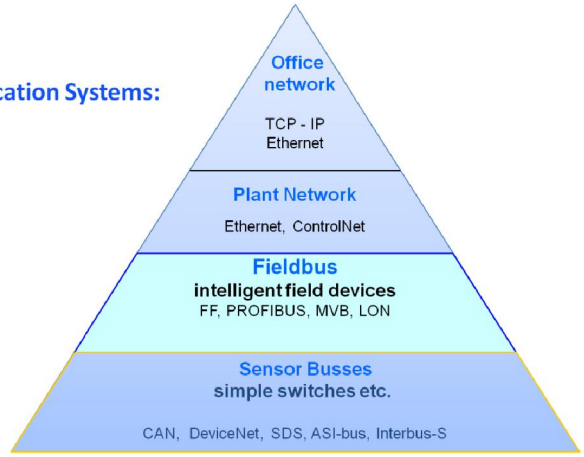
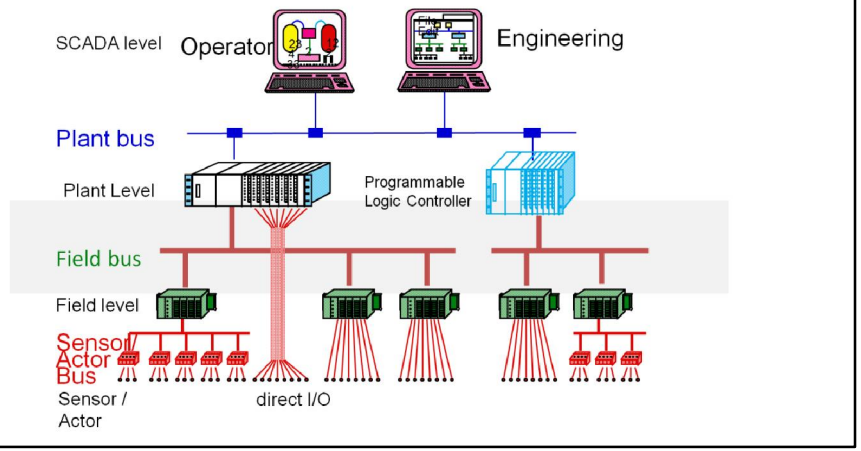
INDUSTRIAL NETWORKS - FIELDBUS

Industrial Communication Systems:

Field Bus
Buses de terreno
Bus de terrain
Feldbusse



Location of the field bus in the plant hierarchy



What is fieldbus?

A data network, interconnecting an automation system, characterized by:

- many small data items (process variables) with bounded delay (1ms..1s)
- transmission of non-real-time traffic for commissioning and diagnostics
- harsh environment (temperature, vibrations, EM-disturbances, water, salt,…)
- robust and easy installation by skilled people
- high integrity (no undetected errors) and high availability (redundant layout)
- intrinsic safety (for oil & gas, mining, chemicals,..)

clock synchronization (milliseconds to microseconds)

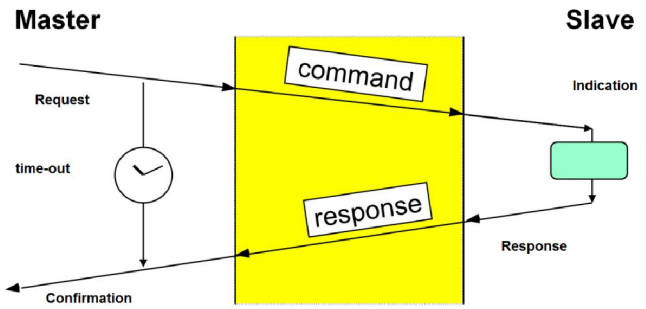
- continuous supervision and diagnostics
- low attachment costs (€ 5.- .. €50 / node)
- moderate data rates (50 kbit/s - 5 Mbit/s), large distance range (10m - 4 km)

requirements from PLCs carry over to the field bus as well (real-time constraints, harsh environments, high integrity and availability). In addition synchronization is necessary to assign precise timestamps to data, attachment cost should be kept low. While data rates are moderate, large distances need to be covered (depending on plant type).

Sensor bus

- Information connection of sensors/actuators with control system at the lowest level
- Modern sensors uses often microcontroller ⇒ digital communication bus
- Typical characteristics
  - Short distance (usually up to tens of meters)
  - Transfer rate: 100b - 10Mb/s
  - Small data blocks, asynchronous transfer, irregular data
  - Simple and low-cost cabling
  - Modern sensor bus = Field bus

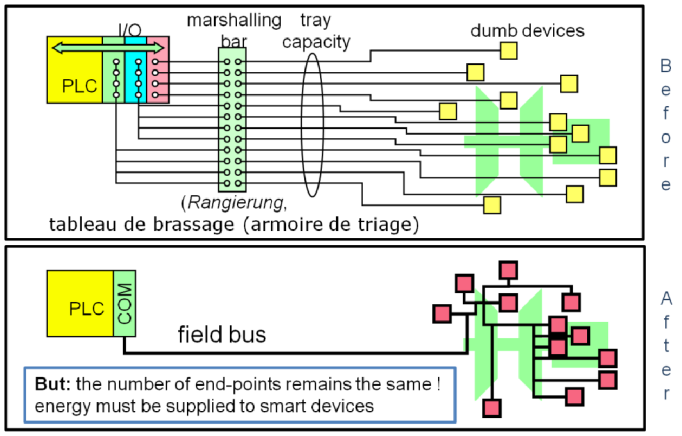
Mater - slave



Expectations

- reduce cabling
- increased modularity of plant (each object comes with its computer)
- easy fault location and maintenance
- simplify commissioning (mise en service, IBS = Inbetriebssetzung)
- simplify extension and retrofit
- off-the-shelf standard products to build “Lego”-control systems

The original idea: save wiring

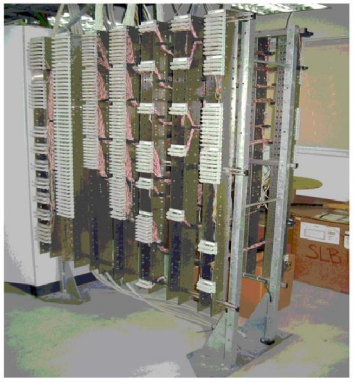


Marshalling (Rangierschiene, Barre de rangement)

The marshalling is the interface between the PLC people and the instrumentation people.

The fieldbus replaces the marshalling bar or rather moves it piecewise to the process

(intelligent concentrator / wiring)

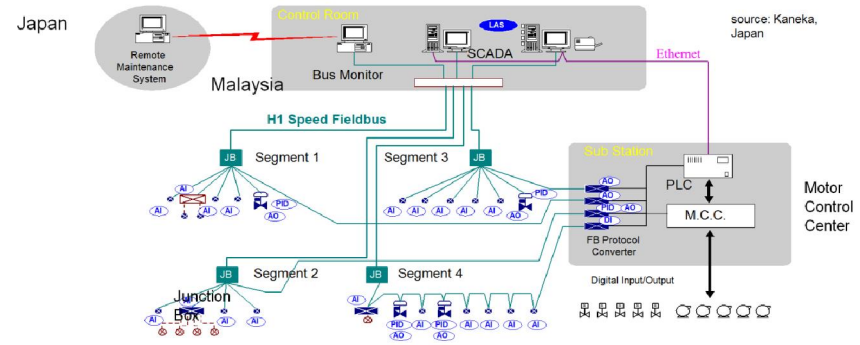


Geographical extension of industrial plants

The field bus requirements follow the physical extension of the plant

1 km .. 1000 km	**Transmission & Distribution** Control and supervision of large distribution networks: • water - gas - oil - electricity - ...
1 km .. 5 km	**Power Generation** Out of primary energy sources: • waterfalls - coal - gas - oil - nuclear - solar - ...
50 m .. 3 km	**Industrial Plants** Manufacturing and transformation plants: • cement works - steel works - food silos - printing - paper pulp processing - glass plants - harbors - ...
500m .. 2 km	**Building Automation** • energy - air conditioning - fire - intrusion - repair - ...
1 m .. 1 km	**Manufacturing** flexible manufacturing cells - robots
1 m .. 800 m	**Vehicles** • locomotives - trains - streetcars - trolley buses - vans - buses - cars - airplanes - spacecraft - ...

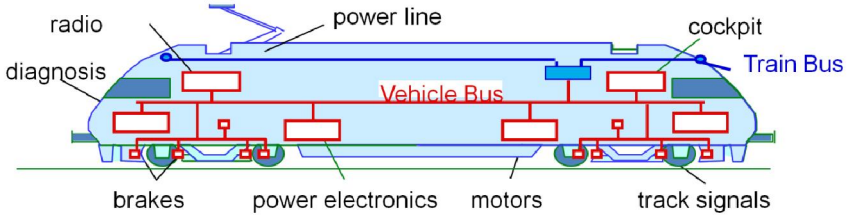
Fieldbus over a wide area: example wastewater treatment



Numerous analog inputs/outputs (AI/AO),
low speed (37 kbit/s) segments (Hart) merged to 1 Mbit/s links (H1 Speed Fieldbus).

plant has demanding requirements with respect to large distances and need cope with explosives. A hierarchical structure with H1 Speed Fieldbus on the high level and Hart Fieldbus on the lower levels is used for the water treatment part. The lower levels and higher level are combined with junction box gateways. On the other hand the electrical grid part (substation) uses Ethernet. For the interconnecting of these two parts fieldbus protocol converters are used.

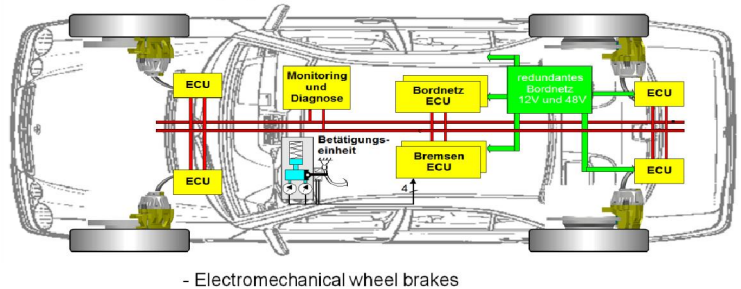
Fieldbus Application: locomotives and drives



data rate	1.5 Mbit/second
delay	1 ms (16 ms for skip/slip control)
medium	twisted wire pair, optical fibers (EM disturbances)
number of stations	up to 255 programmable stations, 4096 simple I/O
integrity	very high (signaling tasks)
cost	engineering costs dominate

. Each car of a train has its own vehicle bus to control the brakes, power electronics and motors and gather track signals. In addition there is a train bus that connects the different cars.

Fieldbus Application: automobile



- Electromechanical wheel brakes
- Redundant Engine Control Units
- Pedal simulator
- Fault-tolerant 2-voltage on-board power supply
- Diagnostic System

DAIMLERBENZ
AKTIENGESELLSCHAFT

The communication network of a car is redundant, there are several engine control units and separate cables connecting them. This allows a highly available and safe operation.

Which field bus ?

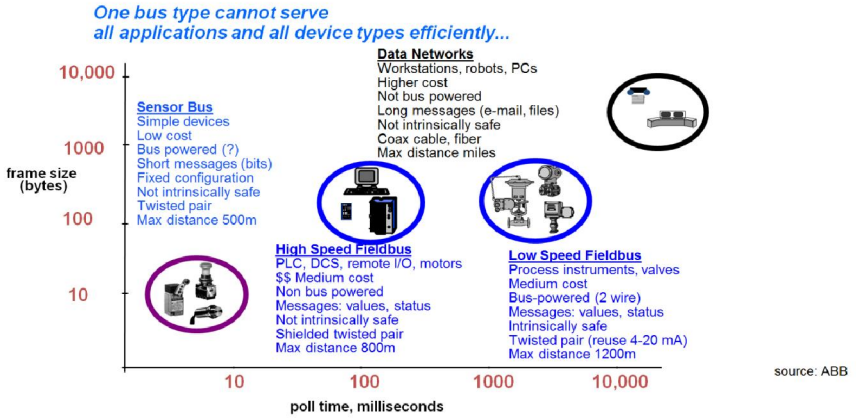
- A-bus
- Arcnet
- Arinc 625
* • ASI
- Batibus
- Bitbus
* • CAN
- ControlNet
- DeviceNet
- DIN V 43322
- DIN 66348 (Meßbus)
- FAIS
- EIB
- Ethernet
- Factor
- Fieldbus Foundation
- FIP
- Hart
- IEC 61158
- IEEE 1118 (Bitbus)
- Instabus
* • Interbus-S
- ISA SP50
- IsiBus
- IHS
- ISP
- J-1708
- J-1850
- LAC
* • LON
- MAP
- Master FB
- MB90
- MIL 1553
- MODBUS
* • MVB
- P13/42
- P14
- Partnerbus
- P-net
* • Profibus-FMS
- Profibus-PA
- Profibus-DP
- PDV
* • SERCOS
- SDS
- Sigma-i
- Sinec H1
- Sinec L1
- Spabus
- Suconet
- VAN
- WorldFIP
- ZB10
- ...

Worldwide most popular field busses

Bus	User*	Application	Sponsor
CANs	25%	Automotive, Process control	CiA, OVDA, Honeywell
Profibus (3 kinds)	26%	Process control	Siemens, ABB
LON	6%	Building systems	Echelon, ABB
Ethernet	50%	Plant bus	all
Interbus-S	7%	Manufacturing	Phoenix Contact
Fieldbus Foundation, HART	7%	Chemical Industry	Fisher-Rosemount, ABB
ASI	9%	Building Systems	Siemens
Modbus	22%	obsolete point-to-point	many
ControlNet	14%	plant bus	Rockwell

Sum > 100%, since firms support more than one bus

Different classes of field busses



SERIAL INTERFACES RSXXX

RS232/422/423/485

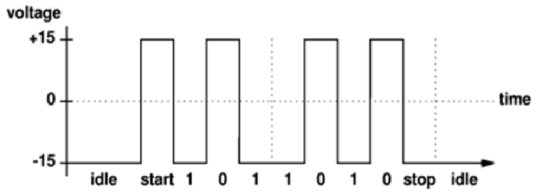
- Serial data transfer: bit by bit
- RSxxx defined at physical layer
- Backbone of many fieldbus (mainly RS485)
- Fieldbuses define higher layers (protocol, data coding, interpretation, …)

RS232

- Originally for connection of modem to computer
  - The Computer Terminal Equipment (CTE - modem) and the Data Terminal Equipment (DTE = PC).
- Today: computer to device (null modem)
- Basic characteristics
  - The connection must be no longer than 50 feet.
  - Maximal data rate 19200baud (even more than 100kbaud in some applications).
    - Typical data rates: 75, 110, 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 33600, 56000, 115000 and (rarely) 330000 baud.
  - Signaling unbalanced, negative logic
    - 2 unidirectional lines for data transfer (TxD, RxD) + GND,
    - Additional lines for data transfer control (optional)
  - Asynchronous data transfer (STAR and STOP bits)
  - Point to point

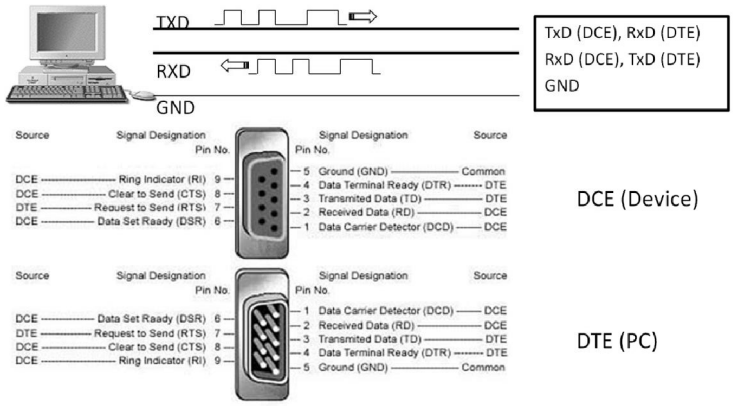
RS232 frame

- Every RS-232 frame consists of:
  - 1 start bit (log. 0)
  - 5 - 8 data bits (optional)
  - (optional 1 parity bit)
  - 1+ stop bits (log. 0)



log. 1: -15V - -3V
log. 0: +3V - +15V

RS232 3-wire connections



TxD (DCE), RxD (DTE)
RxD (DCE), TxD (DTE)
GND

DCE (Device)

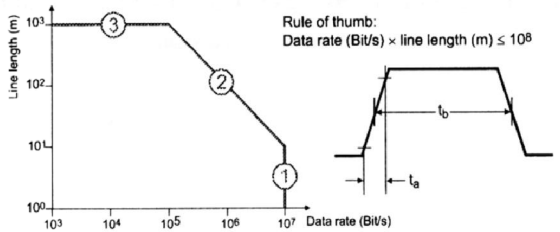
DTE (PC)

Handshake

- Handshaking is the process of ensuring that data not be transmitted when the receiver is not ready and to ensure error free transmission.
- Receivers pause communication when data buffer is full
- Handshaking can be either hardware or software
  - Hardware: DTR/DSR, CTS/RTS
  - Software: Xon/Xoff (codes send trough data lines)
- Handshaking may not be required if the amount of data is small.

RS 422, RS 485

- Newer standards
  - Data rate up to 10Mb/s
  - Distance up to 1200m
  - RS422: one transmitter, 10 receivers
  - RS485: 32 transmitter, 10 receivers (network)



Rule of thumb:
Data rate (Bit/s) × line length (m) ≤ 10^8

① Data rate is limited by the rise time of the output signal of the generator (ITU-T V.11: $t_r \leq 0.1 \times t_b$; ISO8482: $t_r \leq 0{,}3 \times t_b$).

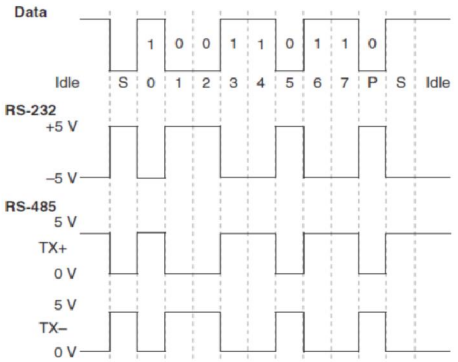
② Data rate is limited due to signal distortion caused by losses on the transmission line (e.g. skin effect).

③ Data rate is limited due to the attenuation caused by the ohmic resistance of the wires (a 1000 m twisted pair cable with wires of 0.6 mm diameter has a resistance of ≈ 100 Ω).

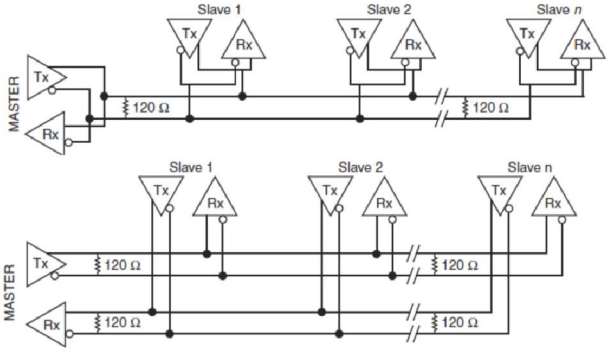
Signaling

RS422 & RS485:

- Balanced signaling (TxD+, TxD-, RxD+, RxD-, alternatively A, B)
- 423: unbalance signaling (TxD+, RxD+, GND from RS422)



Full and half duplex RS 485



RSxxx

	RS232	RS422	RS485
Cabling	single ended	single ended multi-drop	multi-drop
Number of Devices	1 transmit 1 receive	1 transmitter 10 receivers	32 transmitters 32 receivers
Communication Mode	full duplex	full duplex half duplex	full duplex half duplex
Max. Distance	50 feet at 19.2 Kbps	4000 feet at 100 Kbps	4000 feet at 100 Kbps
Max. Data Rate	19.2 Kbps for 50 feet	10 Mpbs for 50 feet	10 Mpbs for 50 feet
Signaling	unbalanced	balanced	balanced
Mark (data 1)	-5 V min. -15 V max.	2 V min. (B>A) 6 V max. (B>A)	1.5 V min. (B>A) 5 V max. (B>A)
Space (data 0)	5 V min. 15 V max.	2 V min. (A>B) 6 V max. (A>B)	1.5 V min. (A>B) 5 V max. (A>B)
Input Level Min.	+/- 3 V	0.2 V difference	0.2 V difference
Output Current (short circuit)	500 mA *(Note that the driver ICs normally used in PCs are limited to 10 mA)*	150 mA	250 mA

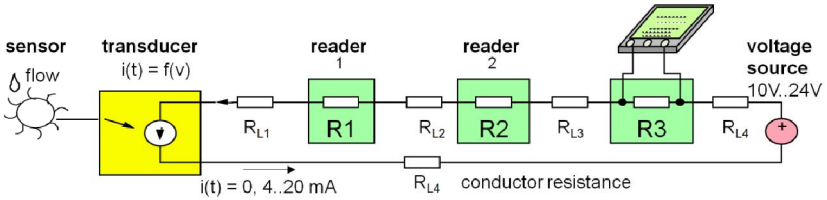
Serial Interfaces are probably the most often used interfaces in DAQ systems especially in the industrial systems (see the relevant lecture of prof. Grimaldi). Here we only shortly remind some basic parameters which enable comparison of some RSxxx interfaces. So, what the heck does the "RS" stand for? It means Recommended Standard. However, today RS-485 is formally known as TIA/EIA-485, for the associations that oversee the specifications for electronics and telecommunications standards. You can find also other RSxxx as telecommunication standards by ITU (former CCITT, e.g. RS 422 = ITU-T V.11, RS232 = V.24 + V.28) or ISO (RS 485 = ISO 8482).

Here is the short version of the critical specifications. Unfortunately, these are subject to interpretation by individual manufacturers. That is why RS232 is often regarded as an incredibly non-standard communications protocol.

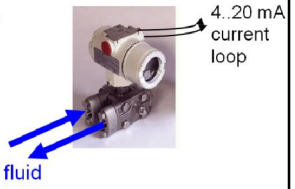
One important note. You will see that one of the major differences between RS232 and RS422/RS485 is the signaling mode. RS232 is unbalanced while RS422/RS485 is balanced. An unbalanced signal is represented by a single signal wire where a voltage level on that one wire is used to transmit/receive binary 1 and 0: this can be considered a push signal driver. On the other hand, a balanced signal is represented by a pair of wires where a voltage difference is used to transmit/receive binary information: sort of a push-pull signal driver. In short, unbalanced voltage level signal travels slower and shorter than a balanced voltage difference signal.

4-20 mA loop - the conventional, analog standard (recall)

The 4-20 mA is the most common analog transmission standard in industry

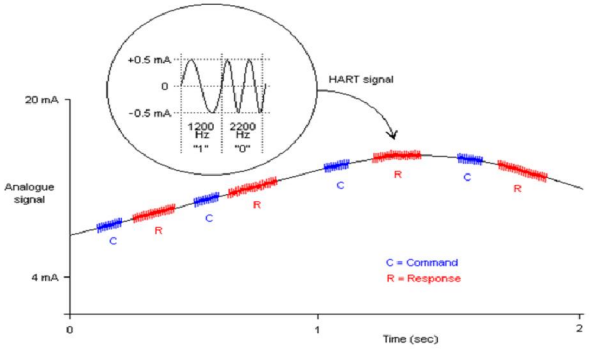


- The transducer limits the current to a value between 4 mA and 20 mA, proportional to the measured value, while 0mA signals an error (wire break)
- The voltage drop along the cable and the number of readers induces no error.
- Simple devices are powered directly by the residual current (4mA), allowing to transmit signal **and** power through a single pair of wires.
- Remember: 4-20mA is basically a point-to-point communication (one source)



HART - Principle

HART (Highway Addressable Remote Transducer) was developed by Fisher-Rosemount to retrofit 4-to-20mA current loop transducers with digital data communication.



Transmission of device characteristics is normally not real-time critical

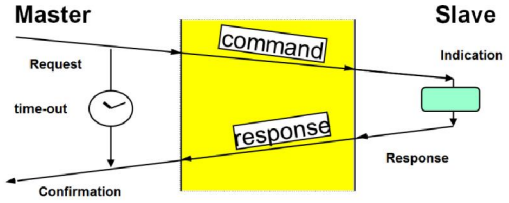
HART modulates the 4-20mA current with a low-level frequency-shift-keyed (FSK) sine-wave signal, without affecting the average analogue signal.

HART uses low frequencies (1200Hz and 2200 Hz) to deal with poor cabling, its rate is 1200 Bd - but sufficient.

HART uses Bell 202 modem technology, ADSL technology was not available in 1989, at the time HART was designed

HART - Protocol

Hart communicates point-to-point, under the control of a master, e.g. a hand-held device



Hart frame format (character-oriented):

preamble	start	address	command	bytecount	[status]	data	data	checksum
5..20 (xFF)	1	1..5	1	1	[2] (slave response)	0..25 (recommended)		1

Profibus I

- Origin: German Government in cooperation with automation manufacturers, 1989
- Implemented on ASIC chips produced by multiple vendors. Based on RS485 and the European EN50170 Electrical specification.
- Formats: Profibus DP (Master/Slave), Profibus FMS (Multi- master/Peer to Peer), and Profibus PA (intrinsically safe).
- Connectors: 9-Pin D-Shell connector (impedance terminated) or 12mm IP67 quick-disconnect.
- Maximum Number of Nodes: 127
- Distance: 100M to 24 KM (with repeaters and fiber optic transmission)
- Baudrate: 9600 to 12M Bit/sec
- Message size: up to 244 bytes of data per node per message
- Messaging formats: Polling (DP/PA) and Peer-to-Peer (FMS)
- Supporting Trade Organization: Profibus Trade Organization, www.profibus.com.

Profibus II

- Profibus is commonly found in Process Control and large assembly, and material handling machines.
- Advantages:
  - Profibus is the most widely accepted international networking standard especially in Europe
- Disadvantages:
  - High overhead to message ratio for small amounts of data;
  - no power on the bus;
  - slightly higher cost than some other buses;

Profibus III

Pros:

- Standardized by CENELEC (EN 50 170-3)
- Wide support by Siemens (Profibus DP is backbone of Simatic S7) and active Profibus User Organization (PNO) with >1000 companies.
- 200,000 applications, > 2 Mio devices
- Low entry price (originally simple UART protocol at 500 kbit/s with RS 485 drivers)
- Several implementations based on most commons processors and micro controllers (8051, NEC V25, 80186, 68302).
- Development tools available (Softing, I-tec).
- Market: industry automation

Cons:

- Exists in four incompatible versions (FMS, DP, PA, 12 Mbit/s), evolving specifications.
- Most products do not implement all the Profibus functionality, interoperability is questionable outside of one manufacturer
- Additional protocols exist within Siemens
- Weak physical layer (RS 485 at 1,5 Mb/s); to remedy this, a 12 Mb/s version has been developed (does not significantly improve response time, but limits distance).
- Complex configuration - all connections must be set up beforehand (except network management): tools required.
- Little used outside of Europe (identified in USA / Asia with Siemens/Germany)

CAN

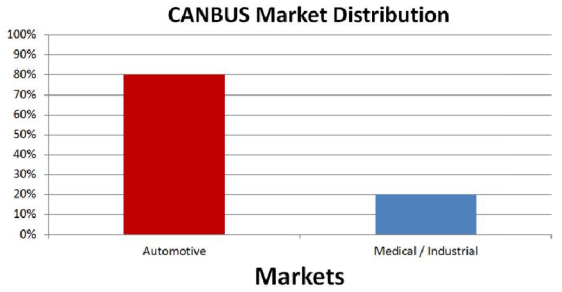
CANBUS or CAN bus – **C**ontroller **A**rea **N**etwork **bus**
An automotive serial bus system developed to satisfy the following requirements:

- Network multiple microcontrollers with 1 pair of wires.
- Allow microcontrollers communicate with each other.
- High speed, real-time communication.
- Provide noise immunity in an electrically noisy environment.
- Low cost

33

Who uses CANBUS?

- Designed specifically for automotive applications
- Today - industrial automation / medical equipment



34

CANBUS History

- First idea - The idea of CAN was first conceived by engineers at Robert Bosch Gmbh in Germany in the early 1980s.
- Early focus - develop a communication system between a number of ECUs (electronic control units).
- New standard - none of the communication protocols at that time met the specific requirements for speed and reliability so the engineers developed their own standard.

35

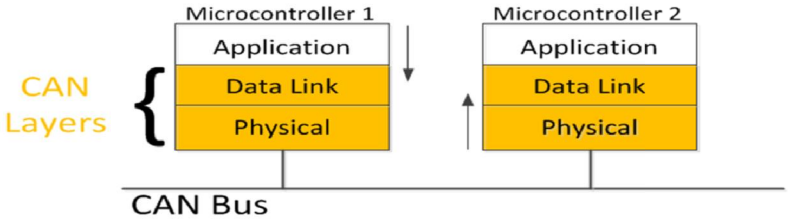
CANBUS Timeline

- 1983 : First CANBUS project at Bosch
- 1986 : CAN protocol introduced
- 1987 : First CAN controller chips sold
- 1991 : CAN 2.0A specification published
- 1992 : Mercedes-Benz used CAN network
- 1993 : ISO 11898 standard
- 1995 : ISO 11898 amendment
- Present : The majority of vehicles use CAN bus.

36

CANBUS and the OSI Model

- CAN is a closed network
  - – no need for security, sessions or logins.
  - - no user interface requirements.
- Physical and Data Link layers in silicon.



37

CANBUS Physical Layer

- Physical medium – two wires terminated at both ends by resistors.
- Differential signal - better noise immunity.
- Benefits:
  - Reduced weight, Reduced cost
  - Fewer wires = Increased reliability

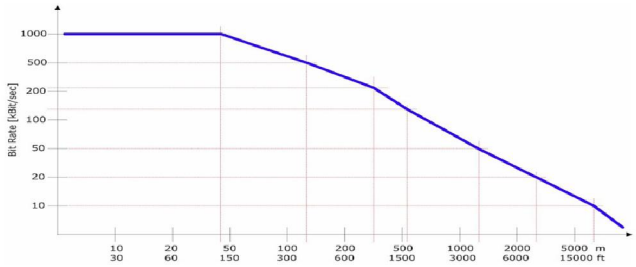
Conventional multi-wire looms

vs.

CAN bus network

Transmission Characteristics

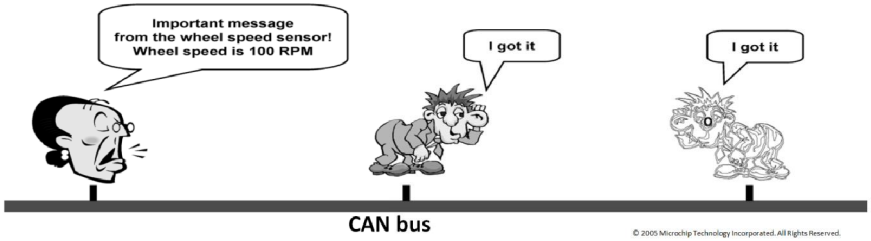
- Up to 1 Mbit/sec.
- Common baud rates: 1 MHz, 500 KHz and 125 KHz
- All nodes – same baud rate
- Max length:120' to 15000' (rate dependent)



39

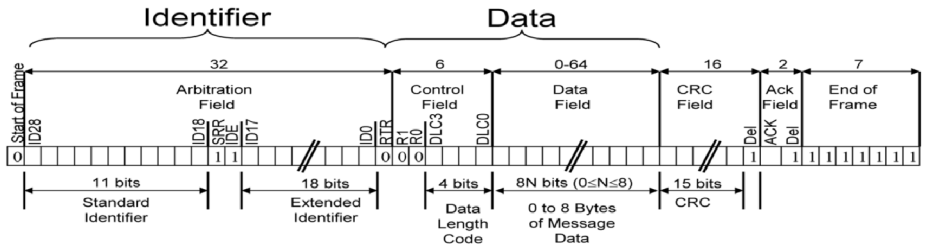
Message Oriented Transmission Protocol

- Each node – receiver & transmitter
- A sender of information transmits to all devices on the bus
- All nodes read message, then decide if it is relevant to them
- All nodes verify reception was error-free
- All nodes acknowledge reception



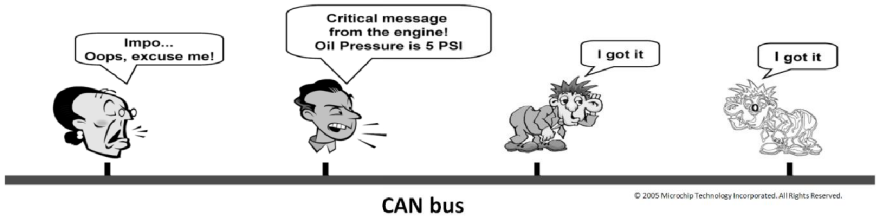
Message Format

- Each message has an ID, Data and overhead.
- Data –8 bytes max
- Overhead – start, end, CRC, ACK



Bus Arbitration

- Arbitration – needed when multiple nodes try to transmit at the same time
- Only one transmitter is allowed to transmit at a time.
- A node waits for bus to become idle
- Nodes with more important messages continue transmitting

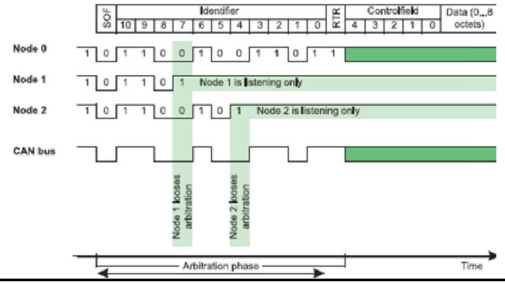


Bus Arbitration

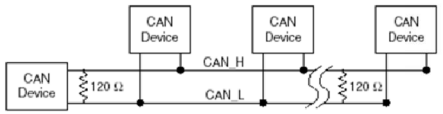
- Message importance is encoded in message ID.

  Lower value = More important

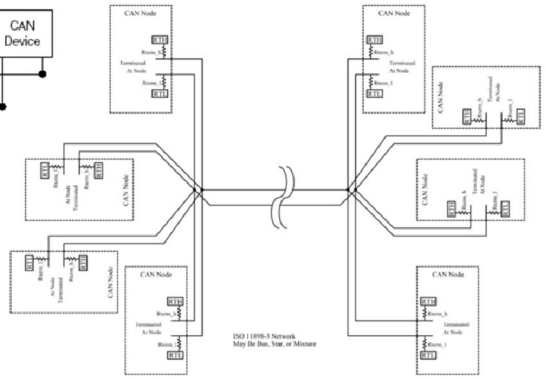
- As a node transmits each bit, it verifies that it sees the same bit value on the bus that it transmitted.
- A "0" on the bus wins over a "1" on the bus.
- Losing node stops transmitting, winner continues.



High and Low speed



- HS – impedance matched
- LS – any configuration



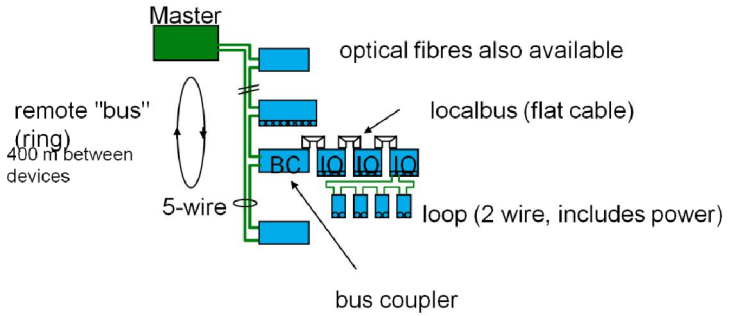
DeviceNet

- Origin: Allen-Bradley, USA, 1994
- Based on CAN (Controller Area Network) technology, borrowed from the automotive industry
- Maximum Number of Nodes: 64
- Connectors: Popular 'Mini' 18mm and 'Micro' 12mm waterproof quick- disconnect plugs and receptacles, and 5 pin phoenix terminal block.
- Distance: 100M to 500M
- Baudrate: 125, 250 and 500 Kbits/sec
- Maximum Message size: 8 bytes of data per node per message
- Supporting Trade Organization: Open DeviceNet Vendor Association, www.odva.org
- Typical Applications: Most commonly found in assembly, welding and material handling machines.
- Single-cable wiring of multi-input sensor blocks, smart sensors, pneumatic valves, barcode readers, drives and operator interfaces. DeviceNet is especially popular in automotive and semiconductor.
- Advantages: Low cost, widespread acceptance, high reliability, and efficient use of network bandwidth, power available on the network.
- Disadvantages: Limited bandwidth, limited message size and maximum length.

Interbus-S

- Origin: Phoenix Contact, 1984
- High Speed Shift Register topology
- Maximum Number of Nodes: 256
- Connectors: 9 Pin D-Shell and 23mm circular DIN; Cabling options allow for twisted pair, fiber optic and infrared connections
- Distance: 400M per segment, 12.8 KM Total
- Baudrate: 500 Kbits/sec (2Mbit also available)
- Message size: 512 bytes of data per node, unlimited block transfers
- Messaging formats: I/O scanning and PCP channel for data transfer
- Supporting Trade Organization: The Interbus Club, www.interbusclub.com

Interbus-S - Topology



Interbus-S - Analysis

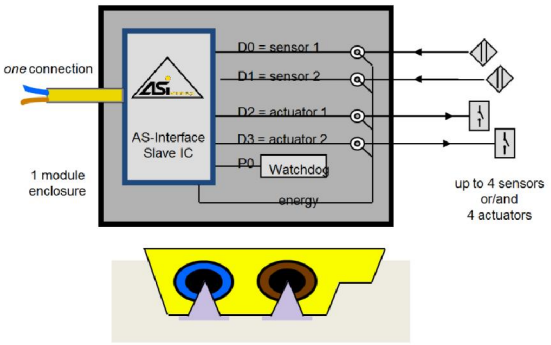
+	-
+ standard in CENELEC	– market centered on manufacturing
+ 1700 products, 270 manufacturers, 375.000 applications	– limited number of variables (4096 bits)
+ good experience in field wiring (intelligent wiring bar)	– ring structure sensitive to disruptions
+ easy to engineer	– sensitive to misplacement
+ easy to program (IEC 61131)	– clumsy and slow message service
+ far extension (400m .. 13 km)	– medium user community
+ good response time	– few and costly tools
+ conformance test	– strong ties to Phoenix Contact

AS-I (Actuator Sensor Interface)

- Origin: AS-I Consortium, 1993
- Maximum Number of Nodes: 31 slaves, 1 master
- Connectors: Insulation displacement connectors on flat yellow cable, 2 position terminal block or 12mm 'micro' quick-disconnect connectors.
- Distance: 100M, 300M with repeaters
- Baudrate: 167 Kbits/sec
- Message size: 8 bits (4 inputs, 4 outputs) per node per message
- Messaging formats: Strobing
- Supporting Trade Organization: AS-I Trade Organization, http://www.as-interface.com/
- Typical Applications: Commonly found in assembly, packaging and material handling machines. Single-cable wiring of multi-input sensor blocks, smart sensors, pneumatic valves, switches and indicators.

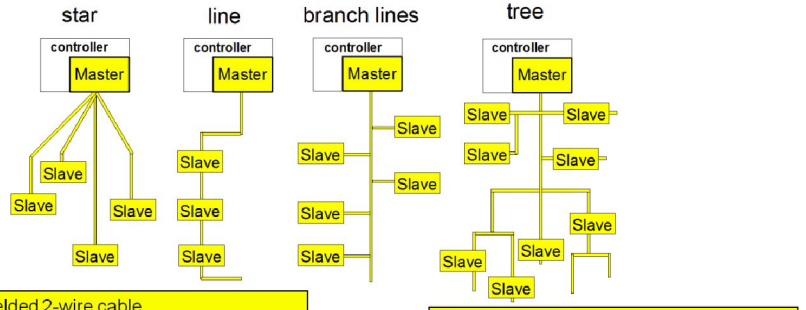
ASI - Sensor bus Wiring

Very simple sensor bus for building automation, combining power and data on the same wires, transmitting mostly binary signals



- **mechanically coded flat cable**
  - **two wires for data and power**
- **insulation piercing connectors**
  - **simple & safe**
  - **protection class up to IP67, even after disconnecting**
- **directly connected slaves**
  - **sensors, actuators**
  - **valve terminals**
  - **electrical modules etc.**

ASI - Topography



- unshielded 2-wire cable
- data and power on one cable
- extension: 100 m (300 m with extender)

- no terminating resistor necessary
- free tree structure of network
- protection class up to IP67

CONTROLNET

- Origin: Allen-Bradley, 1995
- Based on RG6/U cabling (popular in cable TV applications)
- Rockwell ASIC chip
- Maximum Number of Nodes: 99
- Connectors: Twin redundant BNC
- Maximum Distance: 250 to 5000M (with repeaters)
- Baud rate: 5M bit/Sec
- Message Size: 0-510 bytes
- Messaging Formats: Based on Producer/Consumer model; multi-master, peer to peer, fragmented, prioritized and deterministically scheduled repeatable messages; dual transmission paths for built-in redundancy.
- Supporting Trade Organization: ControlNet International, http://www.controlnet.org/

Foundation Fieldbus (H1 and HSE)

- Foundation Fieldbus: The Open International Standard for Mission Critical, Process Control and Intrinsically Safe Environments
- Origin: ISA, 1998
- Implemented on chips produced by multiple vendors.
- "H1" Intrinsically Safe, 31.25Kbit/sec; "HSE" High Speed Ethernet, 100Mbit/sec. Based on ISA SP50/IEC 61158
- Maximum Number of Nodes: 240 per segment; 65,000 possible segments.
- Distance: 1900M for H1
- Baudrate: 31.25K and 100M Bit/sec
- Message size: 128 Octets
- Messaging format: Client/Server, Publisher/Subscriber, Event Notification
- Supporting Trade Organization: Fieldbus Foundation (www.fieldbus.org)
- Typical Applications: Distributed Control Systems; Continuous process control, Batching, Oil and Gas

GPIB – IEEE 488 Std.

History

- 1965 Hewlett-Packard designs HP-IB for instrumentation systems
- 1975 HP-IB becomes IEEE 488standard
- 1987 IEEE 488.2 adopted; IEEE 488-1978 becomes IEEE 488.1-1987
- 1992 SCPI Specification introducedfor IEEE 488 instruments
- 1990 IEEE 488.2 Standard revised
- 1993 National Instruments proposes high-speed extensions to IEEE 488.1 called HS488

In 1965, Hewlett-Packard designed the Hewlett-Packard Interface Bus (HP-IB) to connect their line of programmable instruments to their computers. Because of its high transfer rates (nominally 1 Mbytes/s), this interface bus quickly gained popularity. It was later accepted as IEEE Standard 488-1975, and has evolved to ANSI/IEEE Standard 488.1-1987. Today, the name General Purpose Interface Bus (GPIB) is more widely used than HP-IB. ANSI/IEEE 488.2-1987 strengthened the original standard by defining precisely how controllers and instruments communicate. Standard Commands for Programmable Instruments (SCPI) took the command structures defined in IEEE 488.2 and created a single, comprehensive programming command set that is used with any SCPI instrument.

Basic characteristics

- Any device
- Max. 15 devices with unique primary address from the range 0-30 (extendable by the secondary address from the range 0-30)
- Linear, star or combined configuration of cabling
  - 24/25 wires
  - 3 connector types (Amphenol CHAMP or Cinch Series 57 MICRO RIBBON type)
  - The total cable length max. 20m, max. 4m per device
- TTL levels, negative logic

Almost any instrument can be used with the IEEE-488 specification, because it says nothing about the function of the instrument itself, or about the form of the instrument's data. Instead the specification defines a separate component, the interface, that can be added to the instrument. The signals passing into the interface from the IEEE-488 bus and from the instrument are defined in the standard. The instrument does not have complete control over the interface. Often the bus controller tells the interface what to do.

The IEEE-488 standard allows up to 15 devices to be interconnected on one bus. Each device is assigned a unique primary address, ranging from 0-30, by setting the address switches on the device. A secondary address may also be specified, ranging from 0-30. See the device documentation for more information on how to set the device primary and optional secondary address.

You can link devices in either a linear, star or combination configuration using a shielded 24-conductor cable. The standard IEEE-488 cable has both a plug and receptacle connector on both ends. This connector is the Amphenol CHAMP or Cinch Series 57 MICRO RIBBON type. Special adapters and non-standard cables are available for special interconnect applications.

The IEEE-488 bus specifies a maximum total cable length of 20 meters with no more than 20 devices connected to the bus and at least two-thirds of the devices powered on. A maximum separation of 4 meters between devices and an average separation of 2 meters over the full bus should be followed. Bus extenders and expanders are available to overcome these system limits.

GPIB Messages

- Device-dependent messages (data) – related to the instrument (measurement) functions of device (measured values, ranges and measurement functions to be applied, etc. transmitted in parallel format across the data bus DIO)
- Interface messages (commands) – control of interfaces and communication within the measurement system (addressing, remote/local mode, reset communication interface, … transmitted in parallel format across the data bus DIO or across single wires of the management bus)

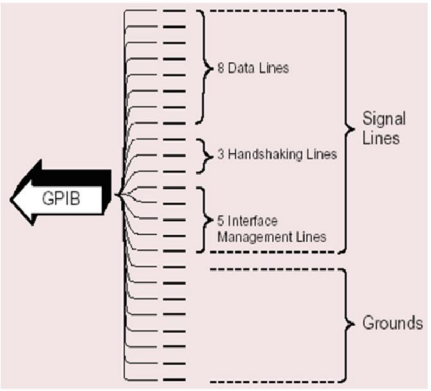
GPIB devices communicate with other GPIB devices by sending device-dependent messages and interface messages through the interface system.

• Device-dependent messages, often called data or data messages, contain device-specific information, such as programming instructions, measurement results, machine status, and data files.

• Interface messages manage the bus. Usually called commands or command messages, interface messages perform such functions as initializing the bus, addressing and unaddressing devices, and setting device modes for remote or local programming. The term "command" as used here should not be confused with some device instructions that are also called commands. Such device-specific commands are actually data messages as far as the GPIB interface system itself is concerned.

Wires and signals

- 8-bits data bus similar to a multiplexed microcontroler bus
- Handshake lines – synchronize data transfer across the data bus
- Management lines to control communication
- 8 ground lines used in twisted pairs with some control wires.



The IEEE-488 interface system consists of 16 signal lines and 8 ground lines. The 16 signal lines are divided into 3 groups (8 data lines, 3 handshake lines, and 5 interface management lines).

Data Lines: The lines DIO1 through DIO8 are used to transfer addresses, control information and data. The state of ATN signal determines meaning of bytes on DIO lines (command/data). The formats for addresses and control bytes are defined by the IEEE 488 standard. Data formats are undefined and may be ASCII (with or without parity) or binary. DIO1 is the Least Significant Bit (note that this will correspond to bit 0 on most computers).

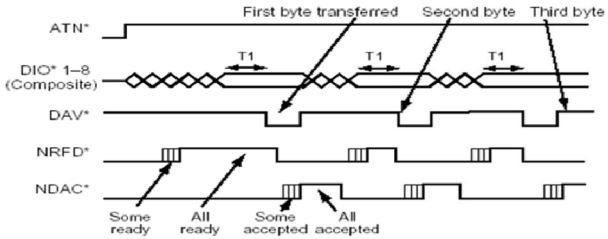
Main Interface function

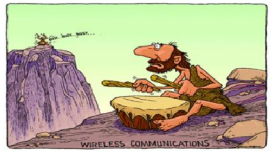
- System Controller and Active Controller (usually PC) control the communication: determine devices to send and receive device dependent messages by addressing, send interface messages, monitor the communication, etc.
- Listener receives device dependent messages prijíma prístrojové správy (one or more active in a time)
- Talker sends device dependent messages (only one active in a time) Talker and Listeners communicates with support of function Source of handshake and Acceptor of handshake.

At power-up time, the IEEE-488 interface that is programmed to be the System Controller becomes the Active Controller in charge. The System Controller has several unique capabilities including the ability to send Interface Clear (IFC) and Remote Enable (REN) commands. IFC clears all device interfaces and returns control to the System Controller. REN allows devices to respond to bus data once they are addressed to listen. The System Controller may optionally Pass Control to another controller, which then becomes Active Controller.
It is possible to have several Controllers on the bus but only one may be active at any given time. The Active Controller may pass control to another controller which in turn can pass it back or on to another controller. A Listener is a device that can receive data from the bus when instructed by the controller and a Talker transmits data on to the bus when instructed. The Controller can set up a talker and a group of listeners so that it is possible to send data between groups of devices as well.
Some GPIB configurations do not require a Controller. For example, a device that is always a Talker, called a talk-only device, is connected to one or more listen-only devices. A Controller is necessary when the active or addressed Talker or Listener must be changed. The Controller function is usually handled by a computer.

Handshaking

- NRFD (Not Ready for Data) – Acceptor of HS
- NDAC (Not Data Accepted) – Acceptor of HS
- DAV (Data Valid) – Source of HS





Handshake Lines: The three handshake lines (NRFD, NDAC, DAV) control the transfer of message bytes among the devices and form the method for acknowledging the transfer of data. This handshaking process guarantees that the bytes on the data lines are sent and received without any transmission errors and is one of the unique features of the IEEE-488 bus.
The NRFD (Not Ready for Data) handshake line is asserted by a Listener to indicate it is not yet ready for the next data or control byte. Note that the Controller will not see NRFD released (i.e., ready for data) until all devices have released it.
The NDAC (Not Data Accepted) handshake line is asserted by a Listener to indicate it has not yet accepted the data or control byte on the data lines. Note that the Controller will not see NDAC released (i.e., data accepted) until all devices have released it.
The DAV (Data Valid) handshake line is asserted by the Talker to indicate that a data or control byte has been placed on the data lines and has had the minimum specified stabilizing time. The byte can now be safely accepted by the devices.
The handshaking process is outlined as follows. When the Controller or a Talker wishes to transmit data on the bus, it sets the DAV line high (data not valid), and checks to see that the NRFD and NDAC lines are both low, and then it puts the data on the data lines. When all the devices that can receive the data are ready, each releases its NRFD (not ready for data) line. When the last receiver releases NRFD, and it goes high, the Controller or Talker takes DAV low indicating that valid data is now on the bus. In response each receiver takes NRFD low again to indicate it is busy and releases NDAC (not data accepted) when it has received the data. When the last receiver has accepted the data, NDAC will go high and the Controller or Talker can set DAV high again to transmit the next byte of data.

Interface management lines

- ATN (Attention) – asserted by Controller controls meaning messages on DIO (+ some other meanings)
- EOI (end or identify) – two meanings:
  - Asserted by Talker: indication of the message last byte,
  - Asserted by Controller: Parallel Poll = reading status bit from instruments.
- IFC (interface clear) – reset of communication interfaces in the system (by system controller)
- REN (remote enable) – switching instruments to remote control (only by controller).
- SRQ (service request) – like an interrupt from any instrument to controller.

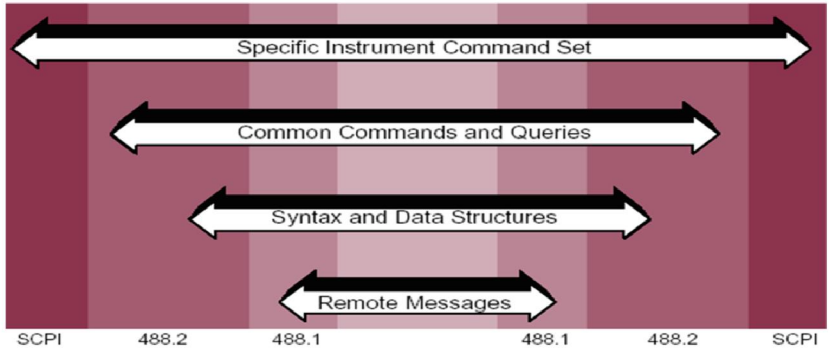
Interface Management Lines: The five interface management lines (ATN, EOI, IFC, REN, SRQ) manage the flow of control and data bytes across the interface.
The ATN (Attention) signal is asserted by the Controller to indicate that it is placing an address or control byte on the data bus. ATN is released to allow the assigned Talker to place status or data on the data bus. The Controller regains control by reasserting ATN; this is normally done synchronously with the handshake to avoid confusion between control and data bytes.
The EOI (End or Identify) signal has two uses. A Talker may assert EOI simultaneously with the last byte of data to indicate end-of-data. The Controller may assert EOI along with ATN to initiate a parallel poll. Although many devices do not use parallel poll, all devices should use EOI to end transfers (many currently available ones do not).
The IFC (Interface Clear) signal is asserted only by the System Controller in order to initialize all device interfaces to a known state. After releasing IFC, the System Controller is the Active Controller.
The REN (Remote Enable) signal is asserted only by the System Controller. Its assertion does not place devices into remote control mode; REN only enables a device to go into remote mode when addressed to listen. When in remote mode, a device should ignore its local front panel controls.
The SRQ (Service Request) line is like an interrupt: it may be asserted by any device to request the Controller to take some action. The Controller must determine which device is asserting SRQ by conducting a serial poll. The requesting device releases SRQ when it is polled.

IEEE 488.2 and SCPI

- 488.1 (1975) – focused only mechanical, electrical and hardware, missed: data formats, coding, what to do if …, poor status indication, …
- 488.2 (1987) extension of 488.1 (compatible): focused on software, data compatibility, etc.
- SCPI – compatibility of messages from/to equivalent instruments but by different vendors

Standard 488-1975, now called IEEE 488.1, greatly simplified the interconnection of programmable instrumentation by clearly defining mechanical, electrical, and hardware protocol specifications. For the first time, instruments from different manufacturers were interconnected by a standard cable. Although this standard went a long way towards improving the productivity of test engineers, the standard did have a number of shortcomings. Specifically, IEEE 488.1 did not address data formats, status reporting, message exchange protocol, common configuration commands, or device-specific commands. As a result, each manufacturer implemented these items differently, leaving the test system developer with a formidable task.
IEEE 488.2 enhanced and strengthened IEEE 488.1 by standardizing data formats, status reporting, error handling, Controller functionality, and common commands to which all instruments must respond in a defined manner. By standardizing these issues, IEEE 488.2 systems are much more compatible and reliable. The IEEE 488.2 standard focuses mainly on the software protocol issues and thus maintains compatibility with the hardware-oriented IEEE 488.1 standard.
SCPI built on the IEEE 488.2 standard and defined device-specific commands that standardize programming instruments. SCPI systems are much easier to program and maintain. In many cases, you can interchange or upgrade instruments without having to change the test program. The combination of SCPI and IEEE 488.2 offers significant productivity gains, and finally, delivers as sound a software standard as IEEE 488.1 did a hardware standard.

Comparison of IEEE 488.1 488.2 a SCPI



The SCPI and IEEE 488.2 standards addressed the limitations and ambiguities of the original IEEE 488 standard. IEEE 488.2 makes it possible to design more compatible and productive test systems. SCPI simplifies the programming task by defining a single comprehensive command set for programmable instrumentation, regardless of type or manufacturer. The scope of each of the IEEE 488, IEEE 488.2, and SCPI standards is shown in the figure. The ANSI/IEEE Standard 488-1975, now called IEEE 488.1, greatly simplified the interconnection of programmable instrumentation by clearly defining mechanical, electrical, and hardware protocol specifications. For the first time, instruments from different manufacturers were interconnected by a standard cable. Although this standard went a long way towards improving the productivity of test engineers, the standard did have a number of shortcomings. Specifically, IEEE 488.1 did not address data formats, status reporting, message exchange protocol, common configuration commands, or device-specific commands. As a result, each IEEE 488.2 enhanced and strengthened IEEE 488.1 by standardizing data formats, status reporting, error handling, Controller functionality, and common commands to which all instruments must respond in a defined manner. By standardizing these issues, IEEE 488.2 systems are much more compatible and reliable. The IEEE 488.2 standard focuses mainly on the software protocol issues and thus maintains compatibility with the hardware-oriented IEEE 488.1 standard. SCPI built on the IEEE 488.2 standard and defined device-specific commands that standardize programming instruments. SCPI systems are much easier to program and maintain. In many cases, you can interchange or upgrade instruments without having to change the test program. The combination of SCPI and IEEE 488.2 offers significant productivity gains, and finally, delivers as sound a software standard as IEEE 488.1 did a hardware standard.

IEEE 488.2 controller mandatory sequencies

Description	Control Sequence
Send ATN-true commands	SEND COMMAND
Set address to send data	SEND SETUP
Send ATN-false data	SEND DATA BYTES
Send a program message	SEND
Set address to receive data	RECEIVE SETUP
Receive ATN-false data	RECEIVE RESPONSE MESSAGE
Receive a response message	RECEIVE
Pulse IFC line	SEND IFC
Place devices in DCAS	DEVICE CLEAR
Place devices in local state	ENABLE LOCAL CONTROLS
Place devices in remote state	ENABLE REMOTE
Place dev. in remote with local lockout state	SET RWLS
Place devices in local lockout state	SEND LLO
Read IEEE 488.1 status byte	READ STATUS BYTE
Send group execution trigger (GET) message	TRIGGER

IEEE 488.2 Control Sequences – The IEEE 488.2 standard defined control sequences that specify the exact IEEE 488.1 messages that are sent from the Controller as well as the ordering of multiple messages. IEEE 488.2 defined 15 required control sequences and four optional control sequences, as shown in the table. The IEEE 488.2 control sequences describe the exact states of the GPIB and the ordering of command messages for each of the defined operations. IEEE 488.2 control sequences remove the ambiguity of the possible bus conditions, so instruments and Controllers are much more compatible. By exactly defining the state of the bus and how devices should respond to specific messages, IEEE 488.2 removes such system development problems.

IEEE 488.2 controller protocols

Keyword	Name	Compliance
RESET	Reset System	Mandatory
FINDRQS	Find Device Requesting Service	Optional
ALLSPOLL	Serial Poll All Devices	Mandatory
PASSCTL	Pass Control	Optional
REQUESTCTL	Request Control	Optional
FINDLSTN	Find Listeners	Optional
SETADD	Set Address	Optional, but requires FINDLSTN
TESTSYS	Self-Test System	Optional

IEEE 488.2 Protocols – Protocols are high-level routines that combine a number of control sequences to perform common test system operations. IEEE 488.2 defines two required protocols and six optional protocols, as shown in the table. These protocols reduce development time because they combine several commands to execute the most common operations required by any test system. The RESET protocol ensures that the GPIB has been initialized and all devices have been cleared and set to a known state. The ALLSPOLL protocol serial polls each device and returns the status byte of each device.
The PASSCTL and REQUESTCTL protocols pass control of the bus between a number of different devices. The TESTSYS protocol instructs each device to run its own self-tests and report back to the Controller whether it has a problem or is ready for operation. Perhaps the two most important protocols are FINDLSTN and FINDRQS. The FINDLSTN protocol takes advantage of the IEEE 488.2 Controller capability of monitoring bus lines to locate listening
devices on the bus. The Controller implements the FINDLSTN protocol by issuing a particular listen address and then monitoring the NDAC handshake line to determine if a device exists at that address. The result of the FINDLSTN protocol is a list of addresses for all the located devices. FINDLSTN is used at the start of an application program to ensure proper system configuration and to provide a valid list of GPIB devices that can be used as the input parameter to all other IEEE 488.2 protocols. The bus line monitoring capability of an IEEE 488.2 Controller is also useful to detect and diagnose problems within a test system. The FINDRQS protocol is an efficient mechanism for locating and polling devices that are requesting service. It uses the IEEE 488.2 Controller capability of sensing the FALSE to TRUE transition of the SRQ line. You prioritize the input list of devices so that the more critical devices receive service first. If the application program can jump to this protocol immediately upon the assertion of the SRQ line, you increase program efficiency and throughput.

IEEE 488.2 minimal configuration

- Minimal combination of interface functions:
  - To send and receive data (T5, T6, TE5, TE6, L3, L4, LE3, LE4, AH1, SH1)
  - To request service (SR1)
  - To respond to a device clear (DC1)

+ other capabilities are optional:

  RL0/RL1; PP0/PP1; DT0/DT1; E1/E2; C0/C in some version.

IEEE 488.2 defines a minimum set of IEEE 488.1 interface capabilities that an instrument must have. All devices must be able to send and receive data, request service, and respond to a device clear message. IEEE 488.2 defines precisely the format of commands sent to instruments and the format and coding of responses sent by instruments.
All instruments must perform certain operations to communicate on the bus and report status.

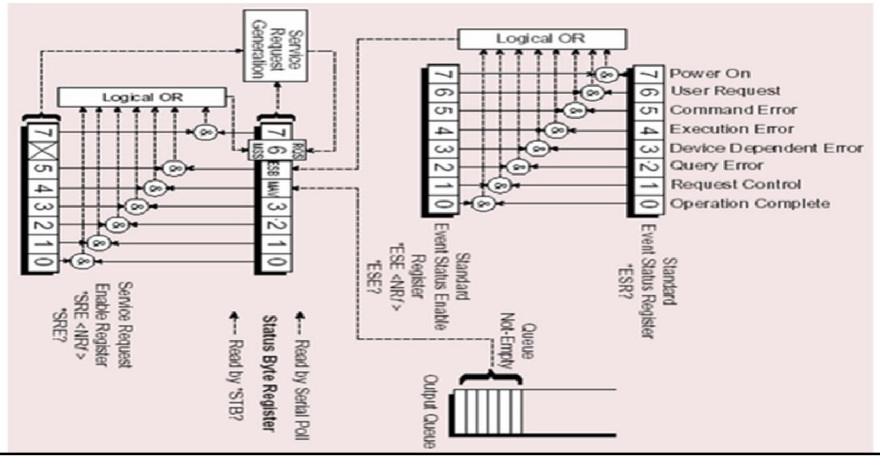
IEEE 488.2 common commands and queries

Mnemonic	Group	Description
*IDN?	System Data	Identification query
*RST	Internal Operations	Reset
*TST?	Internal Operations	Self-test query
*OPC	Synchronization	Operation complete
*OPC?	Synchronization	Operation complete query
*WAI	Synchronization	Wait to complete
*CLS	Status and Event	Clear status
*ESE	Status and Event	Event status enable
*ESE?	Status and Event	Event status enable query
*ESR?	Status and Event	Event status register query
*SRE	Status and Event	Service request enable
*SRE?	Status and Event	Service request enable query
*STB?	Status and Event	Read status byte query

Because these operations are common to all instruments, IEEE 488.2 defined the programming commands used to execute these operations and the queries used to receive common status information. These common commands and queries are shown in the table.

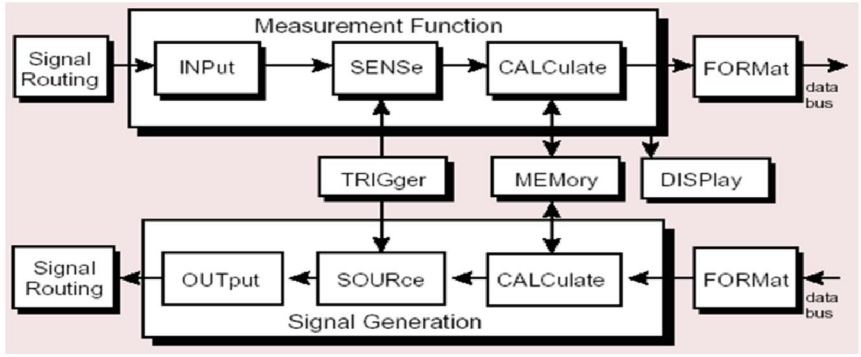
Because IEEE 488.2 standardizes status reporting, the Controller knows exactly how to obtain status information from every instrument in the system. This status reporting model builds upon the IEEE 488.1 status byte to provide more detailed status information.

IEEE 488.2 device status model



Because IEEE 488.2 standardizes status reporting, the Controller knows exactly how to obtain status information from every instrument in the system. This status reporting model builds upon the IEEE 488.1 status byte to provide more detailed status information.

SCPI instrument model



In April 1990, a group of instrument manufacturers announced the SCPI specification, which defines a common command set for programming instruments. Before SCPI, each instrument manufacturer developed its own command sets for its programmable instruments. This lack of standardization forced test system developers to learn a number of different command sets and instrument-specific parameters for the various instruments used in an application, leading to programming complexities and resulting in unpredictable schedule delays and development costs. By defining a standard programming command set, SCPI decreases development time and increases the readability of test programs and the ability to interchange instruments.

SCPI is a complete, yet extendable, standard that unifies the software programming commands for instruments. The first version of the standard was released in mid-1990. Today, the SCPI Consortium continues to add commands and functionality to the SCPI standard. SCPI has its own set of required common commands in addition

to the mandatory IEEE 488.2 common commands and queries. Although IEEE 488.2 is used as its basis, SCPI defines programming commands that you can use with any type of hardware or communication link. SCPI specifies standard rules for abbreviating command keywords and uses the IEEE 488.2 message exchange protocol rules to format commands and parameters. You may use command keywords in their long form (MEASure) or their short form shown in capital letters (MEAS).

The SCPI Instrument Model – As a means of achieving compatibility and categorizing command groups, SCPI defined a model of a programmable instrument. This model applies to all the different types of instrumentation. All of the functional components of the instrument model may not apply to every instrument. For example, an oscilloscope does not have the functionality defined by the signal generation block in the SCPI model. SCPI defines hierarchical command sets to control specific functionality within each of these functional components.