Design with Microprocessors

Lecture 6

Interfaces for serial communication

Year 3 CS
Academic year 2023/2024
1st Semester

Lecturer: Radu Dănescu

Serial communication modules on AVR MCUs

Serial Peripheral Interface (SPI)

- Synchronous serial communication
- Full duplex operation
- Master or Slave configuration
- Variable frequency (bit rate)
- Can be used for connecting two MCUs, or for connecting the MCU with different peripherals

Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART)

- Synchronous or asynchronous serial communication
- Variable frequency (baud rate)
- Data frames of 5-9 bits, with or without parity
- Can use interrupts for transmission control
- Error detection
- Can be used for connecting the MCUs with the PC (Serial port), or with other MCUs or peripherals

Serial communication modules on AVR MCUs

Two Wire Serial Interface (TWI)

- Complex protocol based on only two lines (clock and data)
- Atmel implementation of the I2C (Inter Integrated Circuit) protocol
- The TWI controller supports master and slave operation modes
- 7 bit addressing
- Multiple masters arbitration support
- Programmable slave address

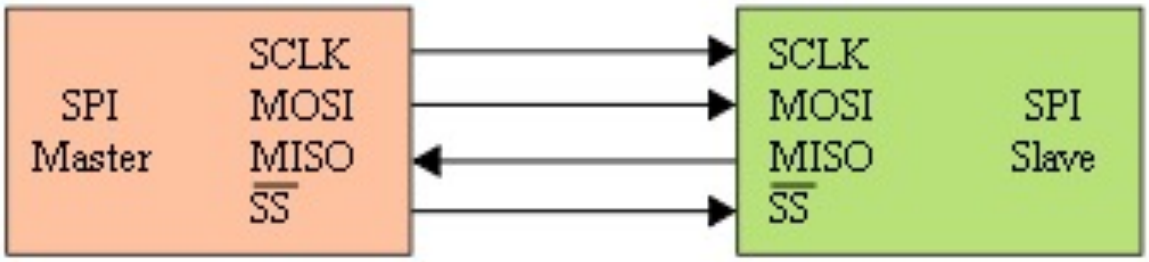
Serial Peripheral Interface (SPI)

Signals

- SCLK – Serial clock, generated by Master
- MOSI – Master Output, Slave Input, data sent by Master
- MISO – Master Input, Slave Output, data received by Master
- SS – Slave select – Slave device activation signal, driven by Master, active LOW

Operation

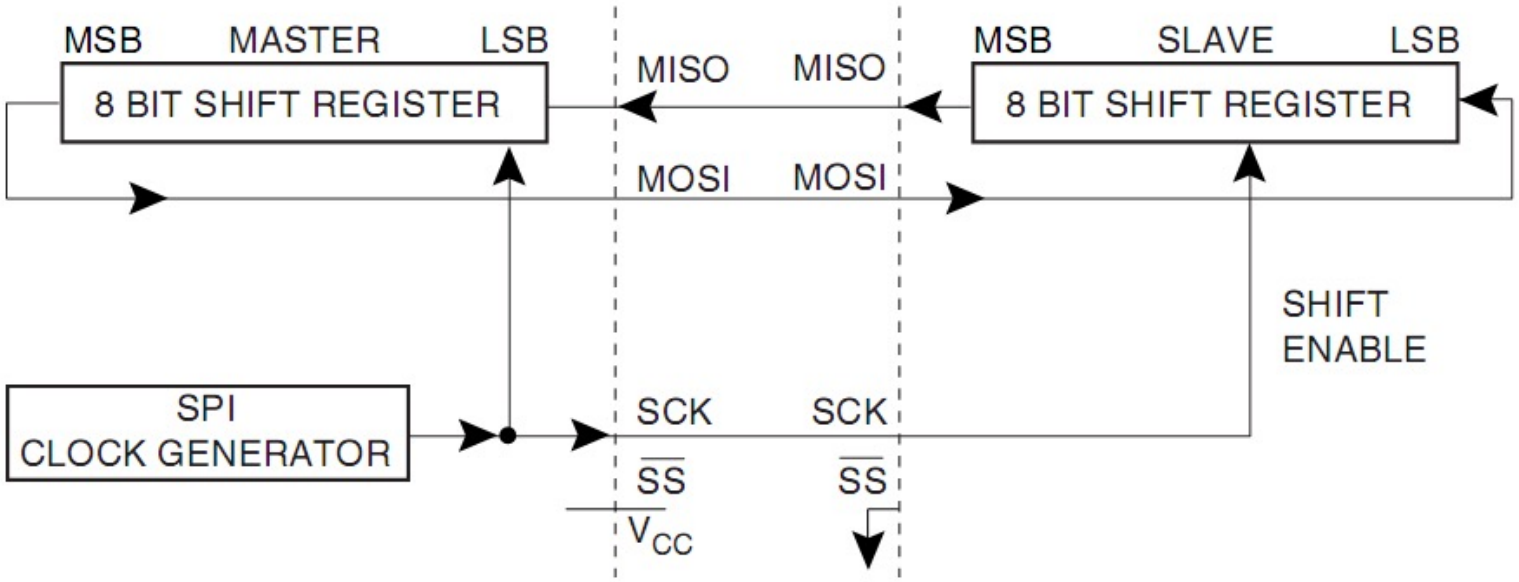
- Master initializes communication by activating SS
- Master generates the clock signal SCLK
- Every clock period, a bit is transferred from Master to Slave, and one bit from Slave to Master
- After each data frame (8, 16 bits,…) SS is de-activated, for synchronization



Serial Peripheral Interface (SPI)

Operating principle

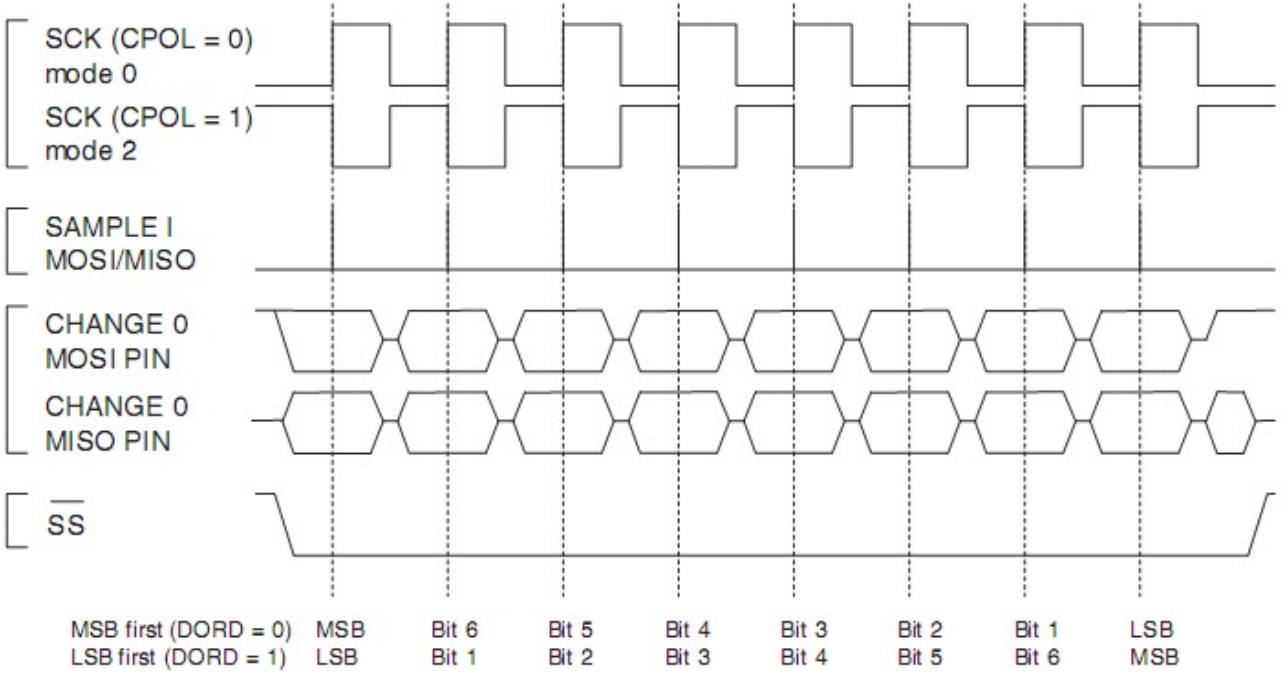
- Both partners have an internal shift register, their ends connected to MISO and MOSI
- Both registers operate on the same clock, SCLK
- Together the shift registers form a rotation register
- After a number of clock periods equal to the size of a shift register, Master and Slave exchange data.



Serial Peripheral Interface (SPI)

Data synchronization with the clock signal

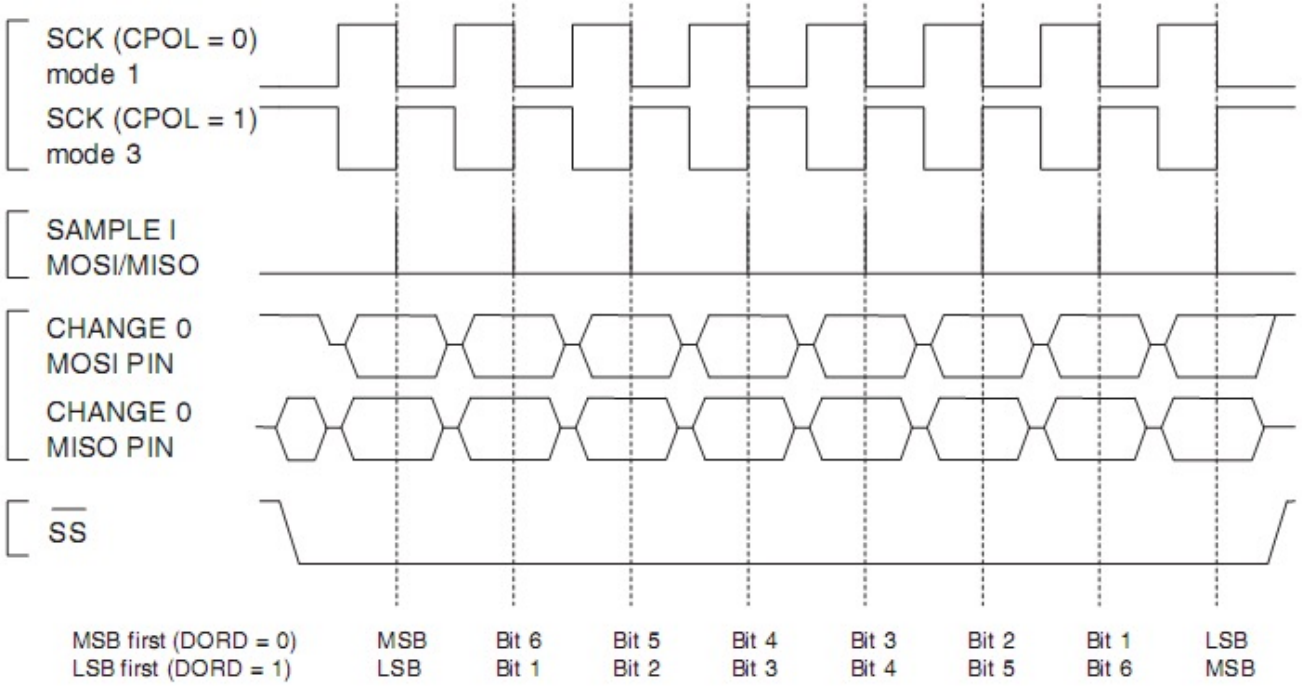
- Shifting and reading the data are done on opposing clock edges
- CPOL – clock polarity – whether the first edge is rising or falling
- CPHA – clock phase
- For CPHA = 0
  - Latch data on the first clock edge
  - Shift (set up) the data on the second clock edge



Serial Peripheral Interface (SPI)

Data synchronization with the clock signal

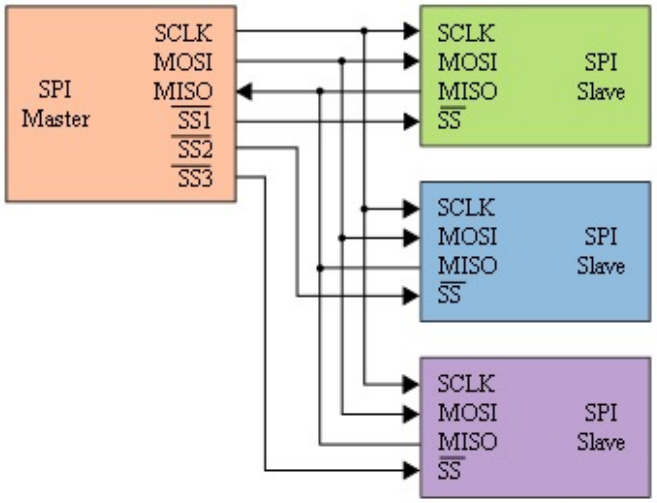
- For CPHA = 1
  - Shift the data on the first clock edge
  - Latch the data on the second clock edge

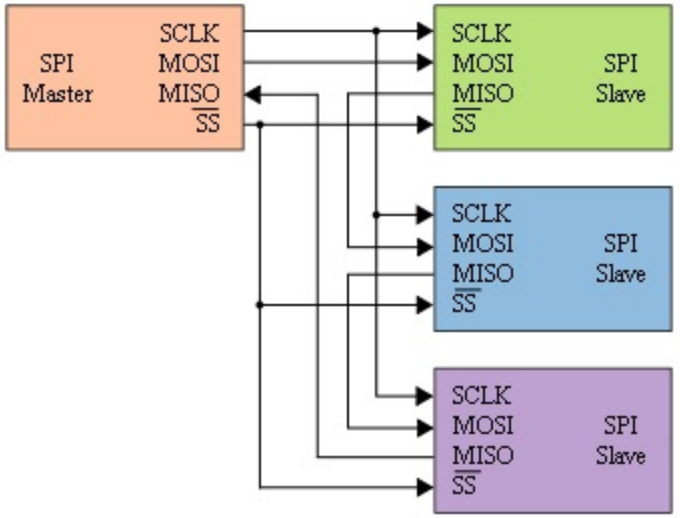


Serial Peripheral Interface (SPI)

Using the SS signal

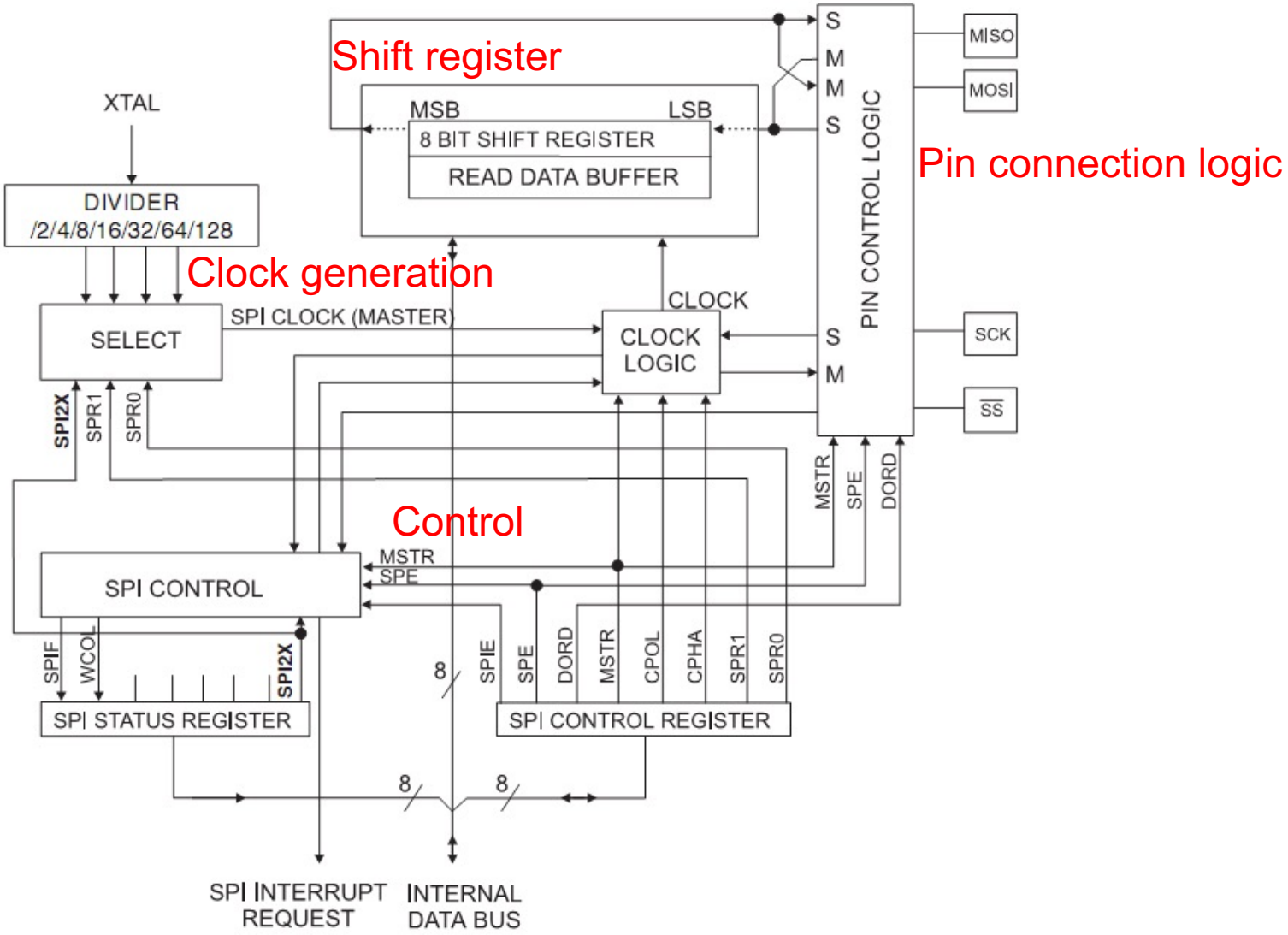
- For a Slave device, **SS** is an input signal
  - SS = 0 means the slave device is activated. A transition from 0 to 1 means re-setting the transfer cycle (marks the end of a data frame)
  - SS = 1 – the slave device is inactive
- For a Master device, **SS** can be either:
  - Output – used for activating the Slave for communication
  - Input – if more Masters are allowed, a '0' on the SS line means the device enters Slave mode.
- Multiple devices configuration: independent **SS** signals, or "daisy chain"





SPI on AVR MCUs

SPI sub-system architecture



SPI on AVR MCUs

SPI Configuration

Bit	7	6	5	4	3	2	1	0	
0x0D (0x2D)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- The **SPCR** register:

  SPIE – SPI Interrupt Enable, activates interrupt generation at the end of transmission
  SPE – SPI Enable. Must be set to 1 to use the SPI system.
  DORD – Data Order. 1=LSB first, 0 = MSB first
  MSTR – Master, if 1, Slave, if 0
  CPOL, CPHA – select the clock polarity and phase

	Leading Edge	Trailing Edge
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)

SPR1, SPR0 – along with SPI2x from SPSR configure the speed of the SPI clock

SPI on AVR MCUs

SPI configuration - continued

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- The **SPSR** register:

  SPI2X – Along with SPR1 and SPR0 from SPCR configure the speed of the SPI clock

  WCOL – Write collision – set if **SPDR** is written before transmission is complete
  SPIF – SPI Interrupt flag – set when the transmission is complete. If SPIE is set, an interrupt request is generated.

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

SPI on AVR MCUs

Using the SPI Master mode

1. Configuring the direction of the I/O pins:
   The SPI pins are found on Port B (on ATMega 2560)

Port Pin	Alternate Functions
PB7	OC0A/OC1C/PCINT7 (Output Compare and PWM Output A for Timer/Counter0, Output Compare and PWM Output C for Timer/Counter1 or Pin Change Interrupt 7)
PB6	OC1B/PCINT6 (Output Compare and PWM Output B for Timer/Counter1 or Pin Change Interrupt 6)
PB5	OC1A/PCINT5 (Output Compare and PWM Output A for Timer/Counter1 or Pin Change Interrupt 5)
PB4	OC2A/PCINT4 (Output Compare and PWM Output A for Timer/Counter2 or Pin Change Interrupt 4)
PB3	MISO/PCINT3 (SPI Bus Master Input/Slave Output or Pin Change Interrupt 3)
PB2	MOSI/PCINT2 (SPI Bus Master Output/Slave Input or Pin Change Interrupt 2)
PB1	SCK/PCINT1 (SPI Bus Serial Clock or Pin Change Interrupt 1)
PB0	$\overline{SS}$/PCINT0 (SPI Slave Select input or Pin Change Interrupt 0)

2. Configuring the SPCR and SPSR registers with the chosen work mode and speed
3. Activating SS (PB(0) <- ‘0’, explicit!)
4. Write data to SPDR – starts the transmission
5. Wait until SPIF is set in SPSR – transmission complete
6. Read data from SPDR – data sent by Slave
7. De-activate SS (PB(0) <- ‘1’)

SPI on AVR MCUs

Using the SPI Master mode – Source code

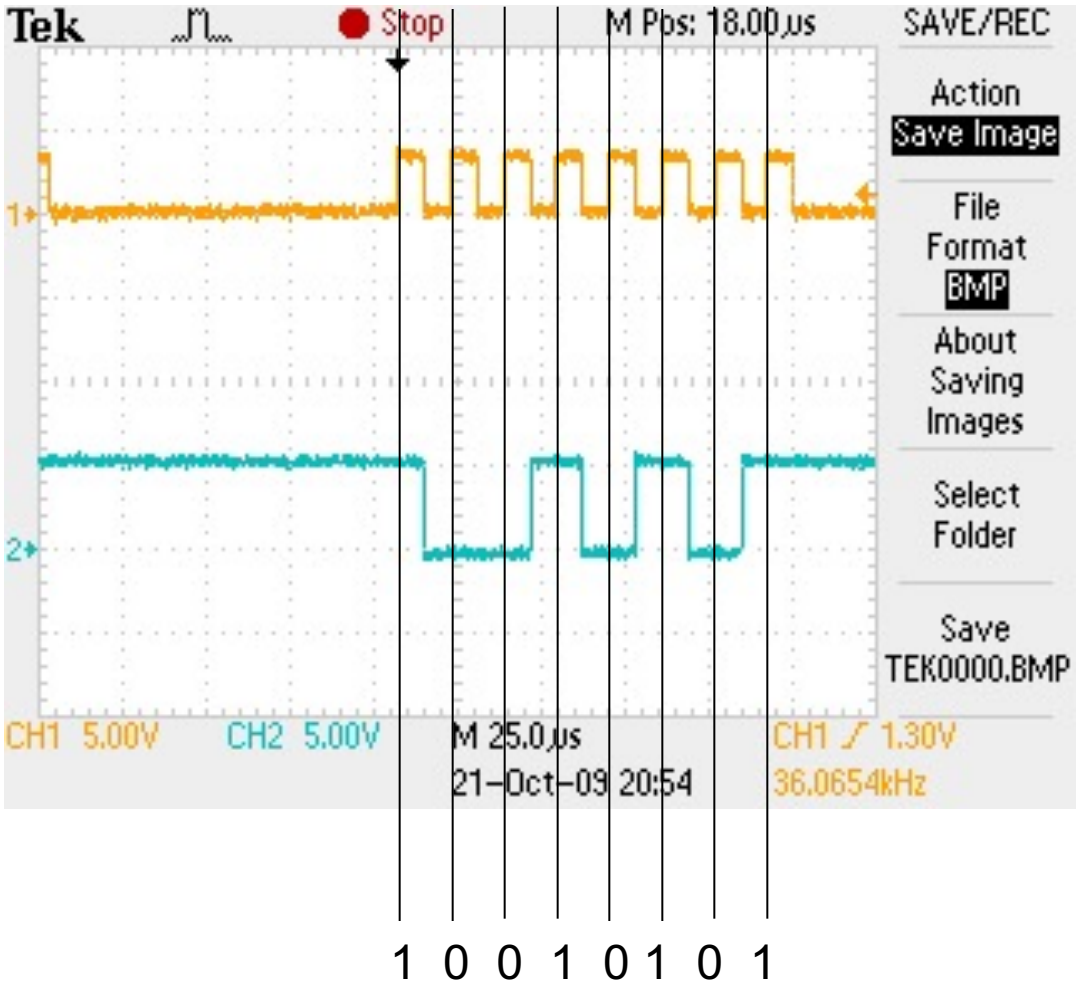
```
.org 0x0000
jmp reset
reset:
ldi r16,0b00000111      ;MISO input, MOSI, SCK and SS output
out DDRB,r16
ldi r16, 0b00000001     ;Initially, SS<--1, SPI Slave inactive
out PORTB, r16
cbi SPSR, 0             ; set bit 0 of SPSR to zero – for clock speed selection
ldi r16,0b01010011      ;No interrupts, SPI Enabled, MSB first, Master, CPOL=0 – first edge is rising, CPHA
                        = 0 – latch on the first edge, Slowest clock speed
out SPCR,r16
loop:
   cbi PORTB, 0         ; SS <- 0
   ldi r16, 0b10010101  ; data to send
   out SPDR, r16
wait:
   sbis SPSR, 7         ; check bit 7 of SPSR for transmission complete
   rjmp wait
   in r16,SPDR
   sbi PORTB, 0         ; SS <- 1
   ldi r18, 0
wait2:                  ; short pause between transmissions, then go again
   dec r18
   brne wait2
rjmp  loop
```

SPI on AVR MCUs

Using the SPI Master mode – Result

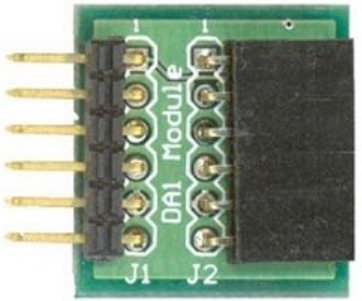
SCK

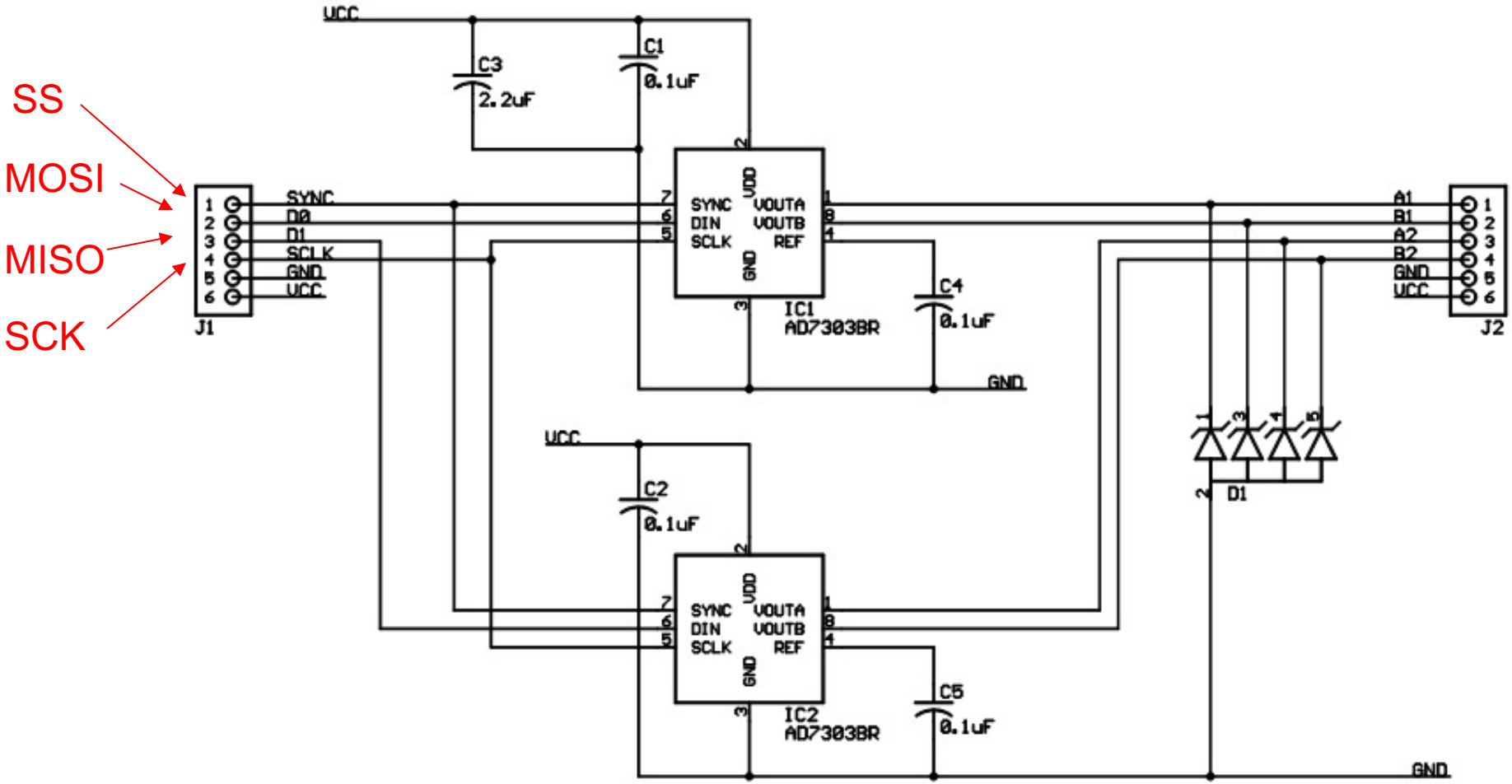
MOSI



SPI on AVR MCUs

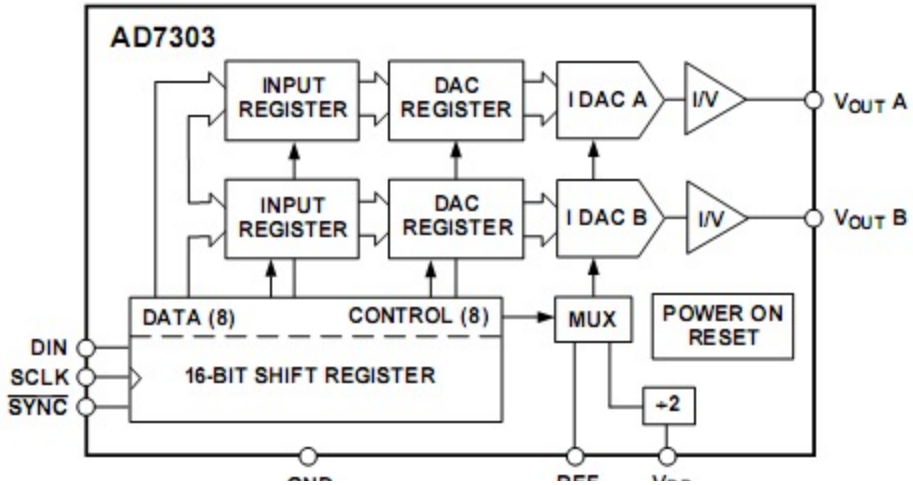
Connecting peripherals using SPI
Digilent PMOD DA1 – Digital to Analog Converter

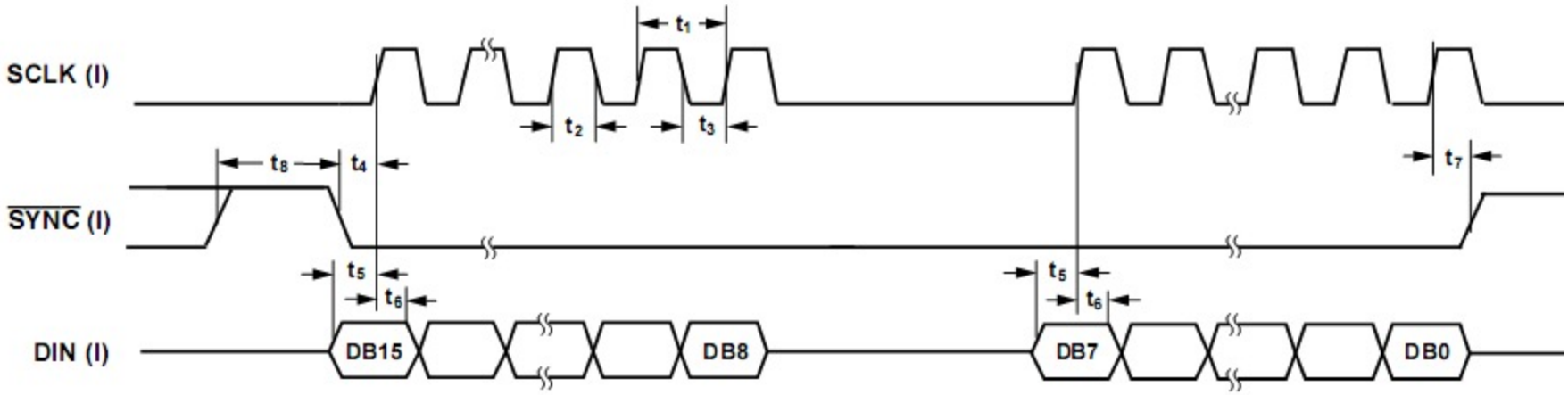




SPI on AVR MCUs

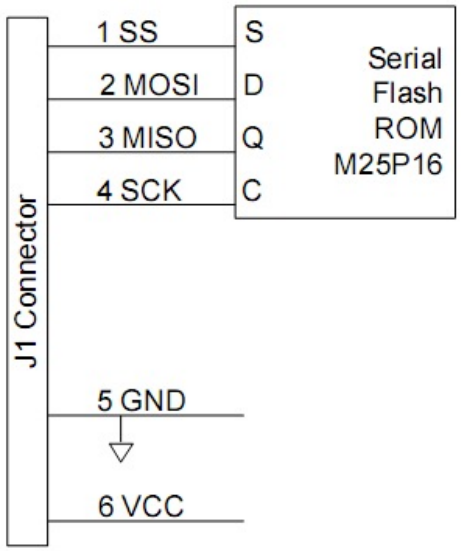
Connecting peripherals using SPI
Digilent PMOD DA1 – Digital to Analog Converter
Transmission of 16 bits (2x8 bits) – first 8 data, next 8 command



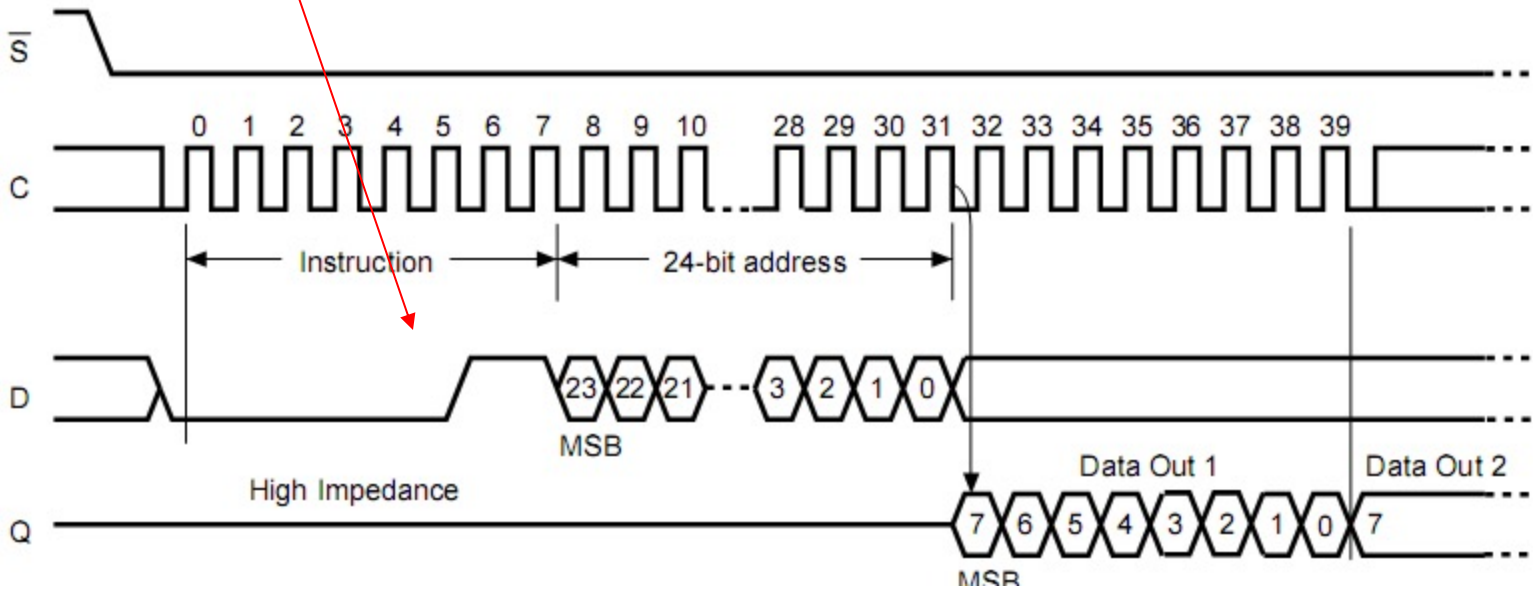


SPI on AVR MCUs

Connecting peripherals using SPI
Digilent PMOD SF – Serial Flash



00000011 = 'READ'

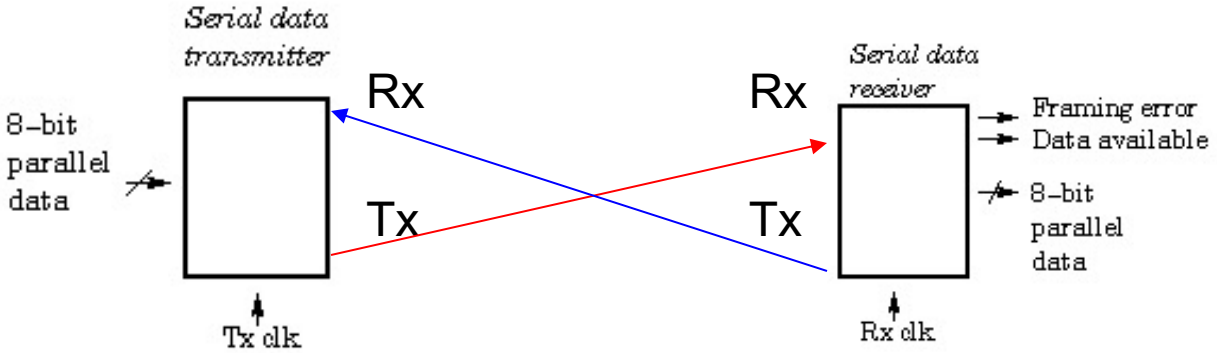


USART

USART – UART with optional synchronization using a clock signal

UART – Asynchronous serial communication interface

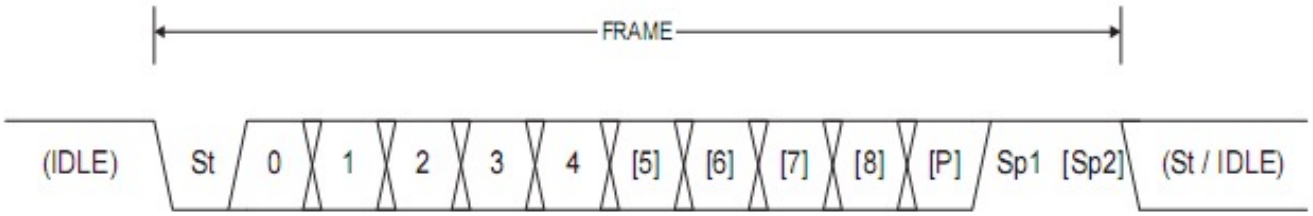
- **Asynchronous** – the interval between data frames is undefined. The receiver detects the start and end of a frame.
- The time interval between bits (bit frequency, **baud rate**) is fixed and must be known by the transmitter and by the receiver.
- Transmission and reception can be performed simultaneously (full duplex). Each side can initiate a transmission.
- The basic UART signals
  - Rx – input, reception
  - Tx – output, transmission
- USART has the additional xck (external clock) signal, input or output, which will synch the transmission and reception.
- We will only discuss the UART operation.



USART

Data transmission: A frame (packet) is made of

- **St**: 1 start bit, of value ‘0’
- **D**: Data bits (5…9, size known by both participants)
- **P**: 1 parity bit. Parity can be:
  - Absent: no P bit
  - *Even*
  - *Odd*
- **Sp**: 1 or 2 stop bits, of value ‘1’ – the number of stop bits must be known by both participants



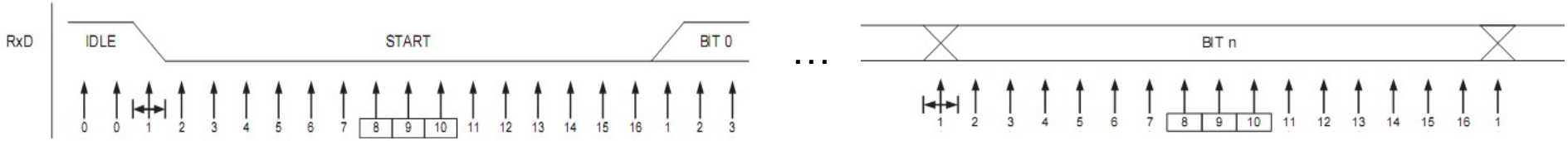
$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

USART

Data reception: - The receiver must know the transmission parameters (Baud, Number of data bits, Number of stop bits, Parity).

1. A transition from 1 to 0 on Rx is detected – **Reception initiated!**
2. Check the middle bit period for the start bit. If Rx is still '0', continue with the reception sequence, otherwise go back to idle more (*assume noise*).
3. Check the middle of interval for the next bits (data, parity, stop), and reconstruct the data frame.
4. If a zero is found in the position of the stop bits, generate a **framing error.**
5. If the computed parity at the receiver does not match the parity bit P, generate a **parity error.**

For robustness, the receiver samples the input signal with a frequency 8-16 times higher than the *baud rate*.



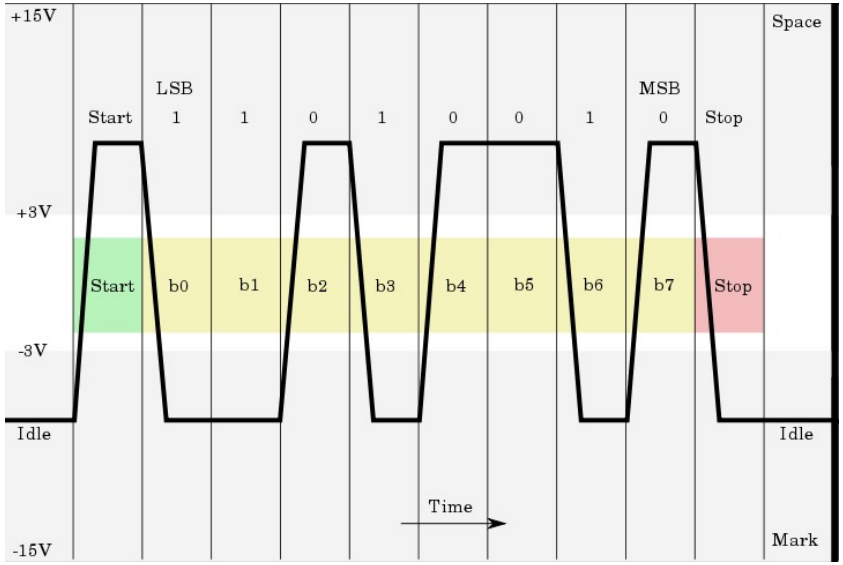
Different baud rates at transmitter and at receiver may lead to bit sampling errors! These errors may be detected as framing errors, parity errors, but they may also be undetected, leading to false reception!

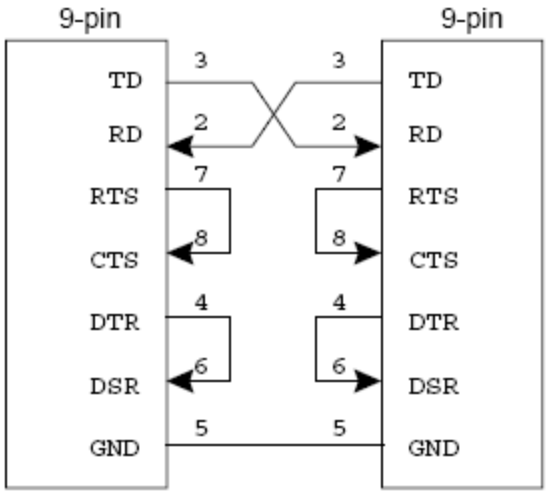
USART

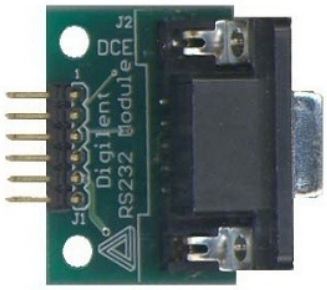
UART and RS232:
Adapting voltage levels
RS232 logic ‘1’ -5… -15 V
RS232 logic ‘0’ +5…+15 V

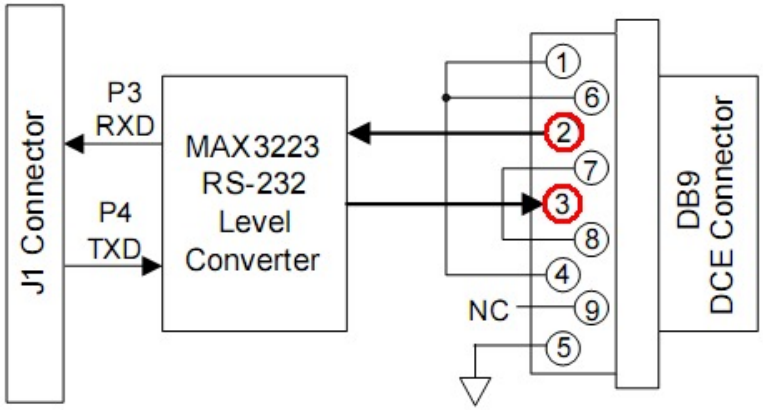
Logic level conversion between AVR UART and RS232 is needed

Pin correspondence









USART

UART and USB:

Use of a FTDI (Future Technology Devices International Ltd) adapter

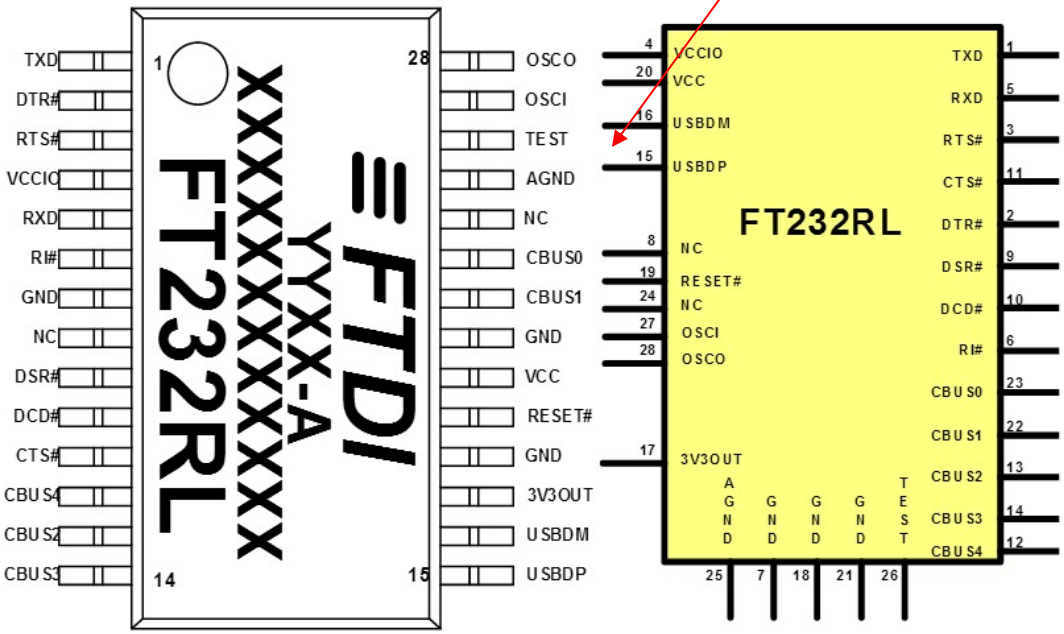
Arduino Mega uses the FT232RL chip

- Seen as a virtual COM port on the PC
- Bi-directional conversion between USB and UART

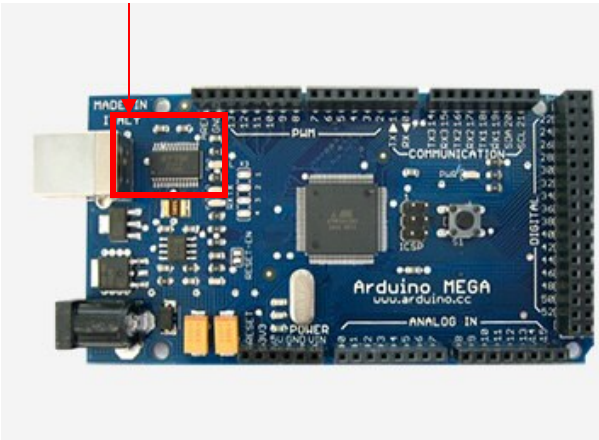
http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

UART Tx and Rx

USB D+ and D-



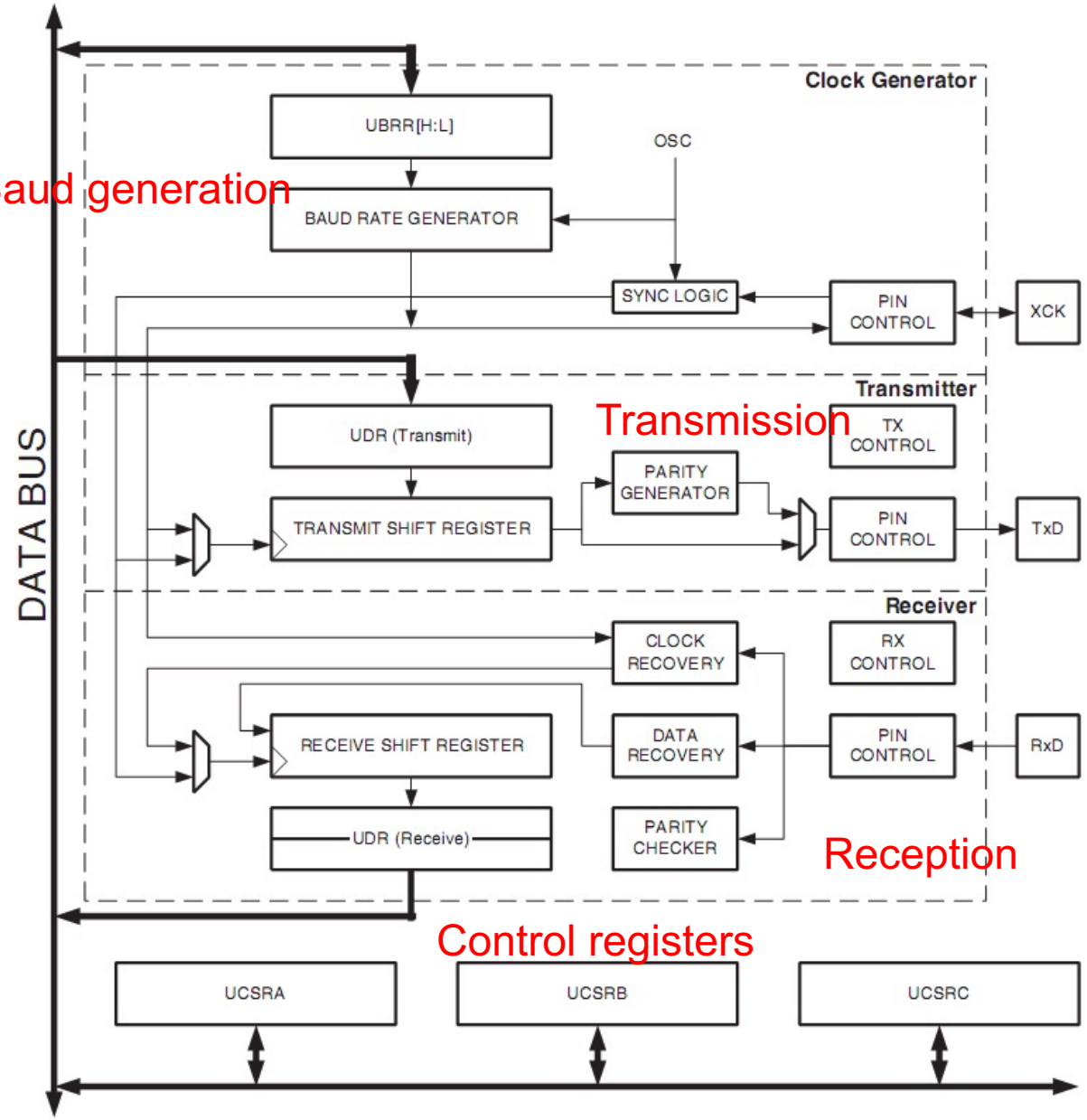
FTDI 232 RL



USART on AVR MCUs

Four USART units on ATMega 2560:
USART0 … 3

General architecture:



USART on AVR MCUs

System configuration:

- Status and control register **UCSRnA**

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **RXCn** – is ‘1’ when reception is completed. Can trigger an interrupt request.
- **TXCn** – is ‘1’ when transmission is completed. Can trigger an interrupt request.
- UDREn – Data Register Empty, signals that the data register can be written.
- FEn – signals Frame Error
- DORn – Data overrun – when a reception start is detected before the already received data are read from the UDRn register
- UPEn – signals Parity Error
- U2Xn –When ‘1’ = Doubling the USART data rate (baud rate)
- MPCMn – Activates multiprocessor communication mode

USART on AVR MCUs

System configuration:

- Status and control register **UCSRnB**

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	**TXCIEn**	**UDRIEn**	**RXENn**	**TXENn**	**UCSZn2**	**RXB8n**	**TXB8n**	**UCSRnB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- RXCIEn – If set to ‘1’, interrupt request is generated at the end of reception
- TXCIEn – If set to ‘1’, interrupt request is generated at the end of transmission
- UDRIEn - If set to ‘1’, interrupt request is generated when the data register is empty
- **RXEn** – activate reception
- **TXEn** – activate transmission
- UCSZn2 – combined with UCSZn1 and UCSZn0 from **USCRnC** sets the packet size
- RXB8n – 9-th received bit, when using 9 bit data frames.
- TXB8n – 9-th bit to transmit, when using 9 bit data frames.

USART on AVR MCUs

System configuration:

- Status and control register **UCSRnC**

Bit	7	6	5	4	3	2	1	0	
	–	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **UMSELn** – Selects asynchronous ‘0’ or synchronous ‘1’ mode
- UPMn1 si UPMn0 – Selection of parity mode
- **USBSn** – stop bits configuration: ‘0’ – 1 bit, ‘1’ – 2 bits
- **UCSZn1:UCSZn0** – combined with UCSZn2 of UCSRnB, set the packet size
- UCPOLn – clock polarity for the synchronous mode

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

USART on AVR MCUs

System configuration:

- Frequency control registers: **UBRRnH** si **UBRRnL**
- Together they form **UBRRn**, 12 bits

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR + 1n)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$

Reading received data / writing data to be transmitted

- Both actions are performed using the register **UDRn**

USART on AVR MCUs

Example: communication between AVR and the PC – simple ECHO
Requirements: UART-USB adapter, USB cable, we use UART 1

$$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$$

f_{osc} = 16000000
UBRRn = 103

1. **Configuration**
   Baud: 9600
   Data size: 8 bits
   Stop bits: 2
   Parity: none

2. **Wait for a character to be received**
   - Check **RXCn** of **UCSRnA**, wait until it becomes 1
3. **Read received character, from UDRn**
4. **Write the character back, to UDRn**
5. **Wait for the character to be transmitted**
   - Check **TXCn** of **UCSRnA**, wait until it becomes 1
6. **Jump to 2**

USART on AVR MCUs

Example: communication between AVR and the PC – simple ECHO
Source code

```
ldi r16, 0b00011000          ; activate Rx and Tx
sts UCSR1B,r16
ldi r16, 0b00001110          ; frame size 8 bits, no parity, 2 stop bits
sts UCSR1C,r16
ldi r16, 103                 ; Computed Baud rate, fits in 8 bits
ldi r17, 0                   ; The higher order bits of UBRR are zero
sts UBRR1H, r17
sts UBRR1L, r16
mainloop:

  recloop:
      lds r20, UCSR1A
      sbrs r20, 7            ; bit 7 of UCSR1A – reception complete
  rjmp recloop

lds r16, UDR1                ; read the received data
sts UDR1,r16                 ; write back the received data to UART

  txloop:
     lds r20, UCSR1A         ; wait for the end of transmission
     sbrs r20, 5
  rjmp txloop
rjmp mainloop
```

ASCII codes

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com