Unit 6 : Introduction to Communication Protocol
6.1 Introduction of RS-232, Study of RS-232 Pinout
RS232 connector:

Introduction: RS232 is a port used for data exchange between equipments. It was designed for data exchange between DTE (Data Terminal Equipment) or PC and DCE (Data Communication Equipment) or MODEM. The need for RS232 came from limitations raised by parallel data exchange. **RS232 uses serial communication protocol** where data exchange is done bit by bit. Although RS232 is later replaced by faster **USB** (Universal Serial Bus) it is still popular in some areas. RS232 used to have 25 pin, now it is shrunk to just 9 pin.

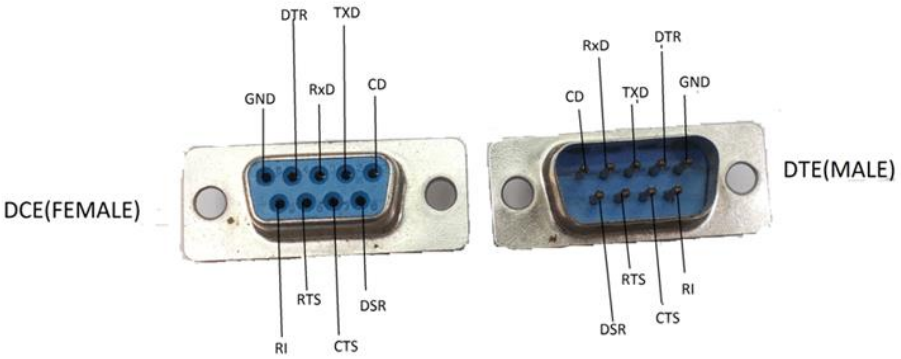
RS-232 was first introduced in 1960 by the Electronic Industries Association (EIA) as a Recommended Standard.[

RS232 Pin Configuration

New RS232 has nine pins as mentioned earlier. These nine pins are arranged in the port as shown in RS232 Connector Pinout. The DCE and DTE ports are exactly similar except for the direction of data flow. These nine pins are roughly divided in to three categories and we will discuss each category below.

Pin Number	**Pin Name**	**Description**
DATA pins (Data flow takes through these pins)		
2	RXD	Receive Data (Data is received though this pin)
3	TXD	Transmit Data (Data is transmitted through this pin)
CONTROL pins (These pins are for establishing interface and to avoid data loss)		
1	CD	Carrier Detect(Set by MODEM when answer is received by remote MODEM)
4	DTR	Data Terminal Ready(Set by PC to prepare MODEM to be connected to telephone circuit)
6	DSR	Data Set Ready(Set by MODEM to tell PC it is ready to receive and send data)
7	RTS	Request To Send(Set by PC to tell MODEM that MODEM can begin sending data)
8	CTS	Clear To send(Set by MODEM to tell PC that it is ready to receive data)
9	RI	Set by MODEM to tell PC a ringing condition has been detected.
5	GND	Ground (**Used as reference for all pin voltage pulses**)

RS232 PINOUT



RS232 Features and Specifications

1. RS232 uses Asynchronous communication so no clock is shared between PC and MODEM.
2. Logic '1' on pin is stated by voltage of range '-15V to -3V' and Logic '0' on pin is stated by voltage of range '+3V to +15V'. The logic has wide voltage range giving convenience for user.
3. MAX232 IC can be installed easily to establish RS232 interface with microcontrollers.
4. Full duplex interface of RS232 is very convenient.
5. Two pin simplex RS232 interface can also be established easily if required.
6. A maximum data transfer speed of 19 Kbps(Kilobits per second) is possible through RS232
7. A maximum current of 500mA can be drawn from pins of RS232
8. The interface can be established up to a distance of 50 feet.

Disadvantages of RS232

1. There is no pin dedicated for powering devices (No VCC)
2. More communication pins
3. Switching voltages between +15v and -15v is difficult at higher speeds
4. A maximum speed of 19 Kbps
5. A maximum distance of 50 feet
6. More pins lead to higher noise
7. Only a single device can be connected to RS232 connector unlike I2C
8. Need hardware to convert high voltage logic of RS232 to be compatible to TTL (controller and processor units)

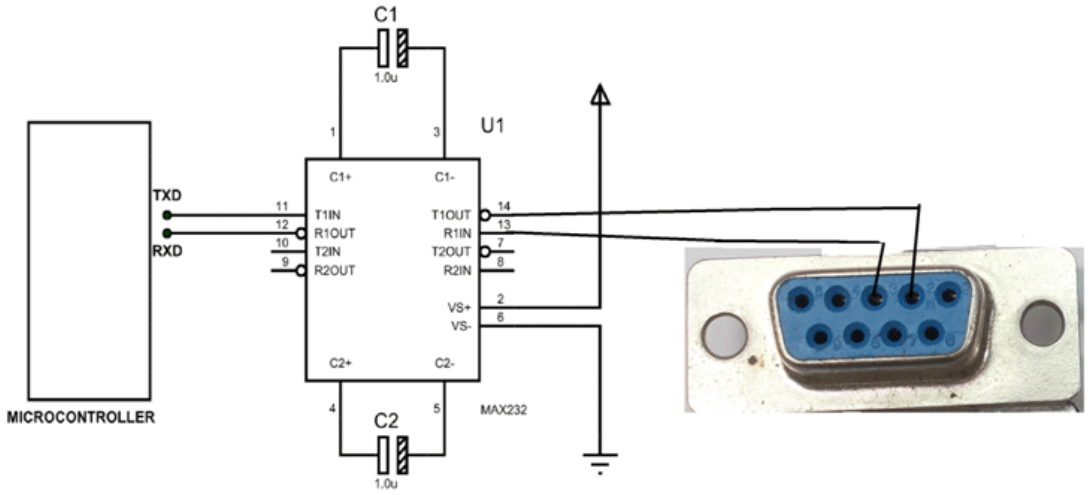
Applications of RS232.

A few examples where RS232 are suitable to use:

1. In a simple communication interface between two units. A two pin full duplex communication can be establishes easily on RS232 port.
2. RS232 is used in systems where clock sharing is difficult. RS232 is ASYNCHRONOUS so there will be no clock sharing between systems. Once baud rate is set the units will sample the data according to set baud rate.
3. RS232 is also used to control a single unit specifically without delay or errors.
4. RS232 interface also delivers data with more accuracy which is a requirement in some cases.
5. Modem, servers , PC, printer scanners etc

How to use RS232 connector?

As mentioned before we cannot connect RS232 directly to controller we need MAX232 IC to covert high voltage signals to TTL and vice versa. A typical circuit for it is shown below.



In here we are connecting controller to female RS232 through MAX232 converter chip. The communication voltages reach as high as +15V and as low as -15V in RS232. There voltage levels cannot be used in sensitive electronics so we use a mediator that is MAX232.

The chip coverts TTL logic pulses of controller to RS232 voltage level pulses and vice versa. Without this chip you may damage something permanently.

Voltage levels for RS232

The RS-232 standard defines the voltage levels that correspond to logical one and logical zero levels for the data transmission and the control signal lines. Valid signals are either in the range of +3 to +15 volts or the range −3 to −15 volts with respect to the "Common Ground" (GND) pin; consequently, the range between −3 to +3 volts is not a valid RS-232 level. For data transmission lines (TxD, RxD, and their secondary channel equivalents), logic one is represented as a negative voltage and the signal condition is called "mark". Logic zero is signaled with a positive voltage and the signal condition is termed "space". Control signals have the opposite polarity: the asserted or active state is positive voltage and the de-asserted or inactive state is negative voltage. Examples of control lines include request to send (RTS), clear to send (CTS), data terminal ready (DTR), and data set ready (DSR).

MAX232 IC

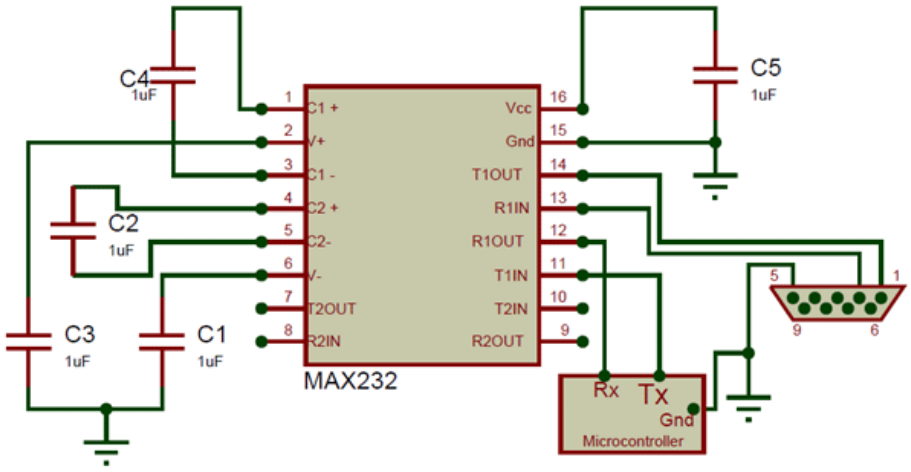
IC MAX232 is used for TTL/CMOS to RS232 conversion. Meaning most of the Microcontrollers (PIC/ARM/Atmel) operates on TTL/CMOS logic that is it communicates through either 0V or +5V, but computers work with the help of RS232 which operates at logic level -24V or +24V. So, if we have to interface these microcontrollers with Computer we need to convert the TTL/CMOS logic to RS232 logic. Hence if you are looking for an IC to perform this conversion and interface a Microcontroller with your computer then **MAX232 IC** is the right one for you

This IC is easy to set-up and can be used easily. The IC work with the help of +5V, hence power the Vcc with +5V and the ground pin to circuit ground. The IC also needs four capacitors to work; these capacitors can be of any value from 1uF to 22uF.

MAX232 IC can help you to convert two logic level conversion that is you can use two Microcontrollers, but to get started we will use only one MCU and connect it to a computer, the complete circuit is to do the same is shown below. As you can see the pins (T2 out, R2 in, T2 in, R2 out) are left free since we are not using the second module.

Every microcontroller that has Serial communication capability will have a Tx pin and a Rx pin. These two pins should be connected to the T1 in (pin 10) and the R1 out (pin 13) pin respectively. This is the TTL/CMOS logic inputs, and then the converter RS232 logic signals can be obtained from the pins R1 in (pin 13) and T1 out (pin 14). You might wonder how a 5V signal is being converter to +25V signal when the IC itself is powered with +5V. This is made possible by a method called charge pump, which consists of a capacitor that charges up to provide 25V.

Interfacing of IC MAX232 with RS232.



6.2 Serial protocols: I2C, CAN, Fire wire, USB introduction & Comparison

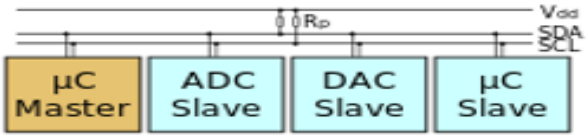
6.2.1 I²C (Inter-Integrated Circuit)

Introduction:

I²C (Inter-Integrated Circuit), pronounced I-squared-C, is a synchronous, multi-master, multi-slave, packet switched, single ended, serial computer bus invented in 1982 by Philips Semiconductor (now NXP Semiconductors). It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. Alternatively I²C is spelled I2C (pronounced I-two-C) or IIC (pronounced I-I-C).

I²C is appropriate for peripherals where simplicity and low manufacturing cost are more important than speed. Common applications of the I²C bus are:

Describing connectable devices via small ROM configuration tables to enable "plug and play" operation, such as Serial Presence Detect (SPD) EEPROMs on dual in-line memory modules (DIMMs), and Extended Display Identification Data (EDID) for monitors via VGA, DVI and HDMI connectors.



An example schematic with one master (a microcontroller), three slave nodes (an ADC, a DAC, and a microcontroller), and pull-up resistors R_p

Applications

- Accessing real-time clocks and NVRAM chips that keep user settings.
- Accessing low-speed DACs and ADCs.
- Changing contrast, hue, and color balance settings in monitors (via Display Data Channel). Changing sound volume in intelligent speakers.
- Controlling small (e.g. feature phone) OLED or LCD displays.
- Reading hardware monitors and diagnostic sensors, like a CPU thermistor or fan speed. Turning on and turning off the power supply of system components.

I²C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V, although systems with other voltages are permitted.

The I²C reference design has a 7-bit address space, with a rarely-used 10-bit extension. Common I²C bus speeds are the 100 kbit/s standard mode and the 400 kbit/s Fast mode. There is also a 10 kbit/s low-speed mode, but arbitrarily low clock frequencies are also allowed. Recent revisions of I²C can host more nodes and run at faster speeds (400 kbit/s Fast mode, 1 Mbit/s Fast mode plus or Fm+, and 3.4 Mbit/s High Speed mode). These speeds are more widely used on embedded systems than on PCs.

Note the bit rates are quoted for the transfers between master and slave without clock stretching or other hardware overhead. Protocol overheads include a slave address and perhaps a register address within the slave device, as well as per-byte ACK/NACK bits. Thus the actual transfer rate of user data is lower than those peak bit rates alone would imply. For example, if each interaction with a slave inefficiently allows only 1 byte of data to be transferred, the data rate will be less than half the peak bit rate.

The maximal 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. The relatively high impedance and low noise immunity requires a common ground potential, which again restricts practical use to communication within the same PC board or small system of boards.

The aforementioned reference design is a bus with a clock (SCL) and data (SDA) lines with 7-bit addressing. The bus has two roles for nodes: master and slave:

Master node – node that generates the clock and initiates communication with slaves.

Slave node – node that receives the clock and responds when addressed by the master . The bus is a multi-master bus, which means that any number of master nodes can be present. Additionally, master and slave roles may be changed between messages (after a STOP is sent).

There may be **four potential modes** of operation for a given bus device, although most devices only use a single role and its two modes:

master transmit – master node is sending data to a slave, **master receive** – master node is receiving data from a slave, **slave transmit** – slave node is sending data to the master , **slave receive** – slave node is receiving data from the master .

In addition to 0 and 1 data bits, the I²C bus allows special START and STOP signals which act as message delimiters and are distinct from the data bits. (This is in contrast to the start

bits and stop bits used in asynchronous serial communication, which are distinguished from data bits only by their timing.)

The master is initially in master transmit mode by sending a START followed by the 7-bit address of the slave it wishes to communicate with, which is finally followed by a single bit representing whether it wishes to write (0) to or read (1) from the slave.

The address and the data bytes are sent most significant bit first. The start bit is indicated by a high-to-low transition of SDA with SCL high; the stop bit is indicated by a low-to high transition of SDA with SCL high. All other transitions of SDA take place with SCL low .

If the master wishes to write to the slave, then it repeatedly sends a byte with the slave sending an ACK bit. (In this situation, the master is in master transmit mode, and the slave is in slave receive mode.)

If the master wishes to read from the slave, then it repeatedly receives a byte from the slave, the master sending an ACK bit after every byte except the last one. (In this situation, the master is in master receive mode, and the slave is in slave transmit mode.)

An I²C transaction may consist of multiple messages. The master terminates a message with a STOP condition if this is the end of the transaction or it may send another START condition to retain control of the bus for another message (a "combined format" transaction).

I²C defines basic types of transactions, each of which begins with a START and ends with a STOP:

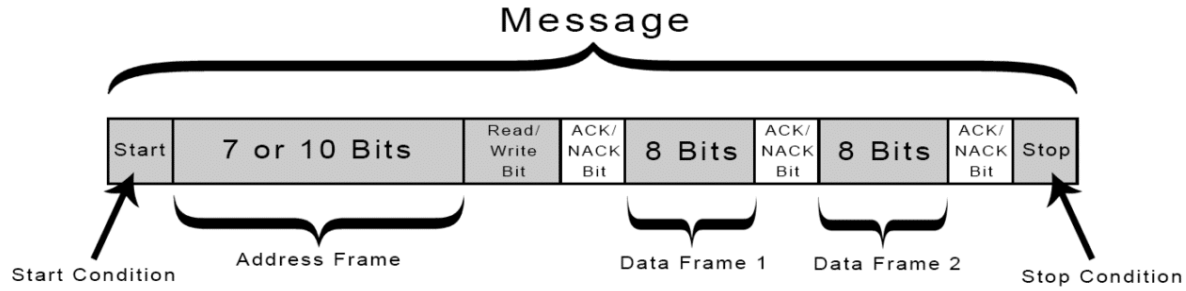
Single message where a master writes data to a slave. Single message where a master reads data from a slave. Combined format, where a master issues at least two reads or writes to one or more slaves. In a combined transaction, each read or write begins with a START and the slave address. The START conditions after the first are also called repeated START bits. Repeated STARTs are not preceded by STOP conditions, which is how slaves know that the next message is part of the same transaction.

Any given slave will only respond to certain messages, as specified in its product documentation.

Addressing structure

7-bit addressing

Field:	S	I^2C address field							R/W'	A	I^2C message sequences...	P
Type	Start	Byte 1								ACK	Byte X etc... Rest of the read or write message goes here	Stop
Bit position in byte X		7	6	5	4	3	2	1	0			
7-bit address pos		7	6	5	4	3	2	1				
Note		MSB						LSB	1 = Read 0 = Write			



Start Condition**:** The SDA line switches from a high voltage level to a low voltage level *before* the SCL line switches from high to low.

Stop Condition**:** The SDA line switches from a low voltage level to a high voltage level *after* the SCL line switches from low to high.

Address Frame**:** A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.

Read/Write Bit**:** A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).

ACK/NACK Bit**:** Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

ADDRESSING

I2C doesn't have slave select lines like SPI, so it needs another way to let the slave know that data is being sent to it, and not another slave. It does this by *addressing*. The address frame is always the first frame after the start bit in a new message.

The master sends the address of the slave it wants to communicate with to every slave connected to it. Each slave then compares the address sent from the master to its own address. If the address matches, it sends a low voltage ACK bit back to the master. If the address doesn't match, the slave does nothing and the SDA line remains high.

READ/WRITE BIT

The address frame includes a single bit at the end that informs the slave whether the master wants to write data to it or receive data from it. If the master wants to send data to the slave, the read/write bit is a low voltage level. If the master is requesting data from the slave, the bit is a high voltage level.

THE DATA FRAME

After the master detects the ACK bit from the slave, the first data frame is ready to be sent. The data frame is always 8 bits long, and sent with the most significant bit first. Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully. The ACK bit must be received by either the master or the slave (depending on who is sending the data) before the next data frame can be sent.

After all of the data frames have been sent, the master can send a stop condition to the slave to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low to high transition on the SCL line, with the SCL line remaining high.e next data frame can be sent.

https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Limitations IN I^2C

The assignment of slave addresses is one weakness of I²C. Seven bits is too few to prevent address collisions between the many thousands of available devices. What alleviates the issue of address collisions between different vendors and also allows to connect to several identical devices is that manufacturers dedicate pins that can be used to set the slave address to one of a few address options per device. Two or three pins is typical, and with many devices, there are three or more wiring options per address pin

I²C supports a limited range of speeds. Hosts supporting the multi-megabit speeds are rare. Support for the Fm+ 1 Mbit/s speed is more widespread, since its electronics are simple variants of what is used at lower speeds. Many devices do not support the 400 kbit/s speed (in part because SMBus does not yet support it). I²C nodes implemented in software (instead of dedicated hardware) may not even support the 100 kbit/s speed; so the whole range defined in the specification is rarely usable. All devices must at least partially support the highest speed used or they may spuriously detect their device address.

Devices are allowed to stretch clock cycles to suit their particular needs, which can starve bandwidth needed by faster devices and increase latencies when talking to other device addresses. Bus capacitance also places a limit on the transfer speed, especially when current sources are not used to decrease signal rise times.

Because I²C is a shared bus, there is the potential for any device to have a fault and hang the entire bus. For example, if any device holds the SDA or SCL line low, it prevents the master from sending START or STOP commands to reset the bus. Thus it is common for designs to include a reset signal that provides an external method of resetting the bus devices. However many devices do not have a dedicated reset pin, forcing the designer to put in circuitry to allow devices to be power-cycled if they need to be reset.

6.2.2 Controller Area Network (CAN)

Introduction: The CAN bus was developed by BOSCH (1) as a multi-master, message broadcast system that specifies a maximum signaling rate of 1 megabit per second (bps). The CAN Standard CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for high immunity to electrical interference and the ability to self-diagnose and repair data errors. These features have led to CAN's popularity in a variety of industries including building automation, medical, and manufacturing. The CAN communications protocol, ISO-11898: 2003, describes how information is passed between devices on a network and conforms to the Open Systems Interconnection (OSI) model. that is defined in terms of layers. Actual communication between devices connected by the physical medium is defined by the physical layer of the model. The ISO 11898 architecture defines the lowest two layers of the seven layer OSI/ISO model as the data-link layer and physical layer in Figure 1.

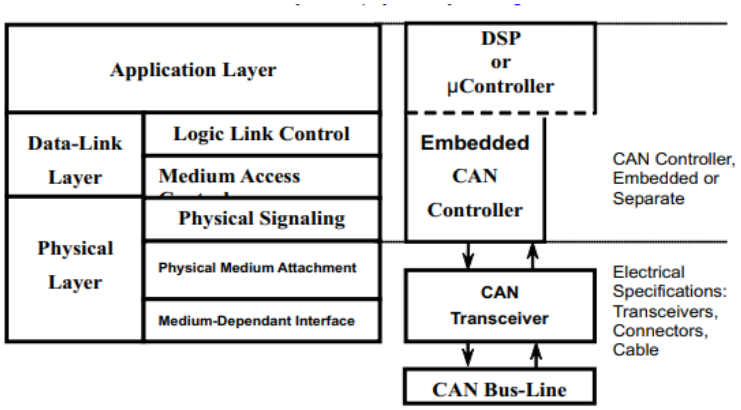


Figure 1. The Layered ISO 11898 Standard Architecture

Standard CAN or Extended CAN: The CAN communication protocol is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP). CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message. CD+AMP means that collisions are resolved through a bit-wise arbitration, based on a preprogrammed priority of each message in the identifier field of a message. The higher priority identifier always wins bus access. That is, the last logic high in the identifier keeps on transmitting because it is the highest priority. Since every node on a bus takes part in writing every bit "as it is being written," an arbitrating node knows if it placed the logic-high bit on the bus. The ISO-11898:2003 Standard, with the standard 11-bit identifier, provides for signaling rates from 125 kbps to 1 Mbps. The standard was later amended with the "extended" 29-bit identifier. The standard 11-bit identifier field in Figure 2 provides 2048 different message identifiers, whereas the extended 29- bit identifier in Figure 3 provides for 537 million identifiers.

The Bit Fields of Standard CAN and Extended CAN

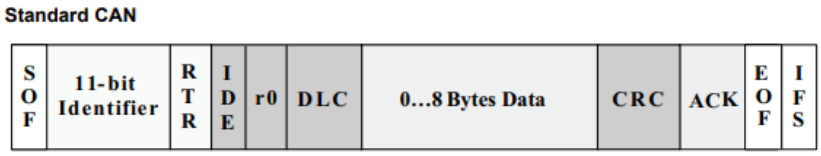


Figure 2. Standard CAN: 11-Bit Identifier

The meaning of the bit fields of Figure 2 are:

• SOF–The single dominant start of frame (SOF) bit marks the start of a message, and is used to synchronize the nodes on a bus after being idle.

• Identifier-The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.

• RTR–The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way, all data being used in a system is uniform.

• IDE–A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.

• r0–Reserved bit (for possible use by future standard amendment).

• DLC–The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.

• Data–Up to 64 bits of application data may be transmitted.

• CRC–The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.

• ACK–Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message and the sending node repeats the message after rearbitration. In this way, each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgment bit and the second is a delimiter.

• EOF–This end-of-frame (EOF), 7-bit field marks the end of a CAN frame (message) and disables bitstuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.

• IFS–This 7-bit interframe space (IFS) contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area.

3.1.2 Extended CAN

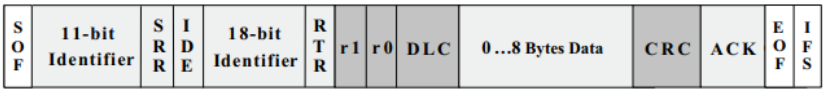


Figure 3. Extended CAN: 29-Bit Identifier

As shown in Figure 3, the Extended CAN message is the same as the Standard message with the addition of:

• SRR–The substitute remote request (SRR) bit replaces the RTR bit in the standard message location as a placeholder in the extended format.

• IDE–A recessive bit in the identifier extension (IDE) indicates that more identifier bits follow. The 18-bit extension follows IDE.

• r1–Following the RTR and r0 bits, an additional reserve bit has been included ahead of the DLC bit

Connection to the physical medium is implemented through a line transceiver

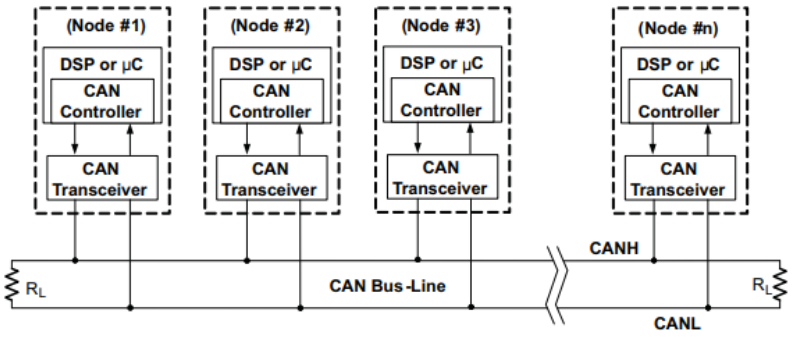


Figure 6. Details of a CAN Bus

Signaling is differential which is where CAN derives its robust noise immunity and fault tolerance. Balanced differential signaling reduces noise coupling and allows for high signaling rates over twisted-pair cable. Balanced means that the current flowing in each signal line is equal but opposite in direction, resulting in a field-canceling effect that is a key to low noise emissions. The use of balanced differential receivers and twisted-pair cabling enhance the common-mode rejection and high noise immunity of a CAN bus. The High-Speed ISO 11898 Standard specifications are given for a maximum signaling rate of 1 Mbps with a bus length of 40 m with a maximum of 30 nodes. The ISO 11898 Standard defines a single line of twisted-pair cable as the network topology as shown in Figure 6, terminated at both ends with 120-Ω resistors, which match the characteristic impedance of the line to prevent signal reflections.

The CAN standard defines a communication network that links all the nodes connected to a bus and enables them to talk with one another. There may or may not be a central control node, and nodes may be added at any time, even while the network is operating (hot-plugging).

6.2.3 Universal Serial Bus (USB)

Introduction : USB is an industry standard that establishes specifications for cables and connectors and protocols for connection, communication and power supply between computers, peripheral devices and other computers. Released in 1996, the USB standard is currently maintained by the USB Implementers Forum (USB-IF). There have been four generations of USB specifications: USB 1.*x*, USB 2.0, USB 3.*x* and USB4.

USB Universal Serial Bus

USB, Universal Serial Bus has versions including USB 1.1, USB 2, & USB3 / 3.1 and provides convenient and effective connectivity for computer related systems.

USB, Universal Serial Bus is one of the most common interfaces for connecting a variety of peripherals to computers and providing relatively local and small levels of data transfer.

USB interfaces are found on everything from personal computers and laptops, to peripheral devices, mobile phones, cameras, flash memory sticks, back up hard-drives and very many other devices.

The Universal Serial Bus, USB provides a very simple and effective means of providing connectivity, and as a result it is very widely used.

Whilst USB provides a sufficiently fast serial data transfer mechanism for data communications, it is also possible to obtain power through the connector making it possible to power small devices via the connector.

USB evolution

The USB interface was developed as a result of the need for a communications interface that was convenient to use and one that would support the higher data rates being required within the computer and peripherals industries.

The first proper release of a USB specification was Version 0.7 of the specification. This occurred in November 1994. This was followed in January 1996 by USB 1.0. USB 1.0 was widely adopted and became the standard on many PCs as well as many printers using the standard. In addition to this a variety of other peripherals adopted the USB interface, with small memory sticks starting to appear as a convenient way for transferring or temporarily storing data.

SUMMARY OF USB VERSIONS AND PERFORMANCE	
USB VERSION	**DETAILS**
USB 1	Low speed: 1.5 Mbps, ; Full speed: 12 Mbps
USB 2	'High Speed' rate of 480 Mbps
USB 3	Raw data transfer rates of 4.8 Gbit/s

With USB 1.0 well established, faster data transfer rates were required, and accordingly a new specification, USB 2 was released. With the importance of USB already established it did not take long for the new standard to be adopted.

With USB defining its place in the market, other developments of the standard were investigated. With the need for mobility in many areas of the electronics industry taking off, the next obvious move for USB was to adopt a wireless interface. In doing this wireless USB would need to retain the same flexible approach that provided the success for the wired interface. In addition to this the wireless USB interface needs to be able to transfer data at rates which will be higher than those currently attainable with the wired USB 2 connections. To achieve this ultra-wideband UWB technology is used.

USB capabilities

The basic concept of USB was for an interface that would be able to connect a variety of computer peripheral devices, such as keyboards and mice, to PCs. However, since its introduction, the applications for USB have widened and it has been used for many other purposes including, including measurement and automation.

In terms of performance, USB 1.1 enabled a maximum throughput of 12 Mbps, but with the introduction of USB 2.0 the maximum speed is 480 Mbps.

In operation, the USB host automatically detects when a new device has been added. It then requests identification from the device and appropriately configures the drivers. The bus topology allows up to 127 devices to run concurrently on one port. Conversely, the

classic serial port supports a single device on each port. By adding hubs, more ports can be added to a USB host, creating connections for more peripherals.

USB advantages & disadvantages

USB has many advantages when compared to other technologies, but it also has a number of disadvantages which need to be considered when deciding on a technology to be used.

ADVANTAGES & DISADVANTAGES OF USB	
Advantages • Ease of use • Acceptable data rate for many applications • Robust connector system • Variety of connector types / sizes available • Low cost	**Disadvantages** • Data transfer not as fast as some other systems • Limited capability & overall performance

USB has many advantages and this is why it is so widely used. However, its simplicity and ease of use, mean that it is not always applicable in applications where more sophisticated interfaces are required for very high speed data transfer.

Wired USB standards

- ***USB1.1:*** This was the original version of the USB, Universal Serial Bus and was released in September 1998 after a few problems with the USB 1.0 specification released in January 1996 were resolved.. It provided a Master / Slave interface and a tiered star topology which was capable of supporting up to 127 devices and a maximum of six tiers or hubs. The master or "Host" device is normally a PC with the slaves or "Devices" linked via the cable.

  One of the aims of the USB standard was to minimise the complexity within the Device by enabling the Host to perform the processing. This meant that devices would be cheap and readily accessible.
  The data transfer rates of USB 1.1 are defined as:

  - *Low speed:* 1.5 Mbps
  - *Full speed:* 12 Mbps

  The cable length for USB 1.1 is limited to 5 metres, and the power consumption specification allows each device to take up to 500mA, although this is limited to 100mA during start-up.
  USB 1.1 does not allow extension cables or the inclusion of pass-through monitors (due to timing and power limitations).

  USB 2.0*:* The USB 2.0 standard is a development of USB 1.1 which was released in April 2000. The main difference when compared to USB 1.1 was the data transfer speed increase up to a "High Speed" rate of 480 Mbps. However it should be noted that even though devices are labelled USB 2.0, they may not be able to meet the full transfer speed.

USB 3.0*:* This improved USB standard which was first demonstrated at the Intel Developer Forum in September 2007. The major feature is what is termed the SuperSpeed bus, which provides a fourth transfer mode which gives data transfer rates of 4.8 Gbit/s. Although the raw throughput is 4 Gbit/s, data transfer rates of 3.2 Gbit/s, i.e.0.4 GByte/s more after protocol overhead are deemed acceptable within the standard.
Often USB ports on computers, etc. may have the USB symbol with 'SS' added, i.e. SS USB. SS USB denotes USB 3, i.e. Super Speed USB.

Firewire : Along with USB, Firewire (also called IEEE 1394) is another popular connector for adding peripherals to your computer. Firewire is most often used to connect digital camcorders, external hard drives, and other devices that can benefit from the high transfer rates (up to 480 Mbps) supported by the Firewire connection. The iSight camera used for chatting on the Mac connects using a Firewire cable. In addition to connecting peripherals such as camcorders or external hard drives, Firewire can be used to connect two computers to transfer files.

Firewire has the advantage of being able to transfer power to the device through the same cable that does the data transfer. A disadvantage of Firewire is that cables tend to be more expensive.

Firewire was originally developed by Apple and comes standard on many Macintosh computers.

6.2.3 Parallel protocols–PCI bus, PCI-X bus, introduction & comparison

The name **PCI** has been derived from **P**eripheral **C**omponent **I**nterconnect which describes a set of industry standard computer bus architectures which are used to connect components on the computer main board to each other, and also provides an expansion bus to install add-in cards.
These bus architectures have been fully standardized by the PCI Special Interest Group (PCI SIG) Over the years the PCI SIG has enhanced and released newer versions of their standards in order to match the increasing requirements for a high performance computer bus system.
The PCI standards include a full plug-and-play capability. This means the computer BIOS or operating system can determine the resource requirements (memory, I/O, or interrupt) of such cards automatically and assign resources in a way that there are no conflicts with other cards.

Each card is identified by a unique vendor ID assigned by the PCI SIG, and a device ID assigned by the manufacturer of a card. This lets the driver software properly identify the cards it supports.

Conventional PCI with 5V signal levels

The original PCI standard specified 5V bus signal levels since most of the processors and peripheral chips were specified for those signal levels those days. Also the first PCI cards manufactured by Meinberg were specified to operate using 5V bus signal levels.

Newer chips were designed to work with lower bus signal levels, so the PCI SIG specified a newer standard using 3.3V bus signal levels.

Conventional PCI with 3.3V signal levels

In order to increase the maximum clock speed, and not at least to decrease power consumption, the PCI standard was extended to support 3.3V bus signal levels.
In order to prevent installation of 5V-only cards into 3.3V slots and vice-versa, the slot connectors are keyed so that cards can only be inserted into slots with the corresponding signal level.
Many PCI add-in cards can work with either 3.3V or 5V signal levels, so they are keyed for either of the slot types. All current versions of Meinberg PCI cards are keyed this way, so all those cards can be installed in either a 3.3V slot or a 5V slot.
Conventional PCI slots use a data width of 32 or 64 bits, and a maximum bus clock up to 33 or 66 MHz. However, as the maximum data transfer rate which can be achieved via this conventional PCI bus didn't meet the requirements of modern computer systems anymore, the PCI SIG released a follow-up standard called PCI-X. The orignal 5V or 3.3V versions of the PCI bus are thus often referred to as "Conventional PCI" which supports data widts of 32 or 64 bit and a bus clock of up to 33 or 66 MHz.

PCI-X

PCI-X is an approach to increase the maximum transfer rate beyond the maximum rate achievable by conventional PCI. Since this is mostly a requirement for expansion cards used in servers, e.g. network or hard disk controllers, PCI-X slots can mainly be found on special server main boards.
The maximum bus clock of a PCI-X v1.0 slot is 133 MHz, and those slots support 3.3V signal levels only. However, the PCI-X bus specification is backward compatible with the conventional 3.3V PCI specs, so conventional 3.3V PCI cards which support up to 66 MHz bus clock can be installed in a PCI-X slot. Since all current PCI cards manufactured by Meinberg meet these requirements, all current types of Meinberg PCI cards can be installed in PCI-X slots. However, it is not possible to operate 5V-only add-in cards in a PCI-X slot.
Note:
If a conventional PCI card is installed in a PCI-X slot then the clock speed of all PCI-X slots connected to the same bus is reduced to the maximum clock speed supported by that add-in card. Real PCI-X cards benefit from a high transfer rate provided by the PCI-X bus, so they should preferably run with the maximum rather than a decreased bus clock speed.
However, due to electrical limitations on the length of the wires between the slots the PCI-X specs allow only up to two slots to be connected to a single bus. So if there are more PCI-X slots available then they are normally connected to several independent buses, each bus with 2 slots.

This is why the limited transfer rate due to a conventional PCI card in a PCI-X slot normally affects only one other slot, usually the next one. Other slots won't be affected since they are physically connected to different buses. The manual of the computer main board should mention which of the PCI-X slots are grouped together to a single bus.