Хартов В.Я.

Методические указания
Для проведения лабораторных работ по курсу
Микропроцессорные системы

Лабораторная работа №5. Работа последовательного канала SPI

Цель работы - изучить организацию приёма и передачи информации по последовательному каналу SPI (Serial Peripheral Interface).

Интерфейс SPI используется для организации высокоскоростного канала связи между микроконтроллером и различными периферийными устройствами, а также для обмена данными между микроконтроллерами.

В состав модуля SPI (рис.1) входят:

8-разрядный сдвиговый регистр SPDR, который принимает байт данных с шины данных микроконтроллера, сдвигает его вправо или влево с выдачей последовательного кода на вывод микроконтроллера, одновременно с выводом принимает последовательный код со входа микроконтроллера и через буферный регистр передает его в шину данных микроконтроллера;

8-разрядные регистр управления SPCR и регистр состояния SPSR;

предварительный делитель частоты и схемы управления.

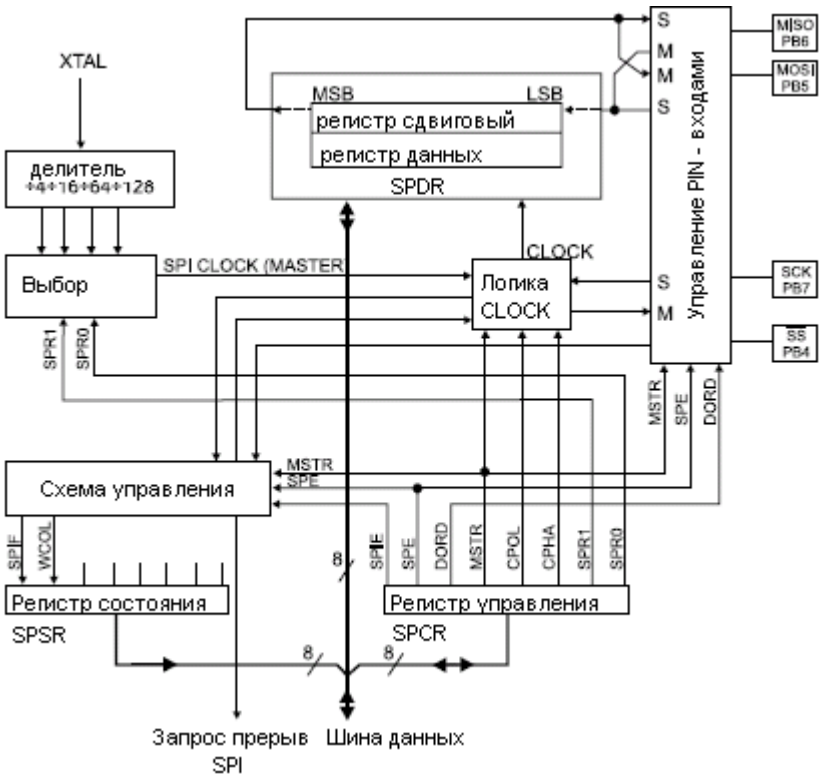


Рис.1. Структурная схема SPI

В микроконтроллерах ATx8515 для сигналов интерфейса SPI выделены 4 линии порта PB: PB5 - MOSI, PB6 - MISO, PB7 - SCK, PB4 - /SS.

Порт SPI может работать в режиме ведущего (*master*) или ведомого (*slave*). Выбор режима определяется установкой бита MSTR управляющего слова SPCR (табл.1).

Таблица 1. Управляющее слово SPCR порта SPI

№ разряда	7	6	5	4	3	2	1	0
Имя	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

При MSTR=1 порт SPI работает в режиме ведущего. При этом вывод MOSI является выходом данных, вывод MISO – входом данных, вывод SCK – выходом для импульсов, используемых в качестве сдвиговых при приеме данных в порту ведомого микроконтроллера. Функция вывода /SS зависит от состояния разряда DDRB.4. При DDRB.4=1 вывод /SS порта SPI не подключен к выводу порта PB4, при DDRB.4=0 значение сигнала на входе влияет на работу порта SPI. Если PB4(/SS)=1 порт работает в режиме ведущего, а при появлении сигнала «0» переключается в режим ведомого. Перевод порта SPI в рабочее состояние осуществляется путем установки бита SPE регистра SPCR.

При MSTR=0 порт SPI работает в режиме ведомого. При этом вывод MOSI является входом данных, вывод MISO – выходом данных, вывод SCK – входом для импульсов сдвига, вывод /SS – входом. Выводы SPI подключаются к выводам порта PB также при установке бита SPE регистра SPCR. Перевод порта в рабочее состояние по сигналу логического «0» на входе /SS.

На рис.2 приведена схема соединения двух микроконтроллеров для обмена данными по каналу SPI.

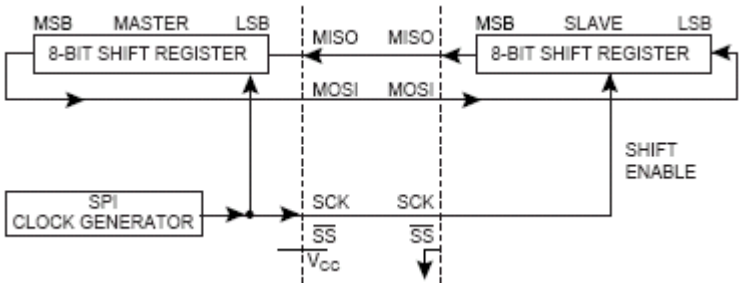


Рис.2. Схема соединения 2-х микроконтроллеров

Передача данных начинается после записи данных в регистр SPRD ведущего микроконтроллера. Диаграммы сигналов при передаче данных по интерфейсу SPI. приведены на рис. 3.

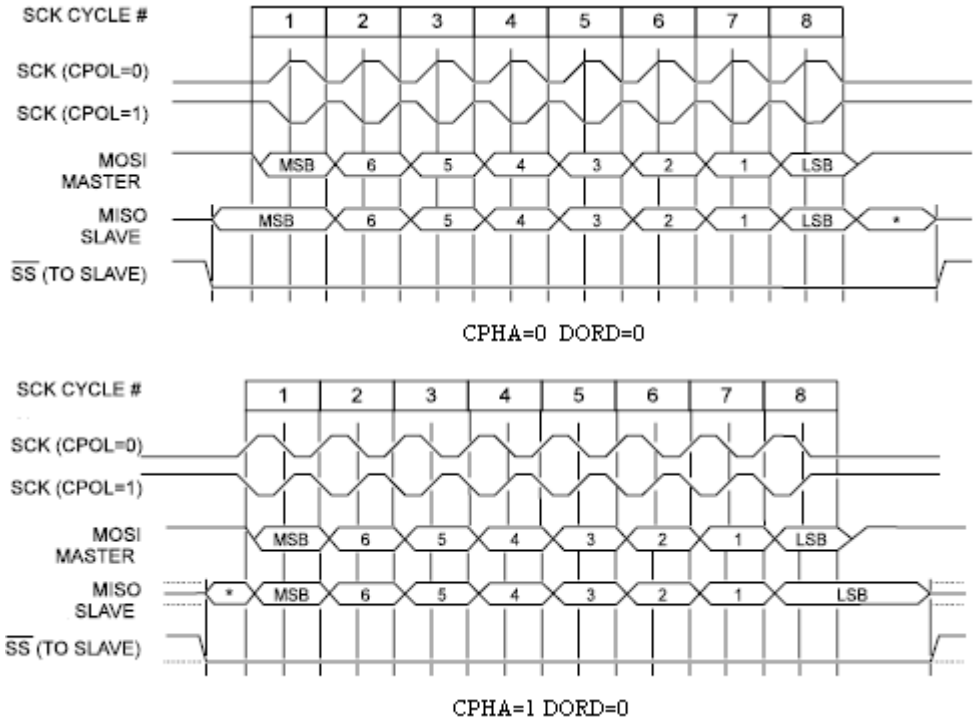


Рис.3. Временные диаграммы сигналов интерфейса SPI

Порядок выдачи определяется состоянием бита DORD при настройке канала. Если DORD=1, вывод начинается с младшего разряда (LSB), иначе – со старшего(MSB). После выдачи последнего разряда устанавливается в «1» флаг SPIF (бит 7 регистра состояния SPSR) и одновременно вырабатывается запрос прерывания SPI STC, если флаг разрешения прерывания SPIE от модуля SPI при настройке канала был установлен в «1». Одновременно с передачей производится прием данных от ведомого микроконтроллера. По окончании приема данных в ведомом микроконтроллере также устанавливается в «1» флаг SPIF и вырабатывается запрос прерывания.

Скорость передачи устанавливается для ведущего микроконтроллера с помощью битов SPR1, SPR0 регистра SPCR. Используемые для сдвига импульсы вырабатываются в результате деления тактовой частоты CK на коэффициент K согласно табл.2. Значения этих битов для ведомого МК не оказывают влияния на работу порта. Значения битов DODR (формат обмена) и CPOL (полярность сигналов сдвига на линии SCK) для ведущего и ведомого микроконтроллеров должны быть одинаковыми.

Таблица 2. Выбор коэффициентов деления K

SPR1	**SPR0**	**K**
0	0	4
0	1	16
1	0	64
1	1	128

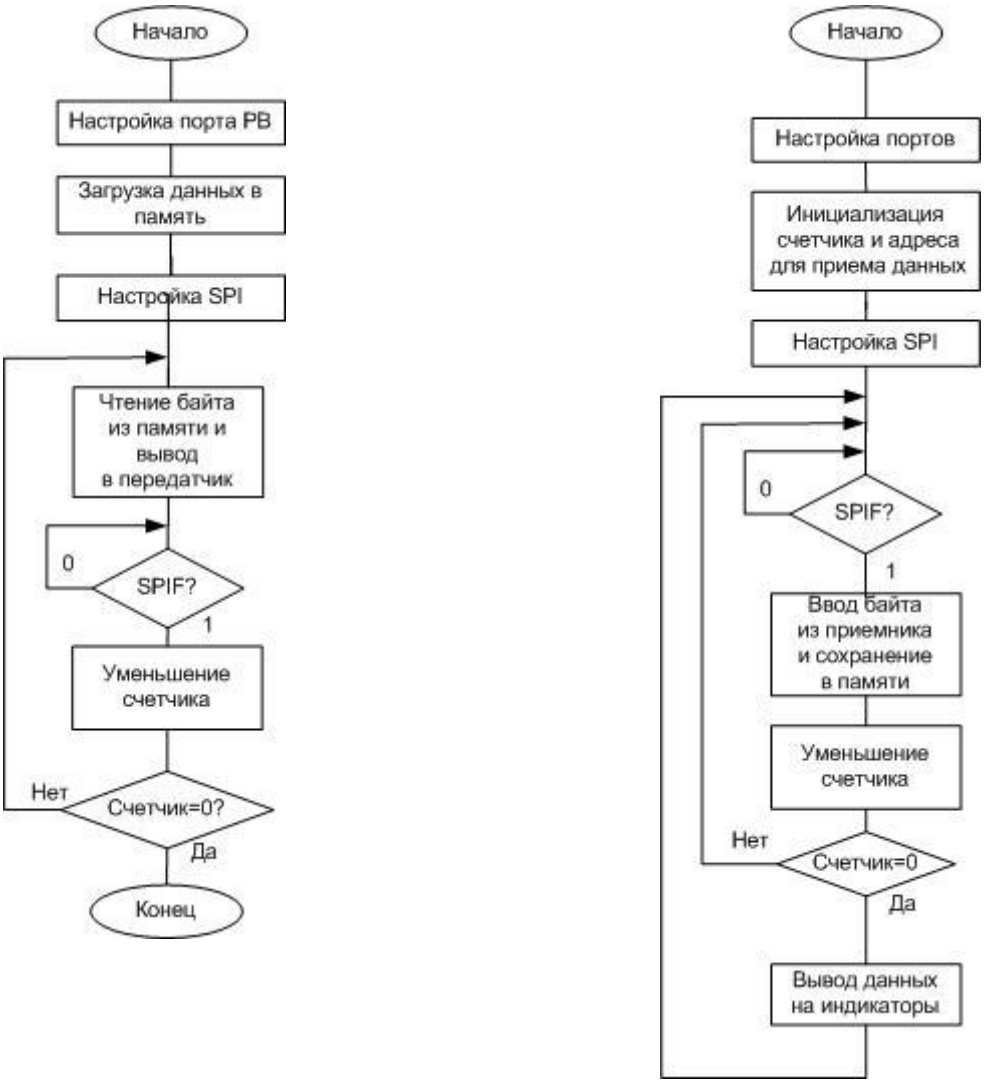
Практическая часть

Подготовить программу для исследования передачи и приёма по последовательному каналу SPI. Напомним, что в качестве выхода передатчика используется вывод ведущего микроконтроллера MOSI (PB5), а в качестве входа приёмника ведомого микроконтроллера – вывод MOSI (PB5).

Для контроля работы канала SPI используем два стартовых набора STK500. Запрограммируем микроконтроллер первого STK500 для передачи данных, микроконтроллер второго - для приема. Данные для передачи, три байта, загрузим в ячейки памяти SRAM,

начиная с адреса $170, используя команды для записи констант. Затем выполним в цикле последовательный вывод данных. После приема каждого байта данных второй микроконтроллер сохраняет его в своей памяти SRAM, начиная с адреса $180. Закончив прием, второй микроконтроллер последовательно выводит при нажатии кнопки “Просмотр” полученные данные на светодиоды.

Схемы алгоритмов передачи и приема приведены рис.4. Переход к следующей итерации в циклах передачи (приема) осуществляется после установки флага завершения передачи (приема) SPIF очередного байта данных. После завершения цикла приёма включаются все светодиоды линейки индикации, сигнализируя об окончании приема. Далее, при последовательном нажатии на кнопку SW5 (Просмотр) осуществляется вывод принятых байтов данных на светодиоды. Ниже приведены тексты программ для передающего и принимающего микроконтроллеров.



а) б)

Рис. 4. Схема алгоритма передачи (а) МК1 и приема (б) МК2

Программа 1

```
;***********************************************************
;Программа 1 для демонстрации работы канала SPI
;для передающего микроконтроллера ATx8515 в режиме MASTER.
;После сброса  МК1 происходит передача трёх байтов,
;считываемых из ячеек SRAM по адресам из регистра Z
;Соединения: PB5мк1-PB5мк2, PB7мк1-PB7мк2, PB0мк1-PB4мк2
;***********************************************************
.include "8515def.inc"          ;файл определений AT90S8515
;.include "m8515def.inc"        ;файл определений ATMEGA8515
.equ DD_MOSI = 5
.equ DD_SCK = 7
.def temp = r16                 ;временный буфер
```

```
.def count = r17                ;счётчик
.org $000
          rjmp init
;***Инициализация МК
INIT:     ldi temp,0xB1         ;DD_MOSI, DD_SCK,
SS, PB0 для вывода
          out DDRB,temp
          ldi ZL,0x70           ;загрузка
          ldi ZH,0x01           ; данных в
          ldi temp,0x41         ;  память
          st Z+,temp            ;   данных
          ldi temp,0x56         ;    с использованием
          st Z+,temp            ;     косвенной
          ldi temp,0x52         ;      адресации с
          st Z+,temp            ;       постинкрементом
          ldi count,0x03        ;установка счётчика передач

;***Настройка SPI в режиме MASTER на передачу данных
          ldi temp,(1<<SPE)|(1<<MSTR)
          out SPCR,temp
OUTPUT:   ldi temp,0x41         ;переключение
          out PORTB,temp        ;ss из 1 в 0
          ldi temp,0x40
          out PORTB,temp
          ld temp,-Z            ;считывание байта из памяти
          out SPDR,temp         ;вывод байта в передатчик
Wait_Transmit:
          sbis SPSR,SPIF        ; проверка флага передачи
          rjmp Wait_Transmit
          dec count             ;уменьшение счётчика на 1
          brne OUTPUT
loop:     rjmp loop
```

Программа 2

```
;*****************************************************************
;Программа 2 для демонстрация работы канала SPI
;микроконтроллера ATx8515 в режиме SLAVE.
;После сброса  МК2 происходит прием трёх байтов, записываемых в SRAM
;по адресам из регистра X.
```

```
;По окончании приёма загораются все светодиоды.
;При последовательном нажатии на SW5 (SHOW) происходит чтение данных
;и вывод их на светодиоды.
;Соединения: SW5-PD5, шлейфом порт PC-LED
;*****************************************************************
.include "8515def.inc"          ;файл определений AT90S8515
;.include "m8515def.inc"        ;файл определений ATMEGA8515
.equ DD_MISO = 6
.def temp = r16                 ;временный буфер
.def count = r17                ;счётчик
.equ SHOW = 5                   ;5-й вывод порта PD

.org $000
          rjmp init
;***Инициализация МК
     INIT:
          ldi temp,low(RAMEND)  ;установка
          out SPL,temp          ; указателя стека
          ldi temp,high(RAMEND) ;  на последнюю
          out SPH,temp          ;   ячейку ОЗУ
          ldi temp,(1<<DD_MISO)
          out DDRB,temp
          ldi temp,0xB0
          out PORTB,temp
          clr temp              ;настройка
          out DDRD,temp         ; вывода
          ldi temp,0x20         ;  порта PD5
          out PORTD,temp        ;   на ввод
          ser temp              ;настройка
          out DDRC,temp         ; выводов порта PC
          out PORTC,temp        ;  на вывод
          ldi count,0x03        ;установка счётчика байтов
          ldi XL,0x80           ;в регистре X адрес, по которому
          ldi XH,0x01           ; происходит запись принятых данных
;***Настройка SPI в режиме SLAVE на приём данных
          ldi temp,(1<<SPE)
          out SPCR,temp
INPUT:    sbis SPSR,SPIF        ;проверка флага приема
          rjmp INPUT
```

```
            in temp,SPDR          ;ввод байта из приёмника
            st X+,temp            ;сохранение байта в памяти
            dec count
            brne INPUT            ;уменьшение счётчика на 1
            clr temp              ;сигнализация - передача и
            out PORTC,temp        ; приём завершены
            rcall DELAY           ;задержка
            ldi count,3           ;установка счётчика байтов
WAIT_SHOW:  sbic PIND,SHOW        ;ожидание нажатия
            rjmp WAIT_SHOW        ; кнопки SHOW
            ld temp, -X           ;считывание байта из памяти
            com temp              ;инвертирование и
            out PORTC,temp        ;вывод на светодиоды
            rcall DELAY           ;задержка
            dec count             ;если показаны не все данные,
            brne WAIT_SHOW        ; то продолжение по нажатию SHOW
            rjmp INPUT
;***Задержка***
DELAY:      ldi r19,10
            ldi r20,255
            ldi r21,255
dd:         dec r21
            brne dd
            dec r20
            brne dd
            dec r19
            brne dd
            ret
```

Задание 1. Создать в AVR Studio 4 проект для передачи данных с помощью программы 1. Проверить работу программы в режиме симуляции, наблюдая состояния регистров и битов состояния канала SPI. Сохранить файл с отлаженной программой (*File/Save As...*).

Создать в AVR Studio 4 проект для приема данных с помощью программы 2. Закомментировав команды вызова подпрограммы задержки (rcall DELAY) проверить работу программы в режиме симуляции в шаговом режиме, наблюдая состояния регистров и битов

состояния канала SPI и вручную обновляя флаг SPIF и регистр SPDR перед вводом данных. Сохранить файл с отлаженной программой (*File/Save As...*).

Для совместной отладки программ и симуляции передачи/приема воспользуемся демонстрационной версией программы ISIS 6 Professional из пакета Proteus 6 Professional фирмы Labcenter Electronic с сайта http://www.labcenter.co.uk.

Создаем проект для схемы на рис. 5. Микроконтроллер МК1 (U1) работает в режиме master (ведущий), микроконтроллер МК2 (U2) в режиме slave (ведомый).

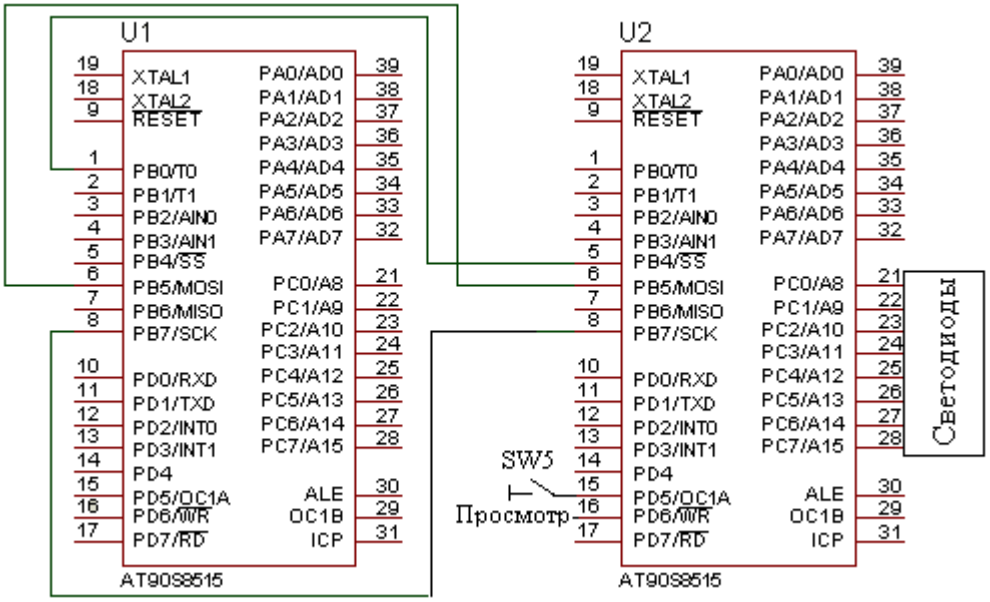


Рис.5. Схема взаимодействия микроконтроллеров по интерфейсу SPI

Выбрав из библиотеки компонентов *Component/**P**ickDevices/Micro* микроконтроллер AT90S8515, в окно редактора вводим два микроконтроллера, которые соединяем линиями связи. Добавляем периферийные устройства: переключатель (SW-SPST) из библиотеки компонентов *Component/Active* и при желании светодиоды, так как выводы всех портов индицируются программой автоматически. Присоединяем выводы переключателя: один - к выводу порта PD5, второй – к общей шине GROUND, выбрав ее из списка Inter-sheet Terminal на панели инструментов. Предварительно выделив правой кнопкой мыши переключатель и щелкнув левой кнопкой, в открывающемся окне назначаем ему новое имя SW5. Затем, выделив на схеме правой клавишей мыши условное графическое обозначение первого

микроконтроллера, с помощью команд меню *Source/(Add/Remove Source files...)* добавляем файл с программой передачи. Для этого воспользуемся подготовленным в AVR Studio 4 файлом с расширением *.asm. Такую же операцию повторяем для второго микроконтроллера. Выполняем совместную компиляцию программ командой *Source/Build All.* При отсутствии ошибок переходим к отладке, выполнив команду меню *Debug/(Start/Restart Debugging)*. Открыв окна *I/O Registers, Internal RAM, Source Code* для обоих AVR устройств (U1, U2) выполняем пошаговую отладку, нажимая клавишу *F11*, наблюдая за состоянием регистров SPI и ячеек SRAM и замыкая/размыкая SW5 для вывода данных в порт PC (для светодиодов).

Задание 2. Изменить обе программы, задав CPOL=1. Проверить работу интерфейса на модели путем симуляции. Обратить внимание на изменение полярности данных на линии MOSI.

Задание 3. Изменить обе программы, задав DORD=1. Проверить работу интерфейса на модели. Обратить внимание на изменение порядка вывода битов данных на линии MOSI, начиная с младшего разряда.

Задание 4. Изменить проект, подключив к входу $PB4_{MK2}$ уровень GND (логический «0»). Проверить работу интерфейса на модели. Обратить внимание на состояние флага SPIF ведомого микроконтроллера.

Задание 5. Убедившись в правильной работе программ, можно перейти к экспериментальной проверке в STK500.

Выполним программирование микроконтроллеров МК1 и МК2, используя два набора STK500. Перед программированием МК2 (программой приема 5.5) восстановить ранее закомментированные команды (rcall delay). Отключив питание, выполним необходимые соединения между портами микроконтроллеров: пары $(PB5,PB7)_{MK1}$ и $(PB5,PB7)_{MK2}$, $PB0_{MK1}$ - $PB4_{MK2}$, обеспечивающие передачу данных в одном направлении - из МК1 в МК2. Соединим между собой выводы GND обоих плат. На плате STK500-2 соединим шлейфом PC – LED, PD5-SW5. Включив питание, кнопкой RESET запустим программу микроконтроллера МК2 для

приема данных, переведя его в режим ожидания установки флага приемника SPIF. Затем, запустив кнопкой RESET программу микроконтроллера МК1, выполняем передачу. По окончании приема (светодиоды включены) проверяем работу интерфейса. Нажимая кнопку SW5 на плате STK500 второго МК, наблюдаем на индикаторах принятые байты данных. Повторить передачу несколько раз, выполняя кнопкой RESET сброс микроконтроллера МК1 и просматривая принимаемые данные.