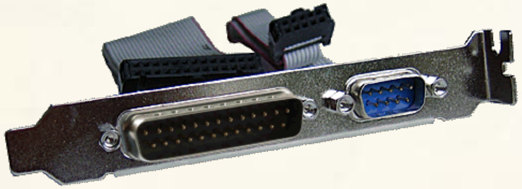
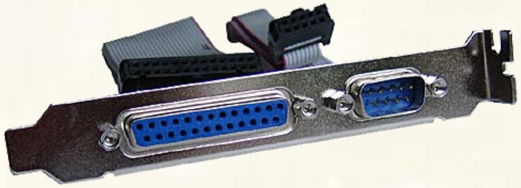
RS232 (TIA/EIA-232-F)

- Historical (1962, revision C dates from 1969, revision F 1997) universal serial peripheral interface
- The maximum recommended speed was 20 kbit/s, PC implementation had 115 200 b/s, some specific implementations have more
- The maximum range was 15 m (revision C), often had been reported 25 m; according to the revision F the range is limited to a maximum cable capacitance of 2500 pF, with a low capacitance cable it can reach up to several times larger distance, but at lower speed (900 m at a speed of 2.4 kBaud)
  If we want to reach greater distances, we can use the 20 mA current loop converter (up to 6 km), or use the RS-422 or RS-485 bus
- Asynchronous data transfer; each side has its own clock generator, which must be synchronized before transmission of each byte
- In the past, this interface was commonly used to connect the computer (DTE) with a modem (DCE), but more than 10 years ago began to be replaced by USB. So far it is used mainly for special measuring devices (e.g. electrical inspection devices), at point of sale (POS) systems (e.g. IBM SurePOS 500) including peripherals (touch screens, customer displays, card readers, thermal printers, ... – although even here USB begins to dominate) and industry. Many microcontrollers supports this interface.
- In the PC based on UART chips, i8250 / 16450 / 16550 compatible, COM ports (Communications port, COM1 – COM4)



Former PC bracket has 2 RS232 connectors – original DB25M and later DE9M
(also known incorrectly as DB9P)



More recent version with DB25F and DE9M
DB25F is not serial, but parallel bus LPT

Replacement of missing RS232 port by USB/RS232 converter based on FT232XX chip

D subminiature connectors: Canon, 1952; Marking: **DB25M**

D – indicating the type of connector
(D subminiature)

B – connector size; the same as number of contacts *in 2 rows*

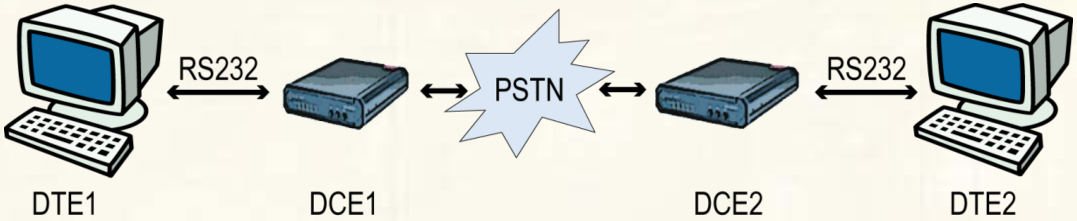
A …	15
B …	25
C …	37
D …	50
E …	9

M – connector design / gender
M (male) / P (plug)
F (female) / S (socket)

25 – the actual number of contacts

RS232 (TIA/EIA-232-F)

DTE / DCE devices



The RS232 interface was originally designed to connect between the terminal and modem to communicate through some other technology (such as public telephone networks)

DTE – Data Terminal Equipment ~ PC; male gender of connector

DCE – Data Communication Equipment ~ modem; female gender of connector

RS232 (TIA/EIA-232-F)

There is no galvanic isolation, grounded circuits in the PC are connected directly to grounded circuits of connected peripherals
Hot plug / unplug was not recommended, at least…
At various outlets *(in the case of different of distribution points)* the ground voltages may differ by many (dozens) volts
– the risk of interface destruction!

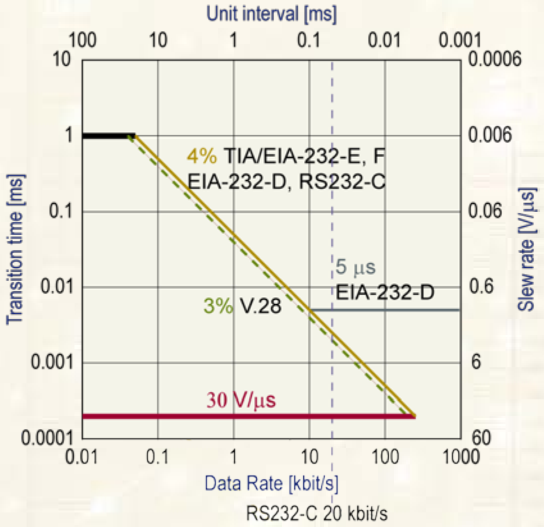
Electrical Characteristics

- TIA/EIA-232-F standard allows maximum driver voltage (open circuit) 25 V, loaded 5 … 15 V
- At receiver side the minimum input level is 3 V (undefined "dead zone"; the voltage drop between transmitter and receiver is on the cable resistance)
- The typical driver voltage (loaded) was 15 V (but PC uses, due to the supply voltage, 12 V, present drivers are often supplied from 0/5 V source, the output voltage is generated by a regulated charge pumps on voltage level converter chips; the voltage is then 10 V or less)

`-15 V … -5 V:`	logic 1
`+5 V … +15 V:`	logic 0
`-3 V … +3 V:`	undefined *(transition region)*

The maximum allowable potential difference of the earths: 2 V
(hence ±3V and ±5V the limits on receiver and transmitter)
Recommended transmitter voltage levels: -15 V … -5 V / +5 V … +15 V

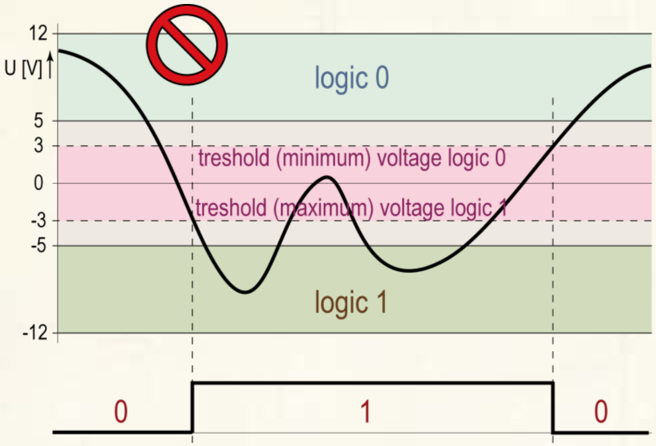
- Input impedance on receiver side 3 – 7 kΩ
- Output current *(short circuit tolerance!)*
  standard allows output current to 500 mA, IO have typically 10 … 30 mA
- Max. cable + input capacitance 2500 pF – limits the maximum cable length
- Timing requirements:
  - Maximum slew rate 30 V/µs
  - Rise time through the transition region 1 ms (below 40 b/s),
    4% of Unit Interval — 30 bit/s to 20 kbit/s (depends also on a version)
  - Signal should not change its direction in the transition region



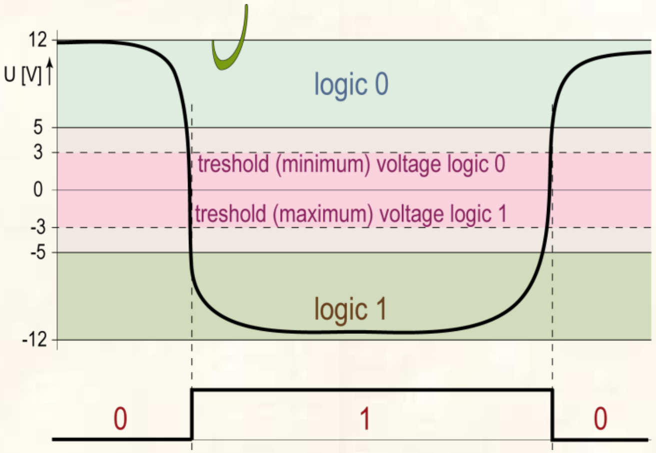
The relationship between transition time and data rate on RS232

RS232 (TIA/EIA-232-F)

Good and wrong voltage waveform in a transition region



12
U [V]
5
3
0
-3
-5
-12
logic 0
treshold (minimum) voltage logic 0
treshold (maximum) voltage logic 1
logic 1
0
1
0



12
U [V]
5
3
0
-3
-5
-12
logic 0
treshold (minimum) voltage logic 0
treshold (maximum) voltage logic 1
logic 1
0
1
0

Although you can occasionally find in literature a picture similar to that on the left, which illustrates the meaning of threshold voltage levels, with a commentary the receiver accepts it as a sequence of a logical 0 1 0, this shape does not satisfy the requirements both by its slope and swinging in logic 1. Present circuits differs in their voltage levels, especially the integrated one, so they may understand the signal in a different ways.

RS232 (TIA/EIA-232-F)

Signal pins of DE9 connector *(front view)*

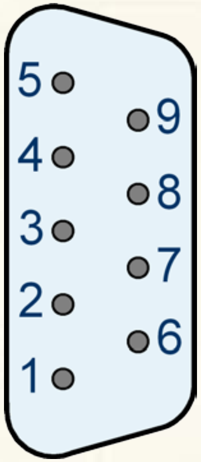
SG – Signal Ground

DTR – Data Terminal Ready

TD – Transmit Data

RD – Receive Data

DCD – Data Carrier Detected

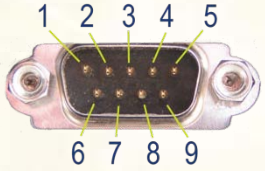


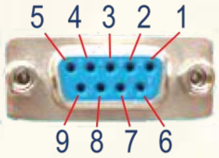
RI – Ring Indicator

CTS – Clear To Send

RTS – Request To Send

DSR – Data Set Ready





Signal	Meaning
DCD	*Carrier detected*: DCE is now connected to the phone line *(local modem detect carrier of a remote modem)*
RD (RxD)	*Receive data*: transmission of series data from modem into DTE (e.g. PC)
TD (TxD)	*Transmit data*: transmission of series data from DTE into DCE (modem)
DTR	*Data terminal ready*: DTE informs DCE, it is ready for data transmission *(e.g. the PC wants to use modem)*
SG (GND)	*ground*: signal ground
DSR	*Data set ready*: DCE (modem) informs, it *(finish negotiation with remote modem and)* is ready to transmit data
RTS	*Request to send*: DTE informs modem, it is ready to transmit data (and wait for confirmation by CTS signal)
CTS	*Clear to send*: modem give permission to DTE to send data (when it returns back to logic 0 it informs, it isn’t possible to continue in data transmission /e.g. modem buffer is full/)
RI	*Ring indicator*: informs, that modem receive an incoming tone signal
PG	Protected Ground: is connected to the connector’s body and shielding

RS232 (TIA/EIA-232-F)

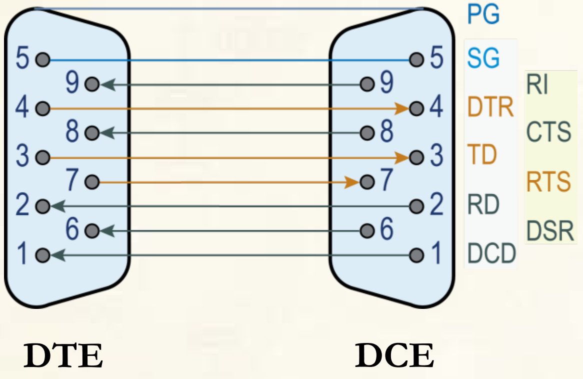
Basic cable types

Standard DB9 pinout

Direct connection of DTE to DCE device

Female (on DTE), male (DCE) connector

View from cable side

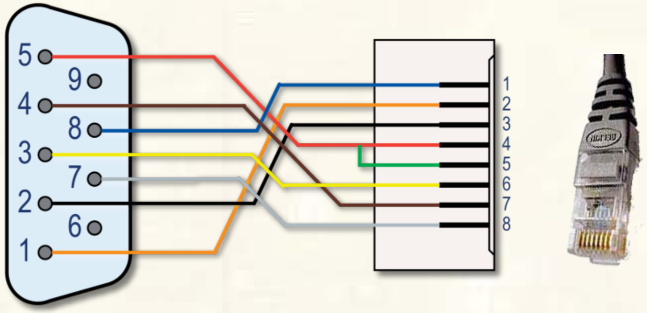


DB9F / RJ45

EIA-232-D

DB9F, male RJ45 connector (but also DB9M is used)

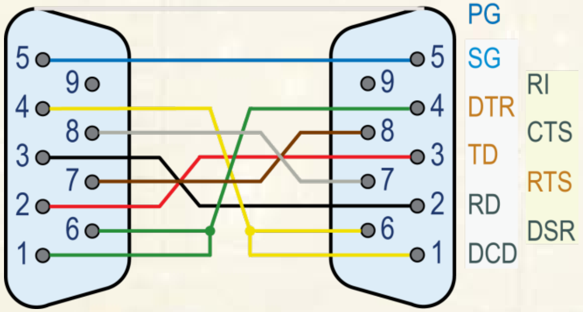
View from cable side / upper view



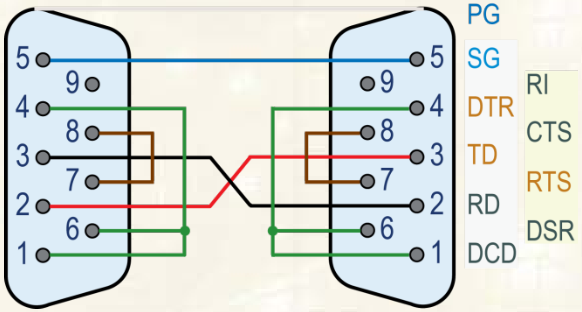
DB9 null-modem

Direct connection of two DTE devices (PC) without modems (female / female), with and without handshaking

Full

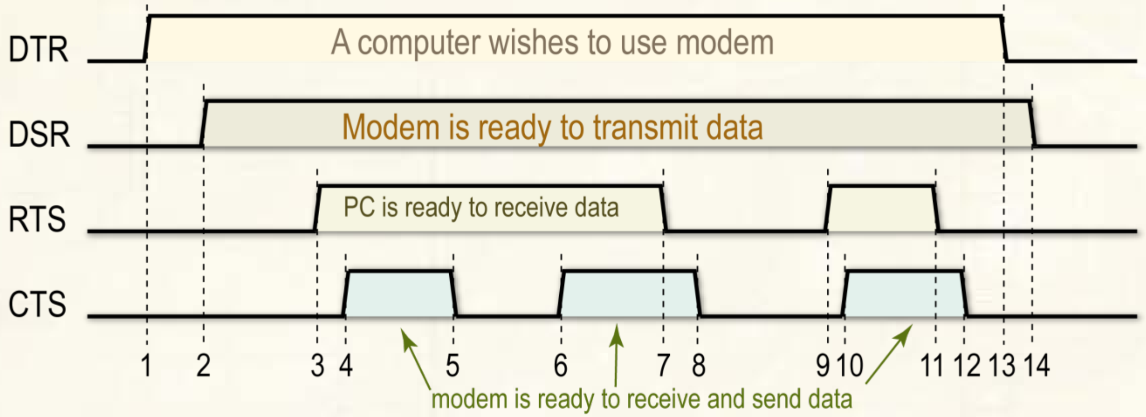


Minimum



RS232 (TIA/EIA-232-F)

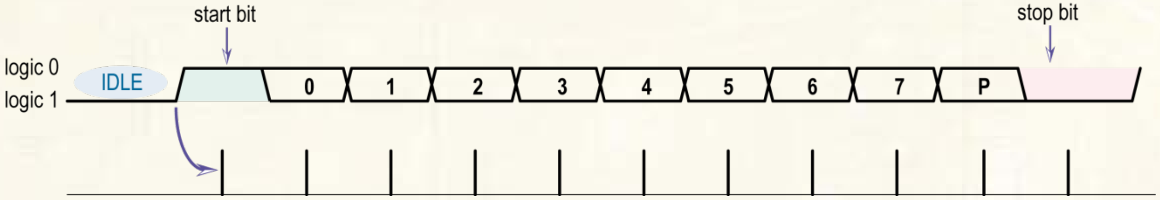
The basic control signals, and their sequences


DTR
A computer wishes to use modem
DSR
Modem is ready to transmit data
RTS
PC is ready to receive data
CTS
1
2
3
4
5
6
7
8
9
10
11
12
13
14
modem is ready to receive and send data

1. DTE device asserts DTR signal (logic '0', positive voltage) when it wishes to open a communications channel
2. DCE device (modem) confirms by setting DSR signal to logic '0' that:
   a) The modem is connected to an active telephone line
   b) The modem is in data mode, not voice or dialing mode
   c) The modem has completed dialing or call setup functions and is generating an answer tone
3. DTE asserts RTS signal, when it is ready to send data to modem; DCE may spend some time by preparation for transmission
4. DCE asserts CTS signal, when it is ready receive data from computer
5. DCE device is not able to receive any other data from DTE at this moment (e.g. when its buffer is full), so it asserts logic '1' (low voltage) and DTE device must stop data transmission
6. At this moment DCE is again able to receive new data from DCE, so it asserts logic '0' again on CTS signal
7. At this moment DTE is not able to receive data from modem, so it signals this to DCE device asserting RTS signal to logic '1' *(DCE mustn't send any data into DTE)*
8. DCE acknowledge the message from the step 7 asserting CTS signal low (logic '1')
9. DTE is able to receive data again, so it asserts RTS signal to logic '0'
10. DCE acknowledges it is also able communicate
11. DTE is again stopping communication
12. DCE acknowledges
13. DTE is finishing communication (requests modem hang-up)
14. DCE acknowledges the communication is finished *(modem is hang-up)*

RS232 (TIA/EIA-232-F)

Asynchronous transmission on RS-232

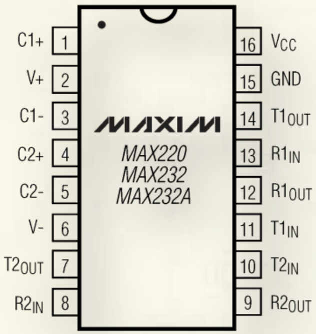


☞ The basic condition is that the receiver and transmitter is configured with the same data transfer rate

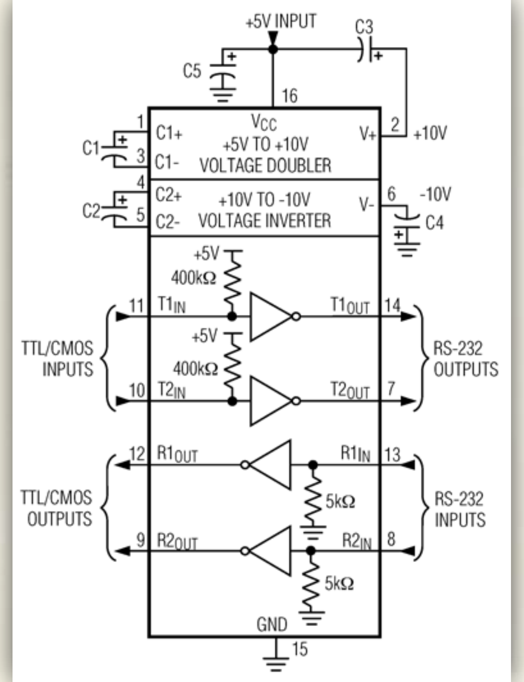
- **IDLE**: logic 1 („marking state“)
- **Start bit**: initiates the transmission of one byte; line goes into a logical 0 („space state“)
  At this moment the internal clock generator of receiver is synchronized; its counter is reset, and it will be used to generate internal sampling pulses, which controls sampling of distinct bits values (in ideal case) in the middle of their periods
- **Data bits**: usually 5 – 8, LSB is transmitted first
- **Parity bit**: not mandatory, both transmitter and receiver must use same type of parity
  The ODD parity was used (the number of mark bits + parity bit = odd number, EVEN parity, SPACE parity – always logic 0, and MARK parity – always logic 1)
- **Stop bit**: logic 1 (mark state) (the same as idle state); serves to bring the receiving device to rest in preparation for the reception of the next symbol (receiving device may have slower clock generator, so then it may miss next start bit), some receiving devices need distinct time for processing of just received symbol.

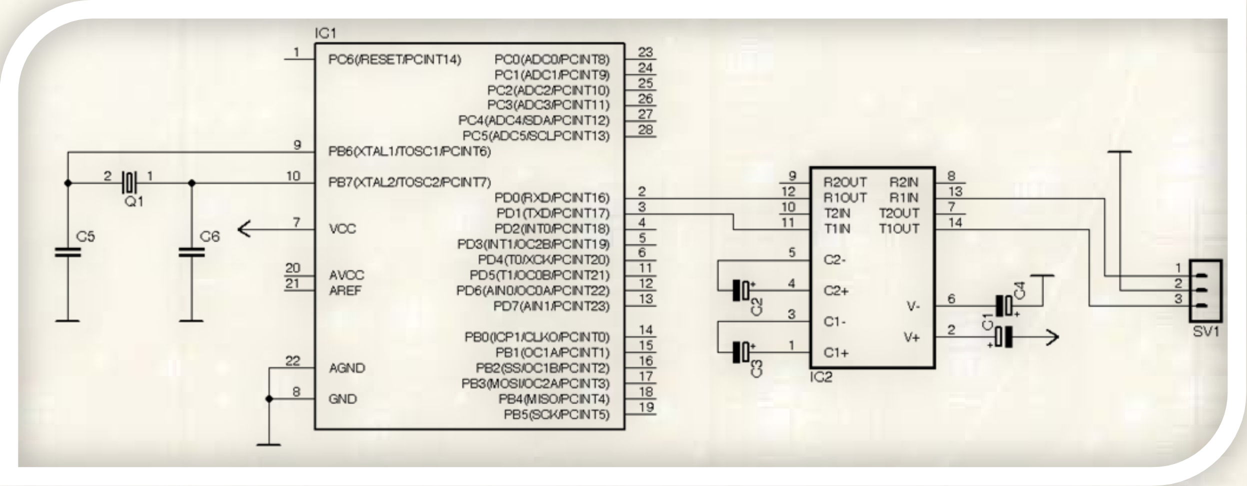
RS232 (TIA/EIA-232-F)

Voltage level converters (RS232 interfacing circuits)



- Although many µCs have UART logic interface, it use µC voltage levels, not compatible with RS232 standard
- These circuits converts TTL (CMOS) voltage levels on RS232 voltage levels and oppositely
- The best known include Maxim's MAX232 (and its variants), the National Semiconductor's DS14C232, their clones are produced by a number of other companies
- Usually the circuit is supplied by the source +5 V / 0 V, required positive and negative voltage is generated internally by a voltage doubler / inverter on the principle of a charge pump (capacitors C1, C2)
- Voltages +10 V / -10 V (open circuit) are available for supplying of a low-drain devices on pins 2 and 6 *(but with increasing loaded power the voltage, of course, drops!)*





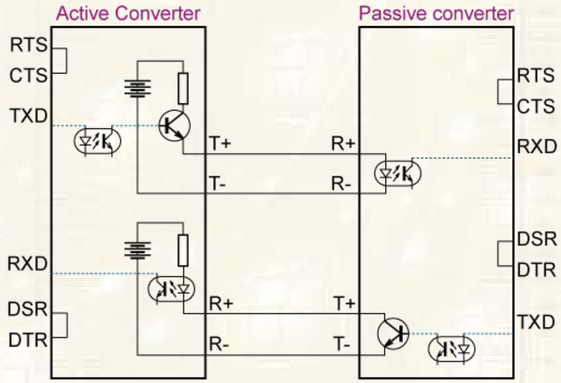
Example of connection of a MAX 232 to µC (ATMEGA168-20PU)

Internal: typical operating circuit of MAX 232

RS232 (TIA/EIA-232-F)

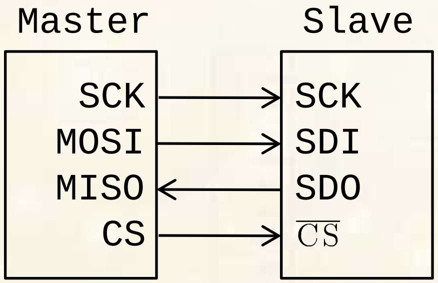
Current loop

- To extend the range of the RS232 bus the current loop 0/20 mA may be used. But, don't confuse it with an analog 4/20 mA current loop.
  - Analog current loop is used mainly to transmit control signals to valves, pumps, ..., or, conversely, to receive measured values from sensors, etc.
  - Through analog current loop may pass arbitrary current in an interval ⟨4, 20⟩ mA, which is equivalent to ⟨0, 100⟩% (deviation, value measured by a sensor, …).
    Zero current indicates a malfunction.
  - Digital current loop has only two states – in the current loop flows the current of 20 mA (logic 1 / mark state of RS232) / or no current flows in the current loop (logic 0 – space state of RS232)
    There is no special value which may indicate a malfunction.
    Digital current loop is often supplied from common voltage source with such internal resistor, which resistivity adjust the current in current loop at 20 mA (supposing zero resistivity of cabling, but counting the voltage drop across LED diode) – e.g. 12 V voltage source of internal resistivity of 470 Ω, in controlling circuits usually 24 V); with resistivity of cabling passing current drops.
- Maximum range is limited by a cable capacitance and resistivity, usually the maximum is 200 Ω, practically the range may be from few hundreds meters up to 6 km
- The advantage of a current loop is noise immunity. Galvanic isolation is common.
- Converters are passive (the transmitter and receiver is optically isolated, but there is no source) or active (the voltage / current source is included); it is necessary connect one active to one passive converter, or two passive connect to external voltage source
- Typical current loop transmits only TxD, RxD signals, no control signals - another current loop must be used if controlling signals are needed.



SPI BUS

- Serial peripheral bus (Microwire)
- Developed by Motorola, it has no official specification; used by many companies; 4 different communication modes (SPI 0 … SPI 3)
- SPI dedicated to fast communication among IC chips, so that **it is used just only on printed circuit boards**, or for communication with nearby peripherals (e.g. control objectives of digital cameras)
- Designed for synchronous serial communication between the microcontroller and peripheral circuits
  - Microcontrollers:
    - Motorola: MC68HC11A8, …
    - Atmel: used as programming interface on AVR microcontrollers (In-System Programming)
    - Microchip: PICmicro MCU (PIC16F876, …)
  - A/D converters (programming, not data transmission - Intersil KAD5512, …), slower D/A converters (Maxim)
  - Potentiometers (Intersil ISL22424, …)
  - MMC and SD cards, EEPROMs
  - FLASH memories (BIOS on PC mainboards, connected via ICH)
  - EF objectives Canon
- If the processor has no hardware support for SPI interface, it can be implemented in software using four pins of one of its ports
- Data are latched on an edge of the clock signal SCK (master) and updated on opposite edge of SCK (slave)



3+1 signal:

SCK:	Serial Clock, generated by Master
MOSI/SDI:	Master Output Slave Input/Serial Data In
MISO/SDO:	Master Input Slave Output/Serial Data Out
CS:	Chip Select – selects the device, substitute an addressing *on master sometimes called SS (Slave Select)*

- Data transfer rate typically up to 10 Mbit/s
- 1 master / 1 or more slave devices, multimaster possible but complicated

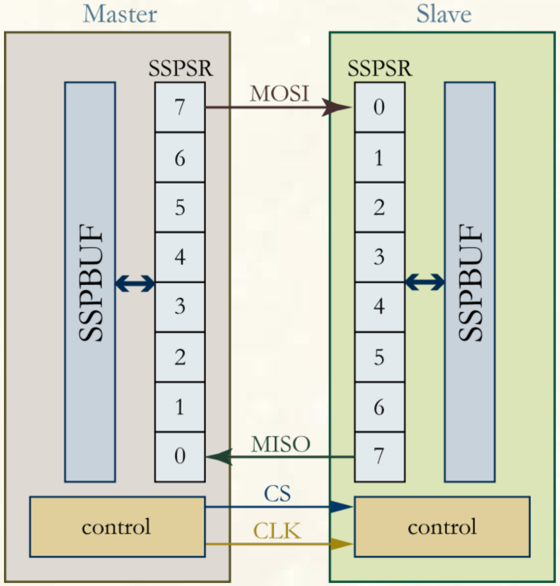
SPI BUS

Principles of operation of a SPI bus

When data are transferred, the SSPSR register is shifted bit by bit, MOSI and MISO transfers bits simultaneously from and into master, starting from MSB

Before the transfer begins, single byte is copied from SSPBUF data buffer into shift register SSPSR

Before the transfer begins, single byte is copied from SSPBUF data buffer into shift register SSPSR



After the transmission is finished, data from shift SSPSR register must be copied into SSPBUF data buffer asap;

It must be done before next transmission – but master controls communication

After the transmission is finished, data from shift SSPSR register must be copied into SSPBUF data buffer immediately;

It must be done before next transmission, or data in SSPSR are lost – slave doesn't know, when new transmission begins

The transmission is always initiated by Master – it pulls CS low; this is usually controlled by a μC program

Only Master controls clock signal, its frequency is not (technically) important

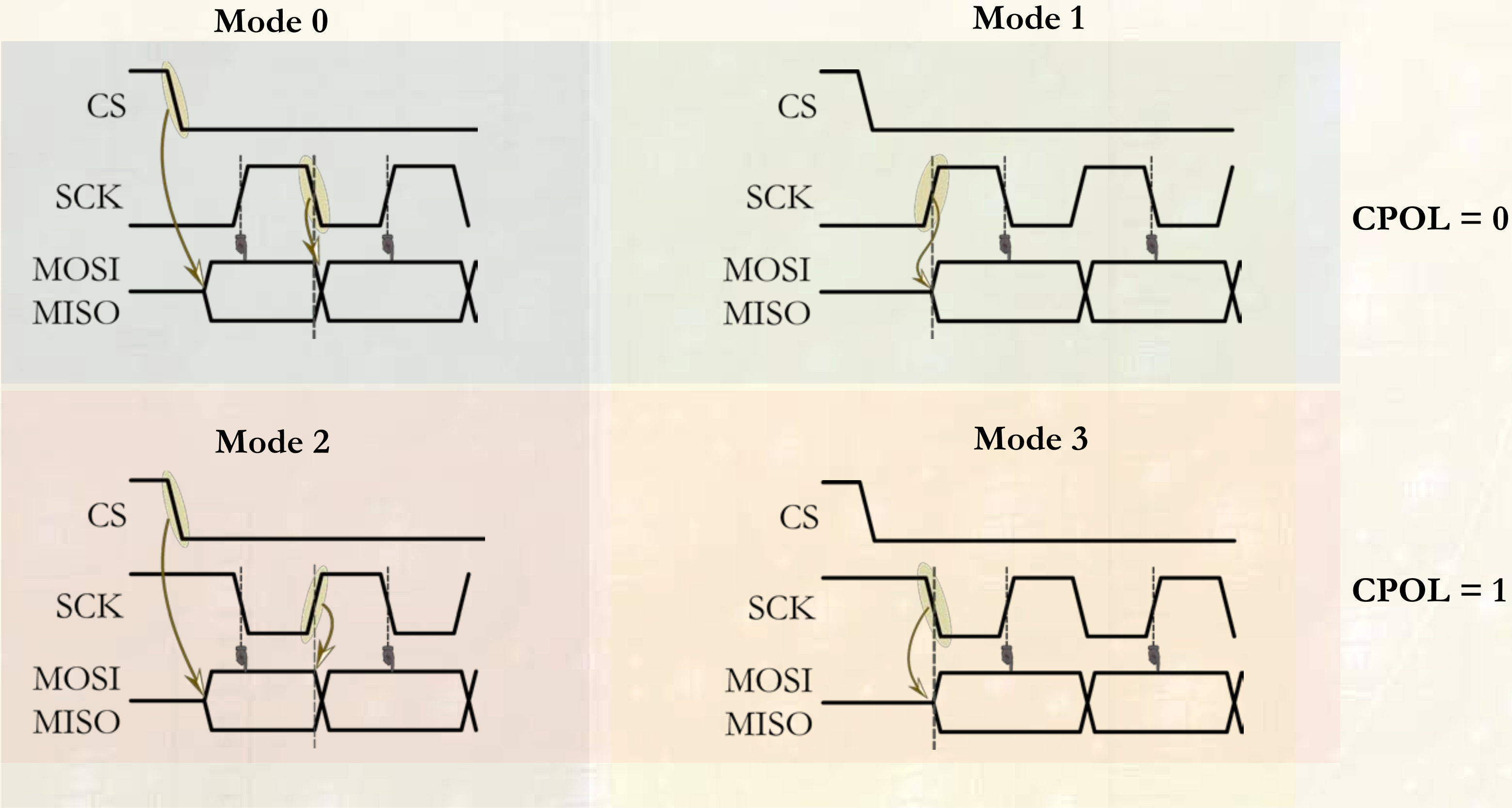
Slave waits for CS is asserted low, which activates it; when not selected (CS is high), the Slave is IDLE (tri-state data pins are in a high-impedance state)

Since slave doesn't know, when new transmission begins, SSPSR register must be loaded by new value to transmit immediately

SPI BUS

4 different SPI modes

- According to clock polarity and phase SPI has 4 transmission modes:
  - CPOL means idle value of a clock signal
  - CPHA determines the phase (edge) of a clock signal, which latches the data
    - CPHA = 0: rising edge, if CPOL = 0, but falling when CPOL = 1
    - CPHA = 1: opposite case



CPHA = 0

CS must be asserted high after each transmitted byte, since its high-low transition controls data pins state

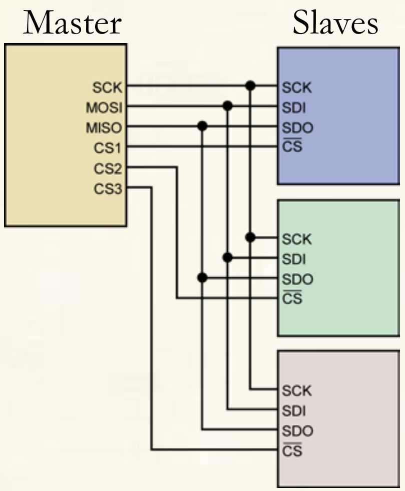
CPHA = 1

CS may remain low, if next byte will be transmitted, because SCK transition controls data pins state

SPI BUS

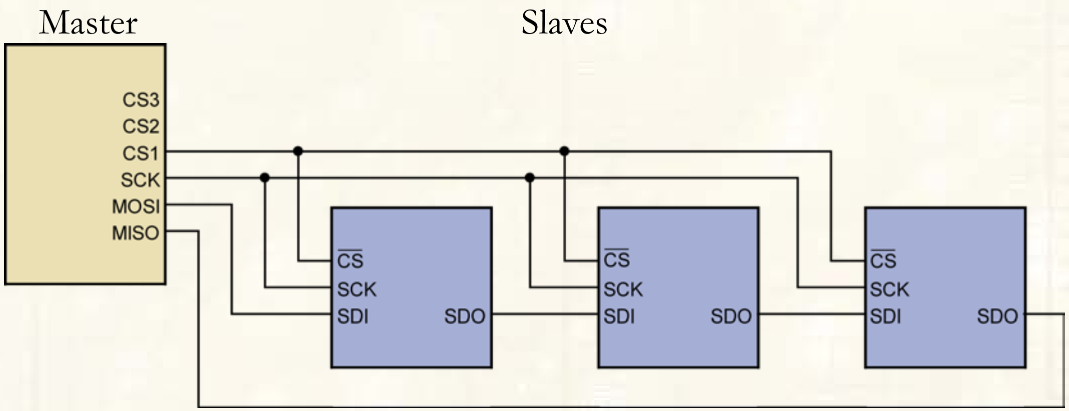
SPI bus configuration

Parallel



- Is possible to connect different devices
- Each device require one CS signal ⇒ number of connected devices is limited particularly by a number of pins, dedicated on master to CS

Daisy chain



- All slaves have chained shift registers – so isn't possible (or just with big problems) to combine different types of devices within single chain
- Daisy-chained slaves require the master to provide only one slave-select signal
- It is possible to combine both parallel and daisy chain structures on single μC

SPI BUS

- ✓ Simple hardware interfacing, no special requirements for cabling and its impedance matching (to prevent reflections), no external circuitry required (like pull-up resistors, nevyžaduje žádné externí obvody (pull-up rezistory, external bus driver transistors, signal edge controllers, ...)
- ✓ Simple protocol to implement, even in software
- ✓ Full duplex communication
- ✓ Slaves use the master's clock, and don't need precision oscillators
- ± Addressing not needed, replaced by a CS signal – higher data rates, simple implementation, at the expense of versatility, possibility of a hot-plugging, and identification of connected devices
- ± In theory not limited to 8-bit words (in practice limited by shift registers bit length, usually 8 bit)
- ✗ The more devices you have the more μC pins and connections necessary (CS signaling)
- ✗ No slave acknowledgment (master may even “talk” to nothing and not know it)
- ✗ Relatively short distances a few dozens cm (at higher speeds) – limited mostly on PCB communication
- ✗ Does not support multi-master architecture

SPI System Errors

Some (but not all) μC may detect some bus errors

- If the CS pin on a master device go low, it means another device on a same bus try to acts like master device ⇒ „mode fault error“; if a device detecting this error has hardware SPI support, bus drivers are disconnected and interrupt is generated
- Write collision error – if any data is written in SSPBUF (but e.g. in MC68HC11D3 this register is described as SPDR) when transmission is running, data is lost and error is generated

SPI BUS

Example of a software implementation of SPI bus using four pins of port 0 on 8051 processor in C language:

The function receives 8 bit character as a data to send, activates slave via CS signal and transmit (and receives at the same time) character and deactivates slave

```
// assign bits of port 0 to SPI bus signals

sbit MOSI = P0^0;   // Master Out / Slave In (output)
sbit MISO = P0^1;   // Master In / Slave Out (input)
sbit SCK  = P0^2;   // Serial Clock
sbit CS   = P0^3;   // Chip (Slave) Select

char SPI_Transfer (char SPI_byte) {
    unsigned char SPI_count;
    CS = 0x00;
    for (SPI_count = 8; SPI_count > 0; SPI_count--) {
        MOSI = SPI_byte & 0x80;
        SPI_byte = SPI_byte << 1;
        SCK = 0x01;
        SPI_byte |= MISO;
        SCK = 0x00;
    }
    CS = 0x01;
    return (SPI_byte);
}
```

SPI BUS

Example of SPI bus implementation using four bits of port 0 on 8051 processor in assembler:

```
// assign bits of port 0 to SPI bus signals

MOSIBIT      P0.0                ; Master Out / Slave In (output)
MISOBIT      P0.1                ; Master In / Slave Out (input)
SCK   BIT    P0.2                ; Serial Clock
CS    BIT    P0.3                ; Chip (Slave) Select

SPI_Transfer:
      CLR CS
      MOV A, R7
      MOV R7, #08H
      RLC A
SPI_Loop:
      MOV MOSI, C
      SETB SCK
      MOV C, MISO
      RLC A
      CLR SCK
      DJNZ R7, SPI_Loop
      SETB CS
      MOV R7, A
      RET
```

SPI BUS

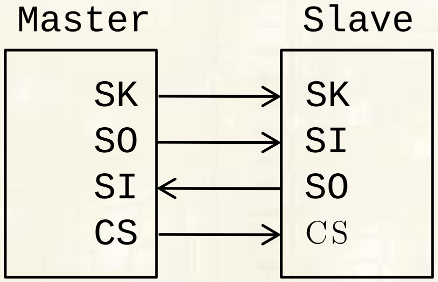
Example of a piece of code for hardware implementation SPI bus using interrupt on AT89S8252:

```
void SPI_ISR(void) interrupt 4 {
    SPI_DATA = SPDR;    // save received byte in variable
}


…
IE = 0x90;          /* Interrupt control register 10010000; Global interrupt + ES (SPI) interrupt enable */
SPCR = 0xD3;        /* 11010011 enable SPI interrupt (together with ES = 1 in IE register), it activates SPI bus,
                    MSB first, the device is master, CPOL = CPHA = 0 -> mode 0, frequency fclk/128 */
…
if (SPSR != 0x80)       /* status register; … if no transmission active; SPIF bit is set before interrupt is set
                          -> is not necessary to check it again in interrupt service routine */
    SPDR = SPI_DATA;    // writing in data register starts new transmission on SPI bus
…
```

MICROWIRE BUS

- Initially developed and used by National Semiconductor, older than SPI
- Very similar to SPI, sometimes is (*although not quite correctly*) described as its subset with parameters CPOL = 0, CPHA = 0
- Application: similar to SPI bus (some microcontrollers have hardware implementation, which can be configured both in SPI and Microwire mode)
    - A/D converters
    - Serial memories EEPROM (e.g. 93XXXXX; *different producers, e.g.. Microchip 93LCS56/66, Atmel AT93C66, ...*)
    - Display drivers (e.g. COP472 National Semiconductor)
    - Microcontrollers COP8, National Semiconductor
    - Also exists hardware interfaces, which allows Microwire communication even to general processors including Intel and Motorola –e.g. TP3465 MICROWIRETM Interface Device (MID) from National Semiconductor
- The number of bits can be different in distinct communication blocks;
  e.g. serial EEPROM AN993 (Microchip) may have 11 bits (command which allows EWEN / or prohibits write into memory EWDS), or 27 bits (write/ read data; however 11 bits of command + address must be of constant length, and command bits (if command has no address) must be completed to 11 bits; when reading / writing, 11 command and address bits are followed by 16 data bits)



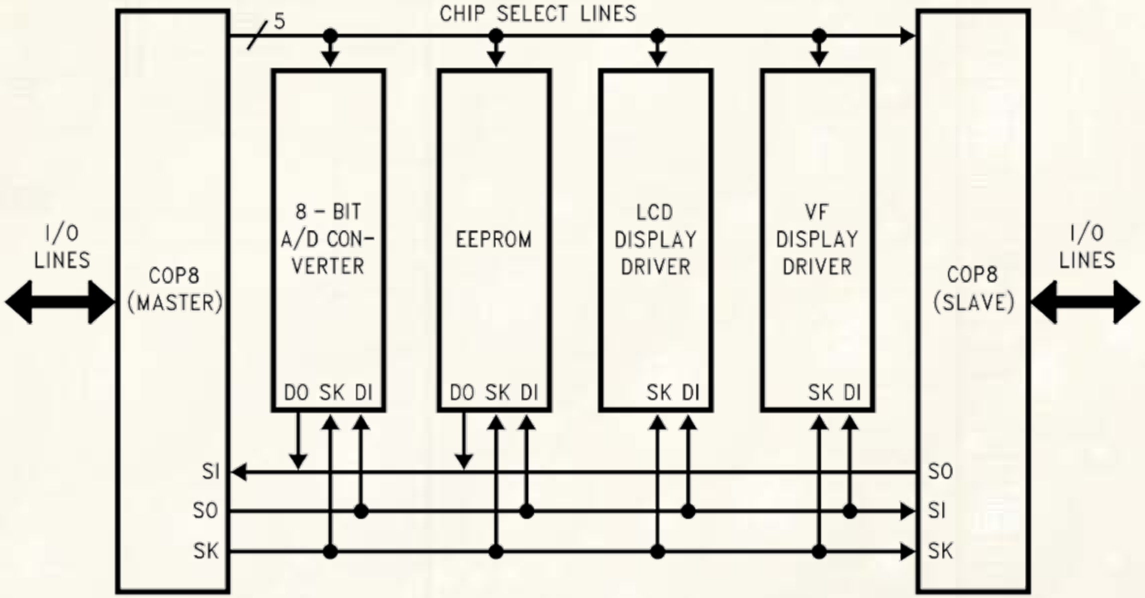
3+1 pin:

SK:	Serial Clock, generated by Master
SI (DI):	Serial Input (Data In)
SO (DO):	Serial Output (Data Out)
CS:	Chip Select – selects the device, substitute an addressing can be active at logic 0 (National Semiconductor), or logic 1 (Microchip, Atmel, ...)

MICROWIRE BUS

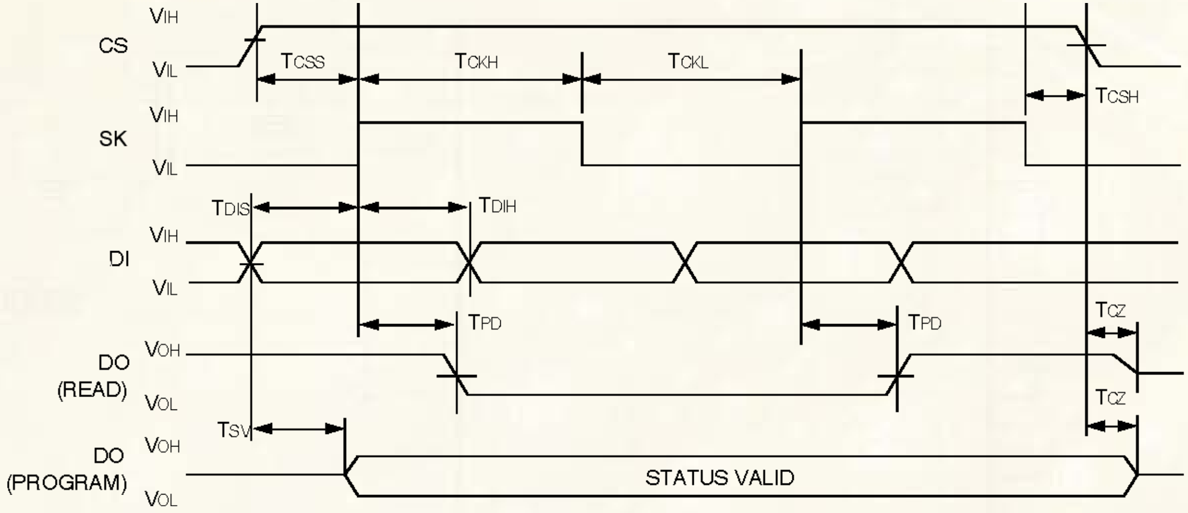
Example of a connection among COP8 microcontroller, periferials and slave microcontroller:

(*source: National Semiconductor, COP8CBR9/COP8CCR9/COP8CDR9 datasheet*)



MICROWIRE BUS

Bus timing

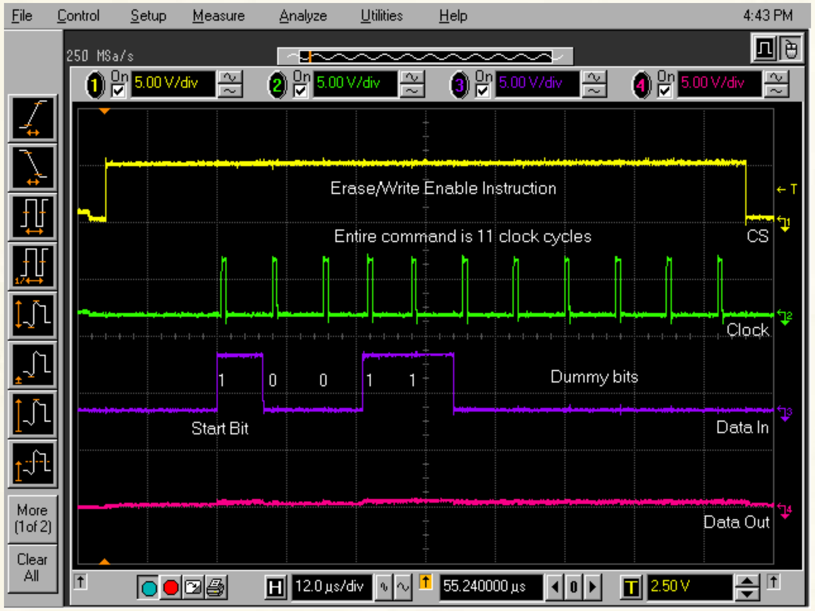


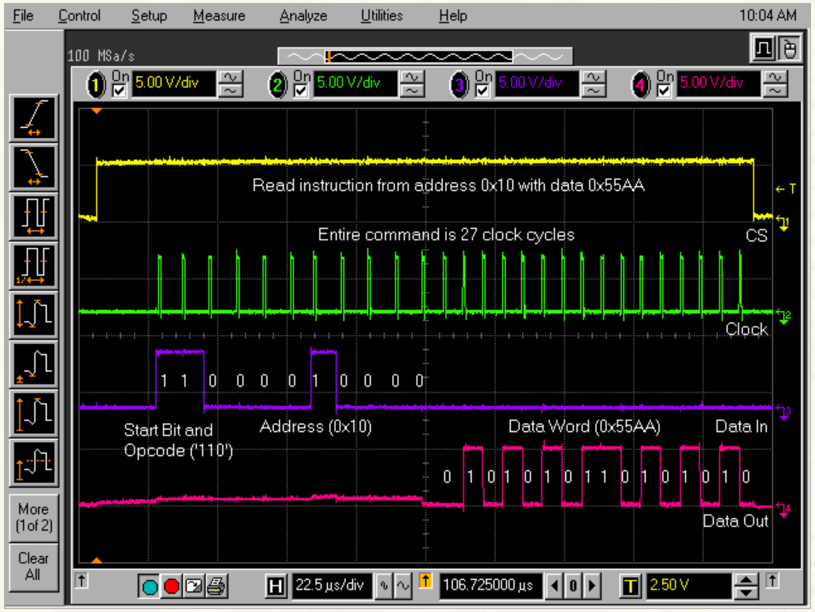
Symbol	Parameter	Min / Max	Description
T_{CSS}	Chip Select Setup Time	50 ns	Minimum time between CS activation and rising edge of first clock pulse
T_{CKH}	Clock High Time	250 ns	Minimum width of clock pulse at logic 1
T_{CKL}	Clock Low Time	250 ns	Minimum width of clock pulse at logic 0
T_{CSH}	Chip Select Hold Time	0	Minimum time, for which the CS signal must be held after falling edge of last clock pulse
T_{DIS}	Data Input Setup time	100 ns	Minimum time for which the data must be ready before the rising edge of clock pulse (to latch data)
T_{DIH}	Data Input Hold time	100 ns	Minimum time for which valid data must held after rising edge of clock signal
T_{PD}	Data output delay time	400 ns	Maximum time when valid output data must be ready after rising edge of clock signal
T_{CZ}	Data output disable time	100 ns	Maximum time after which the output must go in the high impedance state (after falling edge of CS signal)
T_{SV}	Status valid time	500 ns	Maximum time between CS go high and DO transition from high impedance state

These values are valid for EEPROM Microchip 93LCS56/66; maximum clock frequency of this chip is1 MHz,or 2 MHz according to supplying voltage

MICROWIRE BUS

An example of a variable number of bits in a single transaction on the bus:
In the figure on the left WRITE Enable command is send into EEPROM AN993 – start bit, another 4 bits of a command and 6 dummy bits, which completes transmission to total required 11 bits , in the figure on the right is the read operation – start bit, 2 bits of the command, address (11 bits in total), followed by16 data bits

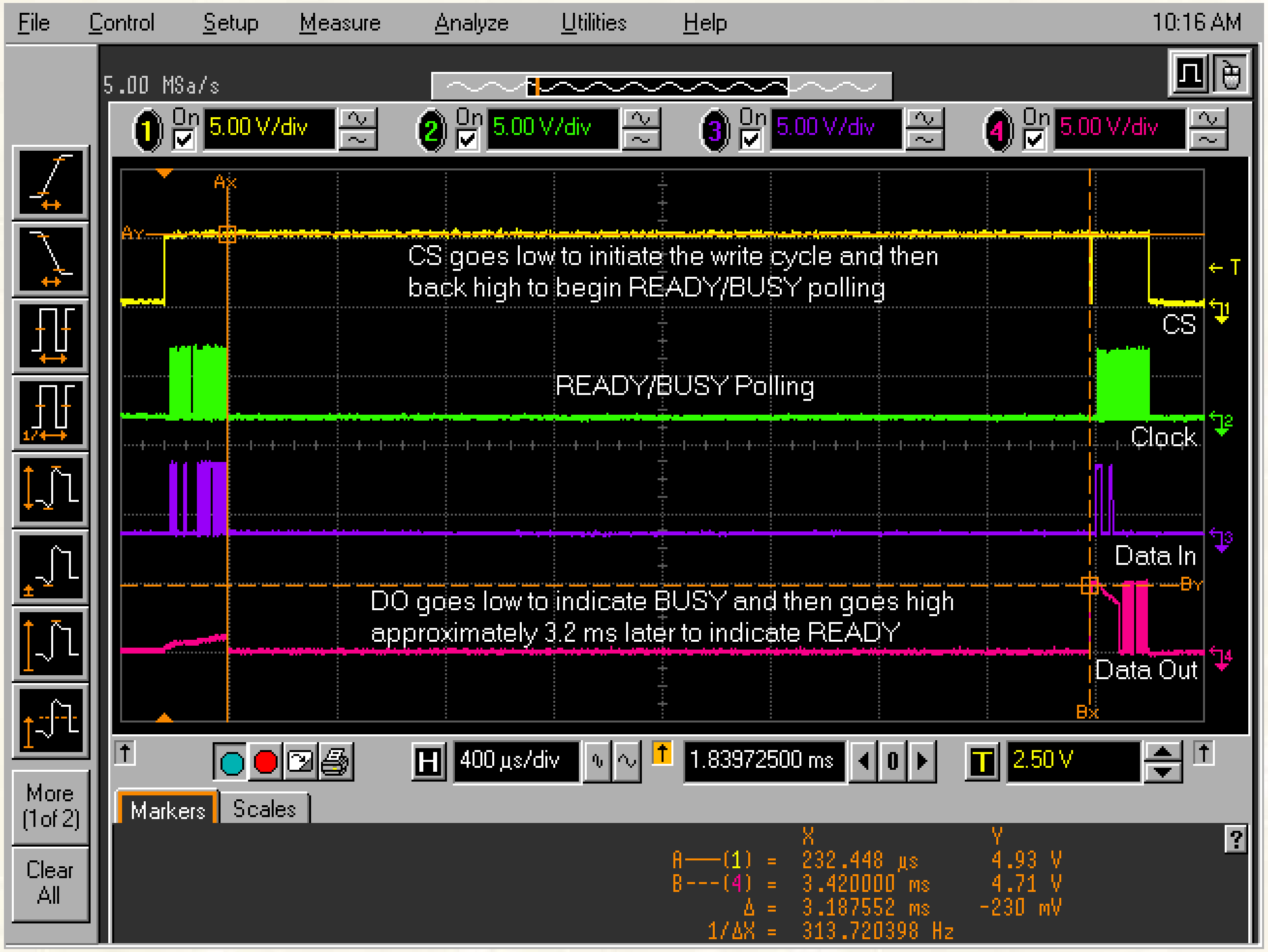

File Control Setup Measure Analyze Utilities Help 4:43 PM
250 MSa/s
5.00 V/div
Erase/Write Enable Instruction
Entire command is 11 clock cycles
CS
Clock
1 0 0 1 1
Dummy bits
Start Bit
Data In
Data Out
More (1of 2)
Clear All
12.0 µs/div
55.240000 µs
2.50 V



EEPROM Microchip AN993 – *from application note 00993a*

MICROWIRE INTERFACE

READY / BUSY polling: write operation into memory is finished approx. 3.2 ms after command is send; logic low of DO signal indicates, the write operation still proceeds, transition of DO signal in logic high value indicates the write operation is finished



5.00 MSa/s

1: 5.00 V/div 2: 5.00 V/div 3: 5.00 V/div 4: 5.00 V/div

CS goes low to initiate the write cycle and then back high to begin READY/BUSY polling

READY/BUSY Polling

DO goes low to indicate BUSY and then goes high approximately 3.2 ms later to indicate READY

H 400 µs/div 1.83972500 ms T 2.50 V

```
                  X                 Y
A——(1) =   232.448 µs         4.93 V
B---(4) =  3.420000 ms        4.71 V
      Δ =  3.187552 ms       -230 mV
   1/ΔX =  313.720398 Hz
```

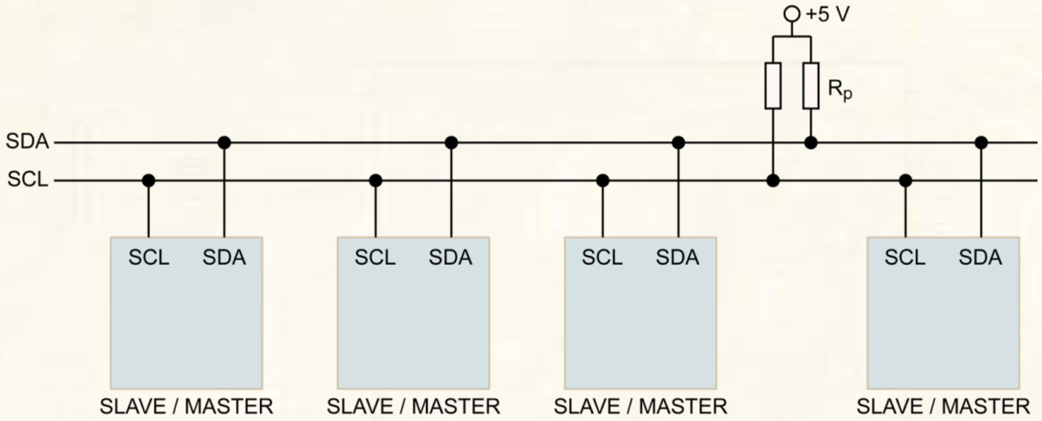
EEPROM Microchip AN993 – *from application note 00993a*

I2C

- Inter IC Bus – synchronous bidirectional 2-wire serial bus, developed for efficient inter-IC control and communication
  - TV chips: microcontroller PCB83C528, PLL syntetizer TSA5512, EEPROM PCF8582E, PAL/NTSC/SECAM decoder/sync processor TDA9160A, TDA4685 video processor, TDA9840 TV and VTR stereo/dual sound processor, …
  - Telephones and modems: PCD3311 DTMF/modem/musical-tone generators, PCA1070 Multistandard programmable analog CMOS transmission IC (telephones), …
  - Port expanders: PCF8574, MCP 23016, …
  - …
- Developed by Philips
- 2 + 1 wires: clock (SCL – Serial Clock), data (SDA – Serial Data) + ground; bus drivers with open collector → pull up resistors required
- Data rates: *it is the slowest bus from the SPI / Microwire / I2C family*
  - Standard Mode (S): 0 – 100 kb/s
  - Fast Mode (F): 0 – 400 kb/s
  - High Speed (Hs): 0 – 3.4 Mb/s (specification 2.0, 1998; 2.1, 2000)

  need not to be constant, low speed slave device can slow down communication holding clock signal at logic 0 as long as necessary
- In contrast to SPI / Microwire bus I2C is true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfers; any device may act like master device any time; software implementation is more complicated and machine time consuming than that of SPI / Microwire
- The devices have 7(+1) or 10 bits address; in the case of 7 bits the first 4 bits from MSB are fixed, its allocation is coordinated by the I2C- bus committee, and represents the kind of device (e.g. 1010 means EEPROM), next 3 bits represents device number on the bus – each device has 3 pins, which are set to logic 1 / 0; 8th LSB is R/W bit
- Signal levels: 5 V, 3.3 V, at Hs 2 V or lower, depending on configuration; maximum capacitance of the bus is 400 pF
  - Devices of any IC fabrication process (CMOS, BiCMOS, NMOS, bipolar or other) may be connected to the same bus, or even devices with different supply voltage
    - Pull-up voltage must be in such case 5 V ± 10 %

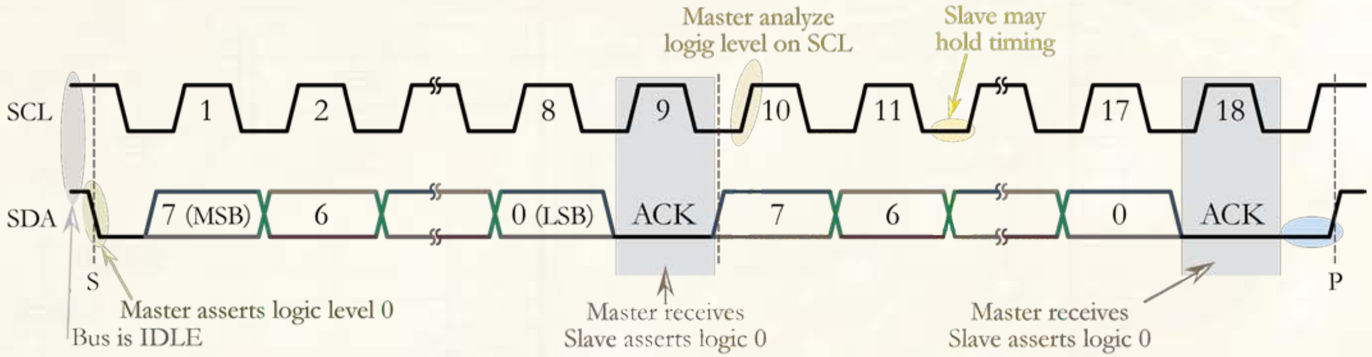
I2C



- The resistivity of pull-up resistors is not explicitly specified; their value is based on device current 3 mA – R_p is the function of supply voltage, bus capacitance and of a number of connected devices; typically ranges from 1.8 kΩ to more than 10 kΩ
- Between bus wire and device protective resistor of resistivity roughly hundreds of ohms is sometimes connected; *the graphs for both resistors you can find at I2C specification available on* http://www.i2c-bus.org/fileadmin/ftp/i2c_bus_specification_1995.pdf
- In IDLE both signals are in logic 1 (asserted by pull-up resistors)
- Logic value on SDA may be changed only if SCL is in logic 0; the only exception is initiation of communication (START condition), and termination of communication (STOP condition); transmission is always initiated by master

I2C

Transfer protocol



Master analyze logig level on SCL
Slave may hold timing
SCL
1
2
8
9
10
11
17
18
SDA
7 (MSB)
6
0 (LSB)
ACK
7
6
0
ACK
S
P
Master asserts logic level 0
Bus is IDLE
Master receives Slave asserts logic 0
Master receives Slave asserts logic 0

- Initiation of transmission: Start condition (S), asserted by master device, if the bus is IDLE
- Termination of transmission: Stop condition (P), asserted by master device
  - Bus arbitration – collision on bus:
    - Two or more devices detects the bus is IDLE, and try to initiate communication
    - Each device must scan the actual logic value on the bus; if the device transmits logic 1 and at the same moment detects logic 0 on SDA signal wire (*the bus drivers have open-collectors*), then such device lose the arbitration and must stop transmission
    - If the device which lose the transmission lose arbitration can act as slave, it must switch in the slave mode (*it may be addressed*)
- Repeated start (Sr): next transaction on the bus is initiated without preceding Stop condition
- Transmission of each byte is acknowledged by receiving device (Ack)
  - ACK = 0 is successful transmission = acknowledgement
  - ACK = 1 means, the target devices is not present or is busy ⇒ the Stop condition must be asserted
  - When Master receives data, after each byte ACK = 0 must be asserted, with exception of the last one, when ACK = 1; **the Stop condition is asserted always by Master**
- The clock signal has not constant period, but is hold down in logic 0 by the slowest device as long as necessary

I2C

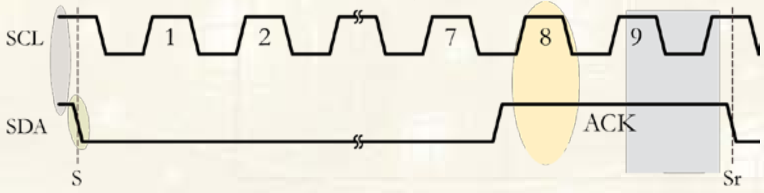
Addressing

- First byte after S, Sr conditions is an address of a slave device (in 7 bit addressing)

Reserved addresses on I2C

Bits[7:1]	Bit 0 (RW)	Description
0000 000	0	General call address; 2nd byte: LSB = „B“ B = 0: 00000110 RESET (RESET and take in the programmable part of device address) 00000100 no RESET (take in the programmable part of device address without RESET) B = 1: hardware general call (The seven bits remaining in the second byte contain the address of the master, data bytes follows /to every listening slave/)
0000 000	1	Start byte: attended to devices which implements I2C protocol by software (it is much slower than hardware implementation so such microcontroller for performance reasons scans the bus in IDLE state on a low sampling rate, so it needs long time to detect transmission and switch to a higher sampling rate of bus levels
0000 001	X	CBUS compatibility (*I2C device must not respond*)
0000 010	X	Address reserved for different bus format (*I2C device must not respond*)
0000 011	X	*Reserved for future use*
0000 1XX	X	8-bit Hs master code (initiates high-speed communication on bus with Hs devices only – mixed configuration possible, but only at L/F speed, or with bus segments bridge)
1111 1XX	X	*Reserved for future use*
1111 0XX	X	10 bits addressing (XX are two MSB of the address, second 8 bits of address follows)

Start byte



I2C

- In 7 bit addressing can be 8 devices of the same type on the same bus (we have just 3 bits of a device number available)
- 112 devices is maximum theoretical number of devices which may be on the same bus (16 groups identified by first 4 bits allocated by the I2C- bus committee of 8 devices each: 16 * 8 = 128 – 16 reserved addresses)

10 bits addressing

- 1024 addresses available
- Transmission master → slave:
  - 1st byte: `1111 0XX0` XX are 2 MSB of the address, RW bit = 0
  - 2nd byte: remaining 8 bits of the address
  - Data bytes
- Transmission slave → master (*address asserted by master*):
  - 1st byte: `1111 0XX0` XX are 2 MSB of the address, RW bit = 0 (*!!! – or slave can not receive 2nd byte of the address*)
  - 2nd byte: remaining 8 bits of the address
  - Sr condition
  - `1111 0XX1 -` XX are 2 MSB of the address, RW bit = 1 (!!!), no second byte of a slave's address
  - Slave transmits the data

Hs mode

- F/S devices must be separated from Hs devices by a bridge (or filter), which doesn't pass Hs communitation
- Doesn't support F/S arbitration
- Is not synchronized bit by bit → serial clock signal has a fixed HIGH to LOW ratio of 1 to 2
- Each Hs device operates normally as full speed, control byte `0000 1XXX` switch to Hs mode, after P condition the devices are switched back to full speed mode

1-WIRE

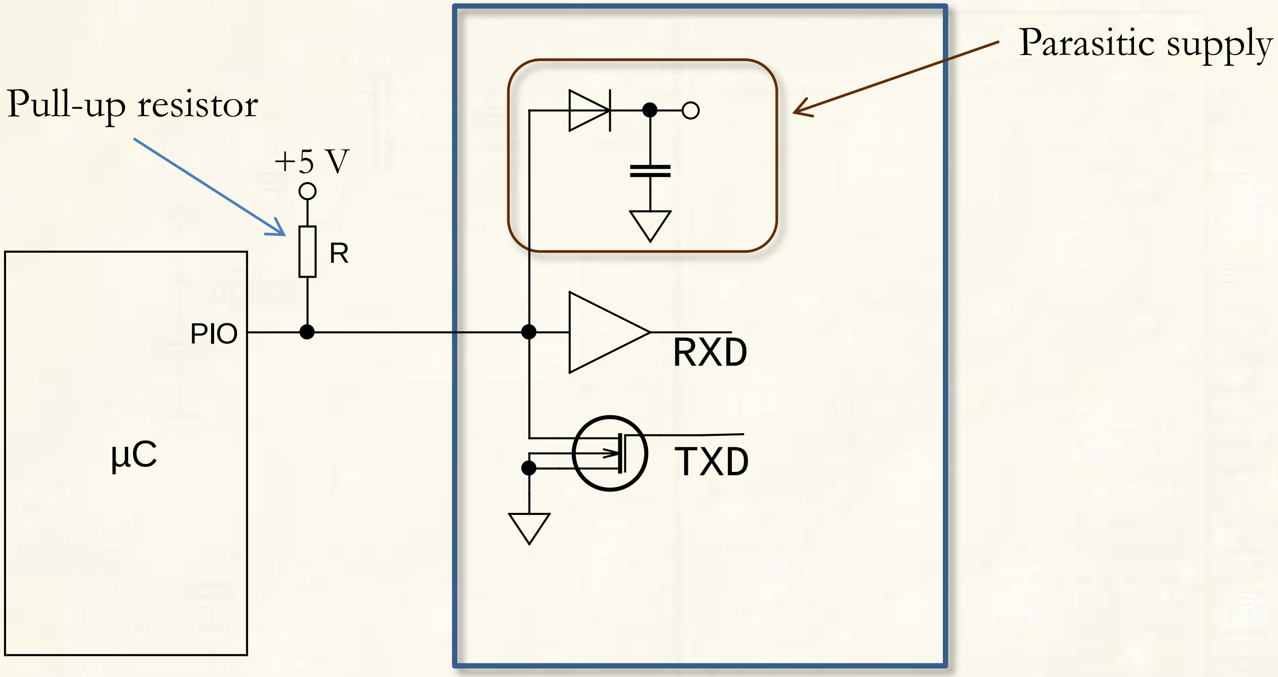
- Just 2 wires (combined data / supply) and ground
- Trademark of Maxim company (formerly Dallas Semiconductor Corp., bought in 2001)
- (Slow) serial bus, intended for connection of special HW devices to a microcontroller or computer
- Time-coded communication – no clock signal, correct timing of logic levels at the bus is essential
- Connected (slave) devices can be supplied directly from data wire
- Each chip has its unique 64-bit identification number (Serial Code), which may be used for identification purposes

Applications:

- ☞ Digital thermometers, temperature monitoring (e.g. Dallas DS18B20)
- ☞ Identification and authentication systems
  - Identification and calibration of a medical sensors (e.g. test strips in glucometers)
  - Medical Reagent Bottle ID
  - Printer cartridge identification and usage monitoring
  - AC-Adapter ID and Authentication
  - Customer payment systems (smart tickets in metro etc)
  - Personal identification (PC authentication, access control)
- ☞ Electronic potentiometer (DS2890)
- ☞ iButton (special packaging of 1-wire devices)

1-WIRE

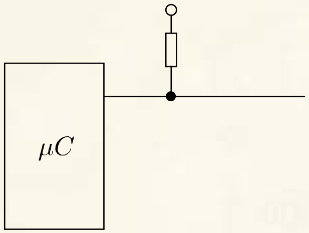
Basic connection



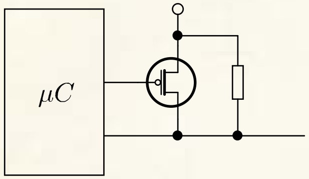
- Microcontroller is the **only and single master** of the serial bus
- **Hundreds of devices operating in slave mode** can be connected to the bus
- 1 wire bus is actually implemented by 2 wires – by data / supply wire and a ground wire, at greater distances by UTP 5 cable
- Bus drivers of master and slave devices are with open collector, the bus has single pull-up resistor near the master device; its resistivity is usually 4.7 kΩ, may be 2.2 kΩ
- Slave devices can be supplied from independent sources if their power consumption is too high, or directly from data wire; in such case pull up resistor acts as power supply, when the bus is in logical 0, the energy is drawn from capacitor (800 pF), integrated in each slave device

1-WIRE

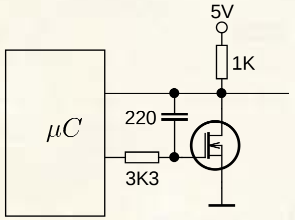
Pull-up resistor and its alternatives



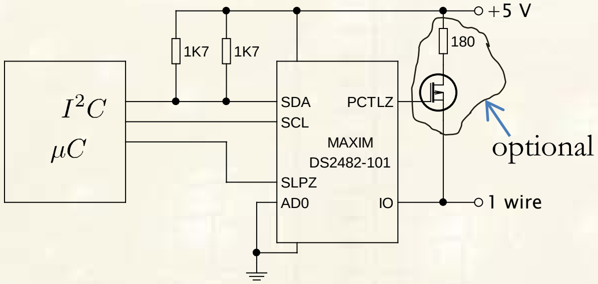
Specification requires voltage 2.8 – 6 V; if the voltage is 5 V, the current drain of all slave devices connected to the bus must not exceed 1 mA; but e.g. peek current drain of digital thermometer DS18B20 is 1.5 mA ⇒ **resistor 2K2 is possible use only if the number of devices is small, and the distance is less than 3 m**



This solution (according to the datasheet of DS18B20) guarantee sufficient power delivery during temperature conversion, but software must guarantee the transistor will not be open for more than 10 μs, only after the temperature conversion or copy command is send ⇒ **this solution is specific for given application**



This solution **sets the slew rate**, can be used **up to 200 m**, but **does not deal with higher current drain** of slave devices; also the bus, as well as in the solutions above, **is not impedance matched**

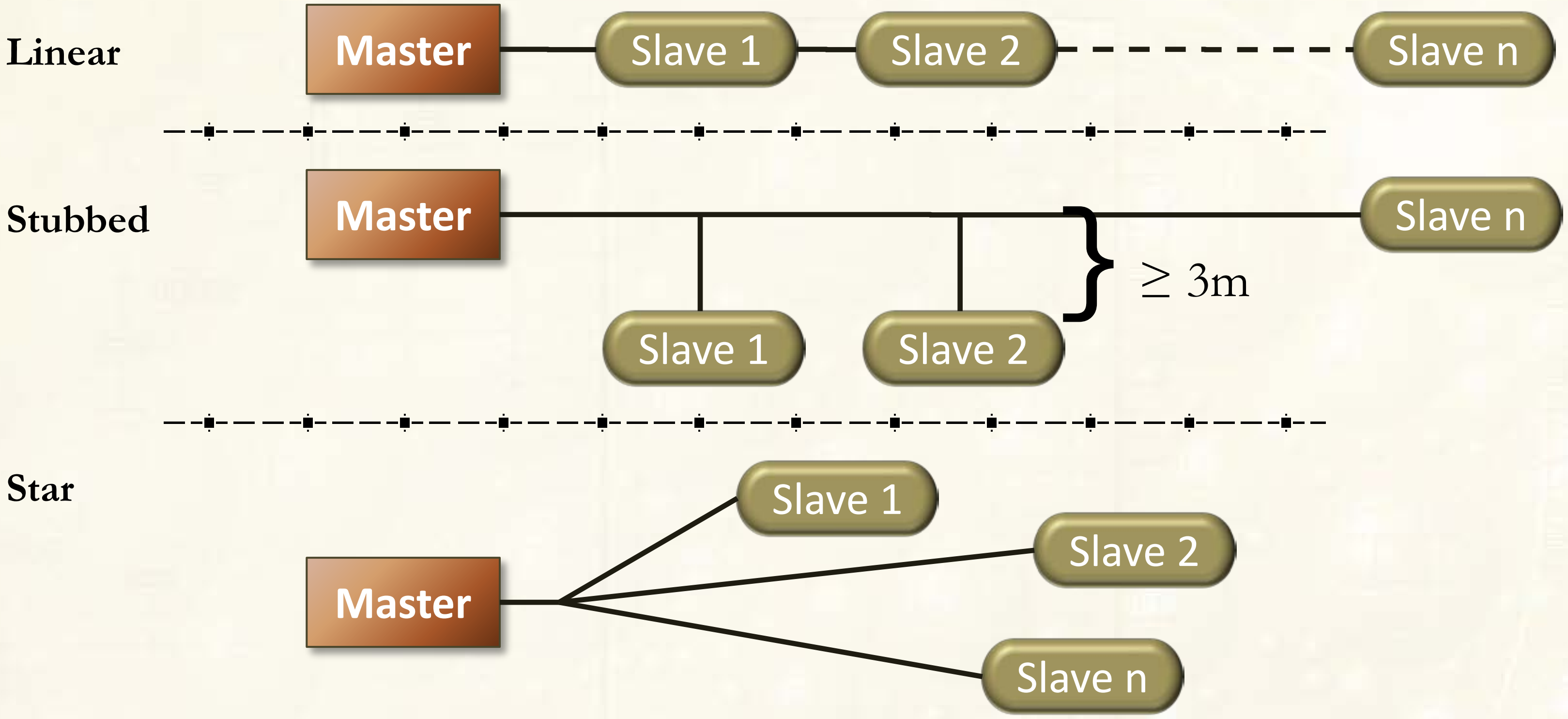


Bus drivers:

- **DS2482-101** – I2C / 1Wire bridge, solves a higher capacitance load (longer wires, higher number of slave devices) – Active Pullup (APU) mode, connection of devices with higher current demand is possible (digital thermometer e.g.) – Strong Pullup (SPU) mode, impedance matching
- **DS2480B** – RS232 / 1Wire bridge
- **DS9490** – USB / 1Wire bridge

1-WIRE

1-Wire Network Topologies



It is not recommended unless individual branches switched – *reflections especially on large distances (and impedance unmatched connections of distinct branches) are causing the greatest problems*

Radius: the wire run distance from the master end to the furthest slave device
- establishes the timing of the slowest signal reflections
- no 1-Wire network may ever have a radius greater than 750m. At this distance, the protocol will fail due to the time delay of the cable

Weight: the total amount of connected wire in the network, plus an equivalent weight of connected slave devices (0.5m general devices, 1m iButtons)
- limits the rise time on the cable – transient (charging of a capacitance – typically 24 pF/m)

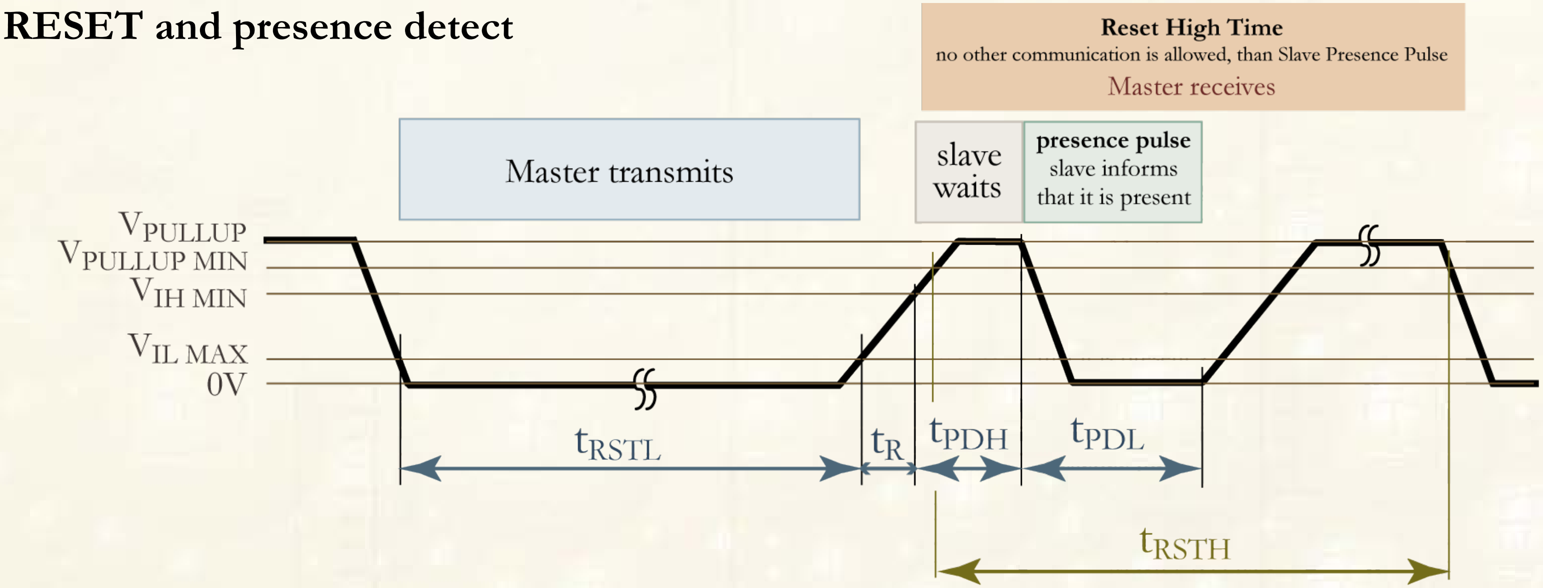
1-WIRE

Logic values, timing, read and write data time slots

- **Logic values**

$V_{IL\ MAX}$	slave's maximum-input low voltage
$V_{IH\ MIN}$	minimum slave-input high voltage
$V_{PULLUP\ MIN}$	minimum permissible pullup voltage, which is required for the parasite power supply to function properly is given by a minimum slave's operating voltage plus voltage drop across diode
V_{PULLUP}	1-Wire pullup voltage

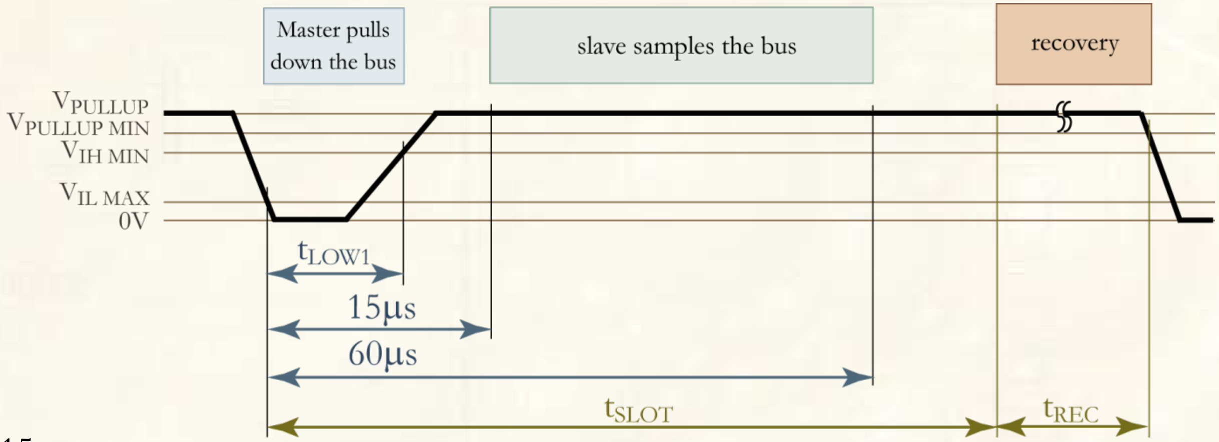
- **RESET and presence detect**



t_{RSTL}	480 µs min. – Reset Low Time, 960 µs max
t_{RSTH}	480 µs min. – Reset High Time
t_{PDH}	30 µs typically, 15 – 60 µs min / max
t_{PDL}	120 µs typically, 60 – 240 µs min / max

1-WIRE

- **Write time slot** (master → slave)
  - **Logic 1**



1 µs ≤ t_{RSTL} ≤ 15 µs

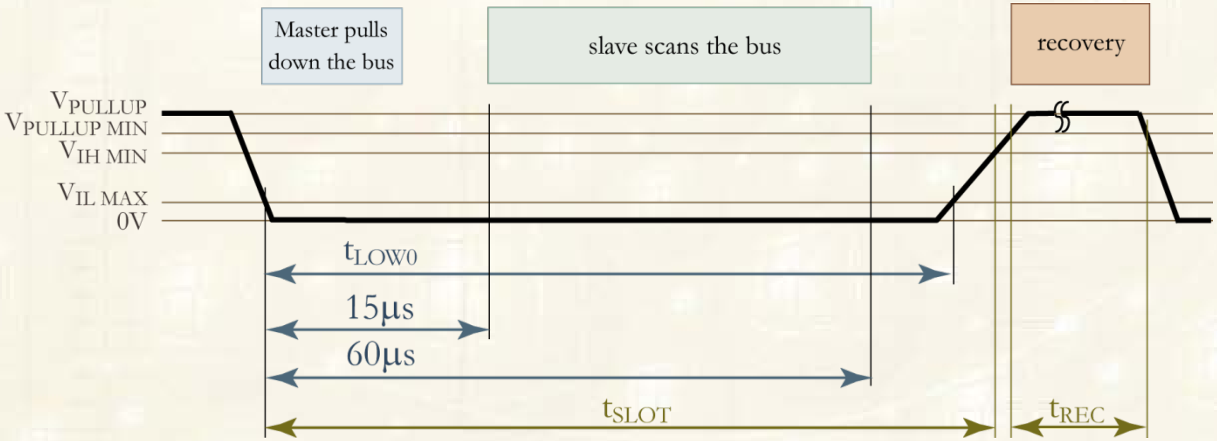
60 µs ≤ t_{SLOT} ≤ 120 µs (¼ of the RESET pulse), typically 60 µs

time slot – standard communication unit (1 bit) – RESET pulse is 8 + 8 time slots

t_{REC} ≥ 1 µs recovery time (legacy – extra time to one bit time slot, new style – included in 1 bit time slot)

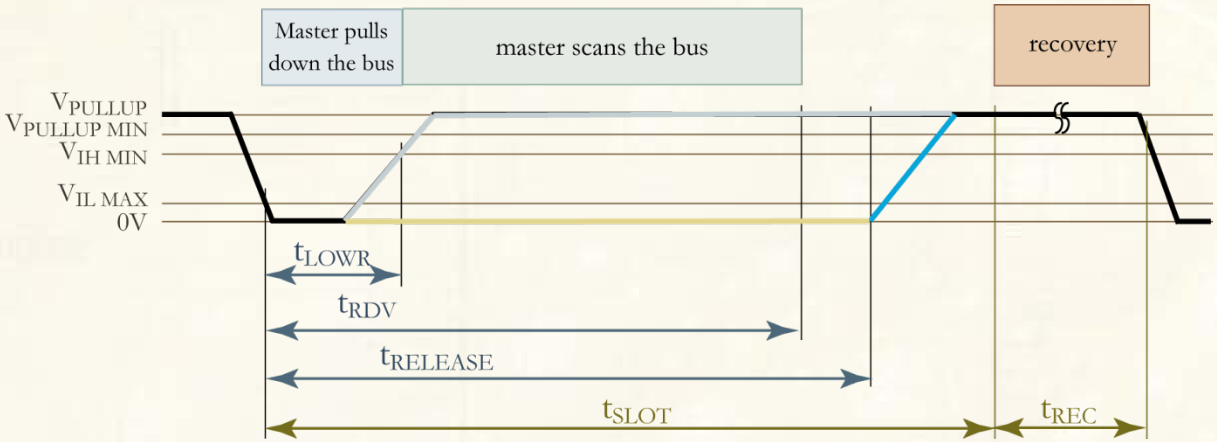
Reading of the bus: allowed time window is 15 – 60 µs, 30 µs is recommended

  - **Logic 0**



1-WIRE

- **Read time slot** (slave → master)



1 μs ≤ t_{LOWR} ≤ 15 μs
60 μs ≤ t_{SLOT} ≤ 120 μs (¼ of the RESET pulse), typically 60 μs
t_{REC} ≥ 1 μs recovery time
t_{RDV} = 15 μs

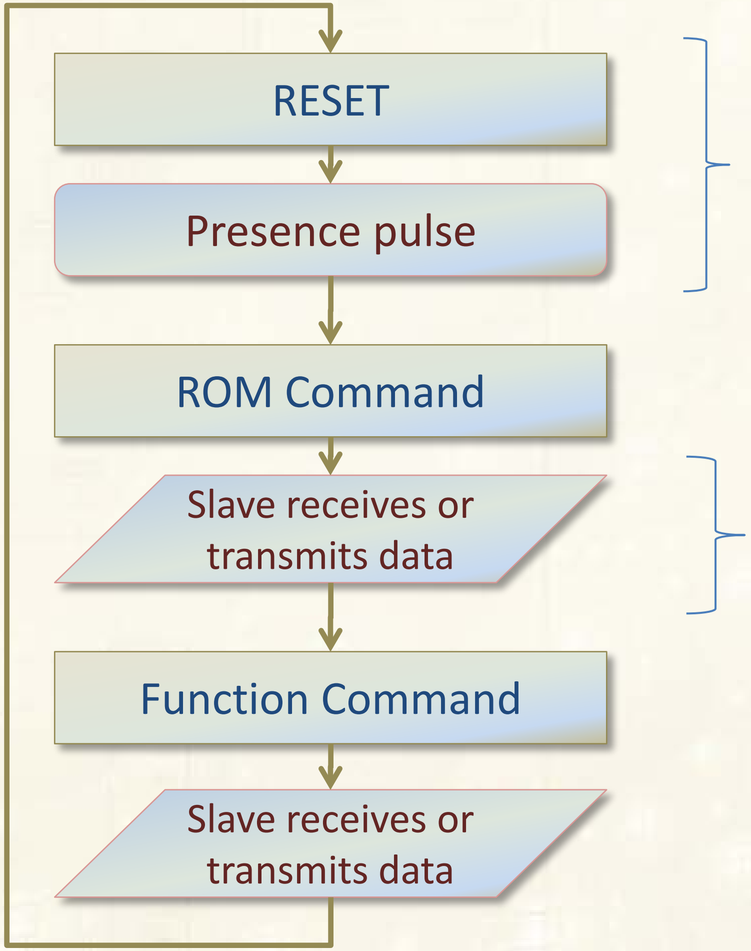
Read 1: read time slot is initiated by the master device pulling the 1-Wire bus low for a t_{LOWR}, and then releasing the bus, slave transmits a 1 by leaving the bus high

Read 0: read time slot is initiated by the master device pulling the 1-Wire bus low for a t_{LOWR}, and then releasing the bus, slave pulls the bus low for a t_{RDV}

It is recommended, the t_{LOWR} should be as short as possible and master sample time during read time slots should be towards the end of the 15μs period (bus capacitance!!!)

1-WIRE

General communication sequence


RESET
Presence pulse
ROM Command
Slave receives or transmits data
Function Command
Slave receives or transmits data

Initiates each communication on the bus

Necessary only when more slave devices are present on the same bus (addressing, discovering of registration numbers, …)

1-WIRE

Data rate

- Maximum theoretical: $\frac{1}{61\,\mu s} = 16.39\,\mathrm{kb\,s}^{-1}$ $\frac{1}{7\,\mu s} = 142.85\,\mathrm{kb\,s}^{-1}$ in the overdrive mode

  (necessary RESET, nor the device address /optional in the case of a single slave device/ is not considered)

- Actual speed:
  - More slave devices on the same bus:

RESET + Presence pulse: 16 time slots	960 µs
ROM address: 8 + 64 time slots	72 * 61 µs = 4.392 ms
Command: 8 time slots	488 µs
Data (plus CRC)	8 * 61 = 488 µs / byte
	6.328 ms ~ 1.26 kb s^{-1} (1 byte transmission)

  - Single slave device on the bus:

RESET + Presence pulse: 16 time slots	960 µs
Skip ROM: 8 time slots	8 * 61 µs = 488 µs
Command: 8 time slots	488 µs
Data (plus CRC)	8 * 61 = 488 µs / byte
	2.424 ms ~ 3.3 kb s^{-1} (1 byte transmission)

CAN BUS

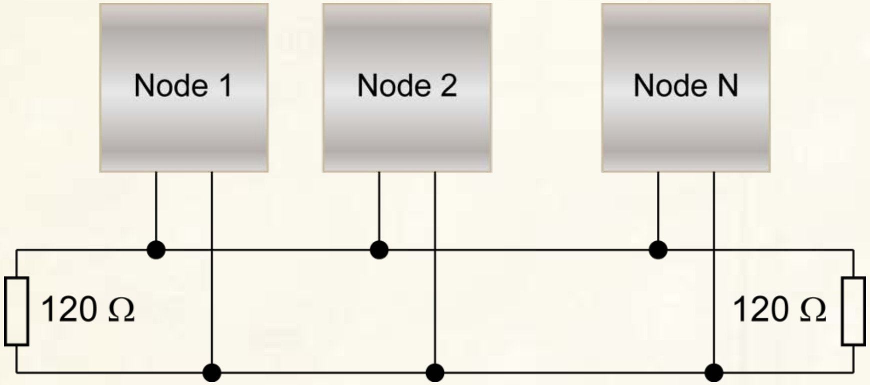
- **C**ontroller **A**rea **N**etwork is a data bus, used mainly for communication between functional units in the car (including diagnostics)
- Can bus was developed in 1983 by Robert Bosch GmbH; the first car equipped with CAN BUS system was in 1986 the BMW 850 coupe (it reportedly saved about 2 km of cables); this bus soon penetrated into the automation technology;
  in 1992 was founded an association of users and manufacturers “CAN in Automation” (CiA); today is used in industry for connecting a variety of intelligent sensors and actuators, there are a number of expansion cards for connecting of control PC
- A number of microcontrollers presently have integrated hardware CAN-Controller, both that produced by manufacturers targeted on industry and automotive applications (Infineon, 8, 16, and 32 bits microcontrollers), and that produced by manufacturers of general-purpose microcontrollers, e.g. Stellaris (core ARM Cortex-M3) from the company Luminary Micro (Texas Instruments), and even including 80C51 architecture (Atmel T89C51CC01UA).
- In automotive technology, however, CAN bus is not the only bus. There is also used a simple and inexpensive LIN bus, multimedia MOST bus, and in some luxury cars (Audi A8, BMW 5 and 7, ...), the speed of CAN bus is no longer sufficient, and therefore is replaced by a new FlexRay bus.

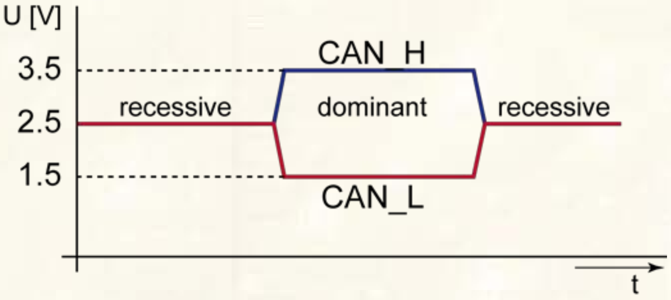
- CAN bus is a multimaster bus, or the nodes of the system are equivalent - Peer to peer
- frames (or messages) are transmitted and received by all nodes as needed
- There are no addresses of transmitting or receiving node – the frame always includes an identifier which must uniquely define its content; each device performs filtration based on this identifier, and according to the result of filtration the message is further processed or discarded. Each of the receivers so accepts only those data frames that are important for it.
- if the frame is successfully received by at least one node, then the receipt is confirmed
- if an error is detected during reception (by any node including the transmitting one), the error sequence is sent and transmission must be repeated
- The maximum data rate on the bus is 1Mbit/sec.

CAN BUS

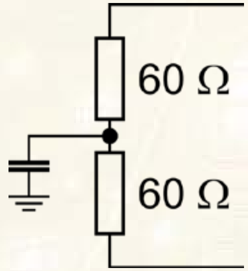
Physical layer

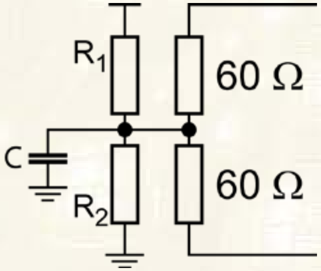
- Original Bosch specification does not contain any requirements for the transmission medium, or voltage levels
- Defined are only formal logical values – recessive (r) and dominant (d). If several devices are connected to the bus and at least one of them sends d value while others transmit r value, then the bus is in the d value. Only if all devices transmits r value, the r value is on the bus.
- Electrical parameters (but not connectors and cables) are specified just by the standard ISO-11898:





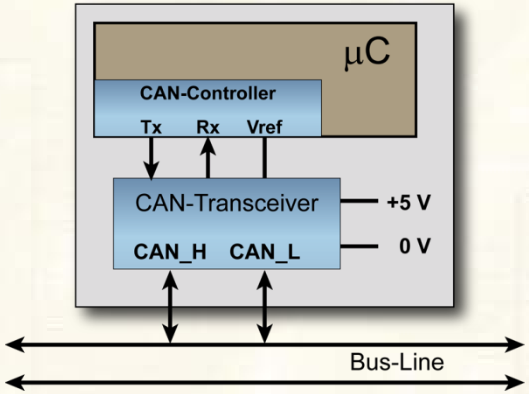
- To transfer data in accordance with this standard symmetrical lines with characteristic impedance of the cable 120 Ω is used. To avoid reflections termination resistors 120 Ω are connected at both ends to the lines. Besides the standard termination by one resistor, sometimes due to lower EMC emissions the split terminating is used, which can also be supplemented by voltage divider, which maintains a constant recessive level.
- „Soft“ sources in bus drivers in distinct nodes maintains recessive value at 2.5 V (the difference must be less than 0.5 V on the receiver's side and 1.5 V on the transmitter's side), any node by the „strong“ source in the bus driver set up dominant voltage level between 2.75 – 4.5 V to CAN_H, or 0.5 – 2.25 V to CAN_L (minimum difference is 0.9 V).





CAN BUS

- While the CAN controller is usually part of the microcontroller and provides functions of data link layer (error detection, data frames coding, the detection of identifiers and message filtering) and partly functions of the physical layer ("bit stuffing" - inserting of synchronization bits), bus drivers are implemented as a separate circuit, e.g . SN65HVD1050 (Texas Instruments), TLE6250 (Infineon), PCA82C250 (Philips), L9615 (SGS-Thomson) ...



- Maximum data rate is affected by the length of a line:

CIA DS-102

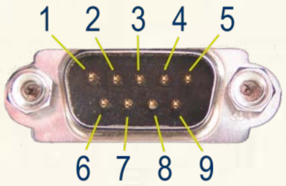
Length [m]	Line resistivity per 1 m [μΩ]	Data rate [kbit/s]
< 25	70	1000
< 50	60	800
< 100	60	500
< 250	60	250
< 500	40	125
< 1000	26	50
< 2500		20
< 5000		10

ISO-11898

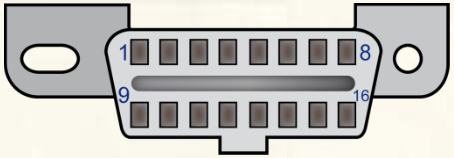
Length [m]	Line resistivity per 1 m [μΩ]	Data rate [kbit/s]
< 40	70	1000
< 300	60	500
< 600	40	100
< 1000	26	50

CAN BUS

Connectors



CAN bus CiA DS-102 connector – male



J1962 On-Board Diagnostic II (OBD II)
Diagnostic Link Connector (DLC) – female

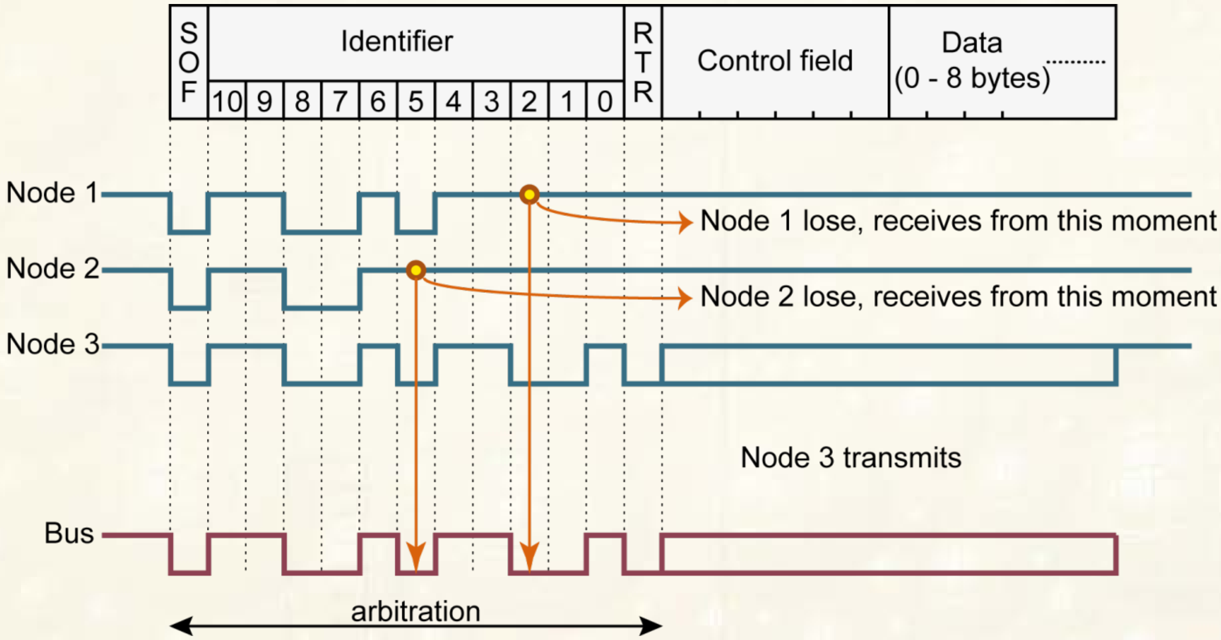
Pin	Signal	Description
1	-	*Reserved*
2	CAN_L	CAN_L bus line dominant low
3	CAN_GND	CAN Ground
4	-	*Reserved*
5	(CAN_SHLD)	*Optional: CAN shield*
6	GND	*Optional: ground*
7	CAN_H	CAN_L bus line dominant high
8	-	*Reserved*
9	(CAN_V+)	*Optional: CAN external supply*

Pin	Signal	Description
1		*GM: J2411 GMLAN/SWC/Single-Wire CAN*
2	J1850 Bus+	SAE J1850 PWM / SAE J1850 VPW protocol
3	DCL+ / CCD+	Ford / Chrysler
4	CGND	*Chassis ground*
5	SGND	*Signal ground*
6	CAN High	ISO 15765-4 a SAE-J2284
7	K-LINE	ISO 9141-2. LIN like protocol
8	-	
9	-	
10	J1850 Bus-	
11	DCL- / CCD-	Ford / Chrysler
12	LS CAN Bus	Renault: low speed CAN bus, (multimedia CAN bus)
13	LS CAN Bus	Renault: low speed CAN bus, (multimedia CAN bus)
14	CAN Low	
15	L-LINE	(ISO 9141-2 and ISO/DIS 14230-4)
16	+12v	Battery power

CAN BUS

Medium Access Control, Frame coding, Error detection

- The CAN protocol uses a modified version of the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA), called CSMA/CR (Carrier Sense Multiple Access with Collision Resolution):
  - any node can transmit if it has determined the bus to be free
  - The following arbitration is used: when node starts transmitting, it must simultaneously scan the actual logic value on the bus; when that node transmit bit in recessive logical value, and the actual state is dominant, it means, another node decide to transmit in the same instant; the node in recessive value lose the arbitration and must stop transmission and release the bus



- Bit stuffing: each node generates the internal clock signal that is synchronized with the transmitting node by data signal. Therefore, if the data contains 5 consecutive bits of the same logical value, one additional bit of the opposite logical value is inserted (just like the USB bus)
- 15 bit CRC and Acknowledge field as well as EOF delimiter terminates the transmission. When error is detected, error frame is send (6 same bits with violates bit stuffing rule).