```
/*
 * timer0.h
 *
 * Created: 2018-04-20 16:01:21
 *  Author: ine14gba
 */


#ifndef TIMER0_H_
#define TIMER0_H_

#include <avr/io.h>

#include <avr/interrupt.h>

#define TEST         0

#define BTN_LEFT_MASK                     0b00000100                    //
Dessa riktningar Σr tΣnka frσn spelare grΦns perspektiv.
#define BTN_RIGHT_MASK                    0b00000010
#define BTN_UP_MASK                       0b00001000
#define   BTN_DOWN_MASK                   0b00010000
#define BTN_SELECT_MASK                   0b00000001


//*GAMMALGAMMAL*/
/*
#define BTN_DOWN_MASK                     0b00000100                    //
Dessa riktningar Σr tΣnka frσn spelare grΦns perspektiv.
#define BTN_UP_MASK                       0b00000010
#define BTN_LEFT_MASK                     0b00001000
#define BTN_RIGHT_MASK                    0b00010000
#define BTN_SELECT_MASK                   0b00000001
*/
void restart_game(void);
void initialize_screens(void);
void check_busy_green(void);
void check_busy_yellow(void);
void send_command_lcd_green1(unsigned char command);
void send_command_lcd_green2(unsigned char command);
void send_command_lcd_yellow1(unsigned char command);
void send_command_lcd_yellow2(unsigned char command);
void send_command_all_lcd(unsigned char command);
void send_character_green1(unsigned char command);
void send_character_green2(unsigned char command);
void send_character_yellow1(unsigned char command);
void send_character_yellow2(unsigned char command);
void send_character_all_lcd(unsigned char command);
void clear_lcd_green(void);
void clear_lcd_yellow(void);
void clear_all_lcd(void);
void clear_tile_green1(int x, int y);
void clear_tile_green2(int x, int y);
void clear_tile_yellow1(int x, int y);
void clear_tile_yellow2(int x, int y);
void fill_tile_green1(int x, int y);
void fill_tile_green2(int x, int y);
void fill_tile_yellow1(int x, int y);
```

```
void fill_tile_yellow2(int x, int y);
void create_grid_green1();
void create_grid_green2();
void create_grid_yellow1();
void create_grid_yellow2();
void create_grid_tile_green1(int x, int y);
void create_grid_tile_green2(int x, int y);
void create_grid_tile_yellow1(int x, int y);
void create_grid_tile_yellow2(int x, int y);
void clear_cursor_green1(int x, int y);
void draw_cursor_green1(int x, int y);
void draw_cursor_yellow1(int x, int y);
void clear_cursor_yellow1(int x, int y);
void clear_cursor_green2(int x,int y);
void draw_cursor_green2(int x,int y);
void clear_cursor_yellow2(int x,int y);
void draw_cursor_yellow2(int x,int y);
void draw_line_green1(int x, int y);
void draw_line_yellow1(int x, int y);
void draw_cross_green1(int x, int y);
void draw_cross_yellow1(int x, int y);
void draw_line_green2(int x, int y);
void draw_line_yellow2(int x, int y);
void draw_cross_green2(int x, int y);
void draw_cross_yellow2(int x, int y);
void read_button_green();
void read_button_yellow();
void create_matrices();
void place_ships_green();
void place_ships_yellow();
void place_aircraft_carrier_green();
void place_carrier_green();
void place_cruiser_green();
void place_destroyer_green();
void place_submarine_green();
void place_aircraft_carrier_yellow();
void place_carrier_yellow();
void place_cruiser_yellow();
void place_destroyer_yellow();
void place_submarine_yellow();
int victory(uint8_t matrix[10][10]);
void fill_ship_green();
void fill_ship_yellow();
void clear_ship_green();
void clear_ship_yellow();
int get_btn_up();
int get_btn_up();
int get_btn_left();
int get_btn_right();
int get_btn_down();
int get_btn_select();
int get_overflow_blink();
int blink_ships();
void blink_ship_green();
void blink_ship_yellow();
void select_player();
```

```
extern volatile uint8_t button_state, btn_up, btn_down, btn_left, btn_right,
btn_select, previous_state, green_turn, cursorW, cursorH, toggle_blink,
double_click_counter, placing_ships, ship_length;
volatile uint8_t GreenEnemy[10][10]; // Gr÷ns h÷gra skrΣm dΣr guls skepp ska
beskjutas
volatile uint8_t GreenShips[10][10];
volatile uint8_t YellowEnemy[10][10];
volatile uint8_t YellowShips[10][10];

void init_timer0();

void enable_timer0_interrupt();

void disable_timer_interrupt();


#endif /* TIMER0_H_ */


/*
 * timer0.c
 *
 * Created: 2018-04-20 16:01:00
 *  Author: ine14gba
 */

#include "timer0.h"


volatile uint8_t button_state, btn_up, btn_down, btn_left, btn_right,
btn_select, previous_state, green_turn, cursorW, cursorH, toggle_blink,
double_click_counter, placing_ships, ship_length, overflow_count,
overflow_blink;
volatile uint8_t GreenEnemy[10][10];  //Matris f÷r gr÷ns h÷gra skΣrm
volatile uint8_t GreenShips[10][10];  //Matris f÷r gr÷ns vΣnstra skΣrm
volatile uint8_t YellowEnemy[10][10];  //Matris f÷r guls h÷gra skΣrm
volatile uint8_t YellowShips[10][10];  //Matris f÷r gr÷ns vΣnstra skΣrm

void init_timer0() {

          TCCR0 = (1<<CS00) | (0<<CS01) | (1<<CS02);

}

void enable_timer0_interrupt() {

          TIMSK |= (1<<TOIE0);

}

void disable_timer0_interrupt() {

          TIMSK &= ~(1<<TOIE0);

}

ISR(TIMER0_OVF_vect) {
          PORTC |= 0b10000000;
```

```
		overflow_count++; //timer f÷r att lΣsa av knapptryck
		overflow_blink++; //timer f÷r att initiera blinkningar

		if(double_click_counter != 0) {
		double_click_counter--;
		}

		if (overflow_count == 2) { // lΣser av knapptrycken var 96:e ms nΣr
				button_state  = (~PINB) & 0b00011111;
				if(previous_state == 1 && button_state != 0) {
		//metod f÷r att debounca knappar. previous_state = 1 om knappen var
nedtryckt vid f÷rra avlΣsningstillfΣllet
					button_state = 0;
				} else if(previous_state == 1) {
					previous_state = 0;
				}
				if(green_turn == 1) { // sΣtter rΣtt mask till rΣtt spelare
				btn_up = BTN_UP_MASK & button_state;
				btn_down = BTN_DOWN_MASK & button_state;
				btn_right = BTN_RIGHT_MASK & button_state;
				btn_left = BTN_LEFT_MASK & button_state;
				btn_select = BTN_SELECT_MASK & button_state;
				}
				else if(green_turn == 0) { //upp-och-ner-vΣnd skΣrm,
inverterar knapparna f÷r gul
					btn_up = BTN_DOWN_MASK & button_state;
					btn_down = BTN_UP_MASK & button_state;
					btn_right = BTN_LEFT_MASK & button_state;
					btn_left = BTN_RIGHT_MASK & button_state;
					btn_select = BTN_SELECT_MASK & button_state;
				}
				overflow_count = 0;
				if(button_state != 0) {
					previous_state = 1;
				}

				}

		if(placing_ships != 0 && overflow_blink%5 == 0 && green_turn == 1) {
				//blinkmetod f÷r lΣget dσ gr÷n spelare placerar ut sina
skepp
				if(toggle_blink == 0) {
					fill_ship_green();
					toggle_blink = 1;
				}
				else{
					clear_ship_green();
					toggle_blink = 0;

				}
		}
		if(placing_ships != 0 && overflow_blink%5 == 0 && green_turn == 0) {
				//blinkmetod f÷r lΣget dσ gul spelare placerar ut sina
skepp
				if(toggle_blink == 0) {
					fill_ship_yellow();
					toggle_blink = 1;
				}
```

```
                    else {
                              toggle_blink = 0;
                              clear_ship_yellow();
                    }
          }


          if(placing_ships == 0 && overflow_blink%5==0 && green_turn == 1) {
                    if(toggle_blink == 0) {
                              fill_tile_green1(cursorW,cursorH);
                              toggle_blink = 1;
                              } else {
                              clear_cursor_green1(cursorW,cursorH);
                              toggle_blink = 0;
                    }
          }else if(placing_ships == 0 && overflow_blink%5==0 && green_turn ==
0) {
                    if(toggle_blink == 0) {
                              fill_tile_yellow1(cursorW,cursorH);
                              toggle_blink = 1;
                              } else {
                              clear_cursor_yellow1(cursorW,cursorH);
                              toggle_blink = 0;
          }
          }

}

int get_btn_up() {              //Kollar om uppσtknappen Σr nedtryckt
          if(btn_up != 0) {
                    btn_up = 0;
                    return 1;
          }
          return 0;
}
int get_btn_left() {
          if(btn_left != 0) {
                    btn_left = 0;
                    return 1;
          }
          return 0;
}
int get_btn_right() {
          if(btn_right != 0) {
                    btn_right = 0;
                    return 1;
          }
          return 0;
}
int get_btn_down() {
          if(btn_down != 0) {
                    btn_down = 0;
                    return 1;
          }
          return 0;
}

int get_btn_select() {
```

```
		if(btn_select != 0) {
				btn_select = 0;
				return 1;
		}
		return 0;
}

int get_overflow_blink() {
				return overflow_blink;
}



/*
 * GccApplication1.c
 *
 * Created: 2018-04-17 13:42:15
 * Author : ine14gba
 */

#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <time.h>
#include <stdlib.h>
#include "timer0.h"
#include <avr/wdt.h>
#include <stdio.h>


#define setHigh(port,bit) (port) |= (1<<(bit))		//Metod f÷r att
sΣtta till high och low, orkade inte T—NKA
#define setLow(port,bit) (port) &= ~(1<<(bit))

#define DATABUS					PORTA
#define DATABUS_DIR			DDRA

#define INTERFACEBUS			PORTB
#define INTERFACEBUS_DIR		DDRB
#define LED_RED					7
#define LED_GREEN				6
#define LED_YELLOW				5

#define ENABLEBUS				PORTC
#define ENABLEBUS_DIR			DDRC
#define ENABLE_YELLOW			1
#define ENABLE_GREEN			0		// Statisk h÷g,
togglar dσ h÷g -> lσg -> h÷g.

#define CONTROLBUS				PORTD
#define CONTROLBUS_DIR			DDRD
#define CS2GREEN				7		// Aktiv lσg.
#define CS1GREEN				6
#define CS2YELLOW				5
#define CS1YELLOW				4
#define DI						2		// Data = h÷g,
instruction = lσg.
```

```
#define RW                                  1                   // Read = h÷g,
write = lσg.
#define RES                                             0                   //
Aktiv lσg.
#define DELAY                                           200



// Metoder.



int ship_direction; //LodrΣt = 0




int main(void) {
          wdt_disable();
          // Avaktiverar timer till reset_game();
          initialize_screens();
          INTERFACEBUS_DIR = 0b11100000;            // SΣtter LED's till outputs
och knappar till inputs.
          clear_all_lcd();
          create_grid_green2();
          create_grid_yellow2();
          init_timer0();
          //SΣtter pσ timern
          enable_timer0_interrupt();
          sei();
          create_matrices();

          while(1) {
          place_ships_green();
          place_ships_yellow();
          read_button_green();
          }
}

void restart_game(void) { //startar om spelet
          wdt_enable(0);
          while (1) {
          }
}
void initialize_screens() {
          ENABLEBUS_DIR = 0xff;
          CONTROLBUS_DIR = 0xff;
          DATABUS_DIR = 0xff;
          setHigh(CONTROLBUS, RES);
          setLow(CONTROLBUS, RW);
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, CS1GREEN);
          setHigh(CONTROLBUS, CS2GREEN);
          setHigh(CONTROLBUS, CS1YELLOW);
          setHigh(CONTROLBUS, CS2YELLOW);
```

```
          setHigh(ENABLEBUS, ENABLE_GREEN);
          setHigh(ENABLEBUS, ENABLE_YELLOW);                              // Fram
hit Σr det en rΣtt rimlig setup f÷r att b÷rja skicka commands.

          send_command_all_lcd(0x3f);
          // SΣtt pσ screen
          send_command_all_lcd(0x01);
          // Clear screen
}
void send_command_green1(unsigned char command) {             //f÷rbereder
displayen f÷r att ta emot information
          check_busy_green();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(150);
          setLow(CONTROLBUS, CS1GREEN);
          setLow(CONTROLBUS, RW);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_GREEN);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_GREEN);
          setHigh(CONTROLBUS, CS1GREEN);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_GREEN);

          DATABUS = 0;
}
void send_command_green2(unsigned char command) {
          check_busy_green();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(150);
          setLow(CONTROLBUS, CS2GREEN);
          setLow(CONTROLBUS, RW);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_GREEN);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_GREEN);
          setHigh(CONTROLBUS, CS2GREEN);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_GREEN);

          DATABUS = 0;
}
void send_command_yellow1(unsigned char command) {
          check_busy_yellow();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
```

```
          _delay_us(150);
          setLow(CONTROLBUS, CS1YELLOW);
          setLow(CONTROLBUS, RW);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_YELLOW);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_YELLOW);
          setHigh(CONTROLBUS, CS1YELLOW);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_YELLOW);

          DATABUS = 0;
}
void send_command_yellow2(unsigned char command) {
          //f÷rbereder displayen f÷r att ta emot information
          check_busy_yellow();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(150);
          setLow(CONTROLBUS, CS2YELLOW);
          setLow(CONTROLBUS, RW);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_YELLOW);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_YELLOW);
          setHigh(CONTROLBUS, CS2YELLOW);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_YELLOW);

          DATABUS = 0;

}
void check_busy_green() {                                   //Anv΢nds f÷r att
kolla om LCDn ΢r busy (samma kod f÷r gul och gr÷n)
          DATABUS_DIR = 0x00;
          setHigh(ENABLEBUS, ENABLE_GREEN);
          setLow(CONTROLBUS, RW);
          setLow(CONTROLBUS, DI);

          setLow(ENABLEBUS, ENABLE_GREEN);
          _delay_us(100);
          setHigh(CONTROLBUS, RW);
          setLow(CONTROLBUS, CS1GREEN);
          setLow(CONTROLBUS, CS2GREEN);
          setHigh(CONTROLBUS, DI);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_GREEN);

          while(DATABUS >= 0b10000000) {
                    // V΢nta tills ej busy.
          }
          setLow(CONTROLBUS, RW);
```

```
          setHigh(CONTROLBUS, CS1GREEN);
          setHigh(CONTROLBUS, CS2GREEN);
          DATABUS_DIR = 0xff;
}
void check_busy_yellow() {

          DATABUS_DIR = 0x00;

          setHigh(ENABLEBUS, ENABLE_YELLOW);
          setLow(CONTROLBUS, RW);
          setLow(CONTROLBUS, DI);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(100);
          setHigh(CONTROLBUS, RW);
          setLow(CONTROLBUS, CS1YELLOW);
          setLow(CONTROLBUS, CS2YELLOW);
          setHigh(CONTROLBUS, DI);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_YELLOW);

          while(DATABUS >= 0b10000000) {
                    // VΣnta tills ej busy.
          }
          setLow(CONTROLBUS, RW);
          setHigh(CONTROLBUS, CS1YELLOW);
          setHigh(CONTROLBUS, CS2YELLOW);
          DATABUS_DIR = 0xff;
}
void send_command_all_lcd(unsigned char command) {
          send_command_green1(command);
          send_command_green2(command);
          send_command_yellow1(command);
          send_command_yellow2(command);
}
void send_character_green1(unsigned char command) {
          //Skickar information till displayen
          check_busy_green();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(150);
          setLow(CONTROLBUS, CS1GREEN);
          setLow(CONTROLBUS, RW);
          setHigh(CONTROLBUS, DI);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_GREEN);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_GREEN);
          setHigh(CONTROLBUS, CS1GREEN);
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_GREEN);

          DATABUS = 0;
```

```
}
void send_character_green2(unsigned char command) {
          check_busy_green();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(150);
          setLow(CONTROLBUS, CS2GREEN);
          setLow(CONTROLBUS, RW);
          setHigh(CONTROLBUS, DI);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_GREEN);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_GREEN);
          setHigh(CONTROLBUS, CS2GREEN);
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_GREEN);

          DATABUS = 0;
}
void send_character_yellow1(unsigned char command) {
          check_busy_yellow();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(150);
          setLow(CONTROLBUS, CS1YELLOW);
          setLow(CONTROLBUS, RW);
          setHigh(CONTROLBUS, DI);
          _delay_us(50);
          setHigh(ENABLEBUS, ENABLE_YELLOW);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_YELLOW);
          setHigh(CONTROLBUS, CS1YELLOW);
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_YELLOW);

          DATABUS = 0;
}
void send_character_yellow2(unsigned char command) {
          check_busy_yellow();
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);

          setLow(ENABLEBUS, ENABLE_YELLOW);
          _delay_us(150);
          setLow(CONTROLBUS, CS2YELLOW);
          setLow(CONTROLBUS, RW);
          setHigh(CONTROLBUS, DI);
          _delay_us(50);
```

```
          setHigh(ENABLEBUS, ENABLE_YELLOW);
          DATABUS = command;
          _delay_us(80);
          setLow(ENABLEBUS, ENABLE_YELLOW);
          setHigh(CONTROLBUS, CS2YELLOW);
          setLow(CONTROLBUS, DI);
          setHigh(CONTROLBUS, RW);
          _delay_us(30);
          setHigh(ENABLEBUS, ENABLE_YELLOW);

          DATABUS = 0;
}

void send_character_all_lcd(unsigned char command) {
          send_character_green1(command);
          send_character_green2(command);
          send_character_yellow1(command);
          send_character_yellow2(command);
}
void clear_lcd_green(void) {                         //T÷mmer gr÷ns skΣrm pσ
pixlar
          for(int i = 0; i < 8; i++) {
                    for(int j = 0; j < 8; j++) {
                              clear_tile_green1(i,j);
                              clear_tile_green2(i,j);
                    }
          }
}
void clear_lcd_yellow(void) {                        //T÷mmer guls skΣrm pσ
pixlar
                              for(int i = 0; i < 8; i++) {
                                        for(int j = 0; j < 8; j++) {
                                                  clear_tile_yellow1(i,j);
                                                  clear_tile_yellow2(i,j);
                                        }
                              }
}
void clear_all_lcd(void) {
          clear_lcd_yellow();
          clear_lcd_green();
}
void clear_tile_green1(int x, int y) { //Ritar en tom ruta pσ gr÷ns h÷gra skrΣm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green1(sendY);
          send_command_green1(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_green1(0b000000);
          }
}
void clear_tile_green2(int x, int y) {
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green2(sendY);
          send_command_green2(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_green2(0b000000);
          }
```

```
}
void clear_tile_yellow1(int x, int y){    //Ritar en tom ruta pσ guls h÷gra
skrΣm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow1(sendY);
          send_command_yellow1(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_yellow1(0b000000);
          }
}
void clear_tile_yellow2(int x, int y) {
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow2(sendY);
          send_command_yellow2(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_yellow2(0b000000);
          }
}
void fill_tile_green1(int x, int y) {
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green1(sendY);
          send_command_green1(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_green1(0b11111111);
          }
}
void fill_tile_green2(int x, int y) {
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green2(sendY);
          send_command_green2(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_green2(0b11111111);
          }
}
void fill_tile_yellow1(int x, int y) {

          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow1(sendY);
          send_command_yellow1(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_yellow1(0b11111111);
          }
}
void fill_tile_yellow2(int x, int y) {
          unsigned char sendY = 64 + (7-y) * 8;
          unsigned char sendX = 184 + (7-x);
          send_command_yellow2(sendY);
          send_command_yellow2(sendX);
          for(int i = 0; i < 8; i++) {
                    send_character_yellow2(0b11111111);
          }
}
```

```
void fill_ship_green() { //Fyller i rutorna f÷r skeppet vσgrΣtt/lodrΣtt beroende
pσ ship_direction
          for(int i = 0; i < ship_length; i++) {
                    if(ship_direction == 0) {
                    fill_tile_green2(cursorW, cursorH + i);
}         else {
                    fill_tile_green2(cursorW + i, cursorH);
                    }
          }
}
void fill_ship_yellow() {
          for(int i = 0; i < ship_length; i++) {
                    if(ship_direction == 0) {
                              fill_tile_yellow2(cursorW, cursorH + i);
                              }         else {
                              fill_tile_yellow2(cursorW + i, cursorH);
                    }
          }
}
void clear_ship_green() {  //Suddar ut rutorna nΣr mark÷ren flyttas till nΣsta
ruta
          for(int i = 0; i < ship_length; i++) {
                    if(ship_direction == 0 && GreenShips[cursorW][cursorH+i] !=
1) {
                              create_grid_tile_green2(cursorW, cursorH + i);
                    }
                    else if(ship_direction == 1 && GreenShips[cursorW +
i][cursorH] != 1) {
                              create_grid_tile_green2(cursorW + i, cursorH);
                    }
          }
}
void clear_ship_yellow() {
          for(int i = 0; i < ship_length; i++) {
                    if(ship_direction == 0 && YellowShips[cursorW][cursorH+i]
!= 1) {
                              create_grid_tile_yellow2(cursorW, cursorH + i);
                              }         else if(ship_direction == 1 &&
YellowShips[cursorW + i][cursorH] != 1) {
                              create_grid_tile_yellow2(cursorW + i, cursorH);
                    }
          }
}
void create_grid_green2() {    //Skapar en matris pσ gr÷ns vΣnstra skΣrm
          for(int i = 0; i < 8; i++) {
                    for(int j = 0; j < 8; j++) {
                              create_grid_tile_green2(i,j);
                    }
          }
}
void create_grid_yellow2() {//Skapar en matris pσ guls vΣnstra skΣrm
          for(int i = 0; i < 8; i++) {
                    for(int j = 0; j < 8; j++) {
                              create_grid_tile_yellow2(i,j);
                    }
          }
}
void create_grid_tile_green2(int x, int y) {
```

```
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green2(sendY);
          send_command_green2(sendX);
          send_character_green2(0b11111111);
          for(int i = 0; i < 6; i++) {
                    send_character_green2(0b10000001);
          }
          send_character_green2(0b11111111);
}
void create_grid_tile_yellow2(int x, int y) {
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow2(sendY);
          send_command_yellow2(sendX);
          send_character_yellow2(0b11111111);
          for(int i = 0; i < 6; i++) {
                    send_character_yellow2(0b10000001);
          }
          send_character_yellow2(0b11111111);
}
void draw_line_green1(int x, int y) {     //Ritar ett snett streck pσ gr÷ns
h÷gra skΣrm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green1(sendY);
          send_command_green1(sendX);
          send_character_green1(0b00000000);
          send_character_green1(0b00000000);
          send_character_green1(0b00111100);
          send_character_green1(0b00111100);
          send_character_green1(0b00111100);
          send_character_green1(0b00111100);
          send_character_green1(0b00000000);
          send_character_green1(0b00000000);
}
void draw_line_yellow1(int x,int y) {
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow1(sendY);
          send_command_yellow1(sendX);
          send_character_yellow1(0b00000000);
          send_character_yellow1(0b00000000);
          send_character_yellow1(0b00111100);
          send_character_yellow1(0b00111100);
          send_character_yellow1(0b00111100);
          send_character_yellow1(0b00111100);
          send_character_yellow1(0b00000000);
          send_character_yellow1(0b00000000);
          }
void draw_cross_green1(int x,int y) {     //Ritar ett kryss pσ gr÷ns h÷gra skΣrm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green1(sendY);
          send_command_green1(sendX);
          send_character_green1(0b10000001);
          send_character_green1(0b01000010);
          send_character_green1(0b00100100);
```

```
          send_character_green1(0b00011000);
          send_character_green1(0b00011000);
          send_character_green1(0b00100100);
          send_character_green1(0b01000010);
          send_character_green1(0b10000001);
}
void draw_cross_yellow1(int x,int y) {    //Ritar ett kryss pσ guls h÷gra skΣrm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow1(sendY);
          send_command_yellow1(sendX);
          send_character_yellow1(0b10000001);
          send_character_yellow1(0b01000010);
          send_character_yellow1(0b00100100);
          send_character_yellow1(0b00011000);
          send_character_yellow1(0b00011000);
          send_character_yellow1(0b00100100);
          send_character_yellow1(0b01000010);
          send_character_yellow1(0b10000001);
}
void draw_line_green2(int x, int y) {     //Ritar ett kryss pσ gr÷ns vΣnstra
skΣrm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green2(sendY);
          send_command_green2(sendX);
          send_character_green2(0b11111111);
          send_character_green2(0b10000011);
          send_character_green2(0b10000101);
          send_character_green2(0b10001001);
          send_character_green2(0b10010001);
          send_character_green2(0b10100001);
          send_character_green2(0b11000001);
          send_character_green2(0b11111111);
}
void draw_line_yellow2(int x,int y) {     //Ritar ett kryss pσ guls vΣnstra
skΣrm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow2(sendY);
          send_command_yellow2(sendX);
          send_character_yellow2(0b11111111);
          send_character_yellow2(0b10000011);
          send_character_yellow2(0b10000101);
          send_character_yellow2(0b10001001);
          send_character_yellow2(0b10010001);
          send_character_yellow2(0b10100001);
          send_character_yellow2(0b11000001);
          send_character_yellow2(0b11111111);
}
void draw_cross_green2(int x,int y) {     //Ritar ett kryss pσ gr÷ns vΣnstra
skΣrm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_green2(sendY);
          send_command_green2(sendX);
          send_character_green2(0b11111111);
          send_character_green2(0b11000011);
```

```
          send_character_green2(0b10100101);
          send_character_green2(0b10011001);
          send_character_green2(0b10011001);
          send_character_green2(0b10100101);
          send_character_green2(0b11000011);
          send_character_green2(0b11111111);
}
void draw_cross_yellow2(int x,int y) {    //Ritar ett kryss pσ guls vΣnstra
skΣrm
          unsigned char sendY = 64 + (7 - y) * 8;
          unsigned char sendX = 184 + (7 - x);
          send_command_yellow2(sendY);
          send_command_yellow2(sendX);
          send_character_yellow2(0b11111111);
          send_character_yellow2(0b11000011);
          send_character_yellow2(0b10100101);
          send_character_yellow2(0b10011001);
          send_character_yellow2(0b10011001);
          send_character_yellow2(0b10100101);
          send_character_yellow2(0b11000011);
          send_character_yellow2(0b11111111);
}
void read_button_green() {
                              //lΣser av gr÷ns knapptryck
          setHigh(INTERFACEBUS, LED_GREEN);
          cursorW = 0;
                                                  //utgσngsposition f÷r
mark÷ren i vΣnstra h÷rnet
          cursorH = 0;
                                                  //utgσngsposition f÷r
mark÷ren i vΣnstra h÷rnet
          while(green_turn == 1) {
          if(get_btn_down() != 0) {
                              //h÷gerknapp spelare ett
                              if(cursorW < 7){
                                        //GrΣnser varierar beroende pσ vilken
knapp. Fσr inte ÷verstiga matrisens mσtt.
                              clear_cursor_green1(cursorW, cursorH);
                              cursorW++;
                              toggle_blink = 0;
                                        //NΣr timern uppdateras fylls rutan
f÷rsta gσngen
                    }
          }

          else  if(get_btn_up() != 0){
                              //vΣnsterknapp
                              if(cursorW > 0){
                                        clear_cursor_green1(cursorW, cursorH);
                                        cursorW--;
                                                  toggle_blink = 0;
                    }
          }
          else if(get_btn_left() != 0){
                              //nedσtknapp
                              if(cursorH > 0){
                                        clear_cursor_green1(cursorW, cursorH);
                                        cursorH--;
```

```
                                        toggle_blink = 0;
                                }
                        }
                else if(get_btn_right() != 0){
                                 //uppσtknapp
                                if(cursorH < 7){
                                        clear_cursor_green1(cursorW, cursorH);
                                        cursorH++;
                                        toggle_blink = 0;
                        }
                }

                else if(get_btn_select() != 0){
                                 //mittenknapp
                        if(GreenEnemy[cursorW][cursorH] == 0){
                        //Fσr bara "skjuta" om platsen inte Σr beskjuten sedan
tidigare (representerat av en nolla)
                                        if(YellowShips[cursorW][cursorH] ==
1){                     // Om positionen = plats dΣr det finns bσt

                draw_cross_green1(cursorW,cursorH);

                draw_cross_yellow2(cursorW,cursorH);
                                                for(int i = 0; i < 10; i++)
{                               //trΣff symboliseras av blikande lampor
                                                        _delay_ms(100);
                                                        INTERFACEBUS ^=
0b10000000;
                                                        _delay_ms(100);
                                                }
                                                setLow(INTERFACEBUS,
LED_RED);

                YellowShips[cursorW][cursorH] = 3;                      // 3 = ruta med
skepp som blivit trΣffat (kryss pσ guls vΣnstra skΣrm)
                                                GreenEnemy[cursorW][cursorH]
= 2;                    // 2 = ruta med skepp som blivit trΣffat (kryss pσ gr÷ns
h÷gra skΣrm)
                                                if(victory(YellowShips) ==
1) {                            // Kollar om gr÷n har vunnit
                                                        for(int i = 0; i
< 20; i++) {

                _delay_ms(50);

                INTERFACEBUS = 0b10000000;

                _delay_ms(50);

                INTERFACEBUS = 0b01000000;

                _delay_ms(50);

                INTERFACEBUS = 0b00100000;
                                                        }
                                                        restart_game();
                                                        //Startar om
spelet
```

```
                                                  }
                                        }else{
                                                                        //Om
skottet missade

          draw_line_green1(cursorW,cursorH);

          draw_line_yellow2(cursorW,cursorH);

          YellowShips[cursorW][cursorH] = 2;                  // 2 = ruta som
blivit beskjuten men ingen trΣff(snett streck pσ guls vΣnstra skΣrm)
                                                  GreenEnemy[cursorW][cursorH]
= 1;                // 1 = ruta som blivit beskjuten men ingen trΣff(snett
streck pσ gr÷ns h÷gra skΣrm)
                                                  _delay_ms(50);
                                                  green_turn = 0;
                                                            // Turen gσr ÷ver
till gul.
                              }
                    }
          }
}
          setLow(INTERFACEBUS, LED_GREEN);
          read_button_yellow();
                                        // Guls tur
          }
void read_button_yellow() {
                              //lΣser av guls knapptryck
          setHigh(INTERFACEBUS, LED_YELLOW);
          cursorW = 0;
          cursorH = 0;
          while(green_turn == 0) {

          if(get_btn_up() != 0){
                                        //h÷gerknapp spelare ett
                              if(cursorW > 0){
                                        //Fσr ej ÷verstiga bredden eller
h÷jden
                              clear_cursor_yellow1(cursorW, cursorH);
                              cursorW--;
                              toggle_blink = 0;
                    }
          }

          else if(get_btn_left() != 0){
                     //vΣnsterknapp
                              if(cursorH > 0){
                                        clear_cursor_yellow1(cursorW,
cursorH);
                                        cursorH--;
                                        toggle_blink = 0;
                    }
          }
          else if(get_btn_right() != 0){
                     //nedσtknapp
                              if(cursorH < 7){
                                        clear_cursor_yellow1(cursorW,
cursorH);
```

```
                                        cursorH++;
                                        toggle_blink = 0;

                                }
                        }
                else if(get_btn_down() != 0){
                        //uppσtknapp
                                if(cursorW < 7){
                                        clear_cursor_yellow1(cursorW,
cursorH);
                                        cursorW++;
                                        toggle_blink = 0;
                        }
                }

                else if(get_btn_select() != 0){
                        //mittenknapp
                        if(YellowEnemy[cursorW][cursorH] == 0){
                                if(GreenShips[cursorW][cursorH] == 1){
                // Om positionen = plats dΣr det finns bσt
                                        draw_cross_green2(cursorW,cursorH);
                                        draw_cross_yellow1(cursorW,cursorH);
                                        for(int i = 0; i < 10; i++) {
                                                _delay_ms(100);
                                                INTERFACEBUS ^= 0b10000000;
                                                _delay_ms(100);
                                        }
                                        setLow(INTERFACEBUS, LED_RED);
                                        GreenShips[cursorW][cursorH] = 3;
                                        YellowEnemy[cursorW][cursorH] = 2;
                                        if(victory(GreenShips) == 1) {
                                                for(int i = 0; i < 20; i++)
{
                                                        _delay_ms(50);
                                                        INTERFACEBUS =
0b10000000;
                                                        _delay_ms(50);
                                                        INTERFACEBUS =
0b01000000;
                                                        _delay_ms(50);
                                                        INTERFACEBUS =
0b00100000;
                                                }
                                                restart_game();
                                        }
                                        }else{
                                        draw_line_green2(cursorW,cursorH);
                                        draw_line_yellow1(cursorW,cursorH);
                                        GreenShips[cursorW][cursorH] = 2;
                                        YellowEnemy[cursorW][cursorH] = 1;
                                        _delay_ms(50);
                                        green_turn = 1;
                                }
                        }
}
                        }
                        setLow(INTERFACEBUS, LED_YELLOW);
                        read_button_green();
```

```
          }
void clear_cursor_green1(int cursorW,int cursorH) {
           //Ser till att rutan som mark÷ren stod pσ fσr rΣtt utseende nΣr
mark÷ren flyttas
          if(GreenEnemy[cursorW][cursorH] == 0){
          clear_tile_green1(cursorW,cursorH);
          }
          else if(GreenEnemy[cursorW][cursorH] == 1){
          draw_line_green1(cursorW, cursorH);
          }
          else if(GreenEnemy[cursorW][cursorH] == 2){
          draw_cross_green1(cursorW, cursorH);
          }
          }
void clear_cursor_yellow1(int cursorW,int cursorH) {
          //Hanterar mark÷rens utsuddning nΣr man gσr vidare till nΣsta ruta
          if(YellowEnemy[cursorW][cursorH] == 0){
                    clear_tile_yellow1(cursorW,cursorH);
          }
          else if(YellowEnemy[cursorW][cursorH] == 1){
                    draw_line_yellow1(cursorW, cursorH);
          }
          else if(YellowEnemy[cursorW][cursorH] == 2){
                    draw_cross_yellow1(cursorW, cursorH);
          }
}
void clear_cursor_green2(int cursorW,int cursorH) {
          if(GreenEnemy[cursorW][cursorH] == 0){
                    clear_tile_green2(cursorW,cursorH);
          }
          else if(GreenEnemy[cursorW][cursorH] == 1){
                    draw_line_green2(cursorW, cursorH);
          }
          else if(GreenEnemy[cursorW][cursorH] == 2){
                    draw_cross_green2(cursorW, cursorH);
          }
}
void clear_cursor_yellow2(int cursorW,int cursorH) {
          if(YellowEnemy[cursorW][cursorH] == 0){
                    create_grid_tile_yellow2(cursorW,cursorH);
          }
          else if(YellowEnemy[cursorW][cursorH] == 1){
                    draw_line_yellow2(cursorW, cursorH);
          }
          else if(YellowEnemy[cursorW][cursorH] == 2){
                    draw_cross_yellow2(cursorW, cursorH);
          }
}
int victory(uint8_t matrix[10][10]) {
                                        // returnerar 1 om alla skepp Σr
trΣffade, alltsσ om nσgon spelare har vunnit
          for(int i = 0; i < 9; i++) {
                    for(int j = 0; j < 9; j++) {
                              if(matrix[i][j] == 1) {
                                        return 0;
                              }
                    }
          }
```

```
          return 1;
}
void create_matrices() {
                                                    // Fyller alla fyra matriser
med nollor, vilket Σr standardvΣrdet f÷r en tom ruta
          for(int i = 1; i < 9; i++) {
                    for(int j = 1; j < 9; j++) {
                              GreenEnemy[i][j]=0;
                              GreenShips[i][j]=0;
                              YellowEnemy[i][j]=0;
                              YellowShips[i][j]=0;
                    }
          }
}
void place_ships_green() {
                                                    //placerar gr÷ns skepp
          setHigh(INTERFACEBUS, LED_GREEN);
          placing_ships = 1;
                                                              //aktiverar
timern som g÷r att skeppen kan blinka
          place_aircraft_carrier_green();
          place_carrier_green();
          place_cruiser_green();
          place_destroyer_green();
          place_submarine_green();
          _delay_ms(50);
          green_turn = 0;
          setLow(INTERFACEBUS, LED_GREEN);
}
void place_ships_yellow() {
                                                    //placerar guls skepp
          setHigh(INTERFACEBUS, LED_YELLOW);
          place_aircraft_carrier_yellow();
          place_carrier_yellow();
          place_cruiser_yellow();
          place_destroyer_yellow();
          place_submarine_yellow();
          _delay_ms(50);
          placing_ships = 0;
          setLow(INTERFACEBUS, LED_YELLOW);
          select_player();
}
void place_aircraft_carrier_green(){
          ship_direction=0;
                                                              //lodrΣtt
          ship_length = 5;
                                                              //5 rutor lσgnt
          green_turn = 1;
          cursorH = 0;

          cursorW = 0;
          int placed = 0;

while(placed == 0) {


if(get_btn_right() != 0) {
```

```
          if(cursorH < 7 && ship_direction == 1){
                              //Før ej ÷verstiga bredden eller h÷jden
                    clear_ship_green();
                                                                  //Rensar platsen
skeppet stod pσ innan knappen trycktes ned
                    cursorH++;
                                                                                //╓ka
ett i bredd
                    toggle_blink=1;
                                                                                //Fylls
f÷rsta gσngen timen avbryter
          }

          else if(cursorH < 8-ship_length && ship_direction == 0) {
                    clear_ship_green();
                    cursorH++;
                    toggle_blink=1;

          }
}
if(get_btn_left() != 0 ){

          if(cursorH > 0 && ship_direction == 1){
                    clear_ship_green();
                    cursorH--;
                    toggle_blink=1;

          } else if(cursorH > 0 && ship_direction == 0) {
                    clear_ship_green();
                    cursorH--;
                    toggle_blink=1;
                    }
          }

if(get_btn_down() != 0 ) {

                    if(cursorW < 8-ship_length && ship_direction == 1){
                              clear_ship_green();
                              cursorW++;
                    } else if(cursorW < 7 && ship_direction == 0){

                              clear_ship_green();
                              cursorW++;
          }
                    toggle_blink=1;
          }

if(get_btn_up() != 0){

                    if(cursorW > 0 && ship_direction == 1){
                              clear_ship_green();
                              cursorW--;
                    } else if(cursorW > 0 && ship_direction == 0){

                              clear_ship_green();
                              cursorW--;
                    }
                    toggle_blink=1;
```

```
          }
if(get_btn_select()!= 0) {
          fill_ship_green();
                                        //Fyll i rutorn som skeppet stσr pσ
permanent
          if(ship_direction == 1) {
                    for(int i = 0; i<ship_length;i++) {
                              GreenShips[cursorW +i][cursorH] = 1;
          // SΣtter vΣrdet pσ matrisen
                              placed = 1;
                    }
          }
          else{
                    for(int i = 0; i<ship_length;i++) {
                              GreenShips[cursorW][cursorH+i] = 1;
                              placed = 1;
                    }
          }
}
}
}
void place_carrier_green() {
          ship_direction=1;
          ship_length = 4;
          green_turn = 1;
          cursorH = 0;
          cursorW = 0;
          int placed = 0;

          while(placed == 0) {
                    if(get_btn_right() != 0) {

                              if(cursorH < 7 && ship_direction == 1){

                                        clear_ship_green();
                                        cursorH++;
                                        toggle_blink=1;
                              }

                              else if(cursorH < 8-ship_length &&
ship_direction == 0) {
                                        clear_ship_green();
                                        cursorH++;
                                        toggle_blink=1;
                              }
                    }
                    if(get_btn_left() != 0 ){
                              if(cursorH > 0 && ship_direction == 1){
                                        clear_ship_green();
                                        cursorH--;
                                        } else if(cursorH > 0 &&
ship_direction == 0) {
                                        clear_ship_green();
                                        cursorH--;
                              }
                              toggle_blink=1;
                    }
```

```
                    if(get_btn_down() != 0 ) {
                              if(cursorW < 8-ship_length && ship_direction ==
1){
                                        clear_ship_green();
                                        cursorW++;
                                        } else if(cursorW < 7 &&
ship_direction == 0){
                                        clear_ship_green();
                                        cursorW++;
                              }
                                        toggle_blink=1;
                    }

                    if(get_btn_up() != 0){
                              if(cursorW > 0 && ship_direction == 1){
                                        clear_ship_green();
                                        cursorW--;
                                        } else if(cursorW > 0 &&
ship_direction == 0){
                                        clear_ship_green();
                                        cursorW--;
                              }
                              toggle_blink=1;
                    }
                    if(get_btn_select()!= 0) {
                              if(check_ships_nearby_green()== 0) {
                              fill_ship_green();
                              if(ship_direction == 1) {
                                        for(int i = 0; i<ship_length;i++) {
                                                  GreenShips[cursorW
+i][cursorH] = 1;
                                                  placed = 1;
                                        }
                              }
                              else{
                                        for(int i = 0; i<ship_length;i++) {

          GreenShips[cursorW][cursorH+i] = 1;
                                                  placed = 1;
                                        }
                              }
                              }
                    }
          }
}
void place_cruiser_green(){
ship_direction=0;
ship_length = 3;
green_turn = 1;
cursorH = 0;
cursorW = 0;
int placed = 0;

while(placed == 0) {
```

```
if(get_btn_right() != 0) {

          if(cursorH < 7 && ship_direction == 1){
                    clear_ship_green();
                    cursorH++;
                    toggle_blink=0;                }

          else if(cursorH < 8-ship_length && ship_direction == 0) {
                    clear_ship_green();
                    cursorH++;
                    toggle_blink=0;
          }
          }
if(get_btn_left() != 0 ){
          if(cursorH > 0 && ship_direction == 1){
                    clear_ship_green();
                    cursorH--;
                    } else if(cursorH > 0 && ship_direction == 0) {
                    clear_ship_green();
                    cursorH--;
          }
          toggle_blink=0;
}

if(get_btn_down() != 0 ) {
          if(cursorW < 8-ship_length && ship_direction == 1){

                    clear_ship_green();
                    cursorW++;
                    } else if(cursorW < 7 && ship_direction == 0){

                    clear_ship_green();
                    cursorW++;
          }
          toggle_blink=0;
}

if(get_btn_up() != 0){
          if(cursorW > 0 && ship_direction == 1){
                    clear_ship_green();
                    cursorW--;
                    } else if(cursorW > 0 && ship_direction == 0){

                    clear_ship_green();
                    cursorW--;
          }
                    toggle_blink=0;
}
if(get_btn_select()!= 0) {
          if(check_ships_nearby_green()== 0) {
          fill_ship_green();
          if(ship_direction == 1) {
                    for(int i = 0; i<ship_length;i++) {
                              GreenShips[cursorW +i][cursorH] = 1;
                              placed = 1;
                    }
          }
          else{
```

```
                                        for(int i = 0; i<ship_length;i++) {
                                                  GreenShips[cursorW][cursorH+i] = 1;
                                                  placed = 1;
                                        }
                              }
                              }
                    }
                    }
}
void place_destroyer_green(){
ship_direction=1;
ship_length = 2;
green_turn = 1;
cursorH = 0;
cursorW = 0;
int placed = 0;

while(placed == 0) {



                    if(get_btn_right() != 0) {

                              if(cursorH < 7 && ship_direction == 1){
                                        clear_ship_green();
                                        cursorH++;
                                        toggle_blink=0;
                              }

                              else if(cursorH < 8-ship_length && ship_direction == 0) {
                                        clear_ship_green();
                                        cursorH++;
                                        toggle_blink=0;
                              }
                    }
                    if(get_btn_left() != 0 ){
                              if(cursorH > 0 && ship_direction == 1){
                                        clear_ship_green();
                                        cursorH--;
                                        } else if(cursorH > 0 && ship_direction == 0) {
                                        clear_ship_green();
                                        cursorH--;
                              }
                                        toggle_blink=0;
                    }

                    if(get_btn_down() != 0 ) {
                              if(cursorW < 8-ship_length && ship_direction == 1){

                                        clear_ship_green();
                                        cursorW++;
                                        } else if(cursorW < 7 && ship_direction == 0){

                                        clear_ship_green();
                                        cursorW++;
                              }
                                        toggle_blink=0;
                    }
```

```
		if(get_btn_up() != 0){
			if(cursorW > 0 && ship_direction == 1){
				clear_ship_green();
				cursorW--;
				} else if(cursorW > 0 && ship_direction == 0){

				clear_ship_green();
				cursorW--;
			}
			toggle_blink=0;
		}
		if(get_btn_select()!= 0) {
			if(check_ships_nearby_green()== 0) {
			fill_ship_green();
			if(ship_direction == 1) {
				for(int i = 0; i<ship_length;i++) {
					GreenShips[cursorW+i][cursorH] = 1;
					placed = 1;
				}
			}
			else{
				for(int i = 0; i<ship_length;i++) {
					GreenShips[cursorW][cursorH+i] = 1;
					placed = 1;
				}
			}
			toggle_blink=0;
			}
		}
}
}
void place_submarine_green(){
ship_direction=0;
ship_length = 1;
green_turn = 1;
cursorH = 0;
cursorW = 0;
int placed = 0;

while(placed == 0) {


		if(get_btn_right() != 0) {

			if(cursorH < 7 && ship_direction == 1){
				clear_ship_green();
				cursorH++;
				toggle_blink=0;
			}

			else if(cursorH < 8-ship_length && ship_direction == 0) {
				clear_ship_green();
				cursorH++;
				toggle_blink=0;
			}
		}
		if(get_btn_left() != 0 ){
```

```
                    if(cursorH > 0 && ship_direction == 1){
                              clear_ship_green();
                              cursorH--;
                              } else if(cursorH > 0 && ship_direction == 0) {
                              clear_ship_green();
                              cursorH--;
                    }
                    toggle_blink=0;

          }

          if(get_btn_down() != 0 ) {
                    if(cursorW < 8-ship_length && ship_direction == 1){

                              clear_ship_green();
                              cursorW++;
                              } else if(cursorW < 7 && ship_direction == 0){

                              clear_ship_green();
                              cursorW++;
                    }
                    toggle_blink=0;
          }

          if(get_btn_up() != 0){
                    if(cursorW > 0 && ship_direction == 1){
                              clear_ship_green();
                              cursorW--;
                              } else if(cursorW > 0 && ship_direction == 0){

                              clear_ship_green();
                              cursorW--;
                    }
                    toggle_blink=0;
          }
          if(get_btn_select()!= 0) {
                    if(check_ships_nearby_green()== 0) {
                    fill_ship_green();
                    if(ship_direction == 1) {
                              for(int i = 0; i<ship_length;i++) {
                                        GreenShips[cursorW+i][cursorH] = 1;
                                        placed = 1;
                              }
                    }
                    else{
                              for(int i = 0; i<ship_length;i++) {
                                        GreenShips[cursorW][cursorH+i] = 1;
                                        placed = 1;
                              }
                    }
                    }
          }
}
          }
void place_aircraft_carrier_yellow(){
          ship_direction=0;
          ship_length = 5;
          cursorH = 0;
```

```
		cursorW = 0;
		int placed = 0;

		while(placed == 0) {

				if(get_btn_right() != 0) {

					if(cursorH < 7 && ship_direction == 1){
						clear_ship_yellow();
						cursorH++;
					}

					else if(cursorH < 8-ship_length &&
ship_direction == 0) {
						clear_ship_yellow();
						cursorH++;
					}
					toggle_blink=0;
				}
				if(get_btn_left() != 0 ){
					if(cursorH > 0 && ship_direction == 1){
						clear_ship_yellow();
						cursorH--;
						} else if(cursorH > 0 &&
ship_direction == 0) {
						clear_ship_yellow();
						cursorH--;
					}
					toggle_blink=0;
				}

				if(get_btn_down() != 0 ) {
					if(cursorW < 8-ship_length && ship_direction ==
1){
						clear_ship_yellow();
						cursorW++;
						} else if(cursorW < 7 &&
ship_direction == 0){
						clear_ship_yellow();
						cursorW++;
					}
					toggle_blink=0;
				}

				if(get_btn_up() != 0){
					if(cursorW > 0 && ship_direction == 1){
						clear_ship_yellow();
						cursorW--;
						} else if(cursorW > 0 &&
ship_direction == 0){
						clear_ship_yellow();
						cursorW--;
					}
					toggle_blink=0;
				}
				if(get_btn_select()!= 0) {
						fill_ship_yellow();
						if(ship_direction == 1) {
```

```
                                                  for(int i = 0;
i<ship_length;i++) {

          YellowShips[cursorW+i][cursorH] = 1;
                                                            placed = 1;
                                                  }
                                        }
                                        else{
                                                  for(int i = 0;
i<ship_length;i++) {

          YellowShips[cursorW][cursorH+i] = 1;
                                                            placed = 1;
                                                  }
                                        }
                    }
}
                    }
void place_carrier_yellow(){
          ship_direction=1;
          ship_length = 4;
          cursorH = 0;
          cursorW = 0;
          int placed = 0;

          while(placed == 0) {


                    if(get_btn_right() != 0) {

                              if(cursorH < 7 && ship_direction == 1){

                                        clear_ship_yellow();
                                        cursorH++;
                              }

                              else if(cursorH < 8-ship_length &&
ship_direction == 0) {
                                        clear_ship_yellow();
                                        cursorH++;
                              }
                              toggle_blink=0;
                    }
                    if(get_btn_left() != 0 ){

                              if(cursorH > 0 && ship_direction == 1){
                                        clear_ship_yellow();
                                        cursorH--;
                                        } else if(cursorH > 0 &&
ship_direction == 0) {
                                        clear_ship_yellow();
                                        cursorH--;
                              }
                              toggle_blink=0;
                    }

                    if(get_btn_down() != 0 ) {
```

```
                                if(cursorW < 8-ship_length && ship_direction ==
1){
                                          clear_ship_yellow();
                                          cursorW++;
                                          } else if(cursorW < 7 &&
ship_direction == 0){
                                          clear_ship_yellow();
                                          cursorW++;
                                }
                                toggle_blink=0;
                     }

                     if(get_btn_up() != 0){

                                if(cursorW > 0 && ship_direction == 1){
                                          clear_ship_yellow();
                                          cursorW--;
                                          } else if(cursorW > 0 &&
ship_direction == 0){
                                          clear_ship_yellow();
                                          cursorW--;
                                }
                                toggle_blink=0;
                     }
                     if(get_btn_select()!= 0) {
                                if(check_ships_nearby_yellow()== 0) {
                                          fill_ship_yellow();
                                          if(ship_direction == 1) {
                                                    for(int i = 0;
i<ship_length;i++) {

          YellowShips[cursorW+i][cursorH] = 1;
                                                              placed = 1;
                                                    }
                                          }
                                          else{
                                                    for(int i = 0;
i<ship_length;i++) {

          YellowShips[cursorW][cursorH+i] = 1;
                                                              placed = 1;
                                                    }
                                          }
                                }
                     }
          }
}
void place_cruiser_yellow(){
          ship_direction=0;
          ship_length = 3;
          cursorH = 0;
          cursorW = 0;
          int placed = 0;

          while(placed == 0) {
```

```
                    if(get_btn_right() != 0) {

                              if(cursorH < 7 && ship_direction == 1){

                                        clear_ship_yellow();
                                        cursorH++;
                              }
                              else if(cursorH < 8-ship_length &&
ship_direction == 0) {
                                        clear_ship_yellow();
                                        cursorH++;
                              }
                              toggle_blink=0;
                    }
                    if(get_btn_left() != 0 ){
                              if(cursorH > 0 && ship_direction == 1){
                                        clear_ship_yellow();
                                        cursorH--;
                                        } else if(cursorH > 0 &&
ship_direction == 0) {
                                        clear_ship_yellow();
                                        cursorH--;
                              }
                              toggle_blink=0;
                    }

                    if(get_btn_down() != 0 ) {
                              if(cursorW < 8-ship_length && ship_direction ==
1){
                                        clear_ship_yellow();
                                        cursorW++;
                                        } else if(cursorW < 7 &&
ship_direction == 0){
                                        clear_ship_yellow();
                                        cursorW++;
                              }
                              toggle_blink=0;
                    }

                    if(get_btn_up() != 0){
                              if(cursorW > 0 && ship_direction == 1){
                                        clear_ship_yellow();
                                        cursorW--;
                                        } else if(cursorW > 0 &&
ship_direction == 0){
                                        clear_ship_yellow();
                                        cursorW--;
                              }
                              toggle_blink=0;
                    }
                    if(get_btn_select()!= 0) {
                              if(check_ships_nearby_yellow()== 0) {
                                        fill_ship_yellow();
                                        if(ship_direction == 1) {
                                                  for(int i = 0;
i<ship_length;i++) {

          YellowShips[cursorW+i][cursorH] = 1;
```

```
                                                                placed = 1;
                                                        }
                                                }
                                                else{
                                                        for(int i = 0;
i<ship_length;i++) {

            YellowShips[cursorW][cursorH+i] = 1;
                                                                placed = 1;
                                                        }
                                                }
                                        }
                                }
            }
}
void place_destroyer_yellow(){
            ship_direction=1;
            ship_length = 2;
            cursorH = 0;
            cursorW = 0;
            int placed = 0;

            while(placed == 0) {


                        if(get_btn_right() != 0) {

                                    if(cursorH < 7 && ship_direction == 1){

                                                clear_ship_yellow();
                                                cursorH++;
                                                toggle_blink=1;
                                    }

                                    else if(cursorH < 8-ship_length &&
ship_direction == 0) {
                                                clear_ship_yellow();
                                                cursorH++;
                                                toggle_blink=0;
                                    }
                        }
                        if(get_btn_left() != 0 ){
                                    if(cursorH > 0 && ship_direction == 1){
                                                clear_ship_yellow();
                                                cursorH--;
                                                } else if(cursorH > 0 &&
ship_direction == 0) {
                                                clear_ship_yellow();
                                                cursorH--;
                                    }
                                    toggle_blink=0;
                        }

                        if(get_btn_down() != 0 ) {
                                    if(cursorW < 8-ship_length && ship_direction ==
1){
                                                clear_ship_yellow();
                                                cursorW++;
```

```
                                        } else if(cursorW < 7 &&
ship_direction == 0){
                                        clear_ship_yellow();
                                        cursorW++;
                                }
                                toggle_blink=0;
                        }

                        if(get_btn_up() != 0){
                                if(cursorW > 0 && ship_direction == 1){
                                        clear_ship_yellow();
                                        cursorW--;
                                        } else if(cursorW > 0 &&
ship_direction == 0){
                                        clear_ship_yellow();
                                        cursorW--;
                                }
                                toggle_blink=0;
                        }
                        if(get_btn_select()!= 0) {
                                if(check_ships_nearby_yellow()== 0) {
                                        fill_ship_yellow();
                                        if(ship_direction == 1) {
                                                for(int i = 0;
i<ship_length;i++) {

            YellowShips[cursorW+i][cursorH] = 1;
                                                                placed = 1;
                                                }
                                        }
                                        else{
                                                for(int i = 0;
i<ship_length;i++) {

            YellowShips[cursorW][cursorH+i] = 1;
                                                                placed = 1;
                                                }
                                        }
                                }
                        }
            }
}
void place_submarine_yellow(){
            ship_direction=0;
            ship_length = 1;
            cursorH = 0;
            cursorW = 0;
            int placed = 0;

            while(placed == 0) {


                        if(get_btn_right() != 0) {

                                if(cursorH < 7 && ship_direction == 1){

                                        clear_ship_yellow();
                                        cursorH++;
```

```
                                }

                                else if(cursorH < 8-ship_length &&
ship_direction == 0) {
                                        clear_ship_yellow();
                                        cursorH++;
                                }
                                toggle_blink=0;
                                select_player();
                        }
                        if(get_btn_left() != 0 ){

                                if(cursorH > 0 && ship_direction == 1){
                                        clear_ship_yellow();
                                        cursorH--;
                                        } else if(cursorH > 0 &&
ship_direction == 0) {
                                        clear_ship_yellow();
                                        cursorH--;
                                }
                                toggle_blink=0;
                        }

                        if(get_btn_down() != 0 ) {

                                if(cursorW < 8-ship_length && ship_direction ==
1){
                                        clear_ship_yellow();
                                        cursorW++;
                                        } else if(cursorW < 7 &&
ship_direction == 0){
                                        clear_ship_yellow();
                                        cursorW++;
                                }
                                toggle_blink=0;
                        }

                        if(get_btn_up() != 0){

                                if(cursorW > 0 && ship_direction == 1){
                                        clear_ship_yellow();
                                        cursorW--;
                                        } else if(cursorW > 0 &&
ship_direction == 0){
                                        clear_ship_yellow();
                                        cursorW--;
                                }
                                toggle_blink=0;
                        }
                        if(get_btn_select()!= 0) {
                                if(check_ships_nearby_yellow()== 0) {
                                        fill_ship_yellow();
                                        if(ship_direction == 1) {
                                                for(int i = 0;
i<ship_length;i++) {

          YellowShips[cursorW+i][cursorH] = 1;
                                                                placed = 1;
```

```
                                                  }
                                        }
                                        else{
                                                  for(int i = 0;
i<ship_length;i++) {

          YellowShips[cursorW][cursorH+i] = 1;
                                                            placed = 1;
                                                  }
                                        }
                              }
                    }
          }
}
int check_ships_nearby_green() {                  //kontrollerar att spelaren
inte f÷rs÷ker placera skeppet pσ ett otillσtet stΣlle (brevid ett annat skepp)
          int nearby = 0;
          if(ship_direction == 0) {
                    for(int i = 0; i < ship_length; i++) {

                              if(GreenShips[cursorW][cursorH+i] == 1 ||
GreenShips[cursorW][cursorH+i+1] == 1 || GreenShips[cursorW][cursorH+i-1] == 1
||
                               GreenShips[cursorW+1][cursorH+i] == 1 ||
GreenShips[cursorW+1][cursorH+i+1] == 1 || GreenShips[cursorW+1][cursorH+i-1] ==
1 ||
                                GreenShips[cursorW-1][cursorH+i] == 1 ||
GreenShips[cursorW-1][cursorH+i+1] == 1 || GreenShips[cursorW-1][cursorH+i-1] ==
1) {
                                        nearby = 1;
                               }
                    }
          }
          if(ship_direction == 1) {
                    for(int i = 0; i < ship_length; i++) {
                              if(GreenShips[cursorW+i][cursorH] == 1 ||
GreenShips[cursorW+i][cursorH+1] == 1 || GreenShips[cursorW+i][cursorH-1] == 1
||
                              GreenShips[cursorW+1+i][cursorH] == 1 ||
GreenShips[cursorW+1+i][cursorH+1] == 1 || GreenShips[cursorW+1+i][cursorH-1] ==
1 ||
                              GreenShips[cursorW-1+i][cursorH] == 1 ||
GreenShips[cursorW-1+i][cursorH+1] == 1 || GreenShips[cursorW-1+i][cursorH-1] ==
1) {

                                        nearby = 1;
                              }
                    }
          }
          return nearby;
}
int check_ships_nearby_yellow() {
          int nearby = 0;
          if(ship_direction == 0) {
                    for(int i = 0; i < ship_length; i++) {
                              if(YellowShips[cursorW][cursorH+i] == 1 ||
YellowShips[cursorW][cursorH+i+1] == 1 || YellowShips[cursorW][cursorH+i-1] == 1
||
```

```
                                    YellowShips[cursorW+1][cursorH+i] == 1 ||
YellowShips[cursorW+1][cursorH+i+1] == 1 || YellowShips[cursorW+1][cursorH+i-1]
== 1 ||
                                     YellowShips[cursorW-1][cursorH+i] == 1 ||
YellowShips[cursorW-1][cursorH+i+1] == 1 || YellowShips[cursorW-1][cursorH+i-1]
== 1) {
                                                  nearby = 1;
                                    }
                          }
                }
                if(ship_direction == 1) {
                          for(int i = 0; i < ship_length; i++) {
                                    if(YellowShips[cursorW+i][cursorH] == 1 ||
YellowShips[cursorW+i][cursorH+1] == 1 || YellowShips[cursorW+i][cursorH-1] == 1
||
                                    YellowShips[cursorW+1+i][cursorH] == 1 ||
YellowShips[cursorW+1+i][cursorH+1] == 1 || YellowShips[cursorW+1+i][cursorH-1]
== 1 ||
                                    YellowShips[cursorW-1+i][cursorH] == 1 ||
YellowShips[cursorW-1+i][cursorH+1] == 1 || YellowShips[cursorW-1+i][cursorH-1]
== 1) {
                                                  nearby = 1;
                                    }
                          }
                }
                return nearby;
}
void select_player() {                    //slumpar fram vilken spelare som ska
b÷rja genom att anvΣnda vΣrdet pσ overflow_blink
                if (get_overflow_blink() % 2 == 0){
                          green_turn = 0;
                }
                else {
                          green_turn = 1;
                }

}
```