```c
#include "defs.h"
#include <string.h>
#include "game.h"
#include "LCDG.h"
#include "switch.h"
#include "Timer.h"
#include "Music.h"
#include "xbee.h"

#define VERTICAL 0
#define HORIZONTAL 1

typedef struct {
  unsigned char x;
  unsigned char y;
  unsigned char orientation;
  unsigned char size;
  unsigned char hits;
} ShipType;

typedef struct {
  unsigned int type:1;
  unsigned int x:4;
  unsigned int y:4;
} AttackType;

CursorType cursor;

static int state;

static int buttonFlag;

static char string[10];

static ShipType ships[5] = {
  {0, 0, VERTICAL, 2, 0},
  {0, 0, VERTICAL, 3, 0},
  {0, 0, VERTICAL, 3, 0},
  {0, 0, VERTICAL, 4, 0},
  {0, 0, VERTICAL, 5, 0}
};

static ShipType computerShips[5] = {
  {0, 0, VERTICAL, 2, 0},
  {0, 0, VERTICAL, 3, 0},
  {0, 0, VERTICAL, 3, 0},
  {0, 0, VERTICAL, 4, 0},
  {0, 0, VERTICAL, 5, 0}
};

static unsigned char field[10][10];

static int numShips;

static AttackType enemyAttacks[100];
static int numEnemyAttacks;

static AttackType playerAttacks[100];
static int numPlayerAttacks;

int findValidPos(ShipType * array, int index);

int checkDead(ShipType * array) {
  int i;
  for(i=0; i<5; i++) {
    if(array[i].size != array[i].hits) {
      return 0;
    }
  }
  return 1;
}

void incState(void) {
  switch(state) {
    case WELCOME:
      state = WAITING_FOR_OPPONENT;
      break;
    case WAITING_FOR_OPPONENT:
      numShips = 1;
      state = PLACING_SHIPS;
      LCD_Clear(0);
      break;
    case PLACING_SHIPS:
      cursor.x = 0;
      cursor.y = 0;
      state = PLAYER_TURN_WAITING;
      LCD_Clear(0);
      strcpy(string, " ATTACK!!!");
      break;
    case PLAYER_TURN_DONE:
      if(checkDead(computerShips)) {
        state = WIN;
      }
      else {
        state = OPPONENT_TURN_WAITING;
        strcpy(string, " Opponent");
      }
      break;
    case COMPUTER_SCREEN:
      state = PLAYER_TURN_WAITING;
      break;
    case OPPONENT_TURN_WAITING:
      state = OPPONENT_TURN_DONE;
      break;
    case OPPONENT_TURN_DONE:
      if(checkDead(ships)) {
        state = LOSE;
      }
      else {
        state = PLAYER_TURN_WAITING;
        strcpy(string, " ATTACK!!!");
      }
      break;
  }
  Game_Update();
}

unsigned char random(unsigned char max) {
  unsigned static char seed1 = 0;
  unsigned static char seed2;
  unsigned static short last = 0;

  unsigned short tcnt = TCNT;
  seed1 = (tcnt&0xFF00) >> 8;
  seed2 = (tcnt&0x00FF);

  last = ((unsigned short) seed1)*last + seed2;

  return (unsigned char) (last%max);
}

int shipInBounds(ShipType * array, int index) {
  ShipType * ship = &array[index];

  if(ship->x < 0 || ship->x > 9 || ship->y < 0 || ship->y > 9 ||
    (ship->orientation == VERTICAL && ship->x + ship->size > 10) ||
    (ship->orientation == HORIZONTAL && ship->y + ship->size > 10)) {
      return 0;
  }

  return 1;
}

int checkHit(ShipType * array, int x, int y) {
  int i, j;
  for(i=0; i<5; i++) {
    for(j=0; j<array[i].size; j++) {
      if(array[i].orientation == HORIZONTAL) {
        if(x == array[i].x && y == array[i].y + j) {
          return i;
        }
      }
      else if(x == array[i].x + j && y == array[i].y) {
        return i;
      }
    }
  }

  return -1;
}

int validShipPos(ShipType * array, int index) {
  ShipType ship = array[index];
  int i;

  for(i=0; i<index; i++) {
    if(ship.orientation == HORIZONTAL) {
      if(array[i].orientation == HORIZONTAL) {
        if(ship.x == array[i].x) {
          if((ship.y + ship.size > array[i].y &&
              ship.y + ship.size <= array[i].y + array[i].size) ||
              (ship.y >= array[i].y &&
              ship.y < array[i].y + array[i].size) ||
              (array[i].y + array[i].size > ship.y &&
              array[i].y + array[i].size <= ship.y + ship.size) ||
              (array[i].y >= ship.y &&
              array[i].y < ship.y + ship.size)) {

            return 0;
          }
        }
      }
      else {
        if(ship.x >= array[i].x &&
           ship.x < array[i].x + array[i].size &&
           array[i].y >= ship.y &&
           array[i].y < ship.y + ship.size) {

          return 0;
        }
      }
    }
    else {
      if(array[i].orientation == HORIZONTAL) {
        if(ship.y >= array[i].y &&
           ship.y < array[i].y + array[i].size &&
           array[i].x >= ship.x &&
           array[i].x < ship.x + ship.size) {

          return 0;
        }
      }
      else {
        if(ship.y == array[i].y) {
          if((ship.x + ship.size > array[i].x &&
              ship.x + ship.size <= array[i].x + array[i].size) ||
              (ship.x >= array[i].x &&
              ship.x < array[i].x + array[i].size) ||
              (array[i].x + array[i].size > ship.x &&
              array[i].x + array[i].size <= ship.x + ship.size) ||
              (array[i].x >= ship.x &&
              array[i].x < ship.x + ship.size)) {

            return 0;
          }
        }
      }
    }
  }

  return 1;
}

void createField(ShipType * shipArray, int shipSize, AttackType * attackArray, int attackSize)         ¬
{
  int i, j;

    for(i=0; i<10; i++) {
      for(j=0; j<10; j++) {
        field[i][j] = EMPTY;
      }
    }

    for(i=0; i<shipSize; i++) {
```

```c
      ShipType ship = shipArray[i];
      if(ship.orientation == HORIZONTAL) {
        field[ship.x][ship.y] = SHIPEND_LEFT;
        for(j=1; j<ship.size-1; j++) {
          field[ship.x][ship.y+j] = SHIP_HORIZ;
        }
        field[ship.x][ship.y+ship.size-1] = SHIPEND_RIGHT;
      }
      else {
        field[ship.x][ship.y] = SHIPEND_UP;
        for(j=1; j<ship.size-1; j++) {
          field[ship.x+j][ship.y] = SHIP_VERT;
        }
        field[ship.x+ship.size-1][ship.y] = SHIPEND_DOWN;
      }
    }

    for(i=0; i<attackSize; i++) {
      AttackType attack = attackArray[i];
      field[attack.x][attack.y] = attack.type;
    }
}

void enemyInit(void) {
  int i;

  for (i=0; i<5; i++) {
    ShipType * ship = &computerShips[i];
    ship->x = random(10);
    ship->y = random(10);
    ship->orientation = random(2);

    findValidPos(computerShips, i);
  }
}

void enemyPickMove(void) {
  int i, x, y, moveFlag, hit;

  do {
    moveFlag = 0;
    x = random(10);
    y = random(10);

    for(i=0; i<numEnemyAttacks; i++) {
      if(enemyAttacks[i].x == x && enemyAttacks[i].y == y) {
        moveFlag = 1;
      }
    }
  }while(moveFlag);

  enemyAttacks[numEnemyAttacks].x = x;
  enemyAttacks[numEnemyAttacks].y = y;
  hit = checkHit(ships, x, y);
  if(hit == -1) {
    enemyAttacks[numEnemyAttacks++].type = MISS;
    strcpy(string, "    Miss ");
  }
  else {
    enemyAttacks[numEnemyAttacks++].type = HIT;
    strcpy(string, "    Hit ");
    Music_EnableOC7(EXPLODE);
    asm cli
  }
}

void Game_Init(void) {
  state = WELCOME;
  numShips = 0;
  numEnemyAttacks = 0;
  numPlayerAttacks = 0;
  cursor.x = 0;
  cursor.y = 0;
  LED_DDR0 = 1;
  LED_DDR1 = 1;
  LED_DDR2 = 1;
  LED_DDR3 = 1;
  LED_DDR4 = 1;
  LED_DDR5 = 1;

  LED0 = 1;
  LED1 = 1;
  LED2 = 1;
  LED3 = 1;
  LED4 = 1;
  LED5 = 1;
  Game_Update();
}

void Game_Update(void) {
  int frameFlag = 1;
  switch(state) {
    case WELCOME:
      LCD_Clear(0);
      LCD_GoTo(4, 1);
      LCD_OutString("       Welcome       ");
      LCD_GoTo(5, 1);
      LCD_OutString("    to Battleship    ");
      Timer_Wait10ms(100);
      incState();
      break;
    case WAITING_FOR_OPPONENT:
      enemyInit();
      incState();
      break;
    case PLACING_SHIPS:
      //LCD_Clear(0);
      LCD_GoTo(3,1);
      LCD_OutString("   Place  ");
      LCD_GoTo(4,1);
      LCD_OutString("   your  ");
      LCD_GoTo(5,1);
      LCD_OutString("   ships  ");
      createField(ships, numShips, enemyAttacks, 0);
      LCD_DrawGrid(field);
      break;
    case PLAYER_TURN_WAITING:
      //LCD_Clear(0);
      LCD_GoTo(4,1);
      LCD_OutString(string);
      createField(ships, 0, playerAttacks, numPlayerAttacks);
      LCD_DrawGrid(field);
      break;
    case PLAYER_TURN_DONE:
      //LCD_Clear(0);
      LCD_GoTo(4,1);
      LCD_OutString(string);
      createField(ships, 0, playerAttacks, numPlayerAttacks);
      LCD_DrawGrid(field);
      break;
    case OPPONENT_TURN_WAITING:
      //LCD_Clear(0);
      createField(ships, numShips, enemyAttacks, numEnemyAttacks);
      LCD_GoTo(4,1);
      LCD_OutString(string);
      LCD_DrawGrid(field);
      Music_EnableOC7(WHISTLE);
      asm cli
      Timer_Wait10ms(102);
      enemyPickMove();
      incState();
      break;
    case OPPONENT_TURN_DONE:
      //LCD_Clear(0);
      createField(ships, numShips, enemyAttacks, numEnemyAttacks);
      LCD_GoTo(4,1);
      LCD_OutString(string);
      LCD_DrawGrid(field);
      Timer_Wait10ms(140);
      incState();
      break;
    case COMPUTER_SCREEN:
      //LCD_Clear(0);
      createField(computerShips, 5, playerAttacks, numPlayerAttacks);
      LCD_DrawGrid(field);
      break;
    case WIN:
      //LCD_Clear(0);
      LCD_GoTo(4, 1);
      LCD_OutString("       You Win       ");
      break;
    case LOSE:
      //LCD_Clear(0);
      LCD_GoTo(4, 1);
      LCD_OutString("       You Lose      ");
      break;
  }
}

int findValidPos(ShipType * array, int index) {
  if(validShipPos(array, index) && shipInBounds(array, index)) {
    return 1;
  }
  else {
    ShipType * ship = &array[index];
    unsigned int tempX   = (ship->x + 9)%10;
    unsigned int tempY   = (ship->y + 9)%10;
    unsigned int tempDir = ship->orientation ^ 1;

    for(ship->orientation = tempDir ^ 1; ship->orientation != tempDir; ship->orientation = (++ship->orientation)%2) {
      for(ship->x = (tempX+1)%10; ship->x != tempX; ship->x = (++ship->x)%10) {
        for(ship->y = (tempY+1)%10; ship->y != tempY; ship->y = (++ship->y)%10) {
          if(validShipPos(array, index) && shipInBounds(array, index)) {
            return 1;
          }
        }
      }
    }

    ship->x = (tempX+1)&0x0F;
    ship->y = (tempY+1)&0x0F;
    ship->orientation = (tempDir+1)&0x01;

    return 0;
  }
}

void flag(void) {
  buttonFlag = 0;
}

void Game_DPad(unsigned char direction) {
  unsigned int tempX, tempY;
  if(!buttonFlag) {
    switch(state) {
      case PLACING_SHIPS:
        tempX = ships[numShips-1].x;
        tempY = ships[numShips-1].y;

        do {
          switch(direction) {
            case UP:
              ships[numShips-1].x--;
              break;
            case DOWN:
              ships[numShips-1].x++;
              break;
            case LEFT:
              ships[numShips-1].y--;
              break;
            case RIGHT:
              ships[numShips-1].y++;
              break;
          }
        }while(!validShipPos(ships, numShips-1) && shipInBounds(ships, numShips-1));

        if(validShipPos(ships, numShips-1) && shipInBounds(ships, numShips-1)) {
          Game_Update();
        }
        else {
          ships[numShips-1].x = tempX&0x0F;
          ships[numShips-1].y = tempY&0x0F;
        }
```

```c
          break;
        case PLAYER_TURN_WAITING:
          switch(direction) {
            case UP:
              cursor.x = (cursor.x+9)%10;
              break;
            case DOWN:
              cursor.x = (cursor.x+1)%10;
              break;
            case LEFT:
              cursor.y = (cursor.y+9)%10;
              break;
            case RIGHT:
              cursor.y = (cursor.y+1)%10;
              break;
          }
          Game_Update();
          break;
        case COMPUTER_SCREEN:
          incState();
          break;
      }

      buttonFlag = 1;
      enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
    }
  }

void LEDflash(void) {
  int i;
  for(i=0; i<10; i++) {
    LED0 ^= 1;
    LED1 ^= 1;
    LED2 ^= 1;
    LED3 ^= 1;
    LED4 ^= 1;
    LED5 ^= 1;
    Timer_Wait1ms(100);
  }
}

void Game_A(void) {
  int i, attackFlag;
  if(!buttonFlag) {
    switch(state) {
      case PLACING_SHIPS:
        if(findValidPos(ships, numShips)) {
          numShips++;
        }

        if(numShips == 6) {
          numShips--;
          incState();
        }
        else {
          Game_Update();
        }
        break;
      case PLAYER_TURN_WAITING:
        attackFlag = 0;
        for(i=0; i<numPlayerAttacks; i++) {
          if(playerAttacks[i].x == cursor.x && playerAttacks[i].y == cursor.y) {
            attackFlag = 1;
          }
        }
        if(!attackFlag) {
          int hit = checkHit(computerShips, cursor.x, cursor.y);
          playerAttacks[numPlayerAttacks].x = cursor.x;
          playerAttacks[numPlayerAttacks].y = cursor.y;
          if(hit == -1) {
            playerAttacks[numPlayerAttacks++].type = MISS;
            state = PLAYER_TURN_DONE;
            Music_EnableOC7(WHISTLE);
            asm cli
            Timer_Wait10ms(102);
            strcpy(string, "   MISS!   ");
            Game_Update();
            Timer_Wait10ms(100);
          }
          else {
            computerShips[hit].hits++;
            playerAttacks[numPlayerAttacks++].type = HIT;
            state = PLAYER_TURN_DONE;
            Music_EnableOC7(WHISTLE);
            asm cli
            Timer_Wait10ms(102);
            strcpy(string, "   HIT!   ");
            Game_Update();
            Music_EnableOC7(EXPLODE);
            asm cli
            LEDflash();
          }
          incState();
        }
        break;
      case COMPUTER_SCREEN:
        incState();
        break;
    }

    buttonFlag = 1;
    enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
  }
}

void Game_B(void) {
  if(!buttonFlag) {
    switch(state) {
      case PLACING_SHIPS:
        ships[numShips-1].orientation ^= 1;
        if(validShipPos(ships, numShips-1) && shipInBounds(ships, numShips-1)) {
          Game_Update();
        }
        else {
          ships[numShips-1].orientation ^= 1;
        }
        break;
      case PLAYER_TURN_WAITING:
        state = COMPUTER_SCREEN;
        Game_Update();
        break;
      case COMPUTER_SCREEN:
        incState();
        break;

    }

    buttonFlag = 1;
    enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
  }
}

CursorType Game_GetCursor(void) {
  return cursor;
}

int Game_GetState(void) {
  return state;
}
```