

CriticalRealTimeSystems- JEvans-p151713

by Jonathan Ieuan Evans

FILE	4766_JONATHAN_IEUAN_EVANS_CRITICALREALTIMESYSTEMS- JEVANS-P151713_178149_590337371.DOCX (332.7K)		
TIME SUBMITTED	08-APR-2019 10:11PM (UTC+0100)	WORD COUNT	1470
SUBMISSION ID	104405640	CHARACTER COUNT	8572

Critical and Real-Time Application Programming

Assignment 2

Jonathan Evans

P151713

8/04/2019

Contents

1. Introduction.....	3
2. Critical Systems	4
3. Washing Machine UML using MARTE.....	5
4. Real-time Operating Systems.....	6
4.1 Scheduler.....	6
4.2 Kernel.....	6
4.3 RTOS Comparison (OS-II vs OS-III)	6
5. Device Drivers	7
6. References.....	8

1. Introduction

This report will examine Critical Systems and Real-Time Operating Systems.

The first half will investigate three critical systems, safety critical, mission critical, and business critical. In this section, an example of a safety critical system will be investigated.

The last half of this report will provide a Real-Time UML diagram to illustrate the functionalities of a washing machine using the MARTE UML Profile. This will be followed up with an explanation of Real-Time Operating Systems and an understanding of device drivers (in particular, USB drivers).

2. Critical Systems

A critical system is a system where the functionality and timing are detrimental to the purpose of the system. There are three kinds of such systems, safety, mission, and business critical.

Safety Critical System

A safety critical system may result in loss of life if deadlines are missed or functionality is not accurate.

Mission Critical System

A mission critical system is a system where failure of operation can have a direct impact on the company that provides the service. An example of such a system could be a banking service that heavily depends on their banks being accessible by the public. If this were to become unavailable due to DDOS (Distributed Denial of Service) attack or other means, the business will not be able to operate due to the downtime incurred.

Business Critical System

Business critical systems account for providing operational services to a business. If this system fails, a business may become un-operational, which may result in loss of money (due to resources becoming unavailable).

Safety Critical System Example

A heart rate monitor is an example of a safety critical system. The primary function of a heart rate monitor is to show a patient (or doctor) an accurate reading of a heart rate at a desirable frequency. Providing that the system can do this successfully, a doctor will be able to examine heart rate data and provide information to the patient on their health. Such a system may even alert the patient when their heart rate has passed a specified threshold – prompting them to calm down or call for help.

However, if this system does not function correctly, it can result in the patient or doctor not being able to view their current heart rate accurately, or even loss of life in the situation where the heart rate provides seemingly reassuring information to the patient, where in fact they are at risk of having a heart attack.

Designing critical systems

As detailed in John C Knight's paper [1], total system modelling is needed to ensure that the design of a system cannot result in having undesirable outcomes.

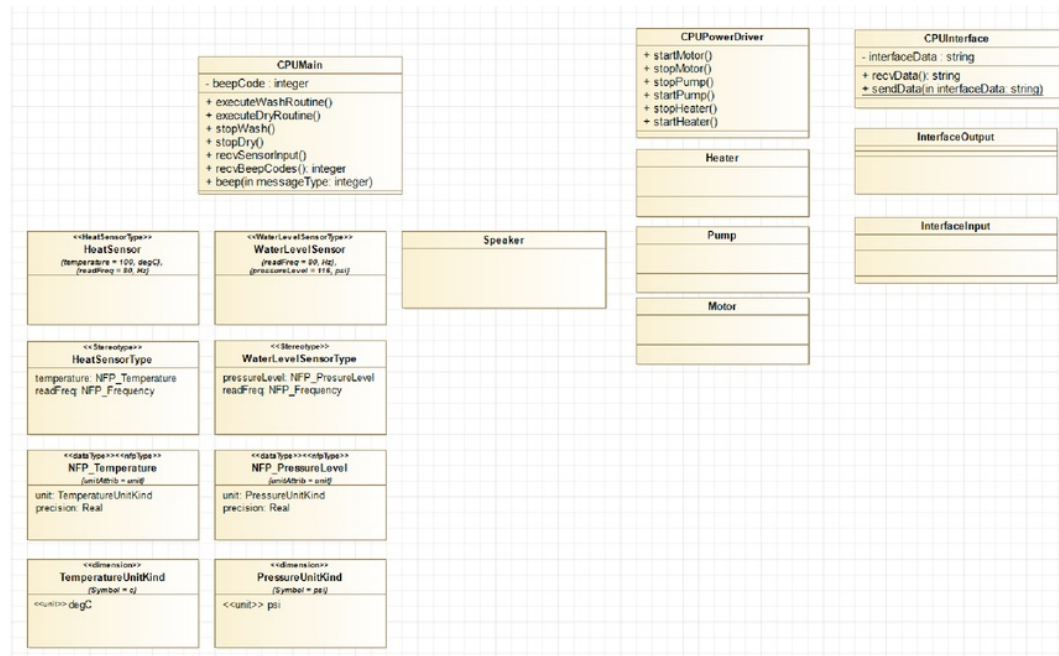
Formal methods could aid development of such systems as methods for mathematically evaluating a system are developed when designing a critical system.



3. Washing Machine UML using MARTE

This section shows the UML diagram for a washing machine. It contains a class diagram, which illustrates the functions of a washing machine. Where necessary, stereotypes have been incorporated to specify units of measurements.

Class diagram



Shown above is the class diagram for a washing machine. This makes use of the MARTE UML Profile. MARTE allows developers to specify properties of a system that are beneficial in the field of embedded system development. It provides a list of types and constraints that allow developers to create more specific diagrams that take into account additional factors such as non-functional properties and time constraints when designing software. [2]

4. Real-time Operating Systems

A Real-Time Operating System (RTOS) is a kind of operating system that must complete tasks by a given deadline. There are two kinds of RTOS systems, hard real-time, and soft-real time. A hard real-time has deadlines in which a task must be completed. If any of these deadlines are missed, consequences could be fatal, however, with a soft real-time system, some deadlines can be missed, however performance may be degraded over time. [3, p. 283]

A washing machine is an example of a system that uses a real-time operating system. There are tasks that must be completed by given deadlines (which are largely mechanical, such as opening/closing valves) – if these are not done in-time, a washing machine may have obtained too much water. There are also tasks that are less important for operation, such as triggering a beep message, releasing the lock – these are not as fatal, thus a deadline can be missed.

When running multiple tasks on RTOS, only one task can be executed at a time (depending on how the system was implemented and how many processing cores are available). However, it is a priority that tasks with strict deadlines must be given CPU time. When multiple tasks are available, a process scheduler is required to determine what tasks should be run.

4.1 Scheduler

A process scheduler is the component of the operating system that schedules when processes can be run. A scheduling algorithm determines how processes are scheduled. With Real-Time Operating Systems, it is common to use a pre-emptive scheduler (as used in popular systems such as OS-II, detailed in *section 4.3*).

A pre-emptive scheduler consists of processes that are capable of yielding (giving) CPU time to another process during execution. This allows for tasks to be interrupted by a higher priority task. As a result, tasks with crucial deadlines should not miss their deadlines, and less important tasks will still be able to execute, although when the preceding process has finished or yielded to the current process.

An alternative method of scheduling is known as Round Robin. This is a time sharing algorithm, that allows processes to run on time slices (usually nanoseconds), and then switches to the next process. [3, p. 272].

In terms of RTOS, it may be more useful to utilise a pre-emptive scheduler as this allows a process to more than likely finish (unless it yields CPU time in favour of a higher priority process).

4.2 Kernel

The kernel is the core component of an Operating System. It includes important components such as the process scheduler, and any device drivers that may be included with the kernel.

4.3 RTOS Comparison (OS-II vs OS-III)

MicroC/OS is an example of a RTOS. It makes use of pre-emptive scheduling. Each task has a corresponding TCB (Task control block) that contains details pertaining to the given process, in addition to the task priority and state, (where each task can have one of the following states: dormant, running, waiting, and terminated). [4, p. 48] These states are added to a FIFO data structure, a queue. The scheduler will refer to these queues when selecting a process to have CPU time.

The main difference between OS-II and OS-III is that the latter is a newer RTOS that provides unlimited tasks, where-as, OS-II only allows a maximum of 255 tasks. As a result, the OS-III can queue more tasks at a time.

5. Device Drivers

A device driver is software that has been written to allow for a computer component to be able to function with a given operating system.



USB Device Drivers

A USB device is an example of a device that will require a device driver to operate with an operating system (this is usually written by the manufacturers, or already included within the kernel of the operating system itself).

USB requires a computer to poll the USB bus for data that has been added to it by a given device. For example, a keyboard will generate an interrupt that will be added to the USB bus when a key is pressed. [5] When the computer is ready, it will read data from the USB bus and display the pressed key on screen (depending on the function of the operating system). With modern computers, this can happen at a frequency of 1000 Hz (or every millisecond).

DMA

Some low-level components (such as a graphics processor) may opt to use DMA. DMA (Direct Memory Access) controllers provide a mechanism that allows the developer to bypass the CPU for accessing memory, thus reducing the operational overhead slightly. In case of a GPU, the developer can write code that allows the GPU to read directly from memory (providing fast access to graphics data).



6. References

- [1] J. C. Knight, 'Safety Critical Systems: Challenges and Directions', p. 4.
- [2] 'A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, Beta 2' . .
- [3] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts*. New York: Wiley, 2012.
- [4] Jean J. Labrosse, 'µC/OS-II TM The Real-Time Kernel User's Manual'. Micrium Press.
- [5] 'USB Made Simple - Part 1'. [Online]. Available: http://www.usbmadesimple.co.uk/ums_1.htm. [Accessed: 08-Apr-2019].

CriticalRealTimeSystems-JEvans-p151713

GRADEMARK REPORT

FINAL GRADE

62/100

GENERAL COMMENTS

Instructor

Critical System (10) 10
Cat heart device (15) 12 Implications on design (15) 6
UML diagrams (20) 8
Research RTOS (20) 16
Device driver Int/DMA (20) 10

Very good description of critical systems. Although the discussion of the design implications was a bit too brief. UML sequence diagram not provided. Instead you have provided class diagrams with NFP values defined. Did you run out of time? Nice general discussion of RTOSs, The comparison of OS-II vs OS-III was a bit brief perhaps because they are so similar. I needed more description of device drivers, and the analysis of the serial driver could have been more targeted at interrupts and DMA as the question asked.

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

Little brief

PAGE 5

PAGE 6

PAGE 7



Comment 2

I need a bit more detail than this.



Comment 3

Why might we use it in a serial driver?