```
/****************************************************************************
 Module
   ServoService.c

 Revision
   1.0.1

 Description
   This service controls the servo to allow "push to talk" functionality during the game.

 Notes
 Since each of us have different servos and different mechanical set-ups, feel free to
 #define different numbers in the duty cycle if they work better for you (just make sure
 everyone can use their settings, so use if statements). Make sure to keep
 your duty cycle values between 8 (corresponds to 1ms pulse width. In PWM6DCH, BIT1HI,
 low everywhere else) and 15 (corresponds to 2ms pulse width. In PWM6DCH, BIT0HI | BIT1HI and
 BIT7HI | BIT6HI in PWM6DCL).

 History
 When           Who     What/Why
 -------------- ---     --------
 01/16/12 09:58 jec      began conversion from TemplateFSM.c
 05/14/20       LG       wrote service functions
****************************************************************************/
/*---------------------------- Include Files ----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine
*/
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "ServoService.h"
#include "GameplayFSM.h"

/*---------------------------- Module Defines ----------------------------*/
#define PWMPeriodTicks 155 // this corresponds to 50 Hz PWM frequency
#define RELEASE_DELAY 30

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
void InitializePWM(void);
/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;

/*---------------------------- Module Code ----------------------------*/
/****************************************************************************
 Function
    InitTemplateService

 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
    J. Edward Carryer, 01/16/12, 10:00
****************************************************************************/
bool InitServoService(uint8_t Priority)
```

```
{
 ES_Event_t ThisEvent;

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

/****************************************************************************
 Function
    PostServoService

 Parameters
    EF_Event_t 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
    J. Edward Carryer, 10/23/11, 19:25
****************************************************************************/
bool PostServoService(ES_Event_t ThisEvent)
{
 return ES_PostToService(MyPriority, ThisEvent);
}

/****************************************************************************
 Function
   RunServoService

 Parameters
  ES_Event_t : the event to process

 Returns
  ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
  add your description here
 Notes

 Author
  J. Edward Carryer, 01/15/12, 15:23
****************************************************************************/
ES_Event_t RunServoService(ES_Event_t ThisEvent)
{
 ES_Event_t ReturnEvent;
 ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

 if(ThisEvent.EventType == ES_PRESS_KEYBOARD) {
    PressKeyboard();
 }

 if(ThisEvent.EventType == ES_RELEASE_KEYBOARD) {
    ES_Timer_InitTimer(SERVO_RELEASE_TIMER, RELEASE_DELAY);
 }
```

```
 if(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam == SERVO_RELEASE_TIMER) {
    ReleaseKeyboard();
 }

 return ReturnEvent;
}

/****************************************************************************
 Function
 PressKeyboard

 Parameters
 None

 Returns
 Nothing

 Description
 This function sets the pulse width of the servo to be 1.66 ms, which will move it
 towards the right (CW). This action should engage the keyboard.

 Notes
 Feel free to exceed this value, but do not go higher than 14. The servos
 might not actually be able to take 2 ms pulses.

 Author
   Lisa Gardner 05/14/2020
****************************************************************************/

void ReleaseKeyboard (void) {

   PWM6DCH = 8;
   PWM6DCL = 3;

   return;
}

/****************************************************************************
 Function
 ReleaseKeyboard

 Parameters
 None

 Returns
 Nothing

 Description
 This function sets the servo back to the left (CCW) by creating a pulse width
 of 1 ms.

 Notes
 Again feel free to adjust to your set-up. Just do not go lower than 8 in the duty cycle
 registers.

 Author
 Lisa Gardner 05/14/2020
****************************************************************************/

void PressKeyboard (void) {
   PWM6DCH = 6; // set initial pulse width to 0
   PWM6DCL = 3;

   return;
}

/****************************************************************************
 Function
```

```
 InitializePWM

 Parameters
 None

 Returns
 Nothing

 Description
 Sets up PWM for Timer 2 on RC2. Initial duty cycle is set to 0.

 Notes

 Author
 Lisa Gardner 05/14/2020
************************************************************************/

void InitializePWM(void) {
    PWM6POL = 0; // output active high (default value is 0)
    T2PR = PWMPeriodTicks; // load T2PR with PWM period (155) that corresponds to 50 Hz

    PWM6DCH = 8; //
    PWM6DCL = 3;

    //combination sets 150

    T2CLKCON = 6; // MFINTOSC (31.25 kHz)
    T2CKPS1 = 1; // prescaler is 1:4
    TMR2IF = 0; // clear the interrupt flag
    T2ON = 1; // timer 2 is on

    TMR2IE = 0; // disable the interrupt for timer 2

    ANSELC &= BIT2LO; // set pin as digital I/O
    TRISC &= BIT2LO; // set pin as output

    RC2PPS = 0x0E; // set up RC2 for PWM6
    PWM6EN = 1; // enable the PMW module

    return;
}

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