Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В.Ф. Уткина»

Кафедра «ЭВМ»

Отчет о практической работе №4

«Вывод информации с использованием дисплея МЭЛТ МТ-12864J»

по дисциплине

«Специализированные ЭВМ»

Выполнили:

Студенты группы 045

Вашкулатов Н.А.

Анохин В.А.

Проверил:

доц. каф. ЭВМ Устюков Д.И.

**Цель работы**: изучить особенности использования жидкокристаллических дисплеев для вывода информации от микроконтроллеров.

**Ход работы**

**Задание 1.1.** Создайте новый проект. Добавьте необходимые библиотеки. Добавьте в main.c код из файла материалов к практическим работам, каталог CortexM3C\Текст\_программ\ПР4LCD\1986ВЕ92QI.c или CortexM3C\ Текст\_программ\ПР4LCD\1986ВЕ93У.c. Выполните сборку программы. Загрузите программу в контроллер. Проанализируйте результаты выполнения.

**Код программы:**

#include "MDR32F9Qx\_port.h" //библиотека работы с портами

#include "MDR32F9Qx\_rst\_clk.h"//библиотека работы с модулем тактирования

#include "mlt\_lcd.h" //библиотека работы с дисплеем

//структура инициализации порта

PORT\_InitTypeDef PortStruct;

void LCDPins (void) //процедура инициализации выводов МК для работы с LCD

{ //все линии цифровые в режиме PORT с медленным фронтом

//линии DB5:DB0 - PA5:PA0 по умолчанию ввод без подтяжек и фильтров

PortStruct.PORT\_Pin = PORT\_Pin\_0|PORT\_Pin\_1|PORT\_Pin\_2|PORT\_Pin\_3|PORT\_Pin\_4|PORT\_Pin\_5;

PortStruct.PORT\_OE = PORT\_OE\_IN;

PortStruct.PORT\_FUNC = PORT\_FUNC\_PORT;

PortStruct.PORT\_SPEED = PORT\_SPEED\_SLOW;

PortStruct.PORT\_MODE = PORT\_MODE\_DIGITAL;

PORT\_Init(MDR\_PORTA,&PortStruct);

//линии DB7:DB6 - PF3:PF2 по умолчанию ввод без подтяжек и фильтров

PortStruct.PORT\_Pin = PORT\_Pin\_2|PORT\_Pin\_3;

PORT\_Init(MDR\_PORTF,&PortStruct);

//линии E1 - PB7, E2 - PB8,RES - PB9, R/~W - PB10 вывод

PortStruct.PORT\_OE = PORT\_OE\_OUT;

PortStruct.PORT\_Pin = PORT\_Pin\_7|PORT\_Pin\_8|PORT\_Pin\_9|PORT\_Pin\_10;

PORT\_Init(MDR\_PORTB,&PortStruct);

//Линии A0 - PC0, E - PC1 вывод

PortStruct.PORT\_Pin = PORT\_Pin\_0|PORT\_Pin\_1;

PORT\_Init(MDR\_PORTC,&PortStruct);

}

void CPUinit (void) //процедура инициализации внешнего тактового генератора для процессора

{

//включение внешнего высокочастотного генератора

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

//запрос состояния, только если готов, то переход к следующим действиям

while (RST\_CLK\_HSEstatus()!=SUCCESS);

//включение модуля PLL

RST\_CLK\_CPU\_PLLcmd(ENABLE);

// умножение частоты кварцевого генератора в 10 раз

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul6);

//запрос состояния PLL, если готов, то переход к следующим действиям

while (RST\_CLK\_CPU\_PLLstatus()!=SUCCESS);

//переключение на частоту сформированную PLL

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);

//переключение на тактовый сигнал CPU\_C3

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

void PCLKinit(void) //процедура тактирования портов

{

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA,ENABLE); //включение тактирования PORTA

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTB,ENABLE); //включение тактирования PORTB

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC,ENABLE); //включение тактирования PORTC

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTF,ENABLE); //включение тактирования PORTF

}

void LCDStart(void) //процедура запуска дисплея

{

uint8\_t status; //переменная для анализа состояния чипа

LcdInit(); //установка сигнала перезагрузки, ожидание завершения операции

do

{

status = ReadStatus(1); //чтение состояния первого чипа

}while ((status&0x80)!=0x00); //пока не готов снова прочитать состояние

do

{

status = ReadStatus(2); //чтение состояния второго чипа

}while((status&0x80)!=0x00); //пока не готов снова прочитать состояние

DispOn (1); //включение чипа 1

DispOn (2); //включение чипа 2

LcdClearChip(1); //очистка чипа 1

LcdClearChip(2); //очистка чипа 2

}

void LCDTestPrint(void)

{

SetPage(1,2); //установить 2 страницу 1-ого чипа

SetAdress(1,5); //установить адрес - 5, для 1-ого чипа, (2 страницы)

WriteData (1,0x20); //вывод на экран точки с положением 0x20 (по адресу 5, 2 страница, чип 1)

SetPage(2,3); //установить 3 страницу 2-ого чипа

SetAdress(2,0); //установить адрес - 0, для 2-ого чипа, (3 страницы)

for (int i=0;i<7;i++) //перебор комбинаций от 0 до 7

{

WriteData(2,i); //вывод на экран точки с положением i, адрес увеличивается на 1 после выполнения каждой команды

}

SetPage(2,4); //установить 4 страницу 2-ого чипа

SetAdress(2,0); //установить адрес - 0, для 2-ого чипа, (4 страницы)

for (int i=0;i<63;i++)//перебор комбинаций от 0 до 63

{

WriteData(2,i); //вывод на экран точки с положением i, адрес увеличивается на 1 после выполнения каждой команды

}

}

int main (void) //точка входа в программу

{

CPUinit(); //настройка тактирования процессора

PCLKinit(); //включение тактовых сигналов

LCDPins(); //инициализация выводов

LCDStart(); //запуск LCD

LCDTestPrint(); //вывод тестовых значений на экран

while(1) //бесконечный цикл

{ }

}

**Задание 1.2.** Разработайте подпрограмму, которая преобразует координаты из диапазона x∈[0,127], y∈[0,63] в координаты модуля ЖКИ. Например, точка [0, 0] соответствует значению 0x01, на нулевой странице первого чипа.

**Код программы:**

void setPixel(uint8\_t x, uint8\_t y, uint8\_t data){

uint8\_t chip = (x / 64) + 1; // Определяем номер чипа на основе координаты x

uint8\_t page = y / 8; // Определяем номер страницы на основе координаты y

uint8\_t address = x % 64; // Определяем адрес (колонку) на основе координаты x

SetPage(chip,page);

SetAdress(chip, address);

ReadData(chip);

uint8\_t mask = 1 << (y % 8);

uint8\_t cellData = ReadData(chip);

if (data) {

cellData |= mask; // Установка бита в 1

} else {

cellData &= ~mask; // Сброс бита в 0

}

SetPage(chip,page);

SetAdress(chip, address);

// Записываем обновленное значение обратно в ячейку

WriteData(chip, cellData);

}

**Задание 1.3.** Используя подпрограмму из задания 2, разработайте программу, для отображения на экран координатных осей, которые пересекаются в точке [63,31]. Выполните сборку программы. Загрузите программу в контроллер. Проанализируйте результаты выполнения.

В данной программе используются подпрограммы инициализации из задания 1.1 и процедура setPixel из 1.2. Результат выполнения программы показан на рисунке 1.

**Код программы:**

void LCDCoords(){

uint16\_t width = 128;

uint16\_t height = 64;

for(int x = 0; x < width; x++){

setPixel(x, height/2 -1, 1);

}

for(int y = 0; y < height; y++){

setPixel(width/2-1, y, 1);

}

}

int main (void)

{

CPUinit();

PCLKinit();

LCDPins();

LCDStart();

LCDCoords();

while(1){ }

}



Рисунок 1 – Координатные оси

**Задание 1.4.** Используя подпрограмму из задания 2, вывести на экран координатные оси и функцию y= - (30\*sin(x\*3.14/32))+31 (Для использования функции sin необходимо добавить библиотеку math.h).

**Код программы:**

void LCDFunc(){

uint16\_t width = 128;

for(int x = 0; x < width; x++){

int y = -(30\*sin(x\*3.14/32)) + 31;

setPixel(x,y,1);

}

}

int main (void)

{

CPUinit();

PCLKinit();

LCDPins();

LCDStart();

LCDCoords();

LCDFunc();

while(1){ }

}



Рисунок 2 – График функции

**Задание 2.1.** Создайте новый проект аналогично заданию 1 из первой части работы. Дополнительно добавьте к проекту файл lcdstring.c аналогично файлу mlt\_lcd.c. Убедитесь, что файлы stdio.h, lcdstring.h, mlt\_lcd.h, font.h расположены в директории LIBS в проекте и подключите эти заголовочные файлы с использованием директивы #include к файлу main.c. Добавьте функцию Delay и исправьте функцию main в соответствии с примером, представленным ниже. Выполните сборку программы, загрузите программу в контроллер. Проанализируйте полученный результат.

Сверху на левой стороне дисплея находится рисунок. Справа текст «РГРТУ». Под текстом номер группы: «945». По центру экрана непрерывно перемещается надпись «ВРАГИ». Внизу находится надпись: «PI 3.141592».

**Код программы:**

void Delay(uint32\_t wait)

{

for (int i = 0;i < wait;i++);

}

int main(void) //точка входа в программу

{

CPUinit(); //настройка тактирования процессора

PCLKinit(); //включение тактовых сигналов

LCDPins(); //инициализация выводов

LCDStart(); //запуск LCD

LcdClearChip(1);

LcdClearChip(2);

LcdPutImage(icon\_mil, 0, 0, 1, 1);

LcdPutChar(cyr\_R, 11, 0);

LcdPutChar(cyr\_G, 12, 0);

LcdPutChar(cyr\_R, 13, 0);

LcdPutChar(cyr\_T, 14, 0);

LcdPutChar(cyr\_U, 15, 0);

uint8\_t\* group[3] = { {dig\_9},{dig\_4},{dig\_5} };

LcdScrollString(group, 2, 3, 8);

uint8\_t\* str[16] = { {cyr\_V},{cyr\_R},

{cyr\_A},

{cyr\_G},

{cyr\_I},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp},{sym\_sp}

};

char stroka[16];

sprintf(stroka, "%s %f", "Pi ", 3.141592);

PrintString(stroka, 4);

while (1) //бесконечный цикл

{

for (int i = 0;i < 16;i++)

{

LcdScrollString(str, 3, 5, i);

Delay(1000000);

}

}

}

**Задание 2.2.** Измените программу из задания 1 таким образом, чтобы в нижнем правом углу экрана добавился номер бригады и фамилии студентов. Выполните сборку программы, загрузите программу в контроллер. Проанализируйте полученный результат. Зафиксируйте результат в отчёте.

На рисунке 3 показан результат выполнения программы.

**Код программы:**

int main (void) //точка входа в программу

{

CPUinit(); //настройка тактирования процессора

PCLKinit(); //включние тактовых сигналов

LCDPins(); //инициализация выводов

LCDStart(); //запуск LCD

LcdClearChip (1);

LcdClearChip (2);

LcdPutImage(icon\_mil,0,0,1,1);

LcdPutChar (cyr\_R, 11,0);

LcdPutChar (cyr\_G, 12,0);

LcdPutChar (cyr\_R, 13,0);

LcdPutChar (cyr\_T, 14,0);

LcdPutChar (cyr\_U, 15,0);

uint8\_t \*group[3] = {{dig\_0},{dig\_4},{dig\_5}};

LcdScrollString (group, 2, 3, 8);

uint8\_t \*brigada[9] = {{cyr\_B},{cyr\_r} ,{cyr\_i},{cyr\_g}, {cyr\_a},{cyr\_d},{cyr\_a},{sym\_sp}, {dig\_4}};

LcdScrollString (brigada, 5, 9, 8);

uint8\_t \*fio1[10] = {{cyr\_V},{cyr\_a} ,{cyr\_sh},{cyr\_k}, {cyr\_u},{cyr\_l},{cyr\_a},{cyr\_t}, {cyr\_o}, {cyr\_v}};

LcdScrollString (fio1, 6, 9, 9);

uint8\_t \*fio2[6] = {{cyr\_A},{cyr\_n} ,{cyr\_o},{cyr\_kh}, {cyr\_i},{cyr\_n}};

LcdScrollString (fio2, 7, 9, 5);

while(1) //бесконечный цикл

{ }

}

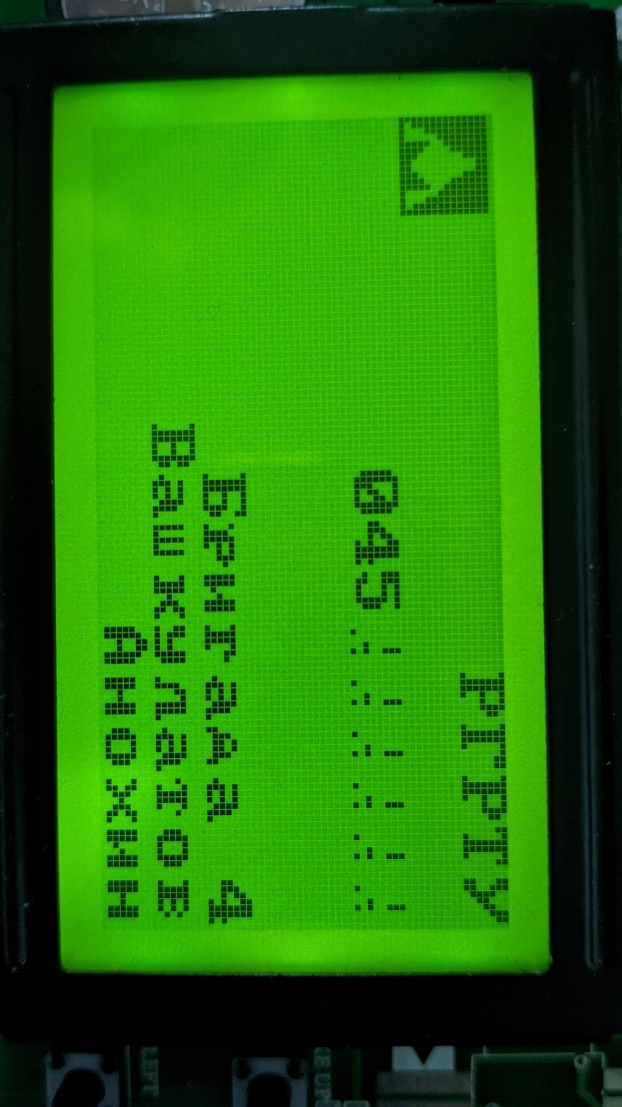


Рисунок 3 – Результат выполнения программы

**Задание 2.3.** Измените программу из задания 1 таким образом, чтобы на экране отображалось меню в следующем виде (рисунок 4).

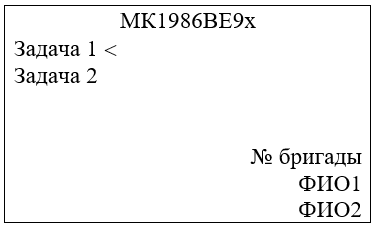


Рисунок 4 – Задание 2.3

Вместо символа «<» используйте курсор из файла font.h. Подключите к проекту кнопки UP и DOWN и реализуйте переходы курсора по меню с использованием этих кнопок. Выполните сборку программы, загрузите программу в контроллер. Проанализируйте полученный результат. Зафиксируйте результат в отчёте.

Результат выполнения программы показан на рисунке 5.

**Код программы:**

int main (void)

{

CPUinit();

PCLKinit();

LCDPins();

LCDStart();

LcdClearChip (1);

LcdClearChip (2);

uint8\_t \*brigada[9] = {{cyr\_B},{cyr\_r} ,{cyr\_i},{cyr\_g}, {cyr\_a},{cyr\_d},{cyr\_a},{sym\_sp}, {dig\_4}};

LcdScrollString (brigada, 5, 9, 8);

uint8\_t \*fio1[10] = {{cyr\_V},{cyr\_a} ,{cyr\_sh},{cyr\_k}, {cyr\_u},{cyr\_l},{cyr\_a},{cyr\_t}, {cyr\_o}, {cyr\_v}};

LcdScrollString (fio1, 6, 9, 9);

uint8\_t \*fio2[6] = {{cyr\_A},{cyr\_n} ,{cyr\_o},{cyr\_kh}, {cyr\_i},{cyr\_n}};

LcdScrollString (fio2, 7, 9, 5);

char model[16];

sprintf(model, " %s ", "MK1986BE92");

PrintString(model,0);

for(int i = 1; i <= 2; i++){

LcdPutChar (cyr\_Z, 0,i);

LcdPutChar (cyr\_a, 1,i);

LcdPutChar (cyr\_d, 2,i);

LcdPutChar (cyr\_a, 3,i);

LcdPutChar (cyr\_ch, 4,i);

LcdPutChar (cyr\_a, 5,i);

LcdPutChar(sym\_sp,6,i);

if(i == 1) LcdPutChar(cursor,7,i);

}

while(1){ }

}

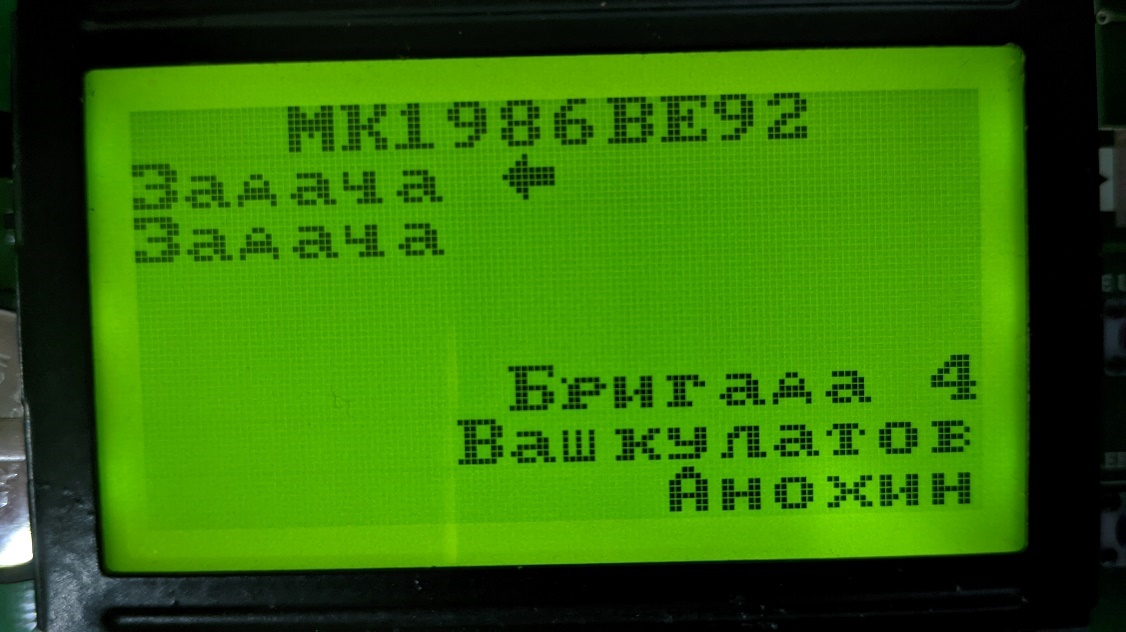


Рисунок 5 – Результат выполнения программы

**Задание 2.4.** Измените программу из задания 3, добавив на каждый пункт меню свою обрабатываемую задачу:

Задача 1 – изменение состояния вывода PORTA6 (PORTC0 для 1986BE93У) на противоположное через 1 секунду с подтверждением текущего состояния на дисплее;

Задача 2 – изменение состояния PORTA7 (PORTF5 для 1986BE93У) на противоположное через 2 секунды с подтверждением текущего состояния на дисплее.

Выполните сборку программы, загрузите программу в контроллер. Подключите логический анализатор к выводам порта А (портов C и F для 1986ВЕ93У) (см. работу 1). Проанализируйте полученный результат. Зафиксируйте результат в отчёте.

На рисунках 6 – 7 показана работа программы.

**Код программы:**

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

#include "mlt\_lcd.h"

#include "font.h"

#include "stdio.h"

#include "lcdstring.h"

uint8\_t width = 16;

uint8\_t height = 8;

struct ButtonStatus {

int UP;

int DOWN;

int SELECT

};

static struct ButtonStatus pressedButtons = {0,0,0};

int selectedMenuItem = 0;

void Delay(uint32\_t wait)

{

for (int i=0;i<wait;i++);

}

PORT\_InitTypeDef PortStruct;

PORT\_InitTypeDef PortInitOutput;

PORT\_InitTypeDef PortInitBtn;

void LCDPins (void)

{

PortStruct.PORT\_Pin = PORT\_Pin\_0|PORT\_Pin\_1|PORT\_Pin\_2|PORT\_Pin\_3|PORT\_Pin\_4|PORT\_Pin\_5;

PortStruct.PORT\_OE = PORT\_OE\_IN;

PortStruct.PORT\_FUNC = PORT\_FUNC\_PORT;

PortStruct.PORT\_SPEED = PORT\_SPEED\_SLOW;

PortStruct.PORT\_MODE = PORT\_MODE\_DIGITAL;

PORT\_Init(MDR\_PORTA,&PortStruct);

PortStruct.PORT\_Pin = PORT\_Pin\_2|PORT\_Pin\_3;

PORT\_Init(MDR\_PORTF,&PortStruct);

PortStruct.PORT\_OE = PORT\_OE\_OUT;

PortStruct.PORT\_Pin = PORT\_Pin\_7|PORT\_Pin\_8|PORT\_Pin\_9|PORT\_Pin\_10;

PORT\_Init(MDR\_PORTB,&PortStruct);

PortStruct.PORT\_Pin = PORT\_Pin\_0|PORT\_Pin\_1;

PORT\_Init(MDR\_PORTC,&PortStruct);

}

void CPUinit (void)

{

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul6);

while (RST\_CLK\_CPU\_PLLstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

void PCLKinit(void)

{

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA,ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTB,ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC,ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTF,ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTE,ENABLE);

}

void LCDStart(void)

{

uint8\_t status;

LcdInit();

do

{

status = ReadStatus(1

}while ((status&0x80)!=0x00);

do

{

status = ReadStatus(2

}while((status&0x80)!=0x00);

DispOn (1);

DispOn (2);

LcdClearChip(1);

LcdClearChip(2);

}

void PrintRight(char \*x, int size, int strnum){

char result[width];

for(int i = width - size; i < width; i++){

result[i] = x[i];

}

PrintString(result, strnum);

}

void ButtonsPinCfg(void){

PORT\_StructInit(&PortInitBtn);

PortInitBtn.PORT\_Pin = PORT\_Pin\_5;

PortInitBtn.PORT\_OE = PORT\_OE\_IN;

PortInitBtn.PORT\_FUNC = PORT\_FUNC\_PORT;

PortInitBtn.PORT\_MODE = PORT\_MODE\_DIGITAL;

PortInitBtn.PORT\_SPEED = PORT\_SPEED\_SLOW;

PortInitBtn.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_OFF;

PortInitBtn.PORT\_PULL\_UP = PORT\_PULL\_UP\_ON;

PortInitBtn.PORT\_GFEN = PORT\_GFEN\_ON;

PORT\_Init(MDR\_PORTB, &PortInitBtn);

PortInitBtn.PORT\_Pin = PORT\_Pin\_1;

PORT\_Init(MDR\_PORTE, &PortInitBtn);

PortInitBtn.PORT\_Pin = PORT\_Pin\_2;

PORT\_Init(MDR\_PORTC, &PortInitBtn);

}

int isUpPressed(){

return PORT\_ReadInputDataBit(MDR\_PORTB, PORT\_Pin\_5) == 0;

}

int isDownPressed(){

return PORT\_ReadInputDataBit(MDR\_PORTE, PORT\_Pin\_1) == 0;

}

int isSelectPressed(){

return PORT\_ReadInputDataBit(MDR\_PORTC, PORT\_Pin\_2) == 0;

}

void changeTask(){

if(selectedMenuItem == 0){

LcdPutChar(sym\_sp,7,2);

LcdPutChar(cursor,7,1);

} else {

LcdPutChar(sym\_sp,7,1);

LcdPutChar(cursor,7,2);

}

}

void SetUpOutput(){

PortInitOutput.PORT\_Pin = PORT\_Pin\_6 | PORT\_Pin\_7;

PortInitOutput.PORT\_OE = PORT\_OE\_OUT;

PortInitOutput.PORT\_FUNC = PORT\_FUNC\_PORT;

PortInitOutput.PORT\_SPEED = PORT\_SPEED\_SLOW;

PortInitOutput.PORT\_MODE = PORT\_MODE\_DIGITAL;

PORT\_Init(MDR\_PORTA,&PortInitOutput);

}

void pressUp(){

selectedMenuItem = (selectedMenuItem + 1)%2;

changeTask();

}

void pressDown(){

selectedMenuItem = (selectedMenuItem -1 + 2)%2;

changeTask();

}

void executeTask(){

uint32\_t pin;

if(selectedMenuItem == 0)

pin = PORT\_Pin\_6;

else pin = PORT\_Pin\_7;

uint8\_t data = PORT\_ReadInputDataBit(MDR\_PORTA, pin);

int str = 3 + selectedMenuItem;

LcdPutChar (lat\_L, 0,str);

LcdPutChar (lat\_D, 1,str);

if(selectedMenuItem == 0){

LcdPutChar (dig\_1, 2,str);

} else {

LcdPutChar (dig\_2, 2,str);

}

Delay(8000000 \* (selectedMenuItem + 1));

PORT\_WriteBit(MDR\_PORTA, pin, data ^ 0x1);

LcdPutChar (lat\_D, 0,str);

LcdPutChar (lat\_N, 1,str);

}

int main (void)

{

CPUinit();

PCLKinit();

LCDPins();

LCDStart();

LcdClearChip (1);

LcdClearChip (2);

ButtonsPinCfg();

SetUpOutput();

uint8\_t \*brigada[9] = {{cyr\_B},{cyr\_r} ,{cyr\_i},{cyr\_g}, {cyr\_a},{cyr\_d},{cyr\_a},{sym\_sp}, {dig\_4}};

LcdScrollString (brigada, 5, 9, 8);

uint8\_t \*fio1[10] = {{cyr\_V},{cyr\_a} ,{cyr\_sh},{cyr\_k}, {cyr\_u},{cyr\_l},{cyr\_a},{cyr\_t}, {cyr\_o}, {cyr\_v}};

LcdScrollString (fio1, 6, 9, 9);

uint8\_t \*fio2[6] = {{cyr\_A},{cyr\_n} ,{cyr\_o},{cyr\_kh}, {cyr\_i},{cyr\_n}};

LcdScrollString (fio2, 7, 9, 5);

char model[16];

sprintf(model, " %s ", "MK1986BE92");

PrintString(model,0);

for(int i = 1; i <= 2; i++){

LcdPutChar (cyr\_Z, 0,i);

LcdPutChar (cyr\_a, 1,i);

LcdPutChar (cyr\_d, 2,i);

LcdPutChar (cyr\_a, 3,i);

LcdPutChar (cyr\_ch, 4,i);

LcdPutChar (cyr\_a, 5,i);

LcdPutChar(sym\_sp,6,i);

if(i == 1) LcdPutChar(cursor,7,i);

}

while(1)

{

if(isSelectPressed()){

if(pressedButtons.SELECT == 0){

executeTask();

pressedButtons.SELECT = 1;

}

} else {

pressedButtons.SELECT = 0;

}

if(isDownPressed()){

if(pressedButtons.DOWN == 0){

pressDown();

pressedButtons.DOWN = 1;

}

} else {

pressedButtons.DOWN = 0;

}

if(isUpPressed()){

if(pressedButtons.UP == 0){

pressUp();

pressedButtons.UP = 1;

}

} else {

pressedButtons.UP = 0;

}

}

}

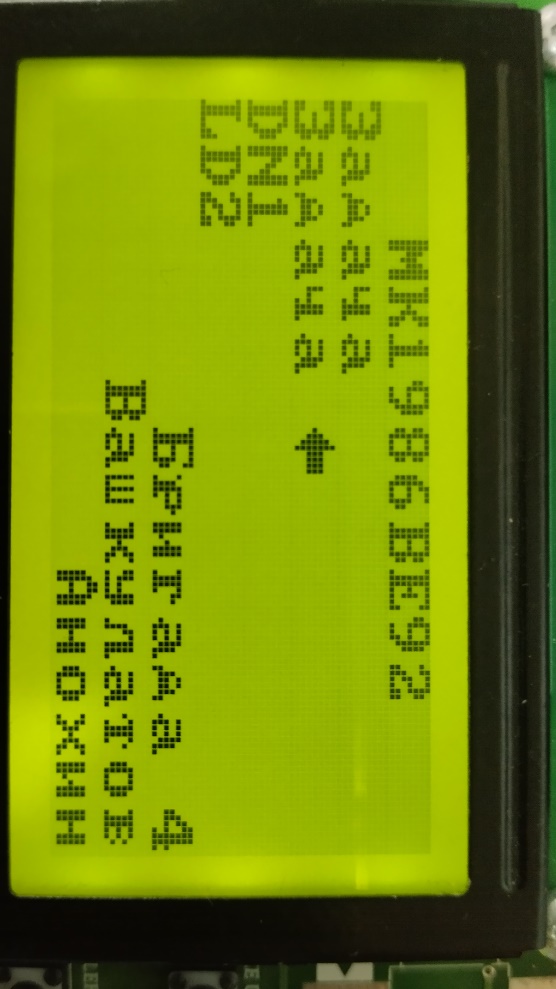


Рисунок 6 – Начало выполнения задачи 2

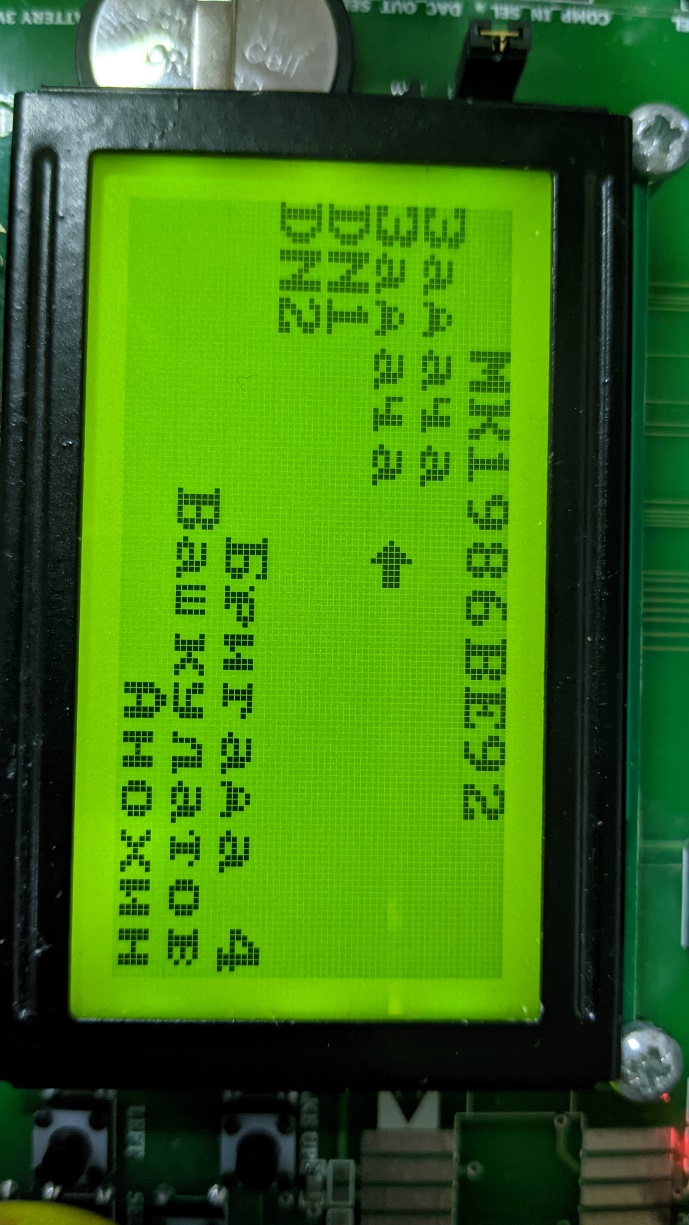


Рисунок 7 – Через 2 секунды после выполнения задачи 2

**Вывод**: в ходе выполнения работы были изучены особенности использования жидкокристаллических дисплеев для вывода информации от микроконтроллеров.