```c
/****************************************************************************
 ***************************************************************************
 Module
   Farmer_RX_SM.c

 Revision
   1.0.1

 Description
   The receiving state machine for the Farmer

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 05/13/17 5:29    mwm               created for the project
****************************************************************************/
/*----------------------------- 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 "FarmerRXSM.h"
#include "Constants.h"
#include "FarmerTXSM.h"
#include "FarmerMasterSM.h"
#include "EventCheckers.h"

#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_nvic.h"
#include "inc/hw_uart.h"
#include "inc/hw_sysctl.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"      // Define PART_TM4C123GH6PM in project
#include "driverlib/gpio.h"
#include "driverlib/uart.h"


/*----------------------------- Module Defines ----------------------------*/

/*---------------------------- Module Functions ---------------------------*/
/* prototypes for private functions for this machine.They should be functions
   relevant to the behavior of this state machine
*/
static void DataInterpreter( void );
static void ClearDataArray( void );
static void ClearDataBufferArray( void );
static void MoveDataFromBuffer( void );

/*---------------------------- 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 FarmerRX_State_t CurrentState;
static FarmerRX_State_t ISRState;

// with the introduction of Gen2, we need a module level Priority var as well
static uint8_t MyPriority, memCnt;
static uint8_t DogAddrMSB;
static uint8_t DogAddrLSB;
static bool paired;
static uint16_t BytesLeft,DataLength,TotalBytes;
static uint8_t Data[RX_DATA_LENGTH] = {0};
static uint8_t DataBuffer[RX_DATA_LENGTH] = {0};
static uint8_t CheckSum;


/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitFarmerRXSM

 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
     Matthew W Miller, 5/13/2017, 17:31
****************************************************************************/
bool InitFarmerRXSM ( uint8_t Priority )
{
  ES_Event ThisEvent;

  MyPriority = Priority;
  // put us into the first state
  CurrentState = Waiting2Rec;
      ISRState = WaitForFirstByte;
  // post the initial transition event
      //Set memCnt to 0
      memCnt = 0;
      //Set paired to false
      paired = false;
      //printf("BytesLeft at startup = %i\r\n", BytesLeft);
  if (ES_PostToService( MyPriority, ThisEvent) == true)
  {
      return true;
  }else
  {
      return false;
  }
}

/****************************************************************************
 Function
     PostFarmerRXSM

 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 PostFarmerRXSM( ES_Event ThisEvent )
{
  return ES_PostToService( MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    RunFarmerRXSM

 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
   uses nested switch/case to implement the machine.
 Author
   Matthew Miller, 05/13/17, 17:54
****************************************************************************/
ES_Event RunFarmerRXSM( ES_Event ThisEvent )
{
  ES_Event ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
      //printf("Farmer Receive CurrentState = %i\r\n",CurrentState);
      //printf("Data[0]: %i, Event: %i", Data[0], ThisEvent.EventType);
  switch ( CurrentState )
  {
            case Waiting2Rec :
                        //if ThisEvent EventType is ES_BYTE RECEIVED
                        if(ThisEvent.EventType == ES_BYTE_RECEIVED){
                              //Set CurrentState to Receive
                              CurrentState = Receive;
                        }
                  break;

            case Receive :

                  //Handle ES_TIMEOUTS
                  //if(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam
== BYTE_TIMER){
                  if(ThisEvent.EventType == ES_LOST_CONNECTION)
                  {


                              //Set CurrentState to Waiting2Rec
                              CurrentState = Waiting2Rec;
                              //Set memCnt to 0
                              memCnt = 0;
                              //Reset ISRState
                              ISRState = WaitForFirstByte;
                              //Clear Data Array
                              ClearDataArray();
                              //Clear Data Buffer
                              ClearDataBufferArray();
                  //}else
                  }
                  //if ThisEvent EventType is ES_MESSAGE_REC
                  else if(ThisEvent.EventType == ES_MESSAGE_REC){
                        //Call Data Interpreter -- Store all of the data for use by
FarmerMasterSM
```

```
                        DataInterpreter();
                        //ClearDataArray();

                        //Post ES_MESSAGE_REC to FarmerMasterSM
                        ES_Event NewEvent;
                        NewEvent.EventType = ES_MESSAGE_REC;
                        //printf("Posting Event to Master\r\n");
                        PostFarmerMasterSM(NewEvent);

                        //Set CurrentState to Waiting2Rec
                        CurrentState = Waiting2Rec;
                  }
                  break;
    default :
      ;
  }  // end switch on Current State
  return ReturnEvent;
}

/****************************************************************************
 Function
     QueryFarmerRXSM

 Parameters
     None

 Returns
     FarmerRX_State_t The current state of the Template state machine

 Description
     returns the current state of the Template state machine
 Notes

 Author
Matthew Miller, 5/13/17, 22:42
****************************************************************************/
FarmerRX_State_t QueryFarmerRXSM ( void )
{
   return(CurrentState);
}
/****************************************************************************
 Function
     FarmerRX_ISR

 Parameters
     None

 Returns
     The interrupt response for the UART receive

 Description
     stores the received byte into the data
 Notes

 Author
Matthew Miller, 5/13/17, 22:42
****************************************************************************/
void FarmerRX_ISR( void ){
      ES_Event ReturnEvent;
      //Set data to the current value on the data register
      if(memCnt > 42)
      {
            printf("FATAL ARRAY OVERFLOW ERROR: %i\r\n", memCnt);
      }
      DataBuffer[memCnt] = HWREG(UART1_BASE + UART_O_DR);

      //Check and handle receive errors
      if((HWREG(UART1_BASE + UART_O_RSR) & UART_RSR_OE) != 0){
            printf("Overrun Error :(\r\n");
      }
      if((HWREG(UART1_BASE + UART_O_RSR) & UART_RSR_BE) != 0){
            printf("Break Error :(\r\n");
      }
      if((HWREG(UART1_BASE + UART_O_RSR) & UART_RSR_FE) != 0){
            printf("Framing Error :(\r\n");
      }
      if((HWREG(UART1_BASE + UART_O_RSR) & UART_RSR_PE) != 0){
            printf("Parity Error :(\r\n");
      }
      HWREG(UART1_BASE + UART_O_ECR) |= UART_ECR_DATA_M;
  switch ( ISRState )
  {
       //Case WaitForFirstByte
            case WaitForFirstByte:
            if(DataBuffer[0] == INIT_BYTE)
            {
                  //Set ISRState to WaitForMSBLen
                  ISRState = WaitForMSBLen;
                  //Increment memCnt
                  memCnt++;

                  //Post ES_BYTE_RECEIVED event to  FarmerRXSM
                  ReturnEvent.EventType = ES_BYTE_RECEIVED;
                  PostFarmerRXSM(ReturnEvent);
            }
            break;

            //Case WaitForMSBLen
            case WaitForMSBLen :
                  //Set IsRState to WaitForLSBLen
                  ISRState = WaitForLSBLen;
                  //Increment memCnt
                  memCnt++;

            break;

            //Case WaitForLSBLen
            case WaitForLSBLen :
                  //Set ISRState to AcquireData
                  ISRState = AcquireData;

                  //Increment memCnt
                  memCnt++;
                  CheckSum = 0;

                  //Combine Data[1] and Data[2] into BytesLeft and DataLength
                  BytesLeft = DataBuffer[1];
                  BytesLeft = (BytesLeft << 8) + DataBuffer[2];
                  //printf("Bytes Left Initial value = %i\r\n", BytesLeft);
                  DataLength = BytesLeft;
                  TotalBytes = DataLength+NUM_XBEE_BYTES;

                  break;

            //Case AcquireData
            case AcquireData :
                  if(BytesLeft !=0)
  				{
  					//Increment memCnt
  					CheckSum += DataBuffer[memCnt];
  					memCnt++;

  					//Decrement BytesLeft
  					BytesLeft--;
  				}
  				else if(BytesLeft == 0)
  				{
  					CheckSum = 0xff - CheckSum;

  					//Set ISRState to WaitForFirstByte
  					ISRState = WaitForFirstByte;


  					//Post ES_MESSAGE_REC to FarmerRXSM
  					//If API is a receive, post a receive message
  					if(DataBuffer[3] == API_81 && (CheckSum ==
  DataBuffer[memCnt]))
  					{
  						ReturnEvent.EventType = ES_MESSAGE_REC;
  						PostFarmerRXSM(ReturnEvent);
  					}
  					else if(CheckSum != DataBuffer[memCnt])
  					{
  						SetBadCheckSum();
  					}

  					//Set memCnt to 0
  					memCnt = 0;

  					//Move and clear DataBuffer
  					MoveDataFromBuffer();
  					//ClearDataBufferArray();
  				}
  				break;

  			default:
  				break;

  		}
  }

  void RXTX_ISR( void ){
  	//get status of the receive and transmit interrupts
  	uint8_t RX_Int = HWREG(UART1_BASE + UART_O_MIS) & UART_MIS_RXMIS;
  	uint8_t TX_Int = HWREG(UART1_BASE + UART_O_MIS) & UART_MIS_TXMIS;

  	//If there was a receive interrupt
  	if(RX_Int != 0){
  		//Clear the source of the interrupt
  		HWREG(UART1_BASE + UART_O_ICR) |= UART_ICR_RXIC;
  		//Call the farmer receive interrupt response
  		FarmerRX_ISR();
  	}

  	//If there was a transmit interrupt
  	if(TX_Int != 0){
  		//Clear the source of the interrupt
  		HWREG(UART1_BASE + UART_O_ICR) |= UART_ICR_TXIC;
  		//Call the farmer transmit interrupt response
  		FarmerTX_ISR();
  	}
```

```
}

void setPair(void)
{
      paired = true;
}

void setUnpair(void)
{
      paired = false;
}

uint8_t getHeader(void)
{
      //Data Header byte corresponds to byte 8 in packet
      return Data[8];
}

uint8_t getAPI_ID(void)
{
      //Frame ID byte corresponds to byte 3 in packet
      return Data[3];
}

uint8_t getDogAddrMSB(void)
{
      //Sender MSB byte corresponds to byte 4 in packet
      return Data[4];
}

uint8_t getDogAddrLSB(void)
{
      //Sender LSB byte corresponds to byte 5 in packet
      return Data[5];
}

uint8_t getDataByte(uint8_t index)
{
      return Data[index];
}

uint8_t getGyroZ_MSB(void)
{
      return Data[19];
}

/***************************************************************************
 private functions
 ***************************************************************************/
static void DataInterpreter()
{
      //first check to see if API ID is 0x81
      //If it is, restart the communication timer

      /*
      for(int i = 0; i<TotalBytes;i++)
      {
            printf("RX %i: %04x\r\n",i,Data[i]);
      }
      */
}


static void ClearDataArray( void ){
      for(int i = 0; i<RX_DATA_LENGTH;i++){
            Data[i] = 0;
      }
}

static void ClearDataBufferArray( void ){
      for(int i = 0; i<RX_DATA_LENGTH;i++){
            DataBuffer[i] = 0;
      }
}

static void MoveDataFromBuffer( void ){
      for(int i = 0; i<RX_DATA_LENGTH;i++){
            Data[i] = DataBuffer[i];
      }
}
```