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

 Revision
   1.0.1

 Description
   This is a template file for implementing a simple service under the
   Gen2 Events and Services Framework.

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 01/16/12 09:58 jec      began conversion from TemplateFSM.c
*******************************************************************************/
/*---------------------------- 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"

/*---------------------------- Module Defines ----------------------------*/
#define PRESCALE 8
#define PBCLK 20000000

#define TENSION_TIMING 2000
#define LOCK_TIMING 1000

#define MIN_POS 2700//lowered
#define MAX_POS 6200//raised

#define MIN_PAIRED 2000
#define MAX_PAIRED 5000

#define MIN_LOCKED 5700
#define MAX_LOCKED 2100

#define LOCK_SERVO OC1RS
#define TENSION_SERVO OC4RS
#define PAIRED_SERVO OC5RS

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
void InitServo(void);

/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;

/*---------------------------- Module Code ----------------------------*/
/*******************************************************************************
 Function
     InitServoService
```

```
 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;

  InitServo();

  // 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

  switch(ThisEvent.EventType){
      case ES_TIMEOUT: {
          if (TENSION_RISE_TIMER == ThisEvent.EventParam){
              // Once tension servo is raised, lock the BYTER in UP posn
//              puts("locking the servo\r\n");
              LOCK_SERVO = MAX_LOCKED;
              ES_Timer_InitTimer(LOCK_TIMER, LOCK_TIMING);
          }

          if (LOCK_TIMER == ThisEvent.EventParam){
              // once locked, lower the tension servo to tension the spring
              OC4RS = MIN_POS;
              ES_Timer_InitTimer(TENSION_FALL_TIMER, TENSION_TIMING);
          }

          if (TENSION_FALL_TIMER == ThisEvent.EventParam){
              // once the tension servo is down, ready to fire

          }
      }
      break;

      case ES_INIT: {
        PAIRED_SERVO = MIN_PAIRED;
        LOCK_SERVO = MIN_LOCKED;

        // Raise tension servo to UP to start priming process
        OC4RS = MAX_POS;
        ES_Timer_InitTimer(TENSION_RISE_TIMER, TENSION_TIMING);
      }
      break;

      case ES_PAIRED: {
        // raise the paired servo
        PAIRED_SERVO = MAX_PAIRED;

        // Raise tension servo to UP to start priming process
        OC4RS = MAX_POS;
        ES_Timer_InitTimer(TENSION_RISE_TIMER, TENSION_TIMING);
      }
      break;
```

```
      case ES_UNPAIR_CMD: {
        // lower all the servos to the default position
        PAIRED_SERVO = MIN_PAIRED;
        LOCK_SERVO = MIN_LOCKED;
        OC4RS = MIN_POS;
      }
      break;

      case ES_PRIME_BYTER: {
        // start process to prime the byter
//        puts("retracting the byter for the next bite\r\n");
        OC4RS = MAX_POS;
        ES_Timer_InitTimer(TENSION_RISE_TIMER, TENSION_TIMING);
      }
      break;

      case ES_BITE: {
        // lower the lock servo
        puts("RAWR I AM BITING\r\n");
        LOCK_SERVO = MIN_LOCKED;
      }
      break;

  }

  return ReturnEvent;
}

/*****************************************************************************
 private functions
 *****************************************************************************/
void InitServo(void){
    // -------------------------- Timer 2 -------------------------------
    T2CONbits.ON = 0;                   // turn of T2
    T2CONbits.SIDL = 0;                 // on in idle mode
    T2CONbits.TCS = 0;                  // use PBCLK
    T2CONbits.TCKPS = 0b011;            // prescale of 8
    T2CONbits.TGATE = 0;                // turn off gated mode
    TMR2 = 0;                           // clear T2
//    PR2 = PBCLK/PRESCALE*20/1000 - 1;    // 20 ms period
    PR2 = 50000;                        // 20 ms period
    // -----------------------------------------------------------------


    // -------------------------- OC 5 -------------------------------
    // Paired servo - RB13 - pin 24
    OC5CONbits.ON = 0;                  // turn off OC5
    TRISBbits.TRISB13 = 0;              // RB13 output
    ANSELBbits.ANSB13 = 0;              // RB13 digital
    OC5CONbits.OCTSEL = 0;              // timer 2 timebase
    OC5CONbits.OC32 = 0;                // 16 bit mode
    OC5CONbits.OCM = 0b110;             // PWM non-fault mode
    RPB13R = 0b0110;                    // map OC5 to RPB15
    OC5RS = MIN_POS;
    OC5R = MIN_POS;
    OC5CONbits.OCM = 0b110;             // PWM non-fault mode
    // -----------------------------------------------------------------
```

```
    // ---------------------------- OC 1 --------------------------------
    // Lock servo - RB15 - pin 26
    OC1CONbits.ON = 0;                  // turn off OC1
    TRISBbits.TRISB15 = 0;              // RB15 output
    ANSELBbits.ANSB15 = 0;              // RB15 digital
    OC1CONbits.OCTSEL = 0;              // timer 2 timebase
    OC1CONbits.OC32 = 0;                // 16 bit mode
    OC1CONbits.OCM = 0b110;             // PWM non-fault mode
    RPB15R = 0b0101;                    // map OC1 to RPB15
    OC1RS = MIN_POS;
    OC1R = MIN_POS;
    OC1CONbits.OCM = 0b110;             // PWM non-fault mode
    // ------------------------------------------------------------------


    // ---------------------------- OC 4 --------------------------------
    // Tension servo - RA4 - pin 12
    OC4CONbits.ON = 0;                  // turn off OC4
//    TRISAbits.TRISA4 = 0;               // RA4 output
//    ANSELA = 0;                         // set all of A to digital
    TRISBbits.TRISB2 = 0;
    ANSELBbits.ANSB2 = 0;
    OC4CONbits.OCTSEL = 0;              // timer 2 timebase
    OC4CONbits.OC32 = 0;                // 16 bit mode
    OC4CONbits.OCM = 0b110;             // PWM non-fault mode
//    RPA4R = 0b0101;                     // map OC4 to RPA4
    RPB2R = 0b0101;
    OC4RS = MIN_POS;
    OC4R = MIN_POS;
    OC4CONbits.OCM = 0b110;             // PWM non-fault mode
    // ------------------------------------------------------------------


    // --------------------- Turn on OC and Timer -----------------------
    T2CONbits.ON = 1;                   // turn on T2
    OC5CONbits.ON = 1;                  // Turn on OC5
    OC1CONbits.ON = 1;                  // Turn on OC1
    OC4CONbits.ON = 1;                  // Turn on OC4
    // ------------------------------------------------------------------
}

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