```
/*---------------------------- Include Files ----------------------------*/
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "ES_Timers.h"
#include "ES_Queue.h"
#include "CheckValidSM.h"
#include "ES_Port.h"
#include <stdio.h>
#include "TransmitXbeeSM.h"
#include "InterpretSM.h"

/*---------------------------- Module Defines ----------------------------*/
#define QUEUE_SIZE 6
#define MAX_DATA_LENGTH 15
#define MIN_DATA_LENGTH 6

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this machine.They should be functions
   relevant to the behavior of this state machine
*/
unsigned char *ReturnDataArray(void);
void InitializePins(void);

/*---------------------------- Module Variables ----------------------------*/
// everybody needs a state variable, you may need others as well.
// type of state variable should match that of enum in header file
static CheckValidState_t CurrentState;
static unsigned char DataLength = 0;
static unsigned char DataLengthTemp = 0;
static unsigned char ByteSum = 0;
static unsigned char DataArray[MAX_DATA_LENGTH + 1];
static unsigned char DataIndex = 0;

// everybody needs a local store for the service's priority too!
static uint8_t MyPriority;


/*---------------------------- Module Code ----------------------------*/
/****************************************************************************
 Function
     InitCheckValidSM
****************************************************************************/
boolean InitCheckValidSM ( uint8_t Priority )
{
  ES_Event ThisEvent;
  InitializePins();
  MyPriority = Priority;
  CurrentState = InitPState;
  ThisEvent.EventType = ES_INIT;
  if (ES_PostToService( MyPriority, ThisEvent) == True)
  {
      return True;
```

```
  }else
  {
      return False;
  }
}

/*****************************************************************************
 Function
     PostCheckValidSM
*****************************************************************************/
boolean PostCheckValidSM( ES_Event ThisEvent )
{
  return ES_PostToService( MyPriority, ThisEvent);

}

/*****************************************************************************
 Function
    RunCheckValidSM
*****************************************************************************/
ES_Event RunCheckValidSM( ES_Event CurrentEvent )
{
  ES_Event ReturnEvent;

  ReturnEvent.EventType = ES_NO_EVENT; // assume no problems will occur

  switch ( CurrentState )
  {
      case InitPState :       // If current state is initial Psedudo State
          if ( CurrentEvent.EventType == ES_INIT )// only respond to EF_Init
          {
              CurrentState = Check0x7E;
           }
           break;

      case Check0x7E :
          switch ( CurrentEvent.EventType )
          {
              case ES_XB_Msg_Rec :
              {
                    if(CurrentEvent.EventParam == 0x7E) CurrentState = Check0;
                    else{CurrentState = Check0x7E;}
               }
               break;       // end switch on CurrentEvent
          }
          break;            // end switch on CurrentState

      case Check0 :
          switch ( CurrentEvent.EventType )
          {
              case ES_XB_Msg_Rec :
              {
                    if(CurrentEvent.EventParam == 0) CurrentState = CheckDataLength;
```

```
                else {CurrentState = Check0x7E;}
            }
            break;

          case ES_TIMEOUT :
          {
              CurrentState = Check0x7E;
            }
            break;       // end switch on CurrentEvent
        }
        break;           // end switch on CurrentState

    case CheckDataLength :
        switch ( CurrentEvent.EventType )
        {
            case ES_XB_Msg_Rec :
            {
                if((CurrentEvent.EventParam >= MIN_DATA_LENGTH) && (CurrentEvent.EventParam <=
                MAX_DATA_LENGTH))
                  {
                      DataLength = (unsigned char)(CurrentEvent.EventParam);
                      DataLengthTemp = DataLength + 1;
                      CurrentState = ReceiveData;
                  }
                  else
                  {
                      CurrentState = Check0x7E;
                  }
            }
            break;

          case ES_TIMEOUT :
          {
              CurrentState = Check0x7E;
            }
            break;       // end switch on CurrentEvent
        }
        break;           // end switch on CurrentState

    case ReceiveData :
        switch ( CurrentEvent.EventType )
        {
            case ES_XB_Msg_Rec :
            {
                  DataLengthTemp -= 1;
                  ByteSum += (unsigned char)(CurrentEvent.EventParam);
                  if(DataLengthTemp == 0 && ByteSum == 0xFF)
                  {
                      ES_Event ThisEvent;
                      ThisEvent.EventType = ES_FRAME_REC;
                      PostInterpretSM(ThisEvent);
                      CurrentState = Check0x7E;
                      ByteSum = 0;
```

```
                    DataIndex = 0;
                }
                else if(DataLengthTemp == 0)
                {
                    CurrentState = Check0x7E;
                    ByteSum = 0;
                    DataIndex = 0;
                }
                else
                {
                    DataArray[DataIndex++] = (unsigned char)(CurrentEvent.EventParam);
                    CurrentState = ReceiveData;
                }
            }
            break;

          case ES_TIMEOUT :
          {
              CurrentState = Check0x7E;
              ByteSum = 0;
              DataIndex = 0;
            }
            break;   // end switch on CurrentEvent

        }
        break;        // end switch on Current State


  }
  return ReturnEvent;
}


/****************************************************************************
 Function
     QueryCheckValidSM
****************************************************************************/
CheckValidState_t QueryCheckValidSM ( void )
{
   return(CurrentState);
}

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

unsigned char *ReturnDataArray(void)
{
    return DataArray;
}
void InitializePins(void)
{
    ANSEL = 0;
```

```
    ANSELH = 0;
    TRISB5 = 1;             //RX pin set to input,
}
```