```
/*******************************************************************************
 Module
   BeaconService.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 "PIC32PortHAL.h"
#include "BeaconService.h"
#include <proc/p32mx170f256b.h>
#include <pic32m_builtins.h>
#include <sys/attribs.h>
#include "math.h"
#include "ES_Events.h"
#include "ES_Timers.h"
#include "SPIFollower.h"
#include "dbprintf.h"

/*---------------------------- Module Defines ----------------------------*/
#define ONE_SEC 1000
#define _100_MS ONE_SEC/10

#define T3_PERIOD 0xFFFF

#define BEACON_A_COMMAND 0x1A
#define BEACON_B_COMMAND 0x1B
#define BEACON_C_COMMAND 0x1C
#define BEACON_D_COMMAND 0x1D
#define NO_BEACON_COMMAND 0x1E

#define BEACON_A_PERIOD 300
#define BEACON_B_PERIOD 500
#define BEACON_C_PERIOD 700
#define BEACON_D_PERIOD 1100

#define TOLERANCE 50


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

```
}PulseCount_t;

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
static void configBeaconInterrupts(void);
static void configInputCapture3(void);
static void configTimer3(void);



/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;
volatile static PulseCount_t counter;
volatile static uint16_t Period;
volatile static uint16_t rollovers;

static uint8_t currentBeacon = NO_BEACON_COMMAND;

static uint16_t currentBeaconPeriod;



/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitBeaconService

 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 InitBeaconService(uint8_t Priority)
{
  ES_Event_t ThisEvent;

  MyPriority = Priority;

  configTimer3(); //setup timer3
  configInputCapture3();//setup IC3
  configBeaconInterrupts();//configure interrupts


  /********************************************
   in here you write your initialization code
   ********************************************/
  // post the initial transition event
  ThisEvent.EventType = ES_INIT;
```

```
  if (ES_PostToService(MyPriority, ThisEvent) == true)
  {
    return true;
  }
  else
  {
    return false;
  }
}

/****************************************************************************
 Function
     PostBeaconService

 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 PostBeaconService(ES_Event_t ThisEvent)
{
  return ES_PostToService(MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    RunBeaconService

 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 RunBeaconService(ES_Event_t ThisEvent)
{
  ES_Event_t ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
  /********************************************
   in here you write your service code
   ********************************************/
   switch (ThisEvent.EventType)
  {
    case ES_INIT:
    {
```

```
    }
    break;

    case ES_TIMEOUT:
    {
         if (ThisEvent.EventParam == NO_PULSE_TIMER){
            currentBeacon = NO_BEACON_COMMAND;//no more pulses means not looking at a
beacon
            //tell SPI follower
            ES_Event_t NewEvent;
            NewEvent.EventType = ES_NEW_BEACON;
            NewEvent.EventParam = currentBeacon;
            PostSPIFollower(NewEvent);
            Period = 0;
            DB_printf("TIMED OUT \r\n");
        }
    }
    break;

     case ES_NEW_PULSE:
    {
      ES_Timer_InitTimer(NO_PULSE_TIMER,100);//timer for resetting if no pulses

      //take period measurement and find out what beacon it is
      currentBeaconPeriod = Period * 2.0 / 10.0; //period in micro seconds

      //check if period is within tolerance
      uint8_t newBeacon = NO_BEACON_COMMAND;
      if (currentBeaconPeriod <= BEACON_A_PERIOD + TOLERANCE && currentBeaconPeriod >=
BEACON_A_PERIOD - TOLERANCE){
            newBeacon =  BEACON_A_COMMAND;
      }else if (currentBeaconPeriod <= BEACON_B_PERIOD + TOLERANCE &&
currentBeaconPeriod >= BEACON_B_PERIOD - TOLERANCE){
            newBeacon =  BEACON_B_COMMAND;
      }else if (currentBeaconPeriod <= BEACON_C_PERIOD + TOLERANCE &&
currentBeaconPeriod >= BEACON_C_PERIOD - TOLERANCE){
            newBeacon =  BEACON_C_COMMAND;
      }else if (currentBeaconPeriod <= BEACON_D_PERIOD + TOLERANCE &&
currentBeaconPeriod >= BEACON_D_PERIOD - TOLERANCE){
            newBeacon =  BEACON_D_COMMAND;
      }

      //DB_printf("%d \r\n",currentBeaconPeriod);

      if (currentBeacon != newBeacon){//if beacon has changed
        ES_Event_t NewEvent;
        NewEvent.EventType = ES_NEW_BEACON;
        NewEvent.EventParam = newBeacon;
        PostSPIFollower(NewEvent);//tell SPI follower what the new beacon is
      }
      currentBeacon = newBeacon; //save current beacon

    }
    break;

    default:
    {}
     break;
```

```
    }

   return ReturnEvent;
}

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

static void configBeaconInterrupts(void){
    __builtin_disable_interrupts();

    //set IC3 priority
    IPC3bits.IC3IP = 7;
    //set Timer3Priority
    IPC3bits.T3IP = 6;

    //Clear any pending interrupt flags
    IFS0CLR = _IFS0_T3IF_MASK;
    IFS0CLR = _IFS0_IC3IF_MASK;

    //Enable IC3 and T3 interrupts
    IEC0SET = _IEC0_T3IE_MASK;
    IEC0SET = _IEC0_IC3IE_MASK;

    __builtin_enable_interrupts();//enable global interrupts
}

static void configInputCapture3(void){
    IC3CONbits.ON = 0; //disable IC3
    IC3CONbits.SIDL = 0;//operate in idle mode
    IC3CONbits.C32 = 0;//16 bit mode
    IC3CONbits.ICTMR = 0;//select timer 3

    IC3CONbits.ICI = 0b00;//capture every event
    IC3CONbits.ICM = 0b011;//every rising edge

    IC3R = 0b0100;//map IC3 to pin B8

    IC3CONbits.ON = 1; //enable IC3
}

static void configTimer3(void){
    T3CONbits.ON = 0;//disable timer 3
    //CONFIG TIMER 3
    T3CONbits.TGATE = 0;//disable gated time accumulation mode
    T3CONbits.TCS = 0; //select PBCLK as source
    T3CONbits.TCKPS = 0b010;//prescale of 4
    TMR3 = 0; //Clear timer register
    PR3 = T3_PERIOD;//set timer period

    T3CONbits.ON = 1;//enable timer 3
}


void __ISR(_INPUT_CAPTURE_3_VECTOR,IPL7SOFT) BeaconCapture(void){
    static PulseCount_t lastTime;
    static uint16_t capturedTime;
    static ES_Event_t NewEvent;
```

```
    do{
        capturedTime = (uint16_t)IC3BUF;//read the captured time

        if (IFS0bits.T3IF == 1 && capturedTime < 0x8000){ //check if a rollover
occurred
            rollovers++; //increment roll-over counter
            IFS0CLR = _IFS0_T3IF_MASK;
        }
        counter.Counters[0] = capturedTime;
        counter.Counters[1] = rollovers;

        Period = counter.FullCount - lastTime.FullCount; //calculate period

        lastTime.FullCount = counter.FullCount; //save last time
    }while(IC3CONbits.ICBNE != 0); //continue until the buffer is empty

    NewEvent.EventType = ES_NEW_PULSE; //post event to service
    PostBeaconService(NewEvent);

    IFS0CLR = _IFS0_IC3IF_MASK;
}

void __ISR(_TIMER_3_VECTOR,IPL6SOFT) Timer3Rollover(void){
    __builtin_disable_interrupts();
    if (IFS0bits.T3IF == 1){ //if flag is set
        rollovers++; //increment roll-over counter
        IFS0CLR = _IFS0_T3IF_MASK; //clear flag
    }
    __builtin_enable_interrupts();

}



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