GamePlay.c
Pseudocode for the GamePlay Module (a service that implements a state machine)
Data private to the module: CurrentState, HighScore, MyPriority

InitGamePlay
The parameter for this function is the priority, an unsigned integer

Initialize tot sense pin to digital input
Initialize solenoid pin to output, low
Set MyPriority to Priority passed in
Set Current State to InitPState
Call ResetGame
Call ActuateSolenoid
Set HighSchore to 0
Post ES_INIT event to self

PostGamePlay
The parameter for this function is of type ES_EVENT

Call ES_PostToService with MyPriority and the event passed in

RunGamePlay
The parameter for this function is of type ES_EVENT

Run one of the following blocks of code based on the value of CurrentState:

```
If CurrentState is InitPState:
    If the passed in event is WATERDB_ButtonDown:
        Set CurrentState to ReadyForTot
        Start the WELCOME_TIMER with the WELCOME_TIME

If CurrentState is Ready4Tot:
    If the passed in event is TOTIn:
        Set CurrentState to Playing
        Post GAME_START event to all services
        Start GAME_TIMER with LENGTH_OF_GAME time
        Start USER_ACTIVE_TIMER with USER_ACTIVE_TIME
        Turn off all LED strips
    If the passed in event is ES_TIMEOUT for SOLENOID_TIMER:
        Call ActuateSolenoid
    If the passed in event is ES_TIMEOUT for WELCOME_TIMER:
        On the first call, initialize CurrentWelcome to 0
```

```
            Increment CurrentWelcome by 1
            Set CurrentWelcome to mod3 of CurrentWelcome
            Turn the CurrentWelcome number of LED strip on
            On the first call, set CornPulse to 600
            Increment CornPulse by 200
            Set CornPulse to mod2400 of CornPulse
            Set the PWM of each Corn Channel to CornPulse
            Set the WELCOME_TIMER to WELCOME_TIME
      If the passed in event is HIGH_SCORE:
            If the event param is greater than saved HighScore:
                  Set HighScore to the event param
            Create a new ES_EVENT of type HIGH_SCORE
            Set the new event param to be the saved HighScore
            Post the HighScore to the SevSeg service

If CurrentState is Playing:
      If the event is ES_TIMEOUT for the GAME_TIMER:
            Create a GAME_END event
            Post the GAME_END event to GameClock and AudioService
            Start the CELEBRATION_TIMER with CELEBRATION_TIME
            Set CurrentState to Celebrating
            Start the WELCOME_TIMER with WELCOME_TIME
      Else if the event is ES_TIMEOUT for the USER_ACTIVE_TIMER:
            Call ResetGame
            Set CurrentState to Ready4Tot
            Start the WELCOME_TIMER with WELCOME_TIME
      Else if the Event is WATERDB_ButtonDown:
            Restart the USER_ACTIVE_TIMER with USER_ACTIVE_TIME
            Create a WATER_CORN event
            Set the event parameter to QueryLaneSelectionFSM
            Post the event to the CornPop service
      Else if the Event is FIREDB_ButtonDown:
            Restart the USER_ACTIVE_TIMER with USER_ACTIVE_TIME
            Create an ALIEN_BLAST event
            Set the event parameter to QueryLaneSelectionFSM
            Post the event to the AlienFSM and Audio services
      Else if the Event is LEVERDB_ButtonDown:
            Restart the USER_ACTIVE_TIMER with USER_ACTIVE_TIME

If CurrentState is Celebrating:
      If the event is ES_TIMEOUT of the CELEBRATION_TIMER:
            Call ResetGame
            Set CurrentState to Ready4Tot
```

```
            Start WELCOME_TIMER with WELCOME_TIME
        Else if the event is ES_TIMEOUT of the WELCOME_TIMER:
            On the first call, initialize CurrentWelcome to 0
            Increment CurrentWelcome by 1
            Set CurrentWelcome to mod3 of CurrentWelcome
            Turn the CurrentWelcome number of LED strip on
            Create a BLINK event and post it to SevSeg service
```

QueryGamePlay

```
Return CurrentState
```

ResetGame

```
Call ActuateSolenoid
Start the SOLENOID_TIMER with SOLENOID_TIME
Create a GAME_RESET event and post it to all services
```

ActuateSolenoid

```
On the first call, initialize SolenoidState to 0
Set SolenoidState to the logical Not of its previous value
If Solenoid state is 0:
     Turn the solenoid off by setting the pin low
Else:
     Turn the solenoid on by setting the pin high
```

AlienFSM.c
Pseudocode for the state machine controlling all three aliens

Data private to the module: Alien1State, Alien2State, Alien3State, MyPriority

InitAlienFSM
Takes as parameter an integer representing the service priority.

Set MyPriority to the priority parameter.
Set Alien1State, Alien2State, and Alien3State to Off
Call AlienClearAll
Post ES_INIT event to self

PostAlienFSM
Takes an event as a parameter.

Post input event to ES service with priority of MyPriority.

RunAlienFSM
Takes an Event as a Parameter.

```
If the input event is an ALIEN_BLAST:
     If the event parameter is LaneOne:
          Call AlienWrite on Alien1 with parameter 0
          Stop ALIEN_1_TIMER
          Set Alien1State to FullUp
     Else if the event parameter is LaneTwo:
          Call AlienWrite on Alien2 with parameter 0
          Stop ALIEN_2_TIMER
          Set Alien2State to FullUp
     Else if the event parameter is LaneThree:
          Call AlienWrite on Alien3 with parameter 0
          Stop ALIEN_3_TIMER
          Set Alien3State to FullUp
Else if the input event is GAME_START:
     Set Alien1State, Alien2State, and Alien3State to FullUp
     Call AlienWrite for all three aliens, with paramter 0

Else if the input event is GAME_RESET:
     Call AlienClearAll

Else if the input event is GAME_END:
```

```
      Set Alien1State, Alien2State, and Alien3State to Off

Else if the input event is ES_TIMEOUT from EAT_CORN_TIMER:
      Create a SCORE_DECR event and ALIEN_CONTACT event
      For each AlienState in FullDown:
            Post the SCORE_DECR event to SevSeg service
            Post the ALIEN_CONTACT event to CornPop service
      Reinitialize the EAT_CORN_TIMER with EATING_RATE time

Else:
      Execute one of the following blocks of code based on
Alien1State:
            If Alien1State is FullUp:
                  If the event is ALIEN_ATTACK with param LaneOne:
                        Call AlienWrite for Alien1 with param 1
                        Init ALIEN_1_TIMER with ALIEN_DROP_TIME
                        Set Alien1State to Down1
            If Alien1State is Down1:
                  If the event is ES_TIMEOUT for ALIEN_1_TIMER:
                        Call AlienWrite for Alien1 with param 2
                        Init ALIEN_1_TIMER with ALIEN_DROP_TIME
                        Set Alien1State to Down2
            If Alien1State is Down2:
                  If the event is ES_TIMEOUT for ALIEN_1_TIMER:
                        Call AlienWrite for Alien1 with param 3
                        Set Alien1State to FullDown
                        Post SCORE_DECR event to SevSeg service
                        Post ALIEN_CONTACT event to CornPop service
                        Init EAT_CORN_TIMER with EATING_RATE
      Execute one of the following blocks of code based on
Alien2State:
            If Alien2State is FullUp:
                  If the event is ALIEN_ATTACK with param LaneTwo:
                        Call AlienWrite for Alien2 with param 1
                        Init ALIEN_2_TIMER with ALIEN_DROP_TIME
                        Set Alien2State to Down1
            If Alien2State is Down1:
                  If the event is ES_TIMEOUT for ALIEN_2_TIMER:
                        Call AlienWrite for Alien2 with param 2
                        Init ALIEN_2_TIMER with ALIEN_DROP_TIME
                        Set Alien2State to Down2
            If Alien2State is Down2:
                  If the event is ES_TIMEOUT for ALIEN_2_TIMER:
```

```
                    Call AlienWrite for Alien2 with param 3
                    Set Alien2State to FullDown
                    Post SCORE_DECR event to SevSeg service
                    Post ALIEN_CONTACT event to CornPop service
                    Init EAT_CORN_TIMER with EATING_RATE
     Execute one of the following blocks of code based on
Alien3State:
          If Alien3State is FullUp:
               If the event is ALIEN_ATTACK with param LaneThree:
                    Call AlienWrite for Alien3 with param 1
                    Init ALIEN_3_TIMER with ALIEN_DROP_TIME
                    Set Alien3State to Down1
          If Alien3State is Down1:
               If the event is ES_TIMEOUT for ALIEN_3_TIMER:
                    Call AlienWrite for Alien3 with param 2
                    Init ALIEN_1_TIMER with ALIEN_DROP_TIME
                    Set Alien3State to Down2
          If Alien3State is Down2:
               If the event is ES_TIMEOUT for ALIEN_1_TIMER:
                    Call AlienWrite for Alien3 with param 3
                    Set Alien3State to FullDown
                    Post SCORE_DECR event to SevSeg service
                    Post ALIEN_CONTACT event to CornPop service
                    Init EAT_CORN_TIMER with EATING_RATE
```

AlienWrite

First parameter is the Alien number, second parameter is the desired position

```
Execute one of the following blocks of code based on Alien
number:

Alien1:
     Pull all Alien1 pins low
     Set the input position of Alien1 pin high
Alien2:
     Pull all Alien2 pins low
     Set the input position of Alien2 pin high
Alien3:
     Pull all Alien3 pins low
     Set the input position of Alien3 pin high
```

AlienClearAll

Set all pins controlling aliens to low

SevSeg.c
Pseudocode for the service controlling the two seven segment displays via encoders.

Data private to this module: MyPriority, CurrentScore, DisplayOn

InitSevSeg
Takes as parameter an integer representing the service priority.

```
Set MyPriority to input priority
Set CurrentScore to 0
Set DisplayOn to 0
Call WriteDisplay
Call WriteScore
Post ES_INIT event to self
```

PostSevSeg
Takes an event as a parameter.

Post input event to ES service with priority of MyPriority.

RunSevSeg
Takes an event as a parameter.

Execute exactly one of the following blocks of code based on the type of event passed in:

```
Event is SCORE_INCR:
     Add one to CurrentScore
     Call WriteScore
Event is SCORE_DECR:
     If CurrentScore is not 0:
          Subtract one from CurrentScore
          Call WriteScore
Event is GAME_START:
     Set Current Score to 0
     Call WriteScore
     Set DisplayOn to 1
     Call WriteDisplay
Event is GAME_RESET:
     Create HIGH_SCORE event with param of CurrentScore
     Post HIGH_SCORE event to GamePlay service
     Set Current Score to 0
```

```
      Call WriteScore
      Set DisplayOn to 0
      Call WriteDisplay
Event is BLINK:
      Call Blink function
Event is HIGH_SCORE:
      Set CurrentScore to event param
      Call WriteScore
      Set DisplayOn to 1
      Call WriteDisplay
```

WriteScore

```
This function takes no input parameters.

If CurrentScore is greater than 99:
      Set CurrentScore to 0
Initialize Score1 to CurrentScore mod 10
Initialize Score2 to the integer value of CurrentScore divided by 10
Clear the 4 bits controlling each seven seg decoder
Set the first sev seg decoder to the 4 bit value of Score1
Set the second sev seg decoder to the 4 bit value of Score2
```

WriteDisplay

```
This function takes no input parameters.

Set the sev seg display control pin to the value of DisplayOn.
```

Blink

```
This function takes no input parameters.

Set DisplayOn to the logical inverse of its previous value
Call WriteDisplay
```

CornPop.c
Pseudocode for a service that handles the position of the corn servos.
Data private to this module: MyPriority, CurrentCorn1Pulse, CurrentCorn2Pulse, CurrentCorn3Pulse

InitCornPop
Takes as parameter an integer representing the service priority.

```
Set MyPriority to input priority
Set PWM groups controlling corn to a period of 20ms (25000 ticks)
Set CurrentCorn1Pulse, CurrentCorn2Pulse, CurrentCorn3Pulse to
     SEED_PULSE
Set all corn servo pwm channel pulse widths to SEED_PULSE
Post ES_INIT event to self
```

PostCornPop
Takes an event as a parameter.

```
Post input event to ES service with priority of MyPriority.
```

RunCornPop
Takes an event as a parameter

```
Initialize update var as (HARVEST_PULSE-SEED_PULSE)/ PUMPS_TO_GROW
Execute one of the following block of code based on input event type:
     If GAME_RESET:
          Set CurrentCorn1Pulse, CurrentCorn2Pulse,
               CurrentCorn3Pulse to SEED_PULSE
          Set all corn servo pwm channel pulse widths to SEED_PULSE

     If GAME_START:
          Set CurrentCorn1Pulse, CurrentCorn2Pulse,
               CurrentCorn3Pulse to SEED_PULSE
          Set all corn servo pwm channel pulse widths to SEED_PULSE

     If ALIEN_CONTACT:
          If Event Param is LaneOne:
               Set CurrentCorn1Pulse to SEED_PULSE
               Set pwm pulse width for corn 1 servo to SEED_PULSE
          Else if Event Param is LaneTwo:
               Set CurrentCorn2Pulse to SEED_PULSE
               Set pwm pulse width for corn 2 servo to SEED_PULSE
```

```
      Else if Event Param is LaneThree:
            Set CurrentCorn2Pulse to SEED_PULSE
            Set pwm pulse width for corn 2 servo to SEED_PULSE

If WATER_CORN:
      If Event Param is LaneOne:
            Add update to CurrentCorn1Pulse
            If CurrentCorn1Pulse is greater than HARVEST_PULSE:
                  Set CurrentCorn1Pulse to SEED_PULSE
                  Post SCORE_INCR event to SevSeg service
            Set corn 1 servo pwm pulse width to CurrentCorn1Pulse
      Else if Event Param is LaneTwo:
            Add update to CurrentCorn2Pulse
            If CurrentCorn2Pulse is greater than HARVEST_PULSE:
                  Set CurrentCorn2Pulse to SEED_PULSE
                  Post SCORE_INCR event to SevSeg service
            Set corn 2 servo pwm pulse width to CurrentCorn2Pulse
      Else if Event Param is LaneThree:
            Add update to CurrentCorn3Pulse
            If CurrentCorn3Pulse is greater than HARVEST_PULSE:
                  Set CurrentCorn3Pulse to SEED_PULSE
                  Post SCORE_INCR event to SevSeg service
            Set corn 3 servo pwm pulse width to CurrentCorn1Pulse
```

AlienAttacks.c
This state machine sets the random alien attacks in the game.
Data private to the module: CurrentState, MyPriority

InitAttack
Takes as parameter an integer representing the service priority.

Set MyPriority to priority passed in as a parameter
Set CurrentState to AttackOff
Post ES_INIT event to self

PostAttack
Takes an event as a parameter.

Post input event to ES service with priority of MyPriority.

RunAttack
Takes an event as a parameter.

```
If CurrentState is AttackOff and event type is GAME_START:
     Set CurrenState to PlottingAttack
     Create a variable AttackTime
     Set AttackTime to ((random integer mod ATTACK_RATE)+1)*1000
     Start ATTACK_TIMER with AttackTime

Else if CurrentState is PlottingAttack:
     If event is ES_TIMEOUT of ATTACK_TIMER:
          Initialize NumAttacks as a random integer mod 3
          Repeat the following NumAttacks times:
               Initialize AlienN as a random integer mod 3, + 1
               Create an ALIEN_ATTACK event with param AlienN
               Post the ALIEN_ATTACK event to AlienFSM service
          End repeat
          Set AttackTime to ((random integer mod
     ATTACK_RATE)+1)*1000
          Start ATTACK_TIMER with AttackTime
     Else if event is GAME_RESET:
          Stop ATTACK_TIMER
          Set CurrentState to AttackOff
```

GameClock.c
Pseudocode for the GameClock module, which controls the game clock servo.
Data private to this module: MyPriority, CurrentPulseWidth

InitGameClock
Takes as parameter an integer representing the service priority.

```
Set MyPriority to input priority
Set CurrentPulseWidth to GAME_START_PULSE
Set pulse width of pwm channel for GameClock servo to
GAME_START_PULSE
Post ES_INIT event to self
```

PostGameClock
Takes an event as a parameter.

```
Post input event to ES service with priority of MyPriority.
```

RunGameClock
Takes an event as a parameter.

```
Set update var to ((GAME_END_PULSE - GAME_START_PULSE) * UPDATE_RATE)
     / LENGTH_OF_GAME;
Execute one of the following blocks of code based on the type of the
event passed in:

GAME_START event:
     Set CurrentPulseWidth to GAME_START_PULSE
     Add update to CurrentPulseWidth
     Set pulse of pwm channel for GameClock servo to
CurrentPulseWidth
     Start GAME_CLOCK_ROTATE_TIMER with UPDATE_RATE time

ES_TIMEOUT event:
     If event param is GAME_CLOCK_ROTATE_TIMER:
          Add update to CurrentPulseWidth
          Set pulse of pwm channel for GameClock servo to
               CurrentPulseWidth
          If CurrentPulseWidth is less than GAME_END_PULSE:
               Start GAME_CLOCK_ROTATE_TIMER with UPDATE_RATE time

GAME_END event:
```

```
      Set CurrentPulseWidth to GAME_END_PULSE
      Set pulse of pwm channel for GameClock servo to
CurrentPulseWidth

GAME_RESET event:
      Set CurrentPulseWidth to GAME_START_PULSE
      Set pulse of pwm channel for GameClock servo to
CurrentPulseWidth
      Stop GAME_CLOCK_ROTATE_TIMER
```

AudioService.c
Pseudocode for the AudioService Module (a service that implements a state machine). This module is for a Adafruit AudioFX soundboard.
Data private to this module: MyPriority, CurrentState

InitAudioService
Takes as parameter an integer representing the service priority. Returns true.

Initialize the MyPriority variable with the passed in parameter.
Set MyPriority to input priority
Get CurrentRegister value
Set all bits corresponding to audio channels high (e.g. AUDIO_1_OFF)
Set CurrentState to WaitingForSignal
Post ES_INIT event to self

PostAudioService
Takes an event as a parameter.

Post input event to ES service with priority of MyPriority.

RunAudioService
Takes an event as a parameter.
Returns ES_NO_EVENT.

```
If CurrentState is WaitingForSignal:
      If event is MEGABLAST
            PlayAudio with parameter AlienBlast
            Set CurrentState to HoldingLow
            Start AUDIO_HOLD_TIMER with AUDIO_DELAY
      Else if event is GAME_END
            PlayAudio with parameter Celebration
            Set CurrentState to HoldingLow
            Start AUDIO_HOLD_TIMER with AUDIO_DELAY
      Else if event is ALIEN_BLAST
            PlayAudio with parameter ButtonPress
            Set CurrentState to HoldingLow
            Start AUDIO_HOLD_TIMER with AUDIO_DELAY
Else if CurrentState is HoldingLow:
      If event is ES_TIMEOUT of AUDIO_HOLD_TIMER
            Call ResetAudioLines
            Set CurrentState to WaitingForSignal
```

PlayAudio
Takes no ClipToPlay parameter, returns nothing.

Pulls corresponding audio pin low on Adafruit soundboard.

If ClipToPlay is AlienBlast
      Set audio 1 bit high (e.g. AUDIO_1_ON)
Else if ClipToPlay is Celebration
      Set audio 2 bit high (e.g. AUDIO_2_ON)
Else if ClipToPlay is ButtonPress
      Set audio 3 bit high (e.g. AUDIO_3_ON)

ResetAudioLine
Takes no parameters, returns nothing.
Pulls all used audio lines high.

Set all audio lines high (e.g. write (AUDIO_1_OFF | AUDIO_2_OFF | …))

MegaBlast.c
Pseudocode for the MegaBlast Module (a service that implements a state machine)
Data private to the module: CurrentState, MyPriority

InitMegaBlast
Takes as parameter an integer representing the service priority.
Returns true.

```
Set MyPriority to input priority
Set CurrentState to LightOff
Post ES_INIT event to self
```

PostMegaBlast
Takes an event as a parameter.

```
Post input event to ES service with priority of MyPriority.
```

RunMegaBlast
Takes an event as a parameter.
Returns ES_NO_EVENT.

```
If event is GAME_RESET
      Stop MEGABLAST_TIMER
      Set CurrentState to BlastInactive
      Call LightOff
Else:
      If CurrentState is BlastInactive
            If event is GAME_START
                  Start MEGABLAST_TIMER with MEGABLAST_TIME
            Else if event is ES_TIMEOUT of MEGABLAST_TIMER
                  Call LightOn
                  Set CurrentState to BlastAvailable
                  Start MEGABLAST_TIMER with BLINK_TIME
      Else if CurrentState is BlastAvailable
            If event is PALM_COVERED
                  Set CurrentState to BlastPrimed
            Else if event is ES_TIMEOUT of MEGABLAST_TIMER
                  Call Blink()
                  Start MEGABLAST_TIMER with BLINK_TIME
      Else if CurrentState is BlastPrimed
            If event is PALM_UNCOVERED
                  Set CurrentState to BlastUnavailable
            Else if event is LEVERDB_ButtonDown
```

```
                Post ALIENBLAST event to Lane 1 of AlienFSM
                Post ALIENBLAST event to Lane 2 of AlienFSM
                Post ALIENBLAST event to Lane 3 of AlienFSM
                POST MEGABLAST event to AudioService
                Start MEGABLAST_TIMER with MEGABLAST_TIME
                Call LightOff()
                Set CurrentState to BlastInactive
          Else if event is ES_TIMEOUT of MEGABLAST_TIMER
                Call Blink()
                Start MEGABLAST_TIMER with BLINK_TIME
```

QueryBlastState

Takes no parameters. Returns CurrentState.

```
Return CurrentState
```

LightOn

Takes no parameters, returns nothing.

```
Get CurrentRegister value
Create SetMask by left shifting 0x1 by MB_LIGHT_PIN_OFFSET
Set CurrentRegister to have light pin on, using set mask
```

LightOff

Takes no parameters, returns nothing.

```
Get CurrentRegister value
Create ClearMask by left shifting 0x1 by MB_LIGHT_PIN_OFFSET and inverting
Set CurrentRegister to have light pin off, using clear mask
```

Blink

Takes no parameters, returns nothing.

```
Initialize static LightStateVariable to 0
InvertLightState to determine NewLightState
If NewLightState is on
      Call LightOn()
If NewLightState is off
      Call LightOff()
```

ShiftRegister.c
Pseudocode for the module that acts as the low level interface to a write-only shift register.
Data private to the module: LocalRegisterImage

SR_Init
Takes no parameters, returns nothing.

```
Set PB0, PB1, and PB2 to be data output
Write data and SCLK to lo, RCLK to hi
```

SR_GetCurrentRegister
Takes no parameters, returns LocalRegisterImage

```
Return LocalRegisterImage
```

SR_Write
Takes uint32_t NewValue as input, returns nothing.

```
Set LocalRegisterImage to NewValue
Lower register clock
For each bit in NewValue
      If bit31 of NewValue is high, write data to HI
      Else write data to LO
      Raise shift clock
      Lower shift clock
      Left shift NewValue by 1
Raise RCLK to HI to latch new data
```

LaneSelection.c

InitLaneSelectionFSM
Takes a priority number, returns True.
Initialize the MyPriority variable with the passed in parameter.
Initialize PWM periods for groups

Set CurrentState to be LanesOff
Set all LaneSelection LEDs to be off
Post Event ES_INIT to LaneSelectionFSM queue (this service)
End of InitLaneSelectionFSM

PostLaneSelectionFSM
Post an event to LaneSelectionFSM queue (this service)
End of PostLaneSelectionFSM

RunLaneSelectionFSM
The EventType field of ThisEvent will be one of: GAME_START, LaneAimed, LEDBrightnessChange, GAME_RESET. The parameter field of LaneAimed event will be an integer between 0-3 indicating which lane is aimed or none of the lane is aimed. The parameter field of LEDBrightnessChange will be a value between 0-4095 which is read from the analog input.
Returns ES_NO_EVENT
Local Variables: CurrentPins
Get a copy of the 32-bit shift register value and store in CurrentPins

Based on the state of the CurrentState variable choose one of the following blocks of code:

```
     CurrentState is LanesOff
          Call InitializeLanes()
          If ThisEvent is GAME_START
               Set CurrentState to LaneEmpty
          Endif
     End LanesOff block

     CurrentState is not LanesOff
          If ThisEvent is LaneAimed
               If EventParameter is 1
                    Mask CurrentPins with Lane1LEDStripe pin high
to SR_Write
                    Set CurrentState to LaneOne
```

```
                Elseif EventParameter is 2
                     Mask CurrentPins with Lane2LEDStrip pin high to
SR_Write
                     Set CurrentState to LaneTwo
                Elseif EventParameter is 3
                     Mask CurrentPins with Lane3LEDStrip pin high to
SR_Write
                     Set CurrentState to LaneThree
                Else
                     Mask CurrentPins with all three LaneLEDStrip
pins low to SR_Write
                     Set CurrentState to LaneEmpty
                Endif
           Elseif ThisEvent is LEDBrightnessChange
                     Set PWM duty to each LaneSelectionLED pin
according to the value of EventParameter
           Endif
     End (not LanesOff) block

If ThisEvent is GAME_RESET
     Mask CurrentPins with all three LaneLEDStrip pins low to
SR_Write
     Set CurrentState to LanesOff
     Set PWM duty to 0 for all three LaneSelectionLED pins
Endif

Return ES_NO_EVENT
End of RunLaneSelectionFSM
```

QueryLaneSelectionFSM

```
     Return the current state of LaneSelection state machine
End of QueryLaneSelectionFSM
```

InitializeLanes

```
Local Variables: ADResults
Read analog inputs and store to ADResults
If reading OK for Lane1
     Post LaneAimed event with EventParameter 1 to LaneSelectionFSM
Elseif reading OK for Lane2
     Post LaneAimed event with EventParameter 2 to LaneSelectionFSM
Elseif reading OK for Lane3
     Post LaneAimed event with EventParameter 3 to LaneSelectionFSM
Else
```

```
      Post LaneAimed event with EventParameter 0 to LaneSelectionFSM
Endif
End of InitializeLanes
```

ButtonSwitchService.c

InitButtonSwitchDebounceService
Initialize the MyPriority variable with the passed in parameter
Set direction of two buttons and one switch pinout to be input
Set CurrentState for all three button/switch to Debouncing
Start all three debounce timers (timer posts to ButtonSwitchDebounceService)
Post Event ES_INIT to ButtonSwitchDebounceService queue (this service)
End of InitButtonSwitchDebounceService

PostButtonSwitchDebounceService
Post an event to ButtonSwitchDebounceService queue (this service)
End of PostButtonSwitchDebounceService

RunButtonSwitchDebounceService
The EventType field of ThisEvent will be one of: ES_TIMEOUT, ButtonUp, ButtonDown. The parameter field of ES_TIMEOUT event will be one of the three debounce timers. The parameter field of ButtonUp and ButtonDown events will be 1 or 2 or 3.
Returns ES_NO_EVENT

The following code repeats for the rest FIRE button and LEVER switch -
Based on the state of the CurrentWaterState variable choose one of the following blocks of code:

```
     CurrentWaterState is Debouncing
          If ThisEvent is ES_TIMEOUT and EventParameter is
WATER_DEBOUNCE_TIMER
               Set CurrentWaterState to Ready2Sample
          Endif
     End of Debouncing block

     CurrentWaterState is Ready2Sample
          If ThisEvent is ButtonUp and EventParameter is 2
               Set CurrentWaterState to Debouncing
          Elseif ThisEvent is ButtonDown and EventParameter is 2
               Set CurrentWaterState to Debouncing
               Post WATERDB_ButtonDown event to GamePlay service
          Endif
     End of Ready2Sample block
```

```
Return ES_NO_EVENT
End of RunButtonSwitchDebounceService
```