Module Functions
// static void UpdatePotato (void);
// static void ReturnPotato (void);

Module Variables
// static uint8_t MyPriority;
///static uint8_t good_potato = 0;
// static uint8_t bad_potato = 1;
// static uint16_t PulseWidthLo_Good = 2050;
// static uint16_t PulseWidthHi_Good = 2500;
// static uint16_t PulseWidthLo_Bad = 2150;
// static uint16_t PulseWidthHi_Bad = 2600;
// static uint8_t group = 0;
// static uint16_t reqPeriod = 50000;
// static CelebServiceState_t CurrentState = CS_Inactivity;
// static uint16_t CelebTime = 20000; //ms
// static uint16_t BobTime = 500; //ms

Function
InitCelebService
Parameters
uint8_t : the priorty of this service
Returns
bool, false if error in initialization, true otherwise

// Initialize InitCelebServiceand pass into Priority variable
	// Initialize the MyPriority variable with the passed in parameter
	/ /Set PWM channels and periods
// end function

Function
PostCelebService
Parameters
EF_Event ThisEvent ,the event to post to the queue
Returns
bool false if the Enqueue operation failed, true otherwise

// Initialize PostCelebServiceand pass into ThisEvent
	// Post to the service this event
// end function

Function
RunCelebService
Parameters
ES_Event : CELEB_START, BOB_TIMEOUT, CELEB_TIMEOUT
Returns

```
ES_Event, ES_NO_EVENT

// Initialize RunNeedleService and pass into ThisEvent
// assume no errors
// Set Current state to next state
// switch current state
        // If CurrentState is CS_Inactivity
                // start celeb time
                // Start bob timer
                // Turn on all LEDs
                // Post service to LEDWriteService
                // initialize count
                // next state is Bobbing
        // end if
        // end case

        // if Current State is Bobbing
                // if EventType is ES_TIMEOUT and EventParameter is BOB_TIMER
                        // if count is even
                                // turn on all LEDS
                                / /Post service to LEDWriteService
                        // else count is odd
                                // turn off all LEDS
                                // Post service to LEDWriteService
                        // append count
                        // initialize BOB_TIMER
                        // check if good potato
                        // set channel to good potato and UpdatePotato
                        // else if bad potato
                        // set channel to bad potato and UpdatePotato
                        // set nextstate to bobbing
                //end if

                // else if EventType is ES_TIMEOUT and EventParameter is CELEB_TIMER
                        //return potatoes to ground
                        // check if good potato
                        // set channel to good potato and return potato
                        //post event to GameService
                        // set nextstate to CS_Inactiivty
                // end if
        // end if
        // Set current state to next state
// end function

Function
UpdatePotato
```

Parameters
Channel – to determine which motor to actuate (good or bad potato motor)
Returns
None

```
// Initialize UpdatePotato
//initialize counter
        //if channel is good potato
                // if counter is odd
                        // set pulsewidth low
                // end if
                // if counter is even
                        // set pulsewidth high
                // end if
        // end if
        // else if channel is bad potato
                // if counter is odd
                        // set pulsewidth low
                // end if
                // if counter is even
                        // set pulsewidth high
                // end if
        // end if
// Increment counter
// end function
```

Function
ReturnPotato
Parameters
Channel – to determine which motor to return to ground (good or bad potato motor)
Returns
None
Description
Actuates motor to the most left position to return potato to the ground

```
// Initialize ReturnPotato
        // if channel is good potato
                // set return pulse to the most left positions
                // post pulsewidth and channel to PWM tiva
        // end if
        // if channel is bad potato
                // set return pulse to the most left position
                // post pulsewidth and channel to PWM tiva
        // end if
// end function
```