Работа с последовательным интерфейсом SPI в программной среде Proteus 8.11.
Часть 2

Татьяна Колесникова (beluikluk@gmail.com)

В статье рассматривается проектирование схем микроэлектронных устройств с использованием интерфейса SPI в Proteus на примере его реализации в микроконтроллерах AVR (семейства Mega) и STM32 (семейства Cortex-M3). Описаны особенности написания программного кода для инициализации интерфейса и работы с ним, а также моделирования схем, в которых проводится передача данных через SPI между двумя и тремя устройствами, сконфигурированными как master и slave. Выполнено отображение принятых ведомым устройством данных на экране виртуального терминала. С помощью осциллографа проведён контроль входных/выходных сигналов, присутствующих на выводах устройств схемы.

Передача данных через интерфейс SPI между тремя микроконтроллерами Cortex-M3

Рассмотрим процесс передачи данных через интерфейс SPI между несколькими микроконтроллерами Cortex-M3 на примере микросхем STM32F103C4, для чего создадим в Proteus новый схемный проект и добавим в рабочее поле три таких микросхемы, два светодиода, два резистора (100 Ом), два символа «земли». При этом микросхема DD1 будет выполнять роль ведущего микроконтроллера, а микросхемы DD2 и DD3 – роль ведомых. Соединим компоненты, как показано на рис. 14, и напишем на языке программирования С программный код управления передачей данных. Необходимо отметить, что программа инициализации пишется как для ведущего, так и для обоих ведомых микроконтроллеров.

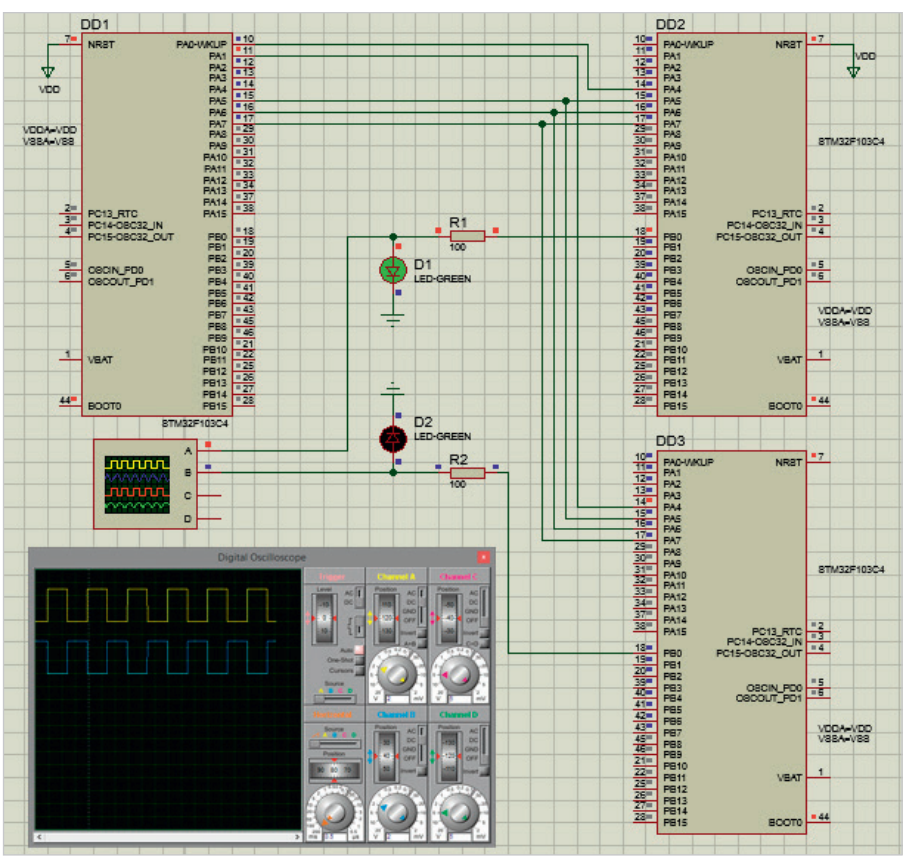


Рис. 14. Моделирование передачи данных между тремя микроконтроллерами STM32F103C4 через интерфейс SPI в программной среде Proteus

Задачей мастера будет послать управляющий сигнал (кодовую комбинацию) сначала первому ведомому устройству, а затем второму. Переключение между ведомыми устройствами выполняется путём установки ведущим микроконтроллером логической единицы на линии NSS (PA4) ведомых микроконтроллеров. При этом при передаче данных по интерфейсу SPI1 между тремя микроконтроллерами в нашем примере этот сигнал выдаётся на линии PA0, PA1 порта PA ведущего микроконтроллера. Задача каждого ведомого устройства – принять кодовую комбинацию, после чего запустить цикл, в котором выполняется последовательное включение и выключение светодиода.

Для удобства соединения в рабочей области проекта отразим по горизонтали микросхему DD1. В окне настроек Edit Component для каждого микроконтроллера в поле Crystal Frequency установим частоту работы 2 МГц. Кнопкой Hidden Pins для каждого микроконтроллера откроем окно Edit Hidden Power Pins, где выполним согласование скрытых выводов питания и цепей питания.

Напишем на языке программирования С следующий код программы инициализации для ведущего микроконтроллера DD1:

```
  #include <stm32f1xx.h> // подключение заголовочного файла

  void delay (int dly) // подпрограмма формирования задержки
  { int i;
  for(; dly>0; dly--)
  for ( i=0; i<10000; i++); }

  int main() // начало программы
  { RCC->APB2ENR |= RCC_APB2ENR_SPI1EN; // включаем тактирование SPI1
```

```
// подсоединение линий порта PA к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;

// настройка линий PA5 (SCK), PA6 (MISO), PA7 (MOSI), PA0-PA4
// порта PA ведущего микроконтроллера
// биты CNF5, CNF7 = 10 (цифровой выход с альтернативной функцией),
// биты MODE5, MODE7 = 11 (вывод данных с частотой переключения 50 МГц),
// биты CNF6 = 10 (вход с подтягивающим резистором),
// биты MODE6 = 00 (приём данных),
// биты CNF0-CNF4 = 00 (цифровой выход),
// биты MODE0-MODE4 = 11 (вывод данных с частотой переключения 50 МГц)
GPIOA->CRL =0xb8b33333;

// конфигурация SPI1
SPI1->CR1 = (0<<11) // формат кадра данных 8 бит
| (0<<7) // направление передачи младшим разрядом вперёд
| (1<<9) // включаем программное управление сигналом NSS
| (1<<8) // NSS в высоком состоянии
| (1<<5)|(0<<4)|(0<<3) // скорость передачи данных: F_PCLK/32
| (1<<2) // режим работы Master (ведущий)
| (0<<1)|(0<<0) // полярность (0) и фаза тактового сигнала (0)
| (1<<6); // включаем SPI

// выбираем для передачи данных по SPI
// первый ведомый микроконтроллер (лог. 1 на линии PA0)
GPIOA->ODR=(1<<0)|(0<<1);
while(!(SPI1->SR & SPI_SR_TXE)) { } // после установки в 1 флага TXE регистра SPI1_SR
// отсылаем кодовую комбинацию для первого ведомого микроконтроллера
SPI1->DR = 0b11111110; delay(10);
// выбираем для передачи данных по SPI
// второй ведомый микроконтроллер (лог.1. на линии PA1)
GPIOA->ODR=(0<<0)|(1<<1);
while(!(SPI1->SR & SPI_SR_TXE)) { } // после установки в 1 флага TXE регистра SPI1_SR
// отсылаем кодовую комбинацию для второго ведомого микроконтроллера
SPI1->DR = 0b11111110; delay(10);
}
```

Для ведомого микроконтроллера DD2 был написан следующий код программы инициализации:

```
#include <stm32f1xx.h> // подключение заголовочного файла

void delay (int dly) // подпрограмма формирования задержки
{ int i;
for(; dly>0; dly--)
for ( i=0; i<10000; i++); }

int main() { // начало программы
RCC->APB2ENR |= RCC_APB2ENR_SPI1EN; // включаем тактирование SPI1
// подсоединение линий порта PA к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
// подсоединение линий порта PB к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;

// настройка линий PA4 (NSS), PA5 (SCK), PA6 (MISO), PA7 (MOSI), PA0-PA3
// порта PA первого ведомого микроконтроллера
// биты CNF4, CNF5, CNF7 = 10 (вход с подтягивающим резистором),
// биты MODE4, MODE5, MODE7 = 11 (приём данных),
// биты CNF6 = 10 (цифровой выход с альтернативной функцией),
// биты MODE6 = 00 (вывод данных с частотой переключения 50 МГц),
// биты CNF0-CNF3 = 00 (цифровой выход),
// биты MODE0-MODE3 = 11 (вывод данных с частотой переключения 50 МГц)
GPIOA->CRL =0x8b883333;
// настройка линий PB0-PB7 порта PB первого ведомого микроконтроллера
// биты CNF0-CNF7 = 00 (цифровой выход),
// биты MODE0-MODE7 = 11 (вывод данных с частотой переключения 50 МГц)
GPIOB->CRL =0x33333333;

// конфигурация SPI1
SPI1->CR1 = (1<<6) | (0<<2); // включаем SPI, режим работы Slave (ведомый)

while (1) // бесконечный цикл
{ while(!(SPI1->SR & SPI_SR_RXNE)) { } // ждём данные в буфере приёмника SPI
if (SPI1->DR !=0b11111110) // если кодовая комбинация не получена
GPIOB->ODR= (0<<0) ; // посылаем на линию PB0 порта PB лог.0
else if (SPI1->DR==0b11111110) // если кодовая комбинация получена
{ while (1) // бесконечный цикл
{GPIOB->ODR=(1<<0); // включить светодиод D1
delay(10); // задержка
GPIOB->ODR=(0<<0); // погасить светодиод D1
delay(10); }}}}
```

Код программы инициализации для ведомого микроконтроллера DD3:

```
#include <stm32f1xx.h> // подключение заголовочного файла

void delay (int dly) // подпрограмма формирования задержки
{ int i;
for(; dly>0; dly--)
for ( i=0; i<10000; i++); }

int main() { // начало программы
RCC->APB2ENR |= RCC_APB2ENR_SPI1EN; // включаем тактирование SPI1
// подсоединение линий порта PA к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
// подсоединение линий порта PB к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;
// настройка линий PA0-PA7 порта PA второго ведомого микроконтроллера
GPIOA->CRL =0x8b883333;
// настройка линий PB0-PB7 порта PB второго ведомого микроконтроллера
GPIOB->CRL =0x33333333;

// конфигурация SPI1
SPI1->CR1 = (1<<6) | (0<<2); // включаем SPI, режим работы Slave (ведомый)

while (1) // бесконечный цикл
```

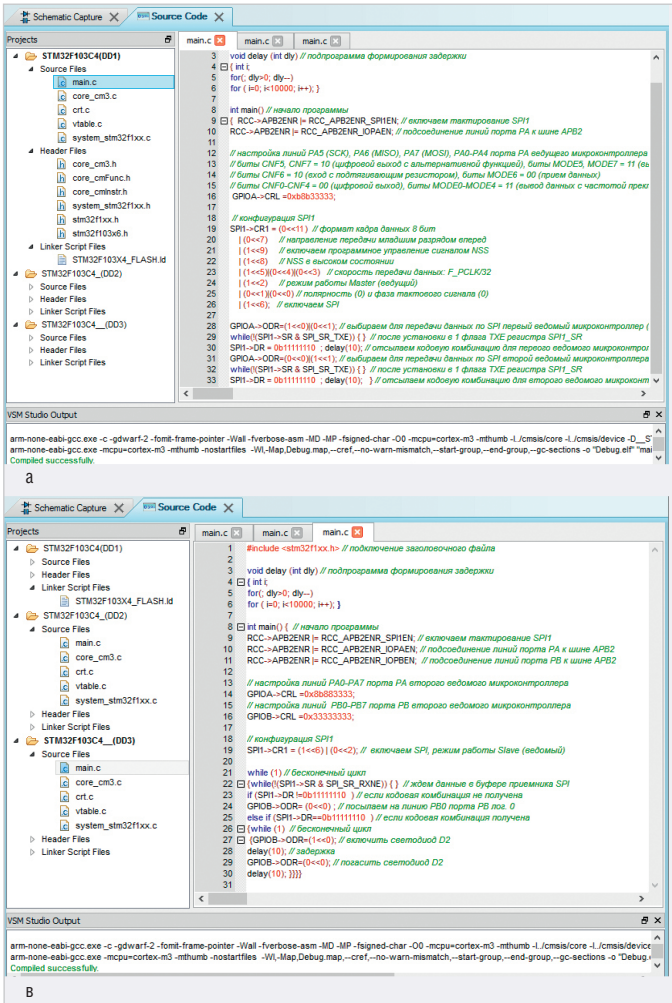


Рис. 15. Передача данных между тремя устройствами через интерфейс SPI. Вкладка Source Code, код программы инициализации: ведущего микроконтроллера (а), первого ведомого микроконтроллера (б), второго ведомого микроконтроллера (в)

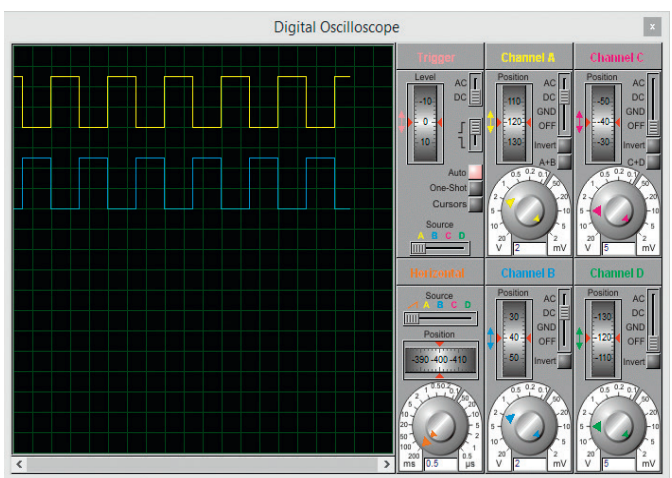


Рис. 16. Осциллограмма работы светодиодов D1 и D2, подключённых к ведомым микроконтроллерам STM32F103C4

```
  {while(!(SPI1->SR & SPI_SR_RXNE))
{ } // ждём данные в буфере при-
ёмника SPI
  if  (SPI1->DR  !=0b11111110)  //
если кодовая комбинация не получена
  GPIOB->ODR= (0<<0) ; // посылаем
на линию PB0 порта PB лог.0
  else if (SPI1->DR==0b11111110) //
если кодовая комбинация получена
  {while (1) // бесконечный цикл
  {GPIOB->ODR=(1<<0); // включить
светодиод D2
  delay(10); // задержка
  GPIOB->ODR=(0<<0); // погасить
светодиод D2
  delay(10); }}}}
```

Код программы инициализации вводится на вкладке Source Code схемного редактора на отдельной закладке для каждого микроконтроллера (см. рис. 15).

Проанализируем работу демонстрационной схемы, представленной на рис. 14. На вкладке Source Code программным путём были даны указания ведущему микроконтроллеру через интерфейс SPI1 отправить каждому ведомому микроконтроллеру кодовую комбинацию. Это действие выполняется последовательно. Сначала ведущий микроконтроллер через линию PA0 своего порта PA подаёт на линию NSS (PA4) микросхемы DD2 логическую единицу, а через линию PA1 на линию NSS (PA4) микросхемы DD3 логический ноль, что оповещает первое ведомое устройство о том, что именно оно выбрано для обмена данными с ведущим по интерфейсу SPI, активизирует интерфейс SPI1 микросхемы DD2 и делает неактивным интерфейс SPI1 микросхемы DD3.

После задержки ведущий микроконтроллер через линию PA1 своего порта PA подаёт на линию NSS (PA4) микросхемы DD3 логическую единицу, а через линию PA0 на линию NSS (PA4) микросхемы DD2 логический ноль, что активизирует интерфейс SPI1 микросхемы DD3 и делает неактивным интерфейс SPI1 микросхемы DD2.

Если ведомое устройство выбрано ведущим, то программа ведомого микроконтроллера выводит на линию PB0 порта PB логический ноль, в результате чего подключённый к порту светодиод будет погашен. Как только по интерфейсу SPI1 получена кодовая комбинация от ведущего микроконтроллера, запускается подпрограмма, дающая указания ведомому микроконтроллеру запустить цикл, в котором последовательно выводятся на линию порта PB0 значения логической 1 и 0. Эти значения удерживаются при помощи команды задержки.

После запуска моделирования при помощи двух светодиодов, подключённых к линиям PB0 порта PB ведомых микроконтроллеров DD2 и DD3, мы можем проверить правильность

работы программы – светодиоды подсвечиваются и гаснут поочерёдно, что наглядно демонстрирует осциллограмма, представленная на рис. 16. В момент времени, когда на выводе PB0 микроконтроллера DD2 единица, на выводе PB0 микроконтроллера DD3 – ноль.

Работа с SPI в микроконтроллерах Mega в Proteus

Передача данных через интерфейс SPI между двумя микроконтроллерами AVR

Рассмотрим процесс передачи данных между двумя микроконтроллерами AVR на примере микросхемы ATmega16, для чего создадим в редакторе Schematic Capture новый проект и добавим в его рабочее поле две микросхемы ATmega16, два светодиода, два резистора (100 Ом), два символа «земли» и соединим компоненты, как показано на рис. 17. Напишем на языке программирования C программный код управления передачей данных. Необходимо отметить, что программа инициализации пишется как для ведущего, так и для ведомого микроконтроллера. Определим микроконтроллер DD1 как ведущий, а микроконтроллер DD2 как ведомый. При этом задачей ведущего контроллера будет послать управляющий сигнал (кодовую комбинацию), задачей ведомого – принять его и последовательно включить и выключить оба светодиода. Для удобства соединения можно отразить в рабочей области микросхему DD1, для чего выделим её при помощи левой кнопки мыши, при помощи правой кнопки мыши вызовем контекстное меню и выберем в нём пункт X-Mirror. В результате микросхема будет отражена по горизонтали в рабочем поле проекта. В таком положении выводы PB4/*SS*, PB5/MOSI, PB6/MISO, PB7/SCK обеих микросхем соединить намного проще, при этом соединительные линии на схеме будут короче.

В окне настроек Edit Component для каждого микроконтроллера установим следующие параметры (см. рис. 18):

- поле CKOPT (Oscillator Options) – (1) Unprogrammed;
- поле BOOTRST (Select Reset Vector) – (1) Unprogrammed;
- поле CKSEL Fuses – (0010) Int.RC 2MHz;

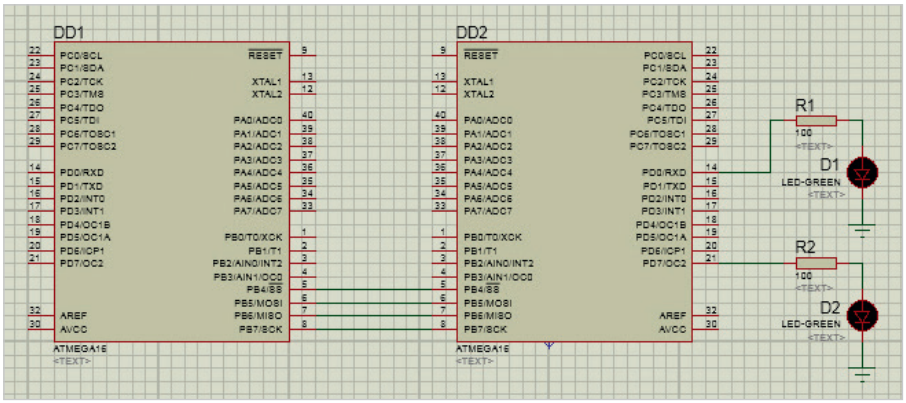


Рис. 17. Демонстрационная схема с использованием двух микроконтроллеров ATmega16 и светодиодов

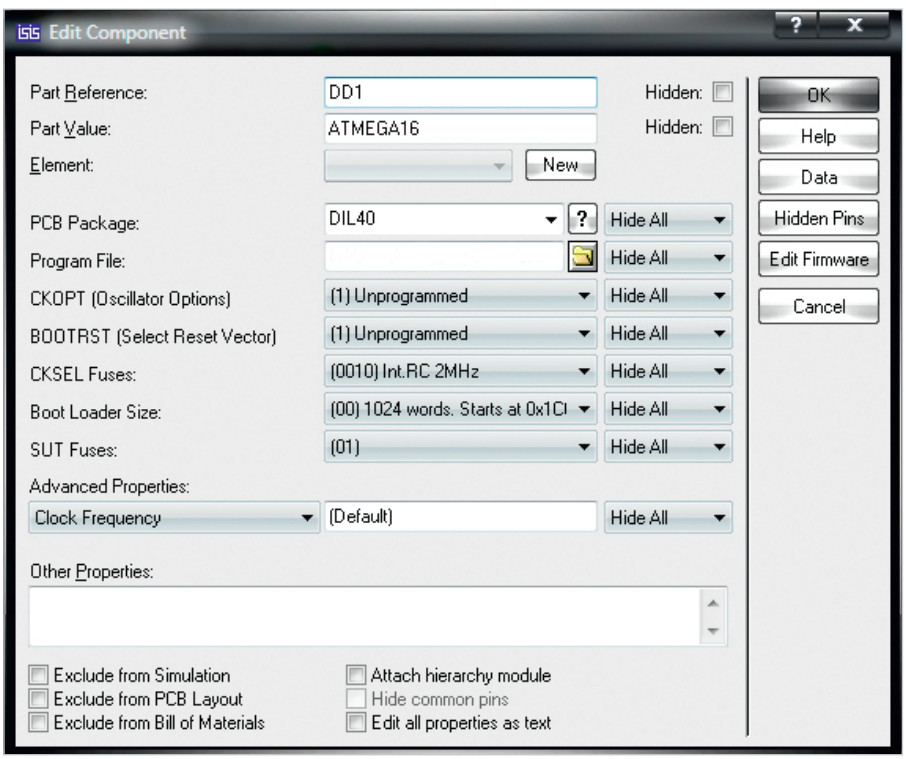


Рис. 18. Настройка параметров микроконтроллера ATmega16 при передаче данных между двумя устройствами через интерфейс SPI

- поле Boot Loader Size – (00) 1024 words. Starts at 0x1C00;
- поле SUT Fuses – (01);
- поле Advanced Properties – Clock Frequency (Default).

Окно настроек можно открыть при помощи двойного щелчка левой кнопкой мыши по выбранному на схеме микроконтроллеру.

В модуле SPI имеется три регистра ввода/вывода:

- SPDR – регистр данных, содержит посылаемый или принятый байт данных;
- SPCR – регистр управления, определяет функционирование модуля SPI;
- SPSR – регистр состояния, отображает состояние модуля SPI.

Включение/выключение SPI выполняется установкой шестого бита (SPE) регистра SPCR, пятый бит (DORD) задаёт порядок передачи данных, а четвёртый бит (MSTR) этого регистра определяет режим работы интерфейса.

Перед выполнением передачи данных необходимо, прежде всего, разрешить работу модуля SPI. Для этого следует установить в единицу шестой бит регистра SPCR. Режим работы определяется состоянием четвёртого бита этого регистра: если бит установлен в 1, микроконтроллер работает в режиме Master, если сброшен в 0 – в режиме Slave. Программно (на языке программирования C) эти действия можно реализовать следующим образом:

```
SPCR=0b01010000;  // установка битов регистра SPCR ведущего микроконтроллера
```

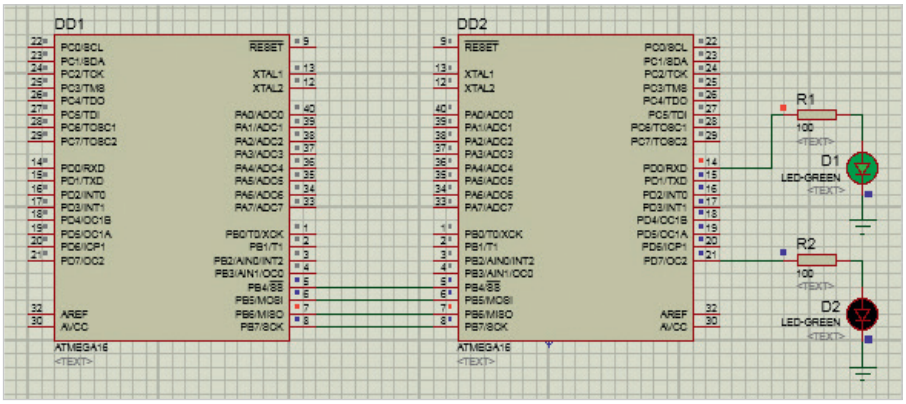


Рис. 19. Процесс моделирования проекта передачи данных между двумя микроконтроллерами ATmega16 через интерфейс SPI в программной среде Proteus

```
  SPCR=0b01000000;  //  установка битов регистра SPCR ведомого микроконтроллера
```

Передача данных осуществляется следующим образом. При записи в регистр данных SPI ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI. Данные начинают побитно выдаваться на вывод MOSI устройства Master и, соответственно, поступать на вывод MOSI устройства Slave. Порядок передачи битов данных определяется состоянием пятого бита регистра SPCR. Если бит установлен в 1, первым передаётся младший бит байта, если же сброшен в 0 – старший бит.

Частота тактового сигнала SCK и соответственно скорость передачи данных по интерфейсу определяются состоянием первого и нулевого битов (SPR1:SPR0) регистра SPCR и нулевого бита (SPI2X) регистра SPSR ведущего микроконтроллера, так как именно он является источником тактового сигнала. Для ведомого микроконтроллера состояние этих битов не имеет значения. Программно частоту тактового сигнала ведущего микроконтроллера можно задать следующим образом:

```
SPCR=0b01010011;
SPSR=0b00000000;
```

Здесь мы установили первый и нулевой бит регистра SPCR в единицу и нулевой бит регистра SPSR в ноль, что приведёт к установке следующей частоты сигнала SCK: fCLK/128, где fCLK – тактовая частота микроконтроллера.

Напишем на языке программирования С следующий код программы инициализации для ведущего микроконтроллера:

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
int main()
{
PORTB=0b00000000; // инициализация порта PB микросхемы DD1
DDRB=0b10110000;  //  указываем направление передачи информации по линиям порта
// линии SS, MOSI, SCK установлены как выходы
SPCR=0b01010011; // инициализация SPI
SPSR=0b00000000;
SPDR=0b11111110;  //  отсылаем кодовую комбинацию для ведомого микроконтроллера
return 0;
}
```

Для ведомого микроконтроллера был написан следующий код программы инициализации:

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
int main()
{
PORTB=0b00000000; // инициализация порта PB микросхемы DD2
DDRB=0b00000000; // линии порта PB работают как входы
PORTD=0b00000000; // инициализация порта PD
DDRD=0b11111111; // линии порта PD работают как выходы
SPCR=0b01000011; // инициализация SPI
while (1) // бесконечный цикл
{
if  (SPDR!=0b11111110)  //  если кодовая комбинация не получена
PORTD=0b00000000; // посылаем на все линии порта PD - 0
else  if  (SPDR==0b11111110)  //  если кодовая комбинация получена
{PORTD=0b00000001;  //  включить светодиод D1
_delay_ms(1000);  //  задержка в 1 секунду
PORTD=0b10000000;  //  погасить светодиод D1 и включить светодиод D2
_delay_ms(1000); } // задержка в 1 секунду
}}
```

После того как в рабочей области проекта собрана схема, а на вкладке Source Code для каждого микроконтроллера (ведомого и ведущего) введён код программы, кнопкой Run the simulation можно запускать моделирование (см. рис. 19). В результате, в том случае, если компилятор не обнаружит ошибок в программе, на диске компьютера в рабочей папке проекта будут созданы файлы *.elf , *.hex и *.c. Для компиляции кода программы, написанного на языке программирования С, в Proteus применяется компилятор WinAVR.

Проанализируем работу демонстрационной схемы, представленной на рис. 19. На вкладке Source Code программным образом были даны указания ведущему микроконтроллеру через интерфейс SPI отправить ведомому микроконтроллеру кодовую комбинацию. Программа ведомого микроконтроллера выводит на линии порта PD все нули, в результате чего два подключённых к порту светодиода будут погашены. Как только по интерфейсу SPI получена кодовая комбинация от ведущего микроконтроллера, запускается подпрограмма, которая даёт команду ведомому микроконтроллеру вывести на линии PD0 и PD7 значения логической 1 и 0 соответственно, которые удерживаются на этих линиях при помощи команды задержки. Затем на линии PD0 и PD7 выводятся значения логического 0 и 1 соответственно, после чего (после задержки) выполнение этого фрагмента программы повторяется. После запуска моделирования при помощи двух светодиодов, подключённых к линиям PD0 и PD7, мы можем проверить правильность работы программы – светодиоды подсвечиваются и гаснут поочерёдно (см. рис. 19).

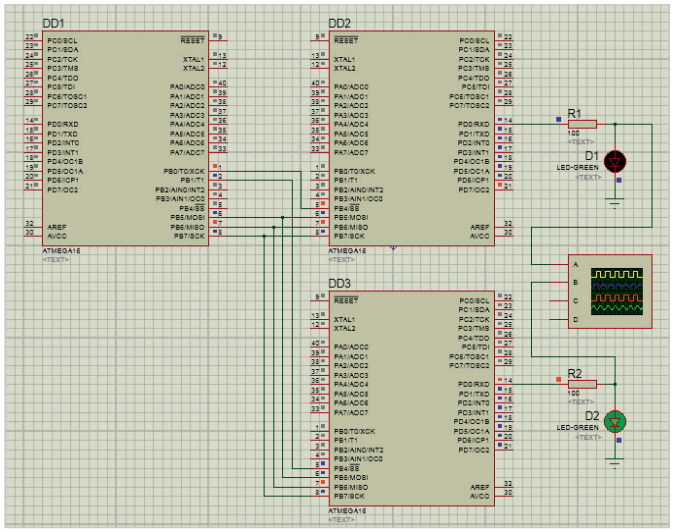


Рис. 20. Процесс моделирования проекта передачи данных между тремя микроконтроллерами ATmega16 через интерфейс SPI в программной среде Proteus

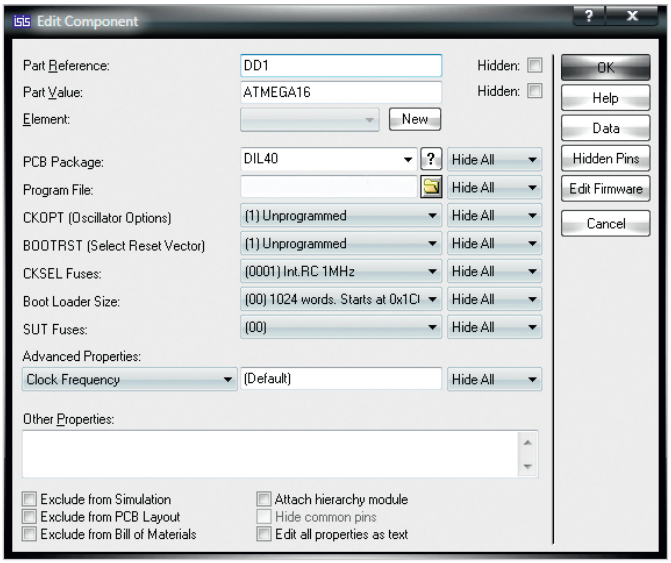


Рис. 21. Настройка параметров микроконтроллера ATmega16 при передаче данных между тремя устройствами через интерфейс SPI

Передача данных через интерфейс SPI между тремя микроконтроллерами AVR

Рассмотрим процесс передачи данных через интерфейс SPI между несколькими микроконтроллерами ATmega16, для чего создадим новый схемный проект и добавим в рабочее поле три таких микросхемы, два светодиода, два резистора (100 Ом), два символа «земли». При этом микросхема DD1 будет выполнять роль ведущего микроконтроллера, а микросхемы DD2 и DD3 – ведомых. Соединим компоненты, как показано на рис. 20, и напишем на языке программирования С программный код управления передачей данных. Программа инициализации пишется как для ведущего, так и для обоих ведомых микроконтроллеров.

Задача ведущего контроллера – послать управляющий сигнал (кодовую комбинацию) сначала первому ведомому, а затем – второму. Переключение между ведомыми выполняется путём установки ведущим микроконтроллером логического нуля на линии *SS* ведомых микроконтроллеров. При передаче данных по интерфейсу SPI между тремя микроконтроллерами в нашем примере этот сигнал выдаётся на линии PB0, PB1 порта PB ведущего микроконтроллера. Задача каждого ведомого – принять кодовую комбинацию, после чего запустить цикл, в котором выполняется последовательное включение и выключение светодиода.

Для удобства соединения в рабочей области проекта отразим по горизонтали микросхему DD1. В окне настроек Edit Component для каждого микроконтроллера установим следующие параметры (см. рис. 21):

- поле CKOPT (Oscillator Options) – (1) Unprogrammed;
- поле BOOTRST (Select Reset Vector) – (1) Unprogrammed;
- поле CKSEL Fuses – (0001) Int.RC 1MHz;
- поле Boot Loader Size – (00) 1024 words. Starts at 0x1C00;
- поле SUT Fuses – (00);
- поле Advanced Properties – Clock Frequency (Default).

Окно настроек открывают двойным щелчком левой кнопки мыши по выбранному на схеме микроконтроллеру.

Напишем на языке программирования С следующий код программы инициализации для ведущего микроконтроллера:

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
int main()
{
PORTB=0b00000000; // инициализация порта PB микросхемы DD1
DDRB=0b10100011;  //  указываем направление передачи информации по линиям порта
// линии MOSI, SCK, PB0, PB1 установлены как выходы
SPCR=0b01010011; // инициализация SPI
SPSR=0b00000000;
PORTB=0b00000010; // выбираем для передачи данных по SPI первый ведомый МК
SPDR=0b11111110;  //  отсылаем кодовую комбинацию для первого ведомого МК
_delay_ms(1000); // задержка
PORTB=0b00000001; // выбираем для передачи данных по SPI второй ведомый МК
SPDR=0b11111110;  //  отсылаем кодовую комбинацию для второго ведомого МК
_delay_ms(1000);
return 0;
}
```

Для ведомого микроконтроллера DD2 был написан следующий код программы инициализации:

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
int main()
{
PORTB=0b00000000; // инициализация порта PB микросхемы DD2
DDRB=0b00000000; // линии порта PB работают как входы
PORTD=0b00000000; // инициализация порта PD микросхемы DD2
DDRD=0b11111111; // линии порта PD работают как выходы
SPCR=0b01000011; // инициализация SPI
while (1) // бесконечный цикл
{
```

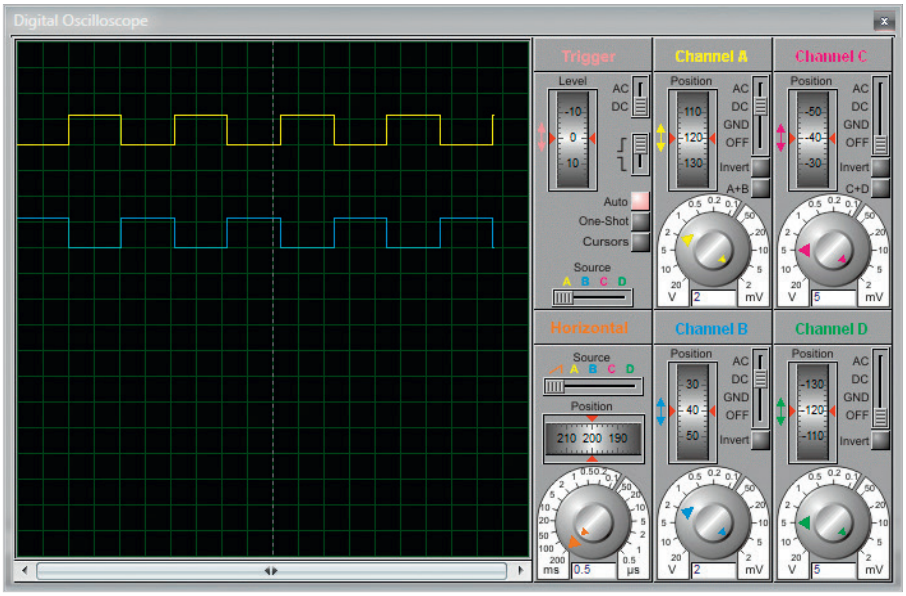


Рис. 22. Осциллограмма работы светодиодов D1 и D2, подключённых к ведомым микроконтроллерам ATmega16

```
  if (SPDR!=0b11111110) // если кодовая комбинация не получена
  PORTD=0b00000000; // посылаем на все линии порта PD - 0
  else if (SPDR==0b11111110) // если кодовая комбинация получена
  {PORTD=0b00000001; // включить светодиод D1
  _delay_ms(1000); // задержка
  PORTD=0b10000000; // погасить светодиод D1
  _delay_ms(1000); } // задержка
  }}
```

Код программы инициализации для ведомого микроконтроллера DD3:

```
  #include <inttypes.h>
  #include <avr/io.h>
  #include <avr/interrupt.h>
  #include <avr/sleep.h>
  #include <util/delay.h>
  int main()
  {
  PORTB=0b00000000; // инициализация порта PB микросхемы DD3
  DDRB=0b00000000; // линии порта PB работают как входы
  PORTD=0b00000000; // инициализация порта PD микросхемы DD3
  DDRD=0b11111111; // линии порта PD работают как выходы
  SPCR=0b01000011; // инициализация SPI
  while (1) // бесконечный цикл
  {
  if (SPDR!=0b11111110) // если кодовая комбинация не получена
  PORTD=0b00000000; // посылаем на все линии порта PD - 0
  else if (SPDR==0b11111110) // если кодовая комбинация получена
  {PORTD=0b00000001; // включить светодиод D2
  _delay_ms(1000); // задержка
  PORTD=0b10000000; // погасить светодиод D2
  _delay_ms(1000); } // задержка
  }}
```

Код программы инициализации вводится на вкладке Source Code схемного редактора на отдельной закладке для каждого микроконтроллера. После того как в рабочей области проекта собрана схема, а на вкладке Source Code введён код программы, можно запускать моделирование. В результате, если компилятор не обнаружит ошибки в программе, на диске компьютера в рабочей папке проекта будут созданы для каждого микроконтроллера файлы *.elf , *.hex и *.c.

Проанализируем работу демонстрационной схемы, представленной на рис. 20. На вкладке Source Code программно были даны указания ведущему микроконтроллеру через интерфейс SPI отправить каждому ведомому микроконтроллеру кодовую комбинацию. Это действие выполняется последовательно. Сначала ведущий микроконтроллер через линию PB0 своего порта PB подаёт на линию *SS* микросхемы DD2 логический ноль, а через линию PB1 на линию *SS* микросхемы DD3 логическую единицу, что оповещает первое ведомое устройство о том, что именно оно выбрано для обмена данными с ведущим устройством по интерфейсу SPI, активизирует интерфейс SPI микросхемы DD2 и делает неактивным интерфейс SPI микросхемы DD3.

После задержки ведущий микроконтроллер через линию PB1 своего порта PB подаёт на линию *SS* микросхемы DD3 логический ноль, а через линию PB0 на линию *SS* микросхемы DD2 – логическую единицу, что активизирует интерфейс SPI микросхемы DD3 и делает неактивным интерфейс SPI микросхемы DD2.

Если ведомое устройство выбрано ведущим, то программа ведомого микроконтроллера выводит на линии порта PD все нули, в результате чего подключённый к порту светодиод будет погашен. Как только по интерфейсу SPI получена кодовая комбинация от ведущего микроконтроллера, запускается подпрограмма, дающая указания ведомому микроконтроллеру запустить цикл, в котором последовательно выводятся на линию порта PD0 значения логической 1 и 0. Эти значения удерживаются при помощи команды задержки.

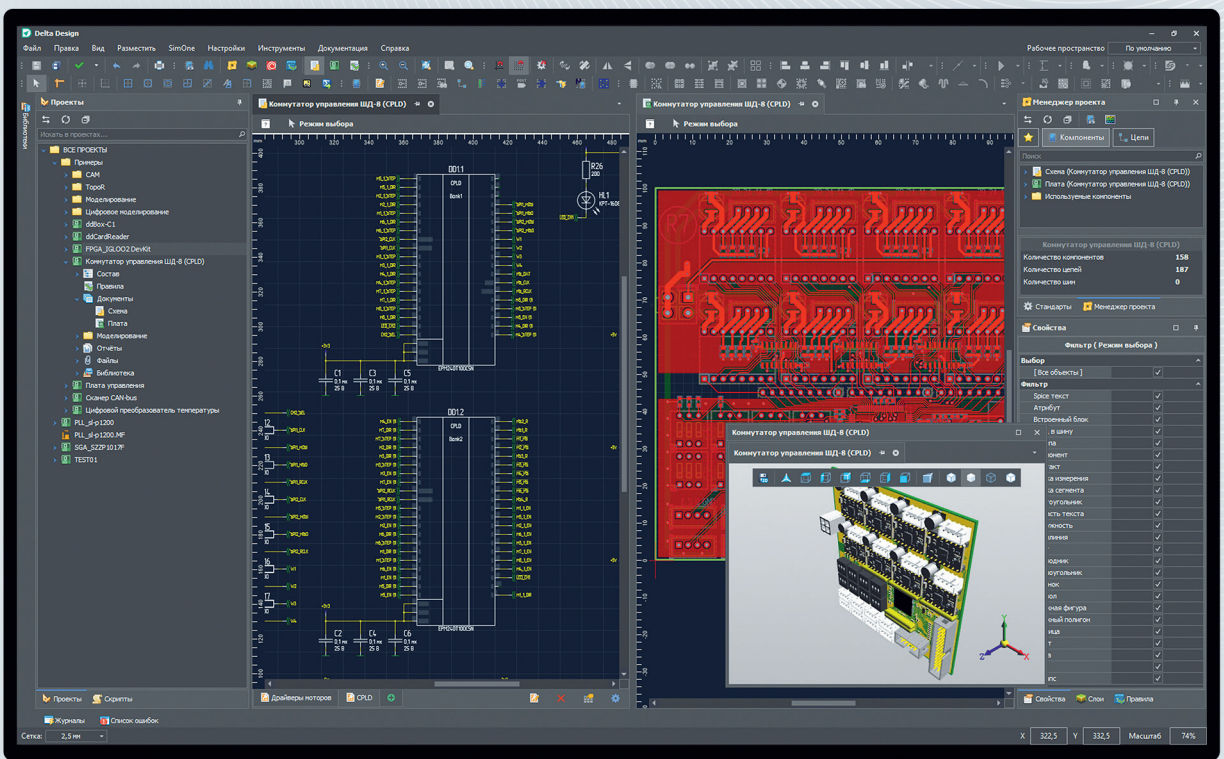
После запуска моделирования при помощи двух светодиодов, подключённых к линиям PD0 порта PD ведомых микроконтроллеров DD2 и DD3, мы можем проверить правильность работы программы – светодиоды подсвечиваются и гаснут поочерёдно, что наглядно демонстрирует осциллограмма, представленная на рис. 22. В момент времени, когда на выводе PD0 микроконтроллера DD2 единица, на выводе PD0 микроконтроллера DD3 – ноль.

Литература

1. STM32F103x4, STM32F103x6 MCU Datasheet. STMicroelectronics. 2009.
2. Proteus VSM Help. Labcenter Electronics. 2020.
3. STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs. Reference manual. STMicroelectronics. 2010.
4. *Колесникова Т.* Работа с универсальным синхронно/асинхронным приёмо-передатчиком USART в программной среде Proteus 8.11 // Современная электроника. 2021. № 8. С. 34.
5. 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash. ATmega16, ATmega16L. Atmel Corporation. 2010.

DeltaDesign 3.5

Новая версия российской САПР электроники



Импорт данных из Mentor PADS и Altium Designer

Поддержка нескольких вариантов трассировки

Обновленный интерфейс в новой оболочке

Новая функциональность схемотехнического редактора и редактора печатных плат

Чтобы получить консультацию по новой версии и внедрить САПР Delta Design на вашем предприятии, обратитесь к специалистам Эремекс

EREMEX +7 (495) 232-18-64 • sales@eremex.ru • www.eremex.ru

Реклама