CONFIGURAR INTERRUPCIONES DEL SCUTIMER

```
/*
* Connect the device driver handler that will be called when an
* interrupt for the device occurs, the handler defined above performs
* the specific interrupt proccesing for the device.
*/
Status = XScuGic_Connect(IntcInstancePtr, TimerIntrId,
        (Xil_ExceptionHandler)TimerIntrHandler,
        (void *)TimerInstancePtr);
if (Status != XST_SUCCESS) {
        return Status;
}

/*
* Enable the interrupt for the device.
*/
XScugic_Enable(IntcInstancePtr, TimerIntrId);

/*
* Enable the timer interrupts for timer mode.
*/
XScuTimer_EnableInterrupt(TimerInstancePtr);
```

```
/*
* Disconnect and disable the interrupt for the Timer.
*/
XScuGic_Disconnect(IntcInstancePtr, TimerIntrId);
```

```
static void TimerIntrHandler(void *CallBackRef)
{
        XScuTimer *TimerInstancePtr = (XScuTimer *) CallBackRef;

        /*
        * Check if the timer counter has expired, checking is not necessary
        * since that's the reason this function is executed, this just shows
        * how the callback reference can be used as a pointer to the instance
        * of the timer counter that expired, increment a shared variable so
        * the main thread of execution can see the timer expired.
        */
        if (XScutimer_IsExpired(TimerInstancePtr)){
                Scutimer_ClearInterruptStatus(TimerInstancePtr);
                TimerExpired++;
                if (TimerExpired == 3){
                        XScuTimer_DisableAutoReload(TimerInstancePtr);
                }
        }
}
```