```
/*******************************************************************************
 Module
   SCI_Receive.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 "SCI_Receive.h"
#include "SCI_Send.h"
#include "ES_Timers.h"
#include "Coach.h"
#define SendInterval 3



/*---------------------------- 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 TXState_t CurrentState;
static ES_Event PostEvent;
static unsigned char Req_Data, Loc_Data, HDR, i, A, B, DiffSum, CheckSum, TotalL; // Flags to tell whether
finish assembling
static unsigned char TX_Message [30];


/*---------------------------- Module Code ----------------------------*/
/*******************************************************************************
 Function
     Init_SCI_Receive

 Parameters
     uint8_t : the priorty of this service

 Returns
     boolean, 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 Init_SCI_Send ( uint8_t Priority )
{
    ES_Event ThisEvent;

    MyPriority = Priority;


//Predefine the known data
    TX_Message[0] = 0x7E;
    TX_Message[1] = 0x00;
    TX_Message[3] = 0x01;
    TX_Message[4] = 0x01; //Depends on the protocol whether to use the frame ID
    TX_Message[7] = 0x00;

    B = 1;

    DiffSum = 0;

    CurrentState = Assembling;


    // post the initial transition event

    printf("SEND INIT");

    ThisEvent.EventType = ES_INIT;
    if (ES_PostToService( MyPriority, ThisEvent) == true)
        {
            return true;
        }
    else
        {
            return false;
        }


}

/****************************************************************************
 Function
      Post_SCI_Receive

 Parameters
      EF_Event ThisEvent ,the event to post to the queue

 Returns
      boolean 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 Post_SCI_Send( ES_Event ThisEvent )
{
    return ES_PostToService( MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    Run_SCI_Receive

 Parameters
   ES_Event : 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 Run_SCI_Send( ES_Event ThisEvent )
{
    ES_Event ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no errors


    switch ( CurrentState )
        {
        case Assembling:

            //printf("In the assembling state");


            if (ThisEvent.EventType == SendPairReq)
                {
                    Req_Data = ThisEvent.EventParam;
                    Send_PAIR_REQ (Req_Data);
                    printf("eventposted");
                    CalChecksum();
                }

            else if (ThisEvent.EventType == SendTAGOUT)
                {
                    Loc_Data = ThisEvent.EventParam;
                    Send_TAG_OUT (Loc_Data);
                    CalChecksum();
                }

            else if (ThisEvent.EventType == SendReset)
                {
                    Send_RESET ();
                    CalChecksum();
                }
```

```
            else if (ThisEvent.EventType == SendCtrl)
                {
                    Send_CTRL ();
                    CalChecksum();
                }

            break;



        case Sending :          //

            //printf("READYTOSEND");

            if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam == Send_TIMER))
// Only sending message out for once
                {

                    if ((SCI1SR1 & _S12_TDRE) == _S12_TDRE && B == TotalL)
                        {
                            SCI1DRL = TX_Message [B-1];
                            printf ("%x\n\r", TX_Message [B-1]);
                            printf ("---Sent---\n\r");
                            //SCheckSum = SCheckSum + TX_Message [A-1];      // Calcualte the
checksum
                            B = 1;
                            CurrentState = Assembling;
                        }


                    else if ((SCI1SR1 & _S12_TDRE) == _S12_TDRE )
                        {
                            SCI1DRL = TX_Message [B-1];                     // B is the index for
TX_Message
                            printf ("%x\n\r", TX_Message [B-1]);


                            B = B + 1;


                            ES_Timer_SetTimer (Send_TIMER, SendInterval);
                            ES_Timer_StartTimer(Send_TIMER);                    //Start timer for
the next byte to be sent out

                        }


                }



            break;

        }

    return ReturnEvent;
}
```

```
TXState_t Query_SendState ( void )
{
    return(CurrentState);
}


/*****************************************************************************
 private functions
 *****************************************************************************/
void CalChecksum(void)
{
    //printf("Comes into calculate checksum\n\r");
    for (i = 0; i < (TotalL - 4); i++)
        {
            A = i+3;
            DiffSum = DiffSum + TX_Message[A];
            CheckSum = 0xFF - DiffSum;
        }
    TX_Message[TotalL - 1] = CheckSum;          //Index need to be changed if the length of data changed due to the protocol
    DiffSum = 0;      // Reset all the values

    CurrentState = Sending;


    ES_Timer_SetTimer (Send_TIMER, SendInterval);
    ES_Timer_StartTimer(Send_TIMER);                        //Start timer for the next byte to be sent out

    Post_SCI_Send (PostEvent);
    printf("calledcalcheck");
}



void Send_PAIR_REQ (unsigned char req_Data)
{
    HDR = 0x03;   //HDR for PAIR_REQ

    TotalL = 11; // Based on 2 data bytes

    TX_Message[2] = 0x07; //2 data bytes
    TX_Message[5] = 0xFF;
    TX_Message[6] = 0xFF;

    // Broadcast PAIR_REQ
    TX_Message[8] = HDR;
    TX_Message[9] = req_Data;
    //printf("SentPair\n\r");
    return;

}



void Send_TAG_OUT (unsigned char loc_Data)
{
    HDR = 0x02;   //HDR for TAG_OUT
```

```
    TotalL = 11; // Based on 2 data bytes

    TX_Message[2] = 0x07; //2 data bytes
    TX_Message[5] = Query_AddMSB();
    TX_Message[6] = Query_AddLSB();

    TX_Message[8] = HDR;
    TX_Message[9] = loc_Data;
    printf("SentTagout");
    return;
}


void Send_RESET (void)
{
    HDR = 0x04;   //HDR for RESET

    TotalL = 11; // Based on 2 data bytes

    TX_Message[2] = 0x07; //2 data bytes
    TX_Message[5] = Query_AddMSB();
    TX_Message[6] = Query_AddLSB();

    TX_Message[8] = HDR;
    TX_Message[9] = 0x00;

    printf("ResetM\n\r");
    return;
}


// Need to be edit!!!
void Send_CTRL (void)
{
    HDR = 0x01;   //HDR for CTRL

    TotalL = 15; // Based on 6 data bytes

    TX_Message[2] = 0x0B; //6 data bytes
    TX_Message[5] = Query_AddMSB();
    TX_Message[6] = Query_AddLSB();

    //TX_Message[5] = 0x20;
    //TX_Message[6] = 0x8c;



    TX_Message[8] = HDR;
    TX_Message[9] = (unsigned char) Query_SPD();
    TX_Message[10] = (unsigned char) Query_DIR();
    TX_Message[11] = Query_AUX();
    TX_Message[12] = Query_WHIM();
    TX_Message[13] = Query_TRANS();

    printf("Control\n\r");

    return;

}
```

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