```
/*******************************************************************************
 Module
   XBeeCommSM.c

 Revision
   1.0.1

 Description
   This is a template file for implementing flat state machines under the
   Gen2 Events and Services Framework.

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 01/15/12 11:12 jec      revisions for Gen2 framework
 11/07/11 11:26 jec      made the queue static
 10/30/11 17:59 jec      fixed references to CurrentEvent in RunTemplateSM()
 10/23/11 18:20 jec      began conversion from SMTemplate.c (02/20/07 rev)
*******************************************************************************/
/*---------------------------- 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 "XBeeCommSM.h"
#include "dbprintf.h"
#include <sys/attribs.h>
#include "ByterSM.h"
#include "MotorService.h"
#include "ServoService.h"

/*---------------------------- Module Defines ----------------------------*/
#define T5_PERIOD 62500 //5Hz w/ 64PS
#define STATUS_FRAME_LENGTH 12
#define PAIR_ACK_FRAME_LENGTH 12

//defines for constructing messages
#define START_DELIMITER 0x7E
#define LENGTH_MSB 0x00
#define LENGTH_LSB_STATUS (STATUS_FRAME_LENGTH-4)
#define LENGTH_LSB_PAIR_ACK (PAIR_ACK_FRAME_LENGTH-4)
#define API_IDENTIFIER 0x01
#define FRAME_ID 0x01
#define MESSAGE_ID_STATUS 0x02 //possible message IDs to send
#define MESSAGE_ID_PAIR_ACK 0x05
#define OPTIONS 0x00

#define BOAT_ADDR_MSB 0x21 //address associated with our BOAT XBee
#define BOAT_ADDR_LSB 0x85

//possible message IDs to receive
#define MESSAGE_ID_CMD 0x01
#define MESSAGE_ID_PAIRING 0x03
#define MESSAGE_ID_UNPAIR 0x04

#define INACTIVITY_TIME 2000 //2 second inactivity timer
```

```
#define READBAYT PORTBbits.RB10

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this machine.They should be functions
   relevant to the behavior of this state machine
*/
static void InitUART2(void);
static void configTimer5(void);
static void initBAYT(void);


static uint8_t calcCheckSum(void);
static void doUnpairActions(void);
static void constructStatusMsg(void);
static void constructPairACKMsg(void);
static void parseData(void);
static void storeData(uint8_t byte);

/*---------------------------- Module Variables ----------------------------*/
// everybody needs a state variable, you may need others as well.
// type of state variable should match htat of enum in header file
static XBeeCommState_t CurrentState;


static uint8_t TXMessageLength = 0;
static uint8_t TX_Index = 0;
static uint8_t TXArray[STATUS_FRAME_LENGTH];//length of max message size

static uint8_t RXArray[20];
static uint8_t RXMsgLength = 0;

static uint8_t PairedBosynAddress_MSB = 0x20;
static uint8_t PairedBosynAddress_LSB = 0x85;

static bool justDeflated = false;
static bool lastStatus = false;
static bool TXEnabled = false;

// with the introduction of Gen2, we need a module level Priority var as well
static uint8_t MyPriority;

static bool prevBAYTstatus;
static bool hasBalloon;


/*---------------------------- Module Code ----------------------------*/
/****************************************************************************
 Function
     InitXBeeCommSM

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
     Saves away the priority, sets up the initial transition and does any
     other required initialization for this state machine
 Notes
```

```
 Author
     J. Edward Carryer, 10/23/11, 18:55
******************************************************************************/
bool InitXBeeCommSM(uint8_t Priority)
{
       //init the uart communication
       //configure timer 5
       //init BAYT switch
       //Set constants in TXArray
       // put us into the Initial PseudoState
       // post the initial transition event
}


/*****************************************************************************
 Function
    RunXBeeCommSM

 Parameters
   ES_Event_t : the event to process

 Returns
   ES_Event_t, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   add your description here
 Notes
   uses nested switch/case to implement the machine.
 Author
   J. Edward Carryer, 01/15/12, 15:23
******************************************************************************/
ES_Event_t RunXBeeCommSM(ES_Event_t ThisEvent)
{
// assume no errors

       // If current state is initial Pseudo State
              // only respond to ES_Init
              // now put the machine into the actual initial state


       // Waiting for pairing state
              // parse data event
                     //parse data
              //pair request received event
                     //show paired indicated
                     //construct pair ack msg
                     //enable timer 5 for status messages
                     //start inactivity timer
                     //switch to paired state
              //Send status event
                     //construct status msg
              //Timeout event
                     //Just deflated timer for pairing restrictions expired
                     //set deflated flag
       // Paired state
              // parse data event
                     //parse data
       //send status event
              //construct status msg
```

```
      //valid control recvd event
            //reset inactivity timer
      //Bayt deflated event
            //complete all actions related to unpairing
            //start conversion timer for pairing restrictions
            //go back to unpaired state

      //unpair cmd event
            //complete all actions related to unpairing
            //go back to unpaired state
      //Timeout event
            //inactivity timer
            //complete all action related to unpairing
            //go back to unpaired state
      }


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

//------------------------------INIT FUNCTIONS--------------------------//
static void InitUART2(void){
    //-------------------------CONFIG UART SETTINGS--------------------//
      // turn UART off
      // map TX to RA3
      // output
      // map RX to RA1
      // digital
      // input
      // clear the U1MODE register
      // 16x baud clock
      // 1 stop bit
      // 8 bit no parity
      // clear U1STA
      // interrupt when RX buffer not empty
      // enable TX pin
      // enable RX pin
      // 9600 baud
      // turn UART on

    //-------------------------CONFIG UART INTERRUPTS--------------------//
      // disable global interrupts
      // enable multivector mode
      // set priority of the interrupt
      // clear RX flag
      // clear TX flag
      // enable U2 RX interrupts
      // enable U2 TX interrupts
      // enable global interrupts
}

static void configTimer5(void){
    //--------------CONFIG TIMER5 SETTINGS---------------//
      //disable timer 5
      //disable gated time accumulation mode
      //select PBCLK as source
      //prescale of 64
      //Clear timer register
      //set timer period
```

```
    //-----------------CONFIG TIMER 5 INTERRUPTS----------------//
       // enable multivector mode
       //set Timer5Priority
       //clear T5 interrupt flag
       //enable T5 Interrupts
       //enable global interrupts
}

static void initBAYT(void){
    // Limit switch - RB10 - pin 9
       // turn on pull-down
       // turn off pull-up
       // input
}

//-------------------------------------CONSTRUCTING MSG
FUNCTIONS-------------------------//
static void constructStatusMsg(void){
    //Construct Status message
       //set Length byte
       //set destination XBee Address
       //set message ID byte
       //set Bite value byte to status
       //Set Pair status byte
       //set message length
       //Set checksum

    //setup to use TX interrupt
       //preload the U2TXREG
       //increment TX index
       // enable U2 TX interrupts
}

static void constructPairACKMsg(void){
    //Construct Pair ACK message
       //set Length byte
       //set destination XBee Address
       //set message ID byte
       //set boat Addr byte
       //set message length
       //Set checksum
    //setup to use TX interrupt
       //preload the U2TXREG
       //increment TX index
       // enable U2 TX interrupts
}


static void doUnpairActions(void){
       //Show unpaired indicator
       //stop motors
       //set last status to be sent flag
}

//-----------------------------------------------STORING AND PARSING
MESSAGES----------------------------//
static void storeData(uint8_t byte){
       //if data has started
              //store incoming bytes
```

```
            //increment message length
            //wait for msg length byte
            //at msg length byte, save it
            //beyond message length
                  //if in the relevant data
                        //increment packet data counter
                        //add to checksum
                  //if at the end of the data/checksum byte
                        //if valid checksum
                              //parse through data
                        //if invalid checksum
                              //do nothing because checksum is invalid
                        //end of message
                        //reset current msg Length
      //if data hasn't started
            //if got a start delimiter
            //set variables to track data
            //reset all static variables
            //init timer for when to reset MSG length

            //store incoming bytes
            //increment message length
 }

static void parseData(void){
      //disable U2 RX interrupts

      //loop through the data(except checksum) and parse
            //msg length byte
            //XBee API Identifier
                  //result from a transmit packet, dont care about it
            //Senders address MSB
                  //if paired
                        //check if correct BOSYN //replace 0x21 with
                  PairedBosynAddress_MSB
                   //received message from a random BOSYN, do nothing

                  //if not paired
                  //if just deflated
                  //if message is from previously paired BOSYN
                        //dont accept message



            //Senders address LSB
                  //if paired
                        //check if correct BOSYN //replace 0x21 with
                  PairedBosynAddress_MSB
                   //received message from a random BOSYN, do nothing

                  //if not paired
                  //if just deflated
                  //if message is from previously paired BOSYN
                        //dont accept message
            //protocol MSG ID
            //protocol data
                  //command protocol data
                        //must be paired to accept commands
                        //reset inactivity timer
                        //drive val 1
```

```
                        //grab motor cmd data
                        //drive val 2
                        //grab motor cmd data
                        //chomp Init byte
                              //tell ByterSM to bite
                        //AES init
                  //pairing protocol data
                        //must be unpaired to pair
                        //save new controller address
                        //tell XBee of pair request
                  //unpair protocol data
                        //must be paired to unpair
                  //if valid unpair command message
                        //send unpair cmd event
                  //invalid message ID for BOAT
      //re-enable U2 RX interrupts
}

//---------------------------------------ISR's-------------------------------//
void __ISR(_UART_2_VECTOR, IPL7SOFT) UARTHandler(void){
    static ES_Event_t NewEvent;

    //if TX flag is set and we enabled the TX interrupt and RX idle
      // clear TX flag
      //wait until the TX shift register is empty

      //if message is complete
            //turn off the transmit interrupt
            //reset TX index
            //load the U2TXREG
      //increment TX index
//if the RX flag is set
      // clear RX flag

       //setup new event
       //read the incoming message
      //post event to self
      // clear TX flag
      // clear RX flag
}

void __ISR(_TIMER_5_VECTOR,IPL6SOFT) StatusMsgLoop(void){
      //clear interrupt flag
      //setup new status event
      //post event to self

      //if last status
            //disable timer 5 to stop constant status messages
            //Clear timer register for good measure
            //reset last status flag
}

// BAYT EVENT CHECKER
bool check4BAYT(void){
   //if button changed and is low
         //Post deflated event
}
```