```c
/**
* Clock.c
* This file contains the service to tick the DDM 60-s clock.
*/

#include "ES_Configure.h"
#include "ES_Framework.h"
#include "PWMTiva.h"
#include "Clock.h"
#include "Mummy.h"

#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_sysctl.h"

#define CLOCK_RATE 42 // reduce servo pulse width by 42 after every tick

static uint8_t tick = 0; // number of clock ticks
static uint8_t MyPriority; // module-level variable for the VibrateMotor module's priority

/* Prototypes */
bool InitializeClock(uint8_t Priority);
ES_Event RunClock(ES_Event CurrentEvent);
bool PostClock(ES_Event ThisEvent);
static void initClock(void);
static void tickClock(void);

/**
* Initializes the Clock service.
* @param Priority: priority of the Clock service
*/
bool InitializeClock(uint8_t Priority) {
    ES_Event ThisEvent;
    MyPriority = Priority; // set the priority
    initClock();

    ThisEvent.EventType = ES_INIT; // post the initial transition event
    if (ES_PostToService(MyPriority, ThisEvent) == true) {
        return true;
    } else {
        return false;
    }
}

/**
* Runs the Clock service.
* @param CurrentEvent: the current event that has occurred
*/
ES_Event RunClock(ES_Event CurrentEvent) {
    ES_Event ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no problems will occur

    if ((CurrentEvent.EventType == ES_TIMEOUT) && (CurrentEvent.EventParam == CLOCK_TIMER)) {
        if (QueryMummySM() != disARMed) {
            tickClock();
        }   else {
            tick = 0;
            initClock();
        }
    }
```

```
    return ReturnEvent;
}

/**
* Initializes the clock servo.
*/
static void initClock(void) {
    PWM_TIVA_SetPulseWidth(LIMIT_1, CLOCK_PIN); // make the clock servo move to one end
}

/**
* Ticks the clock servo.
*/
static void tickClock(void) {
    tick++; // increment tick count
    uint16_t nextPulse = LIMIT_1 - CLOCK_RATE*tick;
    if (nextPulse < LIMIT_2) {
        tick = 0; // reset tick
        initClock();
        return; // after 60 s, stop ticking the clock
    }
    PWM_TIVA_SetPulseWidth(nextPulse, CLOCK_PIN);
    ES_Timer_InitTimer(CLOCK_TIMER, TICK); // reset the timer
}

/**
* Posts events to the Clock service.
* @param ThisEvent: event to post to Clock servce
* @return true if successfully posted; false otherwise
*/
bool PostClock(ES_Event ThisEvent) {
  return ES_PostToService(MyPriority, ThisEvent);
}
```