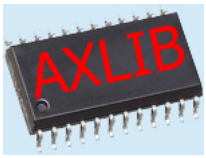
- О проекте
- Обратная связь
- Полезные ссылки
- Полезные программы
- Друзья сайта

Последние комментарии

Алексей: *Библиотека для AtmelStudio 6.x а-ля CodeVisionAVR*
Выставляет на пор...

Алёна: *Подключение SD карты к STM32 через 4-х битный интерфейс SDIO*
mx cube stm32f4 s...

Библиотека для AVR



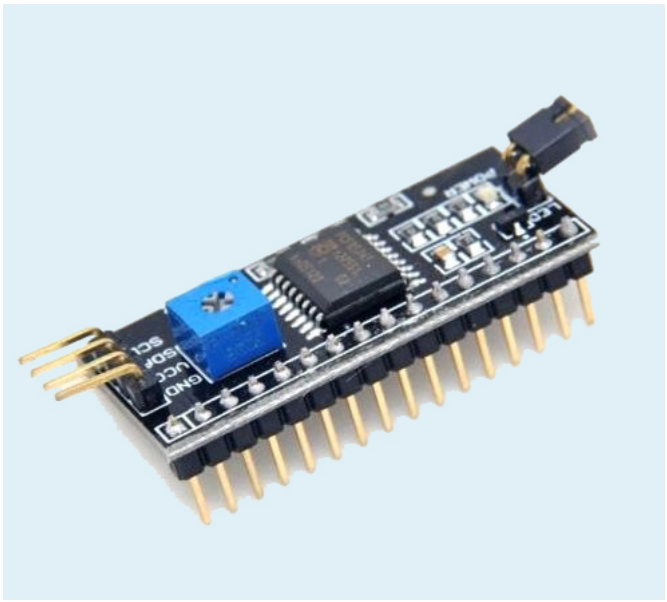
AXLIB Генератор

Помощь сайту

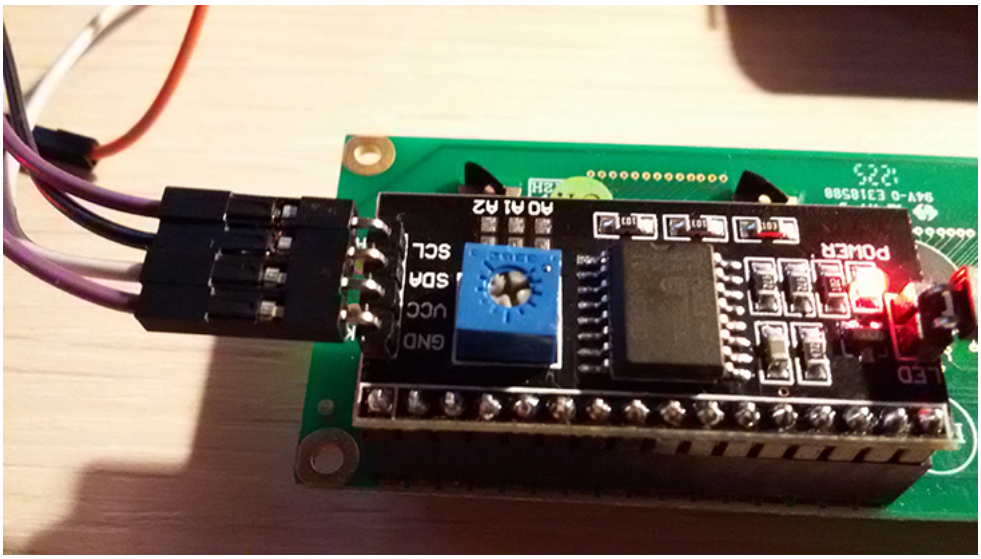
Управление LCD 1602 по шине I2C

Дата: 13 Июля 2015. Автор: Алексей

Заполучил я тут от хорошего магазина Чип Резистор очередной девайс для изучения и применения в полезных устройствах. Сей девайс оказался заточен для управления ЖК дисплеем под управлением контроллера HD44780, в 4-х битном режиме. Для этой цели на плате установлена микросхема PCF8574, которая является преобразователем шины I2C в параллельный 8 битный порт.



Плата разведена таким образом, чтобы ее можно было сразу скрестить с ЖК дисплеем. На вход подается питание и линии I2C. На плате сразу установлены подтягивающие резисторы на линиях SCL и SDA, потенциометр для регулировки контрастности и питание самого дисплея.



Джампер справа включает/отключает подсветку. Далее вооружившись тестером была составлена следующая табличка.

Выводы LCD	Выводы модуля
RS	P0
R/W	P1
E	P2

D4	P4
D5	P5
D6	P6
D7	P7

После изучения модуля было выявлено что **P3** управляет подсветкой. Если джампер установлен, то 1 включает подсветку, а 0 выключает. При снятом джампере подсветка всегда выключена. Далее было принято решение дополнить библиотеку axlib функциями для работы с шиной I2C(программная реализация) и функциями для управления микросхемой PCF8574. В двух словах как работает модуль. Для того чтобы вывести параллельно байт, для этого нужно послать в шину I2C адрес микросхемы (по умолчанию он равен 0x4E. Так же можно менять адрес методом впаивания перемычек на плате и меняя значение трех младших разрядов адреса), затем после получения ACK посылается байт данных. После того как микросхема отвечает ACK, байт появляется на параллельном порту микросхемы. Для управления ЖК дисплеем я взял функции из библиотеки axlib и немного переделал их для работы с шиной I2C.

```
#include <avr/io.h>

#include <axlib/main_init.h>
#include <axlib/i2c.h>
#include <axlib/pcF8574.h>

#define ADD 0x4E // Адрес микросхемы

/*
     LCD       Микросхема
     RS        P0
     RW        P1
     EN        P2
     D4        P4
     D5        P5
     D6        P6
     D7        P7

На ножке P3 подключени подсветка. 1 вкл, 0 выкл.
*/

void com(BYTE com)
{
     com |= 0x08;               // P3 в единицу, дабы горела подсветка
     pcf8574_byte_out(com, ADD);    // Вывод данных
     com |= 0x04;                 // E в единицу
     pcf8574_byte_out(com, ADD);    // Вывод данных
     com &= 0xFB;                 // E в ноль
     pcf8574_byte_out(com, ADD);    // Вывод данных
}

void init(void)
{
     _delay_ms(30);          // Пауза после подачи питания
     com(0x30);         // Переход в 4-х битный режим
     _delay_us(40);          // Задержка для выполнения команды
     com(0x30);         // Переход в 4-х битный режим
     _delay_us(40);          // Задержка для выполнения команды
     com(0x30);         // Переход в 4-х битный режим
     _delay_us(40);          // Задержка для выполнения команды
     com(0x20);         // Переход в 4-х битный режим
     _delay_us(40);          // Задержка для выполнения команды
     com(0x20);         // Установка параметров
     com(0x80);         // Установка параметров
     com(0x00);         // Выключаем дисплей
     com(0x80);         // Выключаем дисплей
     com(0x00);         // Очищаем дисплей
     com(0x10);         // Очищаем дисплей
     com(0x00);         // Устанавливаем режим ввода данных
     com(0x60);         // Устанавливаем режим ввода данных
     com(0x00);         // Включаем дисплей с выбранным курсором
     com(0xC0);         // Включаем дисплей с выбранным курсором
}

void char_out(BYTE data)
{
     BYTE data_h = ((data & 0xF0) + 0x09);
     BYTE data_l = ((data << 4) + 0x09);

     pcf8574_byte_out(data_h, ADD); // Передача старших 4 бит
     data_h |= 0x04;
     pcf8574_byte_out(data_h, ADD); // Передача старших 4 бит
     data_h &= 0xF9;
     pcf8574_byte_out(data_h, ADD); // Передача старших 4 бит

     pcf8574_byte_out(data_l, ADD); // Передача младших 4 бит
     data_l |= 0x04;
     pcf8574_byte_out(data_l, ADD); // Передача младших 4 бит
     data_l &= 0xF9;
     pcf8574_byte_out(data_l, ADD); // Передача младших 4 бит
}

void str_out(BYTE *str)
{
     while((*str) != '\0')
     {
```

```
            char_out(*str);
            str++;
        }
}

int main(void)
{

      init();
      str_out("ËPĽBET MĽP!");

    while(1)
    {

    }
}
```

Собственно что здесь происходит. Сначала подключаем библиотеки для I2C и для PCF8574. Про I2C я писал уже здесь, поэтому распинаться еще раз на буду, а вот что в PCF8574.h я расскажу. В состав библиотеки вошли всего три функции.

```
BYTE pcf8574_test(BYTE add)
{
      BYTE ask = ACK;
      add &= 0xFE;

      i2c_start();
      ask = i2c_send_byte(add);
      i2c_stop();

      return ask;
}
```

Первая функция была написана для проверки наличия устройства на шине. В принципе ее можно применять для поиска любого устройства находящегося на шине. Функция принимает адрес искомого устройства и если оно отвечает, то возвращает ноль. Если устройство с таким адресом нет на шине, то вернет единицу.

```
BYTE pcf8574_byte_out(BYTE data, BYTE add)
{
      BYTE ask = ACK;
      add &= 0xFE;

      i2c_start();
      ask = i2c_send_byte(add);
      if(!ask) ask = i2c_send_byte(data);
      i2c_stop();

      return ask;
}
```

Эта функция уже заточена чисто под данную микросхему. В качестве аргументов ей передаются байт для передачи в шину и адрес микросхемы. Функция сначала запросит микросхему по адресу, а затем пошлет байт. Если микросхема получила байт и ответила ACK, то функция закончит работу с микросхемой и вернет ноль как удачная посылка байта. А микросхема в это время выведет этот байт в свой параллельный порт. Иначе получим NACK и вернем единицу, передача провалилась.

```
BYTE pcf8574_str_out(BYTE *data, BYTE col, BYTE add)
{
      BYTE ask = ACK;
      add &= 0xFE;

      i2c_start();
      ask = i2c_send_byte(add);
      for(BYTE i=0; i<col; i++)
      {
            ask = i2c_send_byte(*data);
            if(ask)
            {
                  i2c_stop();
                  return ask;
            }
            data++;
      }
      i2c_stop();

      return ask;
}
```

Эта функция создана для эксперимента. Принимает указатель на массив однобайтовых данных, количество этих байт и адрес микросхемы. Собственно попытка передать все данные одной сессией, а не одним байтом за сессию. Функция работает, но так для ЖК дисплея и не подошла. А теперь давайте вернемся к основной программе. После подключения библиотек, прописываем адрес микросхемы. Далее создаем три функции по аналогии с lcd.h. Отличие лишь в принципе передачи данных.

```
void com(BYTE com)
{
	com |= 0x08;			// P3 в единицу, дабы горела подсветка
	pcf8574_byte_out(com, ADD);	// Вывод данных
	com |= 0x04;			// E в единицу
	pcf8574_byte_out(com, ADD);	// Вывод данных
	com &= 0xFB;			// E в ноль
	pcf8574_byte_out(com, ADD);	// Вывод данных
}
```

Эта функция передает только команды дисплею. Отсюда появилась первая строка с логическим сложением команды с 0x08. Эта бяка нужна из-за того что мы передаем байт не прямо в порт ЖК дисплея, а через наш ретранслятор. То есть если мы подали байт, а потом нам нужно вывести только один бит, то соизвольте к предыдущему байту присвоит нужный бит и уже его снова отправить в порт. Вот такая заморочка. Сложение с 0x08 необходимо для постоянного удержания единицы на третьем разряде. Помните про подсветку? Вот именно это сложение и включает подсветку. После вызываем функцию передачи байта в шину. О ней написано выше. Затем передаем байт по шине в микросхему. Далее следует выставить в единицу E, чем собственно занимается логическое сложение байта с 0x04. После обнуление E. Таким образом можно послать любую команду дисплею лишь передав в качестве аргумента саму команду.

```
void init(void)
{
	_delay_ms(30);		// Пауза после подачи питания
	com(0x30);		// Переход в 4-х битный режим
	_delay_us(40);		// Задержка для выполнения команды
	com(0x30);		// Переход в 4-х битный режим
	_delay_us(40);		// Задержка для выполнения команды
	com(0x30);		// Переход в 4-х битный режим
	_delay_us(40);		// Задержка для выполнения команды
	com(0x20);		// Переход в 4-х битный режим
	_delay_us(40);		// Задержка для выполнения команды
	com(0x20);		// Установка параметров
	com(0x80);		// Установка параметров
	com(0x00);		// Выключаем дисплей
	com(0x80);		// Выключаем дисплей
	com(0x00);		// Очищаем дисплей
	com(0x10);		// Очищаем дисплей
	com(0x00);		// Устанавливаем режим ввода данных
	com(0x60);		// Устанавливаем режим ввода данных
	com(0x00);		// Включаем дисплей с выбранным курсором
	com(0xC0);		// Включаем дисплей с выбранным курсором
}
```

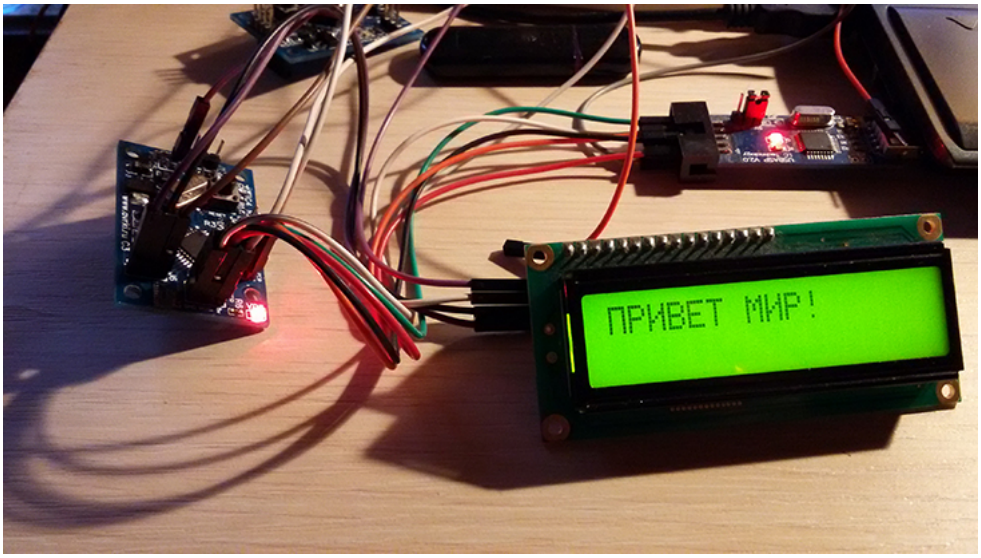
Эта функция занимается лишь инициализацией дисплея. Последовательность команд взята из даташита на ЖК дисплей.

```
void char_out(BYTE data)
{
	BYTE data_h = ((data & 0xF0) + 0x09);
	BYTE data_l = ((data << 4) + 0x09);

	pcf8574_byte_out(data_h, ADD); // Передача старших 4 бит
	data_h |= 0x04;
	pcf8574_byte_out(data_h, ADD); // Передача старших 4 бит
	data_h &= 0xF9;
	pcf8574_byte_out(data_h, ADD); // Передача старших 4 бит

	pcf8574_byte_out(data_l, ADD); // Передача младших 4 бит
	data_l |= 0x04;
	pcf8574_byte_out(data_l, ADD); // Передача младших 4 бит
	data_l &= 0xF9;
	pcf8574_byte_out(data_l, ADD); // Передача младших 4 бит
}
```

Эта функция передает данные ЖК дисплею. Выполняется так же как и команды за исключением того, что передача байта идет сначала старшим полубайтом, а затем младшим. А остальное тоже самое.

```
void str_out(BYTE *str)
{
	while((*str) != '\0')
	{
		char_out(*str);
		str++;
	}
}
```

Ну, а эта функция чисто для передачи строки дисплею. Собственно к нашей теме она никакого отношения не имеет.



Проект для AtmelStudio 6.2

Грамотный 01.08.15 17:11

Запятая пропущена. Правильно: "ПРИВЕТ, МИР!" И сей девайс заточен не только для HD44780. Подтягивающие резисторы ставятся со стороны мастера. Согласно спецификации, запись данных в контроллер LCD идет по спаду E. Отсюда первая же функция упрощается: void com(BYTE com) { com |= 0x08; // подсветка pcf8574_byte_out(com | 0x04, ADD);// Вывод данных pcf8574_byte_out(com, ADD); // E в ноль } Да и остальные тоже существенно меньше могут быть. Например, void char_out(BYTE data) будет всего из двух вызовов, и уж тем более без дополнительных переменных. Инициализация LCD выполнена с нарушениями спецификации таймингов.

Алексей 02.08.15 19:11

Из-за отсутствия запятой, дисплей не пострадает. Сей девайс как раз заточен именно под дисплеи с таким, либо аналогичным контроллером. А вот именно микросхема действительно простой расширитель порта. По поводу E я согласен. Дополнительные переменные нужны. Если передать функции аргумент с выполнением неких действий с логикой, могут возникнуть глюки. Уже с таким сталкивался. Инициализация выполняется без нарушений таймингов. В документации сказано, что между командами ставиться пауза 40 мкс. Из-за того что передача идет по шине i2c, а та в свою очередь программная и медленная, то периоды выполняются с лихвой. Если все же Вам не лень, то напишите свой вариант и пришлите мне. Я его опубликую. В конце концов данный сайт предназначен на любительскую аудиторию и каждый кто хочет может высказать свое мнение и видение на жизнь МК.

Алексей 06.08.15 09:14

Добавлены тайменги при инициализации дисплея по замечанию уважаемого "Грамотного"

Дмитрий 14.06.16 21:57

Здравствуйте Алексей.Можно в генератор кода добавить библиотеку для работы с PCF8574.

Алексей 14.06.16 22:32

Я подумаю.))

ruslan 21.12.16 19:54

Ребята, нашел подробное описание по подключению LCD 1602 по I2C интерфейсу на основе PCF8574 Подключение LCD 1602 I2C

Алексей 21.12.16 21:53

О да. Особенно код на асме. Ардуинщики оценят по полной)))

пы.сы.
Даже если не взирать на асм, то там прога написана под PIC контроллер. Для AVRщиков это "очень" полезная информация? особенно начинающим))) Я ничего не имею против PIC, но даже асм у PIC и AVR разный. А по поводу подробностей работы ЖК дисплея, то можно глянуть тут))) Правда я ее еще писал под CVAVR но все команды разобраны и разложены по полочкам. Но в любом случае решайте сами где понятнее написано))) Автор пишет, читатель выбирает.

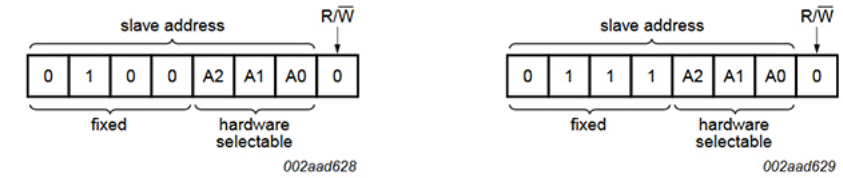
GeK 04.01.17 12:52

"I2C адрес микросхемы (по умолчанию он равен 0x4E"

Старшие 4 бита адреса фиксированы,
префикс У PCF8574 равен 0100, а у PCF8574A — 0111
Младшие 3 бита зависят от состояния входов микросхемы A2-A0. По умолчанию все 3 перемычки разомкнуты, соответственно адрес микросхемы принимает значение 0111111.
// A2 A1 A0 PCF8574 PCF8574A
// 1 1 1 0x20 0x38
// 1 1 0 0x21 0x39
// 1 0 1 0x22 0x3A
// 1 0 0 0x23 0x3B
// 0 1 1 0x24 0x3C
// 0 1 0 0x25 0x3D
// 0 0 1 0x26 0x3E
// 0 0 0 0x27 0x3F

Алексей 04.01.17 14:27

Что-то вы перепутали.
Выписка из документации на микросхему



a. PCF8574 b. PCF8574A

Fig 6. PCF8574 and PCF8574A slave addresses

0b01001110 это 0x4E
Так что тут все верно. А если нужно сменить адрес, то всего лишь нужно его поменять в дефайне.

Юрий 14.12.17 21:26
Доброго времени суток! А можно еще код функции lcdgotoxy и lcdclear для работы с переходником на PCF8574.

Чтобы вставить ссылку используйте форму вида **[url]***http://www.адрес.ru***[/url][text]***текст ссылки***[/text]**
Чтобы вставить код используйте форму вида **[code]***код***[/code]**

Имя: Вставить ссылку Вставить код

Отправить