```
/*******************************************************************************
 Module
   UltrasonicService.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 "UltrasonicService.h"
#include <sys/attribs.h>
#include "dbprintf.h"
#include "SPIFollower.h"

#define T2_PERIOD 0xFFFF
#define DISTANCE_CLOSE_THRESHOLD 500

#define DISTANCE_CLOSE_COMMAND 0x21
#define DISTANCE_FAR_COMMAND 0x22

/*---------------------------- Module Defines ----------------------------*/

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

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



static void configUltrasonicInterrupts(void);
static void configTimer2(void);
static void configInputCapture2(void);
static void configOC3(void);



typedef union{
    uint32_t FullCount;
    uint16_t Counters[2];
    //1 is the rollover counter
    //0 is the current count;
```

```
}PulseCount_t;

volatile static PulseCount_t counter;
volatile static uint16_t Period;
volatile static uint16_t rollovers;
static bool currentDistance;

/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitUltrasonicService

 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 InitUltrasonicService(uint8_t Priority)
{
       //configure input capture 2
       //configure OC3
       //configure timer2
       //configure ultrasonic interrupts
}


/****************************************************************************
 Function
    RunUltrasonicService

 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 RunUltrasonicService(ES_Event_t ThisEvent)
{

      //if new ultra sonic pulse
             //ignore extraneous pulses

             //check if distance measurement is close
             //if distance changed
```

```
                    //if distance close
                          //setup distance close event


                    //if distance far
                          //setup distance far event
                    //tell SPI follower of new distance

             //save current distance
}

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

static void configUltrasonicInterrupts(void){
       //disable global interrupts
    //set IC2 priority
    //set Timer2Priority

    //Clear any pending interrupt flags
    //Enable IC2 and T2 interrupts
       //enable global interrupts
}

static void configTimer2(void){
       //disable timer 2
    //CONFIG TIMER 2
    //disable gated time accumulation mode
    //select PBCLK as source
    //prescale of 32
    //Clear timer register
    //set timer period
    //enable timer 2
}


static void configInputCapture2(void){
       //disable IC2
       //operate in idle mode
       //16 bit mode
       //select timer 2

       //capture every event
       //every edge, specified edge first
       //start with first edge
       //map IC2 to pin B10

       //enable IC2
}

static void configOC3(void){
       //disable OC 3

       //select timer 2 as the source
       //cont operation in idle mode
       // set OC3 mode to PWM mode
       //32 bit compare mode off
       //set initial duty cycle
       //Map RPB9 to OC3
```

```
	//enable OC3
}



void __ISR(_INPUT_CAPTURE_2_VECTOR,IPL7SOFT) UltraSonicCapture(void){



	//read the captured time

	//if rollover has occurred
		//increment roll-over counter
		//clear rollover flag
	//calculate Period
	//save last time

	//repeat until IC buffer is empty
	//tell service new pulse has been received
	//clear IC flag

}

void __ISR(_TIMER_2_VECTOR,IPL6SOFT) Timer2Rollover(void){
	//if flag set
		//increment roll-over counter
		//clear flag
}


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