About the Author

Mr. John Pieper was awarded a Master's Degree in Electrical Engineering from the University of Delft, The Netherlands. He has many years of outstanding industry experience and has been involved with the development of many test and measurement instruments and software products for ATE systems. Mr. Pieper participates in several international ATE standardization activities. He was a member of Working Group 3 of IEC/TC65/SC65C (responsible for GPIB and related standards) and is closely involved with the IEEE Instrumentation and Measurement Society, TC8, responsible for IEEE 488.1, IEEE 488.2, and the new RS 232 based serial instrument interface IEEE 1174. He actively cooperates with the sub committee UK951.1 of the German DKE - Deutsche Elektrotechnische Kommission - on interface technologies. Until recently he was the Vice President of the Board of Directors of the SCPI consortium and a member of their Large Technical Committee. He now is the European SCPI contact address.

Mr. Pieper is the Managing Director of ACEA, a company residing in Wierden, The Netherlands, providing products, consulting services, educational training and applications for Automatic Test and Measurement systems.

IEEE 1174 - A new Serial Instrumentation Interface

By Ir. John M. Pieper

ACEA, Wierden, The Netherlands

Main objective of the new IEEE 1174 "Serial Interface for Programmable Instrumentation" standard is to define a technique which allows the popular serial RS 232 interface to be used with other industry standards in the area of Automatic Test Equipment (ATE).

Interface standards, like GPIB and VXI did already provide a path towards upper layer standards, like IEEE 488.2, which, in its turn, accommodates the implementation of the Standard Commands for Programmable Instruments (SCPI). Up to now, such a path was not provided for serial interfaces, as can be seen from figure 1 which shows the relation between the IEEE 1174 standard and other, important ATE standards.

Furthermore, IEEE 1174 adds interface functionality to serial communication links, that is commonly used in other interface media, like GPIB and VXI. Main benefit of this concept is that, regardless of the interface medium being used, the higher functional layers are kept identical. This results in a simpler design, causing a reduction of the development efforts, and thus saves investment costs.

Additionally, the new IEEE 1174 standard defines a common practice for the popular RS232 interface. As such, the standard is fully compliant to RS 232 standard. The publication of the new standard, which is currently be balloted, is to be expected at about the end of this year.

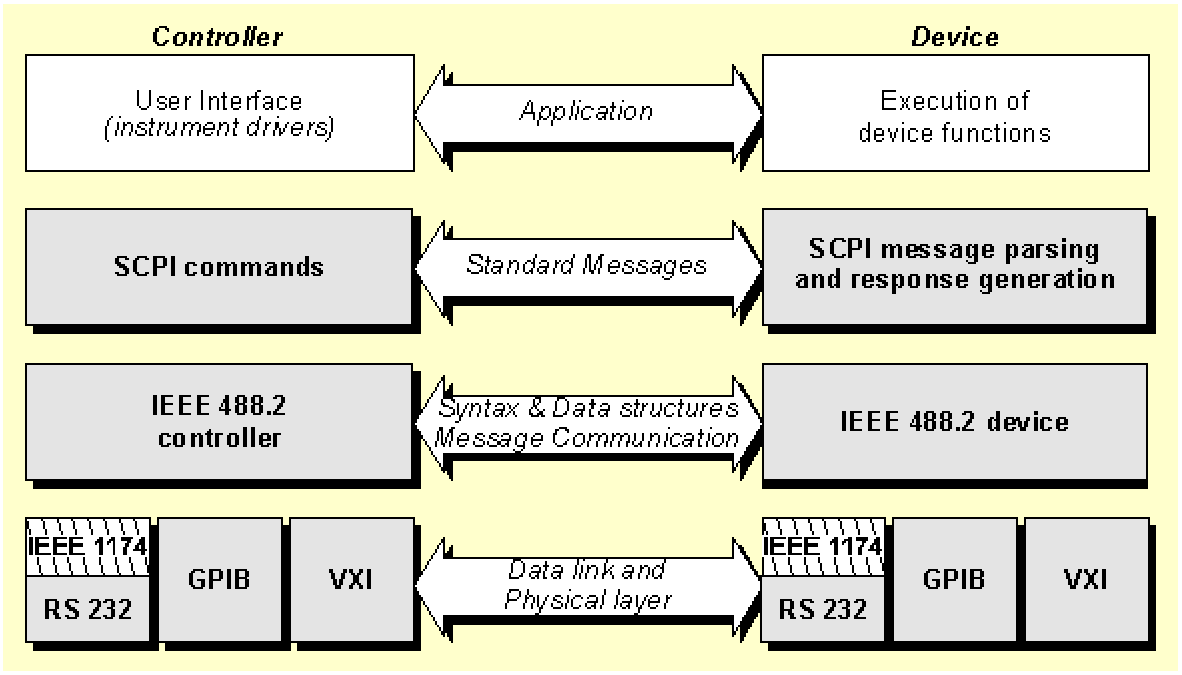

Controller
Device
User Interface (instrument drivers)
Application
Execution of device functions
SCPI commands
Standard Messages
SCPI message parsing and response generation
IEEE 488.2 controller
Syntax & Data structures Message Communication
IEEE 488.2 device
IEEE 1174
RS 232
GPIB
VXI
Data link and Physical layer
IEEE 1174
RS 232
GPIB
VXI

Figure 1. Relation between the ATE standards

Not all instruments equipped with a serial interface need full compliance to the higher level standards. Therefor, the IEEE 1174 standard consists of a leveled set of three documents, each part providing an increased level of compatibility with other standard, which together describe a communication method for a serial link between a controller and an instrument.

IEEE 1174.0.	The first part of the set defines a common usage of the RS232 and related standards for instruments and controllers; it identifies the electrical, mechanical and interchange circuit requirements and describes a full duplex, asynchronous 9-pin DTE connector port.
IEEE 1774.1.	The second part of the standard describes a technique to emulate standard GPIB functionality over a serial link, thus making the valuable properties of this well known parallel interface also available to serial interfaces.
IEEE 1774.2	The last part of the IEEE 1774 set identifies the required IEEE 488.2 functionality and describes its implementation for serial links. It defines how particular syntactical elements from the language construct must be implemented, whereas major part deals with the definition of IEEE 488.2's Message Exchange Control protocol for this interface type. A state machine implementation of this protocol is added as an informative appendix to the standard.

Figure 2 shows the functionality covered by each part and their mutual relations:

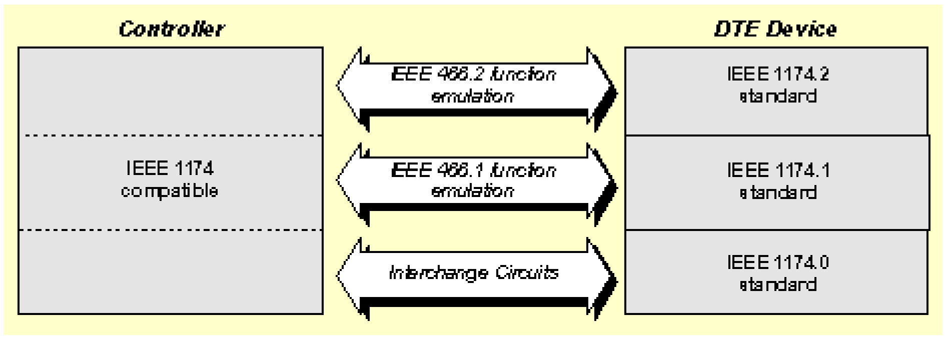


Figure 2. Relation between the IEEE 1174 standard set.

For instruments with limited capabilities, it may be sufficient to provide compatibility with IEEE 1174.0 only, whereas other instruments may be compliant with IEEE 1174.1 or IEEE 1174.2. In all cases, compliance with a particular part, requires compliance with the previous parts. For example, compliance with IEEE 1174.2 requires compatibility with IEEE 1174.1, which in its turn requires compatibility with IEEE 1174.0.

The basic part. *Compliance class 1*

The RS 232 standard basically deals with only a part of a serial communication link; in particular it covers the connection between a so called Data Terminal Equipment (DTE) and the "modulator/demodulator", which is commonly known by its abbreviation "modem'. This "modem' is formally called the Data Communication Equipment (DCE). The RS 232 standard is an American (EIA/ANSI) standard, which is internationally covered by CCITT telecommunication and ISO standards. These standards do not define all details of a connection; for example, electrical levels for balanced and unbalanced signals, and for mechanical characteristics as connector dimensions en pin assignment, are specified by other standards. Notwithstanding the intended purpose and application area of these standards, the serial interface RS 232 is frequently used to just establish a link between two peripherals. Since peripherals are to be considered as DTE's, which have nothing to do with DCE's or "modems", the standard protocols couldn't be applied. The result of this practice was that several deviating methods were created to establish a proper communication. It is not an overstatement to say that any possible solution is somewhere implemented.

For example, the formal protocol considers the opposite of a DTE as a being DCE. Manufacturers of equipment which has to communicate with a DTE, might therefor consider their a serial port as a DCE terminal. It happened that different selections on pinning numbers, signal directions and male and female connector options could be encountered

A commonly found solution for a connection without modems is the 'null modem". This is a cable which crosses the signal lines, as is shown in figure 3. The part of the link that is established by both DCE's and the link in between them, is left out and replaced by a cable connection between both DTE's.

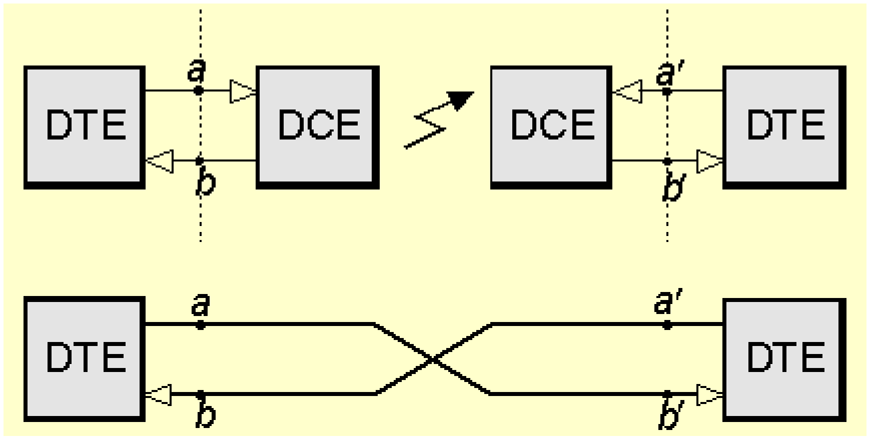


Figure 3. Null modem cabling

As said, the standard protocols were created to establish a link between a DTE and a DCE. A good example of a protocol which isn't applicable for a link between two peripherals is the RTS/CTS protocol. Request To Send (RTS) is a request from the peripheral to the DCE modem to set up its communication. When the DCE has set up a connection with the other modem, it acknowledges the requesting DTE by sending the Clear To Send (CTS). Once CTS is ON, the DTE peripheral is allowed to transmit the data.

It is obvious that such a protocol is not needed in a direct communication link between an instrument and its controller. Such links need different functionality. An important protocol that is always needed in the communication between peripherals is a mechanism for data flow control; it may easily occur that a peripheral receives more data than it can process. In order to prevent the peripheral's input buffer to overflow, it has to signal to the sending DTE, that it has to halt sending data.

The original RS 232-C standard, and its successor RS 232-D, didn't provide such a mechanism. Their international equivalent CCITT standard V24, did provide the needed mechanism, by providing a circuit 133, Ready For Receiving (RFR). However, the ISO 2110 standard which defined the connector pin allocation, did not assign a pin to the circuit 133.

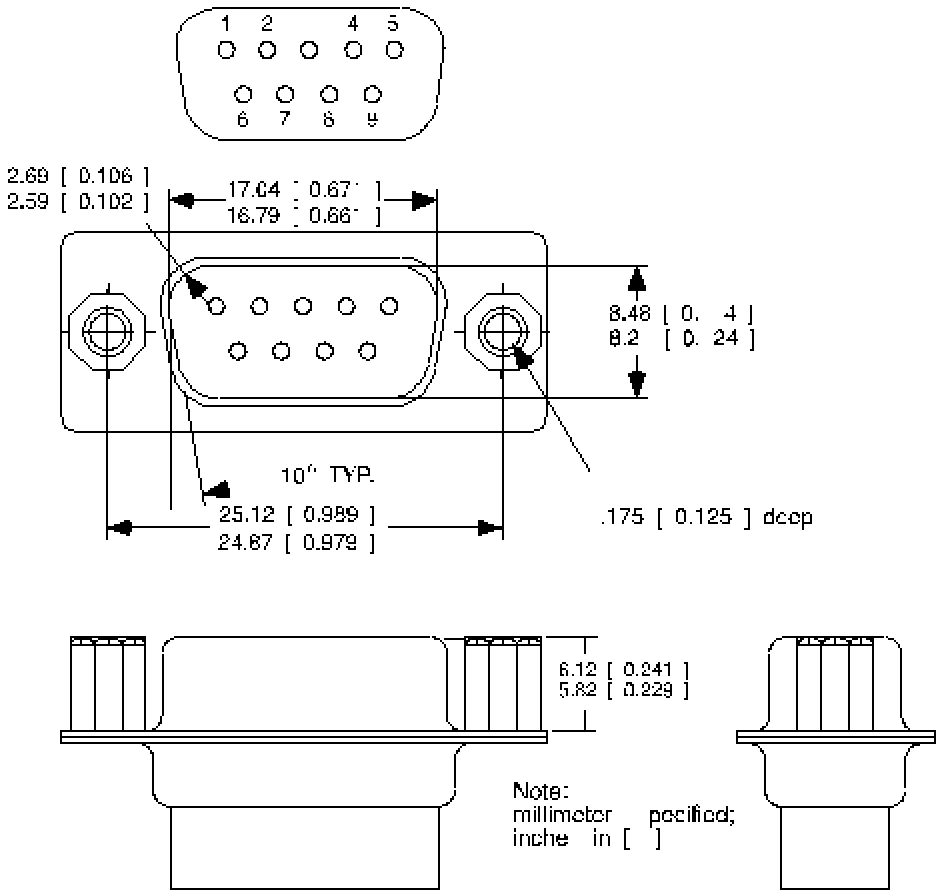
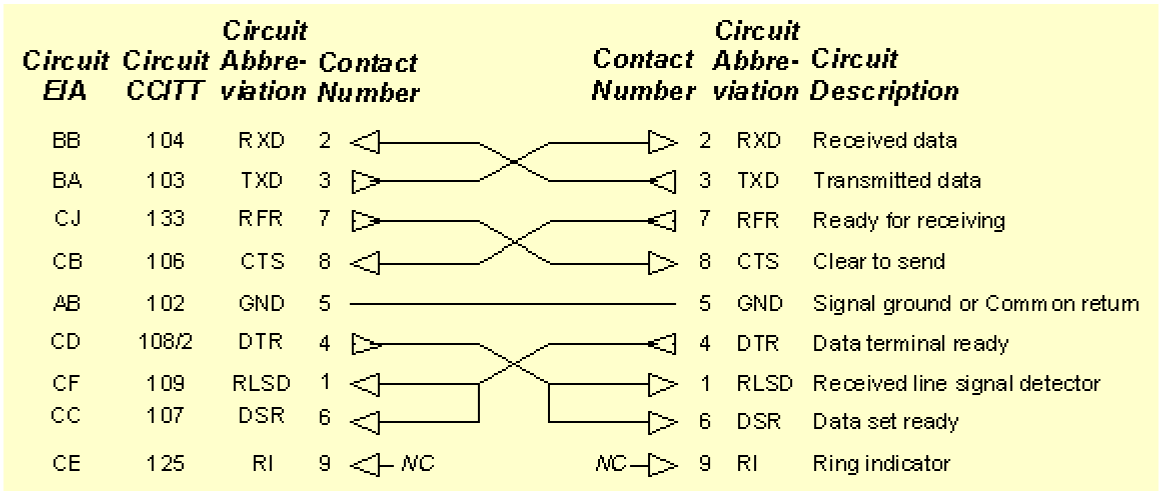


Figure 4. Connector for use with IEEE 1174

Recognizing the common practice and need for DTE-DTE communication on serial links, the latest revision of the RS232 standard allows the RTS circuit to be replaced by the RFR circuit when data flow control is needed. The IEEE 1174 standard uses this circuit and requires its implementation when hard flow control is to be implemented.

For reasons of saving panel space, and because instruments and controllers, communicating as DTE's over a serial link, do not need the extensive circuitry that is available for DTE to DCE connection, the 25-pole connector as is defined for RS 232 is not used. Instead, the 9 pole connector, as defined by the EIA standards 574 and 449 is used, as is shown in figure 4.



Circuit EIA	Circuit CCITT	Circuit Abbre-viation	Contact Number	Contact Number	Circuit Abbre-viation	Circuit Description
BB	104	RXD	2	2	RXD	Received data
BA	103	TXD	3	3	TXD	Transmitted data
CJ	133	RFR	7	7	RFR	Ready for receiving
CB	106	CTS	8	8	CTS	Clear to send
AB	102	GND	5	5	GND	Signal ground or Common return
CD	108/2	DTR	4	4	DTR	Data terminal ready
CF	109	RLSD	1	1	RLSD	Received line signal detector
CC	107	DSR	6	6	DSR	Data set ready
CE	125	RI	9 NC	NC 9	RI	Ring indicator

Figure 5. Circuit and pin assignment

Mechanical and electrical properties.

As an instrument is to be considered as a Data Terminal Equipment, it is to be equipped with the DTE connector. According to the EIA 449 standard, this connector uses male pins and a female shell. The connector is to be labeled as "IEEE 1174".

The electrical characteristics follow the EIA 562 requirements; they guarantee for compatibility with the RS 232-D/E and CCITT V28 signals and allow for baudrates up to 64000.

Circuit operation.

The objective of IEEE 1174 is to establish a standard which allows instruments equipped with a serial interface to communicate with a controller, meanwhile maintaining compatibility with existing ATE standards. Today, commonly used standards exist for serial data communication. These standard provide protocols which meet a number of the demands for instrumentation applications. Rather than (re-)defining new protocols, IEEE 1174 makes use of existing standards and identifies particular protocols which satisfy the communication needs of modern instrumentation.

Data flow control protocols

As is explained before, to prevent input buffer overflow, instruments need a way to signal a transmitting device to stop sending data. For that purpose of IEEE 1174 uses existing data flow control protocols. Two basic methods are defined:

1. Hard flow control, making use of the RFR/CTS circuit.
2. Soft flow control, making use of the so called "XON/XOFF" protocol.

Emulation codes allow for the selection of the flow control method, as will be explained later on.

Hard flow control

This procedure makes use of the two circuits RFR (pin 7) and CTS (pin 8), as is shown in figure 6. Via the "null modem" cable , the RFR circuit is connected to the CTS circuit of the opposite terminal.

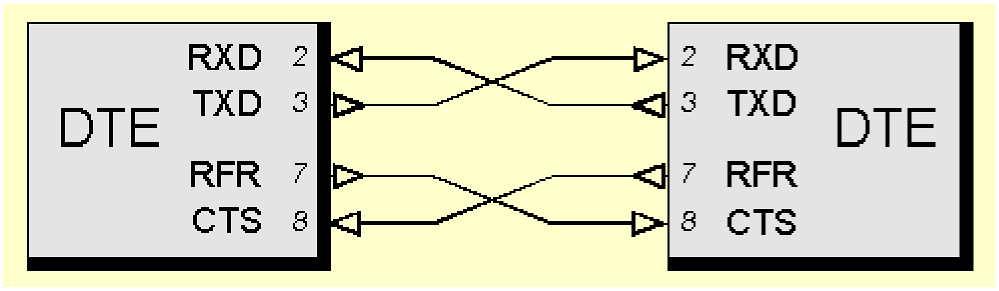


Figure 6. DTE's using hard flow control

A DTE, receiving data on circuit RXD (pin 2), will set:

RFR = ON	to signal to the opposite DTE that it is able to receive data.
RFR = OFF	to the opposite DTE that it has to stop transmitting data. However, before its input bufer overflows, the terminal must still be able to receive another 6 characters, after having switched RFR from ON to OFF.

A DTE being capable to transmit data via circuit TXD (pin 3), which senses:

CTS = ON	is allowed to transmit the data.
CTS = OFF	shall not transmit the data. When the CTS function changes from ON to OFF during a data transmission, the sending of data shall stop within 4 characters after the ON to OFF transition occurred.

Soft flow control

This type of data flow control does not make use of circuit pinning. Instead, it uses particular codes to signal the transmitting peripheral. These codes are sent over the same TXD circuits as the ordinary data. The flow control procedure which is to be used with the IEEE 1174 standards is known as the XON/XOFF protocol. XON is the non printable ASCII character *DC1* (11hex), whereas XOFF is the character *DC3* (13hex).

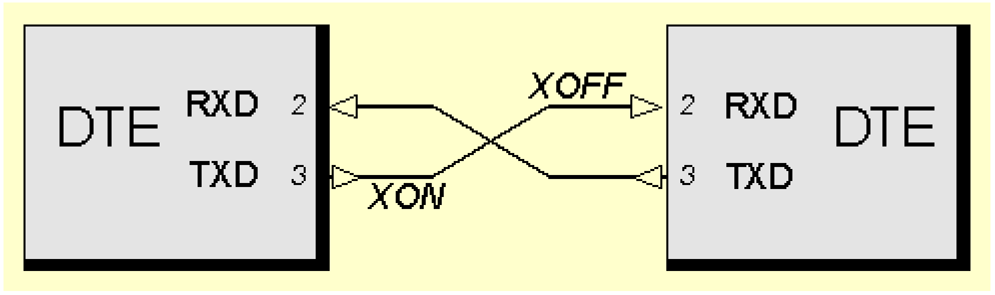


Fig. 7. DTE's using soft flow control

A DTE receiving data on its RXD circuit (pin 2), will transmit:

XOFF	via its TXD circuit (pin 3), to signal to the opposite DTE that it has to stop sending data. However, a peripheral sending XOFF must be able to accept at least another 60 characters before its input buffer overflows. Because of the overhead time which is needed for the detection of the XOFF code within the incoming data stream, this is slightly more than the 6 characters required with the hard flow control.
XON	via its TXD circuit (pin 3) when it has sufficient space again in its input buffer to resume the receipt of data via the RXD circuit (pin 2). An XON shall only be sent after the transmission has been suspended by sending an XOFF code.

A DTE, being capable to transmit data, which receives:

XOFF	via its RXD circuit (pin 2), shall not send data via its TXD circuit (pin 3). When the XOFF code is received in the middle of transmitting characters, the transmission shall be stopped within 30 characters after XOFF.
XON	via its RXD circuit (pin 2), may resume sending data via its TXD circuit (pin 3).

Because full duplex serial interfaces may send and receive data at the same time, it may occur that the XON/XOFF messages need to be transmitted within the ordinary data stream. For that reason, the XON/XOFF soft flow control cannot be used to reliable exchange binary data. Instead, the RFR/CTS hard flow control method is to be used .

Other properties and requirements of the IEEE 1174 standard.

- Devices shall be able to receive and recognize the "break" signal as defined in the CCITT V14 standard. This standard defines the "break" duration to be a 2 character sequence plus 3 stop bits or greater. At a 1200 baud rate, this duration will take about 25 ms. The interpretation of the 'break' is specified by the next part of the IEEE 1174 standard set. No protocol exist for the interpretation by devices that claim compatibility with IEEE 1174.0 only.
- Bit rates which need to be supported are 1200, 2400, 4800 and 9600 bits/second, whereas 19200 and 38400 bits/second are strongly recommended. It is allowed to support other bit rates.
- All supported bit rates need to be selectable by remote messages, as well as by local means, as for example, a front panel selection. Selected bit rates need to survive a power-on cycle.
- The data characters being transferred over the TXD and RXD circuits shall consist of 8 bits, preceded by a start bit (SPACE, 0) and followed by a stop bit (MARK, 1), to create a character frame. No parity bit is to be used.
- Framing errors and input buffer overflow errors need to be recognized. IEEE 1174.2 compliance requires these errors to be reported. Upon occurrence of such an error, all following characters are to be accepted, but discarded, until a LF (Line Feed, 0Ahex) is encountered.

GPIB function emulation. *Compliance class 2*.

The GPIB interface is one of the most popular and frequently applied interfaces in the instrumentation world. Many manufacturers of integrated circuits and hardware boards for test applications support this interface. Almost all vendors of software packages for automatic test equipment sell applications based on this interface bus. Due to its fitness for instrumentation, this bus exists already for decades and has reached an almost unassailable position in the market, where many application use this interface. It is not to be expected that the GPIB interface will disappear soon.

The application of the GPIB bus requires a special controller card, which is to be placed in the computer. Since nowadays, all PC's are equipped with a serial communication port, manufacturers of low end instruments equip their products with serial interfaces. Together with the low cost PC technology, this creates a relative cheap solution for remote instrument programming. Although these type of applications do not require the same performance as GPIB application, they lack the GPIB functionality that is added to meet the particular demands of instrument programming.

For this reason, the second part of the new standard, IEEE 1774.1, defines a technique which emulates this functionality. This allows to take advantage of the "GPIB" functionality over a serial link. Furthermore, these emulation codes are useful to minimize changes to existing application software designed to operate with GPIB. Moreover, the GPIB emulation technique is needed when IEEE 448.2 compatibility is desired.

All emulation codes start with the & character (ampersand, 26H), followed by three literals or digits. Some codes need to be terminated with a *CR* (Carriage Return, 0D`hex`) en *LF* (Line Feed, 0A`hex`) character. To unambiguously distinguish between ordinary data and emulation codes, two modes are available:

The IEEE 488.1 emulation mode	This mode requires emulation codes to be recognized and trapped anywhere in the input data stream. A "double &" mechanism is used to distinguish between the leading & of an emulation code and an ordinary & data character.
The IEEE 488.2 emulation mode.	This mode allows emulation codes to be sent anywhere in the input data stream outside binary data elements (Arbitrary Block data) and String data. Character sequences which are identical to an emulation code, and which are sent within such a data element, are considered as to belong to this data.

Service Request and Serial Polling.

The standard Service Request (SRQ) function of GPIB is used by an instrument to alert the controller that it requires service. As a result, the controller will issue a serial poll to figure out the reason for the Service Request. An instrument being polled, will send its statusbyte to the controller. The statusbyte contains information about the reason for requesting service; it is a single 8 bit code, which is defined by the GPIB standard.

The next procedure is defined by IEEE 1174 to emulated this procedure.

The device sends	&SRQ *CR* *LF*	emulating the Service Request to the controller
The controller responds by	&POL	emulating the Serial Poll of the device

sending		
The device then responds by sending	&ddd *CR LF*	emulating the serial poll response. 'ddd' is the 3 digit, decimal value of the status byte code

Table I. Service Request and Serial Poll

Device Clear

This GPIB message was originally intended to set the instrument functions into a pre-defined initial state. Since the common command *RST performs this functionality, the IEEE 488.2 standard requires the Device Clear message to initialize the communication between the controller and the device. Such functionality is needed to break any communication deadlock, when ordinary messages cannot be transferred.

The device clear functionality is to be executed when the "break" signal, as defined in the CCITT v14 standard occurs. As explained before, all IEEE 1174 device need to be able to recognize this signal. To prevent inadvertently clearing of data, the controller shall not send any data until the clearing process is terminated. Therefor, the device must send the &DCL *CR* *LF* emulation code after it has completed the clear process. This is an indication to the controller that data transfer may be resumed.

Device Trigger

The GPIB interface message Group Execute Trigger (GET) is emulated by the &GET code. Upon receipt of this code, the device shall behave as if the GET message was received. Notice that the IEEE 488.2 common command *TRG will perform the same functionality. Devices may not implement the trigger function. Since all emulation codes need to be implemented, such devices must accept the code, but shall not respond to it. IEEE 488.2 compatible instruments must generate a Command Error.

Remote Local

The GPIB Remote Local (RL) function is used by the controller to enable and disable the local controls from the front panel. Whenever a data byte is received, the RL function is automatically turned into the remote state. Once set into the remote state, instrument functionality cannot be accessed anymore via the front panel.. Otherwise the RL function is in the local state. A controller may use the emulation code &GTL to set the device in the local state again.

When the device is in the remote state, the local user may send the "return to local" message to regain control over the instrument. However, a remote controller may lock this function by sending the emulation code &LLO (Local Lockout). Unless locked, this "return to local" command, which is normally generated by pushing a front panel button, causes the device to enter the local state.

The device is automatically set into the local state when nothing is attached to the serial port, or when the remote controller is powered off. This condition is detected by the

Received Line Signal Detector (RLSD) function at pin 1.

Flow control selection

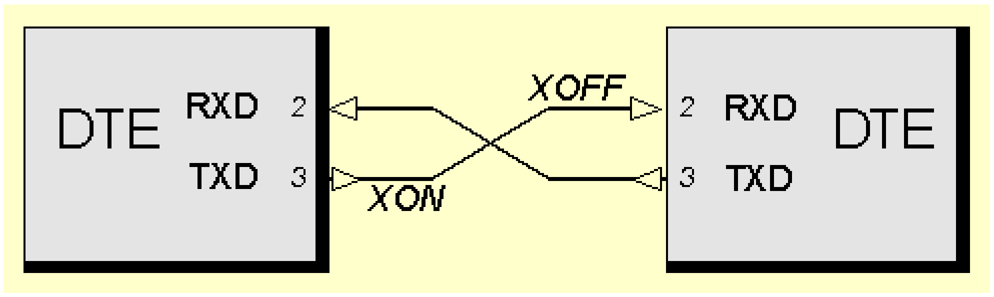


Fig. 8. Flow control selection

Three emulation codes, as shown in figure 8, are available to select the flow control method to be used.

&HFC	Enables the hard flow control
&SFC	Enables the soft flow control
&DFC	Disables flow control

IEEE 488.2 compatibility. *Class 3*

The IEEE 1174.2 part contains a number of definitions that are needed to allow serial interface to comply with the IEEE 488.2 standard. This standard provides a higher layer functionality and deals with several other aspects of the communication between the controller and the instrument. For example, the syntax for remote messages, an solid status reporting structure and a set of so called common commands, which are used for general "house-keeping" functionality, is part of the IEEE 488.2 standard. Furthermore, this standard specifies a structured communication scheme for remote messages, which is defined by the so called Message Exchange Control (MEC) protocol. It is based on the principle that an instrument may not send data until it is asked for. Therefor IEEE 488.2 distinguishes between commands and queries; commands are program messages that cause the device to perform an action, but do not result in a response from the instrument. Queries are program messages that need a response from the instrument and are used to retrieve information from the device. This concept is anchored in the language construction, since commands and queries follow different syntax's.

Figure 8. Data flow control selection

Although the concept of IEEE 488.2 goes beyond the GPIB standard, it makes partly use of its properties, definitions and terminology, which unfortunately, does not always apply to serial interfaces. A simple example will make this clear; a GPIB device can be addressed as either talker or listener. It cannot simultaneously send (talk) and receive data (listen). On the contrary, a full duplex serial interface can transmit and receive data at the same time. A signal which is associated with this functionality is the "byte request" (brq), which is generated when a GPIB interface is addressed as talker and the controller request to send a byte. Because IEEE 488.2's MEC protocol uses this brq signal, it cannot be directly be applied to serial links, since these do not have an addressing mechanism which explicitly requests for data.

The intention of IEEE 1174.2 is therefor to provide a scheme to apply the Message Exchange Control protocol to serial interfaces, meanwhile maintaining the purpose of the original IEEE 488.2 MEC protocol. Therefor, it identifies the serial interface signals which are needed to establish a proper MEC and re-defines this protocol in terms of serial interface characteristics.

Message termination

The IEEE 488.2 defined message terminator uses the End message which is sent over the GPIB's EOI line. Because serial interfaces do not provide such a facility, IEEE 1174 defines the terminator for program messages to be the LF character (Line Feed, 0AH). Because so called "white space" characters may precede the Line Feed, *CR* *LF* is an accepted terminator. The *CR* *LF* is also defined as the mandatory response message terminator for serial interfaces; devices must terminate their response with this character sequence.
