

SERIAL PORT COMMUNICATION

Serial communication is a popular means of transmitting data between a computer and a peripheral device such as a programmable instrument or another computer. Serial communication uses a transmitter to send data, one bit at a time, over a single communication line to a receiver. This method is used when data transfer rates are low or data transfer over long distances is required. Serial communication is popular because most computers have one or more serial ports, so no extra hardware is needed other than a cable to connect instrument to the computer (or two computers to each other).

One must specify four parameters for serial communication: the *baud rate* of the transmission, the number of *data bits* encoding a character, the sense of the optional *parity* bit, and the number of *stop bits*. Each transmitted character is packaged in a character frame that consists of a single start bit followed by the data bits.

- *Baud rate* is a measure of how fast data moves between instruments that use serial communication.
- A *start bit* signals the beginning of each character frame.
- *Data bits* are transmitted “upside down and backwards.” That is, inverted logic is used and the order of transmission is from least significant bit (LSB) to most significant bit (MSB).
- An optional *parity bit* follows the data bits in the character frame. The parity bit, if present, also follows inverted logic. This bit is included as a simple means of error checking.
- The last part of a character frame consists of 1, 1.5, or 2 *stop bits*. These bits are always represented by a negative voltage.




RS-232C

RS-232 stands for Recommend Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. The full RS-232C standard specifies a 25-pin "D" connector of which 22 pins are used. Most of these pins are not needed for normal PC communications, and indeed, most new PCs are equipped with male D type connectors having only 9 pins.

DCE and DTE Devices

Two terms you should be familiar with are DTE and DCE. DTE stands for Data Terminal Equipment, and DCE stands for Data Communications Equipment. These terms are used to indicate the pin-out for the connectors on a device and the direction of the signals on the pins. Your computer is a DTE device, while most other devices are usually DCE devices.

9 Pin Connector on a DTE device (PC connection)	
Male RS232 DB9	
Pin Number	Direction of signal:
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

The TD (transmit data) wire is the one through which data from a DTE device is transmitted to a DCE device. This name can be deceiving, because this wire is used by a DCE device to receive its data. The TD line is kept in a mark condition by the DTE device when it is idle. The RD (receive data) wire is the one on which data is received by a DTE device, and the DCE device keeps this line in a mark condition when idle.

RTS stands for Request To Send. This line and the CTS line are used when "hardware flow control" is enabled in both the DTE and DCE devices. The DTE device puts this line in a mark condition to tell the remote device that it is ready and able to receive data. If the DTE device is not able to receive data (typically because its receive buffer is almost full), it will put this line in the space condition as a signal to the DCE to stop sending data. When the DTE device is ready to receive more data (i.e. after data has been removed from its receive buffer), it will place this line back in the mark condition. The complement of the RTS wire is CTS, which stands for Clear To Send. The DCE device puts this line in a mark condition to tell the DTE device that it is ready to receive the data. Likewise, if the DCE device is unable to receive data, it will place this line in the space condition. Together, these two lines make up what is called RTS/CTS or "hardware" flow control.

DTR stands for Data Terminal Ready. Its intended function is very similar to the RTS line. DSR (Data Set Ready) is the companion to DTR in the same way that CTS is to RTS. Some serial devices use DTR and DSR as signals to simply confirm that a device is connected and is turned on. The DTR and DSR lines were originally designed to provide an alternate method of

hardware handshaking. It would be pointless to use both RTS/CTS and DTR/DSR for flow control signals at the same time. Because of this, DTR and DSR are rarely used for flow control.

CD stands for Carrier Detect. Carrier Detect is used by a modem to signal that it has made a connection with another modem, or has detected a carrier tone.

The last remaining line is RI or Ring Indicator. A modem toggles the state of this line when an incoming call rings your phone.

The Carrier Detect (CD) and the Ring Indicator (RI) lines are only available in connections to a modem. Because most modems transmit status information to a PC when either a carrier signal is detected (i.e. when a connection is made to another modem) or when the line is ringing, these two lines are rarely used.

THE GENERAL PURPOSE INTERFACE BUS (GPIB)

HARDWARE REQUIREMENTS

1. GPIB PCI card
2. PC (acts as Controller)
3. GPIB compatible Instrument (Agilent 34970A DMM)
4. GPIB Cable

GPIB BACKGROUND

- The General Purpose Interface Bus (GPIB) is one of the most common I/O interfaces available in stand-alone instruments.
- Originally designed by Hewlett Packard
- GPIB is a digital, 8-bit parallel communications interface with data transfer rates of up to 8 Mb/s.
- The bus provides one system controller for up to 14 instruments, and cabling is limited to less than 20 m.

GPIB is a digital, 24-conductor parallel bus. It consists of eight data lines (DIO 1-8), five bus management lines (EOI, IFC, SRQ, ATN, REN), three handshake lines (DAV, NRFD, NDAC), and eight ground lines. The GPIB uses an eight-bit parallel, byte-serial, asynchronous data transfer scheme. This means that whole bytes are sequentially handshaked across the bus at a speed that the slowest participant in the transfer determines. Because the unit of data on the GPIB is a byte (eight bits), the messages transferred are frequently encoded as ASCII character strings.

The specifications are listed below:

- A maximum separation of 4 m between any two devices and an average separation of 2 m over the entire bus.
- A maximum cable length of 20 m.
- A maximum of 15 devices connected to each bus with at least two-thirds of the devices powered on.

TALKERS, LISTENERS AND CONTROLLERS

GPIB Devices can be Talkers, Listeners, and/or Controllers. A **Talker** sends data messages to one or more **Listeners**, which receive the data. The **Controller** manages the flow of information on the GPIB by sending commands to all devices. A digital voltmeter, for example, is a Talker and is also a Listener.

The role of the GPIB Controller is comparable to the role of a computer CPU, but a better analogy is to compare the Controller to the switching center of a city telephone system.

The switching center (Controller) monitors the communications network (GPIB). When the center (Controller) notices that a party (device) wants to make a call (send a data message), it connects the caller (Talker) to the receiver (Listener).

The Controller usually addresses (or enables) a Talker and a Listener before the Talker can send its message to the Listener. After the message is transmitted, the Controller may address other Talkers and Listeners.

GPIB SIGNALS

Data Lines

The eight data lines, DIO1 through DIO8, carry both data and command messages. The state of the Attention (ATN) line determines whether the information is data or commands. All commands and most data use the 7-bit ASCII or ISO code set, in which case the eighth bit, DIO8, is either unused or used for parity.

Handshake Lines

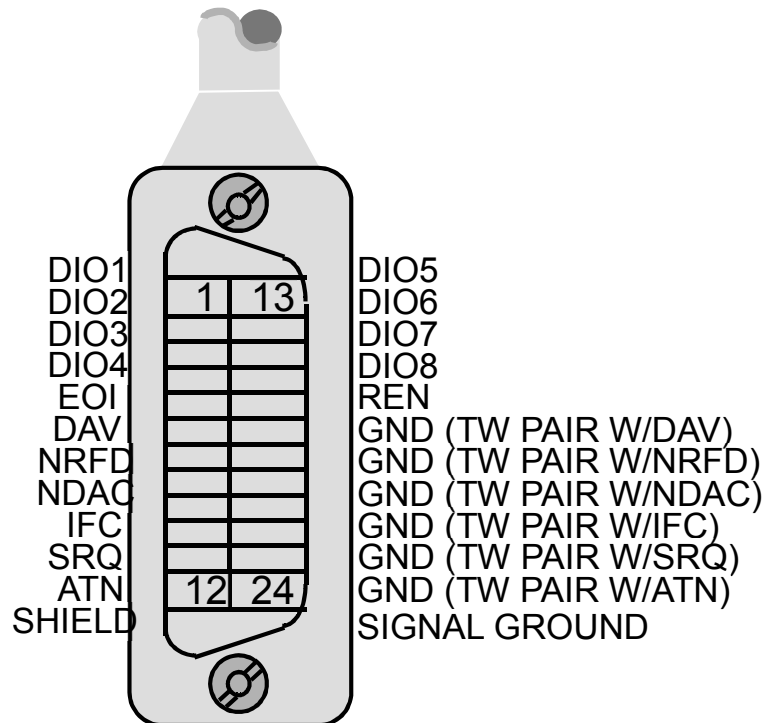
Three lines **asynchronously** control the transfer of message bytes between devices. The process is called a **3-wire interlocked handshake**. It guarantees that message bytes on the data lines are sent and received without transmission error.

- **NRFD (not ready for data)** - Indicates when a device is ready or not ready to receive a message byte. The line is driven by all devices when receiving commands, by Listeners when receiving data messages, and by the Talker when enabling the HS488 protocol.
- **NDAC (not data accepted)** - Indicates when a device has or has not accepted a message byte. The line is driven by all devices when receiving commands, and by Listeners when receiving data messages.
- **DAV (data valid)** - Tells when the signals on the data lines are stable (valid) and can be accepted safely by devices. The Controller drives DAV when sending commands, and the Talker drives DAV when sending data messages.

Interface Management Lines

Five lines manage the flow of information across the interface:

- **ATN (attention)** - The Controller drives ATN true when it uses the data lines to send commands, and drives ATN false when a Talker can send data messages.
- **IFC (interface clear)** - The System Controller drives the IFC line to initialize the bus and become CIC.
- **REN (remote enable)** - The System Controller drives the REN line, which is used to place devices in remote or local program mode.
- **SRQ (service request)** - Any device can drive the SRQ line to asynchronously request service from the Controller.
- **EOI (end or identify)** - The EOI line has two purposes - The Talker uses the EOI line to mark the end of a message string, and the Controller uses the EOI line to tell devices to identify their response in a parallel poll.



GPIB PIN CONFIGURATION



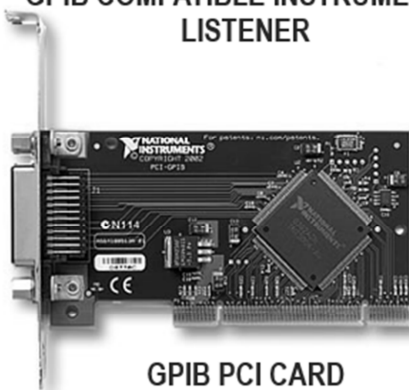
GPIB CABLE



**GPIB COMPATIBLE INSTRUMENT –
LISTENER**



PC - TALKER



GPIB PCI CARD

GPIB HARDWARE REQUIREMENTS

Measurement Commands

```
MEASure:CURRent:AC? [{<range>|AUTO|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
MEASure:CURRent[:DC]? [{<range>|AUTO|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
MEASure:DIGital? {BYTE|1|WORD|2|LWORD|4}, [{<voltage>,} [{NORMal|INVerted} , ] (@<ch_list>)
MEASure:DIGital: {BYTE|1|WORD|2|LWORD|4}? (@<ch_list>)
MEASure:FREQuency? [{<range>|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
MEASure:FRESistance? [{<range>|AUTO|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
MEASure:PERiod? [{<range>|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
MEASure:RESistance? [{<range>|AUTO|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
MEASure:TEMPerature? {TCouple|RTD|FRTD|THERmistors|DEF}, {<type>|DEF} [, 1 [, {<resolution>|MIN|MAX|DEF}] ] [, (@<ch_list>)]
MEASure:TOTALize? [{READ|RRESet},] (@<ch_list>)
MEASure[:VOLTage]:AC? [{<range>|AUTO|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
MEASure[:VOLTage][:DC]? [{<range>|AUTO|MIN|MAX|DEF} [, {<resolution>|MIN|MAX|DEF}] , ] [(@<ch_list>)]
```

ETHERNET COMMUNICATION

HARDWARE REQUIREMENTS

1. Ethernet cable
2. Controller (PC)
3. Ethernet card (Installed in the Controller)
4. Ethernet compatible Instrument (PC with Ethernet card connected to Microcontroller board via RS232)

OVERVIEW OF ETHERNET

Ethernet is a mature technology widely used for measurement systems in many capacities including general networking and remote data storage. Ethernet offers an attractive option for instrument control because of its flexibility. With connectivity via Ethernet, you can control your instrument from anywhere on the network to which it is connected. Ethernet, defined by IEEE Standard 802.3, offers network configurations that provide theoretical data transfer rates of 10 Mb/s (10BaseT), 100 Mb/s (100BaseTX), and 1 Gb/s (1000BaseT). One main reason Ethernet has become so popular in instrument control is the fact that most companies and laboratories have existing Ethernet networks available for instrument control. LXI is a new standard built on the Ethernet protocol. It was developed to implement synchronization of distributed instruments.

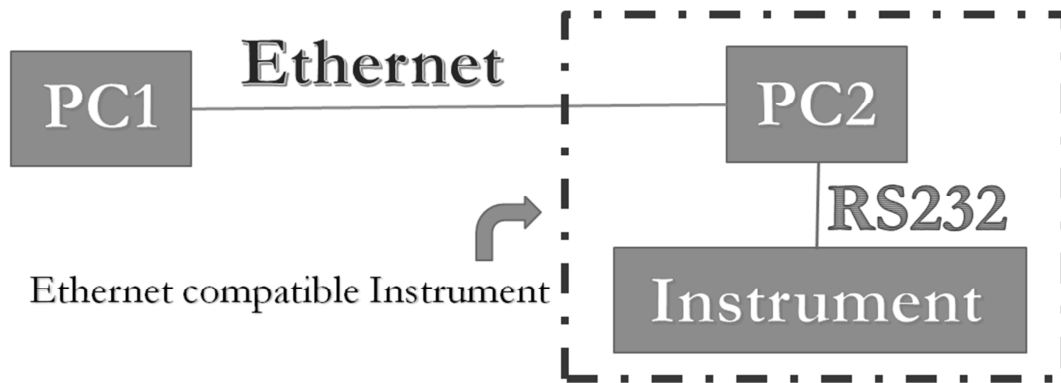
TCP/IP COMMUNICATION

Internet Protocol (IP), User Datagram Protocol (UDP), and Transmission Control Protocol (TCP) are the basic tools for network communication. The name TCP/IP comes from two of the best-known protocols of the internet protocol suite, the Transmission Control Protocol and the Internet Protocol. With TCP/IP one can communicate over single networks or interconnected networks (Internet).

TCP/IP communication provides a simple user interface that conceals the complexities of ensuring reliable network communications. The TCP/IP functions are located on the **Functions » Communication » TCP** palette for TCP communication in LabVIEW. As with instrument, and File I/O communication, the process involves opening the connection, reading and writing the information, and closing the connection.

With most I/O communication, the processor is always the client that initiates a connection to the disk drive server, the external instrument server, or the DAQ board server. With TCP/IP connections, a computer can function either as the client or the server. LabVIEW users can develop custom applications for TCP/IP communication. The programmer is responsible for developing both the client and the server.

INSTRUMENT SETUP



PC1 – Transmitter/Server, PC2+Instrument – Receiver/Client

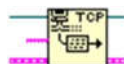
TCP/IP FUNCTIONS IN LabVIEW

TCP/IP Listen



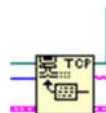
The TCP/IP Listen listens over the network for another computer trying to connect over the port specified in the integer input at the left side of the icon. That port number (0 to 6000) must be a number not used by other parts of the system. When the TCP/IP Listen hears another computer trying to connect over the correct port, it establishes a Connection ID, and an Error Signal, and those outputs are used in later TCP/IP VI blocks. One cannot use another TCP/IP Listen with the same port number once the connection has been established.

TCP/IP Write - which writes data to the other computer over the network.



The TCP/IP Write uses the Connection ID, which it also passes to the next VI, and the Error signal which it also passes to the next VI. The input is a string. It's much harder to use numerical information directly, so one has to convert your data to a string format. One can also specify a timeout, and can get information on the number of bytes written over the network with this VI.

TCP/IP Read - which reads bytes from the remote computer.



There are the usual inputs, the Connection ID and the Error Signal. The integer input (blue line at left) is the number of bytes that you want to read. The output is the string composed of the bytes that are read.