**Chapter 3-3 RTOS Mutex**

Welcome back to WICED WiFi 101. In this video I am going to talk about the mutal exclusion which is most often shorted to just mutex. A mutex is an RTOS primitive that allows a thread to lock a resource until it is done with it. If another thread trys to lock the mutex, it will suspend until the original thread unlocks it. The thread that locks a mutex is the only thread that is allowed to unlock it.

Now, for example this shield has two slave devices on the I2C bus, the display, and the PSoC4 analog co-processor. If you have one thread that wants to talk to the display … and another that wants to talk to the PSoC4 then you have the problem of 1 resource, the I2C Master, which wants to be used by two different threads. If both threads try to use the I2C at the same time, you will find disaster… or at least a locked up system.

To demonstrate this, I will build a simpler test case where you can see this problem…. And then use the mutex to fix it. What I will do is build two threads that will both try to blink an LED when a button is pressed. If you make both threads blink the same LED things will work just fine… at least until you press both buttons at the same time. At which point the LED blinking will go haywire.

Lets start by building this application without a mutex. Start by either creating a new project called 03\_mutex, or by copying one of your other projects. To make things clear I will create two functions, one called ledThread0 and one called ledThread1. The ledthread 0 will blink the LED1 with a delay of 100ms if the button 0 is pressed. And the ledThread 1 will blink the led with a delay of 150ms if the button 1 is pressed.

When I program the project you can see that things work just fine when I press button 0….. then button 1….. but when I press both at the same time I get all kinds of craziness because the shared resource, meaning the LED is getting changed by two different threads.

To solve this problem in application\_start I will initialize a mutex. Then in the two threads I will LOCK the Mutex before I start blinking. If you try to get a mutex that is already gotten, your thread will suspend until the mutex can be gotten.

Now that I have the mutex enabled, you can see that things are good when I press both buttons at the same time because which ever button is pressed first, locks the mutex and the other thread suspends.

As always, you can post your comments and questions in our Wifi developer community or you are welcome to email me at alan\_hawse@cypress.com or tweet me at @askioexpert with your comments, suggestions, criticisms and questions.

In the next video I will talk about Queue