1. **What is the purpose of the timerCallback() function?**

The timerCallback() function is intended to be called every time the timer expires. In many applications, timers are used to trigger actions at regular intervals or after a specific delay. In the code, the timer is initialized in the initTimer() function, and it's set to call timerCallback every 1,000,000 microseconds (or 1 second). However, the current implementation of the timerCallback() function is empty. This means that, in its current form, when the timer expires and the timerCallback function is called, nothing will actually happen. The body of the function is empty, so there's no observable effect. In the future it is where the state machine will go.

1. **What does period mean in this context?**

In the context of timers, the term "period" refers to the duration of time between successive timer expirations. In other words, it's the interval at which the timer "ticks" or "fires".In the code, the following line sets the period of the timer, params.period = 1000000;. Given that the unit for this period is microseconds (as indicated by the line params.periodUnits = Timer\_PERIOD\_US;), this means that the timer is set to expire every 1,000,000 microseconds, or every 1 second.

Once the timer is started with this period setting, it will trigger the associated callback function (timerCallback in this case) every 1 second. If there was code inside the timerCallback function, that code would be executed every second.

In summary, the "period" in this context is the time interval at which the timer expires and, consequently, its associated callback function is called.

1. **How does the Timer\_CONTINUOUS\_CALLBACK parameter impact the driver?**

The Timer\_CONTINUOUS\_CALLBACK mode in the timer driver indicates that the timer will operate in a continuous, periodic manner. Once started, the timer will expire at regular intervals, as defined by its set period. Each time the timer expires, the associated callback function will be invoked. This behavior will continue indefinitely until the timer is explicitly stopped. This contrasts with a potential one-shot mode where the timer would expire just once after its designated duration and would require manual restarting to run again. In the provided code, the timer is configured to use the continuous callback mode, ensuring that the specified timerCallback function is called every second, given the period is set to 1,000,000 microseconds, without any manual intervention to keep it running. This mode is particularly useful for tasks that require periodic attention or action at consistent time intervals.

1. **What is gpioButtonFxn0() used for?**

The gpioButtonFxn0() function serves as a callback tied to a specific GPIO event, primarily triggered by the pressing of a button. Within the code, this function is tasked with toggling the state of an LED connected to the CONFIG\_GPIO\_LED\_0 pin. When the button associated with the CONFIG\_GPIO\_BUTTON\_0 pin is pressed, a falling edge interrupt is detected, leading to the invocation of the gpioButtonFxn0() function. As a result, each button press results in the LED's state being changed: if it's currently on, it'll turn off, and vice versa. The binding of this function to the GPIO pin is established in the main setup, ensuring that the specific action of toggling the LED is consistently linked with the event of the button press.

1. **What is the purpose of GPIO\_CFG\_IN\_INT\_FALLING?**

In the code, GPIO\_CFG\_IN\_INT\_FALLING is employed to configure a GPIO pin, specifically for detecting a falling edge signal, which typically corresponds to the pressing of a button. When the code line GPIO\_setConfig(CONFIG\_GPIO\_BUTTON\_0, GPIO\_CFG\_IN\_PU | GPIO\_CFG\_IN\_INT\_FALLING); is executed, it does two things: Firstly, it configures the GPIO pin represented by CONFIG\_GPIO\_BUTTON\_0 as an input with a pull-up resistor (GPIO\_CFG\_IN\_PU). This means the default state of the pin is high (logic '1') when the button isn't pressed. Secondly, with GPIO\_CFG\_IN\_INT\_FALLING, the GPIO pin is set to generate an interrupt when a falling edge, a transition from high to low, is detected. This is often a result of pressing a button that connects the GPIO pin to ground, producing a falling edge. Once this falling edge interrupt is generated, the system invokes the associated callback function (in this case, gpioButtonFxn0()) to handle the event, such as toggling an LED. This configuration allows the system to efficiently respond to external stimuli, like a button press, in real-time.