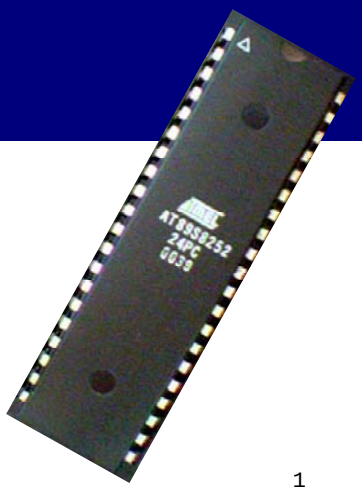
Digital Integrated Circuits & Microcontrollers

Chapter 7. Serial communication bus



Bus types

- USART
- RS232
- RS485
- I2C (I^2C) = Inter-Integrated Circuit
- SPI = Serial Peripheral Interface
- One-wire (Dallas)
- CAN = Controller–Area Network

USART

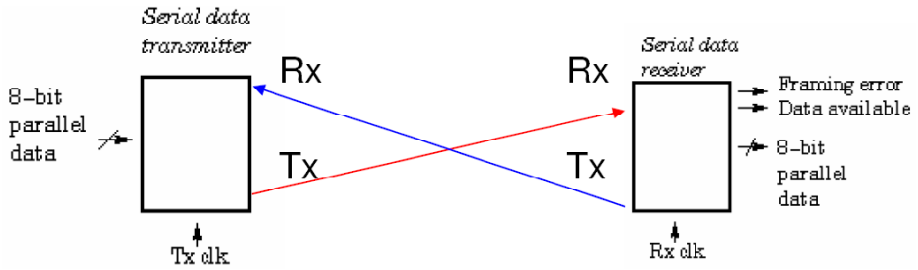
- **Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART)**
- Synchronous or asynchronous serial communication
- Variable frequency (baud rate)
- Supports packet data of 5-9 bits, with or without parity
- Supports interrupt transmission control
- Detection of transmission errors

UART – Universal Asynchronous serial Receiver/ Transmitter

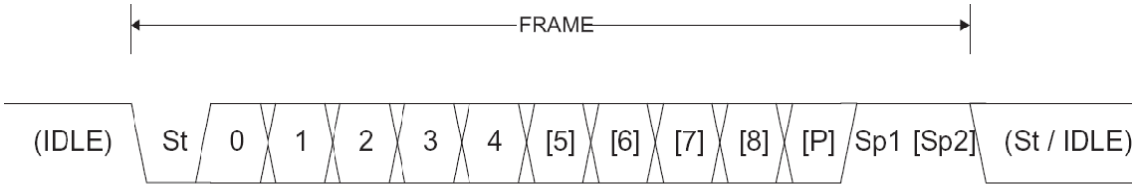
- **Asynchronous** - The interval between data packets can be undefined. The receiver detects the start and the end of packet.
- Frequency transmission bit (**baud rate**) is fixed and must be known on both sides of the transmission
- Transmission and reception can be performed simultaneously (**full duplex**). Each part of the conversation can initiate a transmission.

UART

- UART interface has two signals
  - Rx – Receive
  - Tx – Transmit



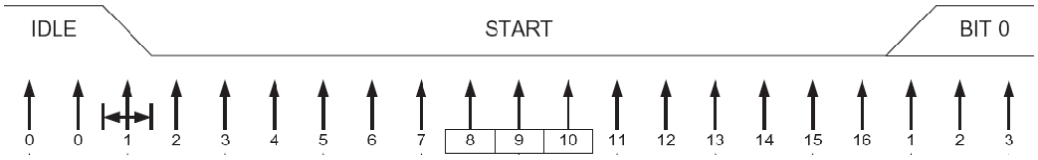
- **USART** has an additional signal, **XCK** (external clock) that can be input or output, and will synchronize the transmission and reception



- **Data transmission:** One packet (frame) is formed from:
  - **St** – 1 Start bit of '0' logic
  - **D** – Data bits (5…9, established by both transmission members)
  - **P** - 1 Parity bit. Parity can be:
    - Absent: P bit no exist
    - *Even parity* $P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$
    - *Odd parity* $P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$
  - **Sp** – 1 or 2 Stop bits of '1' logic

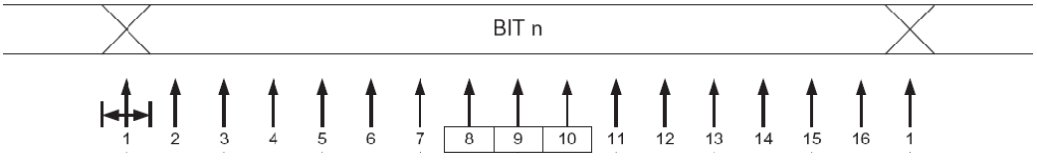
UART

- **Data Reception** - The receiver system must know the transmission parameters (Baud, no. of bits / frame, no. of stop bits, Parity)
  1. A transition from '1 'to '0' is detected on Rx (reception)
  2. Check the middle range for the start bit. If '0 ', receiving sequence is initiated, otherwise the transition is considered noise.



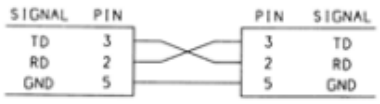
UART

3. Check the middle range for the next bits (data, parity, stop), and recover data package
4. If at the position where the stop bits should be, it is detected ‘0’, it generates **framing error**.
5. If the calculated parity bit P doesn’t corresponds to the destination, it generates a **parity error**.

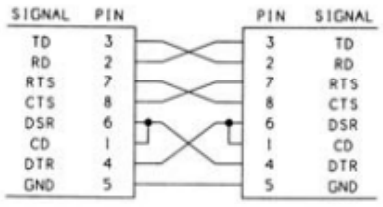


RS232

- More than UART: ***handshake***
  - Flow control
  - Equipment status
- RTS - Request to send,
- CTS - Clear to send
- DSR - Data set ready,
- DTR - Data terminal ready
- DCD - Data carrier detect
- RI - Ring indicator

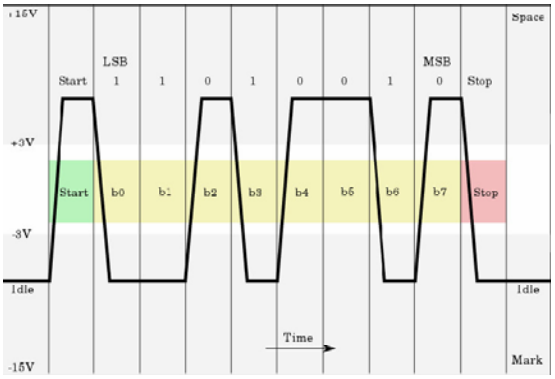


No Handshake



Full Handshake

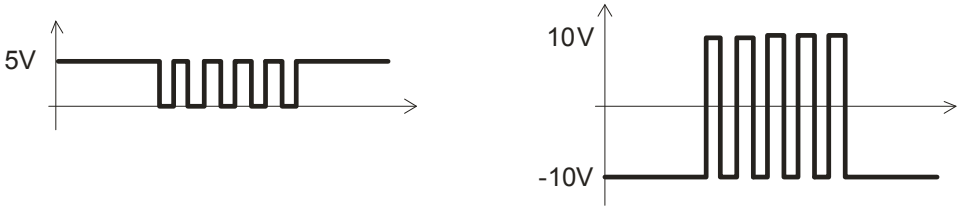
RS232 and UART



- RS232
  - Logic ‘1’: -5… -15 V
  - Logic ‘0’: +5…+15 V
- one character structure:
  - 1 start bit, 8 data bits (8D), 1 stop bit
- the 10 bits on the picture:
  - START, 8D, STOP = 0110100101
  - LSB is transmitted first (so MSB is adjacent to the stop bit) so the 8 bit number must be read from right to left: 01001011

RS232 and UART

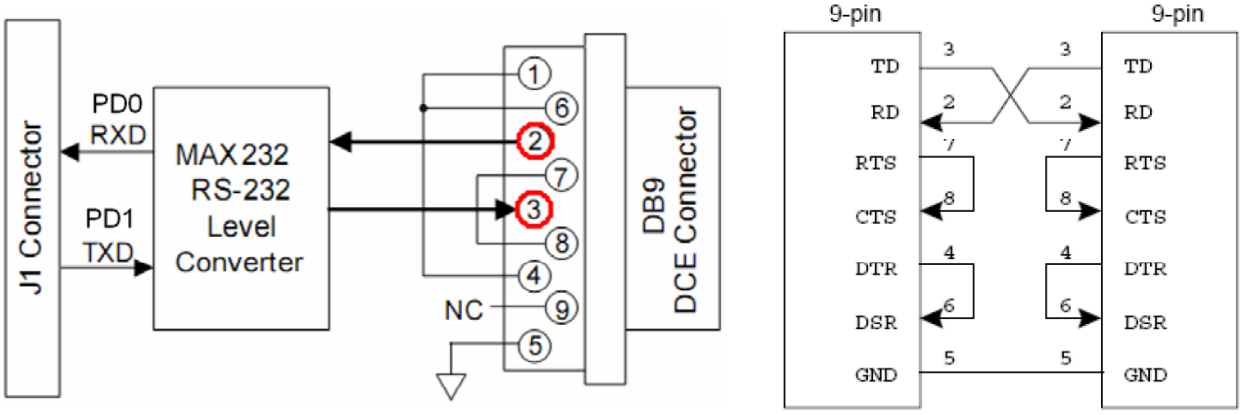
- UART to RS232 levels conversion is needed
  - uC (UART) uses TTL: “0” = 0V, “1” = 5V
  - PC serial line (RS232): “0” = +12V, “1”=“-12V”



- lines are kept at “1” while idle
- compare the above waveforms: TTL, RS232 (the more “natural” one is TTL)

RS232 and UART

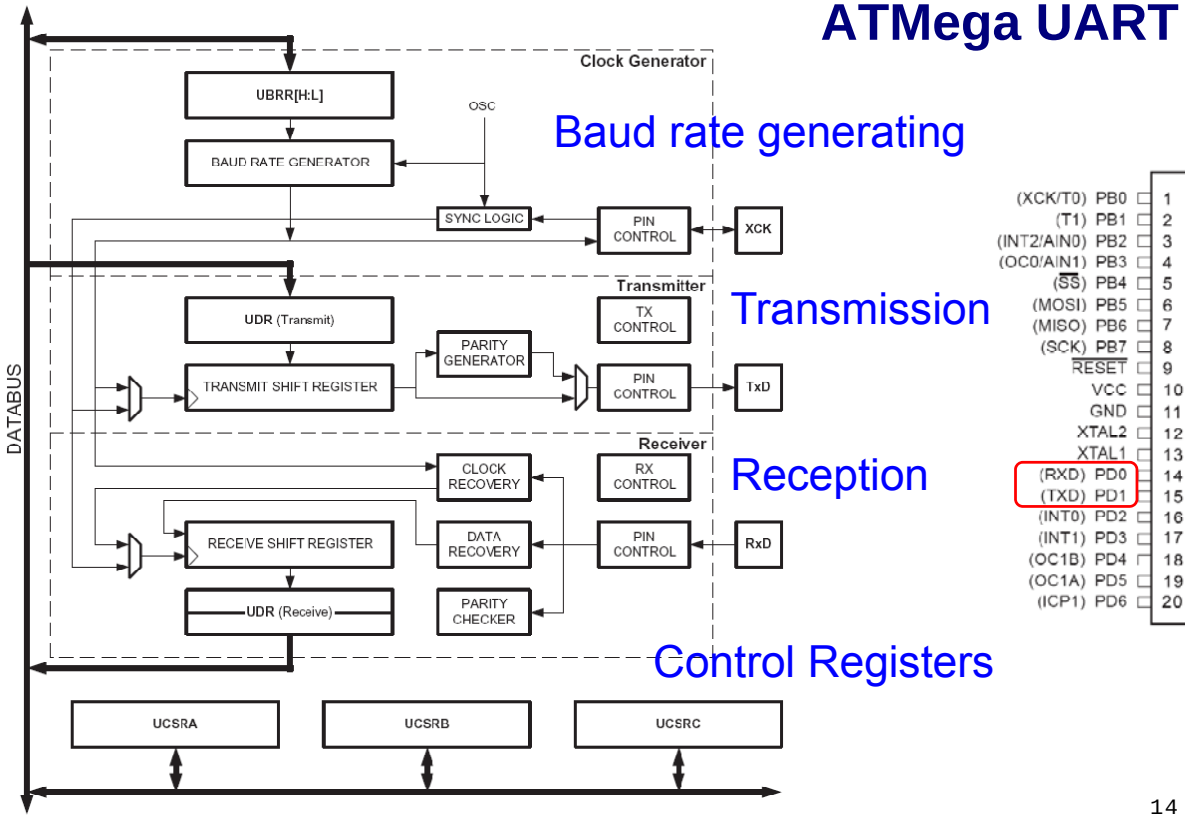
- Electrical level conversion
  - MAX232, MAX202 etc.
- Pins correspondence (9 pins connector DB9):



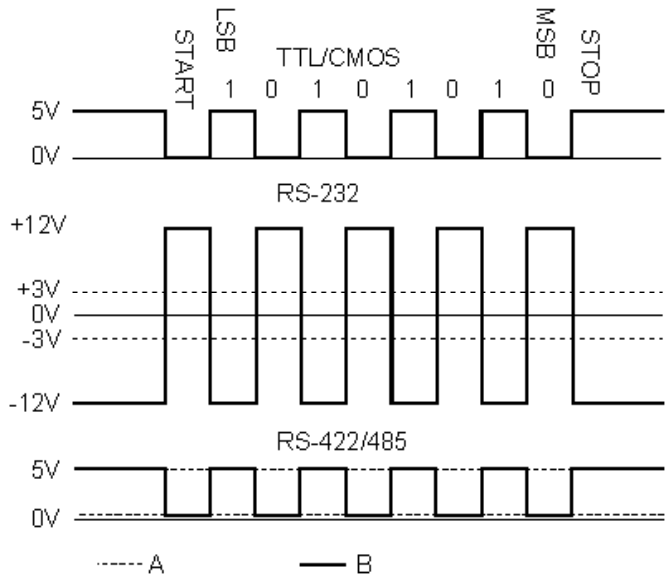
AVR Serial ports

- MAX232, MAX202 etc: electrical level conversion (no “intelligence”)
- the “intelligence” means adding the start and stop bits and removing them upon reception
- the uC contains the “intelligence”
  - it uses pins RxD, TxD
  - It inserts START and STOP bits at transmission and removes them at reception

ATMega UART

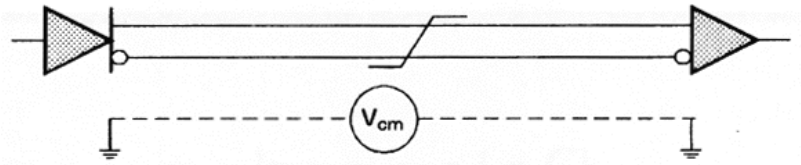


RS232 versus RS485



- RS232 asymmetric, RS485 symmetric (differential)
- RS485: there is no ground wire; A, B complementary

Differential data transmission



- V_{CM} = *common mode* = the difference between the ground level at transmitter and receiver
- *Differential* => the ground level is not important
- The **RS485** receiver compares the *voltage difference* between both lines, instead of the *absolute voltage level* on a signal line.
- It allows longer cables (hundreds of meters, even Km)
- Allows higher speeds (Mbps)

RS232, RS485 and others

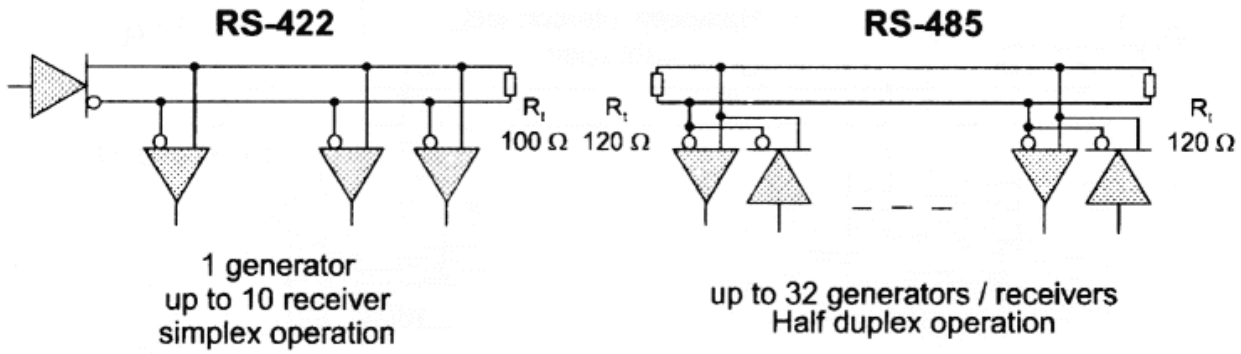
Interface	Format	# of devices (max)	Max. Length (ft)	Max. Data Rate (bits/s)
RS-232	async. serial	2	50 – 100	20K (120K, 250K & 1M with some drivers)
USB 2.0	async. serial	127	16	USB 1.0 12 M USB 2.0 480M
RS-422	async. serial	1 drivers 10 receivers	4000	10M
RS-485	async. serial	256 transceivers	4000	10M (50M w/some drives)
Firewire	isochr. serial	63	15	1394a 400 M 1394b 800M

RS422, RS485

Parameter	RS-422	RS-485
Dr. Output Max Volt	-0.25V to +6V	-7V to 12V
Dr. Output Signal Level	± 2V	± 1.5V
Driver Load Impedance	100	54
Rcr. Input Volt Range	-7V to 7V	-7V to 12V
Rcr Input Resistance	4 K	12 K

- They are the same family
- RS485 is more capable
- RS422: 1 driver => 10 receivers
- RS485: 1 driver => 32 receivers

Bus RS422, RS485



- R_t *termination resistor*
- Transmitter+receiver = transceiver (RS485)

Termination resistor

- $R_t = Z_0$
- Z_0 = characteristic impedance
  - The measured impedance if the line has infinite length
  - or if the line length is finite and is finished with $R_t = Z_0$
  - Z_0 coax: 50 or 75Ω (radio or TV)
  - Z_0 *twisted pair* = 120 Ω
- L_l, C_l = line parameters (corresponding to a long line)
- Signal propagation speed v_{signal} = usual 0.7… 0.8*c*

$$Z_o = \sqrt{\frac{L_\ell}{C_\ell}}$$

$$v_{signal} = \sqrt{\frac{1}{L_\ell C_\ell}}$$

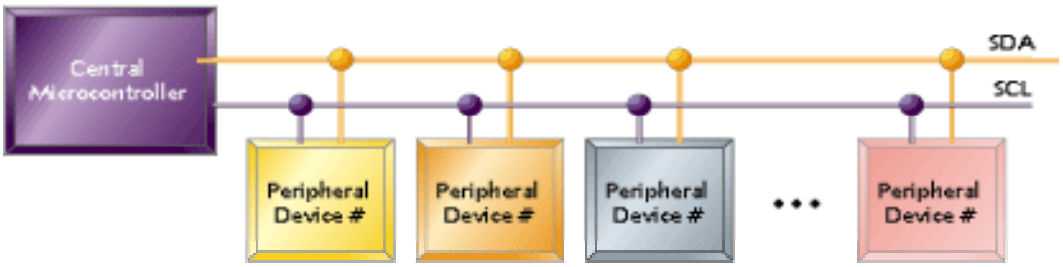
Conclusions RS485

- Specify only the hardware
- No specific flow control, ACK
- Compared to the RS232, not even specify the type of connector
- Default, RS232=Full duplex, RS485=Half duplex
- RS485 Full duplex: possible, with 4 wires !

I²C

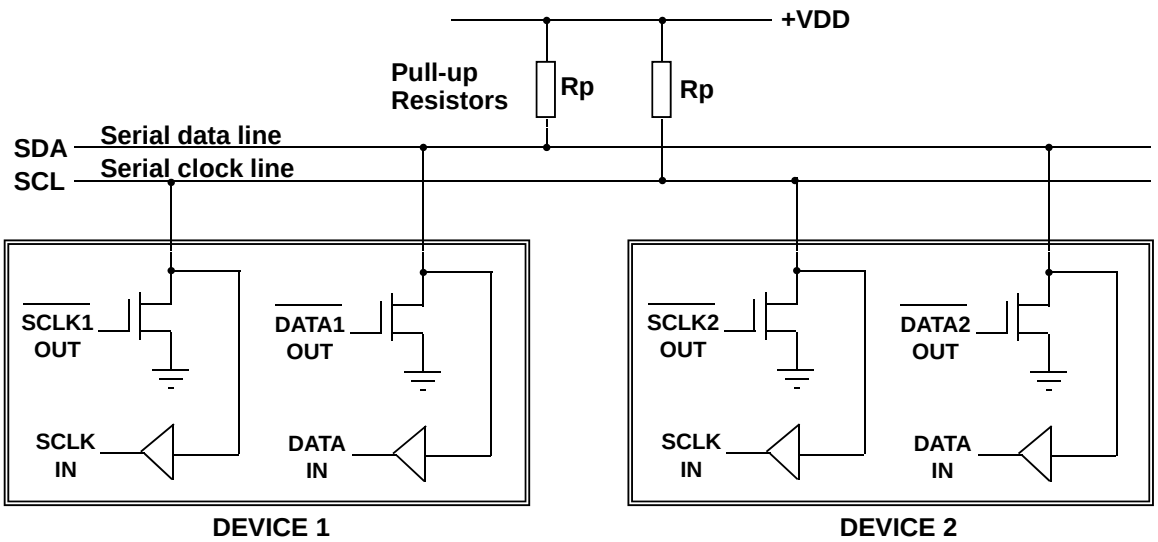
- “Inter-integrated circuit” bus
- Developed for television by Philips Semiconductor, 1980
- I²C Devices: processors, EEPROMs, sensors, real-time clocks
- Used as a control interface
- I²C devices can also have separate data interface (digital TV tuners, video decoders, audio processors ...)
- 3 speed I²C :
  - Slow (under 100 Kbps)
  - Fast (400 Kbps)
  - High-speed (3.4 Mbps) – I²C v.2.0
- Bus length: typical: inside the equipment, <1m, maximum: few meters

Bus I²C



- 2-wire (TWI) – Serial data (SDA) and Serial clock (SCL)
- Common ground wire (totally 3 wires)
- Half-duplex, synchronous, multi-master
- Without “chip select” or arbitration
- 1 logic: pull-up resistors
- 0 logic: open-collector or open-drain driven
- Equivalent to wired-AND

I²C connection schematics



I²C Protocol

1. Master sends “start condition” (S) on SDA and clock is generating on SCL
2. Master sends the slave address (7 bits)
3. Master sends bit read/write (R/W)
   - 0 - slave will receive,
   - 1 - slave will transmit
4. The transmitter (slave or master) sends bit ACK
5. The transmitter sends 1 byte of data

I²C Protocol (cont.)

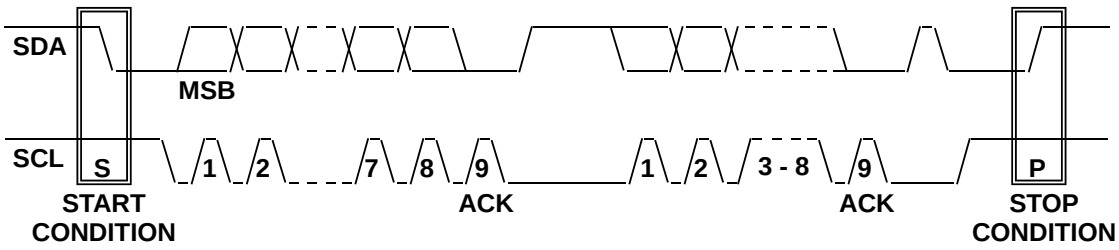
6. The receiver sends an ACK bit for byte received
7. Repeat steps 5 and 6 if more bytes are sent
8. a) for write transaction (master = transmitter) sends stop (P) after the last byte of data
   b) for read transaction (master = receiver), does not send ACK for last byte, just sends stop (P) to signal slaves that transmission ended

I²C Signals

Sample Signals

- Start (S) – falling edge on SDA when SCL is “1”
- Stop (P) – rising edge on SDA when SCL is “1”
- ACK – the receiver “pull down” SDA to 0 (the transmitter leave SDA=1)
- Data – SDA when SCL=0; valid transmission when SCL=1

Example of a data transmission



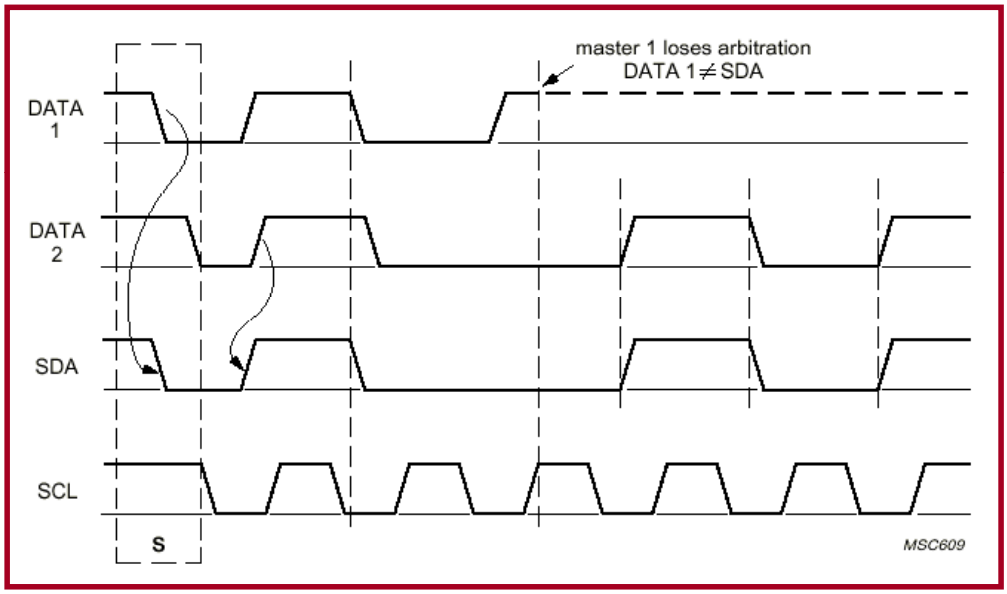
I²C Properties

- “Clock stretching” – when the receiver needs time, pull SCL to 0, the transmitter waits until it restores SCL 0 before transmitting a new bit
- “Clock synchronization” – when there are two masters with different clock, there is a synchronization procedure so that both generate the same clock
- Broadcast “General call” – all slaves on the bus are addressed at 0000000 address
- Initial address: 7 bits; extended address: 10 bits (first 7 bits are 11110xx

Multi-master

- Arbitration procedure
- It done every bit
- A single master "wins" the arbitration
- Master that loses arbitration waits until the line is free and try again

Multi-master arbitration



- Each master generates its bits
- Each master checks if the line is what itself has generated
- Arbitration is lost by first master that detects the first mismatch

Atmel AVR I²C Suport

- Atmel: “Two-wire Serial Interface” (TWI)
- All AVR 8-bit except ATTiny and AT90

TWI mode when TWEN bit form TWCR register is 1

- TWBR – bit rate
- TWCR – start, stop, generate ack, M/S, T/R
- TWDR – transmitted/received
- TWAR – slave address
- TWSR – bus status (start condition transmitted, ACK received, … 26 states)

I²C - conclusions

Advantages:

- Useful for devices that communicate occasionally
- Addressing scheme allows multiple device interconnection without additional wires

Disadvantages:

- Hardware and especially software implementation more complicated than the SPI
- Half-duplex
- Not scalable for large number of devices

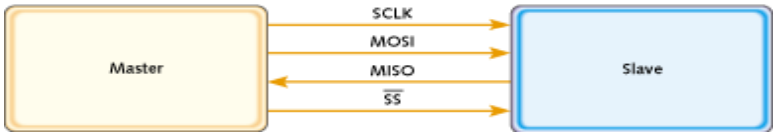
SPI

- “Serial Peripheral Interface”
- Defined by Motorola for MC68HCxx
- Faster than I²C, (by few Mbps)

Applications:

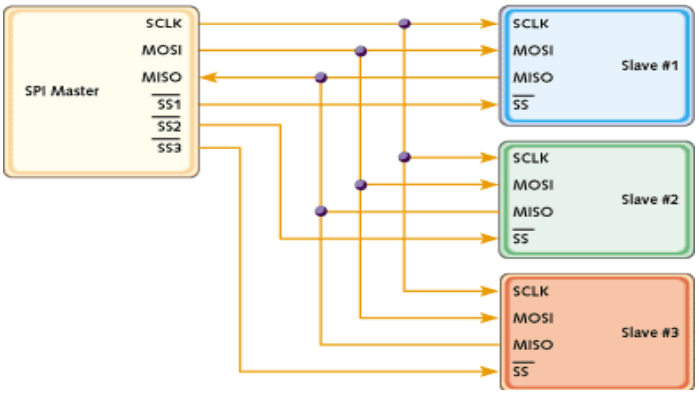
- Like I²C, is used for EEPROM, Flash, real-time clocks...
- Suitable for devices that send continuous flow of data, such as ADC
- Full duplex, suitable for communication between a codec and a signal processor (DSP)

SPI



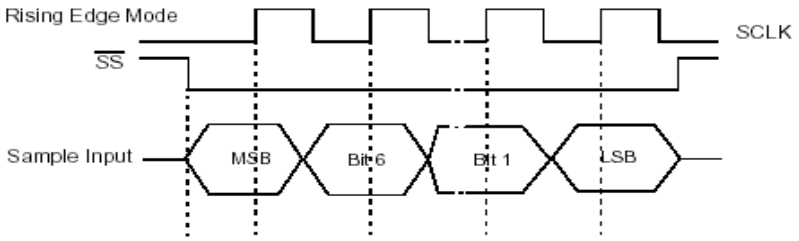
- Synchronous Bus
- Master / slave relationship
- 2 data lines:
  - MOSI – master data output, slave data input
  - MISO – master data input, slave data output
- 2 control lines:
  - SCLK – clock
  - /SS – slave select
- SS means no addressing

SPI vs. I²C



- For Point-to-point, SPI is more simple and efficient
  - Less overhead than I²C due to no addressing.
- For more slaves, each slave needs a SS line
  - More connections than I²C

SPI Protocol



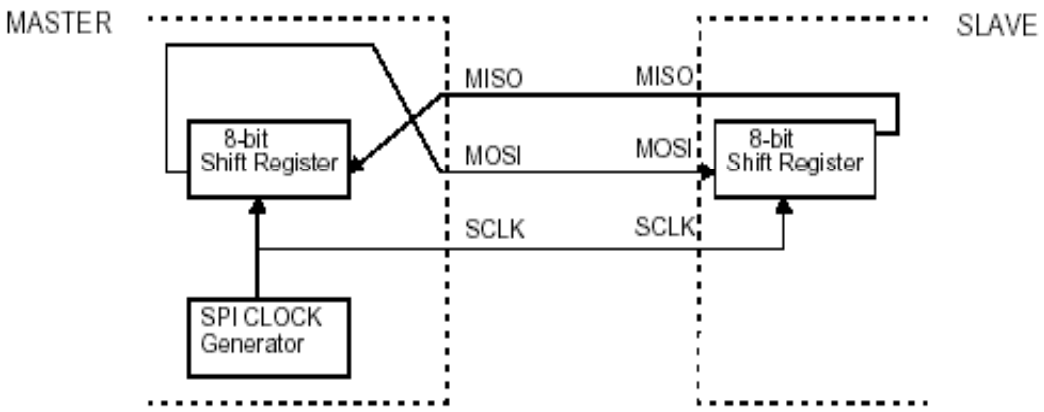
- 2 parameters, Clock Polarity (CPOL) and Clock Phase (CPHA), determine active edge of CLOCK

CPOL	CPHA	Active edge
0	0	Rising
0	1	Falling
1	0	Falling
1	1	Rising

- Master and slave must be configured with the same set of parameters, otherwise can not communicate

SPI Protocol (cont.)

- SPI interface only defines the communication lines and the clock used
- No ACK flow control or ACK mechanisms built in



AVR support for SPI

- Supported by all AVR except ATTiny and some AT90s

SPI Mode when SPE bit in SPCR is 1;

- SPCR – bit rate, CPOL, CPHA, M/S
- SPDR – transfer data to and from SPI shift register

Conclusions, SPI and I²C

- I²C and SPI are highly used for communication with peripherals like serial EEPROMs, clocks, etc.
- I²C vs. SPI:
  - SPI is preferred for higher speed
  - and I²C for the simple routing