```
/*******************************************************************************
 Module
   RX_Service.c

 Revision
   1.0.1

 Description
   This implements parsing of data from audio

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 05/17/20       LG      instituted CheckSum and updated based on state machine
 05/16/20       LG      tried out a simpler version of receive (one sample based on timeout)
 05/06/20 08:22 hbf     began conversion for Lab 10
 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 "ES_Port.h"
#include "RX_Service.h"
#include "MicInputService.h"
#include "dbprintf.h"
#include "LED_Service.h"
#include "MicInputService.h"
#include "DataParserService.h"
#include "GameplayFSM.h"
/*---------------------------- Module Defines ----------------------------*/
#define BIT_TIME 10
#define HALF_BIT_TIME 5
#define NUM_DATA_DIBITS 10
#define FULL_SLOT_TIME 130
/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
static void GetNextBit(void);
/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;
static uint8_t CurrentState;
static bool Starting = false;
static uint8_t RX_Counter;
static uint32_t RX_Data;
static uint32_t RX_ShiftReg;
static uint32_t Sum;

/*---------------------------- Module Code ----------------------------*/
/*******************************************************************************
 Function
    InitRXService

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
```

```
   Saves away the priority, and does any
   other required initialization for this service
 Notes

 Author
   H. Francis, 05/08/20, 10:00
****************************************************************************/
bool InitRXService(uint8_t Priority)
{
 ES_Event_t ThisEvent;

 MyPriority = Priority;
 CurrentState = RX_Idle;

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

/****************************************************************************
 Function
   PostRXService

 Parameters
   EF_Event_t ThisEvent ,the event to post to the queue

 Returns
   bool false if the Enqueue operation failed, true otherwise

 Description
   Posts an event to this state machine's queue
 Notes

 Author
   H. Francis, 05/08/20, 10:00
****************************************************************************/
bool PostRXService(ES_Event_t ThisEvent)
{
 return ES_PostToService(MyPriority, ThisEvent);
}

/****************************************************************************
 Function
   RunRXService

 Parameters
  ES_Event_t : the event to process

 Returns
  ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
  Handles state machine for this service. Handles receiving a series of bits

 Notes

 Author
  H. Francis, 05/08/20, 10:15
****************************************************************************/
ES_Event_t RunRXService(ES_Event_t ThisEvent)
{
```

```
ES_Event_t ReturnEvent;
ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
ES_Event_t PostEvent;

switch (CurrentState)
{
  case RX_Idle:
  {
    if(ThisEvent.EventType == ES_START_LISTENING)
    {
      EnableListen();
      CurrentState = RX_Listening;
    }
    if(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam == RX_TIMER) {
      EnableListen();
      CurrentState = RX_Listening;
    }
    if(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam == SERVO_RELEASE_TIMER) {
      PostEvent.EventType = ES_START_LISTENING;
      PostRXService(PostEvent);
    }
  }
  break;
  case RX_Listening:
  {
    if(ThisEvent.EventType == ES_START_BIT_DETECTED)
    {
      Starting = true; // not used in current code version for same reason as ^
      RX_Counter = 0;
      RX_ShiftReg = 0;
      CurrentState = RX_Receiving;
    }
    if(ThisEvent.EventType == ES_STOP_LISTENING)
    {
      DisableListen();
      CurrentState = RX_Idle;
    }
  }
  break;
  case RX_Receiving:
  {
    if(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam == RX_TIMER)
    {
      if(Starting) { // for same reason as earlier comments, this is technically going unexecuted
        Starting = false;
        if(QueryBitState() == DIBIT_3) //good start bit
        {
          RX_ShiftReg = 0;
          ES_Timer_InitTimer(RX_TIMER, BIT_TIME);
          ES_Timer_InitTimer(TDMA_CLOCK, FULL_SLOT_TIME-HALF_BIT_TIME);
        } else { //bad start bit
          CurrentState = RX_Listening;
          ClearFirstTime();
        }
      } else { //past start bit
       GetNextBit();
      }
    }
    if(ThisEvent.EventType == ES_EOR)
    {
      CurrentState = RX_Idle;
      RX_ShiftReg = 0;
      Sum = 0;
      ClearFirstTime();
    }

    if(ThisEvent.EventType == ES_BAD_RECEPTION)
    {
```

```
            CurrentState = RX_Listening;
            RX_ShiftReg = 0;
            Sum = 0;
            ClearFirstTime();
          }
        }
        break;

    }
    return ReturnEvent;
}
/*****************************************************************************
 Function
   GetNextBit

 Parameters
   none

 Returns
   none

 Description
   helper function that handles collection of bits, validates the CheckSum, and updates the RX_Data variable

 Notes

 Author
   L. Gardner, 05/16/20
*****************************************************************************/
static void GetNextBit(void)
{
   ES_Event_t PostEvent;
   uint8_t BitState = QueryBitState();

   if(RX_Counter < NUM_DATA_DIBITS)
   {
      ES_Timer_InitTimer(RX_TIMER, BIT_TIME);
      RX_Counter++;
      if (BitState == DIBIT_0)
      {
         RX_ShiftReg &= (BIT0LO & BIT1LO);
         RX_ShiftReg = RX_ShiftReg << 2;
         Sum += 0;
      }
      else if (BitState == DIBIT_1)
      {
         //printf("Detected Data bit..... 1\n\r");
         RX_ShiftReg |= BIT0HI;
         RX_ShiftReg = RX_ShiftReg << 2;
         Sum += 1;
      }
      else if (BitState == DIBIT_2)
      {
         //printf("Detected Data bit..... 2\n\r");
         RX_ShiftReg |= BIT1HI;
         RX_ShiftReg = RX_ShiftReg << 2;
         Sum += 2;
      }
      else if (BitState == DIBIT_3)
      {
         //printf("Detected Data bit..... 3\n\r");
         RX_ShiftReg |= (BIT1HI | BIT0HI);
         RX_ShiftReg = RX_ShiftReg << 2;
         Sum += 3;
      }
   } else {
      uint8_t CheckSum = 0;
      Sum &= (BIT1HI | BIT0HI);
```

```
      CheckSum = 3-Sum;
      if(BitState != CheckSum) { //framing error
         printf("BAD DATA\n\r");
      } else {
         RX_Data = RX_ShiftReg;
         ConvertFromPacket(RX_Data>>2);
         printf("GOOD DATA \n\r");
      }
         PostEvent.EventType = ES_EOR;
         PostRXService(PostEvent);
         PostGameplayFSM(PostEvent);
         ES_Timer_InitTimer(RX_TIMER, HALF_BIT_TIME);
         DisableListen();
   }
}
/*****************************************************************************
 Function
   GetRX_Data

 Parameters
  none

 Returns
 RX_Data

 Description
 getter function for the DataParser to call

 Notes

 Author
  L. Gardner, 05/16/20
*****************************************************************************/
uint32_t GetRX_Data(void) {
   return RX_Data;
}
/*****************************************************************************
 Function
   GetRXState

 Parameters
  none

 Returns
   RX state

 Description
   getter function for RXState

 Notes

 Author
  L. Gardner, 05/16/20
*****************************************************************************/
RX_State_t GetRXState(void){
   return CurrentState;
}

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