```
/****************************************************************************
 Module
   AudioService.c

 Revision
   1.0.1

 Description
   This implements a module to control audio feedback.

 Notes

 History
 When           Who     What/Why
 11/12/19                              Amanda S.    Create file
 -------------- ---     --------

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

/*----------------------------- Module Code -----------------------------*/
/****************************************************************************
 Function
     InitAudioService

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
     Saves away the priority, and does any
     other required initialization for this service
 Notes

 Author
  Amanda Spyropoulos, 11/13/19, 14:44
****************************************************************************/
bool InitAudioService(uint8_t Priority)
{
  // enable and set up pins PD0 - PD3, PD6-PD7, PDF0, PF1 as digital open drain
outputs to drive the sound board.
  // PD0 = GAME START
  // PD1 = DMG TAKEN
  // PD2 = RAD BLOCKED
  // PD6 = GAME WON
  // PD3 = GAME LOST
  // PD7 = WELCOME MODE

  //set all the pins to be high (so really floating because open drain output)

  // initialize timer

  //start welcome noise playing on loop


  //post init event to this service

}

/****************************************************************************
 Function
    PostAudioService

 Parameters
    EF_Event ThisEvent ,the event to post to the queue

 Returns
    bool false if the Enqueue operation failed, true otherwise

 Description
    Posts an event to this state machine's queue
 Notes

 Author
    Amanda Spyropoulos, 11/13/19, 14:47
****************************************************************************/
bool PostAudioService( ES_Event_t ThisEvent )
{
  return ES_PostToService( MyPriority, ThisEvent);
}


/****************************************************************************
 Function
   RunAudioService

 Parameters
  ES_Event : the event to process

 Returns
  ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
```

```
  Implements the state machine to control the audio service module.
 Notes

 Author
   Amanda Spyropoulos, 11/13/19, 14:47
****************************************************************************/


ES_Event_t RunAudioService(ES_Event_t ThisEvent)
{
  //Based on the state of the CurrentState variable choose one of the following blocks of
code:
   {
   // if we are in WaitingForEvent state:
   {
     // based on the event type, choose one of the following blocks of code:
     {
      // PD3 = GAME WON
       {
        // make all pins floating / high
        // start timer to make sure we drive pin high for at least 50 ms
        // our next state is GameOver
        // note that we won the game

       };

       // PD6 = GAME LOST
       {
       // make all pins floating / high
        // start timer to make sure we drive pin high for at least 50 ms
        // our next state is GameOver
        // note that we lost the game
       };

       // PD0 = GAME START
       {
       // make all pins floating / high
       // drive corresponding pin low
       // start timer to make sure we drive pin low for at least the required time
       // our next state is WaitingToPlay
       };

       // PD1 = DMG TAKEN
       {
       // make all pins floating / high
       // drive corresponding pin low
       // start timer to make sure we drive pin low for at least the required time
       // our next state is WaitingToPlay
       };
       // PD2 = RAD BLOCKED
       {
       // make all pins floating / high
       // drive corresponding pin low
       // start timer to make sure we drive pin low for at least the required time
       // our next state is WaitingToPlay
       };

     }
   }
     break;
   // if we are in the WaitingToPlay state
   {
     // based on the event type, choose one of the following blocks of code:
     {
       // if we get a TIMEOUT event
        {
         // if we get the TIMEOUT from the corresponding timer, it's time to stop playing
sound
         {
           // make all pins floating / high
           // our next state is ResettingPins;
           // start timer to make sure we drive pin high for at least 50 ms
         }
        }

        // PD3 = GAME WON
       {
         // make all pins floating / high
        // start timer to make sure we drive pin high for at least 50 ms
        // our next state is GameOver
        // note that we won the game

       };

       // PD6 = GAME LOST
       {
        // make all pins floating / high
        // start timer to make sure we drive pin high for at least 50 ms
        // our next state is GameOver
        // note that we lost the game
       };
       // PD0 = GAME START
       {
        // make all pins floating / high
```

```
            // our next state is ResettingPins;
            // start timer to make sure we drive pin high for at least 50 ms
        };
        // PD1 = DMG TAKEN
        {
        // make all pins floating / high
        // our next state is ResettingPins;
        // start timer to make sure we drive pin high for at least 50 ms
        // defer the LifeLost event
        };

        // PD2 = RAD BLOCKED
        {
        // make all pins floating / high
        // our next state is ResettingPins;
        // start timer to make sure we drive pin high for at least 50 ms
        // defer the RadBlocked event
        };

      }

    }
    // if we are in state ResettingPins:
    {
      // based on the event type, choose one of the following blocks of code:
      {
        // if we get a TIMEOUT event
        {
          // if the TIMEOUT is from the corresponding TIMER
          {
            // if we have no deferred events, then go to the WaitingForEvent state
            // otherwise, start playing the deferred event's sound and go to WaitingToPlay

            }
            }

          }
        }

        // PD3 = GAME WON
        {
          // make all pins floating / high
         // start timer to make sure we drive pin high for at least 50 ms
         // our next state is GameOver
         // note that we won the game

        };
        // PD6 = GAME LOST
        {
         // make all pins floating / high
         // start timer to make sure we drive pin high for at least 50 ms
         // our next state is GameOver
         // note that we lost the game
        };

      }
    }

    // if we're in the GameOver state:
    {
      // choose a block of code depending on the event we receive
          // if we get a TIMEOUT
         {
           // if the TIMEOUT is from the corresponding TIMER
          {
                // set the corresponding pin lo to play the appropriate (game won or game
lost) noise
           // set the timer to ensure we play the sound for the correct amount of time
           // our next state is WaitingToPlay;

          }
         }

    }


  //Set CurrentState to NextState

  //Return any event (typically ES_NO_EVENT)

}

/****************************************************************************
 private functions
 ****************************************************************************/



/****************************************************************************
 Function
    ResetPins

 Parameters
```

```
   N/A

 Returns
    nothing

 Description
    sets all pins high (really floating b/c open drain output)
 Notes

 Author
    Amanda Spyropoulos, 11/13/19, 14:47
****************************************************************************/
static void ResetPins(void)
{
  // set all pins high (really floating)
 }

/*------------------------------- Footnotes -------------------------------*/
/*------------------------------ End of file ------------------------------*/
```