

Код реализует меню для микроконтроллера Atmega16 с использованием LCD-дисплея и интерфейса для управления через джойстик и кнопки. Ниже описание основных функций и принципов работы программы:

**Основное назначение**

Программа создает интерактивное меню на LCD-дисплее, где пользователь может:

1. Выбирать скорость (speed) мигания светодиодов.
2. Выбирать паттерн (pattern) для управления светодиодами.
3. Просматривать авторскую подпись.
4. Отображать специальный символ (сердечко).

Меню состоит из **главного уровня** и **вложенного уровня**:

* В главном уровне находятся 4 пункта.
* Во вложенном уровне каждое действие зависит от выбранного пункта.

**Описание функций**

**1. Инициализация**

* **lcd\_init()**: Настраивает LCD-дисплей (режим работы, очистка, настройки курсора).
* **adc\_init()**: Инициализирует АЦП (Analog-to-Digital Converter) для считывания данных с джойстика.
* **button\_init()**: Настраивает кнопки с внешними прерываниями.
* **timer\_init()**: Настраивает таймер для управления миганием светодиодов.

**2. Обработка джойстика и кнопок**

* **Джойстик**:
  + Перемещение вверх/вниз по меню через значения оси **RY**.
  + Вход в подменю/возврат в главное меню через ось **RX**.
* **Кнопки**:
  + **Кнопка 1**: Увеличивает или уменьшает скорость в пункте меню "Select speed".
  + **Кнопка 2**: Переключает паттерн светодиодов в пункте меню "Select pattern".

**3. Работа с меню**

* **displayMenu()**: Формирует текущее состояние главного или вложенного меню на экране LCD.
  + Главное меню:
    - **Select speed**: Изменение скорости мигания.
    - **Select pattern**: Изменение паттерна светодиодов.
    - **Author**: Отображение авторской подписи.
    - **Heart**: Отображение символа сердечка.
  + Вложенное меню:
    - В каждом пункте подменю выполняется соответствующее действие.

**4. Обработка паттернов и скоростей**

* **updateSpeedAndDisplay()**: Устанавливает текущую скорость и обновляет таймер.
* **updatePattern()**: Устанавливает выбранный паттерн светодиодов.

**5. Таймер и управление светодиодами**

* **ISR(TIMER1\_OVF\_vect)**:
  + Отвечает за циклическое сдвиговое отображение светодиодного паттерна.
  + Меняет направление сдвига (вправо или влево) через заданное количество шагов.

**6. Дополнительные функции**

* **create\_custom\_symbol()** и **upload\_custom\_symbol()**: Загружают пользовательские символы в CGRAM дисплея.
* **SetAuthorToLcd()**: Выводит авторскую подпись.
* **SetHeardToLcd()**: Отображает символ сердечка.

**Основной цикл**

1. Чтение значений с джойстика (оси RX и RY).
2. Обновление индекса меню или уровня в зависимости от входного сигнала.
3. Обработка действий на основе текущего уровня меню (главное/вложенное).
4. Вызов функции displayMenu() для отображения изменений.

**Ключевые переменные**

* **menuLevel**: Уровень меню (0 – главное, 1 – вложенное).
* **menuIndex**: Индекс текущего пункта меню.
* **pattern**: Текущий паттерн светодиодов.
* **speed**: Текущая скорость анимации светодиодов.
* **INITIAL\_TIMER\_VALUE**: Значение начальной задержки для таймера, зависит от скорости.

**Пример работы**

1. Пользователь перемещает джойстик вниз, выбирая "Select pattern" в главном меню.
2. Нажимает вправо, чтобы войти в подменю.
3. Нажимает **кнопку 2**, переключая паттерны светодиодов.
4. Возвращается в главное меню, передвигаясь влево на джойстике.

Программа создает гибкий интерфейс для управления и позволяет демонстрировать возможности работы с LCD, джойстиком, кнопками и светодиодами.

**Листинг кода**

#define *F\_CPU* 12000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define RS 2

#define RW 1

#define EN 0

int INITIAL\_TIMER\_VALUE = 55000;

*uint8\_t* pattern = 0b11111000;

*uint8\_t* patternNum = 0;

*uint8\_t* direction = 0; // Направление движения змейки

*uint8\_t* step = 0;

*uint8\_t* speed = 1;

*uint8\_t* speedUp = 1;

*uint8\_t* menuLevel = 0; // Уровень меню (0 - главное меню, 1 - вложенное)

*uint8\_t* menuIndex = 1; // Индекс выбранного пункта меню

// Дисплей

void lcd\_com(unsigned char p){

PORTB &= ~(1<<RS);

PORTB |= (1<<EN);

PORTC = p;

*\_delay\_us*(500);

PORTB &=~(1<<EN);

*\_delay\_us*(500);

}

void lcd\_dat(unsigned char p){

PORTB|=(1<<RS)|(1<<EN);

PORTC=p;

*\_delay\_us*(500);

PORTB&=~(1<<EN);

*\_delay\_us*(500);

}

void lcd\_init(void){

DDRB |= (1<<RS)|(1<<RW)|(1<<EN);

PORTB=0x00; // Обнуляем порт B

DDRC=0xFF; // Устанавливаем порт C как порт вывода

PORTC=0x00; // Обнуляем порт C

*\_delay\_us*(500); // Задержка 500 микросекунд

lcd\_com(0x08); // Инициализация дисплея: выключаем дисплей

*\_delay\_us*(500); // Задержка 500 микросекунд

lcd\_com(0x3C); // Установка режима 8 бит данных, 2 строки, 5x8 точек

*\_delay\_us*(500); // Задержка 500 микросекунд

lcd\_com(0x01); // Очистка дисплея

*\_delay\_us*(500); // Задержка 500 микросекунд

lcd\_com(0x06); // Установка направления пути записи, увеличение адреса на 1

*\_delay\_us*(900); // Задержка 900 микросекунд

lcd\_com(0x0C); // Включаем дисплей без курсора

}

void lcd\_string(char \*str){

char data=0;

while(\*str){

data=\*str++;

lcd\_dat(data);

}

}

void create\_custom\_symbol(unsigned char location, unsigned char \*char\_map) {

unsigned char i;

lcd\_com(0x40 + (location \* 8)); // Установка адреса CGRAM для символа

for (i = 0; i < 8; i++) {

lcd\_dat(char\_map[i]); // Запись байтов символа в CGRAM

}

}

void upload\_custom\_symbol() {

// Массив содержащий 8 символов

unsigned char custom\_symbols[8][8] = {

{0b00000, 0b01010, 0b11111, 0b11111, 0b01110, 0b00100, 0b00000, 0b00000},

{0b00000, 0b00000, 0b01001, 0b01010, 0b01100, 0b01010, 0b01001, 0b00000},

{0b00000, 0b00000, 0b10001, 0b10011, 0b10101, 0b11001, 0b10001, 0b00000},

{0b00000, 0b00000, 0b10001, 0b10001, 0b11111, 0b10001, 0b10001, 0b00000},

{0b00100, 0b01110, 0b10101, 0b10101, 0b10101, 0b01110, 0b00100, 0b00000},

{0b11111, 0b10001, 0b10000, 0b11110, 0b10001, 0b10001, 0b11110, 0b00000},

{0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000},

{0b00000, 0b01010, 0b11111, 0b11111, 0b01110, 0b00100, 0b00000, 0b00000}

};

// Загрузка символов в СGRAM

for (int i = 0; i < 8; i++) {

create\_custom\_symbol(i, custom\_symbols[i]);

}

}

// Инициализация ADC

void adc\_init() {

ADMUX = (1 << REFS0) | (1 << ADLAR); // AVCC как опорное напряжение, выравнивание по левому краю

ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Включение ADC, предделитель 128

}

// Функция чтения канала ADC

int read\_ADC(int ch) {

ADMUX = (ADMUX & 0xF0) | (ch & 0x0F); // Выбор канала, сохранение других битов ADMUX

ADCSRA |= (1 << ADSC); // Запуск преобразования

while (ADCSRA & (1 << ADSC)); // Ожидание завершения преобразования

return ADCH; // Возврат старшего байта (8-битный результат)

}

// Инициализация кнопок

void button\_init() {

DDRD &= ~((1 << PD2) | (1 << PD3)); // PD2 и PD3 как входы

PORTD |= (1 << PD2) | (1 << PD3); // Включение подтягивающих резисторов для PD2 и PD3

MCUCR |= (1 << ISC01) | (1 << ISC00); // INT0 (PD2) срабатывает по фронту

MCUCR |= (1 << ISC11) | (1 << ISC10); // INT1 (PD3) срабатывает по фронту

GICR |= (1 << INT0) | (1 << INT1); // Разрешение прерываний INT0 и INT1

}

// Функция для обновления скорости и дисплея

void updateSpeedAndDisplay() {

switch (speed) {

case 1:

INITIAL\_TIMER\_VALUE = 55000;

lcd\_com(0xC0); lcd\_string("speed = 1");

break;

case 2:

INITIAL\_TIMER\_VALUE = 60000;

lcd\_com(0xC0); lcd\_string("speed = 2");

break;

case 3:

INITIAL\_TIMER\_VALUE = 65000;

lcd\_com(0xC0); lcd\_string("speed = 3");

break;

}

}

// Кнопка 1

ISR(INT0\_vect) {

// Кнопка работает только в одном пункте меню

if (!menuLevel || (menuIndex != 0)) { return; }

point:

if (speedUp) {

if (speed < 3) {

speed += 1;

updateSpeedAndDisplay();

} else {

speedUp = 0; // Меняем направление

}

}

if (!speedUp) {

if (speed > 1) {

speed -= 1;

updateSpeedAndDisplay();

} else {

speedUp = 1; // Меняем направление

goto point;

}

}

};

// Функция для обновления паттерна

void updatePattern() {

lcd\_com(0xC0);

switch (patternNum) {

case 0:

pattern = 0b11111000;

lcd\_string("set pattern 1");

break;

case 1:

pattern = 0b00110011;

lcd\_string("set pattern 2");

break;

case 2:

pattern = 0b10101010;

lcd\_string("set pattern 3");

break;

default:

lcd\_string("Invalid pattern");

break;

}

}

// Кнопка 2

ISR(INT1\_vect) {

// Кнопка работает только в одном пункте меню

if (!menuLevel || (menuIndex != 1)) { return; }

// Переключение patternNum по кругу

patternNum++;

if (patternNum > 3) {

patternNum = 1;

}

updatePattern();

};

// Таймер

void timer\_init(void) {

DDRC=0xFF;

TCCR1B|=(1<<CS12)|(1<<CS10);

TIMSK|=(1<<TOIE1);

TCNT1 = INITIAL\_TIMER\_VALUE;

}

// Прерывание таймера

ISR(TIMER1\_OVF\_vect) {

PORTC = pattern;

if (direction == 0) {

pattern = (pattern << 1) | (pattern >> 7); // Сдвиг вправо

} else {

pattern = (pattern >> 1) | (pattern << 7); // Сдвиг влево

}

step++;

if(step > 4) {

direction = !direction;

step = 0;

}

TCNT1 = INITIAL\_TIMER\_VALUE;

}

void SetAuthorToLcd(){

lcd\_com(0x80); // Курсор на начало первой строки

lcd\_dat(75); // К

lcd\_dat(121); // y

lcd\_dat(1); // к

lcd\_dat(2); // и

lcd\_dat(3); // н

lcd\_dat(16); // Пробел

lcd\_dat(72); // Н

lcd\_dat(46); // Точка

lcd\_dat(65); // A

lcd\_dat(46); // Точка

lcd\_com(0xC3); // Курсор с отступом на вторую строку

lcd\_dat(4); // Ф

lcd\_dat(66); // B

lcd\_dat(5); // Б

lcd\_dat(79); // O

lcd\_dat(45); // -

lcd\_dat(48); // 0

lcd\_dat(49); // 1

lcd\_dat(45); // -

lcd\_dat(50); // 2

lcd\_dat(50); // 2

lcd\_dat(16); // Пробел

}

void SetHeardToLcd() {

lcd\_com(0x80);

lcd\_dat(0);

lcd\_dat(0);

lcd\_dat(0);

lcd\_dat(0);

lcd\_com(0xC0);

lcd\_dat(0);

lcd\_dat(0);

lcd\_dat(0);

lcd\_dat(0);

}

// Функция для отображения меню

void displayMenu() {

lcd\_com(0x01); // Очистка экрана

if (!menuLevel) { // Главное меню

lcd\_com(0x80);

switch (menuIndex) {

// Главное меню пункт 1

case 0:

lcd\_string("> Select seed");

lcd\_com(0xC0); lcd\_string(" Select pattern");

break;

// Главное меню пункт 2

case 1:

lcd\_string(" Select seed");

lcd\_com(0xC0); lcd\_string("> Select pattern");

break;

// Главное меню пункт 3

case 2:

lcd\_string("> Author");

lcd\_com(0xC0); lcd\_string(" Heard");

break;

// Главное меню пункт 4

case 3:

lcd\_string(" Author");

lcd\_com(0xC0); lcd\_string("> Heard");

break;

// Обработка ошибок

default:

lcd\_string("Error");

break;

}

} else { // Вложенное меню

lcd\_com(0x80);

switch (menuIndex){

case 0: lcd\_string("Press button 1"); updateSpeedAndDisplay(); break;

case 1: lcd\_string("Press button 2"); updatePattern(); break;

case 2: SetAuthorToLcd(); break;

case 3: SetHeardToLcd(); break;

default: lcd\_string("Error"); break;

}

}

}

int main(void) {

timer\_init();

lcd\_init();

upload\_custom\_symbol();

button\_init();

adc\_init();

sei();

displayMenu(); // Начальное отображение меню

while (1) {

*uint16\_t* RX = read\_ADC(0); // Считываем ось X (перемещение вверх/вниз)

*uint16\_t* RY = read\_ADC(1); // Считываем ось Y (вход/выход)

// Обработка RY (переключение пунктов меню вверх/вниз)

if (RY >= 170) { // Джойстик вверх - перемещение вверх

if (menuIndex > 0) {

menuIndex--;

displayMenu();

*\_delay\_ms*(200);

}

} else if (RY <= 85) { // Джойстик вниз - перемещение вниз

if (menuIndex < 3) { // Четыре пункта в главном меню

menuIndex++;

displayMenu();

*\_delay\_ms*(200);

}

}

// Обработка RX (вход/выход из меню)

if (RX >= 170) { // Джойстик вправо - вход в пункт меню

if (menuLevel == 0) {

menuLevel = 1; // Переход во вложенное меню

displayMenu();

*\_delay\_ms*(200);

}

} else if (RX <= 85) { // Джойстик влево - выход из пункта меню

if (menuLevel == 1) {

menuLevel = 0; // Возврат в главное меню

displayMenu();

*\_delay\_ms*(200);

}

}

*\_delay\_ms*(200); // Задержка между циклами

}

}