• О проекте
• Обратная связь
• Полезные ссылки
• Полезные программы
• Друзья сайта

Последние комментарии

Алексей: *Библиотека для AtmelStudio 6.x а-ля CodeVisionAVR*
<<WinAVR (2...

Алексей: *Библиотека для AtmelStudio 6.x а-ля CodeVisionAVR*
Изменения внес. К...

Библиотека для AVR

Управление ЖК дисплеем с помощью функций CodeVisionAVR

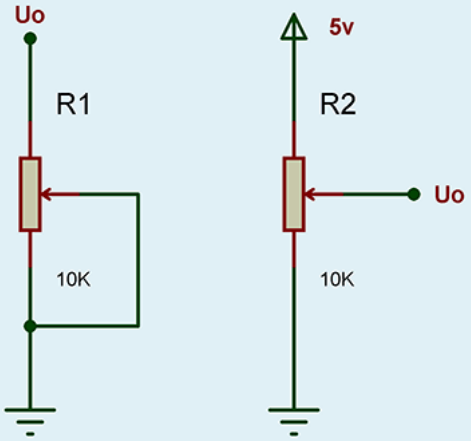
По просьбе трудящихся, да и моим обещаниям решил я описать работу с знаковым ЖК 16x2 в среде CodeVisionAVR. Начнем с описания самого ЖК. Алфовитно-цифровой ЖК дисплей со встроенным чипом HD44780 фирмы Hitachi может выводить символы в одну, две или четыре сроки по 8, 16, 20 или 40 символов в каждой. В данной статье я буду рассматривать ЖК 16x2*(16 символов, 2 строки)*. Данный дисплей для физического подключения к МК имеет 16 выводов*(расположение выводов зависит от фирмы изготовителя)*. Давайте посмотрим на эти выводы. Не мудрствуя лукаво я спер табличку в МЭЛТе. В принципе она подходит для любого ЖК.

Вывод	Обозначение	Назначение вывода
1	GND	Общий вывод (0В)
2	UCC	Напряжение питания (5В/3В)
3	Uo	Управление контрастностью
4	A0	Адресный сигнал — выбор между передачей данных и команд управления
5	R/W	Выбор режима записи или чтения
6	E	Разрешение обращений к индикатору (а также строб данных)
7	DB0	Шина данных (8-ми битный режим)(младший бит в 8-ми битном режиме)
8	DB1	Шина данных (8-ми битный режим)
9	DB2	Шина данных (8-ми битный режим)
10	DB3	Шина данных (8-ми битный режим)
11	DB4	Шина данных (8-ми и 4-х битные режимы)(младший бит в 4-х битном режиме)
12	DB5	Шина данных (8-ми и 4-х битные режимы)
13	DB6	Шина данных (8-ми и 4-х битные режимы)
14	DB7	Шина данных (8-ми и 4-х битные режимы) (старший бит)
15	+LED	+ питания подсветки
16	–LED	– питания подсветки

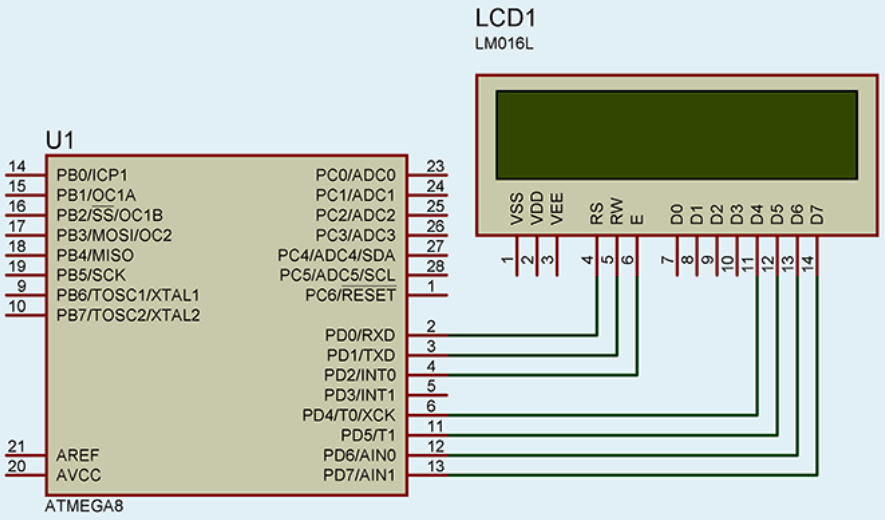
Ну я думаю что объяснять не нужно для чего нужен тот или иной пин. Там все написано по русски. Но есть несколько небольших но.

1) ЖК дисплеи могут быть выпущены в двух вариантах на 5 вольт, либо на 3,3.
2) В цепи питания не всегда установлен токоограничивающий резистор. Смотрите внимательно, может стоять просто перемычка.*(Я так спалил подсветку на двух дисплеях.)*
3) Схема включения резистора для регулировки контрастности.

Управление контрастностью



Так, ну теперь как сие чудо подключить к МК. Работать будем с ATmega8 и кварцем на 4 МГц. Вот собственно и схема.



Как видите ничего сложного нет. Первые три разряда порта **D** служат для управления, а последние четыре для данных. Также можно работать с этими дисплеями по 8-и битной шине, но я думаю отдавать лишние 4 ноги это расточительство. Поэтому будем работать по 4-х битной шине. Со схемой разобрались, теперь давайте с программной частью. Для инициализации дисплея и перевод его в 4-х битный режим нужно выполнить несколько команд. Но перед этим я хочу разъяснить как работают управляющие биты. Бит RS отвечает за то что будет принимать ЖК. Если **RS = 0**, то мы передаем команду, а если **1** то данные. Если бит **RW = 0**, то мы записываем в ЖК, а если **1**, то читаем. Бит **E** просто строб. То есть как только мы захотим ввести команду или данные, то после того как выставили все биты на ножках просто выставляем в **1** бит **E**, а потом опять роняем в **0**.

1 - Включить питание
2 - Выдержать паузу не менее 20 мс
3 - Команда для 4-х бит. шины **(RS=0), (RW=0), (D7=0), (D6=0), (D5=1),(D4=1)**
4 - Выдержать паузу не менее 40 мкс
5 - Команда для 4-х бит. шины **(RS=0), (RW=0), (D7=0), (D6=0), (D5=1),(D4=1)**
6 - Выдержать паузу не менее 40 мкс
7 - Команда для 4-х бит. шины **(RS=0), (RW=0), (D7=0), (D6=0), (D5=1),(D4=1)**
8 - Выдержать паузу не менее 40 мкс
9 - Команда для 4-х бит. шины **(RS=0), (RW=0), (D7=0), (D6=0), (D5=1),(D4=0)**
10 - Выдержать паузу не менее 40 мкс
11 - Выставить параметры **(RS=0), (RW=0), (D7=0), (D6=0), (D5=1),(D4=0)**
(RS=0), (RW=0), (D7=1), (D6=0), (D5=0),(D4=0)
12 - Выключаем дисплей **(RS=0), (RW=0), (D7=0), (D6=0), (D5=0),(D4=0)**
(RS=0), (RW=0), (D7=0), (D6=0), (D5=1),(D4=0)
13 - Очищаем экран **(RS=0), (RW=0), (D7=0), (D6=0), (D5=0),(D4=0)**
(RS=0), (RW=0), (D7=0), (D6=0), (D5=0),(D4=1)
14 - Режим ввода данных **(RS=0), (RW=0), (D7=0), (D6=0), (D5=0),(D4=0)**
(RS=0), (RW=0), (D7=0), (D6=1), (D5=1),(D4=0)

О как. Теперь после этой абракадабры наш дисплей готов принимать данные. Что дальше. А дальше давайте ка рассмотрим команды ЖК. Для передачи команд/данных в ЖК по 4-х битной шине требуется два захода. Первым передаем старшие 4 байта, а

вторым передаем младшие 4 байта. Дальше все команды я буду писать парами.

Команда очистки индикатора и постановка курсора в левый верхний угол.
RS=0, RW=0, D4=0, D5=0, D6=0, D7=0 (E=1 потом 0)
RS=0, RW=0, D4=0, D5=0, D6=0, D7=1 (E=1 потом 0)

Команда перемещения курсора в левую позицию.*(X-значит пофик какое значение)*
RS=0, RW=0, D4=0, D5=0, D6=0, D7=0 (E=1 потом 0)
RS=0, RW=0, D4=0, D5=0, D6=1, D7=X (E=1 потом 0)

Команда устанавливает направление сдвига курсора(ID=0/1 влево/вправо).
Так же разрешение сдвига дисплея (SH=1) при записи в DDRAM.
RS=0, RW=0, D4=0, D5=0, D6=0, D7=0 (E=1 потом 0)
RS=0, RW=0, D4=0, D5=1, D6=ID, D7=SH (E=1 потом 0)

Команда включения дисплея (D=1) и выбора курсора (A, B).
A=0, B=0 Курсора нет, ничего не мигает
A=0, B=1 Курсора нет, мигает весь символ
A=1, B=0 Курсор в виде подчеркивания, не мигает
A=1, B=1 Курсор в виде подчеркивания и мигает
RS=0, RW=0, D4=0, D5=0, D6=0, D7=0 (E=1 потом 0)
RS=0, RW=0, D4=1, D5=D, D6=A, D7=B (E=1 потом 0)

Команда сдвига дисплея/курсора(SC=0/1 курсор/дисплей RL=0/1 влево/вправо).
RS=0, RW=0, D4=0, D5=0, D6=0, D7=1 (E=1 потом 0)
RS=0, RW=0, D4=SC, D5=RL, D6=X, D7=X (E=1 потом 0)

Команда установки разрядности шины(DL=0/1 4/8 бит) A так же страницы знакогенератора P.
RS=0, RW=0, D4=0, D5=0, D6=1, D7=DL (E=1 потом 0)
RS=0, RW=0, D4=1, D5=0, D6=P, D7=0 (E=1 потом 0)

Команда установки адреса следующей операции с установкой туда курсора и выбора области CGRAM(Свои придуманные символы).
RS=0, RW=0, D4=0, D5=1, D6=ACG, D7=ACG (E=1 потом 0)
RS=0, RW=0, D4=ACG, D5=ACG, D6=ACG, D7=ACG (E=1 потом 0)

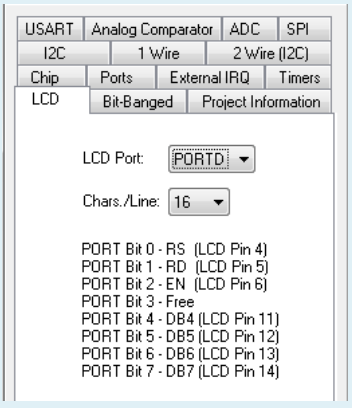
Команда установки адреса последующей операции и выбор области памяти DDRAM (Знакогенератор).
RS=0, RW=0, D4=0, D5=1, D6=ADD, D7=ADD (E=1 потом 0)
RS=0, RW=0, D4=ADD, D5=ADD, D6=ADD, D7=ADD (E=1 потом 0)

Команда Записи данных в текущую область.
RS=1, RW=0, D4=DATA, D5=DATA, D6=DATA, D7=DATA (E=1 потом 0)
RS=1, RW=0, D4=DATA, D5=DATA, D6=DATA, D7=DATA (E=1 потом 0)

Команда Чтения данных в текущую область.
RS=1, RW=1, D4=DATA, D5=DATA, D6=DATA, D7=DATA (E=1 потом 0)
RS=1, RW=1, D4=DATA, D5=DATA, D6=DATA, D7=DATA (E=1 потом 0)

Вот собственно и все команды. Есть еще команда чтения флага занятости, но я ей не пользуюсь, а просто выдерживаю между каждой командой не менее 40 мкс. Вот и все. А теперь после прочтения этого трактата, выпейте чашку чая или кофе и забудьте про все это. Так как всю эту муру на себя берут функции из библиотеки CodeVisionAVR.

Создаем новый проект как это было уже рассказано. Для тех кто не в курсе идем сюда, остальные заходят в код-генераторе на вкладку **LCD** и выбирают **PORTD**.



Что мы этим сделали. Первое мы сказали программе что хотим работать с ЖК дисплеем (выбрав вкладку **LCD**). Потом мы сказали что подключим его к порту **D**. Ниже выпадающий список дает возможность выбрать количество символов в строке. Так как по умолчанию стоит **16**, а мы хотим работать с ЖК 16x2, то ничего менять не надо. Ниже для подсказки расписаны ножки порта для правильного подключения ЖК к МК. Все, сохраняем проект и смотрим на свеже-сгенерированный код.

Первое на что надо обратить внимание - это на кусок кода после директивы препроцессора **#include <mega8.h>**

Вот на этот:

```
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>>
```

Давайте его разберем построчно. Первая строка комментарий в котором говорится о том что мы подключили заголовочный файл с функциями для работы со знаковым ЖК. Второй строкой мы открываем блок для ввода ассемблерных команд. Следующая строка присваивает порт к которому подключен ЖК. Команда **.equ** в ассемблере делает тоже самое что команда **#include** в С. Если вы случайно в генераторе кода выбрали не тот порт, то его можно всегда поменять в этой строке. Номер порта всегда можно узнать в файле инициализации МК. Он всегда подключается в самой первой строке. В нашем случае это **mega8.h**. Следующая строка закрывает блок ассемблерного кода. И последняя строка как раз и подключает все необходимое для работы с ЖК. Теперь давайте пробежимся по основным функциям.

Первая функция которую необходимо вызвать до того как вы начали мучать ЖК - это конечно же функция инициализации дисплея. Выглядит она так:

void lcd_init(unsigned char lcd_columns)

Данная функция инициализирует дисплей, а передаваемым параметром должно быть количества символов в строке. Мотаем нашу программу в самый низ и перед основным циклом видим две строки следующего содержания:

```
// LCD module initialization
 lcd_init(16);
```

Вот те самые 16 строк которые были выбраны в списке код-генератора программа и запихнула аргументом в функцию. Здесь также если вы с перепугу забыли что у вас ЖК 8 или 20 символов на строку, то просто поменяйте значение аргумента в этой функции.

void lcd_gotoxy(unsigned char x, unsigned char y)

Эта функция, судя из ее названия, переводит курсор в позицию **x, y**. Здесь **x** - это буковка. Слева направо от 0 до 15/19/39*(зависит от количества букв в строке)*. А **y** - это строка. Сверху вниз от 0 до 0/1/3*(зависит от количества строк)*.

void lcd_putchar(char c)

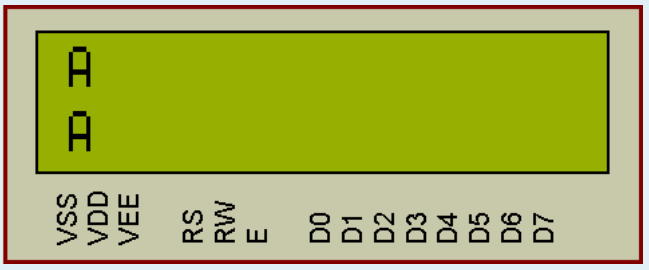
Эта функция выводит один символ в текущую позицию. Пример:

```
lcd_putchar('A')
```
или
```
lcd_putchar(0x41)
```

что на выходе даст один и тот же результат. То есть параметр может быть как символ, так и его код.

```
lcd_gotoxy(0,0);
lcd_putchar('A');
lcd_gotoxy(0,1);
lcd_putchar(0x41);
```

Я думаю комментарии здесь излишне, давайте посмотрим на результат.



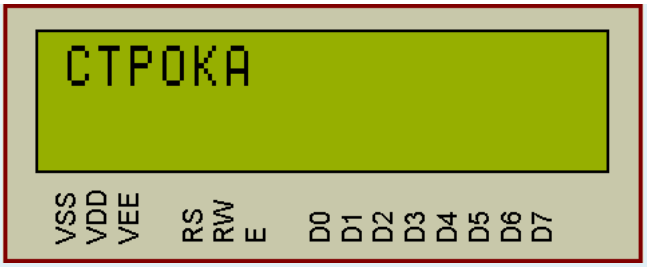
Следующая функция.

void lcd_puts(char *str)

Эта функция выводит строку расположенную в SRAM начиная с текущей позиции. Пример:

```
lcd_gotoxy(0,0);
lcd_puts("СТРОКА");
```

Видим:



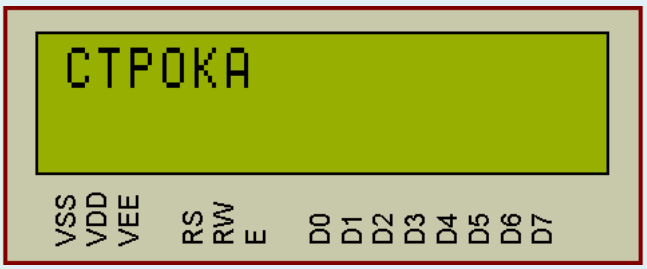
Следующая функция.

void lcd_putsf(char *str)

Эта функция выводит строку расположенную во FLASH начиная с текущей позиции.
Пример:

```
lcd_gotoxy(0,0);
lcd_putsf("СТРОКА");
```

Видим:



Ну и замыкает все это безобразие функция "Ластик"

void lcd_clesr(void)

Вызвав данную функцию вы сотрете все что есть на дисплее, а курсор встанет в крайнее левое положение верхней строки. Вот так для начала можно выводить слова и цифры на ЖК дисплей при помощи готовых функций.

Теперь давайте поговорим о том как выводить значение переменных. Для этих целей нам понадобится еще одна библиотека. Ну те кто программировал на С под ПК про нее должны знать. Называется она **stdio.h**
Поднимаемся на самый верх программы и после директивы препроцессора **#include <lcd.h>** добовляем **#include <stdio.h>** В итоге наш код примет вид.

```
// Alphanumeric LCD Module functions

#asm
  .equ __lcd_port=0x12 ;PORTD
#endasm
#include  <lcd.h >
#include  <stdio.h >
```

Теперь давайте познакомимся с функцией, которая занимается форматированием текста.

void printf(char flash *fmtstr [,arg1, arg2, ...])

Как она работает. В **char flash *fmtstr** задается формат выводимого значения, а в аргументы **arg1, arg2, ...** имя переменной.
Пример.

```
unsigned char temp = 123;
printf("temp = %05d\n", temp);
```

Что означает эта абра-кадабра. Первая строка создает переменную и присваевает ей значение. Тут все понятно, а вот что делает вторая. Все по порядку. Сначала выводится запись **temp =** , затем **00123**. Почему выводится **00123**. А потому что у нас есть условие **%05d\n** которое говорит:

1) **%** - будем форматировать значения первого аргумента
2) **0** - будем выводить n знаков, пустые забьем нулями
3) **5** - выводим 5 знаков, если число меньше 5 знаков, то заполнить пустышки нулями. Об этом говорит пункт 2. Число будет выровнено по правому краю.
4) **d** - выводим число в десятичном формате.
5) **\n** - Заставит после вывода символа перейти на другую строку.

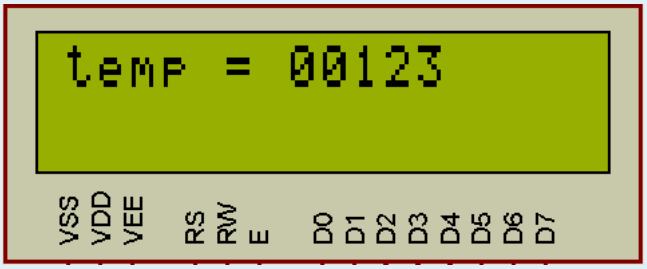
Следующая функция.

void sprintf(char flash, char flash *fmtstr [,arg1, arg2, ...])

Вот эта функция нам наиболее интересна. Она форматирует строку и записывает ее в массив. После мы можем смело массив вывести на экран. Как она работает.

```
unsigned char temp = 123;
unsigned char string[20];
sprintf(string, "temp = %05d\n", temp);
lcd_puts(string);
```

Вот как это выглядит в живую.



Вот мы и научились выводить форматированный текст на ЖК. Далее кратко пробегусь по типам преобразования.

i - Для вывода десятичной целой со знаком
d - Для вывода десятичной целой со знаком
u - Для вывода десятичной целой без знака
e - Для вывода вещественного с плавающей точкой вида **-d.d e-d**
E - Для вывода вещественного с плавающей точкой вида **-d.d E-d**
f - Для вывода вещественного с плавающей точкой вида **-d.d**
x - Для вывода в шеснадцатеричном виде маленькими буквами
X - Для вывода в шеснадцатеричном виде большими буквами
c - Для вывода в символа

Если написать **%-05d** то знак **"-"** заставит выравнивать по левому краю, а пустышки нулями забиваться не будут.
Если вы попытаетесь напечатать число с плавающей точкой, то сильно удивитесь. Число не напечатается. Во засада)) Проблема кроется в настройках компилятора. Для того чтобы компилятор начал понимать формат **float** нужно его немного настроить. Для этого заходим **Project->Configure** и заходим во вкладку **C Compiler**. В своистве **(s)printf Features:** выбираем **float, width, precision**. Вот и все.
Пробуйте, экспериментируйте. Возникнут вопросы, пишите на форуме.
Удачи!

<-Назад