ECEN5623, Real-Time Systems:

Exercise #5 – Cyclic Executives in FreeRTOS and Linux

DUE: As Indicated on Canvas

Please thoroughly read Chapters 9, 10 & 11 in the text.

Please see example code provided on Canvas – seqgen.c and seqgen2x.c. Note that this code has be written with the Jetson and Linux in mind, but with slight modification can be used on a virtual machine, Raspberry Pi or on the DE1-SoC.

This lab emulates a cyclic executive real-time system with multiple services. Each service runs at a separate frequency, so this is a multi-frequency executive.

Exercise #5 Requirements:

1) [20 points] Download seqgen.c and seqgen2x.c and build them in Linux on the Altera DE1-SOC, Raspberry Pi or Jetson board and execute the code. Describe what it is doing and make sure you understand how to use it to structure an embedded system. Determine the worst case execution time (WCET) for each service by printing or logging timestamps between two points in your code or by use of a profiling tool. Determine D, T, and C for each service and create an RM schedule in Cheddar using your WCET estimates. Calculate the % CPU utilization for this system.

2) [40 points] Revise seqgen.c and seqgen2x.c to run under FreeRTOS on the DE1-SoC or TIVA board, by making each service a FreeRTOS task. Use the associated startup file in place of the existing startup file in FreeRTOS. Use an ISR driven by the PIT hardware timer to release each task at the given rate (you could even put the sequencer in the ISR). Build and execute the new code. Determine the worst case execution time (WCET) for each service by printing or logging timestamps between two points in your code or by use of a profiling tool. Determine D, T, and C for each service and create an RM schedule in Cheddar using your WCET estimates. Calculate the % CPU utilization for this system. Compare this with the results you achieved under Linux in (1).

3) [40 points] Revise seqgen.c from both previous systems to increase the sequencer frequency and all service frequencies by a factor of 100 (3000 Hz). Build and execute the code under Linux and FreeRTOS on your target boards as before. For both operating systems determine the worst case execution time (WCET) for each service by printing or logging timestamps between two points in your code or by use of a profiling tool. Determine D, T, and C for each service and create an RM schedule in Cheddar using your WCET estimates. Calculate

the % CPU utilization for these systems. Compare results between Linux and FreeRTOS in this higher-speed case.

Overall, provide a well-documented professional report of your findings, output, and tests so that it is easy for a colleague (or instructor) to understand what you've done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code. I will look at your report first, so it must be well written and clearly address each problem providing clear and concise responses to receive credit.

Grading Rubric

[20 points] Linux Cyclic Executive:

- [10 points] Explanation and Execution of code_____
- [2 points] Timestamps and WCET estimates_____
- [3 points] D, C, and T Table for all services_____
- [5 points] Cheddar schedule and % CPU utilization _____

[40 points] FreeRTOS Cyclic Executive:

- [25 points] Explanation and execution of code on target _____
- [2 points] Timestamps and WCET estimates_____
- [3 points] D, C, and T Table for all services_____
- [5 points] Cheddar schedule and % CPU utilization _____
- [5 points] Comparison between Linux and FreeRTOS_____

[40 points] High-Rate Cyclic Executive:

- [25 points] Explanation and execution of code on target _____
- [2 points] Timestamps and WCET estimates_____
- [3 points] D, C, and T Table for all services_____

[5 points] Cheddar schedule and % CPU utilization _____

[5 points] Comparison between Linux and FreeRTOS_____