Slide 1

Read the C file “Example2.c” carefully. In this example, a queue that can hold up to 5 integers is created, with a handle qh.

Slide 2

A sender task runs the shown task_tx function. Each second, the task will increment a local integer i and send its value to the queue. If queue is full, it will wait for half a second before displaying an error message. Otherwise, it will display the value correctly sent to the queue.

Slide 3

A receiver task will run the function task_rx, continuously trying to read an integer from the queue. If queue is empty for one second, it times out and displays an error message, then tries again. If an integer is correctly received, it is saved in a local variable j, and its value is displayed.

Slide 4

Click on the video to see the output of the tasks when both have the same priority level of 1.

Slide 5

Here, we modified the receiver task to receive every 2 seconds, while the sender still sends every second. As expected, after a while the queue becomes full and the send operation fails after the sender times out.

Slide 6

Here, the receiver is trying to receive continuously, but its time out period is zero. The screen will be full with error messages as a result of several failing attempts to receive.

Slide 7

However, whenever data is sent, it is received correctly as seen here if we disable displaying the error message. In the next slide, we repeat this last version of the receiver task, giving it a higher priority level of 2. Try to expect what will happen before going to the next slide.

Slide 8

Nothing is sent or received since the high priority, continuously running receiver task does not allow the sender to operate. You can experiment more with the effects of receiver delay and timeout period on this last program.