```
/*******************************************************************************
 Module
   DataParserService.c

 Revision
   1.0.1

 Description
   This Service converts from commands to data packets and back

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 05/15/20 12:45 hbf     began conversion for 218C project
 05/05/20 13:LED_DISPLAY_TIME 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 "DataParserService.h"
#include "TX_Service.h"
#include "GameplayFSM.h"
#include "dbprintf.h"

/*---------------------------- Module Defines ----------------------------*/
#define LOWER_NYBBLE 15
#define GET_HI_NYBBLE 4
#define OMNI_RANGE 0
#define DIRECT_RANGE 1
#define DETONATE_RANGE 2

#define OMNI_RANGE_DISTANCE 3
#define DIRECT_RANGE_DISTANCE 6
#define LED_DISPLAY_TIME 200
/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
static void ParseData(void);
static bool checkRange(uint8_t rangeType);
/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t Address;
static uint8_t Source_sub;
static uint8_t Source_station;
static uint8_t Act_Head;
static uint8_t Data;
static uint8_t NUM_TEAMS = 2;
static uint8_t OUR_TEAM_NUM = 1;
static uint8_t StrikeLocation = 0;
static bool NoEchoSent = true;

static struct GameStatus gamestatus;

const static char LEDS[] = "LEDs #lit \r\n";
const static char WE_DEAD[] = "WE DEAD \r\n";
const static char SENT_ECHO[] = "SENT_ECHO \r\n";

/*---------------------------- Module Code ----------------------------*/
/*******************************************************************************
 Function
    setNumTeams

 Parameters
    uint8_t numteams

 Returns
   none

 Description
    sets the number of teams in the game

 Notes
```

```
 Author
   H. Francis, 05/28/20, 12:30
*****************************************************************************/
void setNumTeams(uint8_t numteams) {
   NUM_TEAMS = numteams;
}

/*****************************************************************************
 Function
   setTeamNum

 Parameters
   uint8_t teamnum

 Returns
  none

 Description
   sets our team number in the game

 Notes

 Author
   H. Francis, 05/28/20, 12:35
*****************************************************************************/
void setTeamNum(uint8_t teamnum) {
   OUR_TEAM_NUM = teamnum;
}

/*****************************************************************************
 Function
   InitializeGameStatus

 Parameters
   none

 Returns
  none

 Description
   sets all values in the struct to track GameStatus when the game begins

 Notes

 Author
   H. Francis, 05/19/20, 12:35
*****************************************************************************/
void InitializeGameStatus(void){
   gamestatus.ConnActive = true;       //Things we set
   gamestatus.SonarActive = true;
   gamestatus.TorpedoActive = true;

   gamestatus.OurTeamNum = OUR_TEAM_NUM;
   gamestatus.NumSlots = NUM_TEAMS*3;      //Derived from NumTeams
   gamestatus.NumTeams = NUM_TEAMS;

   printf("Team num: %d, num teams: %d\n\r", gamestatus.OurTeamNum, gamestatus.NumTeams);

   gamestatus.WeOmniPinged = false;
   gamestatus.WeDirectPinged = false;
   gamestatus.WeLaunched = false;
   gamestatus.DistanceToDetonate = 1;
   gamestatus.NextConnMove = NO_MOVE;

   printf("Init X: %d, Init Y: %d, Init heading: %d\n\r", gamestatus.OurPositionX, gamestatus.OurPositionY, gamestatus.OurHeading);

   if(gamestatus.ConnActive) {
   gamestatus.ExpectedNextSlotNumber = (gamestatus.OurTeamNum *3 -1);
   }
   else if (gamestatus.SonarActive) {
      gamestatus.ExpectedNextSlotNumber = ((gamestatus.OurTeamNum *3));
   }
   else {
      gamestatus.ExpectedNextSlotNumber = ((gamestatus.OurTeamNum *3)+1);
   }
   gamestatus.Turn = 0;
   gamestatus.WeAreAlive = true;

   gamestatus.TeamPosUpdate = 0;
   gamestatus.OtherTeamPositionX[0] = 0;
   gamestatus.OtherTeamPositionY[0] = 0;
   gamestatus.OtherTeamPositionX[1] = 0;
   gamestatus.OtherTeamPositionY[1] = 0;
   gamestatus.OtherTeamPositionX[2] = 0;
```

```
    gamestatus.OtherTeamPositionY[2] = 0;
    gamestatus.OtherTeamPositionX[3] = 0;
    gamestatus.OtherTeamPositionY[3] = 0;
    gamestatus.OtherTeamPositionX[4] = 0;
    gamestatus.OtherTeamPositionY[4] = 0;
    gamestatus.OtherTeamPositionX[5] = 0;
    gamestatus.OtherTeamPositionY[5] = 0;
    gamestatus.OtherTeamPositionX[6] = 0;
    gamestatus.OtherTeamPositionY[6] = 0;
    gamestatus.OtherTeamPositionX[7] = 0;
    gamestatus.OtherTeamPositionY[7] = 0;

    gamestatus.OtherStationOnlineStatus = 255;
    uint8_t i = 0;
    while (i < NUM_TEAMS) {
        gamestatus.TheyAreAlive = gamestatus.TheyAreAlive <<1;
        if(i != gamestatus.OurTeamNum) {
        gamestatus.TheyAreAlive |= BIT0HI;
        }
        else {
            gamestatus.TheyAreAlive &=BIT0LO;
        }
        ++i;
    }
    gamestatus.Launched = 0;
    gamestatus.NumMoveOnsInARow = 0;
    gamestatus.JustMoveOnAlready = false;
    gamestatus.onlyHeardNextTurns = false;

}

/*****************************************************************************
 Function
   resetGameStatus

 Parameters
   none

 Returns
  none

 Description
   resets all values in the struct to track GameStatus when we change turns

 Notes

 Author
   H. Francis, 05/22/20, 08:32
*****************************************************************************/
void resetGameStatus(void) {
    gamestatus.NumMoveOnsInARow = 0;

    gamestatus.WeDirectPinged = false;
    gamestatus.WeOmniPinged = false;
    NoEchoSent = true;

    gamestatus.WeLaunched = false;
    gamestatus.DistanceToDetonate = 1;

    gamestatus.TeamPosUpdate = 0;
    gamestatus.Launched = 0;
    gamestatus.OtherTeamPositionX[0] = 0;
    gamestatus.OtherTeamPositionY[0] = 0;
    gamestatus.OtherTeamPositionX[1] = 0;
    gamestatus.OtherTeamPositionY[1] = 0;
    gamestatus.OtherTeamPositionX[2] = 0;
    gamestatus.OtherTeamPositionY[2] = 0;
    gamestatus.OtherTeamPositionX[3] = 0;
    gamestatus.OtherTeamPositionY[3] = 0;
    gamestatus.OtherTeamPositionX[4] = 0;
    gamestatus.OtherTeamPositionY[4] = 0;
    gamestatus.OtherTeamPositionX[5] = 0;
    gamestatus.OtherTeamPositionY[5] = 0;
    gamestatus.OtherTeamPositionX[6] = 0;
    gamestatus.OtherTeamPositionY[6] = 0;
    gamestatus.OtherTeamPositionX[7] = 0;
    gamestatus.OtherTeamPositionY[7] = 0;

    gamestatus.JustMoveOnAlready = false;
}

/*****************************************************************************
 Function
   getPointerToGameStatus
```

```
 Parameters
   none

 Returns
  pointer to the struct GameStatus

 Description
   Allows other functions to edit GameStatus

 Notes

 Author
   H. Francis, 05/19/20, 09:23
*****************************************************************************/
struct GameStatus* getPointerToGameStatus(void) {
   return(&gamestatus);
}

/*****************************************************************************
 Function
   SetStrikeLocation

 Parameters
   uint8_t StrikeLoc

 Returns
  none

 Description
   sets the location to detonate the torpedo

 Notes

 Author
   H. Francis, 05/28/20, 12:30
*****************************************************************************/
void SetStrikeLocation(uint8_t StrikeLoc) {
   StrikeLocation = StrikeLoc;
}

/*****************************************************************************
 Function
   ConvertToPacket

 Parameters
  none

 Returns
  none

 Description


 Notes

 Author
   H. Francis, 05/15/20, 12:58
*****************************************************************************/
uint32_t ConvertToPacket(uint8_t address, uint8_t source, uint8_t act_head)
{
   uint32_t packet = ((gamestatus.Turn) % 3) & 3;
   packet = packet << 1; //Create space for address
   packet |= address;
   packet = packet << 3; //Create space for Team
   packet |= gamestatus.OurTeamNum;
   packet = packet << 2; //Create space for source
   packet |= source;
   packet = packet << 4; //Create space for Act_Head
   packet |= act_head;
   if(act_head == PING_DIRECTED || act_head == POS_UPDATE) {
      packet &= BIT0LO;
      packet |= gamestatus.OurHeading;
   }
   packet = packet << 8; //Create space for Data
   if(act_head == PING_OMNI || act_head == POS_UPDATE
         || act_head == PING_DIRECTED || act_head == ECHO
         || act_head == HIT || act_head == LAUNCHED)
   {
      if(act_head == ECHO) {
         LATB |= BIT5HI;
         printf("%s", LEDS);
         ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
      }
```

```
      if(act_head == HIT) {
        LATB |= BIT3HI;
        printf("%s", LEDS);
        ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
      }
      packet |= (gamestatus.OurPositionY << 4);
      packet |= (gamestatus.OurPositionX);
    } else if (act_head == NEXT_TURN) {
      packet |= (gamestatus.Turn + 1);
    } else if (act_head == DETONATE) {
      packet |= StrikeLocation;
      LATB |= BIT4HI;
      printf("%s", LEDS);
      ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
    }

    return (packet << 12);
}

/****************************************************************************
 Function
   ClearActiveLED

 Parameters
  none

 Returns
  none

 Description
  Turn RX/TX Active LED off

 Notes

 Author
  H. Francis, 05/09/20, 14:22
****************************************************************************/
void ConvertFromPacket(uint32_t packet) {
    Data = packet & 0xFF;
    packet = packet >> 8;
    Act_Head = packet & 0x0F;
    packet = packet >> 4;
    Source_station = packet & 3;
    packet = packet >> 2;
    Source_sub = packet & 7;
    packet = packet >> 3;

    ParseData();
}
/****************************************************************************
 Function
   ParseData

 Parameters
  none

 Returns
  none

 Description
  This function takes the received data and decides how to respond

 Notes

 Author
  H. Francis, 05/15/20, 14:22
****************************************************************************/
static void ParseData(void)
{
    uint8_t slotNumber = (Source_sub + Source_sub + Source_sub + Source_station);
    gamestatus.ExpectedNextSlotNumber = (slotNumber + 1);
    if(gamestatus.WeAreAlive) {
      if(Source_sub == gamestatus.OurTeamNum) { //our teams transmission
        printf("US: ");
        if(Address == ENEMY) {
          if(Act_Head == NEXT_TURN || Act_Head == RIP || Act_Head == SYNC) {
            gamestatus.NumMoveOnsInARow++;
            printf("NEXT_TURN/RIP/SYNC");
          } else if(Act_Head == HIT) {
            printf("%s", WE_DEAD);
            LATB |= BIT3HI;
            ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
            printf("%s", LEDS);
            gamestatus.WeAreAlive = false;
```

```
            gamestatus.NumMoveOnsInARow = 0;
            gamestatus.onlyHeardNextTurns = false;

            printGameStatus(true);
        }
        if (Act_Head == NO_ACTION) printf("NO_ACTION");
    } else { //Address == OWN
        printf("TO SELF, ");
        if(Act_Head == NEXT_TURN || Act_Head == RIP) {
            gamestatus.NumMoveOnsInARow++;
            printf("NEXT_TURN/RIP");
        } else {
            if((Act_Head > ECHO) && (Act_Head < PING_OMNI) ) { //not ping omni or echo
                printf("POS_UPDATE");
                gamestatus.OurHeading = (Act_Head & 7); //mask last 3 bits
                gamestatus.OurPositionX = Data & LOWER_NYBBLE;
                gamestatus.OurPositionY = Data >> GET_HI_NYBBLE;
                printGameStatus(true);
            } else if (Act_Head == HIT) {
                printf("%s", WE_DEAD);
                LATB |= BIT3HI;
                printf("%s", LEDS);
                ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
                gamestatus.WeAreAlive = false;
                printGameStatus(true);
            } else if (Act_Head == POS_REQUEST && gamestatus.ConnActive) {
                printf("POS_REQUEST");
                uint32_t NextPacket = ConvertToPacket(OWN, CONN, POS_UPDATE);
                TransmitEnqueue(CONN, NextPacket);
            }
            gamestatus.NumMoveOnsInARow = 0;
            gamestatus.onlyHeardNextTurns = false;
        }
    }
} else { //not our teams transmission
    printf("ENEMY: ");
    if(Address == ENEMY) {
        if(Act_Head == NEXT_TURN || Act_Head == RIP || Act_Head == SYNC) {
            printf("NEXT_TURN/RIP/SYNC");
            gamestatus.NumMoveOnsInARow++;
        } else {
            if(Act_Head == PING_OMNI && checkRange(OMNI_RANGE)) {
                printf("PING_OMNI    ");
                if(gamestatus.ConnActive && NoEchoSent) {
                    printf("%s", SENT_ECHO);
                    uint32_t NextPacket = ConvertToPacket(ENEMY, CONN, ECHO);
                    TransmitEnqueue(CONN, NextPacket);
                    NoEchoSent = false;
                } else {
                    printf("NO ECHO");
                }
            } else if ((Act_Head > ECHO && Act_Head < PING_OMNI) && checkRange(DIRECT_RANGE)) {
                printf("PING_DIRECTED");
                if(gamestatus.ConnActive && NoEchoSent) {
                    printf("%s", SENT_ECHO);
                    uint32_t NextPacket = ConvertToPacket(ENEMY, CONN, ECHO);
                    TransmitEnqueue(CONN, NextPacket);
                    NoEchoSent = false;
                }  else {
                    printf("NO ECHO");
                }
            } else if (Act_Head == ECHO) {
                printf("RECEIVED ECHO    ");
                if((gamestatus.WeDirectPinged && checkRange(DIRECT_RANGE)) ||
                  (gamestatus.WeOmniPinged && checkRange(OMNI_RANGE))) {
                    gamestatus.TeamPosUpdate |= (1 << Source_sub);
                    gamestatus.OtherTeamPositionX[Source_sub] = Data & LOWER_NYBBLE;
                    gamestatus.OtherTeamPositionY[Source_sub] = Data >> GET_HI_NYBBLE;
                    printf("UPDATE ENEMY  %d: %d, %d", Source_sub, gamestatus.OtherTeamPositionX[Source_sub], gamestatus.OtherTeamPositionY[Source_sub]);
                    printGameStatus(true);
                }
            } else if (Act_Head == HIT) {
                LATB |= BIT6HI;
                ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
                uint8_t mask = 1 << Source_sub;
                gamestatus.TheyAreAlive &= (~mask);
                if (gamestatus.TorpedoActive) {
                    printf("----------Team %d SUNK-------", Source_sub);
                }
            } else if ((Act_Head == DETONATE) && checkRange(DETONATE_RANGE)) {
                printf("DETONATED, WE DIED");
                LATB |= BIT3HI;
                ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
                printf("%s", LEDS);
```

```
                gamestatus.WeAreAlive = false;
                if(gamestatus.ConnActive) {
                    uint32_t NextPacket = ConvertToPacket(ENEMY, CONN, HIT);
                    TransmitEnqueue(CONN, NextPacket);
                }
                printGameStatus(true);
            } else if (Act_Head == LAUNCHED) {
                printf("LAUNCHED");
                gamestatus.TeamPosUpdate |= (1 << Source_sub);
                gamestatus.OtherTeamPositionX[Source_sub] = Data & LOWER_NYBBLE;
                gamestatus.OtherTeamPositionY[Source_sub] = Data >> GET_HI_NYBBLE;
                uint8_t mask = 1 << Source_sub;
                gamestatus.Launched |= mask;
            }
            gamestatus.NumMoveOnsInARow = 0;
            gamestatus.onlyHeardNextTurns = false;
        }
        if (Act_Head == NO_ACTION) printf("NO_ACTION");
    } else { //Address == OWN
        printf("TO OWN, ");
        if(Act_Head == NEXT_TURN || Act_Head == RIP) {
            gamestatus.NumMoveOnsInARow++;
        } else {
            if (Act_Head == HIT) {
                printf("-----SOMEONE SUNK-----");
                LATB |= BIT6HI;
                ES_Timer_InitTimer(LED_TIMER, LED_DISPLAY_TIME);
                uint8_t mask = 1 << Source_sub;
                gamestatus.TheyAreAlive &= (~mask);
            }
            gamestatus.NumMoveOnsInARow = 0;
            gamestatus.onlyHeardNextTurns = false;

        }
    }
  }
  printf("\r\n");
  uint32_t mask = 1 << slotNumber;
  gamestatus.OtherStationOnlineStatus &= mask;
 }
}

/*****************************************************************************
 Function
   printGameStatus

 Parameters
  bool Full

 Returns
  none

 Description
  This function prints the GameStatus in a way that the GUI can understand

 Notes

 Author
  H. Francis, 05/15/20, 14:22
*****************************************************************************/
void printGameStatus(bool Full)
{
  printf("GSC%d %d %d", gamestatus.WeAreAlive, gamestatus.Turn, gamestatus.TeamPosUpdate);
  for(int i = 0; i < MAX_TEAMS; i++) {
      printf(" %d %d", gamestatus.OtherTeamPositionX[i], gamestatus.OtherTeamPositionY[i]);
      gamestatus.OtherTeamPositionX[i] = 0;
      gamestatus.OtherTeamPositionY[i] = 0;
  }
  printf(" %d %d %d", gamestatus.TheyAreAlive, gamestatus.Launched, gamestatus.WeLaunched);
  printf(" %d %d %d ", gamestatus.OurHeading, gamestatus.OurPositionX, gamestatus.OurPositionY);
  printf("\n\r");
  if(Full) return; //suppress compiler warning
}

/*****************************************************************************
 Function
   checkRange

 Parameters
  uint8_t rangeType

 Returns
  bool, true if in range, false if otherwise
```

```
 Description
   This function determines whether or not we are in range of a ping or detonation

 Notes

 Author
   H. Francis, 05/15/20, 14:22
*******************************************************************************/
static bool checkRange(uint8_t rangeType) {
    uint8_t enemyX = Data & LOWER_NYBBLE;
    uint8_t enemyY = Data >> GET_HI_NYBBLE;
    uint8_t distX;
    uint8_t distY;
    uint8_t enemyZ = 21 - (enemyX + enemyY);
    uint8_t distZ;

    if(enemyX > gamestatus.OurPositionX) {
        distX = enemyX - gamestatus.OurPositionX;
    } else {
        distX = gamestatus.OurPositionX - enemyX;
    }
    if(enemyY > gamestatus.OurPositionY) {
        distY = enemyY - gamestatus.OurPositionY;
    } else {
        distY = gamestatus.OurPositionY - enemyY;
    }
    uint8_t OurPositionZ = 21 - (gamestatus.OurPositionX + gamestatus.OurPositionY);

    if(enemyZ > OurPositionZ) {
        distZ = enemyZ - OurPositionZ;
    } else {
        distZ = OurPositionZ - enemyZ;
    }

    if(rangeType == OMNI_RANGE) {
        if((distX + distY + distZ) / 2 <= OMNI_RANGE_DISTANCE) {
            if (gamestatus.SonarActive) {
                //Update Position
                gamestatus.TeamPosUpdate |= (1 << Source_sub);
                gamestatus.OtherTeamPositionX[Source_sub] = Data & LOWER_NYBBLE;
                gamestatus.OtherTeamPositionY[Source_sub] = Data >> GET_HI_NYBBLE;
                printGameStatus(true);
            }
            return true;
        }
        return false;
    } else if (rangeType == DETONATE_RANGE) {
        if(distX == 0 && distY == 0) return true;
        return false;
    } else if (rangeType == DIRECT_RANGE) {
        uint8_t enemyH = Act_Head & 7;
        if(enemyH == UP || enemyH == DOWN) {
            if (!(distX == 0 && distY <= DIRECT_RANGE_DISTANCE)) return false; // Don't do anything (but we'll return later)
        } else if (enemyH == RIGHT_UP || enemyH == LEFT_DOWN) {
            if (!(distY == 0 && distX <= DIRECT_RANGE_DISTANCE)) return false;
        } else if (enemyH == LEFT_UP || enemyH == RIGHT_DOWN) {
            if (!(distX == distY && distY <= DIRECT_RANGE_DISTANCE)) return false;
        }
        if (gamestatus.SonarActive) {
            //Update Position
            gamestatus.TeamPosUpdate |= (1 << Source_sub);
            gamestatus.OtherTeamPositionX[Source_sub] = Data & LOWER_NYBBLE;
            gamestatus.OtherTeamPositionY[Source_sub] = Data >> GET_HI_NYBBLE;
            printGameStatus(false);
        }
        return true;
    }
    return false; //default
}

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