```
// Module WaterService.c
// this file is used for the implementation of our “whack-a-mole” styled water service game
// Module Variables
// Begin Module Code

//*****************************************************************************************************

// bool InitWaterService takes in a priority variable and sets the priority of the service
// initialize the CurrentState to W_Inactivity

//*****************************************************************************************************

// bool PostWaterService takes in ThisEvent and posts it to the service along with the
corresponding MyPriority
// end

//*****************************************************************************************************
// ES_Event_t RunWaterService takes in an event
// assuming there are no errors, ReturnEvent will be of type ES_NO_EVENT
// enter state machine
// first case is W_Inactivity
// if the EventType is WATER_START
        // initialize the WATER_TIMER with its WATER_DURATION
        // call random function to generate first random LED light to turn on
        // set CurrentState to Wait4IR
// endif
// end case

// case Wait4IR
// if ThisEvent is ES_TIMEOUT and the timeout is from the WATER_TIMER event paramter
        // post a command to the LED write service to turn all of the water module LEDs off
        // post to game service ThisEvent
        // call TurnOffWaterLEDS() function
        // set CurrentState to W_Inactivity
// endif

// else if the event type is a rising edge
        // if the event parameter is equal to the SetValue (the user is correct)
                // generate a WATER_CMD that turns the LEDs off
                // call the AddHealth function
                // init the buzz timer
                // call the vibration turn on and off function with the parameter that turns it on
                // set CurrentState equal to Buzzing
```

```
        // else (the user has incorrectly selected a water LED)
                // SubtractHealth
                // turn off the water LEDs
                // init the water flash timer to give feedback that they were wrong
                // set CurrentState to FlashWait
        // end else
// end else if
// end case

// case FlashWait
// if there's a timeout of the WATER_FLASH_TIMER
        // if FlashCount equals 0
                // turn on all water LEDs to show that the user has made a mistake
                // post the WATER_CMD to the LEDWriteService
                // increment FlashCount
                // Start the flash timer again
        // end if

        // else if FlashCount is equal to 1
                // turn off all the LEDs and post the event to the LEDWriteService
                // ++FlashCount
                // init the WATER_FLASH_TIMER
        // end else if

        // else
                // set FlashCount equal to 0
                // call SetRandom function to pick out a new random light
                // set CurrentState to Wait4IR
        // end else
// end if

// if the event timeout is the WATER_TIMER
        // call function TurnOffWaterLEDs
        // post this event to the game service
        // set the CurrentState equal to W_Inactivity
// end if
// end case

// case Buzzing
// if there is a timeout of the WATER_TIMER
        // TurnOffWaterLEDs
        // call the ON_OFF_VIBRATION function with the parameter to turn off the buzzing
```

```
        // post this event to the game service
        // set the CurrentState equal to W_Inactivity
// end if

// else if this event is a timeout of the BUZZ_TIMER
        // call the ON_OFF_VIBRATION function with the parameter to turn off the buzzing
        // randomize the next water LED that turns on
        // set the CurrentState equal to Wait4IR
// end else if
// end case
// end switch case
// return the ReturnEvent
// end run function

//*********************************************************************************************************

// bool Check4IRRise
// declare ThisEvent
// call ADC_MultiRead function with array triadeStatus as a parameter
// initialize IRCurrentState to 0
// call ADC_Convert function

// start for loop iterating through the array triadeStatus to update the current state of the IRs
// end for loop


// if there has been a change in the status of the IRs
        // if the first water LED (0th) is high, then the event is an IR_RISE with parameter BIT0HI
        // else if the second water LED (1st) is high, then the event is an IR_RISE with
parameter BIT1HI
        // else if the third water LED (2nd) is high, then the event is an IR_RISE with parameter
BIT2HI
        // set the IRLastState equal to the IRCurrentState
        // post the event to PostWaterService
        // return true
// end if

// set the IRLastState equal to the IRCurrentState
// return false
// end Check4IRRise

//*********************************************************************************************************
```

```
// void ADC_Convert
// initialize TempSetValue to BIT0HI
// start for loop that goes through each element of triadeStatus
// if triadeStatus[i] is greater than the IR's high threshold
        // update triadeStatus[i], so that the appropriate bit is HI
// end if

// else (the threshold was not met)
        // triadeStatus at that index is set to 0
// end else
// end for loop
// return
// end ADC_Convert

//*********************************************************************************************************

// void ON_OFF_VIBRATION takes in an 8-bit integer called Value
// if Value is equal to BIT3HI (the bit associated with the vibration motor)
        // set the port HI
// end if

// else
        // set the port LO
// end else

//return
// end ON_OFF_VIBRATION

//*********************************************************************************************************

// void SetRandom
// declare ThisEvent
// use srand to seed the random number generator using ES_Timer_GetTime()
// declare a local variable x that is a random number between 0 and 2 inclusive (number of
water LEDs)
// set a random bit in SetValue high according to the random number x
// send the SetValue as a WATER_CMD to the PostLEDWriteService
// end SetRandom

// void TurnOffWaterLEDs
// declare Event2Post
// make the event of event type WATER_CMD and set the parameter equal to 0
// post the event to PostLEDWriteService
```

```
// end TurnOffWaterLEDs
```