```
/*******************************************************************************
 Module
   Transmit_Service.c

 Revision
   1.0.1

 Description
   Contains All Functions to Assist with Transmitting Data via XBEE
*******************************************************************************/
/*----------------------------- 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 <stdio.h>
#include <string.h>
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "Transmit_Service.h"
#include "Outputs_Service.h"
#include "Definitions.h"
#include "Master_SM.h"
#include "Receive_SM.h"

/*----------------------------- Module Defines ----------------------------*/
#define ENCR_RAND_MAX 255;


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

void setCurrentTransmitPacket(uint8_t *packet, uint8_t length);

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

//Create Array with Prologues
uint8_t TX_Packet_Prologue_STAT[PROLOGUE_TRANSMIT_LENGTH] = {START_DELIMITER,
                                                             ZERO_BYTE,
                                                             STATUS_LENGTH,
                                                             API_IDENTIFIER_SEND,
                                                             ZERO_FRAME_ID,
                                                             BROADCAST_1,
                                                             BROADCAST_2,
                                                             OPTIONS_ACK};

static uint8_t CurrentTransmitPacketLength = PROLOGUE_TRANSMIT_LENGTH +
                                             STAT_LENGTH + CHECKSUM_LENGTH;
                                             //Initialize the length of the
                                             current transmit packet
static uint8_t StatTransmitPacket[PROLOGUE_TRANSMIT_LENGTH + STAT_LENGTH +
CHECKSUM_LENGTH] = {START_DELIMITER, ZERO_BYTE, STATUS_LENGTH,
API_IDENTIFIER_SEND, ZERO_FRAME_ID, BROADCAST_1, BROADCAST_2, OPTIONS_ACK,
STATUS_HDR, ZERO_BYTE, ZERO_BYTE, ZERO_BYTE};   //Set the Current Transmit Packet
Equal to the Maximum Value

static uint8_t CurrentTransmitPacketIndex;      //Initialize the current index of
the
transmit packet
```

```
static uint8_t CurrentTransmitPacketDestinationMSB;
static uint8_t CurrentTransmitPacketDestinationLSB;


/*------------------------------ Module Code ------------------------------*/
/***************************************************************************
 Function
     InitTransmit_Service

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
     Initializes the Transmit Service
***************************************************************************/
bool InitTransmit_Service ( uint8_t Priority )
{
    ES_Event ThisEvent;

    MyPriority = Priority;

    // post the initial transition event
    ThisEvent.EventType = ES_INIT;
    if (ES_PostToService( MyPriority, ThisEvent) == true)
    {
        return true;
    }
    else
    {
        return false;
    }
}

/***************************************************************************
 Function
     PostTransmit_Service

 Parameters
     EF_Event ThisEvent ,the event to post to the queue
***************************************************************************/
bool PostTransmit_Service( ES_Event ThisEvent )
{
    return ES_PostToService( MyPriority, ThisEvent);
}

/***************************************************************************
 Function
    RunTransmit_Service

 Parameters
   ES_Event : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description

***************************************************************************/
ES_Event RunTransmit_Service( ES_Event ThisEvent )
```

```
{
    ES_Event ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
    /*****************************************
     in here you write your service code
     *****************************************/
    return ReturnEvent;
}

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

void sendStatus(void)
{

    CurrentTransmitPacketIndex = 0;
    //Set the Destination Bits of the Prologue
    StatTransmitPacket[TX_INDEX_DESTINATION_MSB] = CurrentTransmitPacketDestinati
                                                   onMSB;
    StatTransmitPacket[TX_INDEX_DESTINATION_LSB] = CurrentTransmitPacketDestinati
                                                   onLSB;
    StatTransmitPacket[TX_INDEX_HEADER+1] = getCurrentStatus();
    StatTransmitPacket[TX_INDEX_HEADER+2] = getLastEncryptedCksum();

    //Send the First Byte
    sendNextByte();
}

//Set the destination of the transmit packet
void setTransmitPacketDestinationMSB(uint8_t MSB)
{
    CurrentTransmitPacketDestinationMSB = MSB;
}

void setTransmitPacketDestinationLSB(uint8_t LSB)
{
    CurrentTransmitPacketDestinationLSB = LSB;
}

//Send Next Byte to UART
bool sendNextByte(void)
{
    static uint8_t checkSum; //initialize static variable for checksum

    //Check if this is the last byte
    if (CurrentTransmitPacketIndex == (CurrentTransmitPacketLength - 1))
    {
        StatTransmitPacket[CurrentTransmitPacketIndex] = 0xff - checkSum;
    }

    //Write the Current Byte to the UART Regsiter
    HWREG(UART1_BASE + UART_O_DR) = StatTransmitPacket[
          CurrentTransmitPacketIndex];
    //printf("Tx: %x\n\r", StatTransmitPacket[CurrentTransmitPacketIndex]);

    //Calculate the CheckSum
    if (CurrentTransmitPacketIndex >= TX_INDEX_API)
    {
        checkSum += StatTransmitPacket[CurrentTransmitPacketIndex];
    }
```

```
    //Increment the index
    CurrentTransmitPacketIndex++;

    //If it was the first byte being transmitted enable TXIM
    if (CurrentTransmitPacketIndex == 1)
    {

        HWREG(UART1_BASE + UART_O_IM) |= (UART_IM_TXIM);
    }
    else if (CurrentTransmitPacketIndex == (CurrentTransmitPacketLength))
              //if last byte return true
    {

        checkSum = 0;//Reset the checksum
        return true;
    }

    //If Not Return False
    return false;
}


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