Embedded Systems

Communication devices, RS232, SPI, I2C, CAN

Lesson 09

Francesco Menichelli

francesco.menichelli@uniroma1.it

Introduction

- Embedded & Real-time systems could be standalone or connected
- A real-time system is often composed from a number of *periodic* (time triggered) and *sporadic* (event triggered) tasks which communicate their result by passing messages.
- In a distributed real-time systems these messages are sometimes sent between processors across a communication device.

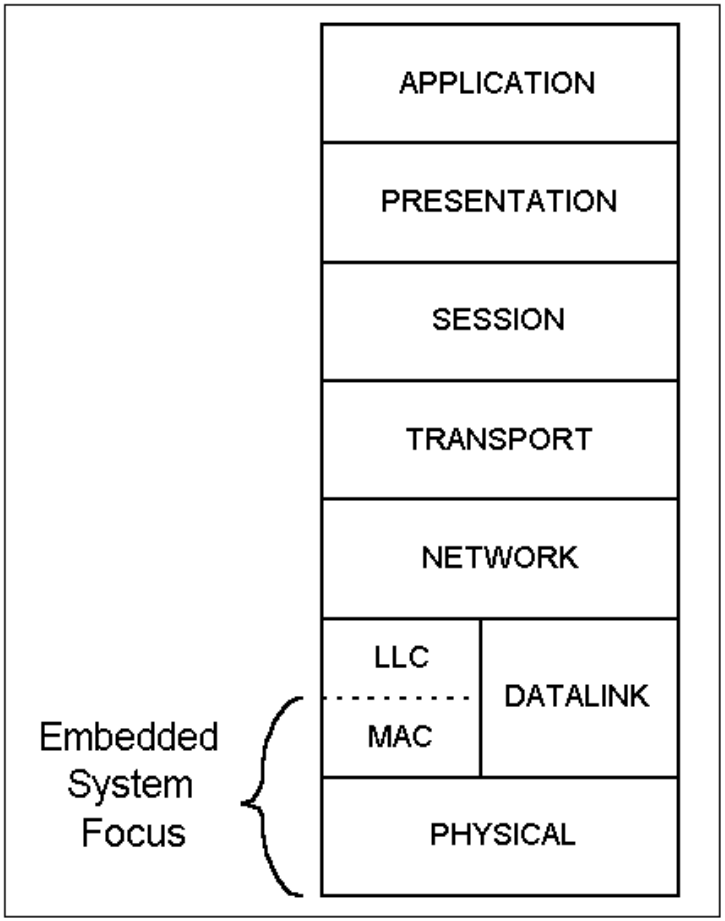
Introduction (cont.)

- To guarantee that the timing requirements of all tasks are met, the communications delay between a sending task and a receiving task being able to access that message must be bounded.

For examples

- Control systems: between sensors and actuators via central computer
- Multiprocessors: between processors, tasks communicating

Open System Interconnection



- Intended for computers
- Designed to solve compatibility problem
- Layers provide standard interface and services
- Embedded systems use some standardisation ideas
- Higher layers require lower layers to work

OSI

Layers

- Application – user interface e.g. Internet explorer
- Presentation – data formatting e.g. compression and encoding
- Session – handle overall connection e.g. OS, scheduling programs
- Transport – ensures data transfer, error checking e.g. TCP
- Network – logical addressing, routing e.g. IP (from TCP/IP)

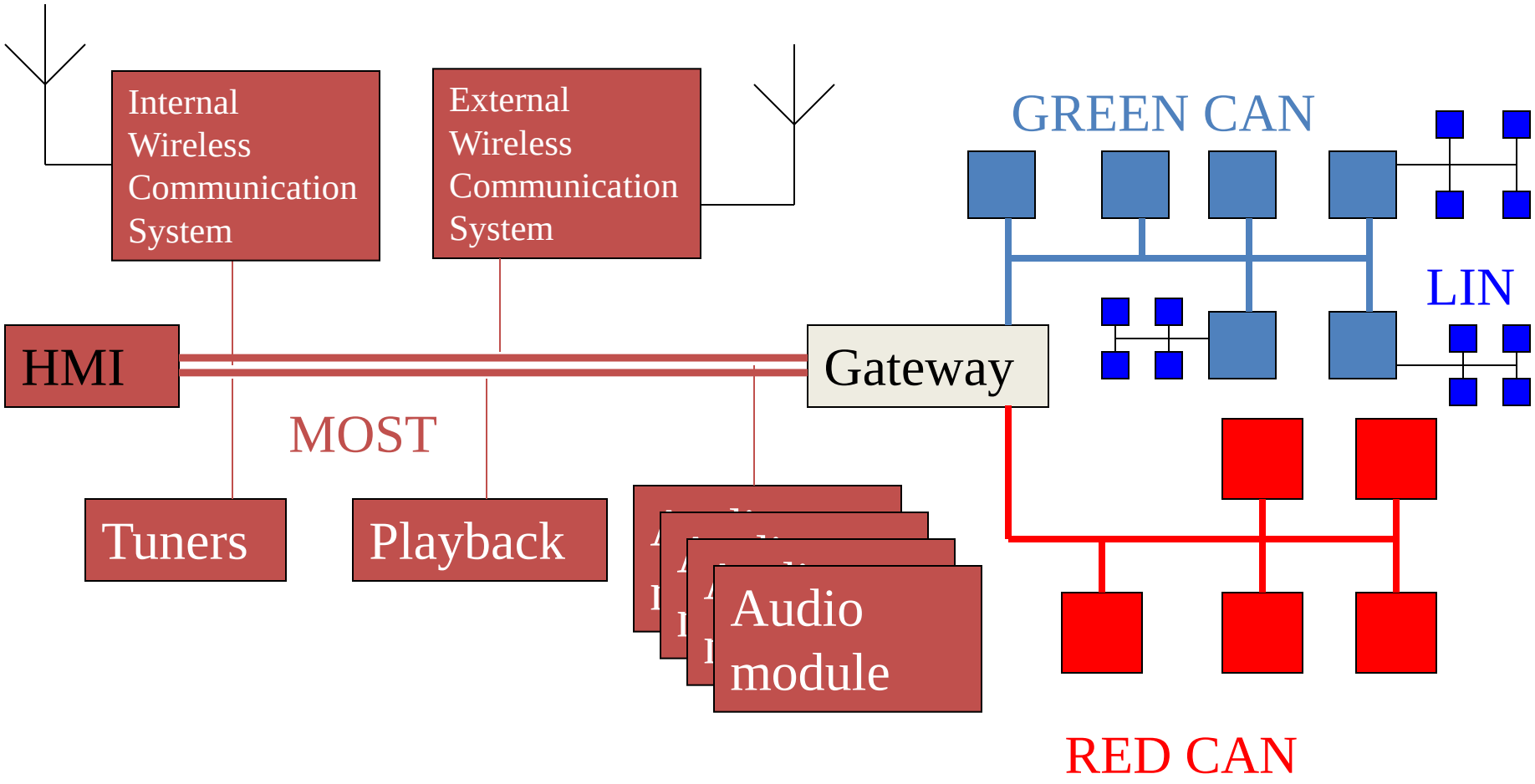
OSI cont.

- Data link – prepares data for transfer, physical addressing such as Media Access Control (MAC)
- Physical – wires and cables, hubs, repeaters.

Embedded Communication

- Point-to-point networks
  - Each node connected to every node
  - Simple and reliable
  - Dedicated links make it easy to meet real-time deadlines
  - Costly due to **many wires** required
- Shared media networks
  - Nodes are connected via bus or other topologies
  - Less wiring and hence cheaper
  - Easily extendable by adding new nodes to network
  - Complex network protocol

Complex distributed architecture



Concepts

- Event based communication
  - E.g. alarm, user inputs, requests for data from other systems
- State based communication
  - E.g. regular sensor readings
  - Predictability

Network resources & qualitative parameters

- Network resources
  - Bandwidth
  - Buffer space
  - Protocol efficiency (data bits/bandwidth). Depends on
    - Message overhead
    - Media access overhead
  - Determinacy (ability to calculate worst-case response time)
  - Robustness
  - Cost

Event based system

- Efficient use of network resource
- Needs high reliability (event based data comes once in a while)
- May need acknowledgement
- Hard to predict delay in case of overloading (e.g. alarm)

State based system

- Messages sent at predefined, regular intervals.
- Less efficient due to regular occupation of communication channel by nodes.
- More tolerance. Missed message may be ok, since the next one will be coming.
- Transient data problem. Sending node has to keep data long enough for other to see. E.g. button pressed may need to be repeated.

Protocol

- No best protocol, depends on applications.
- Embedded systems tends to focus on level 1 and 2 of OSI model, for simplicity and overhead reduction.
- Physical link (Layer 1) – transmission medium
- Data link (Layer 2) provides Media Access Control (MAC)

Advanced communication principles

- Layering
  - Break complexity of communication protocol into pieces easier to design and understand
  - Lower levels provide services to higher level
    - Lower level might work with bits while higher level might work with packets of data
  - Physical layer
    - Lowest level in hierarchy
    - Medium to carry data from one actor (device or node) to another
- **Parallel communication**
  - Physical layer capable of transporting multiple bits of data
- **Serial communication**
  - Physical layer transports one bit of data at a time
- **Wireless communication**
  - No physical connection needed for transport at physical layer

Parallel communication

- Multiple data, control, and possibly power wires
  - One bit per wire
- High data throughput with short distances
- Typically used when connecting devices on same IC or same circuit board
  - Bus must be kept short
    - long parallel wires result in high capacitance values which requires more time to charge/discharge
    - Data misalignment between wires increases as length increases
- Higher cost, bulky

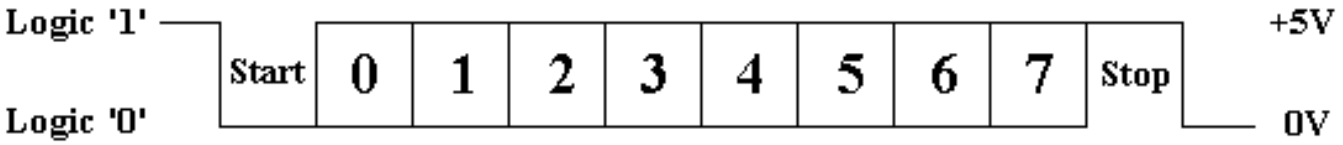
Serial communication

- Single data wire, possibly also control and power wires
- Words transmitted one bit at a time
- Higher data throughput with long distances
  - Less average capacitance, so more bits per unit of time
- Cheaper, less bulky
- More complex interfacing logic and communication protocol
  - Sender needs to decompose word into bits
  - Receiver needs to recompose bits into word
  - Control signals often sent on same wire as data increasing protocol complexity

Asynchronous serial communications

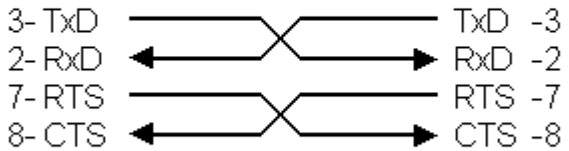
Asynchronous serial communications

- serial
  - A single data line is used for communication
- asynchronous
  - Tx device does not provide an external clock signal
  - The receiver does not know when a packet will be sent (no clock line)
  - Rx device must autonomously find the sampling time on data signal



Asynchronous serial communications

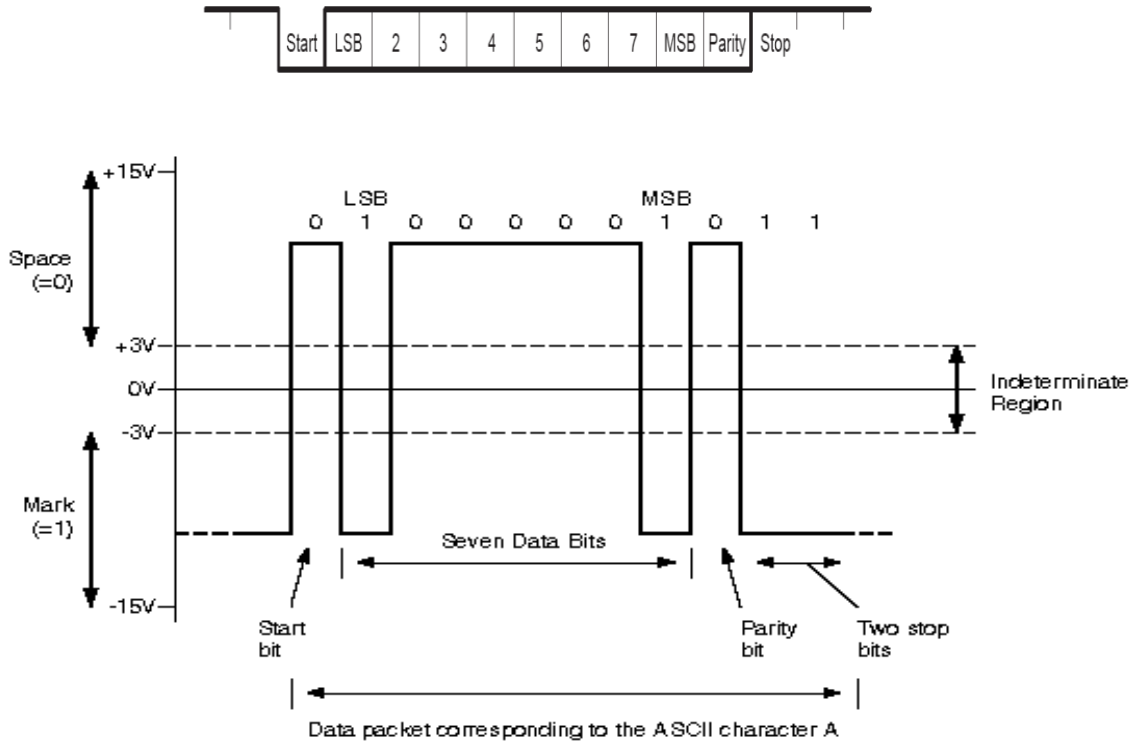
- Point to point link (no MAC)
- Single direction data lines
  - TxD – RxD
- Flow control as separate signals
  - RTS – CTS (RequestToSend – ClearToSend)



- Bit rate limited by lack of clock line
  - typically less that 1Mb/s
- Distance dependent from physical layer and bitrate
  - TTL/CMOS: tipycal distance 1 meter
  - RS232: typical distance 10 meter
  - RS485: typical distance 100 meter

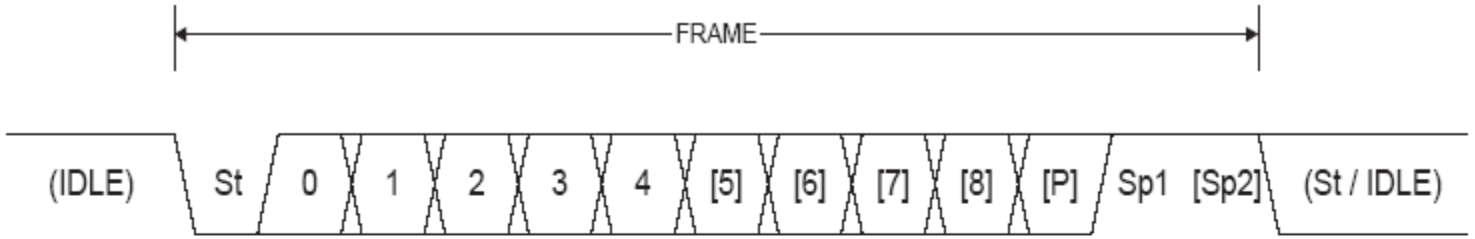
Physical layer

- Electrical levels for ‘0’ and ‘1’ simbols can vary
  - TTL/CMOS: ‘0’=0V ; ‘1’=Vdd (5V/3.3V)
  - RS232: ‘0’=+12V ; ‘1’=-12V
  - RS485: differential balanced line over twisted pair TxD+/TxD RxD+/RxD-



Data link layer

- Asynchronous serial communications data frame format
  - Bits are coded in NRZ format (non-return-to-zero)



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxD or TxD). An IDLE line must be high.

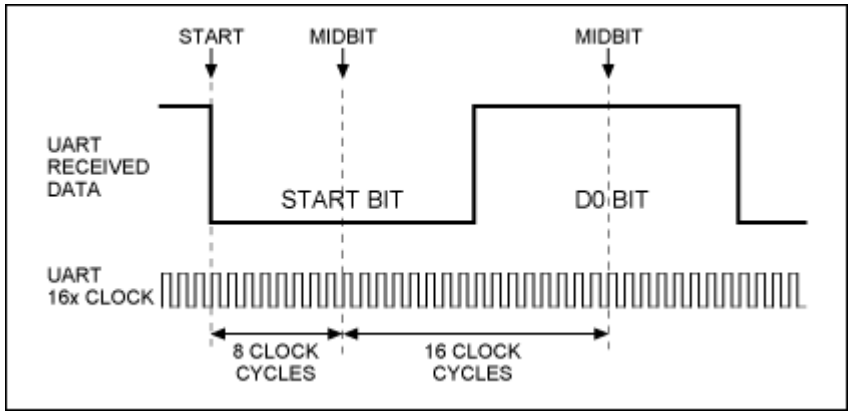
Data link layer

The definition on the frame format requires setting the following

- Bitrate
  - Communication speed
- Data bit number
  - Number of data bits per frame (typ. 5-9)
- Parity
  - Single error revelation
- Stop bit number
  - Number of stop bits (typ. 1-2)
- Flow control
  - If the receiver signals buffer full condition, the transmitter stop sending

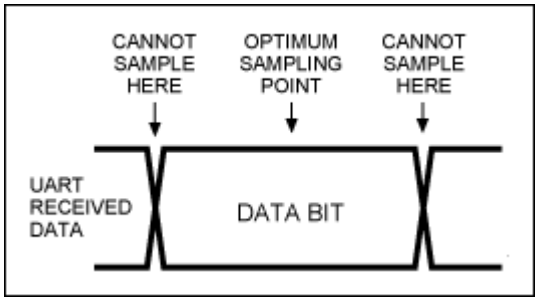
Clock recovery

- The receive UART uses a clock that is a multiple of the data rate (for example 16 times)
- A new frame is recognized by the falling edge at the beginning of the active-low START bit. This occurs when the signal changes from the active-high STOP bit or bus idle condition.
- The receive UART resets its counters on this falling edge, expects the mid-START bit to occur after 8 clock cycles, and anticipates the midpoint of each subsequent bit to appear every 16 clock cycles thereafter.
- The START bit is typically sampled at the middle of bit time to check that the level is still low and ensure that the detected falling edge was a START bit, not a noise spike.
- Another improvement is to sample the START bit three times (clock counts 7, 8, and 9, out of 16) instead of sampling it only at the midbit position (clock count 8 out of 16).

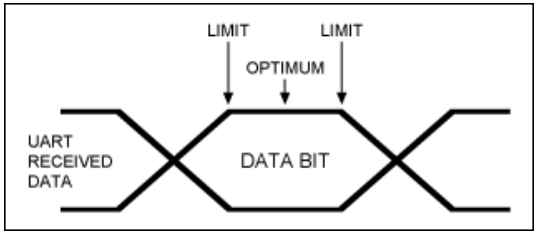


Timing Accuracy

- Since the UART receiver synchronizes itself to the start of each and every frame, we only care about accurate data sampling during one frame.
- There is no buildup of error beyond a frame's STOP bit
- The goal is to sample each bit at the midpoint, with maximum one-half a bit-period too early or too late



- In reality, we cannot sample close to the bit-transition point reliably.



Timing Accuracy

- Supposing 25% rise time + 25% falling time, and referring to the 16x UART receive clock, we have a maximum allowable error of 8 cycles, or ±4 cycles from the center of the bit
- Another error to include in this budget is the synchronization error when the falling edge of the START bit is detected (±1 cycle), hence ±3 cycles from the center of the bit is the maximum allowable error
- Since the timing is synchronized at the falling edge of the START bit, the worst-case timing error will be at the last data sampling point, which is the STOP bit1.
- The optimum sampling point for the STOP bit is its bit center, which is calculated as:

(16 internal clock cycles per bit) × (1 start bit + 8 data bits + ½ a stop bit) = (16) × (9.5) = 152 cycles

- The clock mismatch error can be ±3/152 = ±2%
- Although the problem will materialize at the receive end of the link, clock mismatch is actually a tolerance issue shared between the transmit and receive UARTs. So presuming that both UARTs are attempting to communicate at exactly the same bit rate (baud), the allowable error can be shared, in any proportion, between the two UARTs.

Clock recovering – bit sampling

Figure 83. Start Bit Sampling

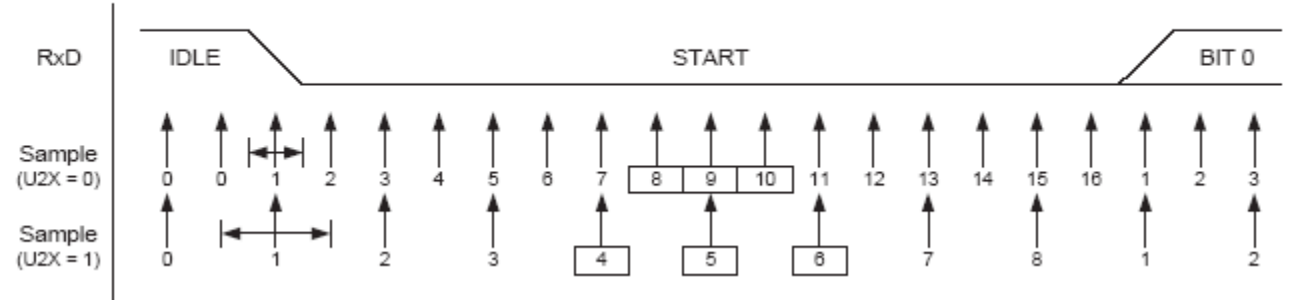


Figure 84. Sampling of Data and Parity Bit

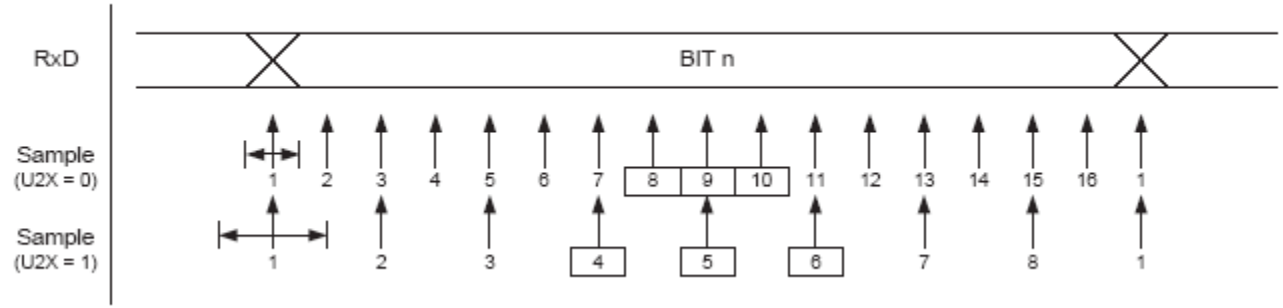
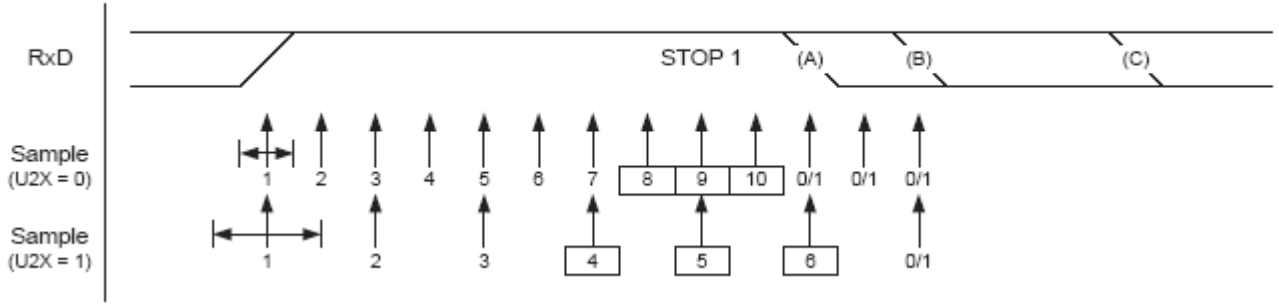
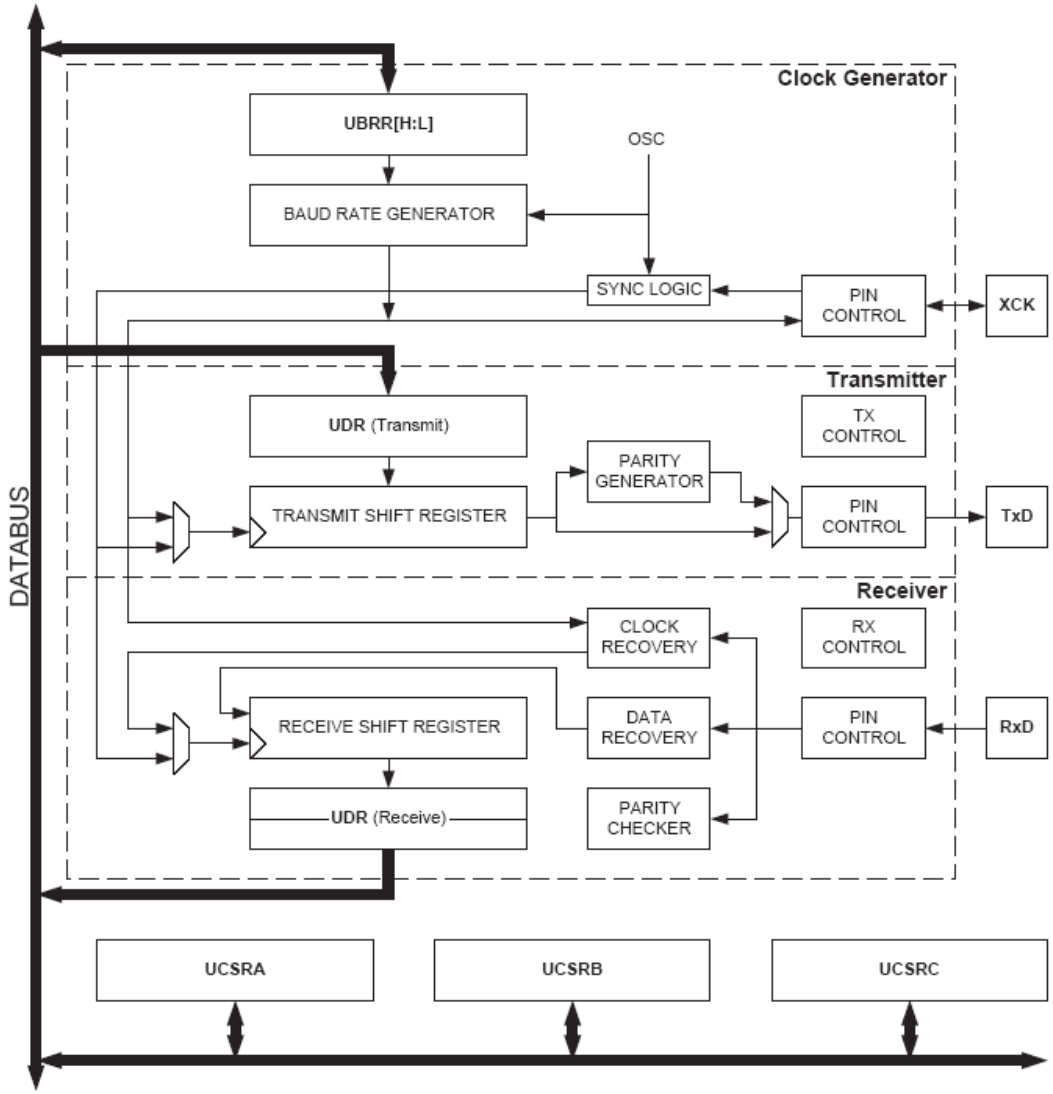


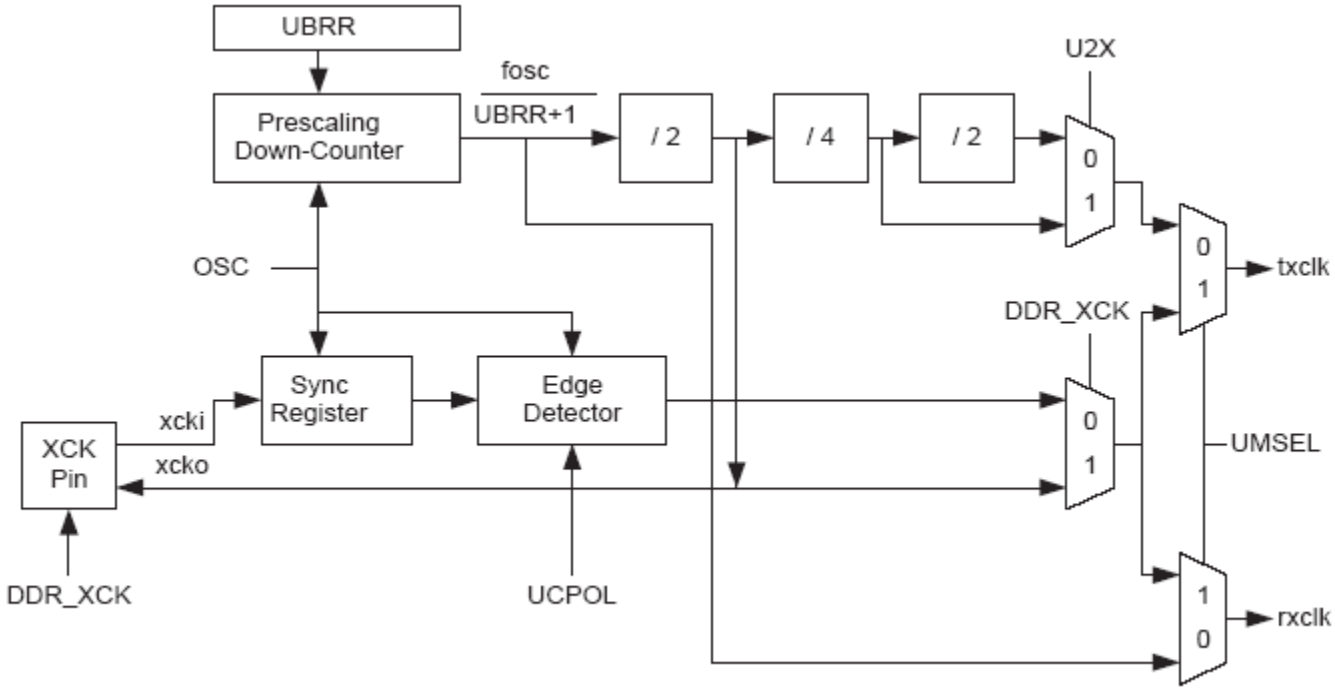
Figure 85. Stop Bit Sampling and Next Start Bit Sampling



USART device on AVR

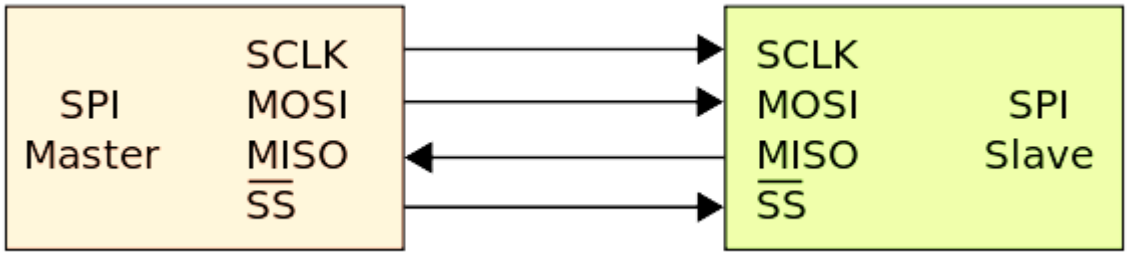


Clock generator for USART on AVR



SPI – Serial Peripheral Interface

- The Serial Peripheral Interface (SPI) bus is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems.
- SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing.
- Multiple slave devices are supported through selection with individual slave select (SS) lines.

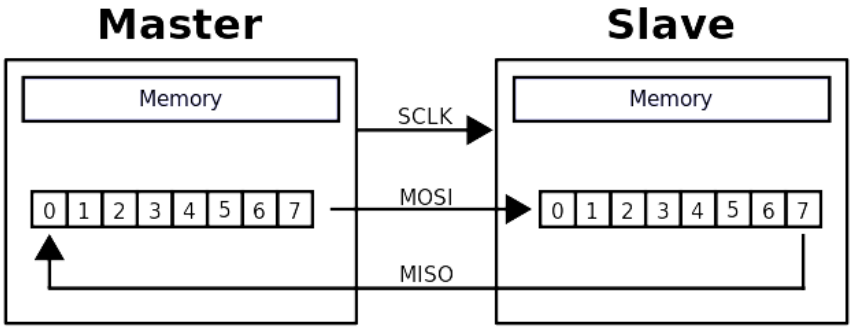


SPI – Serial Peripheral Interface

- The SPI bus specifies four logic signals:
  - SCLK : Serial Clock (output from master).
  - MOSI : Master Output, Slave Input (output from master).
  - MISO : Master Input, Slave Output (output from slave).
  - SS : Slave Select (active low, output from master).
- The SPI bus can operate with a single master device and with one or more slave devices.
- If a single slave device is used, the SS pin may be fixed to logic low if the slave permits it.
- Most slave devices have tri-state outputs, so their MISO signal becomes high impedance
- Devices without tri-state outputs cannot share SPI bus segments with other devices

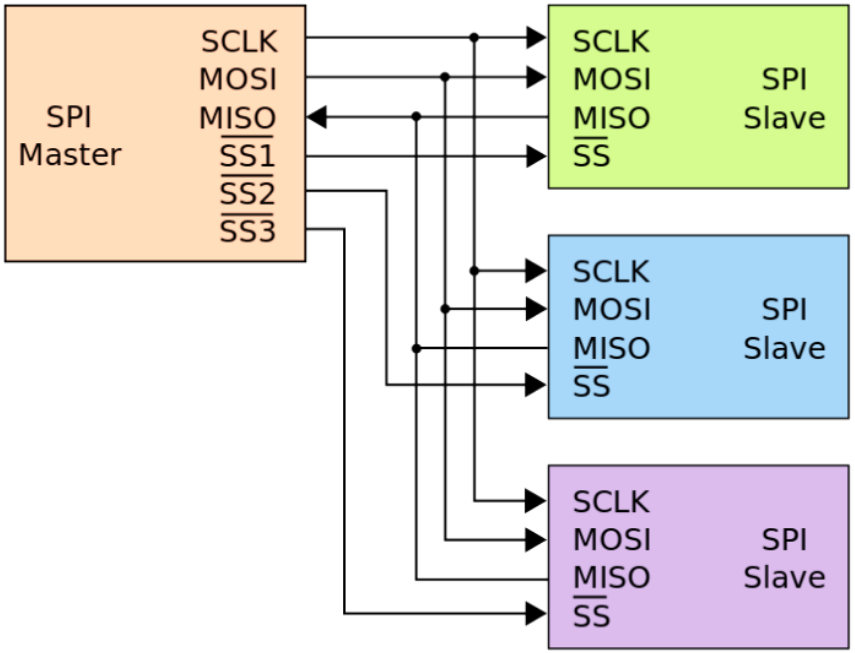
SPI Data transmission

- To begin communication selects the slave device with a logic level 0 on the select line.
- During each SPI clock cycle, a full duplex data transmission occurs. The master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it. This sequence is maintained even when only one-directional data transfer is intended.
- Transmissions normally involve two shift registers of some given word size, such as eight bits, one in the master and one in the slave; they are connected in a virtual ring topology.
- After that register has been shifted out, the master and slave have exchanged register values.
- If more data needs to be exchanged, the shift registers are reloaded and the process repeats.
- When transmission completes, the master stops toggling the clock signal, and typically deselects the slave.
- Transmissions often consist of 8-bit words. However, other word sizes are also common, for example, 16-bit words.



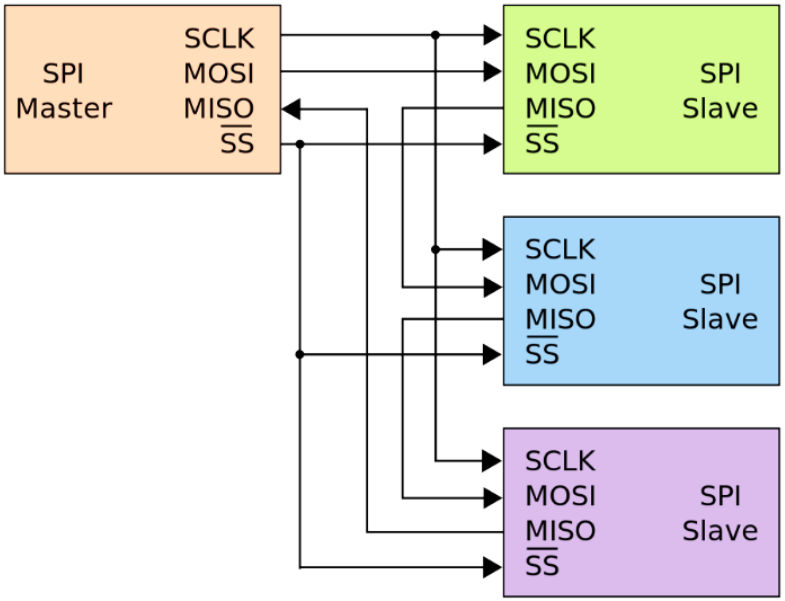
SPI Multiple slave architecture

- Independent slave configuration
  - An independent chip select line for each slave
  - This is the way SPI is normally used
  - Since the MISO pins of the slaves are connected together, they are required to be tri-state pins (high, low or high-impedance).



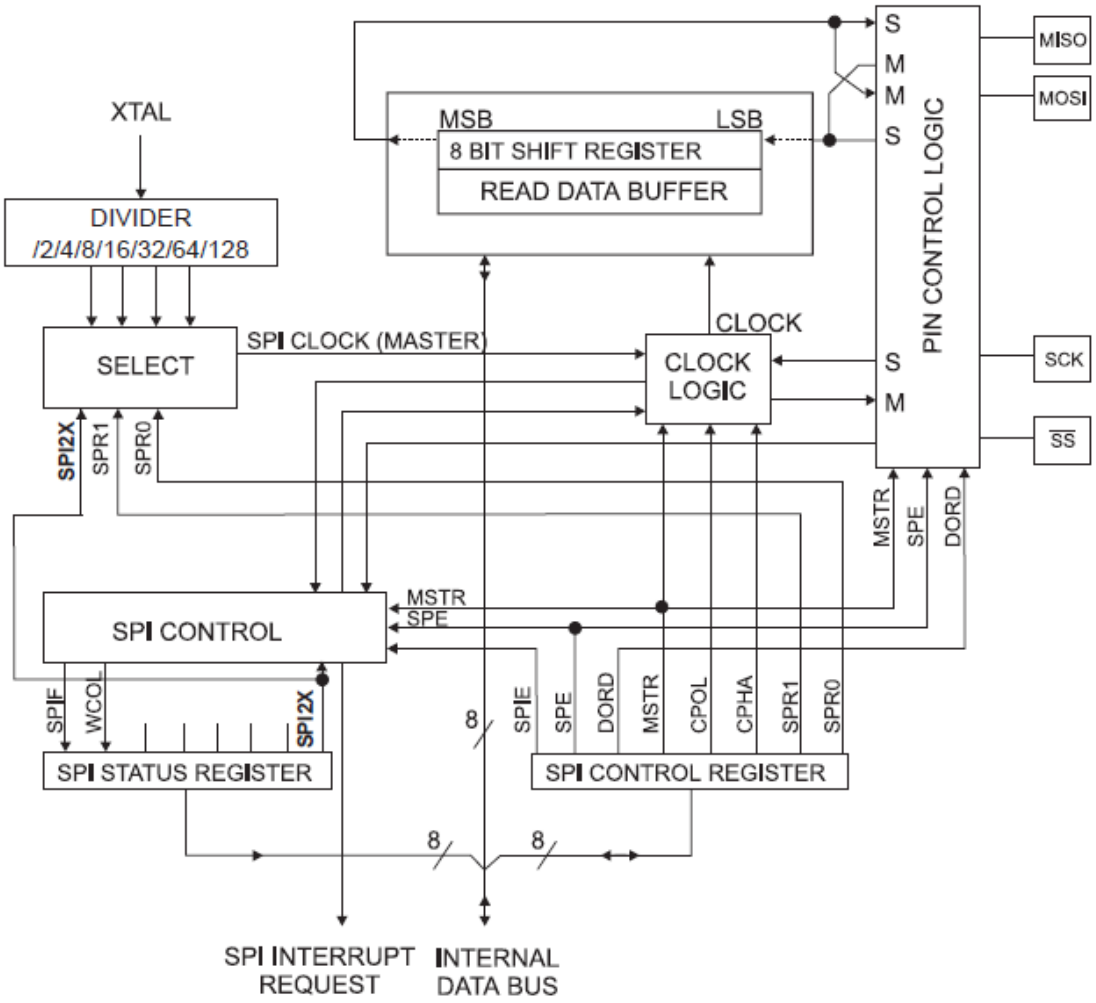
SPI Multiple slave architecture

- Daisy chain configuration
  - The first slave output is connected to the second slave input, etc.
  - The whole chain acts as a communication shift register
  - Such a feature only requires a single SS line from the master, rather than a separate SS line for each slave



SPI device on Atmel ATmega

Figure 75. SPI Block Diagram

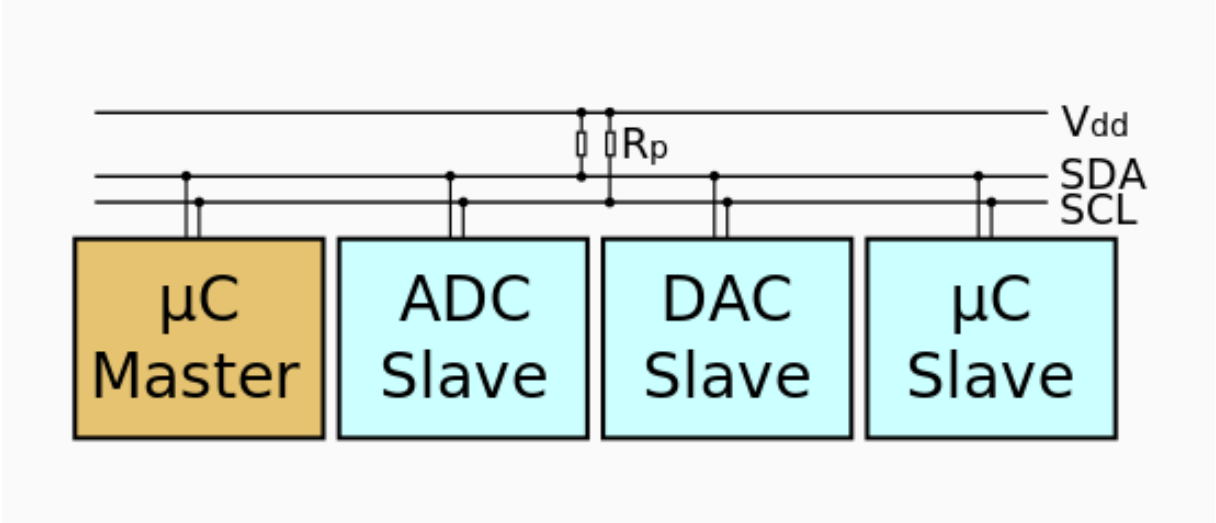


SPI resume

- Advantages
  - Full duplex communication in the default version of this protocol.
  - Push-pull drivers (as opposed to open drain) provide good signal integrity and high speed (MHz range)
  - Higher throughput than I^2C
  - Not limited to 8-bit words, Arbitrary choice of message size, content, and purpose
  - Extremely simple hardware interfacing
  - No arbitration or associated failure modes
  - Slaves use the master's clock, and do not need precision oscillators
  - At most one unique bus signal per device (chip select); all others are shared
- Disadvantages
  - Requires more pins on IC packages than I^2C
  - Chip select signals are required on shared buses
  - No hardware flow control by the slave (but the master can delay the next clock edge to slow the transfer rate)
  - No hardware slave acknowledgment (the master could be transmitting to nowhere and not know it)
  - Supports only one master device
  - Only handles short distances compared to RS-232, RS-485, or CAN-bus
  - Interrupts must either be implemented with out-of-band signals or be faked by using periodic polling similarly to USB 1.1 and 2.0

I²C bus

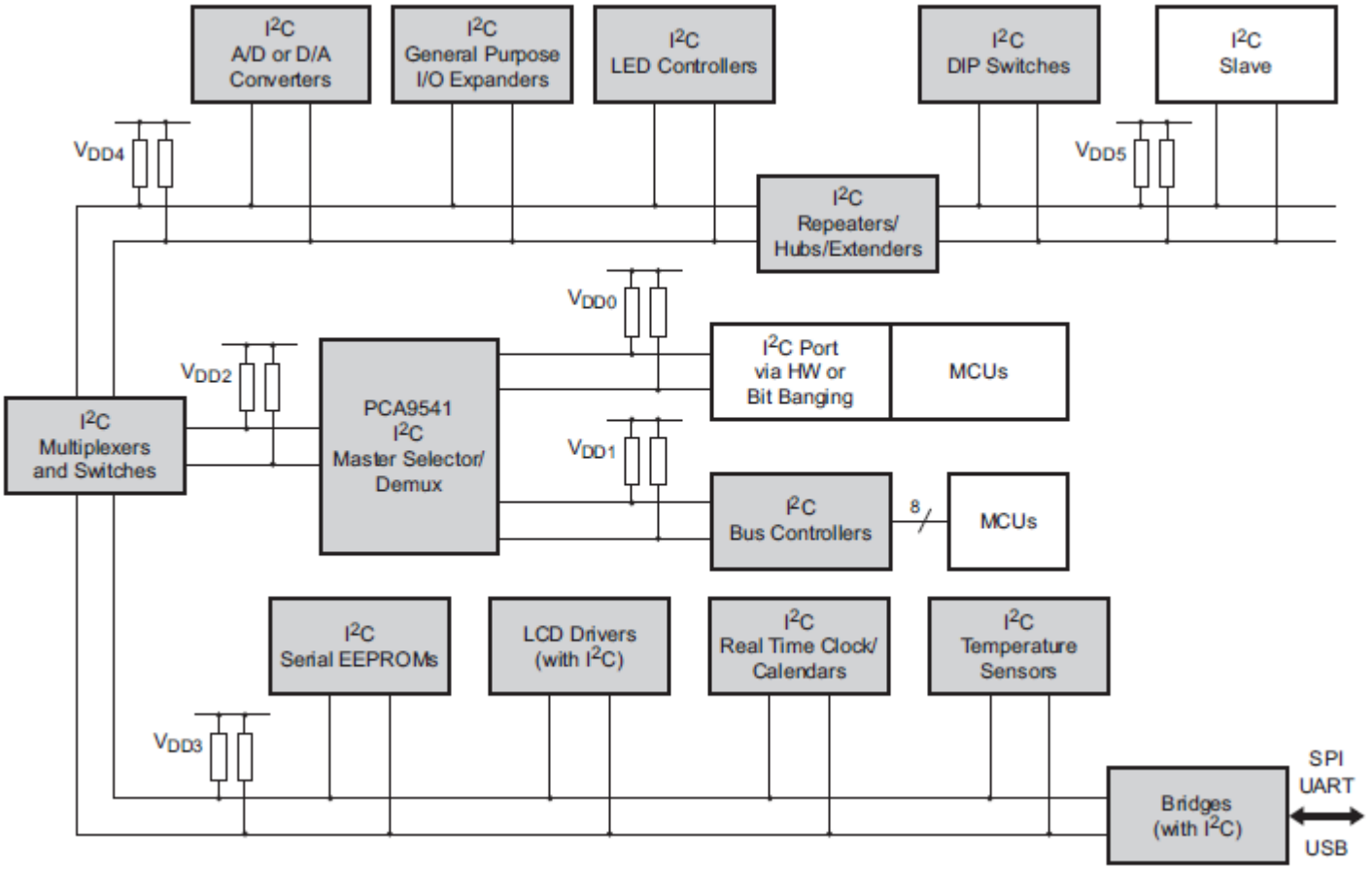
- I²C (Inter-Integrated Circuit) multi-master, multi-slave, single-ended, serial computer bus invented by Philips Semiconductor.
- Used for attaching low-speed peripherals to at board leve board level in embedded systems
- Syncronous serial communication with MAC layer
- Bidirectional data line (SDA) + monodirectional clock line (SCL)
  - Clock is always produced by master devices



I^2C bus

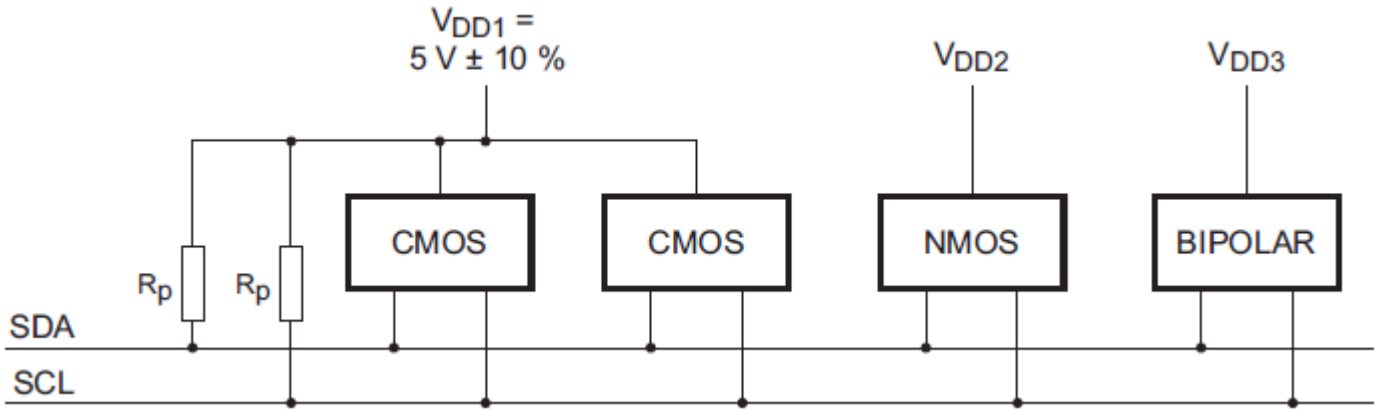
- Uses two bidirectional open-drain lines
  - Serial Data Line (SDA), Serial Clock Line (SCL), pulled up with resistors.
  - Typical voltages used are +5V or +3.3V
- Has a 7-bit or a 10-bit address space.
- Typical speeds
  - 100 kbit/s standard mode, 10 kbit/s low-speed mode
  - 400 kbit/s Fast mode, 1 Mbit/s Fast mode plus
- Protocol overheads include a slave address and perhaps a register address within the slave device as well as per-byte ACK/NACK bits.
  - The actual transfer rate of user data is lower than peak bit rates
- The maximum number of nodes is limited by the address space, and also by the total bus capacitance of 400 pF, which restricts practical communication distances to a few meters.
  - Pull-up resistor and bus capacitance form a RC low-pass filter

I²C bus example



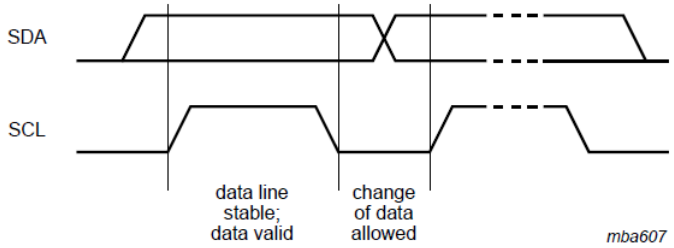
SDA and SCL signals

- Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a current-source or pull-up resistor
- When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-OR function.
- SDA and SCL logic levels
  - Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C-bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of VDD.
  - Input reference levels are set as 30 % and 70 % of VDD



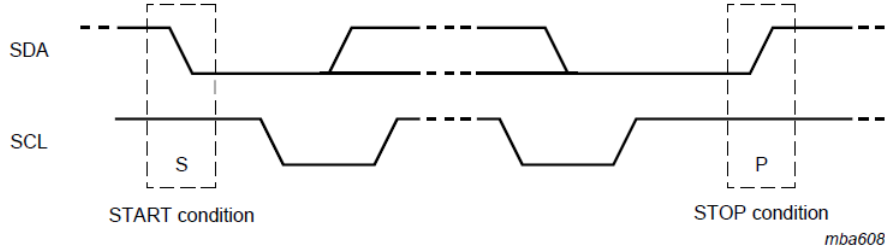
SDA and SCL signals

- Data validity
  - The data on the SDA line must be stable during the HIGH period of the clock.
  - The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW.
  - One clock pulse is generated for each data bit transferred.



Bit transfer on the I²C-bus

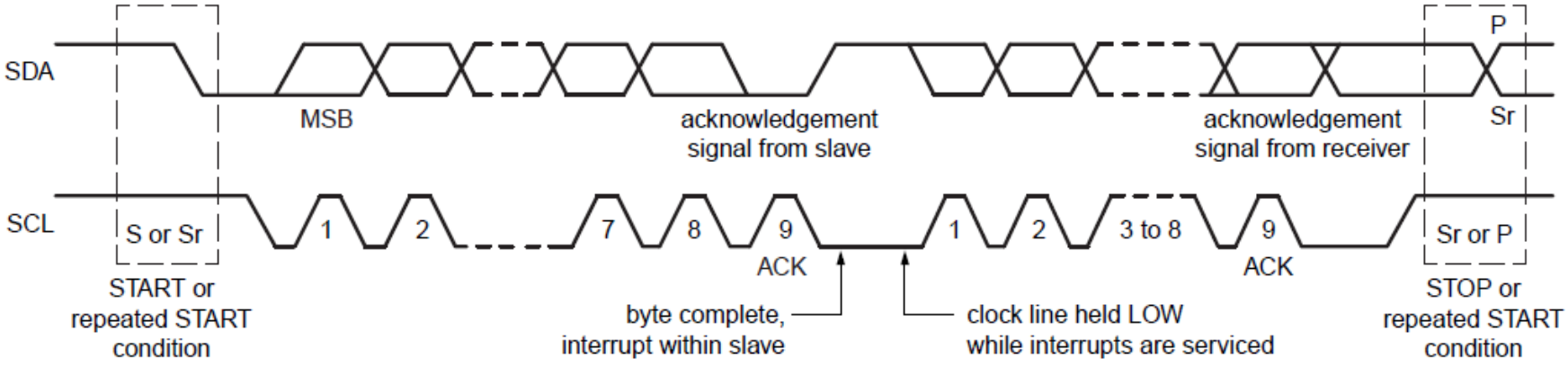
- START and STOP conditions
  - All transactions begin with a START (S) and are terminated by a STOP (P)
  - A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition.
  - A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.



START and STOP conditions

I²C data transmission

- Byte format
  - Every byte put on the SDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted.
  - Each byte must be followed by an Acknowledge bit.
  - Data is transferred with the Most Significant Bit (MSB) first (see Figure 6).
  - If a slave cannot receive or transmit another complete byte of data, it can hold the clock line SCL LOW to force the master into a wait state (clock stretching).

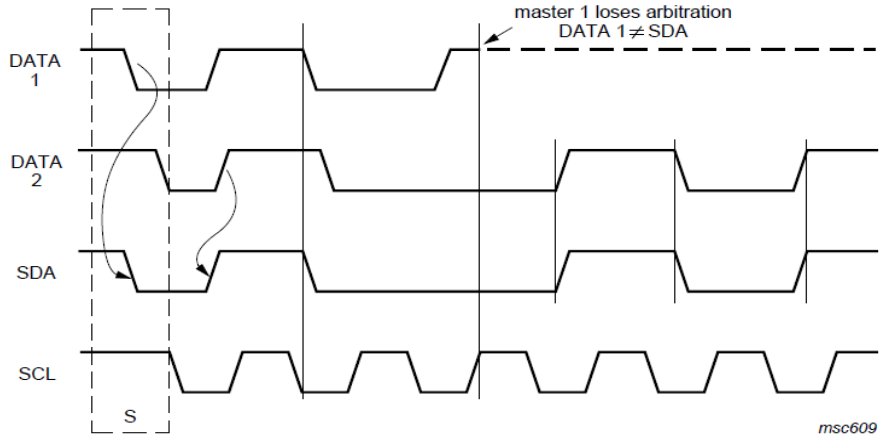


I^2C Acknowledge

- Acknowledge (ACK) and Not Acknowledge (NACK)
  - The acknowledge takes place after every byte.
  - Allow the receiver to signal the transmitter that the byte was successfully received and another byte may be sent.
  - The master generates all clock pulses, including the acknowledge ninth clock pulse.
- The Acknowledge signal is defined as follows:
  - the transmitter releases the SDA line during the acknowledge clock pulse so the receiver can pull the SDA line LOW and it remains stable LOW during the HIGH period of this clock pulse.
  - When SDA remains HIGH during this ninth clock pulse, this is defined as the NOT Acknowledge signal.
  - The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer.

I²C Arbitration

- Arbitration refers to a portion of the protocol required only if more than one master is used in the system.
- A master may start a transfer only if the bus is free. Two masters may generate a START condition within the minimum hold time which results in a valid START condition on the bus.
- Arbitration is then required to determine which master will complete its transmission.

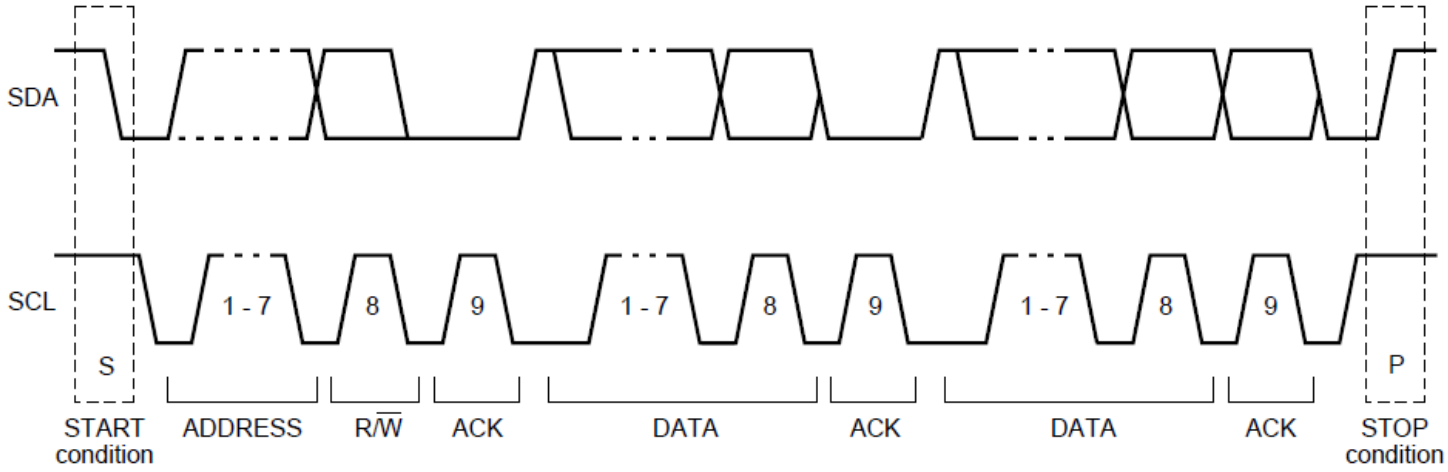


Arbitration procedure of two masters

- Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks to see if the SDA level matches what it has sent. This process may take many bits.
- Two masters can actually complete an entire transaction without error, as long as the transmissions are identical.
- The first time a master tries to send a HIGH, but detects that the SDA level is LOW, the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transaction.
- No information is lost during the arbitration process.

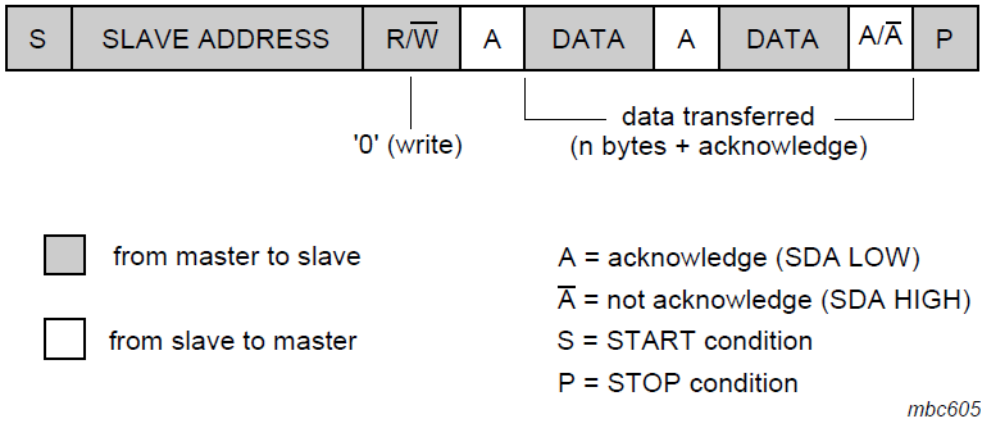
I²C Addressing

- After the START condition (S), a slave address is sent.
- This address is seven bits long followed by an eighth bit which is a data direction bit (R/W)
- A ‘0’ indicates a transmission (WRITE), a ‘1’ indicates arequest for data (READ)
- A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave.



I²C Master Write transfer

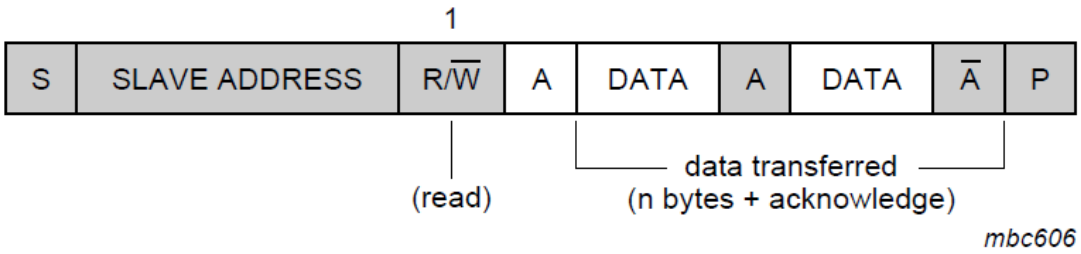
- Master-transmitter transmits to slave-receiver. The transfer direction is not changed.
- The slave receiver acknowledges each byte.



A master-transmitter addressing a slave receiver with a 7-bit address (the transfer direction is not changed)

I²C Master Read transfer

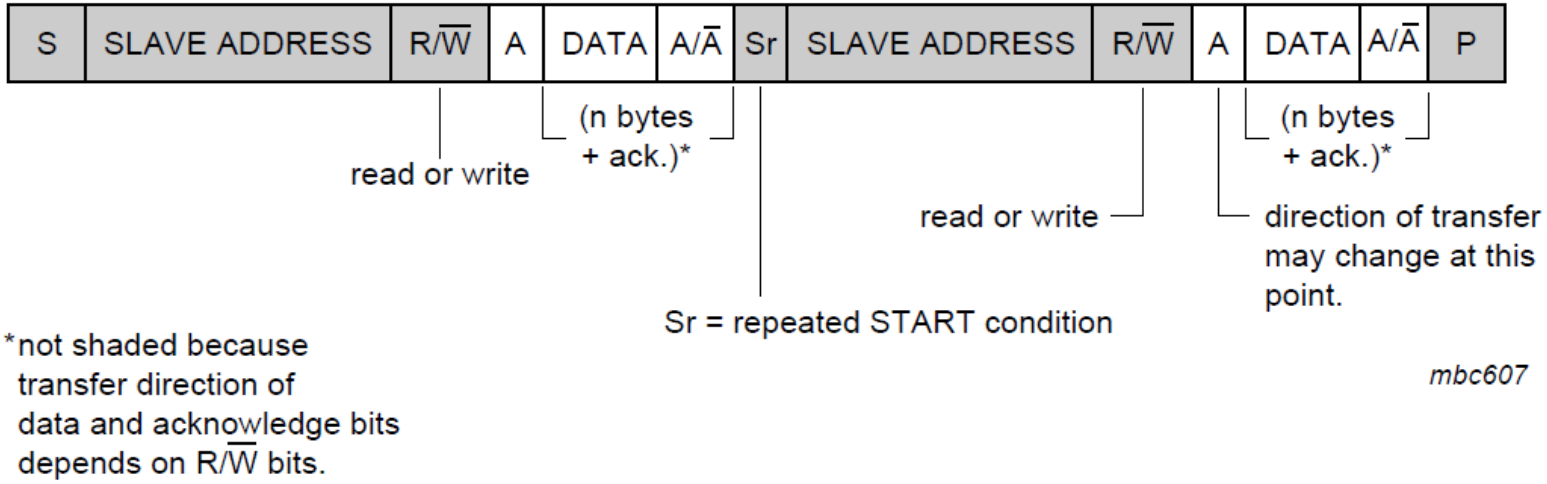
- Master reads slave immediately after first byte.
- At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes a slave-transmitter.
- This first acknowledge is still generated by the slave. The master generates subsequent acknowledges.
- The STOP condition is generated by the master, which sends a not-acknowledge (A) just before the STOP condition.



A master reads a slave immediately after the first byte

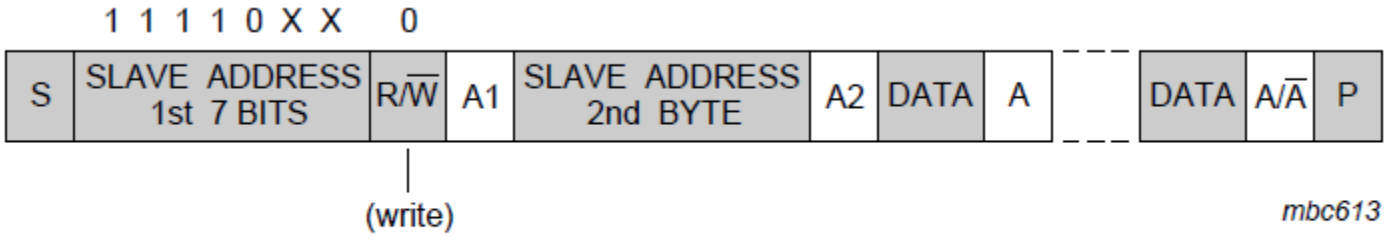
I²C Master Combined transfer

- During a change of direction within a transfer, the START condition and the slave address are both repeated, but with the R/W bit reversed.
- Combined formats can be used, for example, to control a serial memory. The internal memory location must be written during the first data byte. After the START condition and slave address is repeated, data can be transferred.



I²C 10-bit addressing

- 10-bit addressing expands the number of possible addresses.
- Devices with 7-bit and 10-bit addresses can be connected to the same I2C-bus, and both 7-bit and 10-bit addressing can be used in all bus speed modes.
- The 10-bit slave address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr).
- The first seven bits of the first byte are the combination 1111 0XX of which the last two bits (XX) are the two Most-Significant Bits (MSB) of the 10-bit address; the eighth bit of the first byte is the R/W bit that determines the direction of the message.



A master-transmitter addresses a slave-receiver with a 10-bit address

I^2C Electrical Characteristiscs

Table 9. Characteristics of the SDA and SCL I/O stages
n/a = not applicable.

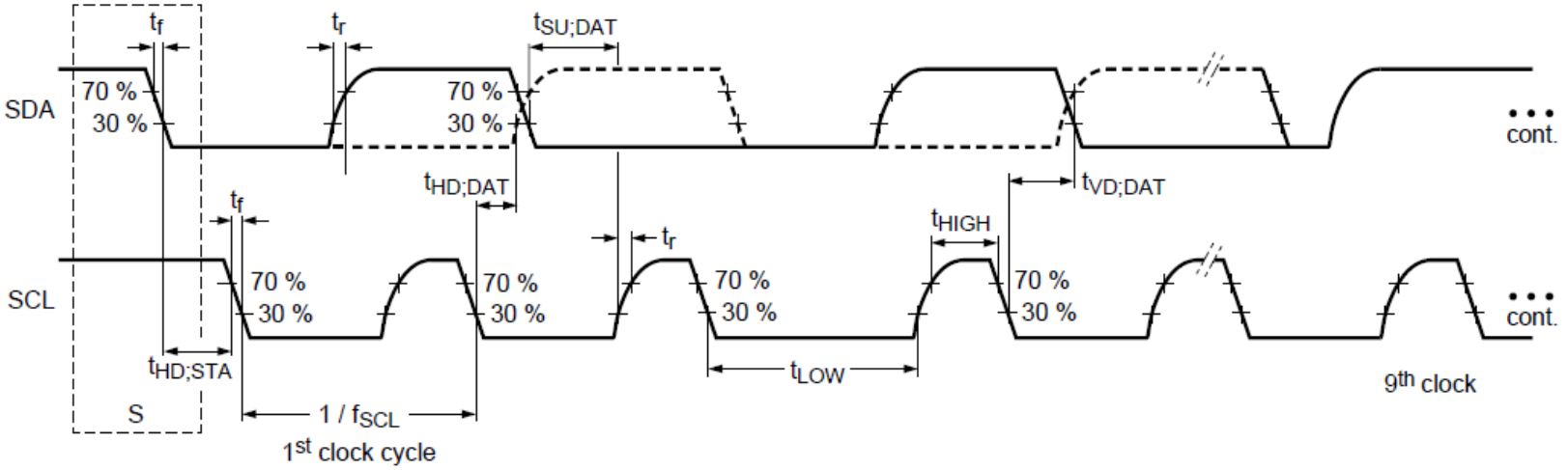
Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
V_{IL}	LOW-level input voltage[1]		–0.5	$0.3V_{DD}$	–0.5	$0.3V_{DD}$	–0.5	$0.3V_{DD}$	V
V_{IH}	HIGH-level input voltage[1]		$0.7V_{DD}$	[2]	$0.7V_{DD}$	[2]	$0.7V_{DD}$[1]	[2]	V
V_{hys}	hysteresis of Schmitt trigger inputs		-	-	$0.05V_{DD}$	-	$0.05V_{DD}$	-	V
V_{OL1}	LOW-level output voltage 1	(open-drain or open-collector) at 3 mA sink current; $V_{DD} > 2$ V	0	0.4	0	0.4	0	0.4	V
V_{OL2}	LOW-level output voltage 2	(open-drain or open-collector) at 2 mA sink current[3]; $V_{DD} \leq 2$ V	-	-	0	$0.2V_{DD}$	0	$0.2V_{DD}$	V
I_{OL}	LOW-level output current	$V_{OL} = 0.4$ V	3	-	3	-	20	-	mA
		$V_{OL} = 0.6$ V[4]	-	-	6	-	-	-	mA
t_{of}	output fall time from V_{IHmin} to V_{ILmax}		-	250[5]	$20 \times (V_{DD} / 5.5\ V)$[6]	250[5]	$20 \times (V_{DD} / 5.5\ V)$[6]	120[7]	ns
t_{SP}	pulse width of spikes that must be suppressed by the input filter		-	-	0	50[8]	0	50[8]	ns
I_i	input current each I/O pin	$0.1V_{DD} < V_I < 0.9V_{DDmax}$	–10	+10	–10[9]	+10[9]	–10[9]	+10[9]	µA
C_i	capacitance for each I/O pin[10]		-	10	-	10	-	10	pF

I²C Electrical Characteristiscs

Table 10. Characteristics of the SDA and SCL bus lines for Standard, Fast, and Fast-mode Plus I²C-bus devices[1]

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency		0	100	0	400	0	1000	kHz
$t_{HD;STA}$	hold time (repeated) START condition	After this period, the first clock pulse is generated.	4.0	-	0.6	-	0.26	-	µs
t_{LOW}	LOW period of the SCL clock		4.7	-	1.3	-	0.5	-	µs
t_{HIGH}	HIGH period of the SCL clock		4.0	-	0.6	-	0.26	-	µs
$t_{SU;STA}$	set-up time for a repeated START condition		4.7	-	0.6	-	0.26	-	µs
$t_{HD;DAT}$	data hold time[2]	CBUS compatible masters (see Remark in Section 4.1)	5.0	-	-	-	-	-	µs
		I²C-bus devices	0[3]	-[4]	0[3]	-[4]	0	-	µs
$t_{SU;DAT}$	data set-up time		250	-	100[5]	-	50	-	ns
t_r	rise time of both SDA and SCL signals		-	1000	20	300	-	120	ns
t_f	fall time of both SDA and SCL signals[3][6][7][8]		-	300	$20 \times (V_{DD} / 5.5\ V)$	300	$20 \times (V_{DD} / 5.5\ V)$[9]	120[8]	ns
$t_{SU;STO}$	set-up time for STOP condition		4.0	-	0.6	-	0.26	-	µs
t_{BUF}	bus free time between a STOP and START condition		4.7	-	1.3	-	0.5	-	µs
C_b	capacitive load for each bus line[10]		-	400	-	400	-	550	pF
$t_{VD;DAT}$	data valid time[11]		-	3.45[4]	-	0.9[4]	-	0.45[4]	µs
$t_{VD;ACK}$	data valid acknowledge time[12]		-	3.45[4]	-	0.9[4]	-	0.45[4]	µs
V_{nL}	noise margin at the LOW level	for each connected device (including hysteresis)	$0.1V_{DD}$	-	$0.1V_{DD}$	-	$0.1V_{DD}$	-	V
V_{nH}	noise margin at the HIGH level	for each connected device (including hysteresis)	$0.2V_{DD}$	-	$0.2V_{DD}$	-	$0.2V_{DD}$	-	V

I^2C Definition of timing

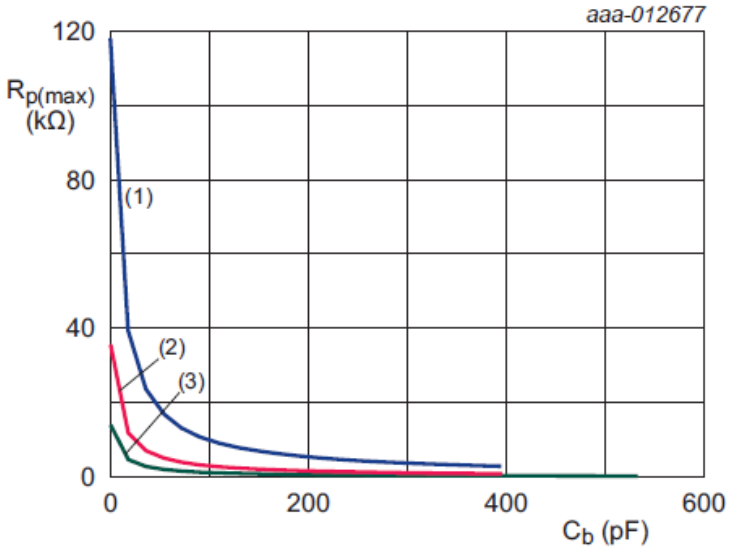


$V_{IL} = 0.3V_{DD}$

$V_{IH} = 0.7V_{DD}$

I^2C Pull-up resistor sizing

- The bus capacitance is the total capacitance of wire, connections and pins. This capacitance limits the maximum value of Rp due to the specified rise time.
- Consider the VDD related input threshold of VIH = 0.7VDD and VIL = 0.3VDD for the purposes of RC time constant calculation. Then :
  - $V(t) = VDD\ (1 - e^{-t/RC})$
- $V(t1) = 0.3 \times VDD = VDD\ (1 - e^{-t1/RC})$; then $t1 = 0.3566749 \times RC$
- $V(t2) = 0.7 \times VDD = VDD\ (1 - e^{-t2/RC})$; then $t2 = 1.2039729 \times RC$
- $T_{rise} = t2 - t1 = 0.8473 \times RC$
- For each mode, the Rp(max) is a function of the rise time maximum (tr) from Table 10 and the estimated bus capacitance (Cb):



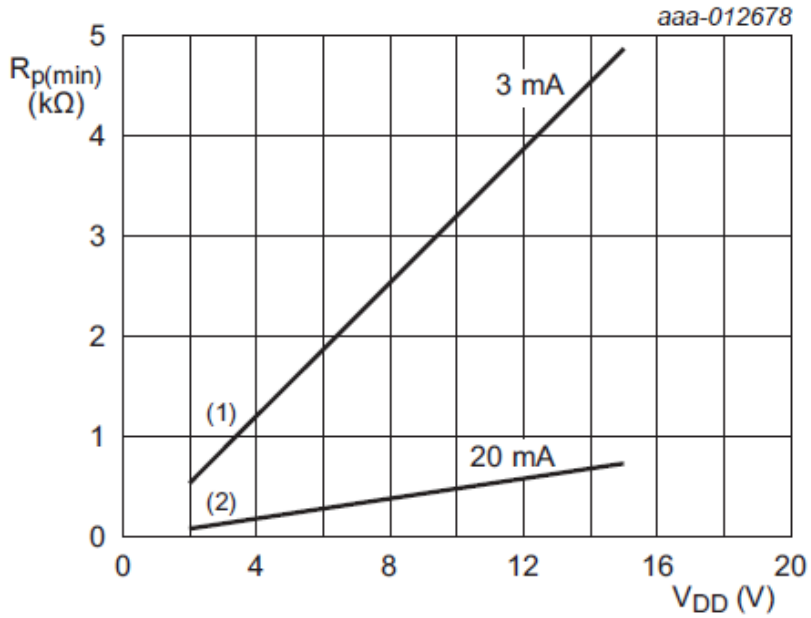
(1) Standard-mode

(2) Fast-mode

(3) Fast-mode Plus

I^2C Pull-up resistor sizing

- The supply voltage limits the minimum value of resistor Rp due to the specified minimum sink current of 3 mA for Standard-mode and Fast-mode, or 20 mA for Fast-mode Plus.
- Rp(min) as a function of VDD is shown in Figure

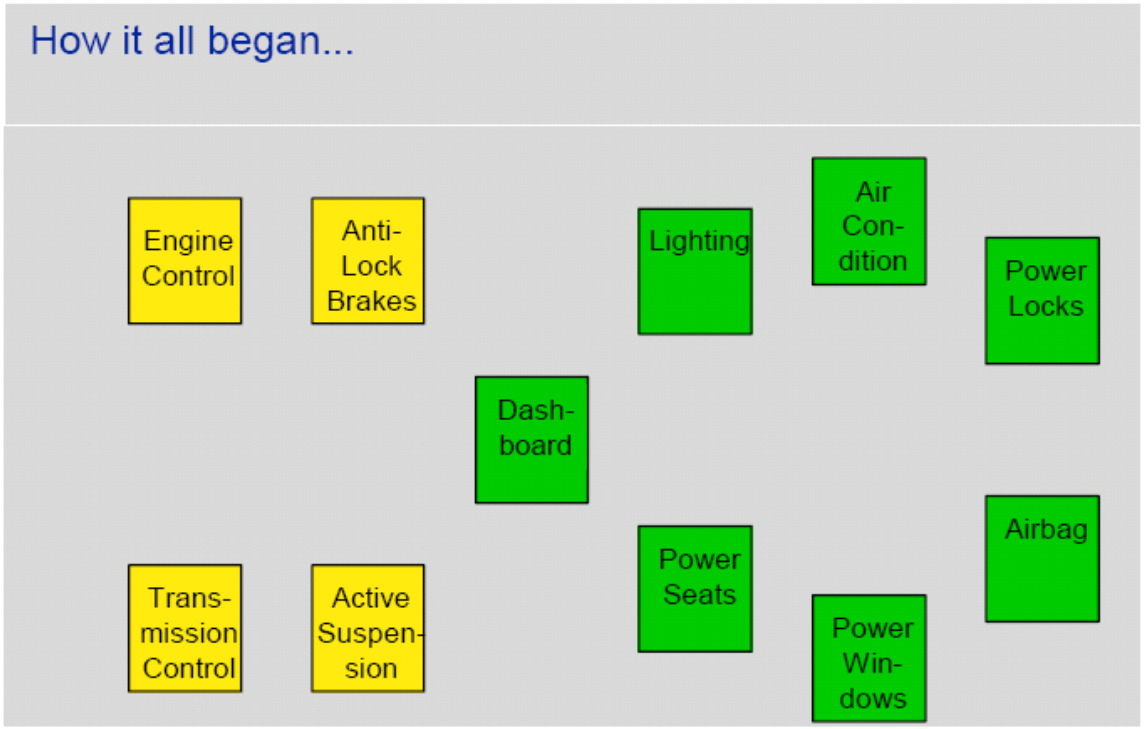


(1) Fast-mode and Standard-mode

(2) Fast-mode Plus

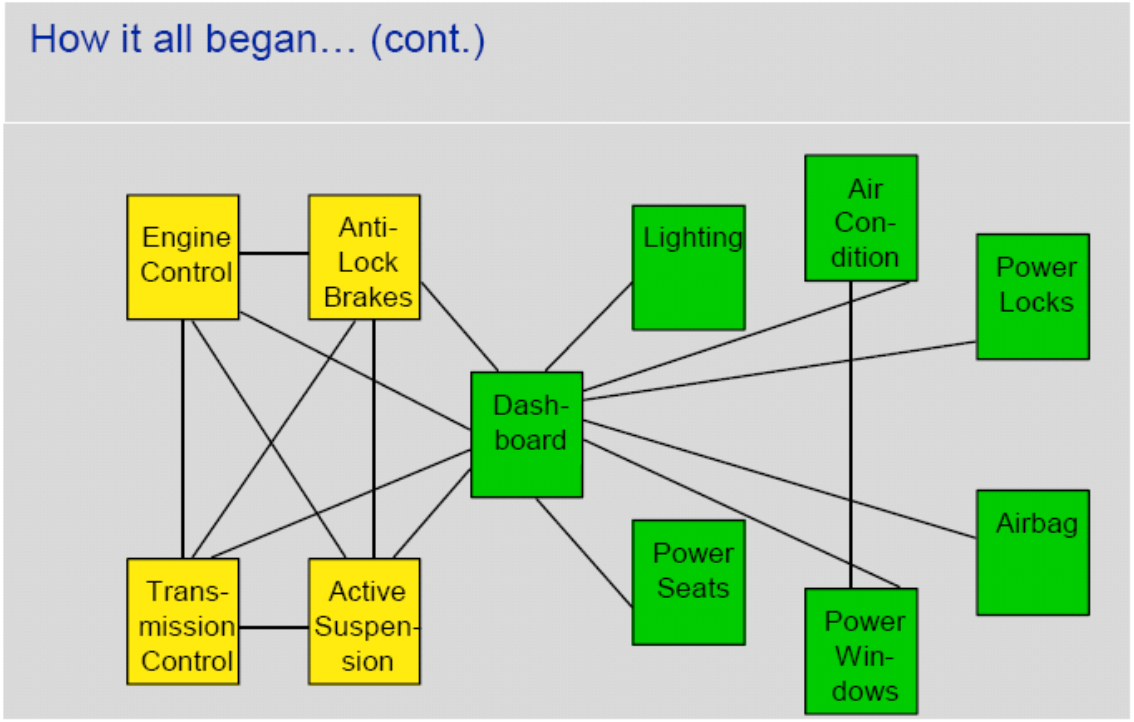
CAN bus

- More and more electronic devices into vehicles to add safety/confort and reduce fuel consumption and emissions



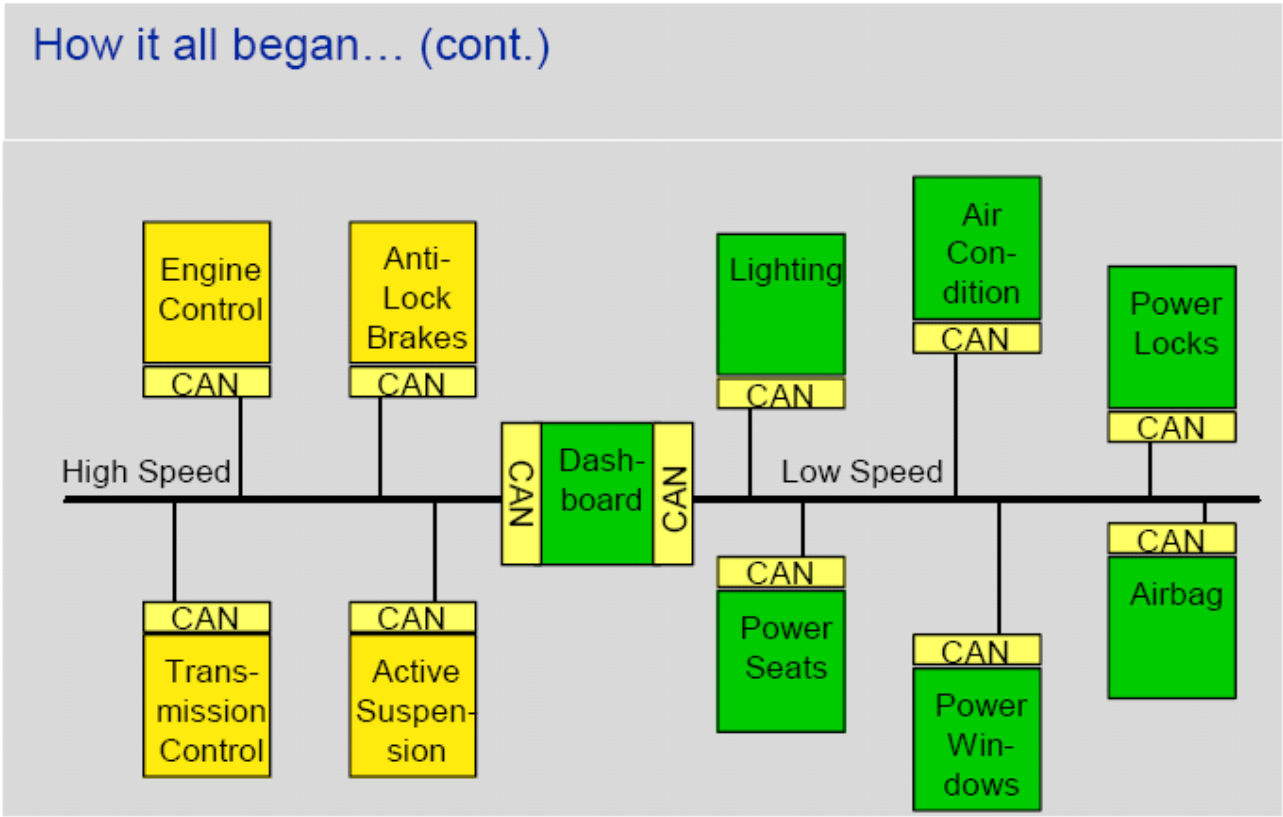
CAN bus

- To improve the behavior of the vehicle even further, it was necessary to exchange information → Several miles of P2P wiring
  - Does not scale
  - Material cost
  - Reliability issues



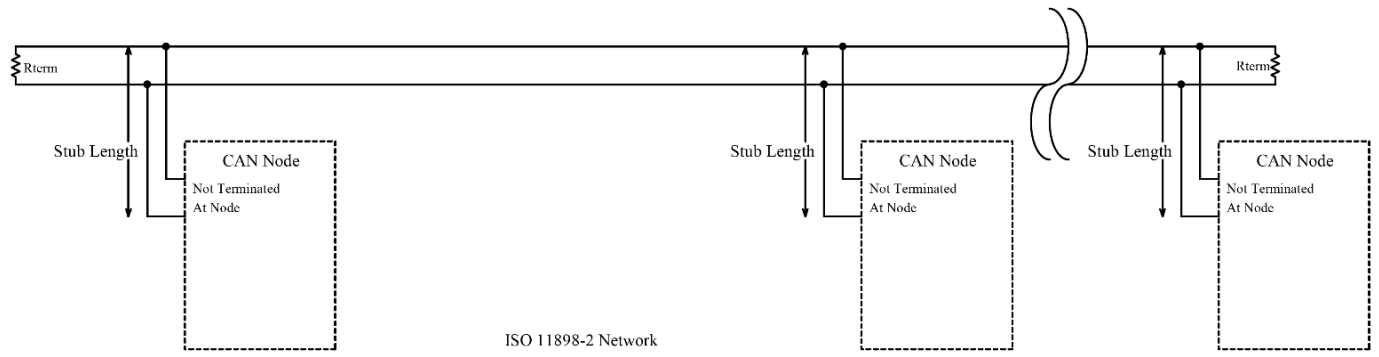
CAN bus

- P2P wiring replaced by a serial bus + CAN specific hw



CAN bus

- Controller Area Network (CAN bus) is a bus standard designed to allow microcontrollers and devices to communicate with each other.
- It is a message-based protocol, designed originally for automotive applications, but is also used in many other contexts.
- Standardized as ISO 11898-1 (data link layer) and ISO 11898-2 (physical layer)
- Today the CAN bus is also used as a fieldbus in general automation environments, primarily due to the low cost



CAN bus – Physical layer

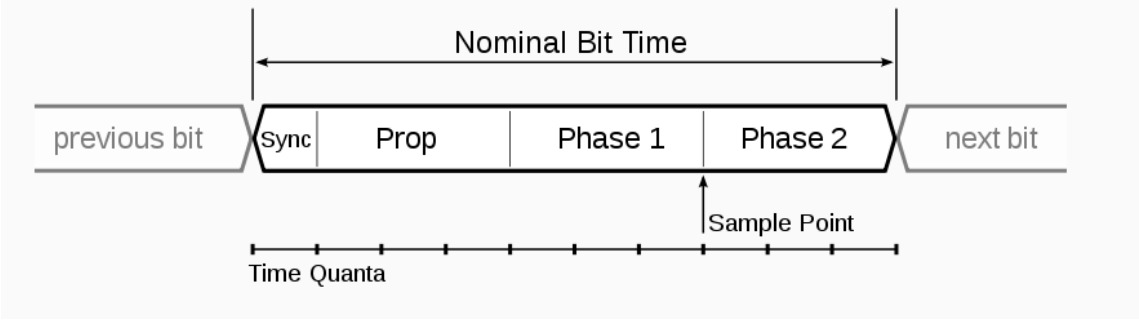
- The bus consists of a single line that carries data bits, and resynchronization information for clock recovery
- The bus has two logic values: dominant or recessive
- The physical line is a differential pair (CAN-Low/CAN High)
  - During a recessive state the signal lines and resistor(s) remain in a high impedances state with respect to both rails. Voltages on both CAN+ and CAN- tend (weakly) towards ½ rail voltage.
  - A recessive state is only present on the bus when none of the transmitters on the bus is asserting a dominant state.
  - During a dominant state the signal lines and resistor(s) move to a low impedance state with respect to the rails so that current flows through the resistor. CAN+ voltage tends to +5 V and CAN- tends to 0 V.
- Noise immunity is achieved by maintaining the differential impedance of the bus at a low level with low-value resistors (120 ohms) at each end of the bus.
- Multiple access on CAN bus is achieved by the electrical logic of the system supporting just two states that are conceptually analogous to a ‘wired OR’ network.

CAN bus – bit timing

- All nodes on the CAN network must operate at the same nominal bit rate, but noise, phase shifts, oscillator tolerance and oscillator drift mean that the nominal bit rate may not be the same as the actual bit rate.
- Since a separate clock signal is not used, a means of synchronizing the nodes is necessary.
- Synchronization is important during arbitration since the nodes in arbitration must be able to see both their transmitted data and the other nodes transmitted data at the same time.
- Synchronization is also important to ensure that variations in oscillator timing between nodes does not cause errors.
- Synchronization starts with a hard synchronization on the first recessive to dominant transition after a period of bus idle (the start bit). Resynchronization occurs on every recessive to dominant transition during the frame.

CAN bus – bit timing

- The CAN controller expects the transition to occur at a multiple of the nominal bit time. If the transition does not occur at the exact time the controller expects it, the controller adjusts the nominal bit time accordingly.
- The adjustment is accomplished by dividing each bit into a number of time slices called quanta, and assigning some number of quanta to each of the four segments within the bit: synchronization, propagation, phase segment 1 and phase segment 2.
- The number of quanta the bit is divided into can vary by controller, and the number of quanta assigned to each segment can be varied depending on bit rate and network conditions.
- A transition that occurs before or after it is expected causes the controller to calculate the time difference and lengthen phase segment 1 or shorten phase segment 2 by this time. This effectively adjusts the timing of the receiver to the transmitter to synchronize them.
- **This resynchronization process is done continuously at every recessive to dominant transition** to ensure the transmitter and receiver stay in sync.



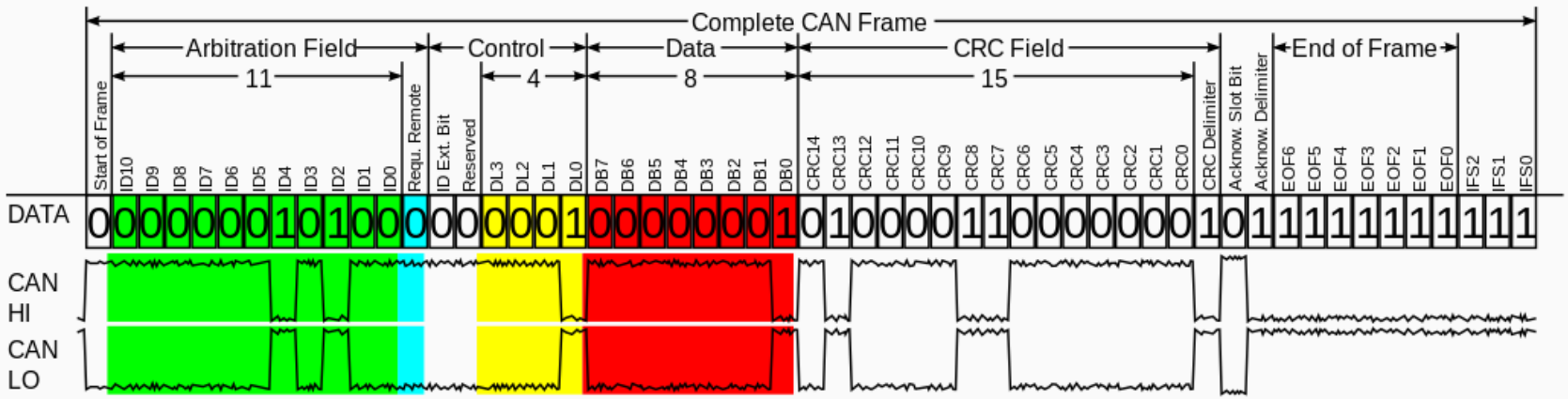
Example CAN bit timing with 10 time quanta per bit.

CAN bus – Frames

- A CAN network can be configured to work with two different message (or "frame") formats
  - the base frame format supports a length of 11 bits for the identifier
  - the extended frame format supports a length of 29 bits for the identifier, made up of the 11-bit identifier ("base identifier") and an 18-bit extension ("identifier extension").
  - The distinction between CAN base frame format and CAN extended frame format is made by using the IDE bit.
- CAN has four frame types:
  - Data frame: a frame containing node data for transmission
  - Remote frame: a frame requesting the transmission of a specific identifier
  - Error frame: a frame transmitted by any node detecting an error
  - Overload frame: a frame to inject a delay between data and/or remote frame
- Data frame
  - The data frame is the only frame for actual data transmission.

CAN bus – Frames

Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier for the data which also represents the message priority
Remote transmission request (RTR)	1	Dominant (0) (see Remote Frame below)
Identifier extension bit (IDE)	1	Declaring if 11 bit message ID or 29 bit message ID is used. Dominant (0) indicate 11 bit message ID while Recessive (1) indicate 29 bit message.
Reserved bit (r0)	1	Reserved bit (it must be set to dominant (0), but accepted as either dominant or recessive)
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)



CAN bus - Bit stuffing

- To ensure enough transitions to maintain synchronization, a bit of opposite polarity is inserted after five consecutive bits of the same polarity.
- This practice is called bit stuffing, and is necessary due to the non-return to zero (NRZ) coding used with CAN. The stuffed data frames are destuffed by the receiver.
- All fields in the frame are stuffed with the exception of the CRC delimiter, ACK field and end of frame which are a fixed size and are not stuffed.
- In the fields where bit stuffing is used, six consecutive bits of the same type (111111 or 000000) are considered an error.
- Bit stuffing means that data frames may be larger than one would expect by simply enumerating the bits shown in the tables above.