SPI-ИНТЕРФЕЙС

В состав всех AVR-входит модуль SPI-интерфейса. Он предназначен для подключения различного периферийного оборудования (АЦП, ЦАП, микросхемы flash-памяти). Рассмотрим программирование этого модуля, а также приведем примеры односторонней и двусторонней передачи данных между двумя микроконтроллерами.

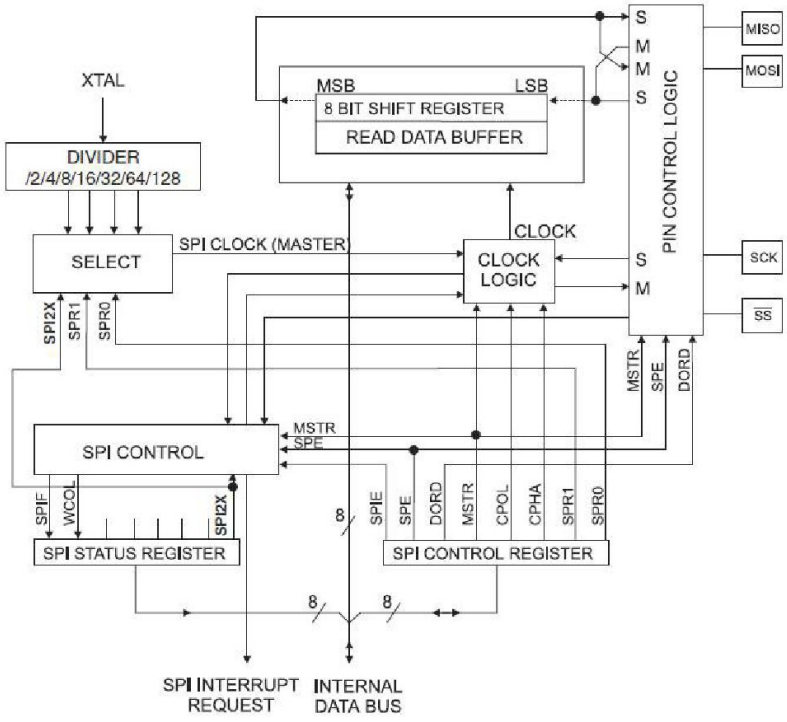


Рис.1 блок-схема модуля интерфейса SPI

Для настройки интерфейса необходимо сконфигурировать три регистра – SPI STATUS REGISTER (SPSR), SPI CONTROLREGISTER (SPCR), и SPI Data Register (SPDR).

SPI CONTROL REGISTER (SPCR):

7	6	5	4	3	2	1	0	
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Бит 7 – SPIE: SPI Interrupt Enable,
Этот бит разрешает прерывания от SPI, если будет установлен SPIF в регистре SPSR, и бит I в регистре SREG (общее разрешение прерываний), будет вызвана продпрограмма обработки прерывание SPI_STC .
-0 прерывания запрещены
-1 – разрешение прерывания.
Бит 6 – SPE: SPI Enable
Этот бит разрешает работу SPI,
-0 в этом разряде означает запрет любых операций в SPI
-1 в этом разряде – разрешение работы.
Бит 5 – DORD: DORD: Data Order
Этот бит определяет порядок передачи данных,
-0 в этом разряде – сначала будет передаваться младший разряд
-1 в этом разряде – сначала будет передаваться старший разряд.
Бит 4 – MSTR: Master/Slave Select
Этот бит определяет режим работы модуля,
-1 в этом разряде – режим ведущего
-0 в этом разряде – режим ведомого.
Следующие 4 разряда определяют параметры тактового сигнала
Бит 3 – CPOL: Clock Polarity
Полярность сигнала синхронизации,
-1 в этом разряде – тактовый сигнал в состоянии ожидания - 1
-0 в этом разряде – тактовый сигнал в состоянии ожидания - 0

CPOL	Передний фронт	Задний фронт
0	Нарастающий	Спадающий
1	Спадающий	Нарастающий

Бит2 - CPHA: Clock Phase
Фаза сигнала синхронизации – определяет, по какому фронту будет производиться установка (выборка) - по переднему или заднему,
-1 в этом разряде – установка (выборка) по переднему фронту
-0 в этом разряде – установка (выборка) по заднему фронту

CPHA	Передний фронт	Задний фронт
0	Выборка	Установка
1	Установка	Выборка

Биты 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0
Совместно с битом SPI2x в регистре SPSR задают скорость передачи данных

SPI2X	SPR1	SPR0	Частота SCK
0	0	0	fosc /4
0	0	1	fosc /16
0	1	0	fosc /64
0	1	1	fosc /128

1	0	0	fosc /2
1	0	1	fosc /8
1	1	0	fosc /32
1	1	1	fosc /64

SPI STATUS REGISTER (SPSR).

7	6	5	4	3	2	1	0	
SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
R	R	R	R	R	R	R	R/W	
0	0	0	0	0	0	0	0	

Бит 7 – SPIF: SPI Interrupt Flag
Флаг завершения передачи - устанавливается после завершения передачи (приема) и вызывает прерывание (как у ведущего, так и у ведомого). В вызванном прерывании можно считывать и записывать новые данные для передачи. Сбрасывается автоматически при входе в подпрограмму обработки прерывания.
Бит6 – WCOL: Write COLlision flag
Флаг наложения записи – устанавливается при чтении либо записи данных при незавершенной передаче. Автоматически сбрасывается чтением регистра SPSR а также при чтении (записи) в регистр данных (SPDR).
Бит0 – SPI2X: Double SPI Speed Bit
Совместно с битами SPR1 и SPR0 в регистре SPCR задает скорость передачи данных, при его установки скорость работы SPI(частота SCK) удвоится, если SPI находится в режиме ведущего. Это означает, что минимальный период SCK будет равен двум периодам fosc. Если SPI работает как ведомый, то работа SPI гарантирована только на частоте fosc /4 или менее.
SPI Data Register – SPDR

7	6	5	4	3	2	1	0	
MSB							LSB	SPDR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

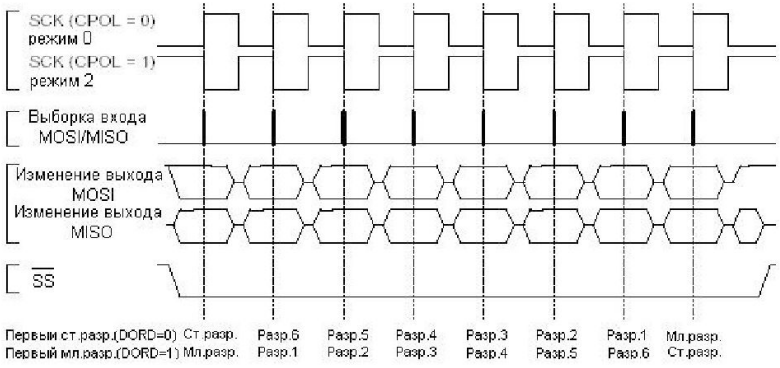
Регистр данных SPI, запись в данный регистр инициирует передачу данных. При чтении данного регистра фактически считывается содержимое приемного буфера сдвигового регистра.

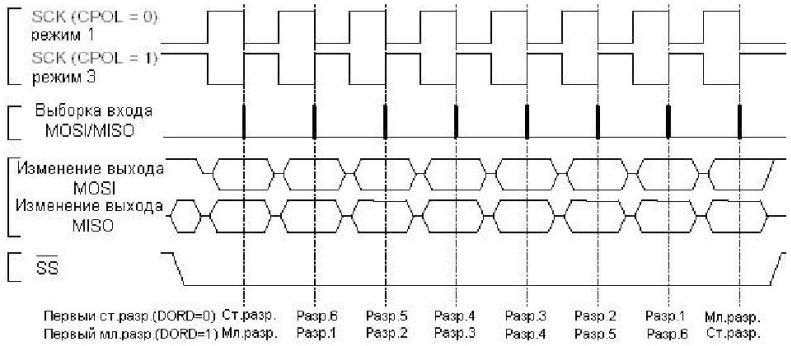
Режимы передачи данных

Комбинация бит CPHA и CPOL задает четыре возможных режима последовательной передачи данных. Форматы передачи данных для SPI представлены в таблице 73, а их временные диаграммы показаны на рис. 77 и 78. Биты данных выводятся сдвигом и фиксируются на входе противоположными фронтами синхросигнала SCK, тем самым гарантируя достаточное время на

установление сигналов данных. Таким образом, можно обобщить информацию из табл. 70 и 71 и представить ее в следующем виде:

	Передний фронт	Задний фронт	Режим SPI
CPOL = 0, CPHA = 0	Выборка нарастающим фронтом	Установка данных падающим фронтом	0
CPOL = 0, CPHA = 1	Установка данных нарастающим фронтом	Выборка падающим фронтом	1
CPOL = 1, CPHA = 0	Выборка падающим фронтом	Установка данных нарастающим фронтом	2
CPOL = 1, CPHA = 1	Установка данных падающим фронтом	Выборка нарастающим фронтом	3





Для нормального подключения необходимо:
Для ведущего настроить MOSI, SCL, SS на выход, MISO на вход.
Для ведомого настроить MOSI, SCL, SS на выход, MISO на выход.
При соединении одноименные выводы подключаются друг к другу

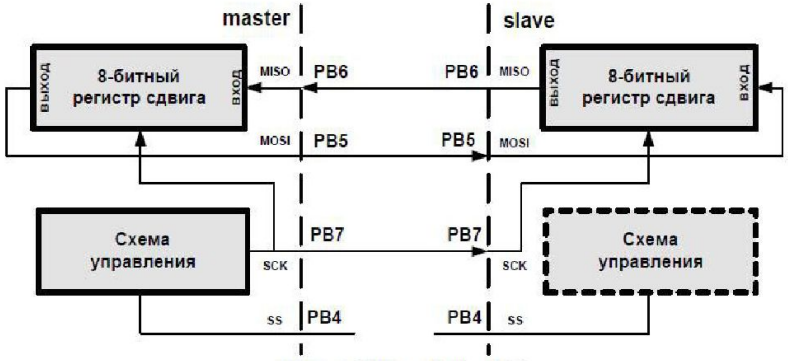
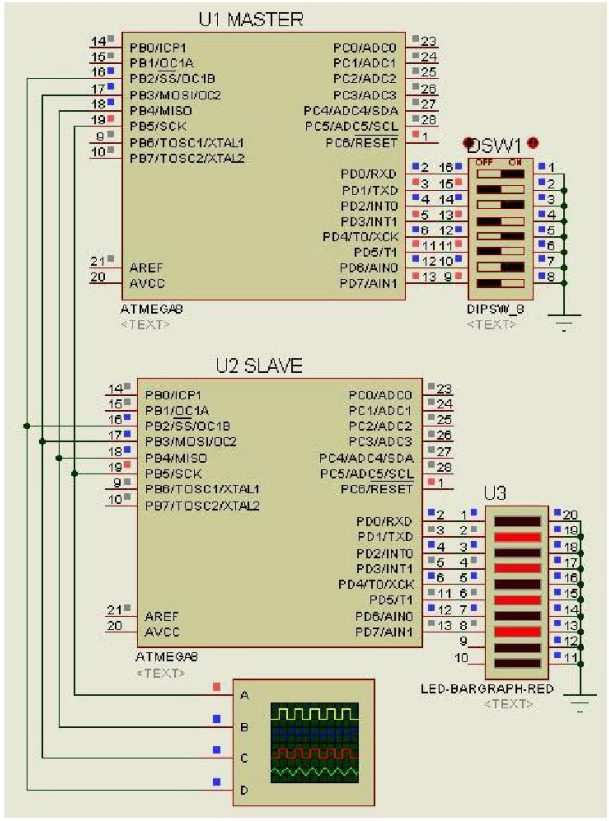
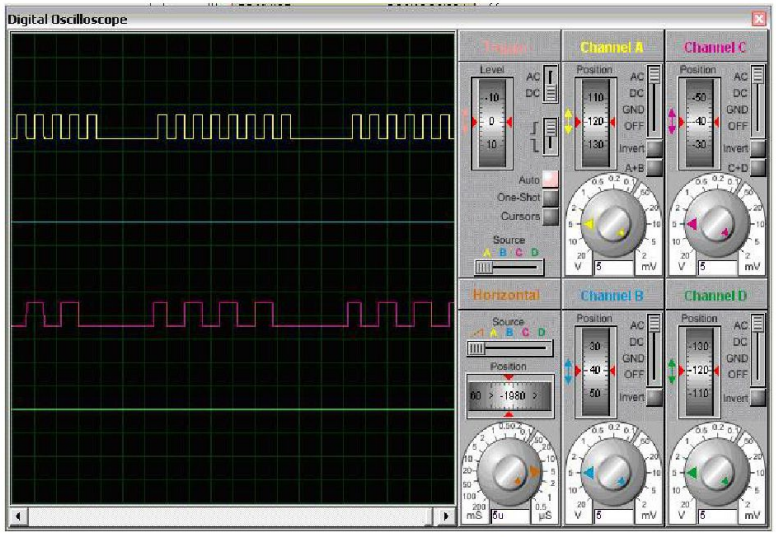


Рис. 3. Интерфейс SPI

1. Примеры программ для обмена данными по SPI.

В данных примерах рассмотрен примеры одностороннего и двустороннего обмена данными .
Работа программ смоделирована в программе PROTEUS 7.5.


U1 MASTER
PB0/ICP1
PB1/OC1A
PB2/SS/OC1B
PB3/MOSI/OC2
PB4/MISO
PB5/SCK
PB6/TOSC1/XTAL1
PB7/TOSC2/XTAL2
AREF
AVCC
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
PC6/RESET
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/T0/XCK
PD5/T1
PD6/AIN0
PD7/AIN1
ATMEGA8
<TEXT>
DSW1
OFF ON
DIPSW_8
<TEXT>
U2 SLAVE
ATMEGA8
<TEXT>
U3
LED-BARGRAPH-RED
<TEXT>
A
B
C
D



Программа ведущего (MASTER):

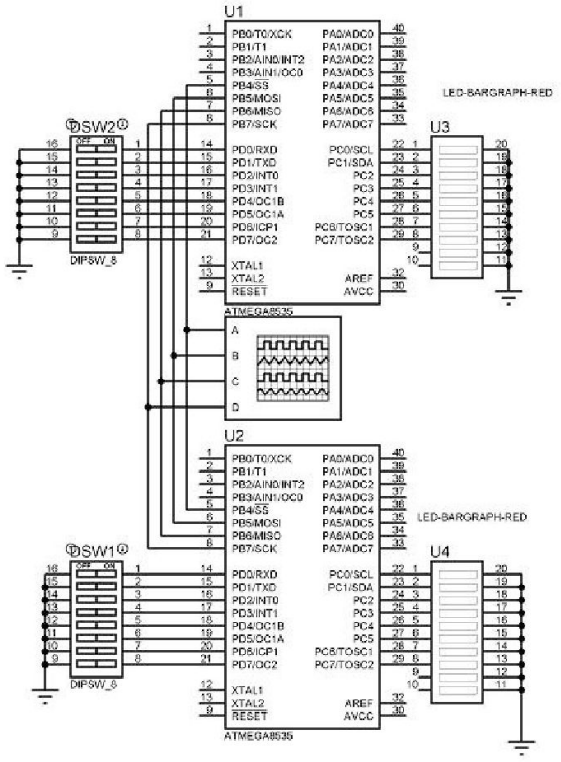
```
.include "C:\Program Files\Atmel\AVR
Tools\AvrAssembler2\Appnotes\m8def.inc"
rjmp RESET ; Reset Handler
reti;rjmp EXT_INT0 ; IRQ0 Handler
reti;rjmp EXT_INT1 ; IRQ1 Handler
reti;rjmp TIM2_COMP ; Timer2 Compare Handler
reti;rjmp TIM2_OVF ; Timer2 Overflow Handler
reti;rjmp TIM1_CAPT ; Timer1 Capture Handler
reti;rjmp TIM1_COMPA ; Timer1 CompareA Handler
reti;rjmp TIM1_COMPB ; Timer1 CompareB Handler
reti;rjmp TIM1_OVF ; Timer1 Overflow Handler
reti;rjmp TIM0_OVF ; Timer0 Overflow Handler
rjmp SPI_STC ; SPI Transfer Complete Handler
reti;rjmp USART_RXC ; USART RX Complete Handler
reti;rjmp USART_UDRE ; UDR Empty Handler
reti;rjmp USART_TXC ; USART TX Complete Handler
reti;rjmp ADC ; ADC Conversion Complete Handler
reti;rjmp EE_RDY ; EEPROM Ready Handler
reti;rjmp ANA_COMP ; Analog Comparator Handler
reti;rjmp TWSI ; Two-wire Serial Interface Handler
reti;rjmp SPM_RDY ; Store Program Memory Ready Han

reset:
ldi r16,(1<<PB3)+(1<<PB5)+(1<<PB2)
out ddrb,r16      ;mosi, SS and sck is output

ldi r16,(1<<PB2)
```

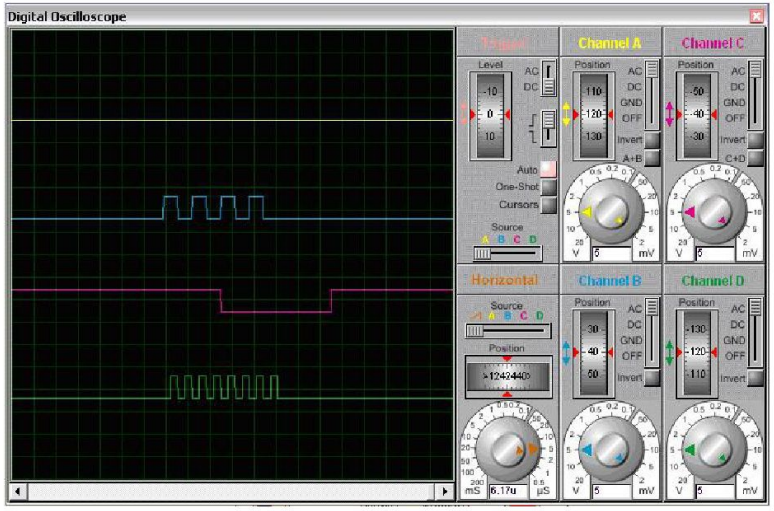
```
out portb,r16

ldi r16,255
out portd,r16     ;input port (key) with pull-up res

ldi r16,(1<<spie)+(1<<spe)+(1<<mstr)
out spcr,r16      ;spi enableand, spi int. enable, master mode


ldi r16,high(RAMEND)
out SPH,r16       ;Set Stack Pointer to top of RAM
ldi r16,low(RAMEND)
out SPL,r16
sei                         ;Enable interrupts

rcall spi_transmit

main:
inc r17
cpi r17,255
brne main
;rcall spi_transmit
;ldi r16,(1<<PB2)
;out portb,r16    ;ss pin->0
rjmp main                   ;Main program start

SPI_STC:
;ldi r16,(1<<spie)+(1<<spe)+(1<<mstr)
;out spcr,r16     ;spi enableand, spi int. enable, master mode

in r16,pind                 ;read keyport
out spdr,r16      ;spi data transmitter write
reti

spi_transmit:
ldi r16,(1<<spie)+(1<<spe)+(1<<mstr)
out spcr,r16      ;spi enableand, spi int. enable, master mode

in r16,pind                 ;read keyport
out spdr,r16      ;spi data transmitter write

ldi r16,(0<<PB2)
out portb,r16     ;ss pin->0
ret
```

Программа ведомого (SLAVE):

```
.include "C:\Program Files\Atmel\AVR
Tools\AvrAssembler2\Appnotes\m8def.inc"
rjmp RESET ; Reset Handler
reti;rjmp EXT_INT0 ; IRQ0 Handler
reti;rjmp EXT_INT1 ; IRQ1 Handler
```

```
reti;rjmp TIM2_COMP ; Timer2 Compare Handler
reti;rjmp TIM2_OVF ; Timer2 Overflow Handler
reti;rjmp TIM1_CAPT ; Timer1 Capture Handler
reti;rjmp TIM1_COMPA ; Timer1 CompareA Handler
reti;rjmp TIM1_COMPB ; Timer1 CompareB Handler
reti;rjmp TIM1_OVF ; Timer1 Overflow Handler
reti;rjmp TIM0_OVF ; Timer0 Overflow Handler
rjmp SPI_STC ; SPI Transfer Complete Handler
reti;rjmp USART_RXC ; USART RX Complete Handler
reti;rjmp USART_UDRE ; UDR Empty Handler
reti;rjmp USART_TXC ; USART TX Complete Handler
reti;rjmp ADC ; ADC Conversion Complete Handler
reti;rjmp EE_RDY ; EEPROM Ready Handler
reti;rjmp ANA_COMP ; Analog Comparator Handler
reti;rjmp TWSI ; Two-wire Serial Interface Handler
reti;rjmp SPM_RDY ; Store Program Memory Ready Han

reset:
ldi r16,(1<<pb4)
out ddrb,r16      ;miso is out
ldi r16,(1<<pb2)
out portb,r16     ;pull-up for ss
ldi r16,255
out ddrd,r16      ;output port (led)

ldi r16,(1<<spie)+(1<<spe)
out spcr,r16      ;spi enable and spi int. enable

ldi r16,high(RAMEND)
out SPH,r16               ; Set Stack Pointer to top of RAM
ldi r16,low(RAMEND)
out SPL,r16
sei                       ; Enable interrupts

main:
rjmp main                  ; Main program loop

SPI_STC:
in r16,spdr                ;read data register
out portd,r16               ;and out to led port
reti
```


U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
LED-BARGRAPH-RED
SW2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
U3
DIPSW_8
XTAL1
XTAL2
RESET
AREF
AVCC
ATMEGA8535
A
B
C
D
U2
SW1
U4
DIPSW_8
ATMEGA8535




```
.include "C:\Program Files\Atmel\AVR
Tools\AvrAssembler2\Appnotes\m8535def.inc"
rjmp RESET ; Reset Handler
reti; rjmp EXT_INT0 ; IRQ0 Handler
reti; rjmp EXT_INT1 ; IRQ1 Handler
reti; rjmp TIM2_COMP ; Timer2 Compare Handler
reti; rjmp TIM2_OVF ; Timer2 Overflow Handler
reti; rjmp TIM1_CAPT ; Timer1 Capture Handler
reti; rjmp TIM1_COMPA ; Timer1 Compare A Handler
reti; rjmp TIM1_COMPB ; Timer1 Compare B Handler
reti; rjmp TIM1_OVF ; Timer1 Overflow Handler
reti; rjmp TIM0_OVF ; Timer0 Overflow Handler
rjmp SPI_STC ; SPI Transfer Complete Handler
reti; rjmp USART_RXC ; USART RX Complete Handler
reti; rjmp USART_UDRE ; UDR Empty Handler
reti; rjmp USART_TXC ; USART TX Complete Handler
reti; rjmp ADC ; ADC Conversion Complete Handler
reti; rjmp EE_RDY ; EEPROM Ready Handler
reti; rjmp ANA_COMP ; Analog Comparator Handler
reti; rjmp TWSI ; Two-wire Serial Interface Handler
reti; rjmp EXT_INT2 ; IRQ2 Handler
reti; rjmp TIM0_COMP ; Timer0 Compare Handler
reti; rjmp SPM_RDY ; Store Program Memory Ready Handler

reset:
ldi r16,(1<<PB4)+(1<<PB5)+(1<<PB7)
out ddrb,r16      ;mosi, SS and sck is output
```

```
ldi r16,255
out portd,r16     ;input port (key) with pull-up res
out ddrc,r16      ;portc is ledport is out

ldi r16,(1<<spie)+(1<<spe)+(1<<mstr)
out spcr,r16      ;spi enableand, spi int. enable, master mode

;in r16,pind       ;read keyport
;out spdr,r16     ;spi data transmitter write

ldi r16,high(RAMEND)
out SPH,r16      ;Set Stack Pointer to top of RAM
ldi r16,low(RAMEND)
out SPL,r16
sei                         ;Enable interrupts


ldi r16,(1<<PB4);ss->1
out portb,r16


ldi r16,(1<<spie)+(1<<spe)+(1<<mstr)
out spcr,r16      ;master mode

main:
inc r17
cpi r17,255
breq spi_transmit
rjmp main                  ;Main program start

delay:
inc r18
brne delay
ret

SPI_STC:

in r16,spdr
out portc,r16

reti


spi_transmit:
ldi r17,0


in r16,pind                ;read keyport
out spdr,r16      ;spi data transmitter write

ldi r16,(0<<PB4)
out portb,r16     ;ss pin->0
```

```
rjmp main
.include "C:\Program Files\Atmel\AVR
Tools\AvrAssembler2\Appnotes\m8535def.inc"
rjmp RESET ; Reset Handler
nop;     rjmp EXT_INT0 ; IRQ0 Handler
nop; rjmp EXT_INT1 ; IRQ1 Handler
nop; rjmp TIM2_COMP ; Timer2 Compare Handler
nop; rjmp TIM2_OVF ; Timer2 Overflow Handler
nop; rjmp TIM1_CAPT ; Timer1 Capture Handler
nop;6 rjmp TIM1_COMPA ; Timer1 Compare A Handler
nop;0x007 rjmp TIM1_COMPB ; Timer1 Compare B Handler
nop;0x008 rjmp TIM1_OVF ; Timer1 Overflow Handler
nop;0x009 rjmp TIM0_OVF ; Timer0 Overflow Handler
rjmp SPI_STC ; SPI Transfer Complete Handler
nop;0x00B rjmp USART_RXC ; USART RX Complete Handler
nop;0x00C rjmp USART_UDRE ; UDR Empty Handler
nop;0x00D rjmp USART_TXC ; USART TX Complete Handler
nop;0x00E rjmp ADC ; ADC Conversion Complete Handler
nop;0x00F rjmp EE_RDY ; EEPROM Ready Handler
nop;0x010 rjmp ANA_COMP ; Analog Comparator Handler
nop;0x011 rjmp TWSI ; Two-wire Serial Interface Handler
nop;0x012 rjmp EXT_INT2 ; IRQ2 Handler
nop;0x013 rjmp TIM0_COMP ; Timer0 Compare Handler
nop;0x014 rjmp SPM_RDY ; Store Program Memory Ready Handler

reset:
ldi r16,(1<<pb6)
out ddrb,r16              ;miso is out

;ldi r16,(1<<pb6)
;out portb,r16             ;pull-up for ss

ldi r16,255
out portd,r16             ;pull-up for keyport

ldi r16,255
out ddrc,r16              ;portc is ledport is out

ldi r16,(1<<spie)+(1<<spe)
out spcr,r16              ;spi enable and spi int. enable


ldi r16,high(RAMEND)
out SPH,r16               ; Set Stack Pointer to top of RAM
ldi r16,low(RAMEND)
out SPL,r16
sei                       ; Enable interrupts

main:
rjmp main                 ; Main program loop
```

```
SPI_STC:

in r16,spdr              ;read data register
out portc,r16            ;and out to led port

IN R16,PIND
out spdr,r16

;ldi r16,(1<<spie)+(1<<spe)
;out spcr,r16            ;spi enableand and spi int. enable

reti
```