Главная › I2C › AVR Урок 16. Интерфейс TWI (I2C). Часть 7

AVR Урок 16. Интерфейс TWI (I2C). Часть 7

Posted on Декабрь 17, 2016 by Narod Stream Опубликовано в I2C, Программирование AVR — 6 комментариев ↓

Урок 16
Часть 7

Интерфейс TWI (I2C)

В предыдущей части занятия уже продолжили свою работу с отправкой значений в память микросхемы EEPROM по интерфейсу **TWI**. Только проверить мы смогли это, исключительно веря статусам.

А в данной части урока мы закрепим полученные знания и подключим по шине **I2C** переходник для модуля символьного дисплея, тем самым также получим возможность подключать дисплей по двум проводам.

Модуль дисплея мы будем подключать с дисплеем символьным высотой в 4 строки и шириной в 20 символов, или как попросту говорят 20×4.

Мы в уроке 12 уже разбиравли с вами подобный дисплей, только он был размерностью **16×2** и подключали мы его по 4-битному интерфейсу.

Теперь у нас дисплей **20×4**. Его модуль также собран с применением контроллера дисплея **HD44780**. Возможны тамже варианты применения

Свежие комментарии

- SmNikolay к записи STM Урок 89. LAN. ENC28J60. TCP WEB Server. Подключаем карту SD
- Narod Stream к записи AVR Урок 3. Пишем код на СИ. Зажигаем светодиод
- strannik2039 к записи AVR Урок 3. Пишем код на СИ. Зажигаем светодиод
- Dmitriy к записи AVR Урок 1. Знакомство с семейством AVR
- Narod Stream к записи STM Урок 9. HAL. Шина I2C. Продолжаем работу с DS3231

Форум. Последние ответы

- Narod Stream в Программирование МК STM32
  1 неделя, 3 дн. назад
- Zandy в Программирование МК STM32
  1 неделя, 4 дн. назад
- Narod Stream в Программирование МК STM32
  3 нед., 1 день назад
- Narod Stream в Программирование МК STM32
  3 нед., 1 день назад
- fireweb в Программирование МК STM32
  3 нед., 3 дн. назад

Январь 2018

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

« Дек

Архивы

- Январь 2018
- Декабрь 2017
- Ноябрь 2017
- Октябрь 2017
- Сентябрь 2017
- Август 2017
- Июль 2017

Мета

- Регистрация
- Войти
- RSS записей
- RSS комментариев
- WordPress.org

искать здесь

Уроки по программированию МК

Программирование МК PIC

Тесты устройств и аксессуаров

Заходите на канал Narod Stream

- Июнь 2017
- Май 2017
- Март 2017
- Февраль 2017
- Январь 2017
- Декабрь 2016
- Ноябрь 2016

аналогов, но программирование их вообще не отличается ни чем.

По 4-битному способу программирования модуль дисплея **LCD2004** мало чем отличается от **LCD1602**, в основном, можно сказать только адресацией знакомест. В конце урока мы просто подправим функцию позиционирования.

Основная задача занятия — научить данный модуль общаться с МК **AVR** посредством шины **I2C (TWI)** с помощью соответствующего переходника.

Поэтому давайте немного познакомимся с данным переходником.

Выглядит он следующим образом

У него черыре ножки для соединения с МК — это питание, общий провод и две ножки I2C.

Также мы видим, что к нему подпаяна гребёнка из 16 контактов для соединения с соответствующими контакта модуля дисплея. Так как контакты на переходнике типа "ПАПА", то на модуль дисплея я, соответственно, подпаял гребёнку с контактами типа "МАМА" ну и посредством этих двух контактных площадок я его с модулем и соединил.

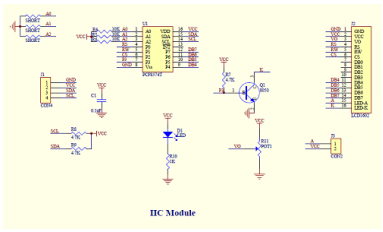
Купить данный дисплей можно много где. В описании видеоверсии под роликом находится ссылка, где его приобретал я. Только там где я приобретал, один было купить невозможно, минимум три, но цена была такова, что я с удовольствием их купил. Сейчас, по-моему, продавец уже предлагает минимальную партию из 5 штук, но цену вроде ещё убавил. Поэтому есть смысл. Я не думаю, что у кого-то, кто постоянно занимается с контроллерами. в наличии есть только один такой дисплей.

Ну что ж, теперь рассмотрим, из чего состоит собственно данный переходник. Там, соответственно, есть регулятор контрастности, поэтому теперь нам не нужно будет заботиться о его отдельном подключении. А сердцем данного переходника служит микросхема **PCF8574**, преобразующая последовательный код I2C в 8-битные логические состояния на выходе.

Вот, собственно, схема данного переходника (нажмите на картинку для увеличения изображения)

Рубрики

- 1-WIRE (3)
- ADC (6)
- DAC (4)
- GPIO (26)
- I2C (19)
- SPI (13)
- USART (8)
- Программирование AVR (131)
- Программирование PIC (7)
- Программирование STM32 (213)
- Тесты устройств и аксессуаров (1)



Здесь прекрасно видно подключение микросхемы. У нее существуют три адресных контакта, которые способны изменить адрес устройства для I2C также, как это было в случае микросхемы EEPROM, рассмотренной в предыдущих частях нашего занятия. Соответственно, существуют контакты питания VCC и GND, а также контакты шины I2C — SDA и SCL, ну и, конечно же, восемь контактов логических состояний, управляемых данной микросхемой. Только подключен будет модуль дисплея. как мы видим посредством 4-битного подключения. Это сделано потому, что контактов на шине микросхемы всего 8, а нам нужно ещё оперировать с помощью них управляющими ножками модуля дисплея (RS, RW и E, который здесь назван CS), а также включением подсветки (контакт P3). Подсветка, соответственно, в целях защиты от больших токов контакта параллельного порта микросхемы управляется с помощью ключевого транзистора 8050. Также установлен светодиод через токоограничивающий резистор, сигнализирующий нам о подаче питания на преобразователь, ну и. как уже было выше сказано, регулятор контрастности. Ещё существуют два подтягивающих резистора по 4,7 килоом, подключенные к ножкам шины I2C, назначение которых мы прекрасно знаем.

Ну вот, собственно, и всё насчёт схематического решения переходника.

Теперь немного по техническим характеристикам самой микросхемы **PCF8574**.

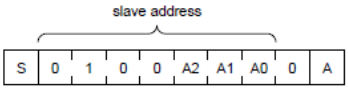
Питание микросхемы осуществляется постоянным напряжением в диапазоне от 2,5 до 7 вольт, так что бояться нам нечего.

Токи по портам входным — максимально 20 милиампер, по выходным — 25.

Также ещё немаловажным для нас является параметр максимальной скорости по I2C — он не должен превышать 100 кГц.

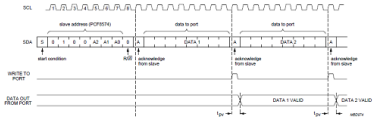
Нам, конечно, спешить некуда, дисплей всё-таки символьный, но если мы захотим на шину I2C повесить ещё что-то и захотим от этого чего-то большего быстродействия, то мы обязаны об этом помнить.

Также посмотрим адрес, по которому мы будем, собственно, с ней общаться по I2C. В даташите их два в зависимости от типа микросхемы. Нам нужен вот этот



a. PCF8574.

Так как мы в данный переходник данные будем только отправлять, то давайте посмотрим диаграмму протокола общения с ним по шине I2C для записи байтов (нажмите на картинку для увеличения изображения)



Запись здесь идёт стандартно. Просто у нас не память EEPROM и никаких ячеек адресных у нас в микросхеме нет. передаём мы сразу данные. Данные можно передавать как по несколько байт. так и по одному. Как только мы передадим условие СТОП, на этом и заканчивается передача. И передавать мы будем только по одному байту, так как со скоростью 100 кГц контроллер дисплея вряд ли сможет работать, да это и не нужно.

Ну вот мы немного и разобрались с переходником.

Теперь, конечно проект.

Проект был создан с именем **I2CLCD80**.

Стандартно создан и подключен файл main.h.

Также подключены файлы lcd.h и lcd.c из проекта урока 12 по подключению подобного дисплея по 4-битной шине **Test09**. Затем данные файлы были переименованы соотвественно для избежания путаницы в **lcdtwi.c** и **lcdtwi.h**, а также в связи с этим хедер-файл немного претерпел изменения и внутри в директивах.

Код в главный файл **I2CLCD80.c** был полностью скопирован из того же проекта Test09 из файла Test09.c.

Ещё, конечно же были подключены файлы **twi.h** и **twi.c** из проекта прошлых частей данного занятия **DS1307Eeprom**.

Ну и. соответственно, все данные файлы были также подключены в файле main.h. Вот его полный исходный код

```
#ifndef MAIN_H_
#define MAIN_H_
#define F_CPU 8000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include "twi.h"
#include "lcdtwi.h"
#endif /* MAIN_H_ */
```

В файле **lcdtwi.c** подключили также файл **lcdtwi.h**

```
#include "lcdtwi.h"
//————————————-
```

Начнём теперь адаптировать понемногу код к переходнику.

Напишем ещё одну функцию в файл twi.c для передачи байта по адресу, чтобы упростить вызов. Напишем её после функции передачи байта в шину, так как мы ею будем пользоваться

```
void I2C_SendByteByADDR(unsigned
char c,unsigned char addr)
{
  I2C_StartCondition(); // Отправим
условие START
  I2C_SendByte(addr); // Отправим в
шину адрес устройства + бит чтения-
записи
  I2C_SendByte(c);// Отправим байт
данных
  I2C_StopCondition();// Отправим
условие STOP
}
```

Также не забываем о прототипе в хедере.

Теперь в файле реализации функций дисплея **lcdtwi.c** мы создадим глобальную переменную, которая будет в себе хранить значение состояния контактов на параллельной шине микросхемы, чтобы нам его не читать из шины I2C, а просто помнить его именно в переменной

```
#include "lcdtwi.h"
//————————————-
unsigned char portlcd = 0; //ячейка
для хранения данных порта микросхемы
расширения
```


Теперь исправим макроподстановки в файле **lcdtwi.h** и добавим ещё некоторые в соответствии со схемой на переходник

```
void sendcharlcd(unsigned char c);
//————————————-
#define e1
I2C_SendByteByADDR(portlcd|=0x04,0b0
1001110) // установка линии E в 1
#define e0
I2C_SendByteByADDR(portlcd&=~0x04,0b
01001110) // установка линии E в 0
#define rs1
I2C_SendByteByADDR(portlcd|=0x01,0b0
1001110) // установка линии RS в 1
#define rs0
I2C_SendByteByADDR(portlcd&=~0x01,0b
01001110) // установка линии RS в 0
#define setled()
I2C_SendByteByADDR(portlcd|=0x08,0b0
1001110) // включение подсветки
#define setwrite()
I2C_SendByteByADDR(portlcd&=~0x02,0b
01001110) // установка записи в
память дисплея
```

Здесь, в принципе всё понятно. Мы отправляем определённые значения в шину I2C, для того чтобы менять состояние ножек параллельной шины на микросхеме, причем не изменив при этом состояние остальных ножек, для чего и идёт постоянное логическое сложение с состоянием данной параллельной шины. Все битовые операции нам давно знакомы и говорить подробно о них мы не будем.

Вернёмся теперь в файл lcdtwi.c и подправим там тоже некоторые вещи.

Первым делом, коненчно же у нас несколько изменится функция отправки половины байта в контроллер дисплея. Там мы изменим только одну строку и уберём установку значения порта D, так как мы его не используем теперь и он у нас свободен

```
void sendhalfbyte(unsigned char c)
{
  c<<=4;
  e1; //включаем линию E
  _delay_us(50);

I2C_SendByteByADDR(portlcd|c,0b01001
110);
  e0; //выключаем линию E
  _delay_us(50);
}
```

Теперь функция setpos. В неё также нужно будет внести определённые изменения. так как у нас изменилось разрешение дисплея. У него увеличилось количество строк и колонок.

Для этого заглянем в техническую документацию на дисплей и посмотрим адресацию в нём памяти **DDRAM**

DDRAM address:

Display position

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

DDRAM address

Здесь всё ясно, нам нужны только адреса первого символа в каждой строке, на них мы и будем смещаться по строкам

```
//————————————-
void setpos(unsigned char x,
unsigned y)
{
  switch(y)
  {
    case 0:
      sendbyte(x|0x80,0);
      break;
    case 1:
      sendbyte((0x40+x)|0x80,0);
      break;
    case 2:
      sendbyte((0x14+x)|0x80,0);
      break;
    case 3:
      sendbyte((0x54+x)|0x80,0);
      break;
  }
}
```

Смысл байта 0x80 я думаю все знают. Здесь мы устанавливаем седьмой бит, чтобы контроллер дисплея распознал команду записи в DDRAM в соответствии с таблицей инструкций.

Также нужно в функции инициализации дисплея включить подстветку и запись

```
  _delay_ms(1);
  setled();//подсветка
  setwrite();//запись
}
```

Здесь, в принципе, всё.

Перейдём в главный файл I2CLCD80.c и удалим там функцию инициализации порта **port_ini**.

Также удалим и в функции main() вызов инициализации порта, а вместо него включим инициализацию шины **TWI**

```
int main(void)
{
  I2C_Init();//инициализируем TWI
  LCD_ini(); //инициализируем
дисплей
```

Также давайте после инициализации дисплея вызовем на всякий случай его очистку, а то частенько мусор остаётся после персборки и перезагрузки

```
LCD_ini(); //инициализируем дисплей
clearlcd();//очистим дисплей
```

Соберём код и прошьём контроллер.

После этого на дисплее мы должны увитель вот такую знакомую картину

Ну, и теперь, я думаю, логично будет проверить работоспособность двух нижних строк дисплея. Добавим код в main()

```
str_lcd("String 2");
setpos(4,2);
str_lcd("String 3");
setpos(6,3);
str_lcd("String 4");
while(1)
```

Соберём код, прошьём контроллер и проверим это

Ну вот теперь всё!

В данной части занятия мы усовершенствовали свои знания в области программирования шины I2C в контроллерах AVR, тем самым также мы неплохо разгрузили значительное количество ножек контроллера.

Предыдущая часть

Программирование МК AVR

Следующий урок

Исходный код

Техническая документация на микросхему PCF8574

Схема переходника

Техническая документация на дисплей LCD 20×4

Программатор и модуль RTC DS1307 с микросхемой памяти можно приобрести здесь:

Программатор (продавец надёжный) **USBASP USBISP 2.0**

Модуль RTC DS1307 с микросхемой памяти

Смотреть ВИДЕОУРОК (нажмите на картинку)

‹ AVR Урок 16. Интерфейс TWI (I2C). Часть 6

STM Урок 40. Знакомство с платой NUCLEO STM32F767ZI ›

6 комментариев на “AVR Урок 16. Интерфейс TWI (I2C). Часть 7”

Виталий:
Февраль 11, 2017 в 7:00 дп

Добрый день,
при запуске симуляции, которая находится в архиве, дисплей ни чего не показывает.

Ответить

admin:
Февраль 11, 2017 в 7:23 пп

Странно. У меня показывал.

Ответить

Denis:
Февраль 21, 2017 в 2:59 дп

ПОдтверждаю. В протеусе ничего не показывает. Также перекомпелированный под mega2560 код тоже не работает…

Ответить

Vadim:
Март 11, 2017 в 1:32 дп

Здравствуйте!
Проверьте пожалуйста, и у меня не работает в Proteus
PROSPICE 8.04.00 (Build 21720) (C) Labcenter Electronics 1993-2016.
Loaded netlist 'C:\Users\Comp\AppData\Local\Temp\LISA5604.SDF' for design 'I2CLCD80.pdsprj'
AVR Release 8.3SP0 build 22019 for ATMEGA8. [U1]
Loading HEX file 'I2CLCD80\Debug\I2CLCD80.hex'. [U1]
Read total of 658 bytes from file 'I2CLCD80\Debug\I2CLCD80.hex'. [U1]
Simulation is not running in real time due to excessive CPU load.

Ответить

admin:
Март 11, 2017 в 4:50 дп

Здравствуйте!
Без проекта проверить не могу.

Ответить

Ростислав Троян:
Июль 3, 2017 в 8:07 пп

Добрый день! Разобрался почему не работает в протеусе, дело в том что в эмулиции когда на десплей передаем полубайт удерживается линия "E" в 1, а считование происходит во время изминения этой линии в 0, но когда это происходит то в этот момент полубайт пустой — без данных. Вот фитч на функцию и все заработает:

```
void sendhalfbyte(unsigned char value)
{
  value<<=4;
  e1;
  _delay_us(50);
  //———
    TWI_SendByteByADDR(portlcd |= value , 0b01001110);
  //   ———
     e0;
     portlcd &= ~value;  //удаление
```

данных для избегания переполнения

```
    _delay_us(50);
}
```

У меня все зароботало, но чет медленно выводит символы. Немогу понять почему. Реального железа нет, пока играюсь только на протеусе. Кто собрал реальную схему отпишитесь, с какой скоростью выводит символы, зарание спасибо.

Ответить

Добавить комментарий

Ваш e-mail не будет опубликован. Обязательные поля помечены *

Комментарий

Имя *

E-mail *

Сайт

2 + = четыре

Отправить комментарий