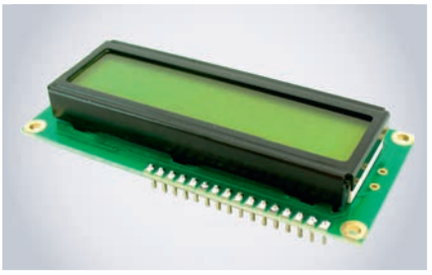
Современные 32-разрядные ARM серии STM32: подключение LCD-дисплея WH1602

Олег Вальпа (sandh@narod.ru)

В статье приведён пример подключения LCD-дисплея WH1602 фирмы Winstar к микроконтроллеру серии STM32 компании STMicroelectronics с целью практического освоения.



Введение

При разработке микропроцессорных устройств довольно часто возникает необходимость в организации человеко-машинного интерфейса. К решению данного вопроса следует относиться тщательно, поскольку от этого зависит удобство эксплуатации устройства, его внешний вид, информативность и в целом эргономика.

К человеко-машинному интерфейсу предъявляется два основных требования: отображение информации и обеспечение возможности управлять устройством. В настоящее время существует множество вариантов решения этой задачи. В качестве элементов управления могут выступать кнопки, манипуляторы, сенсорные панели и т.п. Приборами для отображения информации могут выступать точечные и семисегментные индикаторы, монохромные и цветные графические дисплеи, мониторы и т.д.

Наиболее популярными приборами для отображения информации являются символьные монохромные LCD-дисплеи на базе контроллера HD44780. Для проведения эксперимента воспользуемся одним из таких приборов – дисплеем WH1602 (две строки по 16 символов) от компании Winstar. Он получил широкое распространение благодаря: низкой цене, унификации и простоте интерфейса, возможности отображения нескольких строк, содержащих десятки символов, хорошей яркости и читаемости информации. Следует отметить, что существует много аналогов данного дисплея, совместимых по интерфейсу и системе команд.

Подключение дисплея

Рассмотрим пример подключения дисплея WH1602 к микроконтроллеру серии STM32 [1]. Приведём программу для его использования.

Дисплей можно подключить к микроконтроллеру по четырёх- или восьмибитной шине данных. С целью сокращения количества связей остановимся на первом варианте подключения. Схема подключения дисплея к микроконтроллеру показана на рисунке.

При подключении дисплея необходимо обратить внимание на распиновку выводов дисплея, которые имеют следующее назначение:

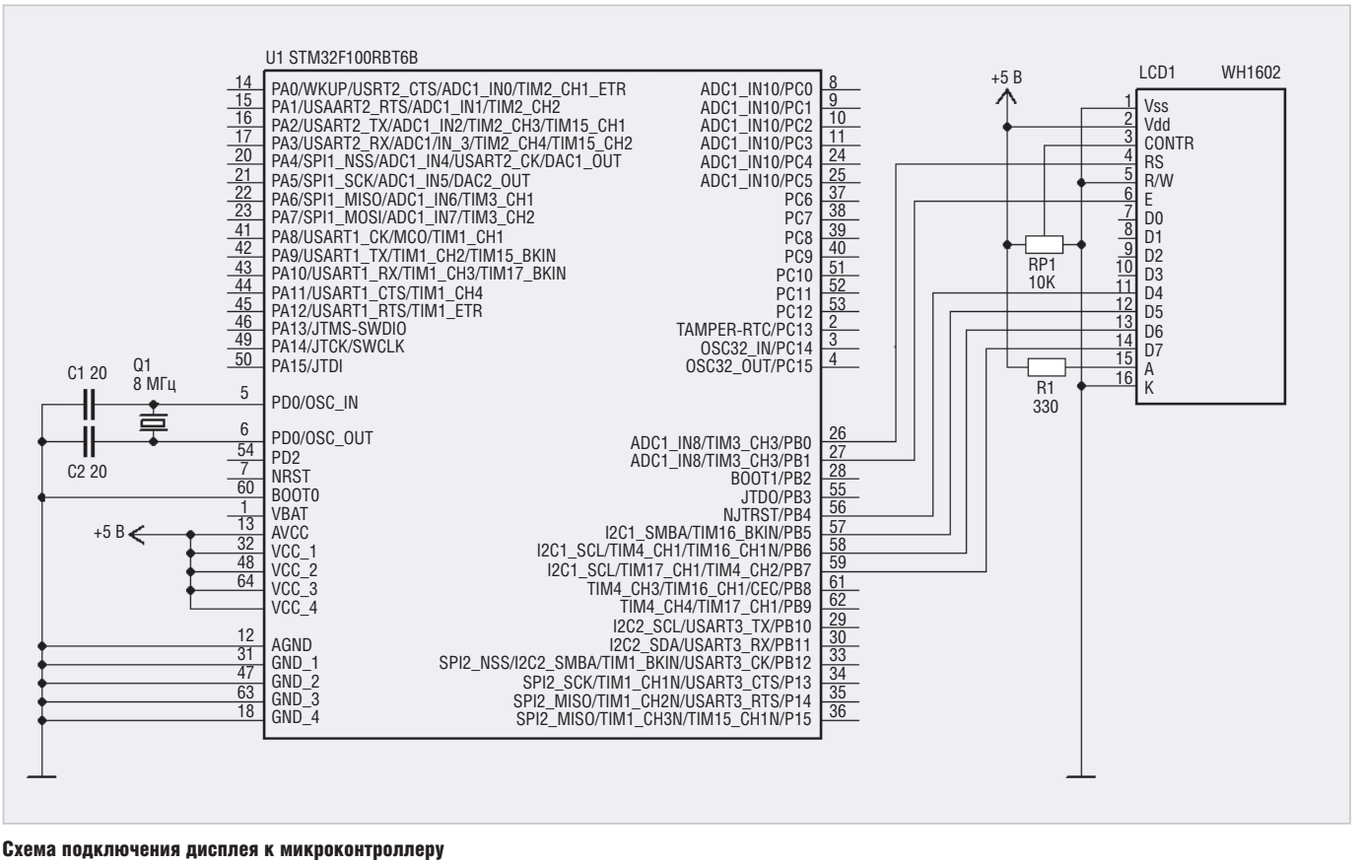


Схема подключения дисплея к микроконтроллеру

Листинг

```
#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
// Макроопределения
#define LCD_OUT GPIOB->ODR
#define LCD_PIN_RS GPIO_Pin_0 // PB0
#define LCD_PIN_EN GPIO_Pin_1 // PB1
#define LCD_PIN_D4 GPIO_Pin_4 // PB4
#define LCD_PIN_D5 GPIO_Pin_5 // PB5
#define LCD_PIN_D6 GPIO_Pin_6 // PB6
#define LCD_PIN_D7 GPIO_Pin_7 // PB7
#define LCD_PIN_MASK ((LCD_PIN_RS | LCD_PIN_EN | LCD_PIN_D7 | LCD_
PIN_D6 | LCD_PIN_D5 | LCD_PIN_D4))

// Инициализация портов микроконтроллера
GPIO_InitTypeDef GPIO_InitStructure;

// Функция задержки
void delay(int a)
{
int i=0, f=0;
while(f < a) {while(i<60) {i++;} f++;}
}

// Функция стробирования дисплея
void StrobLCD()
{
LCD_OUT &= ~LCD_PIN_EN;
delay(230);
LCD_OUT |= LCD_PIN_EN;
delay(230);
LCD_OUT &= (~LCD_PIN_EN);
delay(230);
}

// Функция записи байта в дисплей
void SendByte(char ByteToSend, int IsData)
{
LCD_OUT &= (~LCD_PIN_MASK);
LCD_OUT |= (ByteToSend & 0xF0);
if(IsData == 1) LCD_OUT |= LCD_PIN_RS;
else LCD_OUT &= ~LCD_PIN_RS;
StrobLCD();
LCD_OUT &= (~LCD_PIN_MASK);
LCD_OUT |= ((ByteToSend & 0x0F) << 4);
if (IsData == 1) LCD_OUT |= LCD_PIN_RS;
else LCD_OUT &= ~LCD_PIN_RS;
StrobLCD();
}

// Функция установки позиции курсора
void Cursor(char Row, char Col)
{
char address;
if (Row == 0) address = 0;
else address = 0x40;
address |= Col;
SendByte(0x80 | address, 0);
}

// Функция очистки дисплея
void ClearLCDScreen()
{
SendByte(0x01, 0);
SendByte(0x02, 0);
}

// Функция инициализации дисплея
void InitializeLCD(void)
{
int i;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
GPIO_InitStructure.GPIO_Pin =GPIO_Pin_0 | GPIO_Pin_1| GPIO_Pin_4 |
GPIO_Pin_5| GPIO_Pin_6| GPIO_Pin_7;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOB, &GPIO_InitStructure);
LCD_OUT &= ~(LCD_PIN_MASK);
for(i=0;i<3;i++) delay(32000);
LCD_OUT &= ~LCD_PIN_RS;
LCD_OUT &= ~LCD_PIN_EN;
LCD_OUT = 0x20;
StrobLCD();
SendByte(0x28, 0);
SendByte(0x0E, 0);
SendByte(0x06, 0);
}

// Функция отображения строки
void PrintStr(char *Text)
{
char *c;
c = Text;
while ((c != 0) && (*c != 0)) {SendByte(*c, 1); c++;}
}

// Главный модуль программы
int main(void)
{
InitializeLCD(); // Инициализация дисплея
ClearLCDScreen(); // Очистка дисплея от символов
Cursor(0,2); // Установить курсор на строку 0, в столбец 2
PrintStr("Hello world!"); // Вывод текста
Cursor(1,4); // Установить курсор на строку 1, в столбец 4
PrintStr("0123456789");
while(1)
 {
 // Место для циклического кода программы
 }
}
```

1 – V_{ss} – общий вывод;

2 – V_{dd} – вывод напряжения питания;

3 – V_o – вывод управления контрастностью дисплея;

4 – RS – сигнал назначения данных в качестве информации для отображения или команды, например, для задания позиции отображения символов;

5 – R/W – сигнал управления чтением и записью данных дисплея;

6 – E – сигнал стробирования данных;

7...14 – DB0...DB7 – шина данных для обмена информацией;

15 – A – анод светодиодной подсветки;

16 – K – катод светодиодной подсветки.

Пример программы

В качестве примера представляем программу, отображающую на дисплее две строки текста. В листинге приведён код такой программы с пояснительными комментариями.

Рассмотрим назначение применённых в программе функций.

Функция инициализации дисплея InitializeLCD() должна выполняться при старте программы.

С помощью функции ClearLCDScreen() производится очистка памяти дисплея от предыдущих записей.

Функция Cursor(y,x) служит для установки позиции курсора. Отсчёт начинается с нулевой строки и нулевого столбца.

Функция вывода байта в дисплей SendByte(byte, mode) позволяет либо отобразить символ на дисплее с параметром режима mode=1, либо управлять дисплеем в режиме настройки при mode=0. Эта функция применяется для очистки дисплея, установки курсора, выбора типа курсора и т.п. Например, команда SendByte(0x0C, 0) отключит курсор.

Дисплей позволяет отображать курсор в одном из трёх режимов: мигающий курсор, курсор в виде нижнего подчёркивания и скрытый курсор. Сделать курсор мигающим можно с помощью команды SendByte(0x0F, 0). Курсор в виде нижнего подчёркивания активируется командой SendByte(0x0E, 0).

Получить более подробную информацию о дисплее WH1602 и познакомиться с другими моделями дисплеев можно на сайте производителя [2].

Литература

1. www.st.com
2. www.winstar.com.tw/ru/products