IEEE 488 Troubleshooting

IEEE 488 Systems and Software

Application Note #6

To efficiently diagnose, troubleshoot and verify IEEE 488 systems, you should first have some basic knowledge of the IEEE bus. Since the hardware portion of the IEEE standard is rigorous and stable, most of the problems you will encounter during the system integration process will be in the application software. This note contains a brief IEEE tutorial followed by troubleshooting techniques.

The IEEE 488 Bus Standard

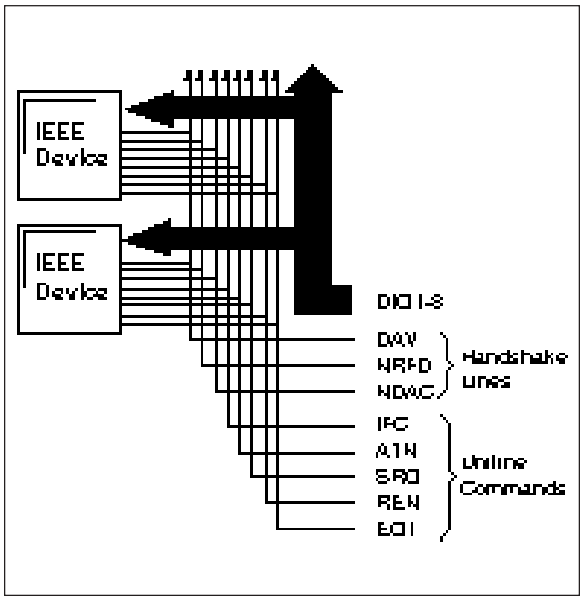
Addresses

Each device on the IEEE 488 bus has a unique address — even the controller. The addresses range from 0 to 30. Addresses are the means by which controllers select specific instruments. To send data to a device, the controller must both address itself to talk and address the device to listen.

Commands

IEEE 488 recognizes two types of commands: Device-Dependent Commands (DDCs) and IEEE 488-specific commands. DDC's such as "R0F0T2X" are simply data sent from one device to another (in this instance from the controller to an instrument). IEEE 488-specific commands come in two forms; multiline commands and uniline commands. Multiline commands are sent on the data bus lines; uniline commands are individual signals on the bus.

The uniline signals are the easiest to decipher because each signal has one specific purpose. All but two of these signals are issued exclusively by the controller.



Uniline Commands

- Interface Clear (IFC) is the most dramatic uniline command. It stops all activities on the bus and returns the interface of every device to a quiescent state.
- Remote Enable (REN) informs the devices on the bus that the IEEE interface is active. It does not lock out an instrument's front panel.
- Attention (ATN) is the signal that differentiates data from multiline commands on the data bus. When ATN is asserted by the controller, the bits on the data bus are actually multiline commands being issued to all of the devices on the bus.
- End Or Identify (EOI) can be issued by any device talking on the bus. Talkers use EOI to notify listeners that the end of the transmission has taken place.
- Service Request (SRQ) can be issued by any device on the bus. It allows devices on the bus to interrupt, or alert, the controller to an internal situation that needs servicing (e.g. "my buffer is full," "I've encountered an error," "my trigger has been satisfied").

Multiline Commands

Now that you have seen the uniline commands and have a sense for their application, we'll discuss multiline commands. Remember, multiline commands are transmitted from the controller to the devices on the data bus. The devices know that they are not data because the ATN line is asserted. When ATN is asserted by the controller all of the devices must listen to the commands.

Multiline commands serve several functions, most notably to address instruments to talk or listen. To address a device to listen, the controller will assert ATN and place the listen address of the selected device on the data bus. There are 31 listen addresses. These are called the Listen Address Group (LAG). A similar process is used for the Talk Address Group (TAG).

Most IEEE drivers, including IOtech's Driver488 for our line of IEEE controllers, have high level commands that perform several elemental IEEE 488 operations. In the IOtech Driver488 manual, every command explanation contains a field called BUS STATES. In BUS STATES, a complete explanation of what is happening on the bus is displayed. For example, let's examine IOtech's Driver488 command ENTER, which simply gets one reading from a specified device:

BUS STATES:
ATN•UNL, MLA, TAG, *ATN, data..., ATN

First, this indicates that ATN is asserted. Next, the multiline command UNListen (UNL) instructs all devices that were in the listen state to exit that state. The controller then issues My Listen Address (MLA), its own address in the listen address group, and issues the Talk Address Group (TAG) for the specified device. Next, it unasserts ATN, which notifies the addressed device that it may now transmit its data. Finally, after the data has been sent (perhaps ending with an EOI), the controller once again asserts ATN.

Analyzing the IEEE Bus

The simplest way to decipher the controller's operations and the reponse of the instruments, regardless of what software or hardware you are using, is with an IEEE analyzer. Analyzer488 from IOtech allows the programmer to view all of the transactions on the bus in real time or to record them into its 32K non-volatile transaction buffer for later inspection.

The following example problems are all diagnosed using the Analyzer488. Analyzer488 can be operated as a portable bench-top analyzer from its easy to use keypad, or from the included Analyst488 PC and PS/2 software. Analyzer488 allows the events on the IEEE bus to be monitored, stored and analyzed. It can also be used to control devices on the bus for exercising and verifying instrument operation. The Analyzer488 will automatically translate the state of the data bus and control lines into easy to read IEEE messages or ASCII equivalents like SPE, TAG16, CR, and LF. Along with its large capture buffer, Analyzer488 contains a comprehensive set of trigger features that allow the desired group of transactions to be easily pinpointed and identified.

Common Problems and Solutions

Occasionally systems will encounter problems due to the interaction of several devices in the system. These are among the most difficult problems to debug. You should connect an Analyzer488 and let it run while the application is processing. Recording the bus transactions as they occur and inspecting the transactions one at a time will usually allow you to diagnose these types of problems rather quickly.

Often the problems encountered in a system are due to interactions between one device and the controller. Here is a list of common symptoms and their suggested solutions:

"I get a time-out error whenever I try to send device-dependent commands to my instrument."

The first thing you should check is the setting of IEEE addresses. Every device on the bus must have a unique address between 0 and 30. When sending Device-Dependent Commands (DDCs) to an instrument to change its state or operating mode, the device will first be addressed to listen, then the data will be sent. If the device has TALK and LISTEN indicators on its front panel, you can tell immediately if the address used by the controller matches the actual address of the instrument. If the LISTEN indicator does not come on when sending commands to the device, you are probably using the wrong address for that device.

As we mentioned in the tutorial section of this note, when the ATN line is asserted by the controller all of the instruments on the bus will handshake with, and accept data from, the controller. After the time-out is received, step through the transactions recorded by the Analyzer488. If no instrument addressing commands such as Listen Address Group 16 (LAG16) were recorded, your instrument is probably off or broken, or the cable is disconnected. Regardless of the present state of the instrument, it should handshake (accept data) when the ATN line is asserted. If the addressing commands were successfully recorded on the analyzer, step through the transactions until the ATN line is unasserted. If there are no more recorded transactions, then no instrument was placed in the Listen mode. The controller had no one to handshake with so it "timed-out." Your instrument is probably set to the wrong address.

"At certain points in my program, the system stops and I receive a time-out error."

If portions of your program are operating correctly, then you can be certain that your addresses are set correctly. If you encounter a time-out error in your program after other instrument tasks have been completed successfully, you may have encountered an instrument-readiness problem.

IEEE interfaces and software like IOtech's Personal488 operate very rapidly and can sometimes out-run the instrument they are controlling. For most instruments, data requests are performed in two steps: sending the necessary setup or inquiry commands via DDCs, then addressing the device to Talk. It is possible to issue the necessary commands to request the data from the instrument and then address it to Talk long before it is prepared to supply the requested data. Many instruments will simply pause the bus until they have prepared the data to send. However, other instruments react poorly by "hanging up."

To check for this "outracing" condition, place the Analyzer488 into the Slow Handshake mode. This will effectively slow the transaction speed of the bus to a rate set by the Analyzer488. If the data request takes place successfully, it is probably an "outracing" condition.

"My instrument seems unaffected by the commands I send to it."

If you have already made certain that you are sending the commands to the right instrument address, you may have left off a crucial piece of information that instructs the instrument to process the commands.

IEEE systems usually use data delimiters called terminators. A Talker will inform a Listener that the data string has come to an end by appending a predefined terminator to the end of its data string. Although terminators are issued solely by the talking device, the listening device(s) must know what terminator to expect. Usually IEEE instruments will issue a carriage return (CR) and a line feed (LF) as their terminator. Some instruments will not process the incoming command string until they detect the proper terminator. You should step through the transactions captured by the Analyzer488 to verify the transmission of the terminator, then make certain that it agrees with the terminator expected by your instrument.

Some instruments have a DDC which instructs the instrument to process all of the previously received commands. This EXECUTE command (typically a character like 'X') allows a programmer to send several commands to an instrument in any order over any length of time and then execute them all simultaneously within the instrument. If the EXECUTE DDC is not sent, the state of the instrument will not change. It will react as if the commands were never received.

"When I ask for data, nothing is returned."

This could be an address or terminator problem like the ones discussed above. See the previous sections to diagnose these problems.

Not all instruments are ready to supply data whenever you ask. Some instruments have nothing to say until they are commanded to acquire or generate data. Some data acquisition instruments have triggering features which allow the instrument to collect and transmit data only after a specified event has occurred. A typical multimeter might have a default trigger of TRIGGER ON TALK which would enable the multimeter to take a reading every time the controller addressed it to Talk. If the same multimeter was set to TRIGGER ON GET, no reading would be available until the controller issued a Group Execute Trigger.

If the device has no data to give, the Analyzer488 will show that the controller has been addressed to Listen and the device was addressed to Talk and then the process stopped. The handshake indicators show that Not Ready For Data (NRFD) was unasserted by the controller but the instrument never asserted DAta Valid (DAV). Make certain that your device has data to transmit before you ask it for some.

IOtech's Driver488 has the capability of assigning a time-out value to the system. If an instrument does not respond within the specified time-out, the process is aborted. In some instances, an instrument may be unable to respond within the specified time-out period and the time-out period will have to be increased.

"When I ask for data, bad data is returned."

Many times the variability of data formats of an instrument will cause problems. Devices can transmit data in binary, ASCII, BCD, packed BCD, or anything else that will fit into 8 bits. Data terminators can be EOI, a byte count, or imbedded characters like CR LF. Data can be sent with prefixes, suffixes, or full headers. IOtech's Driver488 can account for all of these parameters, but some other drivers may not allow this level of flexibility.

When using higher level software packages, the problem of data formats may be impossible to overcome. Usually, menu-driven and turnkey packages go to great lengths to hide the IEEE bus from the operator. The documentation, therefore, makes no attempt to inform the operator of what is actually happening on the bus.

You may encounter a problem if your instrument transmits data in a format that is not recognized by your software package. Check your instrument manual for data format characteristics. Does your instrument transmit non-numeric prefixes or suffixes; is the data in binary or ASCII? Some software drivers will automatically throw away any non-numerics. Others do not. Even if your software throws the non-numerics away, you may encounter problems with instruments that transmit numbers like channel tags in their data prefix.

Most instruments, including IOtech's ADC488 analog to digital data acquisition instrument, can be programmed to adjust their data format for software compatibility. Analyzer488 allows you to quickly inspect the data being transmitted by your instrument, enabling you to make the proper adjustments in your software.

"An SRQ from an instrument sometimes causes a catastrophe."

The asynchronous nature of instrument interrupts can sometimes cause elusive problems. The best way to attack a problem like this is to start the Analyzer488 recording and just let it and the system run. Analyzer488 has a large 32K transaction buffer that is configured in a circular fashion. After 32K transactions have been recorded, new transactions will overwrite the oldest transactions. There is a very high probability that the events leading up to the system "crash" will still be in the recorded memory (not overwritten) after the system has locked-up. Stepping backwards in memory can usually uncover the sequence of operations that caused the problem. The Analyzer488 can also be set up to trigger on the occurrance of one or several SRQs with both a post and pre-trigger assigned. In this way a specified number of events can be captured before and after the occurrance of an SRQ. The Analyzer488 also has comprehensive search features allowing the capture buffer to be scanned for all of the occurences of any event, including an SRQ.

Some instruments have the capability of generating an SRQ for any of several internal events. Usually an SRQ mask is sent to the instrument to instruct it to generate an SRQ for only a selected subset of those events. Some instruments, by default, will interrupt the controller with an SRQ when an internal error is encountered and not respond to any further bus transactions until the interrupt is serviced. The next time your application program requests data from that instrument, your system will fail. By inspecting the Analyzer488 transaction recording working backward from the end, it will be obvious that an SRQ was asserted by someone on the bus and that it remained unserviced.

"My system occassionally locks up."

This is another of those intermittent problems that can take a long time to troubleshoot, especially if the mean time between failures is several hours, days or months. As before, the best way to approach the problem is to allow the Analyzer488 to record all of the transactions occurring on the bus. When the number of transactions goes beyond 32,767, the capture pointer will wrap around and continue to record. The last 32,767 transactions will always be stored in memory. When the system crashes, the processing of IEEE bus transactions will probably end also. With the last 32K transactions captured in memory, it is easy to step back through the capture buffer and decipher the sequence of operations that caused the crash.

One possible cause for an intermittent crash problem is the asynchronous occurance of SRQs as discussed above. There may be areas in your application program that do not react well to being interrupted. Since the SRQ can happen at any time, it may or may not occur during the processing of this sensitive area. But the longer the system runs, the probability that the SRQ will happen at exactly the wrong time increases. A sensitive area may be a part of your code that uses a group of closely related variables that are modified by the SRQ handler. For example, three IEEE 488 counters are used to take readings from three motion encoders. Each counter is programmed to generate an SRQ when its count reaches 256. The SRQ handler reads all three counters and stores them into three separate variables used later by the main program. The main program has a loop which reads the three variables, combines them with some calculation, and sends commands to a motor controller. If the main program was in the process of using the variables and an SRQ occurred (which modifies all three variables), the main program may end up using one old value and two new ones in its calculation.

One way to avoid this kind of problem is to disarm the automatic SRQ vectoring during the processing of sensitive program areas. IOtech's Driver488 has several means by which to arm, disarm and synchronize the servicing of SRQs to your program.

Another source of system malfunctions is from the instruments themselves. Most of today's complex instruments are microprocessor controlled. The internal processor handles the collection of data, the changing of programmable states, the monitoring of trigger events, and the communication on the IEEE interface. These instruments are actually computers, prone to all of the same problems as any other computer.

It is possible that your instrument reacts improperly to a perfectly good application program. The transaction report that Analyzer488 prints out can be used to communicate instrument problems to the manufacturer. The report is easy to read and concisely describes the operation of the controller and the response of the instrument.