

Realtime

by Ruaidhri David Cook

FILE	4148_RUAIDHRI_DAVID_COOK_REALTIME_168540_1101975015.DOCX (29.38K)		
TIME SUBMITTED	22-MAR-2019 08:01PM (UTC+0000)	WORD COUNT	2233
SUBMISSION ID	103044379	CHARACTER COUNT	12047

Critical and Real-Time Application Programming: Part 1

Student Name: Ruaidhri Cook

Student Number: P148970

Module: Critical and Real-Time Application Programming

Lecturer: Mike Dacey

Contents

Knowledge and Understanding Skills.....	3
Soft Real-Time Systems.....	3
Hard Real-Time Systems	3
Scenario.....	3
Practical Skills.....	5
Cognitive and Intellectual Skills	7
Scenario.....	7
Analysis of Scenario	7
Timing Constraints	Error! Bookmark not defined.
Safety Critical Issues.....	8
References	9

Knowledge and Understanding Skills

Hard & Soft Real-Time Systems

Hard real-time systems require responses in a matter of milliseconds or even less so that catastrophic event cannot happen within the system, whereas, Soft real-time systems have can be significantly slower. This is because the response time for Hard real-time systems is demanded and required by the environment, which dictates its control of pace, unlike Soft real-time applications that allow the computer to dictate its control of pace, as the response time is desired and does not have a hard and fast requirement other than what the computer gives it.  1

Hard real-time systems are more than often used for critical and important operations – those which have an element of safety reliant upon them; this is very different from Soft real-time systems as they are used for operations that are non-critical – operations where catastrophic consequences will not occur in the event of a failure or delay. With Soft real-time systems the critical tasks are given priority over other tasks in the system until said critical tasks are completed, this is in contrast with Hard Real-time systems, which require critical tasks to be completed at the correct time, for this reason the delays in the system must be bounded from the time it takes to retrieve any stored data, to the time that the operating system takes to finish any of the requests that are made.  2

Unlike with Soft real-time systems, the size of data files in Hard real-time systems are much smaller, usually using real-time databases. Moreover, with Hard real-time application systems the performance at peak load must be predictable because of how stringent the predefined deadlines for the system are so that catastrophic problems with the system never occur. Soft real-time applications differ as they have more leeway with their performance at peak loads because their systems are not safety critical.  3  4

 5

Scenario

A car satnav is used to guide a driver, and upon identifying a waypoint updates the screen to remind the driver with a latency of 1.5s.

This scenario is worded in a way that could imply that it is either a Hard-real-time system or Soft real-time system. For this reason, the system could be a hybrid of both of the systems, or instead, a Soft real-time system with some properties that are similar to a Hard-real-time system. This is because the latency time is 1.5s, which is a substantially greater amount of time for an update than what would occur if the System was Hard real-time, which responds in a matter of milliseconds. In addition to this, there are not any safety critical aspects of the system as it is essentially just an advanced electronic map. With this said, the control of pace is still dictated by the environment because the SatNav must respond to the position of the device relative to the destination and the turns in the road. However, it could be argued that the set time of 1.5 seconds implies that the computer is dictating the control of pace and will only be controlled by the environment if the SatNav deviates from the pre-defined route that has been calculated based upon the SatNav's relative position to the waypoint in reference to the map in the database. This also brings upon the size of the data files used for the system, it is most likely that the map used to calculate the route is large in size which would indicate a Soft real-time system, however it is possible that smaller chunks of the map are used when calculating the route because only the relative parts of the map to the SatNav's position are required, which would imply the use of a real-time database – which is used

by a Hard real-time system. Another reason why the system could be Soft real-time is that the critical task, the calculating of a new SatNav route after missing a road, would be prioritised when such an event was to occur, and the recalculation of the new route would continue until the new route was found rather than continuing directions.



Practical Skills

Solving the problem requires it to be split up into different parts, which will become the Assembler functions.

Blending a Channel

The first part of solving the problem would be creating a function for blending a channel, which requires the use of the following formula:

$$result = ((foreAlpha * fore) + (bkg * (255 - foreAlpha))) >> 8$$

The assembler function would have the following in each register:

- R0 – This would contain the foreground alpha channel
- R1 – This would contain the foreground colour channel
- R2 – This would contain the background colour channel

These channels would then be manipulated by the following steps to calculate the final channel value in assembler:

1. The contents of each register would be pushed onto the stack with the link register.
2. The foreground alpha(r0) will be multiplied by the foreground colour channel(r1) (e.g. red) with the result being saved in the r0 register.
3. The r3 register will have the value 255 loaded into it, this is because this is the number range of the alpha value and will be needed for the formula.
4. The foreground alpha(r0) will then be subtracted from the value 255(r3), with the result being stored in r3.
5. The value stored in r3, which is (255 – foreAlpha), will then be multiplied by the background colour channel(r2), with the result being stored in the r2 register.
6. The value stored in r1, which is (foreAlpha * fore), will then be multiplied by the value stored in r2, which is (bkg * (255 – foreAlpha)), with the result being stored in the r0 register.
7. The result in the r0 register, (foreAlpha * fore) + (bkg * (255 – foreAlpha)) will then be shifted right by 8, which will be a divide by 256 (The closest to dividing by 255).
8. The contents of each register will then be popped off the stack, as well as the return address into the program counter

Blending all pixel channels

The second part of solving the problem is creating a function used to blend a whole pixel, this would incorporate the channel blending function.

The assembler function would have the following in each register:

- R0 – This would contain the foreground image array
- R1 – This would contain the background image array
- R2 – This would contain the destination image array
- R3 – This would contain the offset value

In memory the ARGB values are stored in a different order – BGRA – therefore, the operations that will be performed on them must act accordingly.

To blend the pixels the following steps would be taken:

1. The contents of each register (r0-r7) would be pushed onto the stack with the link register.

2. The offset value(r3) will be left shifted by 2 so that it is multiplied by 4 to give the correct position in the array for the pixel being blended, with the result getting stored in the r3 register.
3. The values in the registers will be moved to the r4, r5, r6, and r7 registers. The value in r0, the foreground image array, will then be moved into the r4 register. The value in r0, the foreground image array, will then be moved into the r5 register. The value in r1, the background image array, will then be moved into the r6 register. The value in r2, the destination image array, will then be moved into the r7 register.
4. The value in r3, the offset value, will be added to the immediate value 3, to select the alpha channel. This result will then be stored in r4, the register storing the foreground image array.
5. The same operation will also be performed with r5, r6, and r7, but this time adding the immediate value 2, to select the red colour channel. The result being stored in the respective registers.
6. Then the value in r4, the alpha channel, will be loaded into the r0 register from memory, with the same operation being performed with r5 into r1, and r6 into r2.
7. The blend channel function mentioned previously will be called to perform its steps using the alpha, foreground red and background red channels in the r0, r1 and r2 registers.
8. The value will then in r7 will then be stored in memory using the r0 register.
9. Unlike in the previous steps to select the green channel the immediate value 1 will be subtracted from the registers r5, r6, and r7.
10. Steps 6, 7, and 8 will then be repeated.
11. Following this, the blue channel will be selected subtracting the immediate value 1 from the registers r5, r6, and r7.
12. Steps 6, 7, and 8 will then be repeated, again.
13. The contents of each register (r0-r7) will then be popped off the stack, as well as the return address into the program counter.

Blending full image

The third and final part of solving the problem is creating a function used to blend all the pixels, this would incorporate the pixel blending function.

The assembler function would have the following in each register:

- R0 – This would contain the foreground image array
- R1 – This would contain the background image array
- R2 – This would contain the destination image array

This part of the problem is the simplest, and would be done as follows:

1. The contents of each register (r0-r7) would be pushed onto the stack with the link register.
2. The immediate value 0 would be loaded into register r3, so as to set the offset to 0 – So that the first pixel can be calculated.
3. The function for blending a pixel would be called to perform its steps using the foreground image array, background image array, destination image array and the offset value 0 in the r0, r1, r2 and r3 registers.
4. Steps 2 & 3 will be repeated n number of times where n is the number of pixels in the image, with the only alteration being to the offset so as to blend a different pixel.
5. The contents of each register (r0-r7) will then be popped off the stack, as well as the return address into the program counter.

Cognitive and Intellectual Skills

Scenario

A washing machine controller contains the following:

- Mainboard
 - microcontroller containing a CPU, USB and RS485 (USART) communication
 - EEPROM module
 - Speaker (for buzzer)
- Interface module (different modules can be used, starting from simple on/off and finishing on touch-panels)
- Power driver, which is responsible of control of the drive, and for switching on/off such elements like the heater, the valves, the pump, the door lock electromagnet etc.

Analysis of Scenario

Step by Step of Scenario – including timings

1. Program selected through input interface – Sends information to the Interface module CPU which then sends it to the mainboard CPU using slave and master on the Communication bus RS485. (sends message instantly)
2. CPU selects washing program EEPROM. (selects instantly)
3. Power driver CPU locks the washing machine door electromagnet. (Occur 7 seconds after step 2)
4. Mainboard CPU makes beep sound made through speaker to indicate begin of wash cycle. (Occurs 1 second after step 3)
5. The mainboard CPU communicates with the power driver CPU to opens the water valves so hot and cold water enter the machine and fill up the outer and inner drums. (Takes 15 seconds)
6. The Power driver CPU switches off the water valves. (Happens instantly)  7
7. The thermostat measures the temperature of the incoming water. If it's too cold, the power driver CPU switches on the heating element. (takes 5 seconds)
8. When the mainboard sensors detect that the water is hot enough, the power driver CPU makes the inner drum rotate back and forth. (takes 10 seconds) (rotates for 10 seconds to the left, waits for 5 seconds, rotates for 10 seconds to the right) x 3 times
9. The power driver CPU opens a valve so the water drains from both drums. (7 seconds) Then it switches on the pump to help empty the water away. (10 seconds)
10. The power driver CPU opens the water valves again, so clean water enters the drums. (15 seconds)
11. The power driver CPU makes the inner drum rotate back and forth. It empties both drums and repeats this process several times. (3 minutes rotate to the left, 10 second wait, 4 minutes rotate to the right, 15 second wait, empties drum 15 seconds, refill drum 15 seconds) x 12 times
12. When the clothes are rinsed, the power driver CPU makes the inner drum rotate at high speed. (for 10 minutes)

13. The power driver CPU pumps remaining water from the outer drum and the wash cycle comes to an end. (1 minute)
14. The power driver releases the door lock electromagnet (10 seconds after step 13)

Safety Critical Issues

Not allowing the washer door to open in the middle of a spin cycle – hot water

References

Use the "Insert Citation" button to add citations to this document.

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

Knowledge (20): 14

Cognitive (40): 27

Practical (40): 29

The sat nav scenario is discussed in some detail and the arguments provided are reasonable. However, the initial analysis of hard vs soft RT systems was a unconvincing in parts. Specifically where the discussion talks about the use of databases.

Referencing missing.

The requirements analysis could have provided a little more detail but the main scenario is described in good detail.

C code uses image arrays (4 pixels in size) although the results produced are visually checked and were found to be not correct.

Three assembler functions were called from C and the workflow approach taken in breaking down the problem was very good.

Assembler:

Alpha blending equation provided.

Blend individual pixel provided

Byte ordering was not taken into consideration leading to incorrect results.

A simple test function was provided that only processed four pixels.

PAGE 1

PAGE 2

PAGE 3



Comment 1

I lie this description but where is the reference?



Comment 2

This may be one reason but there are others!!



Comment 3

Is this really the only limitation?

What if the RT system doesn't have a database?



Comment 4

Are all hard RT systems safety critical??



Comment 5

References missing

PAGE 4



Comment 6

These are reasonable arguments and conclusions

PAGE 5

PAGE 6

PAGE 7



Comment 7

Why does it switch of the valves...must be measuring water level?

In reality, nothing happens instantly

PAGE 8

PAGE 9
