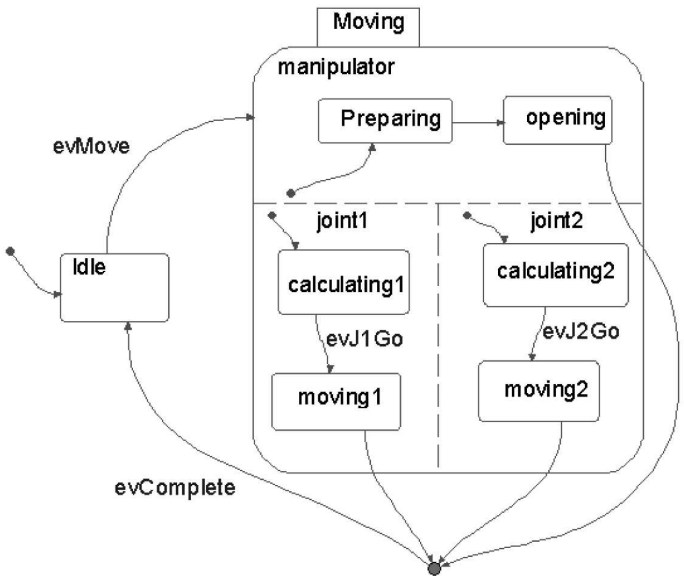


Figure 9: State diagram of a robot arm/manipulator

8. Describe techniques which may be used to identify requirements, stakeholders, objects and messages for a particular application. Apply these techniques to the following passage from [Dou99, "Doing Hard Time"] "

> The Acme Cruise Control user interface consists of four buttons (On/Off, Set Accelerator, Resume, and Coast). When the system is OFF the other buttons have no effect. When the system is ON but a velocity has not been set, the system is *enabled*, but *paused*. When the system is *active* or *paused*, a momentary depression of the Set Accelerator button sets the velocity, making the system *active*; holding the button down more than 1.0 second acts identically to depressing the foot gas pedal, elevating the car's velocity. The Coast button temporarily *pauses* the system as long as it is held down. Once released, the system reverts to *active* velocity control. Any time the difference between the set velocity and the actual velocity exceeds exceeds 10mph, the system *pauses* itself (is *enabled*, but not *active*. The system also *pauses* if the brake pedal is depressed. When the system is *paused*, reactivating the system by pressing the Resume button uses the previous set velocity.

"

9. Flight Controller Based on 2000 EE66C past paper:
We are required to develop a flight controller for flight control in an aircraft. The following are the dominant time constants which were observed experimentally:

   - T_{pitch}=500ms (actuator is the elevators on the tail plane)
   - T_{roll}=100ms (actuators are 2 Ailerons on wings)
   - T_{yaw}= 200ms (actuator is rudder)
   - T_{lift} = 1s (actuator is flaps)
   - $T_{airspeed}$ = 2s (actuator is engine)

   Command Inputs: Speed altitude, direction, bank angle
   Alarms: Stall, Pitch (high), Bank Angle (high), inadequate lift.

   Programs are running on a 2 MIPS processor.

   We are given the control program lengths as:

Algorithm	Length (no. of instructions)
Pitch Control	1000
Roll	1400
Yaw	1000
Lift	2000
Speed	1000

   (a) Choose & Construct appropriate UML diagrams.

   (b) Determine minimum sampling frequency for each signal from the dominant time constants.

   (c) Develop an RM-scheme for, and verify it can be scheduled on the processor.

Plant Interface

Input from plant:
Outlet temperature: analog signal, range 0-10V, corresponding to 20°C to 64°C, linear relationship.

Output from plant:
Heater control: analog signal 0V to -10V corresponding to full heat (0V) to no heat (-10V), linear relationship.

Control

A PID controller with a sampling interval of 40ms is to be used. The sampling interval may be changes, but will not be less than 40 ms. The controllers parameters are to be expressed to the user in standard analog form i.e. proportional band, 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 are to be entered from the keyboard.

Operator communication

Display

The operator display is as shown below:

Set temperature	: $nn.n$°C	Date	: $dd/mm/yyyy$
Actual temperature	: $nn.n$°C	Time	: $hh:mm$
Error	: $nn.n$°C		
Heater ouput	: nn%FS	Sampling interval	: nn ms
Controller settings			
Proportional band	nnn %		
Integral action	$nn.nn$ s		
Derivative action	$nn.nn$ s		

The values on the display will be updated every 5 s.

Operator input

The operator can at any time enter a new set point or new values for the control parameters. This is done by pressing the 'ESC' key. In response to 'ESC' a menu is shown on the bottom of the display screen.

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

Select number of item to change >

In response to the number entered, the present value of the item selected will be deleted from the display and the cursor positioned ready for input of a new value. The process will be repeated until Item 7 – Accept entries is selected at which time the bottom part of the display will be cleared and the new values shown in the top part of the display.

Management Information

On selection of Item 6 of the operator menu a management summary of the performance of the plant over the previous 24 hours will be given. The summary provides the following information:

(a) Average error in °C in 24 hour period.

(b) Average heat demand %FS in 24 hour period.

(c) For each 15 minute period:

  i. average demanded temperature;

  ii. average error; and

  iii. average heat demand.

(d) Date and time of output.

General Information

There will be a requirement for a maximum of 12 control units. A single display and entry keyboard which can be switched between the units is adequate.

Figure 10: Example 5.1 Hot-air blower specification from [Ben88, pp 134–135]

10. Based on [Ben88, Question 5.1] which makes reference to the Hot Air Blower specification shown in Figure 10. Criticise the requirements specification. What information is missing (if any)? Rewrite the specification to include the missing information.

    Illustrate using UML diagrams (all types) as appropriate for a system which meets these revised specifications.

5 Reliability and Fault Tolerance

- Analyse an existing real time system in terms of its reliability and fault tolerance, and recommend methods/changes which would improve performance in either of these areas
  - Quantifying reliability
    - Manipulation of probabilities / error models w.r.t evaluating hardware reliability
    - Appreciation of methods proposed to determine software reliability.
  - Error Masking (detection/correction)
    - Hardware: NMR,
    - Software: N-Version,
    - Data : Checksums/Parity
  - Error Recovery
    - Hardware: Reconfigurable, High Availability, platforms: PC vs PLC
    - Software: Optimal checkpoint placement for recovery blocks
    - Scheduling: Redundant real-time scheduling: FT-RMA
  - Error Reduction/Prevention
    - Hardware/Software/Data: Validation/common standards – UML / IEC61131-3
    - Software/Data/Scheduling: Power awareness/Power management

Readings

Slides based on:

- Fault Tolerance & Reliability Evaluation Techniques[KS97, Ch. 7–8]
- "Integrating UML Real-Time and IEC 61131-3 with Function Block Adaptors" [HT01]
- "Minimum Achievable Utilisation for Fault Tolerant Processing of Periodic Tasks"[PM98].
- "A Survey of Rollback-Recovery Protocols in Message-Passing Systems" [EAWJ02]
- "System-Level Power-Aware Design Techniques in Real-Time Systems" [UK03]
- "SIFT: Design & Analysis of a fault tolerant computer for aircraft control",[WLG+89]
- "Integration Testing of Fixed Priority Scheduled Real-Time Systems",[TPH01]

Assignment D

1. Describe two ways in which each of the systems specified in Figure 4 on page 18, could be made more reliable and/or fault tolerant, while still achieving their real-time requirements. *20 marks*

Review Read/summarize at least one of the following papers:

- [She93, McC92, "A Fault-Tolerant Air Data/Inertial Reference Unit"]. Examine the different ways in which fault tolerance/reliability have been achieved.
- [Sim97, **?**, "Real time recovery of fault tolerant processing elements"] Examine the different ways in which fault tolerance/reliability have been achieved.
- [DDFB02, **?**, "Software-implemented fault-tolerance and separate recovery strategies enhance maintainability (substation automation)"] Examine the different ways in which fault tolerance/reliability have been achieved.
- [SKL85, "A unified method for evaluating real-time computer controllers and its application"] Observe definition of fault-tolerance extended to entire system performance.

References

[Ben88] Stuart Bennett. *Real-Time Computer Control: An Introduction*. Prentice-Hall, 1988.

[BW01] Alan Burns and Wellings. *Real-Time Systems and Programming Languages*. 3rd edition, 2001.

[DDFB02] G. Deconinck, V. De Florio, and O. Botti. Software-implemented fault-tolerance and separate recovery strategies enhance maintainability [substation automation]. *IEEE Transactions on Reliability*, 51(2):158–165, 2002.

[EAWJ02] E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Comput. Surv.*, 34(3):375–408, 2002.

[HT01] Torsten Heverhagen and Rudolph Tracht. Integrating uml-realtime and iec 61131-3 with function block adaptors. In *Proceedings of the IEEE International Symposium on Object Oriented Real Time Distributed Computing (ISORC 2001)*, 2001.

[KS97] C.M. Krishna and Kang G. Shin. *Real-Time Systems*. McGraw-Hill, 1997.

[LaP93] Phillip A. LaPlante. *Real-Time Systems: Design and Analysis, An Engineer's Handbook*. IEEE Press, 1993.

[McC92] C.R. McClary. A fault-tolerant air data/inertial reference system. *IEEE Aerospace and Electronic Systems Magazine*, 7(5):19–23, 1992.

[PM98] M. Pandya and M. Malek. Minimum achievable utilization for fault tolerant processing of periodic tasks. *IEEE Transactions on Computers*, 47(10):1102–1112, 1998.

[She93] M.L. Sheffels. A fault-tolerant air data/inertial reference unit. *IEEE Aerospace and Electronic Systems Magazine*, 8(3):48–52, 1993.

[Sim97] T. Sims. Real time recovery of fault tolerant processing elements. *IEEE Aerospace and Electronic Systems Magazine*, 12(12):13–17, 1997.

[SKL85] Kang Shin, C. Krishna, and Yann-Hang Lee. A unified method for evaluating real-time computer controllers and its application. *IEEE Transactions on Automatic Control*, 30(4):357–366, 1985.

[TPH01] Henrik Thane, Anders Pettersson, and Hans Hansson. Integration testing of fixed priority scheduled real-time systems. In Steve Liu Iain Bate, editor, *IEEE/IEE Real-Time Embedded System Workshop*. Technical Report, Department of Computer Science, University of York, 12 2001.

[UK03] O.S. Unsal and I. Koren. System-level power-aware design techniques in real-time systems. In *Proceedings of the IEEE*, volume 91, pages 1055–1069, 2003.

[WLG+89] J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Milliar-Smith, R. E. Shostak, and C. B. Weinstock. Sift: Design and analysis of a fault-tolerant computer for aircraft control. pages 560–575, 1989. Originally appeared in Proceedings of the IEEE, 66(10):1240–1255, 1978.

Review Questions

1. (a) Differentiate between the terms: dependability, reliability and fault tolerance.
   (b) [BW01, Q.5.1] Is a program reliable if it conforms to an erroneous specification of it's desired behaviour?
   (c) [BW01, Q.5.8] Should the dependability of a system be judged by an independent assessor?

2. Based on [LaP93, Q. 11.4.1-3]
   (a) How is reliability measured?
   (b) Draw the subsystem configuration for a system with four sub-systems and an overall reliability function given by: $r_1(t)r_2(t) + r_3(t)r_4(t) - r_1(t)r_2(t)r_3(t)r_4(t)$
   (c) Calculate the system failure function for a system whose reliability function is $r_1(t)r_3(t) + r_2(t)r_3(t) - r_1(t)r_2(t)r_3(t)$
   (d) Explain how N-modular redundancy improves system reliability. What added overhead does this introduce into a real time system?
   (e) The 'ezplot' function in MATLAB will draw a graph for a symbolic expression. Use this function to explore:
      i. the reliability of clusters as cluster size changes,[KS97, p.335 eqn 8.14]
      ii. the critical ratio of voter reliability when deciding on a cluster size. [KS97, p. 336-7 eqn 8.20]
   (f) Use the Fault Tolerant Computing simulator located at (http://euler.ecs.umass.edu/ece655/simulator/) to explore the reliability of systems which are *not* in series-parallel form.

3. (a) Differentiate between forward and backward error recovery. Highlight their respective effects on task response times and hence overall real time system performance.

   (b) [BW01, Q.6.1] Compare and contrast the exception handling and recovery block approaches to software fault tolerance.

   (c) What is a watchdog timer, and what role can it play in error recovery in a reliable, fault-tolerant real-time control system.?

   (d) [KS97, Q.7.6] What factors govern the optimal placement of checkpoints? Assume that the purpose is to minimize the probability of missing deadlines.

   (e) [Ben88, Question8.1] Examine the list of run-time errors generated by any language system which you use, or for which you have a guide. List the error conditions for which you think the system could either recover or be closed down in a safe manner. Which conditions would require a resort to the run-time support error handling?

4. A solar powered space probe must be designed to run reliably through situations when the supply voltage fluctuates and/or fails. Reliable operation is defined as the **accurate** measurement, conversion, and transmission of sensor readings within a fixed time frame.

   System reliability may be achieved in three main ways: fault tolerance, redundancy and error detection/recovery. These methods may be applied to each of the four aspects of a system: hardware, software, information & task scheduling.

   (a) Identify and explain (at least two) method-aspect combinations which are most appropriate.

   (b) What are the implications, of the chosen combinations, for meeting real-time deadlines?

5. (a) How is reliability typically quantified for hardware, and real-time task schedulers?

   (b) Define the terms "reliability" and "fault tolerance" as they apply to hardware/software systems. How can they be quantified (if at all)?

   (c) [KS97, Q.8.8] A processor has suffered intermittent failure. The characteristics of this failure are that it is in the failed state for a mean time of $1/x$ and in the non-failed state for a mean time of $1/y$. Both times are exponentially distributed. At time 0, the processor is in the failed state. What is the probability that it is in the failed state at time t.

   (d) [KS97, Question 8.11] Suppose processor failure rates are a function $f(u)$ of the processor utilization u. Given seven processors in all, we have the choice of using them either as one 7-MR cluster, or as two 3-MR clusters (with one processor as spare). If two clusters are used, the processor utilization is half that of the single cluster case.

   $$f(u) = (1 - e^{-u}) \times 10^{-5}$$

   For what values of total workload are two clusters better than one given a mission time of t? The total workload is given in terms of the processor utilization.

6. (a) [KS97, Question 7.12] Which of the following codes are separable: parity, checksum, cyclic? Explain your answer.

   (b) Which of the following codes is best suited for use in a real-time system: parity, checksum, cyclic? Explain your answer.

   (c) [KS97, Question 7.5] Suppose you are asked to design a fault tolerant system that uses memory scrubbing to get rid of transient errors. The only fault-tolerance scheme used for the memory is an error-correcting code that can correct up to two bit errors per word. Failure occurs if more than two bit errors occur in a word. Suppose the coded word is 32 bits long, and that memory cells have a probability p of being corrupted in each clock cycle. Assume that these transient cell failures are independent of each other. Calculate the probability of a word suffering failure if the period between consecutive memory scrubs is P clock cycles.

   (d) [KS97, Question 7.13] Design an interlaced parity scheme where the uncoded word has 16 bits and can correct up to two bit errors.

7. (a) For the real-time scheduling algorithms (RM, EDF, VBS) discussed in Section 1, which is the most reliable/fault tolerant when used with a fixed task set. Explain your answer.

   (b) How can the operation of a digital PID controller be made more reliable, either when it is operating solely, or when it is scheduled with other tasks?

   (c) Re-attempt Section 3 Question 10 on page 20. Can you come up with alternate choices and supporting arguments?

   (d) Will using UML for design of a real-time system can improve final system reliability? Explain your answer.

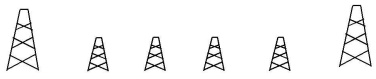
8. Based on Case Study RC5 - Remote Telecommunications System from http://blocksim.reliasoft.com/example

   A telecommunications system is to be constructed in an uninhabited stretch of jungle. The system consists of a transmitter and receiver (100 units apart) with relay stations (which cover up to 30 unit radius) to connect them. The transmitter and receiver are made up of three subassemblies each, while the relay stations have two subassemblies each (all in series). Specifically:

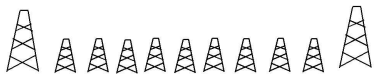
   - Subassembly SPS1 (solar power supply) is common to all.
   - The transmitter has two additional subassemblies, TRC1 and TRC2.
   - The receiver also has two additional subassemblies, RCR1 and RCR2.
   - Relay stations have a subassembly RLYC1 in addition to SPS1.

   The following configurations have been proposed:

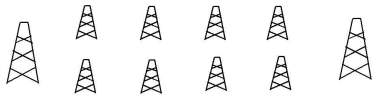
   I use a single line of relay stations at 20 units apart



II use a single line of relay stations 10 units apart



III use two parallel lines (5 units apart) of relay stations at 20 units apart



(a) Presuming that all sub-assemblies have the same failure distributions, and the minimum number of sub-assemblies are used, which configuration offers the best reliability?

(b) For your chosen configuration, identify possible single points of failure, and suggest how these could be corrected.

9. (a) The following statement is taken from http://www.ieeetfcc.org/high-availability.html:

> High Availability (HA for short) refers to the availability of resources in a computer system, in the wake of component failures in the system. This can be achieved in a variety of ways, spanning the entire spectrum ranging at the one end from solutions that utilize custom and redundant hardware to ensure availability, to the other end to solutions that provide software solutions using off-the-shelf hardware components. The former class of solutions provide a higher degree of availability, but are significantly more expensive, than the latter class.

Is this true in the controls industry? Justify your answer(s).

(b) Can a power-aware system be highly available? Can aa power-aware system be dependable? Are these conflicting requirements?

10. (a) Which is more appropriate for use in testing a real-time system: white-box or black-box testing? Justify your answer.

(b) How does the use of UML in the design of a real-time system facilitate system validation?

(c) Investigate the ISO/ITU/IEC standards for fault-tolerant, safety-critical, real-time systems. How is conformity to standards assessed?

6 Real Time Scenarios/Hardware

- Propose a solution for a real-time scenario involving hardware control, and implement the solution utilising a real-time operating system kernel.

Assignment E

[Nis97, "Railway Signalling", Appendix A: Real Time Scenarios, Section A.4 pp 392–397]; Answer the same questions that were used for all scenarios.

Review Questions

In the scenarios given, answer the following questions in each case:

(a) Identify the use cases, actors, and external events, and use the information to prepare a Use Case diagram, and an event table. (4 marks)

(b) Create system models, using UML (and/or any other) diagrams. Identify any additional information which should be specified for this application. (6 marks)

(c) Choose a set of tasks, and make estimates of periods, event arrival rates, deadlines and execution times, from the information provided. Identify any opportunities for shared data/inter-task communication, and thus estimate task blocking time(if any). Justify all choices. (4 marks)

(d) Choose a suitable scheduling algorithm, and verify that your system is schedulable.(2 marks)

(e) (As applicable) recommend strategies for: (2 marks)

- sampling asynchronous signals
- cases where not all tasks are schedulable.
- mapping on to an OS with less priority/messaging than specified.
- fault tolerance/reliability (failure/corruption).
- dealing with interlocking control loops.

(f) Identify any specific real-time features of the Operating System or hardware, which are required to implement the system you have just designed.(2 marks)

(g) Block out pseudo-code for each of your tasks using the real-time OS/hardware features previously identified. Identify any parts of your code which may lead to delay, jitter, or transient errors in the control signals/system response times. (6 marks)

(h) Suggest ways in which the system could be made reliable/fault tolerant. Assess any effects your suggestions may have on the real time performance of the system. (2 marks)

(i) Recommend ways in which the specifications could be enhanced/clarified. (2 marks)

Additional case studies for review/practice can be found throughout the course readings, for example: Motor/Robot Control [Dou99], Temperature Control [SLSS96], Hot Air Blower [Ben88], Radar System [Liu00], Flight Controller

References

[Ben88] Stuart Bennett. *Real-Time Computer Control: An Introduction*. Prentice-Hall, 1988.

[Dou99] Bruce Powell Douglass. *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns*. Object Technology Series. Addision-Wesley, 1999.

[LBC+01] L. J. Lagin, R. C. Bettenhausen, R. A. Carey, C. M. Estes, J. M. Fisher, J. E. Krammen, R. K. Reed, P. J. VanArsdall, and J. P. Woodruff. The overview of the national ignition facility distributed computer control system. In *Proceedings of the 8th International Conference on Accelerator & Large Experimental Physics Control Systems*, 2001. San Jose, California.

[Lei92] J.R. Leigh. *Applied Digital control: Theory, Design & Implementation*. Prentice Hall, second edition, 1992.

[Liu00] Jane W.S. Liu. *Real-Time Systems*. Prentice-Hall, 2000.

[Nis97] Nimal Nissanke. *Realtime Systems*. Prentice Hall, 1997.

[SLSS96] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control systems. In *Proceedings of the IEEE Real Time Systems Symposium*, 1996.

1. **Scenario A – Submarine Sonar system**

   from [Nis97, Appendix A: Real Time Scenarios, Section A.5 pp 398–403];

2. **Scenario B – Intruder Alarm**

from [Nis97]

An intruder alarm receives information about the state of the monitored building from a number of sensors located at every possible entrance and exit. Sensors function basically as switches, indicating whether a given sensor has detected an intruder or not. The alarm is located inside the building. It is set (armed) and reset (disarmed) from inside the building. A digital code of fixed length is required for both setting and resetting the alarm. One of the entrances, which also functions as an exit, is nominated as the entrance and the exit after the alarm has been set.

Timing information is crucial for the proper functioning of an intruder alarm. When the alarm is initially set, a specific time delay is allowed for the user to leave the building through the nominated exit. When the alarm is set, the use of any of the entrances other than the nominated one for re-entry activates the alarm instantly, or at most, within a matter of a few seconds. The sensors monitoring the entrance maintained for re-entry and the route to the alarm control point do not activate the alarm until a set time has elapsed. This set time allows the user to enter the building and disarm the alarm by entering the correct digital code. If this is not done successfully, the alarm is activated at the end of the set time.

The alarm has a siren and a strobe and these are located outside the building. If the intruder is detected or the alarm is not disarmed by the person entering the building through the nominated entrance during the required time, the siren begins to sound and the strobe begins to flash immediately, as mentioned above. In this event, the siren continues to sound for a specified time, usually for a few minutes, and then stops, but the strobe continues to flash. The alarm can be rest by the user only by entering the correct code. If the alarm has already been triggered, this would turn the alarm off. The correct code is the most up to date code entered when arming the system.

When entering the code for disarming the alarm, the user is allowed a maximum period to complete the task. If the user fails to complete this within the given time, the system discards the partial entry and awaits for the next attempt. The user is allowed as many attempts as possible to enter the correct code within the allocated time. If the alarm has already been set off, after this period it cannot be reset except by an appointed independent authority.

Some reasonable limiting values for the timing parameters involved are:

- Time allowed for setting the alarm and leaving the building – 30 seconds
- Time between detecting an intruder and triggering the alarm off – 5 seconds
- Time allowed for re-entry through the nominated entrance and start resetting the alarm – 2 minutes
- Duration for resetting the alarm after re-entry – 1 minute
- Maximum duration for entering the code at each attempt – 20 seconds
- Duration of the siren sound – 5 minutes

3. **Scenario C – Postal Address recognition**

from [Nis97]

Postal address recognition systems are used for automatic sorting of small postal packages such as letters. Sorting is done using a post-code (zip-code in the USA) given as part of the destination address. Sorting involves the recognition of the postcode in the address given on the envelope, establishing the address catchment area by consulting a postal address database according to the recognized postcode and verifying the catchment area using other features such as the street name detected in the address. Packages failing this verification are rejected and are directed for manual sorting

Automatic postcode and address feature recognition use an image of the address produced by a camera. It is a computationally complex task, not only because of the complexity of image processing tasks in general, but also because of the significant variations in handwritten scripts. As a result, the algorithms concerned involve a significant element of heuristics.

Timing of automated sorting is constrained by such requirements as the number of packages processed by each machine per unit time and the capacity of the associated mechanical pipeline. Cuhadar and Downton cite a figure of ten envelopes per second for the processing rate, and 90 envelopes for the maximum capacity of the mechanical pipeline. Downton gives the following statistics for the execution times of the computational tasks and the size of the data.

Processing function	Time sec/image	Approx. execution time ratios	Data packet size (bytes)
Preprocessing (feature extraction)	4.48	3	2114
Classification (postcode identification)	4.45	3	54
Dictionary (data base search)	1.50	1	202
Complete processing	10.43		

4. **Radar Signal Processing[Liu00]**

To search for objects of interest in it's coverage area, the radar scans the area by pointing its antenna in one direction at a time. During the time the antenna dwells in a direction, it first sends a short radio frequency pulse. It then collects and examines the echo signal returning to the antenna.

The echo signal consists solely of background noise if the transmitted pulse does not hit any object. On the other hand, if there is a reflective object (e.g. an airplane or storm cloud) at a distance x meters from the antenna, the echo signal reflected by the object returns to the antenna at approximately $2x/c$ seconds after the transmitted pulse, where $c = 3 \times 10^8$ meters per second is the speed of light.

The echo signal collected at this time should be stronger than when there is no reflected signal. If the object is moving, the frequency of the reflected signal is no longer equal to that of the transmitted pulse. The amount of frequency shift (called Doppler shift) is proportional to the velocity of the object. Therefore by examining the strength and frequency spectrum of the echo signal, the system can determine whether there are objects in the direction pointed at by the antenna and if there are objects, what their positions and velocities are.

Specifically, the system divides the time during which the antenna dwells to collect the echo signal into small disjoint intervals. Each time interval corresponds to a distance range, and the length of the interval is equal to the range resolution divided by c. (For example, if the distance resolution is 300 meters, then the range interval is one microsecond long.) The digital sampled values of the echo signal collected during each range interval are placed in a buffer, called a bin ... The sampled values in each bin are the inputs used ... to produce ... a discrete Fourier Transform of the corresponding segment of the echo signal. Based on the characteristics of the transform, the signal processor decides whether there is an object in that distance range. If there is an object, it generates a *track record* containing the position and velocity of the object and places the record in ... memory.

The time required for signal processing is dominated by the time required to produce the Fourier Transforms, and this time is nearly deterministic ... it takes roughly 10^3 to 10^5 multiplications and additions to generate a Fourier Transform. ... the antenna dwells in each direction for 100 milliseconds and the range of the radar is divided into 1000 range intervals.

extract from: "Real-Time Systems" by Jane W.S. Liu, Prentice-Hall Publishing, 2000. pp 15-16

5. Control of a 1000 tonne press

The control system for a 1000 tonne Fielding & Platt hydraulic forging press, recently installed in the press shop of Thomas Wild Ltd in Sheffield, utilizes a Quarndon electronics VME microcomputer system operating with a 1000 mm analog position transducer, to accurately control the position and motion of the main ram.

The system comprises:

XVME-600/1 68000/68010 processor module with SRAM/EPROM sockets

XVME-201 48 channel I/O module

XVME-500 16SE/8DI channel input module with 10 μs A/D converter

3U component unit – 5 channels of analog signal conditioning and one analog output.

The CVME-201 handles 7 channels of digital input and 13 channels of digital output, all via optical isolators, and includes 11 channels of addressed information. A further 6 channels are used for pump control via the D/A converter on the component board.

The XVME-500 handles five channels of conditioned S/E analog input from the component unit.

AUTO FORGE CYCLE

During the approach sequence the ram speed is monitored, such that when the speed is less than 10mm/s, it is assumed that the work has been contacted. At this point, the ram position is stored and the press forge cycle initiated. The forging speed is controlled by a variable delivery pump. The speeds are:

Fast approach 150mm/s

Slow approach 35mm/s

Forge speed 25mm/s

The system software runs in a multi-tasking PDos operating system loaded into VME Prom 68000 cards.

–from 11.10 Case history C: Control of a 1000 tonne press[Lei92, p.431–2]

6. National Ignition Facility [LBC$^+$01]

The Integrated Computer Control System (ICCS) for the National Ignition Facility (NIF) is a layered architecture of 300 front-end processors (FEP) coordinated by supervisor subsystems including automatic beam alignment and wavefront control, laser and target diagnostics, pulse power, and shot control timed to 30 ps. FEP computers incorporate either VxWorks on PowerPC or Solaris on UltraSPARC processors that interface to over 45,000 control points attached to VME-bus or PCI-bus crates respectively.

Typical devices are stepping motors, transient digitizers, calorimeters, and photodiodes. The front-end layer is divided into another segment comprised of an additional 14,000 control points for industrial controls including vacuum, argon, synthetic air, and safety interlocks implemented with Allen-Bradley programmable logic controllers (PLCs). The computer network is augmented asynchronous transfer mode (ATM) that delivers video streams from 500 sensor cameras monitoring the 192 laser beams to operator workstations. Software is based on an object-oriented framework using CORBA distribution that incorporates services for archiving, machine configuration, graphical user interface, monitoring, event logging, scripting, alert management, and access control.

Software coding using a mixed language environment of Ada95 and Java is one-third complete at over 300 thousand source lines. Control system installation is currently under way for the first 8 beams, with project completion scheduled for 2008.

The NIF contains 192 laser beam lines that are focused on an inertial confinement fusion (ICF) capsule at target chamber center. Each beam requires alignment, diagnostics, and control of power conditioning and electro-optic subsystems. NIF will be capable of firing target shots every 8 hours, allowing time for the components to cool sufficiently to permit precise realignment of the laser beams onto the target.

The NIF requires integration of about 60,000 atypical control points, must be highly automated and robust, and will operate around the clock. Furthermore, facilities such as the NIF represent major capital investments that will be operated, maintained, and upgraded for decades. Therefore, the computers and control subsystems must be relatively easy to extend or replace periodically with newer technology.

The ICCS architecture was devised to address the general problem of providing distributed control for large scientific facilities that do not require real-time capability within the supervisory software. The ICCS architecture uses the clientserver software model with event-driven communications. Some real-time control is also necessary; controls requiring deterministic response are implemented at the edges of the architecture in front-end computer equipment. The software architecture is sufficiently abstract to accommodate diverse hardware, and it allows the construction of all the applications from an object-oriented software framework that will be extensible and maintainable throughout the project life cycle. This framework offers interoperability among different computers and operating systems by leveraging a common object request broker architecture (CORBA). The ICCS software framework is the key to managing system complexity. A brief summary of performance and functional requirements follows

Selected ICCS performance requirements

Computer restart	$<$ 30 minutes
Post-shot data recovery	$<$ 5 minutes
Respond to broad-view status updates	$<$ 10 seconds
Respond to alerts	$<$ 1 second
Perform automatic alignment	$<$ 1 hour
Transfer and display digital motion video	10 frames per second
Human-in-the-loop controls	response within 100 ms

Summary ICCS functional requirements:

- Provide graphical operator controls and equipment status.
- Maintain records of system performance and operational history.
- Automate predetermined control sequences (e.g., alignment).
- Coordinate shot setup, countdown, and shot data archiving.
- Incorporate safety and equipment protection interlocks.

Presentation – 10%

Selection criteria for papers:

- Between 10-20 journal pages in length
- At least two of the following key-words:
  - distributed (embedded) systems,
  - real-time: (real-time) operating systems, (real-time) scheduling, (real-time) control
  - fault tolerance,
  - (minimum) (processor/power) utilization,
  - multi-processor task allocation,
  - error recovery,
  - UML.

For the assigned/selected paper:

- Acquire/Read the paper, and any relevant background/additional material.
- Prepare 1-2 sides of letter-sized paper containing a brief summary of the article content. Highlight the real time systems characteristics found in this application and how they have been addressed. Head your paper with the article title, author and/or source (a sample is shown on page 46)
- Prepare a 20 minute presentation. An OHP projector, black(white) board and copying facilities will be made available.

Marking Scheme:

Timing	0 to 8 minutes; over 20 minutes – 0% 8 to 12 minutes – 1 % 12 to 14 minutes – 1.4 % 14 to 16 minutes; 18 to 20 minutes – 1.6% 16 to 18 minutes – 2%	2%
Presentation Skills	Assessed by each attendee /10	3%
Understanding	Assessed by lecturer .5% deduction for each mis-conception in the Q & A session	2%
Summary	Assessed by lecturer /10	3%

Suggestions for Papers

1. [SSK92]"The Chimera II Real-Time Operating System for Advanced Sensor-Based Control Applications," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 22, No. 6, December, 1992, pp. 1282-1295 (15 pages)
2. [PABD+99]"GUARDS: A generic upgradeable architecture for real-time dependable systems", IEEE Transactions on Parallel and Distributed systems, v. 10 n. 6, June 1999 pp 80–97 (18 pages)
3. [CS00]"A unified wireless LAN architecture for real-time and non-real-time communication services", IEEE/ACM Transactions on Networking, Feb. 2000, Volume: 8 , Issue: 1, pp. 44 - 59 (16 pages)
4. [ZS01]"EMERALDS: A small-memory real-time micro-kernel", IEEE Transactions on Software Engineering, Vol 27, No. 10, October 2001 (18 pages)
5. [CBT05]"Efficient reclaiming in reservation-based real-time systems with variable execution times", IEEE Transactions on Computers, Feb. 2005, Volume: 54, Issue: 2, pp. 198- 213(16 pages)

The IEEE and ACM both publish journals (stocked in our library) which may contain articles on these topics. The following list includes most of these, but is not comprehensive:

- IEEE Transactions on Computers
- IEEE Transactions on Parallel and Distributed systems
- IEEE Transactions on Software Engineering
- IEEE Transactions on Knowledge and Data Engineering
- IEEE Transactions on Automatic Control
- IEEE Transactions on Control System Technology
- ACM transactions on computer systems
- ACM transactions on database systems
- ACM transactions on information systems.
- ACM transactions on programming languages and systems.
- ACM transactions on software engineering and methodology

Apart from the IEEE/ACM journals/transactions, the library also stocks the following journals which may contain relevant articles. Again the list is not comprehensive.

- Microprocessors& Microsystems
- Automatica

Online discussion groups:
To find out what the current "buzz" is about try reading comp.realtime and/or comp.risk.
Online reference pages:

- http://cs-www.bu.edu/pub/ieee-rts/Home.html
- http://dsonline.computer.org/embedded/rts_references.htm

Online databases:
If you want to try searching by keywords, instead of browsing journals, there are several index databases available from the library web site. You may also want to try the more general citeseer (http://citeseer.nj.nec.com/cs), and google (http://www.google.com).

References

[CBT05] M. Caccamo, G.C. Buttazzo, and D.C. Thomas. Efficient reclaiming in reservation-based real-time systems with variable execution times. *IEEE Transactions on Computers*, 54(2):198–213, 2005.

[CS00] Sunghyun Choi and K.G. Shin. A unified wireless lan architecture for real-time and non-real-time communication services. *IEEE/ACM Transactions on Networking*, 8(1):44–59, 2000.

[PABD+99] D. Powell, J. Arlat, L. Beus-Dukic, A. Bondavalli, P. Coppola, A. Fantechi, E. Jenn, C. Rabejac, and A. Wellings. Guards: a generic upgradable architecture for real-time dependable systems. *Parallel and Distributed Systems, IEEE Transactions on*, 10(6):580–599, 1999.

[SE98] J. Hamilton Slye and E.N. Elnozahy. Support for software interrupts in log-based rollback-recovery. 47(10), October 1998.

[SSK92] David B. Stewart, Donald E. Schmitz, and Pradeep Khosla. The chimera ii real-time operating system for advanced sensor-based control applications. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(6):1282–1295, December 1992.

[ZS01] Khawar M. Zuberi and Kang G. Shin. Emeralds: A small-memory real-time microkernel. *IEEE Trans. Softw. Eng.*, 27(10):909–928, 2001.

Presentation – Sample Summary

Support for Software Interrupts in Log-Based Rollback recovery

J. Hamilton Slye and E.N. Elnozahy

IEEE Transactions on Computers, v. 47, no. 10, October 1998

This paper presents an approach to log-based roll-back recovery which allows the replay of asynchronous signals and interrupts in a multi-threaded system, without adding too much overhead (time and code size) to the original code. They focus on "instrument"ing the code to record/maintain information which can be used to generate logs/checkpoints. The authors intentionally omit any discussion of checkpoint and logging protocols, and their respective overheads, as these will be in addition to the instrumentation overhead.

Instrumentation is performed in two steps. Firstly, they alter the compiled code so that it maintains a register-based instruction counter in software. While hardware instruction counters count individual instructions, the software emulation, used in this work, only counts backward branches (within section/page), jumps (out of section/page) and subroutine calls. To alter the code, they search for the relevant instruction (branch, jump, call) and replace it with template instruction code which updates the register. Secondly, they modify the asynchronous signal/interrupt handlers to make log entries using the value of the instruction counter, and/or set/reset/restore the instruction counter. With this information, it is possible to determine precisely when a signal/interrupt was received by a particular thread, and what associated action was taken.

One concern with this work is that the additional code size/run time of code for operation of the instrument will adversely affect system performance. This has been addressed by generating a pair of binaries, one for normal operation and the other for replay. The former is optimized for size and speed. Test results presented on single threaded and multi-threaded tasks, show that the increase in code and run time (during normal operation) on the test applications are less than 30%, on two separate platforms; and in some cases (suggested due to anomalies with cache) the run time even improved for the instrumented code. Replay times and code sizes are slightly larger.

The primary problems with using Backward Error Recovery methods in Real Time Systems are

1. the amount of additional time required for successful checkpointing/logging
2. the non-deterministic amount of time required for recovery and
3. the inability to reproduce asynchronous events.

The work presented here has solved 3, and reduced 1. Bounds for recovery time (i.e. 2) may be addressed through the judicious use of appropriate checkpointing/logging protocols; issues which are not discussed here.

Mini-Project

The purpose of this mini-project is to explore the effects of timing (and other) services (as provided by an RTOS) on control loop performance. It is based on the DC Servo experiments listed in [Cer00], the Control Loop Design Pattern in [Dou99] and the OS performance analysis of [WKSK02].

References

[Cer00] A. Cervin. The real-time control systems simulator reference manual. Technical Report TFRT-7592, Department of Automatic Control, Lund Institute of Technology, April 2000.

[Dou99] Bruce Powell Douglass. *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns*. Object Technology Series. Addision-Wesley, 1999.

[WKSK02] Shige Wang, Sharath Kodase, Kang G. Shin, and Daniel L. Kiskis. Measurement of os services and its application to performance modeling and analysis of integrated embedded software. In *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, pages 113–122. IEEE, September 2002.

Description

A: *Either* *8 marks*

- Write a C program for the IRMX/μCOS-II operating system/kernel which presumes that the following global variables are declared:

  ```
  #define TASKN
  void far (*task[TASKN])(void);
  int far period[TASKN];
  ```

  and performs the following functions:

  - The main program runs each task for a fixed number of iterations and determines the individual response times.
  - The main program creates several instances of a standard task procedure with different priority levels, sleeps, deletes the tasks and then exits
  - The standard task procedure determines what priority level it is running at, in order to know which specific task procedure to invoke.
  - The standard task should repeatedly invoke the specific task procedures provided, and sleep until it is time for the next task invocation.

  OR

- Implement a MATLAB/Simulink model which includes the four task blocks provided and generate/run code for a generic Real-Time target using Real-Time Workshop.

B For the task procedures/blocks provided: *8 marks*

- Using the framework from A, measure computation time for each task when run singly.
- Using the Cheddar simulator, assign periods longer than computation times so that the system has utilization greater than 20% but is schedulable using RMA. What are the predicted response times of each task under both RMA and round-robin scheduling?
- Using the framework from A, measure actual response times on the system when tasks are started with priority ordering according to RMA.
- Using the framework from A, measure actual response times on the system when tasks are started with the same priorities(i.e. round-robin/single-tasking).

C Modify the framework used in A, in order to identify the clock overhead, and observe the behaviour of interval jitter and context switching time as the clock resolution, and task queue length is varied, as per [WKSK02]. Use task4 for this investigation. *6 marks*

D Modify the framework used in A, with an exception handler to trap floating point overflows, and implement forward error recovery. Pass task4 a "bad" argument and measure the error recovery time for your handler. *6 marks*

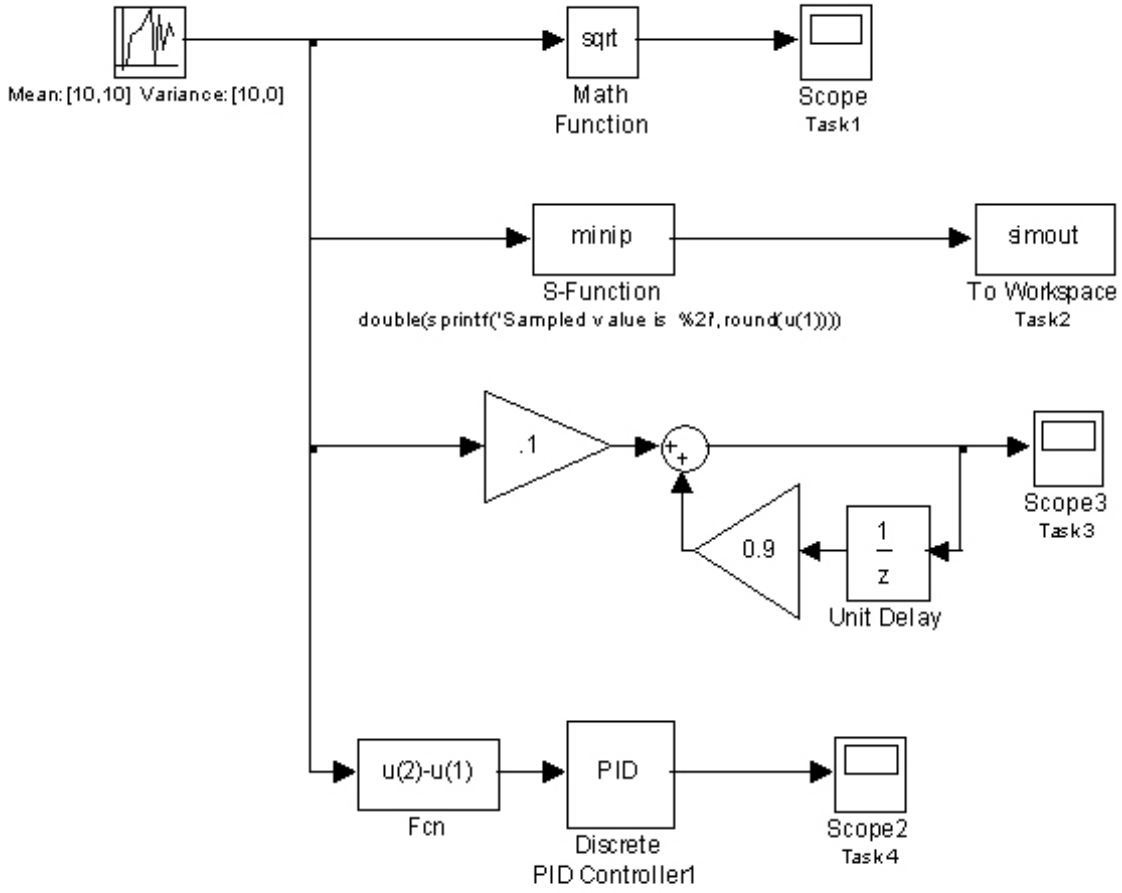
E Modify the MATLAB TrueTime simulation example for control of multiple DC Servo's, in order to accommodate the timing variations discovered above (Suggestion: use randomly generated jitter intervals). Plot the step response of the DC Servo with/without timing variations. *6 marks*

F Write a report which includes, your code for each experiment, the collected data, the clock resolution and speed of the test system/OS and a discussion of/rationale for: *10 marks*

- the differences observed between the measured and predicted response times for RMA and round-robin scheduling.
- the effect of clock overhead and resolution on interval jitter, and context switching
- the effects of error recovery on response times, and system reliability
- the effects of timing on the overall control system

Compare task4 to the standard Control Loop Design Pattern (as described in UML in [Dou99]). Are there any differences which would affect the performance of the control loop?

For testing, use the four tasks specified below (and on the following page).




```
/*Task 1 samples, and performs a calculation on the sampled data */
void task1(void *v) {
    int sample = *(int*)v;
    double d=sqrt((double)*(int *)v);
}

/* Task 2 samples, and converts the sample to a string for logging */
void task2(void *v) {
    int sample = *(int*)v;
    char log[100];
    sprintf(log,"Sampled value is %i", sample);
}
```

```
/* Task 3 accesses the system time to perform low pass filtering on the sample */
void task3(void *v) {
    static double cum=0;
    static time_t last=0;

    int sample = *(int*)v;
    time_t now=time(NULL);
    int frac;

    /* Filter over 1000 time steps */
    frac=now-last;
    frac=(frac<0?0:(frac>1000?1000:frac));
    cum=0.001*(cum*(1000-frac)+sample*(frac));
    last=now;
}

/* Task 4 implements a PID control loop.
The constants and filter parameter are defined. The prior values are stored.
The sampled values are received, and the result returned in an array of
doubles whose address is passed to the procedure. */

void task4(void *v) {
#define K   0.96
#define N   10
#define Td  0.049
#define Ti  0.12

    static double Iold=0;
    static double Dold=0;
    static double yold=0;
    static time_t last=0;
    double sample_r = ((double *)v)[1];
    double sample_y = ((double *)v)[0];
    time_t now=time(NULL);
    frac=now-last;
    frac=(frac<0?0:(frac>1000?1000:frac));

    double P = K*(sample_r-sample_y);
    double I = Iold + K*frac/Ti*(sample_r-sample_y);
    double D = Td/(N*frac + Td)* Dold + N*K*Td/(N*frac+Td)*(yold-sample_y);
    double u = ((double *)v)[2] = P + I + D;

    Iold = I;
    Dold = D;
    yold = y;
    last=now;
#undef K
#undef Ti
#undef Td
#undef N
}

#define TASKN 4
void far (*task[TASKN-1])(void)={task1,task2,task3,task4};
unsigned int far period[TASKN-1]={0,0,0,0};
```

Exam Review

The exam will be 3 hours long. Exam format: 4 structured questions based on the coursework and review questions worth a total of 40 marks; 1 scenario/case study worth 60 marks. There is no choice of questions. You are expected to know the following topics, or be able to perform the following tasks:

- Typical RTOS services and characteristics
  - Define/explain terms used in RT systems
  - Understand operation, properties and analysis of the common scheduling algorithms RM, EDF, LLF, SJF, cyclic executive, round robin, FIFO
  - Utilise techniques for implementing ideal task schedules on "non-ideal" RTOS's
  - Apply rules for deduction of appropriate sampling intervals for (simple control) tasks.
  - Understand techniques for period transformation, and the implications for control loops.
- Understand UML diagram syntax and usage; apply techniques for developing UML diagrams from requirements
- Assessment & Improvement: discuss the relative merits of various hardware/software solutions for a RT system including
  - Techniques for(drawbacks of) software/hardware redundancy/fault tolerance/reliability.
  - Voting schemes used in NMR and in N version software programming,
  - Exception handling mechanisms, and recovery block checkpoint placement issues,
  - Power/Resource management and their effects on RT performance
  - Implications/dangers of single-points of failure
  - Troubleshooting digital control systems based on step response graphs
- Remember Formulae for
  - Utilisation;
  - Utilisation bounds for all scheduling algorithms;
  - Response Time Analysis;
  - Poisson distribution as hardware reliability model;