```c
#include <mc9s12dp512.h>     /* derivative information */
#include "game.h"
#include "LCDG.h"
#include "switch.h"

#define DEBOUNCE_DELAY 30000

#define SINGLE_PLAYER 0
#define MULTI_PLAYER 1

#define VERTICAL 0
#define HORIZONTAL 1

typedef struct {
  unsigned int x:4;
  unsigned int y:4;
  unsigned int orientation:1;
  unsigned int size:3;
} ShipType;

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

struct {
  unsigned int x:4;
  unsigned int y:4;
} cursor;

static int state;

static int buttonFlag;

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

static int numShips;

static AttackType enemyAttacks[100];
static int numEnemyAttacks;

static AttackType playerAttacks[100];
static int numPlayerAttacks;

void incState(void) {
  switch(state) {
    case WELCOME:
      numShips = 1;
      state = PLACING_SHIPS;
      break;
  }
  Game_Update();
}

void Game_Init(void) {
  state = WELCOME;
  numShips = 0;
  numEnemyAttacks = 0;
  numPlayerAttacks = 0;
  cursor.x = 0;
  cursor.y =0;
  Game_Update();
}

void Game_Update(void) {
  int i, j;

  if(state == WELCOME) {

    LCD_Clear(0);
    LCD_GoTo(4, 1);
    LCD_OutString("Welcome to Battleship");

    enableOC6(&incState, 62500, 9, 1);
  }
  else if (state == PLACING_SHIPS) {
    static unsigned  char field[10][10];
    LCD_Clear(0);

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

    for(i=0; i<numShips; i++) {
      ShipType ship = ships[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<numEnemyAttacks; i++) {
      AttackType attack = enemyAttacks[i];
      field[attack.x][attack.y] = attack.type;
    }

    LCD_DrawGrid(field);
  }

  /*

    else {
      for(i=0; i<numPlayerAttacks; i++) {
        AttackType attack = playerAttacks[i];
        field[attack.x][attack.y] = attack.type;
      }
    }
  */
}

int shipInBounds(int index) {
  ShipType ship = ships[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 validShipPos(int index) {
  ShipType ship = ships[index];
  int i;

  for(i=0; i<numShips; i++) {
    if(i != index) {
      if(ship.orientation == HORIZONTAL) {
        if(ships[i].orientation == HORIZONTAL) {
          if(ship.x == ships[i].x) {
            if(ship.y + ship.size > ships[i].y ||
               ship.y < ships[i].y + ships[i].size) {
              return 0;
            }
          }
        }
        else {
          if(ship.x >= ships[i].x &&
             ship.x < ships[i].x + ships[i].size &&
             ships[i].y >= ship.y &&
             ships[i].y < ship.y + ship.size) {
            return 0;
          }
        }
      }
      else {
        if(ships[i].orientation == HORIZONTAL) {
          if(ship.y >= ships[i].y &&
             ship.y < ships[i].y + ships[i].size &&
             ships[i].x >= ship.x &&
             ships[i].x < ship.x + ship.size) {
            return 0;
          }
        }
        else {
          if(ship.y == ships[i].y) {
            if(ship.x + ship.size > ships[i].x ||
               ship.x < ships[i].x + ships[i].size) {
              return 0;
            }
          }
        }
      }
    }
  }

  return 1;
}

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(numShips-1) && shipInBounds(numShips-1));

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

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

void Game_A(void) {
  if(!buttonFlag) {
    switch(state) {
      case PLACING_SHIPS:
      numShips++;
      Game_Update();
      buttonFlag = 1;
      enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
      break;
    }
  }
}

void Game_B(void) {
  if(!buttonFlag) {
    switch(state) {
      case PLACING_SHIPS:
        ships[numShips-1].orientation ^= 1;
        if(validShipPos(numShips-1) && shipInBounds(numShips-1)) {
          Game_Update();
        }
        else {
          ships[numShips-1].orientation ^= 1;
        }
        buttonFlag = 1;
        enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
        break;
    }
  }
}
```