```
LDI R16, 1<<SPE|1<<DORD|1<<MSTR|0<<CPOL|0<<CPHA|0<< SPR1|1<< SPR0
OUT SPCR,R16

LDI R16, 1<< SPI2X
OUT SPSR,R16
```

Éste es el programa para el maestro:

```
;PROGRAMA QUE ENVÍA UN DATO POR SPI
;A OTRO AVR ESCLAVO ATMEGA8515
;ÉSTE ES EL MAESTRO

.INCLUDE "M8515DEF.INC"
.CSEG
.ORG 0

LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,HIGH(RAMEND)
OUT SPH,R16

LDI R16,0b1010_0000
OUT DDRB,R16                    ;MOSI-SALIDA
                                ;MISO-ENTRADA
                                ;SCK-SALIDA

;CONFIGURAMOS VELOCIDAD DE TRANSFERENCIA
;USAREMOS FOSC/8
LDI R16, 1<<SPE|1<<DORD|1<<MSTR|0<<CPOL|0<<CPHA|0<< SPR1|1<< SPR0
OUT SPCR,R16

LDI R16, 1<< SPI2X
OUT SPSR,R16

LDI R16,0b1010_1111             ;DATO A ENVIAR POR SPDR
OUT SPDR,R16
```

```
WAIT_TRANSMIT:
; WAIT FOR TRANSMISSION COMPLETE
SBIS SPSR,SPIF
RJMP WAIT_TRANSMIT
```

Esta sección se copió del manual AVR para hacer esperar al cursor hasta que ha enviado el último dato del registro de datos SPDR al AVR esclavo

```
FIN: RJMP FIN
```

Para el AVR esclavo a fosc/8 la configuración podría quedar de la siguiente forma:

```
LDI R16, 1<<SPE|1<<DORD|0<<MSTR|0<<CPOL|0<<CPHA|0<< SPR1|1<< SPR0
OUT SPCR,R16

LDI R16, 1<< SPI2X
OUT SPSR,R16
```

O en la forma reducida en el esclavo sólo se activa el bit `SPE` (el AVR maestro es el que gobierna la sincronía y la frecuencia, ya no es necesario dar de alta los mismos parámetros que en el maestro):

```
LDI R16, 1<<SPE
OUT SPCR,R16
```

Éste es el programa para el esclavo:

```
;PROGRAMA QUE RECIBE UN DATO POR SPI
;DE OTRO AVR MAESTRO ATMEGA8515
;ÉSTE ES EL SLAVE

.INCLUDE "M8515DEF.INC"
.CSEG
.ORG 0

LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,HIGH(RAMEND)
OUT SPH,R16
```

```
LDI R16,0b0100_0000
OUT DDRB,R16                    ;MOSI-ENTRADA
                                ;MISO-SALIDA
                                ;SCK-ENTRADA
LDI R16,$FF
OUT DDRC,R16                    ;"PUERTO C" PARA SALIDA A LED´s

;CONFIGURAMOS VELOCIDAD DE TRANSFERENCIA
;USAREMOS FOSC/8 (ESTA CONFIGURACIÓN ES OPCIONAL)
LDI R16, 1<<SPE|1<<DORD|0<<MSTR|0<<CPOL|0<<CPHA|
0<< SPR1|1<< SPR0
OUT SPCR,R16

LDI R16, 1<< SPI2X||
OUT SPSR,R16
```

O en la forma reducida

```
SPI_SlaveReceive:
; Wait for reception complete
sbis SPSR,SPIF
rjmp SPI_SlaveReceive
```

Esta sección se copió del manual AVR para hacer esperar al cursor hasta que ha llegado el último dato en el registro de datos SPDR

```
; Read received data
IN R16,SPDR             ;LEE EL DATO RECIBIDO EN SPRD
OUT PORTC,R16           ;Y LO SACA A LOS LEDS

FIN: RJMP FIN
```

```
LDI R16,(0<<USIWM1)|(1<<USIWM0)|(0<<USICS1)|(0<<USICS0)|(0<<USICLK)|(1<<USITC)
OUT USICR,R16

LDI R17,(0<<USIWM1)|(1<<USIWM0)|(0<<USICS1)|(0<<USICS0)|(1<<USICLK)|(1<<USITC)
OUT USICR,R17
```

Subrutina del transmisor en 3-wire:

```
;SPI 3-WIRE MASTER
.INCLUDE "TN2313DEF.INC"
.CSEG
.ORG 0

LDI R16,LOW(RAMEND)
OUT SPL,R16

LDI R16,0b1100_0000
OUT DDRB,R16
;PB7=SCK (SALIDA)
;PB6=DO (DATA OUT)
;PB5=DI (DATA INPUT)

;Bit     7     6     5     4     3     2     1     0
;       ___________________________________________________
;      USISIF USIOIF USIPF USIDC USICNT3 USICNT2 USICNT1 USICNT0  --USISR
;       ---------------------------------------------------

LDI R16,(1<<USIOIF)      ;PARA RESETEAR LA BANDERA DE
                         ;INTERRUPCION DE OVERFLOW (Bit 6)
OUT USISR,R16

LDI R16, 0b0111_0101     ;DATO A ENVIAR
OUT USIDR,R16            ;REGISTRO DE DATOS DEL USI


LDI R16,(0<<USIWM1)|(1<<USIWM0)|(0<<USICS1)|(0<<USICS0)|(0<<USICLK)|(1<<USITC)
                                         ;3-WIRE
                                         ------------
OUT USICR,R16                            ;Software clock strobe


LDI R17,(0<<USIWM1)|(1<<USIWM0)|(0<<USICS1)|(0<<USICS0)|(1<<USICLK)|(1<<USITC)
                                         ;3-WIRE
                                         ------------
OUT USICR,R17                            ;Software clock strobe

RCALL DOS_SEGUNDOS                       ;VAMOS A ESPERAR 2
                                         ;SEGUNDOS ANTES DE
                                         ;TRANSMITIR
```

```
OUT USICR,R16      ;7 MSB
OUT USICR,R17
OUT USICR,R16      ;6
OUT USICR,R17
OUT USICR,R16      ;5
OUT USICR,R17
OUT USICR,R16      ;4
OUT USICR,R17
OUT USICR,R16      ;3
OUT USICR,R17
OUT USICR,R16      ;2
OUT USICR,R17
OUT USICR,R16      ;1
OUT USICR,R17
OUT USICR,R16      ;0 LSB
OUT USICR,R17
```

Se va intercalando el `R16` y el `R17` que corresponde al bit `USICLK` para ir sacando los bits del dato a ser enviado (`0b0111_0101`)

```
SPITransfer_loop:
SBIS USISR,USIOIF
RJMP SPITransfer_loop
```

Esta sección se copió del manual AVR para hacer esperar al cursor hasta que ha enviado el último dato del registro de datos en `USIDR`

```
FIN: RJMP FIN
```

Programa:

```
;PROGRAMA QUE CONFIGURA USI CON 3-WIRE SLAVE
.INCLUDE "TN2313DEF.INC"
.CSEG
.ORG 0

LDI R16,LOW(RAMEND)
OUT SPL,R16

LDI R16,0b0100_0000
OUT DDRB,R16
;PB7=SCK (ENTRADA)
;PB6=DO (DATA OUTPUT)
;PB5=DI (DATA INPUT)

LDI R16,$FF
OUT DDRD,R16

LDI R16,(0<<USIWM1)|(1<<USIWM0)|(1<<USICS1)|(0<<USICS0)|(0<<USICLK) ;3-WIRE
OUT USICR,R16                                                       ;EXTERNAL NEGATIVE EDGE

;SlaveSPITransfer:

LDI R16,(1<<USIOIF)          ;Desactiva la bandera de
                             ;interrupción
OUT USISR,R16                ;Si es que antes se activó

SlaveSPITransfer_loop:       ;Espera hasta que termine de
                             ;recibir
SBIS USISR,USIOIF
RJMP SlaveSPITransfer_loop

IN R16,USIDR
OUT PORTD,R16                ;Lo que recibe el registro USI
                             ;lo envía al "PORT D" para
                             ;comprobar que el dato enviado
                             ;esté en el
FIN: RJMP FIN                ;receptor
```

Copiado del manual

Para el transmisor:

```
;SACA Bit A Bit EL VALOR DEL REGISTRO A TRANSMITIR A OTRO AVR
;(ATTINY2313)
;ENVÍA Bit A Bit POR MEDIO DE DOS PINES DEL "PUERTO C"
;SE USA PARA ENVIAR UN ATMEGA8515
```

Encabezado para ATmega8515

Stack Pointer para ATmega8515

```
; 7 6 5 4 3 2  1    0   ...Bit
;----------------------
;|X|X|X|X|X|DATA|CLK| ...FUNCIÓN
;----------------------

LDI R16,0b0000_0011     ;Estos dos Bits representan el
                        ;CLK=PB0
OUT DDRC,R16            ;Y el PB1 EL Bit dato a sacar
                        ;Los demás Bits no se usan

RCALL 2_SEGUNDOS        ;Vamos a esperar 2 segundos antes de
                        ;la transmisión

LDI R25,0b1100_0010     ;DATO A ENVIAR

SACANDO_BITS:

LDI R21,8               ;8 repeticiones
LDI R22,0               ;Para comparar

ENVIANDO_BIT_A_BIT:

LDI R17,0b00000001
LDI R18,0b00000001
CP R22,R21
BREQ FIN_PARCHE

AND R17,R25             ;Compara el Bit del dato para enviar
                        ;CERO o UNO
CP R17,R18

BREQ BIT_UNO
RJMP BIT_CERO
```

```
FIN_PARCHE:
RJMP FIN

;ENVIANDO EL BIT
;*****************************
;SI EL BIT DEL DATO A ENVIAR ES UNO

BIT_UNO:
DEC R21

LDI R19,0b00000011
OUT PORTC,R19      ;envía DATO al otro AVR

RCALL DELAY_2

LDI R19,0b00000000
OUT PORTC,R19

RCALL DELAY_2

LSR R25

RJMP ENVIANDO_BIT_A_BIT

;*****************************
;SI EL Bit DEL DATO A ENVIAR ES CERO

BIT_CERO:
DEC R21

LDI R19,0b00000001
OUT PORTC,R19

RCALL DELAY_2

LDI R19,0b00000000
OUT PORTC,R19

RCALL DELAY_2

LSR R25

RJMP ENVIANDO_BIT_A_BIT

FIN: RJMP FIN
```

Para el receptor:

```
;PROGRAMA QUE RECIBE Bit A Bit
;EN UN PIN DE PUERTO
```

Encabezado para ATtiny2313

Stack Pointer para ATtiny2313

```
LDI R16,0b0000_0000
OUT DDRD,R16            ;Como entradas PD0=CLOCK, PD1=DATA

LDI R16,$FF             ;Para que prendan los LED´s
OUT DDRB,R16

LDI R16,$00
OUT PORTB,R16

LDI R16,0
MOV R0,R16

LDI R16,1
MOV R1,R16

LDI R16,2
MOV R2,R16

LDI R16,3
MOV R3,R16

LDI R16,4
MOV R4,R16

LDI R16,5
MOV R5,R16

LDI R16,6
MOV R6,R16

LDI R16,7
MOV R7,R16

LDI R16,0
MOV R8,R16              ;PARA EL #DE VUELTAS DE LOS Bits
```

```
RECIBE_BIT_A_BIT:

;LEE DATO PROVENIENTE DEL TRANSMISOR

LDI R16,0b0000_0001     ;Máscara de detección de CLK
LDI R18,0b0000_0011     ;Máscara para UNO
LDI R19,0b0000_0001     ;Máscara para CERO

LEYENDO:

IN R17,PIND             ;R17 contiene el Bit del dato que
                        ;envía el transmisor

MOV R9,R17

AND R17,R16             ;Máscara para saber si existe dato en
                        ;puerto
CP R17,R16
BRNE LEYENDO

MOV R17,R9              ;Regresa valor cargado anteriormente

AND R17,R18             ;Máscara para extraer Bit 0 o 1

CP R17,R18
BREQ DATO_UNO

RJMP DATO_CERO

;****************
DATO_CERO:

LDI R17,0b0000_0000
RJMP GO_ON
;****************
DATO_UNO:

LDI R17,0b0000_0001

GO_ON:

CP R0,R8                ;COMPARA con Bit_0
BREQ CERO_PARCHE
```

```
CP R1,R8                ;COMPARA con Bit_1
BREQ UNO_PARCHE
.
.
.
CP R6,R8                ;COMPARA con Bit_6
BREQ SEIS_PARCHE

CP R7,R8                ;COMPARA con Bit_7
BREQ SIETE_PARCHE


CERO_PARCHE:
RJMP CERO
.
.
.
SEIS_PARCHE:
RJMP SEIS

SIETE_PARCHE:
RJMP SIETE

;**********************************
CERO:

MOV R20,R17             ;R20 adquiere el valor inicial
INC R8

OUT PORTB,R20           ;DATO ACTUAL

RCALL DELAY
RJMP RECIBE_BIT_A_BIT

UNO:

LSL R17
OR R17,R20
MOV R20,R17             ;R20 adquiere el VALOR ACTUAL
INC R8

OUT PORTB,R20           ;DATO ACTUAL

RCALL DELAY
RJMP RECIBE_BIT_A_BIT
```

```
DOS:

LSL R17
LSL R17
OR R17,R20
MOV R20,R17              ;R20 adquiere el VALOR ACTUAL
INC R8

OUT PORTB,R20            ;DATO ACTUAL

RCALL DELAY
RJMP RECIBE_BIT_A_BIT
.
.
.
SIETE:

LSL R17
LSL R17
LSL R17
LSL R17
LSL R17
LSL R17
LSL R17

OR R17,R20
MOV R20,R17

OUT PORTB,R20            ;DATO ACTUAL
LDI R16,0
MOV R8,R16               ;RESETA # DE Bits

RCALL DELAY_2
FIN:RJMP FIN

;RJMP RECIBE_BIT_A_BIT
```

Este programa sólo recibe un dato de 8-bits, ya que el transmisor está programado para ello, por eso la subrutina termina con `FIN: RJMP FIN`, pero si el transmisor va a enviar más datos, la subrutina terminará con `RJMP RECIBE_BIT_A_BIT`