Overview of IEEE-488

INTRODUCTION

IEEE-488 refers to the Institute of Electrical and Electronics Engineers (IEEE) Standard number 488. This standard was first established in 1978, 13 years after Hewlett-Packard (HP) of Palo Alto, CA, began work to enable its broad range of instruments to communicate with one another and with "host" computers.

At the time of its development, IEEE-488 was particularly well-suited for instrument applications when compared with the alternatives. In essence, IEEE-488 comprises a "bus on a cable," providing both a parallel data transfer path on eight lines and eight dedicated control lines. Given the demands of the times, its nominal 1 Mbyte/sec maximum data transfer rate seemed quite adequate; even today, IEEE-488 is sufficiently powerful for many highly sophisticated and demanding applications.

However, IEEE-488, as originally defined, left some ambiguities in the specifics of controller-instrument interaction and communication. While these open issues were likely intended to give instrument and controller designers some latitude, the result was confusion and compatibility problems among instruments from different manufacturers.

During the 1980's, a new layer was added to the IEEE-488 standard, IEEE-488.2. The original standard was re-designated IEEE-488.1. IEEE-488.2 provides for a minimum set of capabilities among "controllers" and "devices," as well as for more specific content and structure of messages and communications protocols.

IEEE-488.2 is fully backward compatible with IEEE-488.1; the use of a "488.2"-compliant controller affords the ability to use the new protocols available with "488.2" instruments while retaining the ability to communicate with and control "488.1"-compliant instruments and associated vendor idiosyncrasies.

Today, IEEE-488 is the most widely recognized and used method for communication among scientific and engineering instruments. Major stand-alone general purpose instrument vendors include IEEE-488 interfaces in their products. Many vertical market instrument makers also rely on IEEE-488 for data communications and control.

IEEE-488 controllers support a variety of personal computers, from the IBM PC/XT/AT and PS/2 and compatibles to the multifaceted Macintosh family. Some of these controllers are plug-in cards; others are protocol converters (*e.g.*, SCSI-to-IEEE-488). All provide at least IEEE-488.1 in compliance, and a growing number adhere to "488.2."

GENERAL INFORMATION

The IEEE-488 interface, sometimes called the General Purpose Interface Bus (GPIB), is a general purpose digital interface system that can be used to transfer data between two or more devices. It is particularly well-suited for interconnecting computers and instruments.

Some of its key features are:

- Up to 15 devices may be connected to one bus
- Total bus length may be up to 20 m and the distance between devices may be up to 2 m
- Communication is digital (as opposed to analog) and messages are sent one byte (8 bits) at a time
- Message transactions are hardware handshaked
- Data rates may be up to 1 Mbyte/sec

Mechanical Specifications

CONNECTOR

The IEEE-488 connector is a 24-pin connector. Devices on the IEEE-488 bus have female receptacles; interconnecting cables have the mating male connectors. Connecting cables will typically have male and female receptacles wired in parallel at each connecting head to allow parallel connection of cables at a device and/or to allow daisychaining between devices.

INTERCONNECTION CABLING

Any individual IEEE-488 bus is limited to 15 devices including the controller. However, the IEEE-488 specification limits the total length of all cabling used to interconnect devices on a common bus to 20 m, or 2 m times the number of interconnected devices (up to 20 m). Cable lengths between devices may vary, as long as total cable length does not exceed these restrictions. Devices may be interconnected in a star or linear topology, or in a combination of the two, as long as the distance limits are observed. For maximum data transfer rates, the total cable length should be reduced to 15 m, with the average interdevice cable 1 m or less.

Electrical Specifications

BUS LINES

The IEEE-488 bus is a multidrop interface in which all connected devices have access to the bus lines. The 24 bus lines group into four categories:

- **Data Lines -** Eight lines (DIO1 through DIO8) used to transfer information (data and commands) between devices on the bus, one byte at a time.
- **Handshake Lines -** Three lines used to handshake the transfer of information across the data lines:
  DAV: Data Valid
  NDAC: Not Data Accepted
  NRFD: Not Ready for Data
- **Bus Management Lines -** Five lines used for general control and coordination of bus activities:
  ATN: Attention
  I FC: Interface Clear
  REN: Remote Enable
  SRQ: Service Request
  EOI: End or Identify
- **Ground Lines -** Eight lines used for shielding and signal returns:
  One Shield
  One General Signal Ground
  Six logic ground lines paired off with ATN, SRQ, IFC, NDAC, NRFD and DAV

HANDSHAKING

The IEEE-488 bus uses three handshake lines in a "We're ready - Here's the data - We've got it" sequence to transfer information across the data bus. The handshake protocol assures reliable data transfer at the rate determined by the slowest Listener. The handshake lines, like all other IEEE-488 lines, are active low. DAV is controlled by the Active Talker. Before sending any data, the Talker verifies that NDAC is asserted (low) which indicates that all Listeners have accepted the previous data byte. The Talker then places a byte onto the data lines and waits until NRFD is unasserted (high), indicating that all Addressed Listeners are ready to accept the information. When NRFD and NDAC are in the proper state, the Talker asserts DAV (active low) to indicate that the data on the bus is valid. NRFD is used by the Listeners to inform the Talker that they are ready to accept the new data. The Talker must wait for each Listener to unassert this line (high), which they do at their own rates when they are ready for more data. This assures that all devices accepting the information are ready to receive it. NDAC, also controlled by the Listeners, indicates to the Talker that each device addressed to listen has accepted the information. Each device releases NDAC (high) at its own rate, but NDAC does not go high until the slowest Listener has accepted the data byte. This type of handshaking permits multiple devices to receive data from a single data transmitter on the bus. All active receiving devices participate in the data handshaking on a byte-by-byte basis and operate the NDAC and NRFD lines in a "wired-or" scheme so that the slowest active device determines the rate at which the data transfers take place.

IEEE-488 FUNCTIONS

When information is placed on the data lines, it can represent either a data byte or a command. If the Attention bus management line (ATN) is asserted while the data is transferred, then the data lines are carrying a multiline command to be received by every bus device. If ATN is not asserted, then a data byte is being transferred, and only the Active Listeners receive that byte.

The IEEE-488 bus also has a number of uniline commands that are carried on a single bus management line. For example, the Interface Clear (IFC) line, when asserted, sends the Interface Clear command to every bus device, causing each to reset its IEEE-488 bus interface.

ADDRESSING

The IEEE-488 standard normally permits up to 15 devices to be configured within one system. Each of these devices has a unique bus address, a number from 0 to 30. Address limits can be circumvented directly by the use of bus expanders or indirectly through the use of an isolator or an extender.

A device becomes Addressed to Talk when it receives a Talk Address Group (TAG) multiline command (a byte transferred with ATN asserted) specifying its own address from the Active Controller. Similarly, it becomes Addressed to Listen when it receives a Listen Address Group (LAG) multiline command. Other address commands include My Talk Address (MTA) and My Listen Address (MLA), which are the TAG and LAG commands of the Active Controller. The secondary Command Group (SCG) is used to refer to subaddresses or subfunctions within a particular device. This permits direct access and control of the subdevices or subinstruments embedded within complex devices or instruments.

THE SYSTEM CONTROLLER

The System Controller, usually a computer with an IEEE-488 board installed, always retains ultimate control of the bus. When the system is first powered up, the System Controller is the Active Controller and controls all bus transactions. The System Controller may Pass Control to a device, making it the New Active Controller, which may then Pass Control to yet another device. Even if it is not the Active Controller, the System Controller maintains exclusive control of the Interface Clear (IFC) and Remote Enable (REN) bus management lines and can take control of the bus whenever it desires.

IEEE-488.2

The IEEE-488.2 standard was developed to simplify the basic process of communicating with instruments. IEEE488.2 extends the 488 standard with code, format and protocol standardization and serves to resolve issues left open in 488.1.

IEEE-488.2 details preferred implementation of many of the issues that were either optional or unspecified on the first standard. IEEE-488.1 covers the key physical issues (connector type, bus length, maximum number of instruments, etc.), electrical issues (open collector TTL, tristate) and low-level protocols (device addressing, control passing and data handshaking/timing). Four basic device functions (Talker, Listener, Controller and System Controller) are specified, as are capability subsets for each type of device.

A number of items not covered by 488.1 can cause problems for the test engineer, particularly regarding equipment compatibility and data corruption.

For example, 488.1 does not cover these specifications:

- **Minimum Device Capability Requirements**
  No minimum set of requirements is mandated in IEEE-488.1 for Talkers, Listeners, Controllers or System Controllers. Hence, a device may implement all, or only some, of the capability sets set forth in 488.1, giving rise to systems containing devices with varying levels of abilities.

  The Controller, in such a situation, has no guarantee of a basic communication subset among system devices. This can lead to confusion for the system operator and miscommunication between devices.

- **Data Coding, Formats and Message Protocol**
  Under 488.1, the messages transferred between the Controller and a device are entirely at the discretion of

the device manufacturer. The use of ASCII, binary or some other form of data code and the choice of terminators such as carriage-return or EOI is arbitrary. Also, the sequence of the sending of commands and the reading of their responses is unspecified and varies from instrument to instrument.

- **Definition of the Status Byte**
  488.1 defines a status byte and one bit within, but the meaning of the other seven bits is at the discretion of the device designer. This forces the user to provide a unique interpretation of each bit of the status byte. Also, the relationship between the status byte and the device's other internal status registers is unspecified.

DRIVER SOFTWARE FOR IBM PC

Great variety is found in the software required to complete the interface between the user's program and the IEEE instruments. Two fundamental techniques are used: the DOS device driver and the subroutine library. These are not mutually exclusive, as subroutine libraries can be implemented via a DOS device driver.

DOS DEVICE DRIVER

A popular form of device driver used by several IEEE-488 controller providers is the Terminate and Stay Resident (TSR) DOS device driver approach. In this method, the driver code is stored in memory as a TSR and waits for access by an application program, much as Borland's Sidekick waits for user "hot key" input. OMEGA's 488 driver establishes a file I/O link with DOS, just as DOS provides file I/O links for system devices such as the keyboard/screen, printer or serial port.

These DOS I/O files may be accessed directly from DOS, from programs with file I/O capability, including spreadsheets such as Lotus 1-2-3 and Borland's Quattro, and from most programming languages. These files provide a direct link to the IEEE-488 bus using HP-style English language commands. This style of Applications Program Interface (API) is often referred to as Character Command Language (CCL), as the IEEE commands are sent as ASCII strings to the driver via the API's file I/O links through DOS.

Controlling Instruments from Any Language

Just as DOS and spreadsheets can access IEEE instruments directly using the file I/O services provided by DOS for device drivers, most programming languages also can use file I/O to quickly and easily access the IEEE-488 bus.

SUBROUTINE IEEE-488 DRIVER INTERFACE

An alternative means of controlling the IEEE-488 hardware is via subroutine calls from high level languages. This method has the advantage of minimizing the overhead of DOS device driver services and the ASCII message (CCL) parser and interpreter. Disadvantages include the loss of the convenience and effectiveness of accessing the IEEE-488 bus from a wide variety of applications programs, as well as from DOS. Also, the use of subroutines, even those with easy-to-use HP-style commands, typically requires compiling and linking to run even simple test codes.

Some IEEE controller implementations on the IBM PC give the user the choice of subroutine calls or CCL.

IEEE-488 SUBROUTINE CONTROL LIBRARIES

The logical complement to subroutine interfaces for a TSR DOS device driver are subroutine libraries that directly access the IEEE-488 hardware from a high-level language with code that is compiled and linked directly into the user's program. This approach eliminates the DOS device driver, integrating the IEEE-488 control functions directly into the applications program code. This method has the potential for the highest performance, as it eliminates possible DOS effects on the speed of commands and data.

MICROSOFT WINDOWS COMPATIBILITY

The growing popularity of the Windows 3.0 Graphical User Interface (GUI) is rapidly spreading to test and measurement applications. Until 1991, few tools were available for the end user to build Windows applications. Now, tools such as Microsoft's Visual Basic and Borland's C++ provide GUI development interfaces that allow users to draw windows and fill them with buttons, scroll bars and dialog boxes. Soon, these tools (and the tools, libraries and utilities that follow) will be widely used by developers of IEEE-488 test programs. IEEE-488 controller package vendors will adapt their offerings to be compatible with Windows, so users will be able to apply Windows solutions to their measurement problems. As these new Windows-oriented drivers and packages debut, there will undoubtedly be a broad range of solutions offered to the end user. It is important to know and understand what makes Windows and Windows applications different from DOS, and what features an IEEE-488 driver should have in order to make the most of the Windows environment. Users should keep the following issues in mind when reviewing new offerings:

- Is the software written as a Windows application, or is it merely a port of DOS software?

Windows performs its own memory management functions; typical DOS ports to Windows do not permit Windows to dynamically allocate memory use, which can lead to "Unrecoverable Application Errors."

As Windows is an event-based system, it provides extensive event handling facilities; Windows applications should take advantage of them.

Windows has no equivalent of the TSR concept used with DOS. Although some DOS TSR's will function while Windows is running, their operation can be erratic and unpredictable.

- Will the driver support concurrent access of different peripherals on a single interface by multiple Windows applications? Windows' pseudo multitasking is one of its reasons for being.
- Will the driver service multiple bus adaptor boards?
- Is the driver IEEE-488.2 compliant?