UPPSALA
UNIVERSITET

Assignment for Real Time System Course,
code:1DT004, Report code:11202

Lab 1: ADA Programming

Group Number: 2

Sagar Shubham Kedar Joshi Akash Goel
Arijeet Laga

September 23, 2018

Contents

List of Figures

1 Part 1: Cyclic Scheduler

1.1 Part 1: Question 1

Explain the difference between relative delay and absolute delay using your own understanding.

Relative Delay:

- It is used delaying the task for certain amount of time.
- It can be accessed by providing command "Delay".
- Expiration time for relative delay is rounded up to the nearest clock tick.
- Form of the relative delay:- Delay Duration expression.
- This Duration expression should be provided by the user in Minutes.

Absolute Delay:

- It is used for delaying the task for a certain of amount of time by giving a reference time.
- It can be accessed by providing command "Delay until".
- Form of the relative delay:- Delay Time expression.
- This Time expression should be provided by the user in by accessing internal clock.

1.2 Part 1: Question 2

Suppose we know the exact execution times of F1, F2 and F3. Is it possible to use relative delay for avoiding drift? Why?

With Exact execution times known, we can provide relative delay to match the periods of the respective procedures. However, relative delays suffer from local drift (Granularity difference between clock and delay, etc.); and hence it is not possible to remove drift entirely with relative delays even with exact execution times known. Therefore it is better to use absolute delays, which cause a delay until a particular time, and are implemented via interrupts, and thus less susceptible to local drifts.

2 Part 2: Cyclic Scheduler with Watch-Dogs

2.1 Part 2: Question 1

How did you synchronize the watchdog task with the start and the end of a F3 execution?

We Provide entry points to the watchdog task for both start time, and end time of procedure F3. We can note the time of these events in the watchdog for further usage.

2.2 Part 2: Question 2

Does it makes sense to use absolute delay in watchdog timeouts? Why/why not?

Using Absolute delays as against relative delay in a timeout operation, will prove more useful here as the former do not suffer from local drifts. Hence it is useful to use absolute delay in watchdog timer of this task.

2.3 Part 2: Question 3

Explain the way you re-synchronize the cyclic executive when F3 misses its deadline.

When the Watchdog Task timeouts due to F3 missing its deadline, we note this fact by setting `F1RescheduleNeeded` as `True` and let F3 run its course. When F3 actually finishes it present run, we now make a note of the Time when it ends as `F3EndTime` . We now have the problem of re-scheduling F1 (which will in turn re-synchronize other two procedures from next cycle) by finding the whole number Time just bigger than `F3EndTime`. This can be found be repetitively adding F1's period (of 1.0 seconds) to the last F1's Scheduled time (after which F3 missed its deadline for the first time), and continuing to do so till we have met the condition of F1's Schedule Time is greater than `F3EndTime`. We re-synchronize the procedures to start their run from this time, bringing F1's schedule time back to whole number value.

3 Part 3: Process Communication

3.1 Part 3: Question 1

Show by a concrete scenario that the producer-consumer-buffer program using blocking rendezvous communication can have deadlocks and explain the mistake that can cause it

ANSWER:

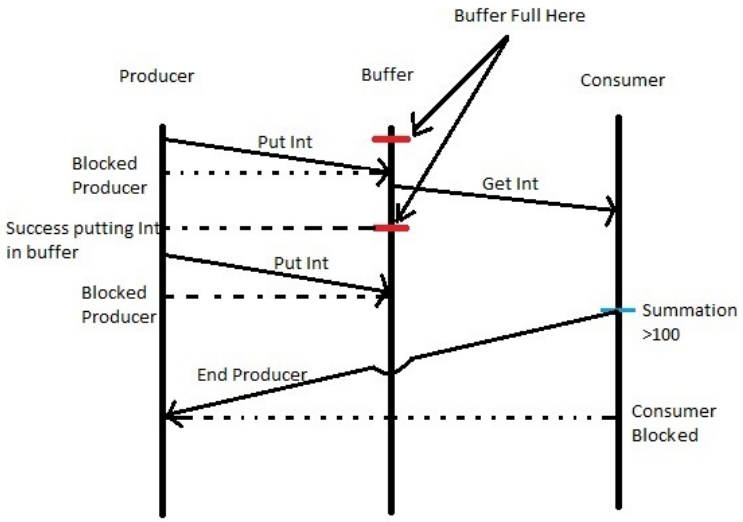

Figure 1: Deadlock Situation for Part 3. Question 1

We assume the situation that through normal course of action , the Buffer is already full(The first red Point in Buffer Task Timeline in Figure 1). When producer tries to put Integer. As buffer is already full, it cannot be succeeded and Producer gets block. Consumer gets Integer from buffer. Now, buffer is vacant for one integer. As we know producer is blocked (ready with next integer) takes the vacant place in buffer. This process continues till the summation of consumer end is 100. When consumer is done with the summation(¿100), Consumer End the produce and consumer itself get blocked. At this point producer is already blocked with one integer. As the consumer is already blocked, consumer cannot get data from buffer, so buffer automatically get blocked. This situation is known as deadlock.

4 Part 4: Data Driven Sychronization

4.1 Part 4: Question 1

Does the producer-consumer-buffer program using protected object suffer from any potential deadlock? If not, please explain the main reason behind it.

No, with a protected buffer object, the producer-buffer-consumer program does not suffer from a potential deadlocks.

This is because, for a protected object, only one entry point can be active at one time, which is to say, only either the producer or the consumer can write or read the buffer object at one go. This avoids the issue of producer being blocked by the buffer and consumber being blocked by producer, resulting in no deadlock situation.