

Portable Game Console

Wygenerowano przez Doxygen 1.8.13

Spis treści

1	Dokumentacja struktur danych	2
1.1	Dokumentacja struktury Button	2
1.2	Dokumentacja struktury Element	2
1.3	Dokumentacja struktury TouchPoint	2
1.4	Dokumentacja struktury Vector	2
2	Dokumentacja plików	3
2.1	Dokumentacja pliku AccelerometerController.c	3
2.1.1	Opis szczegółowy	3
2.1.2	Dokumentacja funkcji	3
2.2	Dokumentacja pliku AccelerometerController.h	4
2.2.1	Opis szczegółowy	4
2.2.2	Dokumentacja funkcji	4
2.3	Dokumentacja pliku ApplicationMenu.c	5
2.3.1	Opis szczegółowy	5
2.4	Dokumentacja pliku ApplicationMenu.h	5
2.4.1	Opis szczegółowy	5
2.5	Dokumentacja pliku Button.c	6
2.5.1	Opis szczegółowy	6
2.5.2	Dokumentacja funkcji	6
2.6	Dokumentacja pliku Button.h	8
2.6.1	Opis szczegółowy	8
2.6.2	Dokumentacja funkcji	8
2.7	Dokumentacja pliku Desktop.c	10
2.7.1	Opis szczegółowy	10
2.7.2	Dokumentacja funkcji	10
2.8	Dokumentacja pliku Desktop.h	10
2.8.1	Opis szczegółowy	11
2.8.2	Dokumentacja funkcji	11

2.9	Dokumentacja pliku DisplayTexts.h	11
2.9.1	Opis szczegółowy	11
2.10	Dokumentacja pliku Font.c	12
2.10.1	Opis szczegółowy	12
2.10.2	Dokumentacja funkcji	12
2.11	Dokumentacja pliku Font.h	14
2.11.1	Opis szczegółowy	14
2.11.2	Dokumentacja funkcji	14
2.12	Dokumentacja pliku GamesMenu.c	15
2.12.1	Opis szczegółowy	15
2.13	Dokumentacja pliku GamesMenu.h	15
2.13.1	Opis szczegółowy	16
2.14	Dokumentacja pliku GlobalErrors.h	16
2.14.1	Opis szczegółowy	16
2.15	Dokumentacja pliku GraphicsPrimitives.c	16
2.15.1	Opis szczegółowy	17
2.15.2	Dokumentacja funkcji	17
2.16	Dokumentacja pliku GraphicsPrimitives.h	20
2.16.1	Opis szczegółowy	21
2.16.2	Dokumentacja funkcji	21
2.17	Dokumentacja pliku I2CController.c	24
2.17.1	Opis szczegółowy	24
2.17.2	Dokumentacja funkcji	24
2.18	Dokumentacja pliku I2CController.h	25
2.18.1	Opis szczegółowy	25
2.18.2	Dokumentacja funkcji	25
2.19	Dokumentacja pliku LCDController.c	26
2.19.1	Opis szczegółowy	26
2.19.2	Dokumentacja funkcji	26
2.20	Dokumentacja pliku LCDController.h	27

2.20.1	Opis szczegółowy	28
2.20.2	Dokumentacja funkcji	29
2.21	Dokumentacja pliku Main.c	29
2.21.1	Opis szczegółowy	30
2.21.2	Dokumentacja funkcji	30
2.22	Dokumentacja pliku MainMenu.c	30
2.22.1	Opis szczegółowy	30
2.23	Dokumentacja pliku MainMenu.h	31
2.23.1	Opis szczegółowy	31
2.24	Dokumentacja pliku MenuOptions.h	31
2.24.1	Opis szczegółowy	31
2.25	Dokumentacja pliku Paint.c	31
2.25.1	Opis szczegółowy	32
2.25.2	Dokumentacja funkcji	32
2.26	Dokumentacja pliku Paint.h	32
2.26.1	Opis szczegółowy	33
2.26.2	Dokumentacja funkcji	33
2.27	Dokumentacja pliku PingPong.c	34
2.27.1	Opis szczegółowy	34
2.27.2	Dokumentacja funkcji	34
2.28	Dokumentacja pliku PingPong.h	35
2.28.1	Opis szczegółowy	35
2.28.2	Dokumentacja funkcji	35
2.29	Dokumentacja pliku Question.c	36
2.29.1	Opis szczegółowy	36
2.29.2	Dokumentacja funkcji	36
2.30	Dokumentacja pliku Question.h	37
2.30.1	Opis szczegółowy	37
2.30.2	Dokumentacja funkcji	37
2.31	Dokumentacja pliku Snake.c	37

2.31.1	Opis szczegółowy	38
2.31.2	Dokumentacja funkcji	38
2.32	Dokumentacja pliku Snake.h	38
2.32.1	Opis szczegółowy	39
2.32.2	Dokumentacja funkcji	39
2.33	Dokumentacja pliku SnakeElements.c	39
2.33.1	Opis szczegółowy	39
2.33.2	Dokumentacja funkcji	40
2.34	Dokumentacja pliku SnakeElements.h	41
2.34.1	Opis szczegółowy	42
2.34.2	Dokumentacja funkcji	42
2.35	Dokumentacja pliku SnakeMove.c	43
2.35.1	Opis szczegółowy	44
2.35.2	Dokumentacja funkcji	44
2.36	Dokumentacja pliku SnakeMove.h	46
2.36.1	Opis szczegółowy	46
2.36.2	Dokumentacja funkcji	46
2.37	Dokumentacja pliku SPIController.c	48
2.37.1	Opis szczegółowy	48
2.37.2	Dokumentacja funkcji	48
2.38	Dokumentacja pliku SPIController.h	49
2.38.1	Opis szczegółowy	49
2.38.2	Dokumentacja funkcji	49
2.39	Dokumentacja pliku ToolSet.c	49
2.39.1	Opis szczegółowy	50
2.39.2	Dokumentacja funkcji	50
2.40	Dokumentacja pliku ToolSet.h	51
2.40.1	Opis szczegółowy	51
2.40.2	Dokumentacja funkcji	51
2.41	Dokumentacja pliku TouchController.c	52
2.41.1	Opis szczegółowy	53
2.41.2	Dokumentacja funkcji	53
2.42	Dokumentacja pliku TouchController.h	55
2.42.1	Opis szczegółowy	55
2.42.2	Dokumentacja funkcji	55
2.43	Dokumentacja pliku Welcome.c	57
2.43.1	Opis szczegółowy	58
2.44	Dokumentacja pliku Welcome.h	58
2.44.1	Opis szczegółowy	58

1 Dokumentacja struktur danych

1.1 Dokumentacja struktury Button

Pola danych

- char * **string**
- uint16_t **textColor**
- uint16_t **buttonColor**
- uint8_t **x**
- uint16_t **y**
- uint8_t **width**
- uint16_t **height**
- [Option](#) **earlierOption**
- void(* **Display**)(struct [Button](#) *)

1.2 Dokumentacja struktury Element

Pola danych

- uint8_t **x**
- uint8_t **y**
- [Color](#) **color**
- [Direction](#) **newDirection**
- [Direction](#) **oldDirection**

1.3 Dokumentacja struktury TouchPoint

Pola danych

- uint8_t **x**
- uint16_t **y**
- uint32_t **xad**
- uint32_t **yad**
- uint8_t **status**

1.4 Dokumentacja struktury Vector

Pola danych

- int8_t **x**
- int8_t **y**

2 Dokumentacja plików

2.1 Dokumentacja pliku AccelerometerController.c

Funkcje

- void [AccelerometerInit](#) (void)
- int8_t [AccelerometerGetAngle](#) ([Axis](#) axis)
- [Direction](#) [AccelerometerGetDirection](#) (void)

2.1.1 Opis szczegółowy

Autor

Mateusz Chudy

2.1.2 Dokumentacja funkcji

2.1.2.1 AccelerometerGetAngle()

```
int8_t AccelerometerGetAngle (  
    Axis axis )
```

Pobiera wartosc proporcjonalna do kata nachylenia do danej osi.

Parametry

<i>axis</i>	- indeks rejestru, pod ktorym znajduje sie wartosc dla danej osi
-------------	--

Zwraca

wartosc proporcjonalna do kata nachylenia do danej osi

2.1.2.2 AccelerometerGetDirection()

```
Direction AccelerometerGetDirection (  
    void )
```

Pobiera kierunek nachylenia.

Zwraca

kierunek nachylenia

2.2 Dokumentacja pliku AccelerometerController.h

Definicje

- `#define X0 0`
- `#define Y0 -5`
- `#define WRITE 0x3A`
- `#define READ 0x3B`
- `#define ACCE_SENSITIVITY 10`

Wyczerpania

- `enum Axis { AXIS_X = 0x29, AXIS_Y = 0x2B, AXIS_Z = 0x2D }`
- `enum Direction { CONST, UP, DOWN, LEFT, RIGHT }`

Funkcje

- `void AccelerometerInit (void)`
- `int8_t AccelerometerGetAngle (Axis axis)`
- `Direction AccelerometerGetDirection (void)`

2.2.1 Opis szczegółowy

Autor

Mateusz Chudy

2.2.2 Dokumentacja funkcji

2.2.2.1 AccelerometerGetAngle()

```
int8_t AccelerometerGetAngle (  
    Axis axis )
```

Pobiera wartosc proporcjonalna do kata nachylenia do danej osi.

Parametry

<code>axis</code>	- indeks rejestru, pod ktorym znajduje sie wartosc dla danej osi
-------------------	--

Zwraca

wartosc proporcjonalna do kata nachylenia do danej osi

2.2.2.2 AccelerometerGetDirection()

```
Direction AccelerometerGetDirection (
    void )
```

Pobiera kierunek nachylenia.

Zwraca

kierunek nachylenia

2.3 Dokumentacja pliku ApplicationMenu.c

Funkcje

- void [ApplicationMenuShow](#) (void)

2.3.1 Opis szczegółowy

Autor

Mateusz Chudy

2.4 Dokumentacja pliku ApplicationMenu.h

Definicje

- #define **APP_PAINT_BUTTON_X1** 0
- #define **APP_PAINT_BUTTON_Y1** 0
- #define **APP_PAINT_BUTTON_X2** 240
- #define **APP_PAINT_BUTTON_Y2** 40
- #define **APP_PAINT_BUTTON_COLOR** DARKBLUE
- #define **APP_PAINT_TEXT_COLOR** WHITE
- #define **APP_SPIRIT_LEVEL_BUTTON_X1** 0
- #define **APP_SPIRIT_LEVEL_BUTTON_Y1** 40
- #define **APP_SPIRIT_LEVEL_BUTTON_X2** 240
- #define **APP_SPIRIT_LEVEL_BUTTON_Y2** 80
- #define **APP_SPIRIT_LEVEL_BUTTON_COLOR** 0X01CA
- #define **APP_SPIRIT_LEVEL_TEXT_COLOR** WHITE

Funkcje

- void [ApplicationMenuShow](#) (void)

2.4.1 Opis szczegółowy

Autor

Mateusz Chudy

2.5 Dokumentacja pliku Button.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- `struct Button * ButtonCreate` (`char *string`, `Option option`, `uint16_t textColor`, `uint16_t buttonColor`, `uint8_t x`, `uint16_t y`, `uint8_t width`, `uint16_t height`)
- `struct Button ** ButtonDestroy` (`struct Button *button`)
- `void ButtonDisplay` (`struct Button *button`)
- `bool ButtonCheck` (`struct Button *button`)

2.5.1 Opis szczegółowy

Autor

Mateusz Chudy

2.5.2 Dokumentacja funkcji

2.5.2.1 ButtonCheck()

```
bool ButtonCheck (  
    struct Button * button )
```

Sprawdza czy przycisk został naciśnięty.

Parametry

<i>button</i>	- wskaźnik na strukturę przycisku
---------------	-----------------------------------

Zwraca

wartość logiczna oznaczająca czy przycisk został wciśnięty

2.5.2.2 ButtonCreate()

```
struct Button* ButtonCreate (  
    char * string,  
    Option option,  
    uint16_t textColor,  
    uint16_t buttonColor,  
    uint8_t x,
```

```
uint16_t y,  
uint8_t width,  
uint16_t height )
```

Tworzy nowy przycisk.

Parametry

<i>string</i>	- napis na przycisku
<i>option</i>	- opcja z ktorej powstal przycisk
<i>textColor</i>	- kolor tekstu
<i>buttonColor</i>	- kolor przycisku
<i>x</i>	- wspolrzeczna x przycisku
<i>y</i>	- wspolrzeczna y przycisku
<i>width</i>	- szerokosc przycisku
<i>height</i>	- wysokosc przycisku

Zwraca

wskaznik na strukture nowego przycisku

2.5.2.3 ButtonDestroy()

```
struct Button** ButtonDestroy (  
    struct Button * button )
```

Niszczy przycisk.

Parametry

<i>button</i>	- wskaznik na strukture przycisku
---------------	-----------------------------------

Zwraca

wskaznik na strukture przycisku

2.5.2.4 ButtonDisplay()

```
void ButtonDisplay (  
    struct Button * button )
```

Wyswietla przycisk.

Parametry

<i>button</i>	- wskaznik na strukture przycisku
---------------	-----------------------------------

2.6 Dokumentacja pliku Button.h

Struktury danych

- struct [Button](#)

Funkcje

- struct [Button](#) * [ButtonCreate](#) (char *string, [Option](#) option, uint16_t textColor, uint16_t buttonColor, uint8_t x, uint16_t y, uint8_t width, uint16_t height)
- struct [Button](#) ** [ButtonDestroy](#) (struct [Button](#) *button)
- void [ButtonDisplay](#) (struct [Button](#) *button)
- bool [ButtonCheck](#) (struct [Button](#) *button)

2.6.1 Opis szczegółowy

Autor

Mateusz Chudy

2.6.2 Dokumentacja funkcji

2.6.2.1 ButtonCheck()

```
bool ButtonCheck (  
    struct Button * button )
```

Sprawdza czy przycisk został naciśnięty.

Parametry

<i>button</i>	- wskaźnik na strukturę przycisku
---------------	-----------------------------------

Zwraca

wartość logiczna oznaczająca czy przycisk został wciśnięty

2.6.2.2 ButtonCreate()

```
struct Button* ButtonCreate (  
    char * string,  
    Option option,  
    uint16_t textColor,  
    uint16_t buttonColor,  
    uint8_t x,
```

```
uint16_t y,  
uint8_t width,  
uint16_t height )
```

Tworzy nowy przycisk.

Parametry

<i>string</i>	- napis na przycisku
<i>option</i>	- opcja z ktorej powstal przycisk
<i>textColor</i>	- kolor tekstu
<i>buttonColor</i>	- kolor przycisku
<i>x</i>	- wspolrzeczna x przycisku
<i>y</i>	- wspolrzeczna y przycisku
<i>width</i>	- szerokosc przycisku
<i>height</i>	- wysokosc przycisku

Zwraca

wskaznik na strukture nowego przycisku

2.6.2.3 ButtonDestroy()

```
struct Button** ButtonDestroy (  
    struct Button * button )
```

Niszczy przycisk.

Parametry

<i>button</i>	- wskaznik na strukture przycisku
---------------	-----------------------------------

Zwraca

wskaznik na strukture przycisku

2.6.2.4 ButtonDisplay()

```
void ButtonDisplay (  
    struct Button * button )
```

Wyswietla przycisk.

Parametry

<i>button</i>	- wskaznik na strukture przycisku
---------------	-----------------------------------

2.7 Dokumentacja pliku Desktop.c

Funkcje

- void [DesktopCheckSelect](#) (void)
- void [DesktopShow](#) (void)
- void [DesktopInvokeShowFunction](#) ([Option](#) option)

2.7.1 Opis szczegółowy

Autor

Mateusz Chudy

2.7.2 Dokumentacja funkcji

2.7.2.1 DesktopInvokeShowFunction()

```
void DesktopInvokeShowFunction (  
    Option option )
```

Wywołuje funkcje uruchomienia wybranej opcji menu.

Parametry

<i>option</i>	- opcja która uruchamiamy
---------------	---------------------------

2.8 Dokumentacja pliku Desktop.h

Definicje

- `#define DESK_MENU_BUTTON_X1 0`
- `#define DESK_MENU_BUTTON_Y1 280`
- `#define DESK_MENU_BUTTON_X2 240`
- `#define DESK_MENU_BUTTON_Y2 320`
- `#define DESK_MENU_BUTTON_COLOR DARKBLUE`
- `#define DESK_MENU_TEXT_COLOR WHITE`
- `#define BACK_BUTTON_X1 0`
- `#define BACK_BUTTON_Y1 280`
- `#define BACK_BUTTON_X2 240`
- `#define BACK_BUTTON_Y2 320`
- `#define BACK_BUTTON_COLOR RED`
- `#define BACK_TEXT_COLOR WHITE`

Funkcje

- void [DesktopCheckSelect](#) (void)
- void [DesktopShow](#) (void)
- void [DesktopInvokeShowFunction](#) ([Option](#) option)

Zmienne

- struct `Button` * `buttonArray` [LAST]

2.8.1 Opis szczegółowy

Autor

Mateusz Chudy

2.8.2 Dokumentacja funkcji

2.8.2.1 DesktopInvokeShowFunction()

```
void DesktopInvokeShowFunction (
    Option option )
```

Wywołuje funkcje uruchomienia wybranej opcji menu.

Parametry

<i>option</i>	- opcja ktora uruchamiamy
---------------	---------------------------

2.9 Dokumentacja pliku DisplayTexts.h

Definicje

- #define **WELCOME_TEXT** "WELCOME!"
- #define **MENU_TEXT** "Menu"
- #define **GAMES_TEXT** "Games"
- #define **APPLICATION_TEXT** "Application"
- #define **PING_PONG_TEXT** "Ping Pong"
- #define **SNAKE_TEXT** "Snake"
- #define **PAINT_TEXT** "Paint"
- #define **BACK_TEXT** "Back"
- #define **EXIT_TEXT** "Exit"
- #define **UNKNOWN_OPTION_TEXT** "Unknown option."
- #define **GAME_OVER_TEXT** "Game Over"
- #define **GAME_RETRY_QUESTION** "Do you wanna play again?"
- #define **YES_TEXT** "Yes"
- #define **NO_TEXT** "No"

2.9.1 Opis szczegółowy

Autor

Mateusz Chudy

2.10 Dokumentacja pliku Font.c

Funkcje

- void `FontDisplayChar` (char character, uint16_t colorFront, uint16_t colorBackground, uint8_t x, uint16_t y)
- void `FontDisplayString` (char *string, uint16_t colorFront, uint16_t colorBackground, uint8_t x, uint16_t y)

Zmienne

- const uint8_t `ascii` [] `PROGMEM`

2.10.1 Opis szczegółowy

Autor

Mateusz Chudy

2.10.2 Dokumentacja funkcji

2.10.2.1 FontDisplayChar()

```
void FontDisplayChar (  
    char character,  
    uint16_t colorFront,  
    uint16_t colorBackground,  
    uint8_t x,  
    uint16_t y )
```

Wyswietla znak.

Parametry

<i>character</i>	- wyswietlany znak
<i>colorFront</i>	- kolor czcionki
<i>colorBackground</i>	- kolor tła
<i>x</i>	- wspolrzeczna x lewego gornego konca znaku
<i>y</i>	- wspolrzeczna y lewego gornego konca znaku

2.10.2.2 FontDisplayString()

```
void FontDisplayString (  
    char * string,  
    uint16_t colorFront,  
    uint16_t colorBackground,
```



```
uint8_t x,  
uint16_t y )
```

Wyswietla napis.

Parametry

<i>string</i>	- wyświetlany napis
<i>colorFront</i>	- kolor czcionki
<i>colorBackground</i>	- kolor tła
<i>x</i>	- współrzędna x lewego górnego końca napisu
<i>y</i>	- współrzędna y lewego górnego końca napisu

2.11 Dokumentacja pliku Font.h

Definicje

- `#define CHAR_MAX_X 8`
- `#define CHAR_MAX_Y 16`
- `#define CHAR_MASK 0x80`
- `#define STRING_MAX_X 230`
- `#define STRING_MAX_Y 300`

Funkcje

- void [FontDisplayChar](#) (char character, uint16_t colorFront, uint16_t colorBackground, uint8_t x, uint16_t y)
- void [FontDisplayString](#) (char *string, uint16_t colorFront, uint16_t colorBackground, uint8_t x, uint16_t y)

Zmienne

- const uint8_t [ascii](#) []

2.11.1 Opis szczegółowy

Autor

Mateusz Chudy

2.11.2 Dokumentacja funkcji

2.11.2.1 FontDisplayChar()

```
void FontDisplayChar (
    char character,
    uint16_t colorFront,
    uint16_t colorBackground,
    uint8_t x,
    uint16_t y )
```

Wyświetla znak.

Parametry

<i>character</i>	- wyswietlany znak
<i>colorFront</i>	- kolor czcionki
<i>colorBackground</i>	- kolor tła
<i>x</i>	- współrzędna x lewego gornego konca znaku
<i>y</i>	- współrzędna y lewego gornego konca znaku

2.11.2.2 FontDisplayString()

```
void FontDisplayString (
    char * string,
    uint16_t colorFront,
    uint16_t colorBackground,
    uint8_t x,
    uint16_t y )
```

Wyswietla napis.

Parametry

<i>string</i>	- wyswietlany napis
<i>colorFront</i>	- kolor czcionki
<i>colorBackground</i>	- kolor tła
<i>x</i>	- współrzędna x lewego gornego konca napisu
<i>y</i>	- współrzędna y lewego gornego konca napisu

2.12 Dokumentacja pliku GamesMenu.c

Funkcje

- void [GamesMenuShow](#) (void)

2.12.1 Opis szczegółowy

Autor

Mateusz Chudy

2.13 Dokumentacja pliku GamesMenu.h

Definicje

- #define **GAMES_PONG_BUTTON_X1** 0
- #define **GAMES_PONG_BUTTON_Y1** 0
- #define **GAMES_PONG_BUTTON_X2** 240

- `#define GAMES_PONG_BUTTON_Y2 40`
- `#define GAMES_PONG_BUTTON_COLOR DARKBLUE`
- `#define GAMES_PONG_TEXT_COLOR WHITE`
- `#define GAMES_SNAKE_BUTTON_X1 0`
- `#define GAMES_SNAKE_BUTTON_Y1 40`
- `#define GAMES_SNAKE_BUTTON_X2 240`
- `#define GAMES_SNAKE_BUTTON_Y2 80`
- `#define GAMES_SNAKE_BUTTON_COLOR 0X01CA`
- `#define GAMES_SNAKE_TEXT_COLOR WHITE`

Funkcje

- `void GamesMenuShow (void)`

2.13.1 Opis szczegółowy

Autor

Mateusz Chudy

2.14 Dokumentacja pliku GlobalErrors.h

Wyliczenia

- `enum ExecuteResult { EXECUTE_OK = 0, OUT_OF_RANGE = 1, INVALID_PARAMETER = 2 }`

2.14.1 Opis szczegółowy

Autor

Mateusz Chudy

2.15 Dokumentacja pliku GraphicsPrimitives.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- `void GraphicsClearScreen (uint16_t color)`
- `void GraphicsClearArea (uint16_t color, uint8_t x, uint16_t y, uint8_t width, uint16_t height)`
- `void GraphicsSetCursor (uint8_t x, uint16_t y)`
- `void GraphicsPutPixel (uint16_t color, uint8_t x, uint16_t y)`
- `void GraphicsDrawBigDot (uint16_t color, uint8_t x, uint16_t y)`
- `void GraphicsDrawCircle (uint16_t color, uint8_t x, uint16_t y, uint8_t r)`
- `ExecuteResult GraphicsDrawLine (uint16_t color, uint8_t x1, uint16_t y1, uint8_t x2, uint16_t y2)`
- `void GraphicsDrawSimpleLine (uint16_t color, uint8_t x1, uint16_t y1, uint8_t x2, uint16_t y2)`
- `void GraphicsDisplayImage (const uint8_t *image, uint8_t x, uint16_t y, uint8_t width, uint16_t height)`

2.15.1 Opis szczegółowy

Autor

Mateusz Chudy

2.15.2 Dokumentacja funkcji

2.15.2.1 GraphicsClearArea()

```
void GraphicsClearArea (
    uint16_t color,
    uint8_t x,
    uint16_t y,
    uint8_t width,
    uint16_t height )
```

Zamalowuje obszar na zadany kolor.

Parametry

<i>color</i>	- kolor zamalowania obszaru
<i>x</i>	- współrzędna x lewego gornego konca obszaru
<i>y</i>	- współrzędna y lewego gornego konca obszaru
<i>width</i>	- szerokosc obszaru
<i>height</i>	- wysokosc obszaru

2.15.2.2 GraphicsClearScreen()

```
void GraphicsClearScreen (
    uint16_t color )
```

Czysci ekran na zadany kolor.

Parametry

<i>color</i>	- kolor czyszczenia ekranu
--------------	----------------------------

2.15.2.3 GraphicsDisplayImage()

```
void GraphicsDisplayImage (
    const uint8_t * image,
    uint8_t x,
    uint16_t y,
```

```
uint8_t width,  
uint16_t height )
```

Wyswietla obrazek.

Parametry

<i>image</i>	- obrazek
<i>x</i>	- wspolrzeczna x lewego gornego konca obrazka
<i>y</i>	- wspolrzeczna y lewego gornego konca obrazka
<i>width</i>	- szerokosc obrazka
<i>height</i>	- wysokosc obrazka

2.15.2.4 GraphicsDrawBigDot()

```
void GraphicsDrawBigDot (  
    uint16_t color,  
    uint8_t x,  
    uint16_t y )
```

Wyswietla kropke 3x3 piksele o zadanych wspolrzecznych.

Parametry

<i>color</i>	- kolor kropki
<i>x</i>	- wspolrzeczna x
<i>y</i>	- wspolrzeczna y

2.15.2.5 GraphicsDrawCircle()

```
void GraphicsDrawCircle (  
    uint16_t color,  
    uint8_t x,  
    uint16_t y,  
    uint8_t r )
```

Wyswietla okrag o srodku w zadanych wspolrzecznych.

Parametry

<i>color</i>	- kolor kropki
<i>x</i>	- wspolrzeczna x
<i>y</i>	- wspolrzeczna y
<i>r</i>	- promien okregu

2.15.2.6 GraphicsDrawLine()

```
ExecuteResult GraphicsDrawLine (  
    uint16_t color,  
    uint8_t x1,  
    uint16_t y1,  
    uint8_t x2,  
    uint16_t y2 )
```

Wyswietla linie pomiedzy dwoma punktami.

Parametry

<i>color</i>	- kolor lini
<i>x1</i>	- wspolrzeczna x pierwszego punktu
<i>y1</i>	- wspolrzeczna y pierwszego punktu
<i>x2</i>	- wspolrzeczna x drugiego punktu
<i>y2</i>	- wspolrzeczna y drugiego punktu

Zwraca

rezultat operacji

2.15.2.7 GraphicsDrawSimpleLine()

```
void GraphicsDrawSimpleLine (  
    uint16_t color,  
    uint8_t x1,  
    uint16_t y1,  
    uint8_t x2,  
    uint16_t y2 )
```

Wyswietla linie w poziomie lub w pionie.

Parametry

<i>color</i>	- kolor lini
<i>x1</i>	- wspolrzeczna x pierwszego punktu
<i>y1</i>	- wspolrzeczna y pierwszego punktu
<i>x2</i>	- wspolrzeczna x drugiego punktu
<i>y2</i>	- wspolrzeczna y drugiego punktu

2.15.2.8 GraphicsPutPixel()

```
void GraphicsPutPixel (  
    uint16_t color,
```

```
uint8_t x,
uint16_t y )
```

Wyswietla punkt o zadanych wspolrzednych.

Parametry

<i>color</i>	- kolor punktu
<i>x</i>	- wspolrzedna x
<i>y</i>	- wspolrzedna y

2.15.2.9 GraphicsSetCursor()

```
void GraphicsSetCursor (
    uint8_t x,
    uint16_t y )
```

Ustawia kursor w punkcie o zadanych wspolrzednych.

Parametry

<i>x</i>	- wspolrzedna x
<i>y</i>	- wspolrzedna y

2.16 Dokumentacja pliku GraphicsPrimitives.h

Definicje

- #define **MIN_ALPHA_SIZE** -180
- #define **MAX_ALPHA_SIZE** 180

Funkcje

- void [GraphicsClearScreen](#) (uint16_t color)
- void [GraphicsClearArea](#) (uint16_t color, uint8_t x, uint16_t y, uint8_t width, uint16_t height)
- void [GraphicsSetCursor](#) (uint8_t x, uint16_t y)
- void [GraphicsPutPixel](#) (uint16_t color, uint8_t x, uint16_t y)
- void [GraphicsDrawBigDot](#) (uint16_t color, uint8_t x, uint16_t y)
- void [GraphicsDrawCircle](#) (uint16_t color, uint8_t x, uint16_t y, uint8_t r)
- [ExecuteResult](#) [GraphicsDrawLine](#) (uint16_t color, uint8_t x1, uint16_t y1, uint8_t x2, uint16_t y2)
- void [GraphicsDrawSimpleLine](#) (uint16_t color, uint8_t x1, uint16_t y1, uint8_t x2, uint16_t y2)
- [__attribute__\(\(deprecated\)\)](#) void [GraphicsDisplayImage](#)(const uint8_t *image)

Zmienne

- uint8_t **x**
- uint8_t uint16_t **y**
- uint8_t uint16_t uint8_t **width**
- uint8_t uint16_t uint8_t uint16_t **height**

2.16.1 Opis szczegółowy

Autor

Mateusz Chudy

2.16.2 Dokumentacja funkcji

2.16.2.1 GraphicsClearArea()

```
void GraphicsClearArea (
    uint16_t color,
    uint8_t x,
    uint16_t y,
    uint8_t width,
    uint16_t height )
```

Zamalowuje obszar na zadany kolor.

Parametry

<i>color</i>	- kolor zamalowania obszaru
<i>x</i>	- współrzędna x lewego gornego konca obszaru
<i>y</i>	- współrzędna y lewego gornego konca obszaru
<i>width</i>	- szerokosc obszaru
<i>height</i>	- wysokosc obszaru

2.16.2.2 GraphicsClearScreen()

```
void GraphicsClearScreen (
    uint16_t color )
```

Czysci ekran na zadany kolor.

Parametry

<i>color</i>	- kolor czyszczenia ekranu
--------------	----------------------------

2.16.2.3 GraphicsDrawBigDot()

```
void GraphicsDrawBigDot (
    uint16_t color,
    uint8_t x,
    uint16_t y )
```

Wyswietla kropke 3x3 piksele o zadanych wspolrzecznych.

Parametry

<i>color</i>	- kolor kropki
<i>x</i>	- współrzędna x
<i>y</i>	- współrzędna y

2.16.2.4 GraphicsDrawCircle()

```
void GraphicsDrawCircle (
    uint16_t color,
    uint8_t x,
    uint16_t y,
    uint8_t r )
```

Wyswietla okrag o srodku w zadanych wspolrzecznych.

Parametry

<i>color</i>	- kolor kropki
<i>x</i>	- współrzędna x
<i>y</i>	- współrzędna y
<i>r</i>	- promień okregu

2.16.2.5 GraphicsDrawLine()

```
ExecuteResult GraphicsDrawLine (
    uint16_t color,
    uint8_t x1,
    uint16_t y1,
    uint8_t x2,
    uint16_t y2 )
```

Wyswietla linie pomiedzy dwoma punktami.

Parametry

<i>color</i>	- kolor lini
<i>x1</i>	- współrzędna x pierwszego punktu
<i>y1</i>	- współrzędna y pierwszego punktu
<i>x2</i>	- współrzędna x drugiego punktu
<i>y2</i>	- współrzędna y drugiego punktu

Zwraca

rezultat operacji

2.16.2.6 GraphicsDrawSimpleLine()

```
void GraphicsDrawSimpleLine (
    uint16_t color,
    uint8_t x1,
    uint16_t y1,
    uint8_t x2,
    uint16_t y2 )
```

Wyswietla linie w poziomie lub w pionie.

Parametry

<i>color</i>	- kolor lini
<i>x1</i>	- wspolrzeczna x pierwszego punktu
<i>y1</i>	- wspolrzeczna y pierwszego punktu
<i>x2</i>	- wspolrzeczna x drugiego punktu
<i>y2</i>	- wspolrzeczna y drugiego punktu

2.16.2.7 GraphicsPutPixel()

```
void GraphicsPutPixel (
    uint16_t color,
    uint8_t x,
    uint16_t y )
```

Wyswietla punkt o zadanych wspolrzecznych.

Parametry

<i>color</i>	- kolor punktu
<i>x</i>	- wspolrzeczna x
<i>y</i>	- wspolrzeczna y

2.16.2.8 GraphicsSetCursor()

```
void GraphicsSetCursor (
    uint8_t x,
    uint16_t y )
```

Ustawia kursor w punkcie o zadanych wspolrzecznych.

Parametry

<i>x</i>	- wspolrzeczna x
<i>y</i>	- wspolrzeczna y

2.17 Dokumentacja pliku I2CController.c

Funkcje

- void `I2CInit` (void)
- void `I2CStart` (void)
- void `I2CWriteByte` (int8_t byte)
- int8_t `I2CReadByte` (void)
- int8_t `I2CReadLastByte` (void)
- void `I2CStop` (void)

2.17.1 Opis szczegółowy

Autor

Mateusz Chudy

2.17.2 Dokumentacja funkcji

2.17.2.1 I2CReadByte()

```
int8_t I2CReadByte (
    void )
```

Odbiera bajt danych.

Zwraca

odebrany bajt danych

2.17.2.2 I2CReadLastByte()

```
int8_t I2CReadLastByte (
    void )
```

Odbiera ostatni bajt danych.

Zwraca

odebrany ostatni bajt danych

2.17.2.3 I2CWriteByte()

```
void I2CWriteByte (
    int8_t byte )
```

Wysyła bajt danych.

Parametry

<i>byte</i>	- wysyłany bajt danych
-------------	------------------------

2.18 Dokumentacja pliku I2CController.h

Funkcje

- void [I2CInit](#) (void)
- void [I2CStart](#) (void)
- void [I2CWriteByte](#) (int8_t byte)
- int8_t [I2CReadByte](#) (void)
- int8_t [I2CReadLastByte](#) (void)
- void [I2CStop](#) (void)

2.18.1 Opis szczegółowy

Autor

Mateusz Chudy

2.18.2 Dokumentacja funkcji

2.18.2.1 I2CReadByte()

```
int8_t I2CReadByte (  
    void )
```

Odbiera bajt danych.

Zwraca

odebrany bajt danych

2.18.2.2 I2CReadLastByte()

```
int8_t I2CReadLastByte (  
    void )
```

Odbiera ostatni bajt danych.

Zwraca

odebrany ostatni bajt danych

2.18.2.3 I2CWriteByte()

```
void I2CWriteByte (  
    int8_t byte )
```

Wysyła bajt danych.

Parametry

<i>byte</i>	- wysyłany bajt danych
-------------	------------------------

2.19 Dokumentacja pliku LCDController.c**Definicje**

- `#define F_CPU 16000000UL`

Funkcje

- void `LCDPWMInit` (void)
- void `LCDWriteIndex` (uint16_t index)
- void `LCDWriteReg` (uint16_t index, uint16_t data)
- void `LCDWriteData` (uint16_t data)
- void `LCDInit` (void)

2.19.1 Opis szczegółowy**Autor**

Mateusz Chudy

2.19.2 Dokumentacja funkcji**2.19.2.1 LCDWriteData()**

```
void LCDWriteData (  
    uint16_t data )
```

Wpisuje dane.

Parametry

<i>data</i>	- wpisywane dane
-------------	------------------

2.19.2.2 LCDWriteIndex()

```
void LCDWriteIndex (  
    uint16_t index )
```

Wpisuje do sterownika indeks rejestru.

Parametry

<i>index</i>	- indeks rejestru
--------------	-------------------

2.19.2.3 LCDWriteReg()

```
void LCDWriteReg (
    uint16_t index,
    uint16_t data )
```

Wpisuje dane do rejestru.

Parametry

<i>index</i>	- indeks rejestru
<i>data</i>	- wpisywane dane

2.20 Dokumentacja pliku LCDController.h

Definicje

- #define LCD_SIZE_X 240
- #define LCD_SIZE_Y 320
- #define INDEX_START_OSCILLATION 0x000
- #define INDEX_DRIVER_OUTPUT_CONTROL 0x001
- #define INDEX_CRYSTAL_DRIVE_INVERSION_CONTROL 0x002
- #define INDEX_ENTRY_MODE 0x003
- #define INDEX_MOVING_IMAGE_DISPLAY_CONTROL 0x006
- #define INDEX_DISPLAY_CONTROL1 0x007
- #define INDEX_DISPLAY_CONTROL2 0x008
- #define INDEX_DISPLAY_CONTROL3 0x009
- #define INDEX_DISPLAY_CONTROL4 0x00B
- #define INDEX_EXTERNAL_DISPLAY_INTERFACE_CONTROL1 0x00C
- #define INDEX_FRAME_CYCLE_ADJUSTMENT_CONTROL 0x00D
- #define INDEX_SOURCE_DRIVER_INTERFACE_CONTROL1 0x012
- #define INDEX_GATE_DRIVER_INTERFACE_CONTROL1 0x013
- #define INDEX_SOURCE_DRIVER_INTERFACE_CONTROL2 0x018
- #define INDEX_GATE_DRIVER_INTERFACE_CONTROL2 0x019
- #define INDEX_POWER_CONTROL1 0x100
- #define INDEX_POWER_CONTROL2 0x101
- #define INDEX_POWER_CONTROL3 0x102
- #define INDEX_POWER_CONTROL4 0x103
- #define INDEX_POWER_CONTROL5 0x110
- #define INDEX_POWER_CONTROL6 0x111
- #define INDEX_HORIZONTAL_RAM_ADDRESS_SET 0x200
- #define INDEX_VERTICAL_RAM_ADDRESS_SET 0x201
- #define INDEX_RAM_DATA_WRITE 0x202
- #define INDEX_RAM_WRITE_DATA_MASK1 0x203

- #define **INDEX_RAM_WRITE_DATA_MASK2** 0x204
- #define **INDEX_HORIZONTAL_RAM_ADDRESS_START_POSITION** 0x210
- #define **INDEX_HORIZONTAL_RAM_ADDRESS_END_POSITION** 0x211
- #define **INDEX_VERTICAL_RAM_ADDRESS_START_POSITION** 0x212
- #define **INDEX_VERTICAL_RAM_ADDRESS_END_POSITION** 0x213
- #define **INDEX_MOVING_PICTURE_CONTROL_ADDRESS_START_POSITION1** 0x214
- #define **INDEX_MOVING_PICTURE_CONTROL_ADDRESS_END_POSITION1** 0x215
- #define **INDEX_MOVING_PICTURE_CONTROL_ADDRESS_START_POSITION2** 0x216
- #define **INDEX_MOVING_PICTURE_CONTROL_ADDRESS_END_POSITION2** 0x217
- #define **INDEX_GAMMA_CONTROL1** 0x300
- #define **INDEX_GAMMA_CONTROL2** 0x301
- #define **INDEX_GAMMA_CONTROL3** 0x302
- #define **INDEX_GAMMA_CONTROL4** 0x303
- #define **INDEX_GAMMA_CONTROL5** 0x304
- #define **INDEX_GAMMA_CONTROL6** 0x305
- #define **INDEX_GAMMA_CONTROL7** 0x306
- #define **INDEX_GAMMA_CONTROL8** 0x307
- #define **INDEX_GAMMA_CONTROL9** 0x308
- #define **INDEX_GAMMA_CONTROL10** 0x309
- #define **INDEX_LINE_NUMBER_CONTROL** 0x400
- #define **INDEX_SCREEN_CONTROL** 0x401
- #define **INDEX_FULL_SCREEN_IMAGE_RAM_AREA_START_LINE** 0x402
- #define **INDEX_FULL_SCREEN_IMAGE_RAM_AREA_END_LINE** 0x403
- #define **INDEX_VERTICAL_SCROLL_CONTROL** 0x404
- #define **LCD_RST_H()** PORTB |= (1 << PB0);
- #define **LCD_RST_L()** PORTB &= ~(1 << PB0);
- #define **LCD_RS_H()** PORTB |= (1 << PB1);
- #define **LCD_RS_L()** PORTB &= ~(1 << PB1);
- #define **LCD_CS_H()** PORTB |= (1 << PB2);
- #define **LCD_CS_L()** PORTB &= ~(1 << PB2);

Wyliczenia

- enum **Color** {
BLACK = 0X0000, **CYAN** = 0x07FF, **PURPLE** = 0xF81F, **RED** = 0XF800,
GREEN = 0X07E0, **BLUE** = 0X001F, **YELLOW** = 0XFFE0, **ORANGE** = 0XFC08,
GRAY = 0X8430, **LGRAY** = 0XC618, **DARKGRAY** = 0X8410, **PORPO** = 0X801F,
PINK = 0XF81F, **GRAYBLUE** = 0X5458, **LGRAYBLUE** = 0XA651, **DARKBLUE** = 0X01CF,
LIGHTBLUE = 0X7D7C, **WHITE** = 0XFFFF }

Funkcje

- void **LCDPWMInit** (void)
- void **LCDWriteIndex** (uint16_t index)
- void **LCDWriteReg** (uint16_t index, uint16_t data)
- void **LCDWriteData** (uint16_t data)
- void **LCDInit** (void)

2.20.1 Opis szczegółowy

Autor

Mateusz Chudy

2.20.2 Dokumentacja funkcji

2.20.2.1 LCDWriteData()

```
void LCDWriteData (
    uint16_t data )
```

Wpisuje dane.

Parametry

<i>data</i>	- wpisywane dane
-------------	------------------

2.20.2.2 LCDWriteIndex()

```
void LCDWriteIndex (
    uint16_t index )
```

Wpisuje do sterownika indeks rejestru.

Parametry

<i>index</i>	- indeks rejestru
--------------	-------------------

2.20.2.3 LCDWriteReg()

```
void LCDWriteReg (
    uint16_t index,
    uint16_t data )
```

Wpisuje dane do rejestru.

Parametry

<i>index</i>	- indeks rejestru
<i>data</i>	- wpisywane dane

2.21 Dokumentacja pliku Main.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- int [main](#) (void)

2.21.1 Opis szczegółowy

Autor

Mateusz Chudy

2.21.2 Dokumentacja funkcji

2.21.2.1 main()

```
int main (  
        void )
```

Główna funkcja programu.

Zwraca

0

2.22 Dokumentacja pliku MainMenu.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- void [MainMenuShow](#) (void)

2.22.1 Opis szczegółowy

Autor

Mateusz Chudy

2.23 Dokumentacja pliku MainMenu.h

Definicje

- `#define MAIN_GAMES_BUTTON_X1 0`
- `#define MAIN_GAMES_BUTTON_Y1 0`
- `#define MAIN_GAMES_BUTTON_X2 240`
- `#define MAIN_GAMES_BUTTON_Y2 40`
- `#define MAIN_GAMES_BUTTON_COLOR DARKBLUE`
- `#define MAIN_GAMES_TEXT_COLOR WHITE`
- `#define MAIN_APP_BUTTON_X1 0`
- `#define MAIN_APP_BUTTON_Y1 40`
- `#define MAIN_APP_BUTTON_X2 240`
- `#define MAIN_APP_BUTTON_Y2 80`
- `#define MAIN_APP_BUTTON_COLOR 0X01CA`
- `#define MAIN_APP_TEXT_COLOR WHITE`

Funkcje

- `void MainMenuShow (void)`

2.23.1 Opis szczegółowy

Autor

Mateusz Chudy

2.24 Dokumentacja pliku MenuOptions.h

Wyliczenia

- `enum Option {
 DESKTOP = 0, MAIN, GAMES, APPLICATIONS,
 PING_PONG, SNAKE, PAINT, BACK,
 YES, NO, LAST }`

2.24.1 Opis szczegółowy

Autor

Mateusz Chudy

2.25 Dokumentacja pliku Paint.c

Funkcje

- `void PaintInit (Color *buttonColors, struct Button *button)`
- `Color PaintCheckColorChange (Color *buttonColors, Color currentColor)`
- `void PaintShow (void)`

2.25.1 Opis szczegółowy

Autor

Mateusz Chudy

2.25.2 Dokumentacja funkcji

2.25.2.1 PaintCheckColorChange()

```
Color PaintCheckColorChange (
    Color * buttonColors,
    Color currentColor )
```

Sprawdza zmianę koloru przez użytkownika.

Parametry

<i>buttonColors</i>	- tablica dostępnych kolorów rysowania
<i>currentColor</i>	- obecny kolor rysowania

Zwraca

obecnie wybrany kolor

2.25.2.2 PaintInit()

```
void PaintInit (
    Color * buttonColors,
    struct Button * button )
```

Inicjalizuje aplikację Paint.

Parametry

<i>buttonColors</i>	- tablica dostępnych kolorów rysowania
<i>button</i>	- wskaźnik na strukturę reprezentującą przycisk exit

2.26 Dokumentacja pliku Paint.h

Definicje

- #define **PAINT_BACKGROUND_COLOR** WHITE
- #define **PAINT_BUTTONS_SIZE** 30

- `#define PAINT_BUTTONS_SIZE 30`
- `#define PAINT_BUTTONS_MAX_X 8`
- `#define PAINT_EXIT_BUTTON_X 5`
- `#define PAINT_EXIT_BUTTON_Y 285`

Funkcje

- `void PaintInit (Color *buttonColors, struct Button *button)`
- `Color PaintCheckColorChange (Color *buttonColors, Color currentColor)`
- `void PaintShow (void)`

2.26.1 Opis szczegółowy

Autor

Mateusz Chudy

2.26.2 Dokumentacja funkcji

2.26.2.1 PaintCheckColorChange()

```
Color PaintCheckColorChange (  
    Color * buttonColors,  
    Color currentColor )
```

Sprawdza zmianę koloru przez użytkownika.

Parametry

<i>buttonColors</i>	- tablica dostępnych kolorów rysowania
<i>currentColor</i>	- obecny kolor rysowania

Zwraca

obecnie wybrany kolor

2.26.2.2 PaintInit()

```
void PaintInit (  
    Color * buttonColors,  
    struct Button * button )
```

Inicjalizuje aplikację Paint.

Parametry

<i>buttonColors</i>	- tablica dostępnych kolorów rysowania
<i>button</i>	- wskaźnik na strukturę reprezentującą przycisk exit

2.27 Dokumentacja pliku PingPong.c

Funkcje

- bool [CalculateBallPosition](#) (uint8_t *newX, uint16_t *newY, uint8_t lineX, [Vector](#) *moveVector)
- void [CalculateLinePosition](#) (uint8_t *newX)
- void [PingPongPlay](#) (void)

2.27.1 Opis szczegółowy

Autor

Mateusz Chudy

2.27.2 Dokumentacja funkcji

2.27.2.1 CalculateBallPosition()

```
bool CalculateBallPosition (
    uint8_t * newX,
    uint16_t * newY,
    uint8_t lineX,
    Vector * moveVector )
```

Oblicza nowe współrzędne piłki.

Parametry

<i>newX</i>	- nowa wartość współrzędnej x
<i>newY</i>	- nowa wartość współrzędnej y
<i>lineX</i>	- współrzędna x linii
<i>moveVector</i>	- wskaźnik na wektor przesunięcia piłki

2.27.2.2 CalculateLinePosition()

```
void CalculateLinePosition (
    uint8_t * newX )
```

Oblicza nowe współrzędne linii.

Parametry

<code>newX</code>	- nowa wartosc wspolrzednej x lini
-------------------	------------------------------------

2.28 Dokumentacja pliku PingPong.h

Struktury danych

- struct [Vector](#)

Definicje

- `#define PONG_ACCE_SENSITIVITY 10`
- `#define PONG_DISP_SENSITIVITY 1`
- `#define PONG_Y_POS 300`
- `#define PONG_LINE_SIZE 60`
- `#define PONG_BALL_SPEED 20`
- `#define PONG_LINE_SPEED 2`
- `#define PONG_MIN_X 0`
- `#define PONG_MAX_X 240`
- `#define PONG_MIN_Y 0`
- `#define PONG_MAX_Y 320`

Funkcje

- bool [CalculateBallPosition](#) (uint8_t *newX, uint16_t *newY, uint8_t lineX, [Vector](#) *moveVector)
- void [CalculateLinePosition](#) (uint8_t *newX)
- void [PingPongPlay](#) (void)

2.28.1 Opis szczegółowy

Autor

Mateusz Chudy

2.28.2 Dokumentacja funkcji

2.28.2.1 CalculateBallPosition()

```
bool CalculateBallPosition (
    uint8_t * newX,
    uint16_t * newY,
    uint8_t lineX,
    Vector * moveVector )
```

Oblicza nowe wspolrzedne pilki.

Parametry

<i>newX</i>	- nowa wartosc wspolrzednej x
<i>newY</i>	- nowa wartosc wspolrzednej y
<i>lineX</i>	- wspolrzedna x lini
<i>moveVector</i>	- wskaznik na wektor przesuniecie pilki

2.28.2.2 CalculateLinePosition()

```
void CalculateLinePosition (
    uint8_t * newX )
```

Oblicza nowe wspolrzedne lini.

Parametry

<i>newX</i>	- nowa wartosc wspolrzednej x lini
-------------	------------------------------------

2.29 Dokumentacja pliku Question.c

Funkcje

- void [QuestionDisplay](#) (char *string, [Option](#) yes, [Option](#) no)

2.29.1 Opis szczegółowy

Autor

Mateusz Chudy

2.29.2 Dokumentacja funkcji

2.29.2.1 QuestionDisplay()

```
void QuestionDisplay (
    char * string,
    Option yes,
    Option no )
```

Wyswietla pytanie i przyciski odpowiedzi.

Parametry

<i>string</i>	- napis z pytaniem
<i>yes</i>	- opcja do ktorej przechodzi gdy odpowiedz jest pozytywna
<i>no</i>	- opcja do ktorej przechodzi gdy odpowiedz jest negatywna

2.30 Dokumentacja pliku Question.h

Definicje

- `#define QUESTION_BACKGROUND_COLOR GRAY`
- `#define QUESTION_TEXT_COLOR BLACK`
- `#define QUESTION_Y_POS 80`
- `#define ANSWER_Y_POS 160`
- `#define YES_X_POS 30`
- `#define NO_X_POS 150`
- `#define YES_COLOR GREEN`
- `#define NO_COLOR RED`
- `#define QUEST_BUTTON_WIDTH 60`
- `#define QUEST_BUTTON_HEIGHT 40`

Funkcje

- void `QuestionDisplay` (char *string, `Option` yes, `Option` no)

2.30.1 Opis szczegółowy

Autor

Mateusz Chudy

2.30.2 Dokumentacja funkcji

2.30.2.1 QuestionDisplay()

```
void QuestionDisplay (  
    char * string,  
    Option yes,  
    Option no )
```

Wyswietla pytanie i przyciski odpowiedzi.

Parametry

<i>string</i>	- napis z pytaniem
<i>yes</i>	- opcja do ktorej przechodzi gdy odpowiedz jest pozytywna
<i>no</i>	- opcja do ktorej przechodzi gdy odpowiedz jest negatywna

2.31 Dokumentacja pliku Snake.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- `Element** SnakeInit (Element**snakeElements, uint8_t *snakeSize, Element *food)`
- `void SnakePlay (void)`

2.31.1 Opis szczegółowy

Autor

Mateusz Chudy

2.31.2 Dokumentacja funkcji

2.31.2.1 SnakeInit()

```
Element** SnakeInit (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Element * food )
```

Inicjalizuje gre Snake.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza
<i>food</i>	- wskaźnik na element pokarmu

Zwraca

wskaźnik na tablice elementów weza

2.32 Dokumentacja pliku Snake.h

Definicje

- `#define SNAKE_START_SIZE 3`
- `#define SNAKE_START_X 6`
- `#define SNAKE_START_Y 0`
- `#define SNAKE_MAX_X 11`
- `#define SNAKE_MAX_Y 15`
- `#define SNAKE_START_DERECTION DOWN`
- `#define SNAKE_BACKGROUND_COLOR 0x1002`
- `#define SNAKE_COLOR_FILTER 0x4444`
- `#define SNAKE_DELAY 200`

Funkcje

- `Element** SnakeInit (Element **snakeElements, uint8_t *snakeSize, Element *food)`
- `void SnakePlay (void)`

2.32.1 Opis szczegółowy

Autor

Mateusz Chudy

2.32.2 Dokumentacja funkcji

2.32.2.1 SnakeInit()

```
Element** SnakeInit (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Element * food )
```

Inicjalizuje gre Snake.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weży
<i>snakeSize</i>	- wskaźnik na rozmiar weży
<i>food</i>	- wskaźnik na element pokarmu

Zwraca

wskaźnik na tablice elementów weży

2.33 Dokumentacja pliku SnakeElements.c

Funkcje

- `void SnakeDisplayElement (Element *element, bool clear)`
- `void SnakeNewFood (Element **snakeElements, uint8_t snakeSize, Element *food)`
- `Element** SnakeElementAdd (Element **snakeElements, uint8_t *snakeSize, Color color, uint8_t x, uint8_t y, Direction direction)`
- `Element** SnakeDestroy (Element **snakeElements, uint8_t *snakeSize)`

2.33.1 Opis szczegółowy

Autor

Mateusz Chudy

2.33.2 Dokumentacja funkcji

2.33.2.1 SnakeDestroy()

```
Element** SnakeDestroy (
    Element ** snakeElements,
    uint8_t * snakeSize )
```

Niszczy tablice elementow wenza.

Parametry

<i>snakeElements</i>	- wskaznik na tablice elementow weza
<i>snakeSize</i>	- wskaznik na rozmiar weza

Zwraca

wskaznik na tablice elementow weza

2.33.2.2 SnakeDisplayElement()

```
void SnakeDisplayElement (
    Element * element,
    bool clear )
```

Wyswietla element.

Parametry

<i>element</i>	- wskaznik na wyswietlany element
<i>clear</i>	- flaga oznaczajaca czy zamazujemy

2.33.2.3 SnakeElementAdd()

```
Element** SnakeElementAdd (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Color color,
    uint8_t x,
    uint8_t y,
    Direction direction )
```

Dodaje element do wenza.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza
<i>color</i>	- kolor elementu
<i>x</i>	- współrzędna x elementu
<i>y</i>	- współrzędna y elementu
<i>direction</i>	- kierunek ruchu elementu

Zwraca

wskaźnik na tablice elementów weza

2.33.2.4 SnakeNewFood()

```
void SnakeNewFood (
    Element ** snakeElements,
    uint8_t snakeSize,
    Element * food )
```

Tworzy nowy pokarm i rysuje go.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- rozmiar weza
<i>food</i>	- wskaźnik na element pokarmu

2.34 Dokumentacja pliku SnakeElements.h

Struktury danych

- struct [Element](#)

Definicje

- #define **ELEMENT_SIZE** 20
- #define **CHANGE_SIZE_FREQUENCY** 100

Funkcje

- void [SnakeDisplayElement](#) ([Element](#) *element, bool clear)
- void [SnakeNewFood](#) ([Element](#) **snakeElements, uint8_t snakeSize, [Element](#) *food)
- [Element](#) ** [SnakeElementAdd](#) ([Element](#) **snakeElements, uint8_t *snakeSize, [Color](#) color, uint8_t x, uint8_t y, [Direction](#) direction)
- [Element](#) ** [SnakeDestroy](#) ([Element](#) **snakeElements, uint8_t *snakeSize)

2.34.1 Opis szczegółowy

Autor

Mateusz Chudy

2.34.2 Dokumentacja funkcji

2.34.2.1 SnakeDestroy()

```
Element** SnakeDestroy (
    Element ** snakeElements,
    uint8_t * snakeSize )
```

Niszczy tablice elementow wenza.

Parametry

<i>snakeElements</i>	- wskaznik na tablice elementow weza
<i>snakeSize</i>	- wskaznik na rozmiar weza

Zwraca

wskaznik na tablice elementow weza

2.34.2.2 SnakeDisplayElement()

```
void SnakeDisplayElement (
    Element * element,
    bool clear )
```

Wyswietla element.

Parametry

<i>element</i>	- wskaznik na wyswietlany element
<i>clear</i>	- flaga oznaczajaca czy zamazujemy

2.34.2.3 SnakeElementAdd()

```
Element** SnakeElementAdd (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Color color,
```

```
uint8_t x,
uint8_t y,
Direction direction )
```

Dodaje element do węża.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów węża
<i>snakeSize</i>	- wskaźnik na rozmiar węża
<i>color</i>	- kolor elementu
<i>x</i>	- współrzędna x elementu
<i>y</i>	- współrzędna y elementu
<i>direction</i>	- kierunek ruchu elementu

Zwraca

wskaźnik na tablice elementów węża

2.34.2.4 SnakeNewFood()

```
void SnakeNewFood (
    Element ** snakeElements,
    uint8_t snakeSize,
    Element * food )
```

Tworzy nowy pokarm i rysuje go.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów węża
<i>snakeSize</i>	- rozmiar węża
<i>food</i>	- wskaźnik na element pokarmu

2.35 Dokumentacja pliku SnakeMove.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- `uint8_t SnakeCheckCollision (Element **snakeElements, uint8_t snakeSize, uint8_t isFall)`
- `Element ** SnakeCheckChangeDirection (Element **snakeElements, uint8_t *snakeSize)`
- `Element ** SnakeCheckFood (Element **snakeElements, uint8_t *snakeSize, Element *food)`
- `Element ** SnakeUpdateDirections (Element **snakeElements, uint8_t *snakeSize)`
- `Element ** SnakeMove (Element **snakeElements, uint8_t *snakeSize, Element *food)`

2.35.1 Opis szczegółowy

Autor

Mateusz Chudy

2.35.2 Dokumentacja funkcji

2.35.2.1 SnakeCheckChangeDirection()

```
Element** SnakeCheckChangeDirection (
    Element ** snakeElements,
    uint8_t * snakeSize )
```

Sprawdza i zmienia kierunek weza.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza

Zwraca

wskaźnik na tablice elementów weza

2.35.2.2 SnakeCheckCollision()

```
uint8_t SnakeCheckCollision (
    Element ** snakeElements,
    uint8_t snakeSize,
    uint8_t isFall )
```

Sprawdza kolizję weza.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- rozmiar weza
<i>isFall</i>	- flaga wskazująca czy w grze obecne są ściany

Zwraca

wartość logiczna wskazująca na kolizję lub jej brak

2.35.2.3 SnakeCheckFood()

```
Element** SnakeCheckFood (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Element * food )
```

Sprawdza czy waz natrafil na pokarm, jesli tak rysuje nowy i zwieksza wenza.

Parametry

<i>snakeElements</i>	- wskaznik na tablice elementow weza
<i>snakeSize</i>	- wskaznik na rozmiar weza
<i>food</i>	- wskaznik na element pokarmu

Zwraca

wskaznik na tablice elementow weza

2.35.2.4 SnakeMove()

```
Element** SnakeMove (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Element * food )
```

Przemieszcza wenza na ekranie.

Parametry

<i>snakeElements</i>	- wskaznik na tablice elementow weza
<i>snakeSize</i>	- wskaznik na rozmiar weza
<i>food</i>	- wskaznik na element pokarmu

Zwraca

wskaznik na tablice elementow weza

2.35.2.5 SnakeUpdateDirections()

```
Element** SnakeUpdateDirections (
    Element ** snakeElements,
    uint8_t * snakeSize )
```

Aktualizuje kierunki dla elementow wenza.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza

Zwraca

wskaźnik na tablice elementów weza

2.36 Dokumentacja pliku SnakeMove.h

Funkcje

- `uint8_t SnakeCheckCollision (Element **snakeElements, uint8_t snakeSize, uint8_t isFall)`
- `Element ** SnakeCheckChangeDirection (Element **snakeElements, uint8_t *snakeSize)`
- `Element ** SnakeCheckFood (Element **snakeElements, uint8_t *snakeSize, Element *food)`
- `Element ** SnakeUpdateDirections (Element **snakeElements, uint8_t *snakeSize)`
- `Element ** SnakeMove (Element **snakeElements, uint8_t *snakeSize, Element *food)`

2.36.1 Opis szczegółowy

Autor

Mateusz Chudy

2.36.2 Dokumentacja funkcji

2.36.2.1 SnakeCheckChangeDirection()

```
Element** SnakeCheckChangeDirection (  
    Element ** snakeElements,  
    uint8_t * snakeSize )
```

Sprawdza i zmienia kierunek węża.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza

Zwraca

wskaźnik na tablice elementów weza

2.36.2.2 SnakeCheckCollision()

```
uint8_t SnakeCheckCollision (
    Element ** snakeElements,
    uint8_t snakeSize,
    uint8_t isFall )
```

Sprawdza kolizje weza.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- rozmiar weza
<i>isFall</i>	- flaga wskazująca czy w grze obecne są ściany

Zwraca

wartość logiczna wskazująca na kolizję lub jej brak

2.36.2.3 SnakeCheckFood()

```
Element** SnakeCheckFood (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Element * food )
```

Sprawdza czy wąż natrafił na pokarm, jeśli tak rysuje nowy i zwiększa węża.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza
<i>food</i>	- wskaźnik na element pokarmu

Zwraca

wskaźnik na tablice elementów weza

2.36.2.4 SnakeMove()

```
Element** SnakeMove (
    Element ** snakeElements,
    uint8_t * snakeSize,
    Element * food )
```

Przemieszcza węża na ekranie.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza
<i>food</i>	- wskaźnik na element pokarmu

Zwraca

wskaźnik na tablice elementów weza

2.36.2.5 SnakeUpdateDirections()

```
Element** SnakeUpdateDirections (
    Element ** snakeElements,
    uint8_t * snakeSize )
```

Aktualizuje kierunki dla elementów węża.

Parametry

<i>snakeElements</i>	- wskaźnik na tablice elementów weza
<i>snakeSize</i>	- wskaźnik na rozmiar weza

Zwraca

wskaźnik na tablice elementów weza

2.37 Dokumentacja pliku SPIController.c**Funkcje**

- void [SPIInit](#) (void)
- uint8_t [SPICommunication](#) (uint8_t data)

2.37.1 Opis szczegółowy**Autor**

Mateusz Chudy

2.37.2 Dokumentacja funkcji**2.37.2.1 SPICommunication()**

```
uint8_t SPICommunication (
    uint8_t data )
```

Wysyła / odbiera bajt danych z magistrali SPI.

Parametry

<i>data</i>	- wysyłany bajt danych
-------------	------------------------

Zwraca

odebrany bajt danych

2.38 Dokumentacja pliku SPIController.h

Funkcje

- void [SPIInit](#) (void)
- uint8_t [SPICommunication](#) (uint8_t data)

2.38.1 Opis szczegółowy

Autor

Mateusz Chudy

2.38.2 Dokumentacja funkcji

2.38.2.1 SPICommunication()

```
uint8_t SPICommunication (  
    uint8_t data )
```

Wysyła / odbiera bajt danych z magistrali SPI.

Parametry

<i>data</i>	- wysyłany bajt danych
-------------	------------------------

Zwraca

odebrany bajt danych

2.39 Dokumentacja pliku ToolSet.c

Funkcje

- uint16_t [ToolsRandom](#) (uint16_t min, uint16_t max)
- void [ToolsSwap](#) (uint16_t *a, uint16_t *b)
- uint32_t [ToolsPow](#) (uint16_t m, uint16_t n)

2.39.1 Opis szczegółowy

Autor

Mateusz Chudy

2.39.2 Dokumentacja funkcji

2.39.2.1 ToolsPow()

```
uint32_t ToolsPow (
    uint16_t m,
    uint16_t n )
```

Realizuje potegowanie.

Parametry

<i>m</i>	- liczba
<i>n</i>	- potega

Zwraca

liczba m do potegi n

2.39.2.2 ToolsRandom()

```
uint16_t ToolsRandom (
    uint16_t min,
    uint16_t max )
```

Generuje liczbe losowa z zadanego przedzialu.

Parametry

<i>min</i>	- ograniczenie dolne przedzialu
<i>max</i>	- ograniczenie gorne przedzialu

Zwraca

wygenerowana liczba losowa

2.39.2.3 ToolsSwap()

```
void ToolsSwap (
    uint16_t * a,
    uint16_t * b )
```

Zamienia dwie zmienne wartosciami.

Parametry

<i>a</i>	- wskaźnik na pierwsza zmienna
<i>b</i>	- wskaźnik na druga zmienna

2.40 Dokumentacja pliku ToolSet.h

Funkcje

- uint16_t [ToolsRandom](#) (uint16_t min, uint16_t max)
- void [ToolsSwap](#) (uint16_t *a, uint16_t *b)
- uint32_t [ToolsPow](#) (uint16_t m, uint16_t n)

2.40.1 Opis szczegółowy

Autor

Mateusz Chudy

2.40.2 Dokumentacja funkcji

2.40.2.1 ToolsPow()

```
uint32_t ToolsPow (
    uint16_t m,
    uint16_t n )
```

Realizuje potegowanie.

Parametry

<i>m</i>	- liczba
<i>n</i>	- potega

Zwraca

liczba m do potegi n

2.40.2.2 ToolsRandom()

```
uint16_t ToolsRandom (
    uint16_t min,
    uint16_t max )
```

Generuje liczbe losowa z zadanego przedzialu.

Parametry

<i>min</i>	- ograniczenie dolne przedzialu
<i>max</i>	- ograniczenie gorne przedzialu

Zwraca

wygenerowana liczba losowa

2.40.2.3 ToolsSwap()

```
void ToolsSwap (
    uint16_t * a,
    uint16_t * b )
```

Zamienia dwie zmienne wartosciami.

Parametry

<i>a</i>	- wskaznik na pierwsza zmienna
<i>b</i>	- wskaznik na druga zmienna

2.41 Dokumentacja pliku TouchController.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- [ISR](#) (INT0_vect)
- void [TouchInit](#) (void)
- uint16_t [TouchGetADC](#) (uint8_t cmdCode)
- uint16_t [TouchGetADCEX](#) (uint8_t cmdCode)
- [ExecuteResult TouchReadADS](#) (uint16_t *xad, uint16_t *yad)
- [ExecuteResult TouchReadADS2](#) (uint32_t *xad, uint32_t *yad)
- void [TouchConvertAdToXy](#) (void)
- [ExecuteResult TouchReadOnce](#) (void)
- [ExecuteResult TouchReadContinue](#) (void)

2.41.1 Opis szczegółowy

Autor

Mateusz Chudy

2.41.2 Dokumentacja funkcji

2.41.2.1 TouchGetADC()

```
uint16_t TouchGetADC (
    uint8_t cmdCode )
```

Pobiera ze sterownika pojedynczy wynik ADC.

Parametry

<i>cmdCode</i>	- komenda wskazująca na os
----------------	----------------------------

Zwraca

wynik konwersji AD

2.41.2.2 TouchGetADCEx()

```
uint16_t TouchGetADCEx (
    uint8_t cmdCode )
```

Pobiera ze sterownika kilka razy wynik ADC i filtruje zwracana wartosc.

Parametry

<i>cmdCode</i>	- komenda wskazująca na os
----------------	----------------------------

Zwraca

przefiltrowany wynik konwersji AD

2.41.2.3 TouchReadADS()

```
ExecuteResult TouchReadADS (
    uint16_t * xad,
    uint16_t * yad )
```

Pobiera ze sterownika przefiltrowane wyniki konwersji AD dla osi x i y.

Parametry

<i>xad</i>	- przefiltrowany wynik konwersji AD dla osi x
<i>yad</i>	- przefiltrowany wynik konwersji AD dla osi y

Zwraca

rezultat operacji

2.41.2.4 TouchReadADS2()

```
ExecuteResult TouchReadADS2 (
    uint32_t * xad,
    uint32_t * yad )
```

Dwukrotnie pobiera ze sterownika przefiltrowane wyniki konwersji AD dla osi x i y.

Wartosc obu odczytów dla porównania, w granicach dopuszczalnego błędu

Parametry

<i>xad</i>	- przefiltrowany wynik konwersji AD dla osi x
<i>yad</i>	- przefiltrowany wynik konwersji AD dla osi y

Zwraca

rezultat operacji

2.41.2.5 TouchReadContinue()

```
ExecuteResult TouchReadContinue (
    void )
```

Pobiera współrzędne dotknięcia w sposób ciągły.

Zwraca

rezultat operacji

2.41.2.6 TouchReadOnce()

```
ExecuteResult TouchReadOnce (
    void )
```

Pobiera jednokrotnie współrzędne dotknięcia.

Zwraca

rezultat operacji

2.42 Dokumentacja pliku TouchController.h

Struktury danych

- struct [TouchPoint](#)

Definicje

- #define **TOUCH_CS_H()** PORTD |= (1 << PD4);
- #define **TOUCH_CS_L()** PORTD &= ~(1 << PD4);
- #define **CMD_READ_X** 0xD0
- #define **CMD_READ_Y** 0x90
- #define **ERR_RANGE** 50
- #define **READ_TIMES** 10
- #define **LOST_VAL** 4
- #define **CORRECT_X** -20
- #define **CORRECT_Y** -2

Funkcje

- [ISR](#) (INT0_vect)
- void [TouchInit](#) (void)
- uint16_t [TouchGetADC](#) (uint8_t cmdCode)
- uint16_t [TouchGetADCEX](#) (uint8_t cmdCode)
- [ExecuteResult](#) [TouchReadADS](#) (uint16_t *xad, uint16_t *yad)
- [ExecuteResult](#) [TouchReadADS2](#) (uint32_t *xad, uint32_t *yad)
- void [TouchConvertAdToXy](#) (void)
- [ExecuteResult](#) [TouchReadOnce](#) (void)
- [ExecuteResult](#) [TouchReadContinue](#) (void)

Zmienne

- [TouchPoint](#) touchPoint
- volatile uint8_t [interruptFlag](#)

2.42.1 Opis szczegółowy

Autor

Mateusz Chudy

2.42.2 Dokumentacja funkcji

2.42.2.1 TouchGetADC()

```
uint16_t TouchGetADC (  
    uint8_t cmdCode )
```

Pobiera ze sterownika pojedynczy wynik ADC.

Parametry

<i>cmdCode</i>	- komenda wskazująca na os
----------------	----------------------------

Zwraca

wynik konwersji AD

2.42.2.2 TouchGetADCEx()

```
uint16_t TouchGetADCEx (
    uint8_t cmdCode )
```

Pobiera ze sterownika kilka razy wynik ADC i filtruje zwracana wartosc.

Parametry

<i>cmdCode</i>	- komenda wskazująca na os
----------------	----------------------------

Zwraca

przefiltrowany wynik konwersji AD

2.42.2.3 TouchReadADS()

```
ExecuteResult TouchReadADS (
    uint16_t * xad,
    uint16_t * yad )
```

Pobiera ze sterownika przefiltrowane wyniki konwersji AD dla osi x i y.

Parametry

<i>xad</i>	- przefiltrowany wynik konwersji AD dla osi x
<i>yad</i>	- przefiltrowany wynik konwersji AD dla osi y

Zwraca

rezultat operacji

2.42.2.4 TouchReadADS2()

```
ExecuteResult TouchReadADS2 (
    uint32_t * xad,
    uint32_t * yad )
```

Dwukrotnie pobiera ze sterownika przefiltrowane wyniki konwersji AD dla osi x i y.

Wartosc obu odczytów dla porownania, w granicach dopuszczalnego błędu

Parametry

<i>xad</i>	- przefiltrowany wynik konwersji AD dla osi x
<i>yad</i>	- przefiltrowany wynik konwersji AD dla osi y

Zwraca

rezultat operacji

2.42.2.5 TouchReadContinue()

```
ExecuteResult TouchReadContinue (  
    void )
```

Pobiera współrzędne dotknięcia w sposób ciągły.

Zwraca

rezultat operacji

2.42.2.6 TouchReadOnce()

```
ExecuteResult TouchReadOnce (  
    void )
```

Pobiera jednokrotnie współrzędne dotknięcia.

Zwraca

rezultat operacji

2.43 Dokumentacja pliku Welcome.c

Definicje

- `#define F_CPU 16000000UL`

Funkcje

- void `WelcomeShow` (void)

2.43.1 Opis szczegółowy

Autor

Mateusz Chudy

2.44 Dokumentacja pliku Welcome.h

Definicje

- `#define WELCOME_X_POS 88`
- `#define WELCOME_Y_POS 140`
- `#define WELCOME_STRING_COLOR WHITE`
- `#define WELCOME_TIME 2000`

Funkcje

- `void WelcomeShow(void)`

2.44.1 Opis szczegółowy

Autor

Mateusz Chudy

Skorowidz

AccelerometerController.c, [3](#)
 AccelerometerGetAngle, [3](#)
 AccelerometerGetDirection, [3](#)
AccelerometerController.h, [4](#)
 AccelerometerGetAngle, [4](#)
 AccelerometerGetDirection, [4](#)
AccelerometerGetAngle
 AccelerometerController.c, [3](#)
 AccelerometerController.h, [4](#)
AccelerometerGetDirection
 AccelerometerController.c, [3](#)
 AccelerometerController.h, [4](#)
ApplicationMenu.c, [5](#)
ApplicationMenu.h, [5](#)

Button, [2](#)
Button.c, [6](#)
 ButtonCheck, [6](#)
 ButtonCreate, [6](#)
 ButtonDestroy, [7](#)
 ButtonDisplay, [7](#)
Button.h, [8](#)
 ButtonCheck, [8](#)
 ButtonCreate, [8](#)
 ButtonDestroy, [9](#)
 ButtonDisplay, [9](#)
ButtonCheck
 Button.c, [6](#)
 Button.h, [8](#)
ButtonCreate
 Button.c, [6](#)
 Button.h, [8](#)
ButtonDestroy
 Button.c, [7](#)
 Button.h, [9](#)
ButtonDisplay
 Button.c, [7](#)
 Button.h, [9](#)

CalculateBallPosition
 PingPong.c, [34](#)
 PingPong.h, [35](#)
CalculateLinePosition
 PingPong.c, [34](#)
 PingPong.h, [36](#)

Desktop.c, [10](#)
 DesktopInvokeShowFunction, [10](#)
Desktop.h, [10](#)
 DesktopInvokeShowFunction, [11](#)
DesktopInvokeShowFunction
 Desktop.c, [10](#)
 Desktop.h, [11](#)
DisplayTexts.h, [11](#)

Element, [2](#)

Font.c, [12](#)
 FontDisplayChar, [12](#)
 FontDisplayString, [12](#)
Font.h, [14](#)
 FontDisplayChar, [14](#)
 FontDisplayString, [15](#)
FontDisplayChar
 Font.c, [12](#)
 Font.h, [14](#)
FontDisplayString
 Font.c, [12](#)
 Font.h, [15](#)

GamesMenu.c, [15](#)
GamesMenu.h, [15](#)
GlobalErrors.h, [16](#)
GraphicsClearArea
 GraphicsPrimitives.c, [17](#)
 GraphicsPrimitives.h, [21](#)
GraphicsClearScreen
 GraphicsPrimitives.c, [17](#)
 GraphicsPrimitives.h, [21](#)
GraphicsDisplayImage
 GraphicsPrimitives.c, [17](#)
GraphicsDrawBigDot
 GraphicsPrimitives.c, [18](#)
 GraphicsPrimitives.h, [21](#)
GraphicsDrawCircle
 GraphicsPrimitives.c, [18](#)
 GraphicsPrimitives.h, [22](#)
GraphicsDrawLine
 GraphicsPrimitives.c, [18](#)
 GraphicsPrimitives.h, [22](#)
GraphicsDrawSimpleLine
 GraphicsPrimitives.c, [19](#)
 GraphicsPrimitives.h, [22](#)
GraphicsPrimitives.c, [16](#)
 GraphicsClearArea, [17](#)
 GraphicsClearScreen, [17](#)
 GraphicsDisplayImage, [17](#)
 GraphicsDrawBigDot, [18](#)
 GraphicsDrawCircle, [18](#)
 GraphicsDrawLine, [18](#)
 GraphicsDrawSimpleLine, [19](#)
 GraphicsPutPixel, [19](#)
 GraphicsSetCursor, [20](#)
GraphicsPrimitives.h, [20](#)
 GraphicsClearArea, [21](#)
 GraphicsClearScreen, [21](#)
 GraphicsDrawBigDot, [21](#)
 GraphicsDrawCircle, [22](#)
 GraphicsDrawLine, [22](#)
 GraphicsDrawSimpleLine, [22](#)
 GraphicsPutPixel, [23](#)
 GraphicsSetCursor, [23](#)

GraphicsPutPixel
 GraphicsPrimitives.c, 19
 GraphicsPrimitives.h, 23
 GraphicsSetCursor
 GraphicsPrimitives.c, 20
 GraphicsPrimitives.h, 23

 I2CController.c, 24
 I2CReadByte, 24
 I2CReadLastByte, 24
 I2CWriteByte, 24
 I2CController.h, 25
 I2CReadByte, 25
 I2CReadLastByte, 25
 I2CWriteByte, 25
 I2CReadByte
 I2CController.c, 24
 I2CController.h, 25
 I2CReadLastByte
 I2CController.c, 24
 I2CController.h, 25
 I2CWriteByte
 I2CController.c, 24
 I2CController.h, 25

 LCDController.c, 26
 LCDWriteData, 26
 LCDWriteIndex, 26
 LCDWriteReg, 27
 LCDController.h, 27
 LCDWriteData, 29
 LCDWriteIndex, 29
 LCDWriteReg, 29
 LCDWriteData
 LCDController.c, 26
 LCDController.h, 29
 LCDWriteIndex
 LCDController.c, 26
 LCDController.h, 29
 LCDWriteReg
 LCDController.c, 27
 LCDController.h, 29

 main
 Main.c, 30
 Main.c, 29
 main, 30
 MainMenu.c, 30
 MainMenu.h, 31
 MenuOptions.h, 31

 Paint.c, 31
 PaintCheckColorChange, 32
 PaintInit, 32
 Paint.h, 32
 PaintCheckColorChange, 33
 PaintInit, 33
 PaintCheckColorChange
 Paint.c, 32
 Paint.h, 33
 PaintInit
 Paint.c, 32
 Paint.h, 33
 PingPong.c, 34
 CalculateBallPosition, 34
 CalculateLinePosition, 34
 PingPong.h, 35
 CalculateBallPosition, 35
 CalculateLinePosition, 36

 Question.c, 36
 QuestionDisplay, 36
 Question.h, 37
 QuestionDisplay, 37
 QuestionDisplay
 Question.c, 36
 Question.h, 37

 SPICommunication
 SPIController.c, 48
 SPIController.h, 49
 SPIController.c, 48
 SPICommunication, 48
 SPIController.h, 49
 SPICommunication, 49
 Snake.c, 37
 SnakeInit, 38
 Snake.h, 38
 SnakeInit, 39
 SnakeCheckChangeDirection
 SnakeMove.c, 44
 SnakeMove.h, 46
 SnakeCheckCollision
 SnakeMove.c, 44
 SnakeMove.h, 46
 SnakeCheckFood
 SnakeMove.c, 44
 SnakeMove.h, 47
 SnakeDestroy
 SnakeElements.c, 40
 SnakeElements.h, 42
 SnakeDisplayElement
 SnakeElements.c, 40
 SnakeElements.h, 42
 SnakeElementAdd
 SnakeElements.c, 40
 SnakeElements.h, 42
 SnakeElements.c, 39
 SnakeDestroy, 40
 SnakeDisplayElement, 40
 SnakeElementAdd, 40
 SnakeNewFood, 41
 SnakeElements.h, 41
 SnakeDestroy, 42
 SnakeDisplayElement, 42
 SnakeElementAdd, 42
 SnakeNewFood, 43
 SnakeInit

- Snake.c, [38](#)
- Snake.h, [39](#)
- SnakeMove
 - SnakeMove.c, [45](#)
 - SnakeMove.h, [47](#)
- SnakeMove.c, [43](#)
 - SnakeCheckChangeDirection, [44](#)
 - SnakeCheckCollision, [44](#)
 - SnakeCheckFood, [44](#)
 - SnakeMove, [45](#)
 - SnakeUpdateDirections, [45](#)
- SnakeMove.h, [46](#)
 - SnakeCheckChangeDirection, [46](#)
 - SnakeCheckCollision, [46](#)
 - SnakeCheckFood, [47](#)
 - SnakeMove, [47](#)
 - SnakeUpdateDirections, [48](#)
- SnakeNewFood
 - SnakeElements.c, [41](#)
 - SnakeElements.h, [43](#)
- SnakeUpdateDirections
 - SnakeMove.c, [45](#)
 - SnakeMove.h, [48](#)
- ToolSet.c, [49](#)
 - ToolsPow, [50](#)
 - ToolsRandom, [50](#)
 - ToolsSwap, [50](#)
- ToolSet.h, [51](#)
 - ToolsPow, [51](#)
 - ToolsRandom, [51](#)
 - ToolsSwap, [52](#)
- ToolsPow
 - ToolSet.c, [50](#)
 - ToolSet.h, [51](#)
- ToolsRandom
 - ToolSet.c, [50](#)
 - ToolSet.h, [51](#)
- ToolsSwap
 - ToolSet.c, [50](#)
 - ToolSet.h, [52](#)
- TouchController.c, [52](#)
 - TouchGetADCEX, [53](#)
 - TouchGetADC, [53](#)
 - TouchReadADS2, [54](#)
 - TouchReadADS, [53](#)
 - TouchReadContinue, [54](#)
 - TouchReadOnce, [54](#)
- TouchController.h, [55](#)
 - TouchGetADCEX, [56](#)
 - TouchGetADC, [55](#)
 - TouchReadADS2, [56](#)
 - TouchReadADS, [56](#)
 - TouchReadContinue, [57](#)
 - TouchReadOnce, [57](#)
- TouchGetADCEX
 - TouchController.c, [53](#)
 - TouchController.h, [56](#)
- TouchGetADC
 - TouchController.c, [53](#)
 - TouchController.h, [56](#)
- TouchController.c, [53](#)
- TouchController.h, [55](#)
- TouchPoint, [2](#)
- TouchReadADS2
 - TouchController.c, [54](#)
 - TouchController.h, [56](#)
- TouchReadADS
 - TouchController.c, [53](#)
 - TouchController.h, [56](#)
- TouchReadContinue
 - TouchController.c, [54](#)
 - TouchController.h, [57](#)
- TouchReadOnce
 - TouchController.c, [54](#)
 - TouchController.h, [57](#)
- Vector, [2](#)
- Welcome.c, [57](#)
- Welcome.h, [58](#)