```c

#include <stdio.h>
#include "simonDisplay.h"
#include "supportFiles/display.h"
#include "buttonHandler.h"
#include "flashSequence.h"
#include "verifySequence_runTest.h"
#include "simonControl.h"
#include "supportFiles/utils.h"

#include "xparameters.h"
#include "supportFiles/leds.h"
#include "supportFiles/globalTimer.h"
#include "supportFiles/interrupts.h"
#include <stdbool.h>
#include <stdint.h>

#define TOTAL_SECONDS 60
// The formula for computing the load value is based upon the formula from 4.1.1 (calculating timer intervals)
// in the Cortex-A9 MPCore Technical Reference Manual 4-2.
// Assuming that the prescaler = 0, the formula for computing the load value based upon the desired period is:
// load-value = (period * timer-clock) - 1
#define TIMER_PERIOD 50.0E-3
#define TIMER_CLOCK_FREQUENCY (XPAR_CPU_CORTEXA9_0_CPU_CLK_FREQ_HZ / 2)
#define TIMER_LOAD_VALUE ((TIMER_PERIOD * TIMER_CLOCK_FREQUENCY) - 1.0)

static uint32_t isr_functionCallCount = 0;

int main()
{
    display_init();
    display_fillScreen(DISPLAY_BLACK);

    interrupts_initAll(true);
    interrupts_setPrivateTimerLoadValue(TIMER_LOAD_VALUE);
    printf("timer load value:%ld\n\r", (int32_t) TIMER_LOAD_VALUE);
    u32 privateTimerTicksPerSecond = interrupts_getPrivateTimerTicksPerSecond();
    printf("private timer ticks per second: %ld\n\r", privateTimerTicksPerSecond);
    interrupts_enableTimerGlobalInts();
    // Initialization of the clock display is not time-dependent, do it outside of the state machine.
    // clockDisplay_init();
    // Start the private ARM timer running.
    interrupts_startArmPrivateTimer();
    // Enable interrupts at the ARM.
    interrupts_enableArmInts();
    // The while-loop just waits until the total number of timer ticks have occurred before proceeding.
    while (interrupts_isrInvocationCount() < (TOTAL_SECONDS * privateTimerTicksPerSecond));
    // All done, now disable interrupts and print out the interrupt counts.
    interrupts_disableArmInts();
    printf("isr invocation count: %ld\n\r", interrupts_isrInvocationCount());
    printf("internal interrupt count: %ld\n\r", isr_functionCallCount);
    return 0;
}
```

```
54 void isr_function() {
55     simonControl_tick();
56     flashSequence_tick();
57     verifySequence_tick();
58     buttonHandler_tick();
59     isr_functionCallCount++;
60 }
61 
62 
```