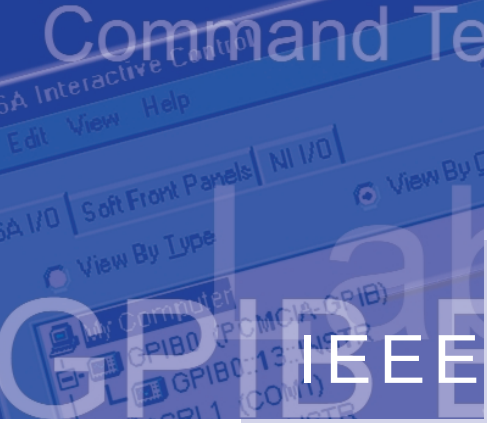
Anritsu

Communicating with Anritsu's IEEE-488.2 Non-Compliant Instruments

Application Note



IEEE-488.2 Non-Compliance

Introduction

The purpose of this application note is to recognize the difficulties associated with remotely controlling some of Anritsu's instruments via the GPIB bus. This note will define the exact cause of the difficulties and then describe a way to reliably communicate with these troublesome models.

Problem Statement

GPIB Bus

Most Anritsu instruments can be remotely controlled by a computer using the GPIB bus. The GPIB bus has been a mainstay in the Test & Measurement world since 1965 and was standardized by publication of the IEEE-488 specification in 1975. There have been two major enhancements to this standard, with the latest being IEEE-488.2 in 1987.

The GPIB bus is used to send a series of 8-bit bytes between instruments or between a computer and an instrument. Most often the bytes transmitted are standard ASCII letters and numbers. For example, if a computer wants to set the frequency in a signal generator to 12345 Hz, the computer might send the command "FREQ 12345". Upon receipt of this command, the signal generator will immediately switch to the new 12345 frequency. That sounds simple enough, but how did the instrument know when it had received the entire command? Why didn't it switch to a new frequency of 123 after receiving the characters "FREQ 123"? How did it know to wait until the "5" was received? This raises the whole issue of command termination on the GPIB bus.

Command Termination

One method of signaling the end of a command is to add one additional special character, known as the EOS (End of Statement) character. The most common EOS is "new line", otherwise known as "line feed" or ASCII 0x0A. The receiving instrument can simply continue to accept bytes until it receives the designated EOS character, and then act on all of the preceding bytes. The EOS technique is the easiest to implement by both the transmitting and receiving devices, but it is slightly inefficient because an extra character must be transmitted with each command. The EOS technique can also become a problem if the data to be passed over the GPIB bus includes the EOS byte. For example, reading a long series of binary bytes from an A-to-D converter may eventually hit upon all possible binary values, including the special EOS value. This would cause the transmission to be prematurely terminated.

The alternative to using a special termination character is to use a termination signal outside the normal path of the data bytes. The GPIB bus is defined as 24 parallel lines, of which 8 lines carry the data bytes. One of the remaining 16 lines is the EOI (End or Identify) line.

The EOI line is always in one of two states; asserted (ground) or unasserted (+5 volts). Normally EOI is unasserted. However, a GPIB transmitter may elect to signal the end of a command string by asserting the EOI line at the same time as it is transmitting the last byte of the command. The EOI termination method is more difficult for the sender and receiver because 9 lines must be controlled or monitored (8 data lines plus EOI). By not sending the EOS character, the transmissions are slightly faster and there are no concerns about the data bytes matching a special EOS value.

The IEEE-488.2 standard mandates that GPIB commands must be terminated in one of three ways:

1. Line feed as the EOS character, with no EOI.
2. EOI during the last command character and no line feed.
3. Both line feed and EOI, meaning EOI is asserted during the line feed.

The IEEE-488.2 standard clearly states that a compliant device that is receiving commands from the GPIB bus must be able to accept commands no matter which of the three termination methods are used. Similarly, a compliant device transmitting commands on the GPIB bus must use one of these three termination methods and any one of the three methods is equally acceptable.

A customer desiring to control one of our instruments via the GPIB bus can use any one of many different programming languages or manual utilities. These tools allow the customer to send and receive GPIB commands to a device that is presumably compliant with the IEEE-488.2 standard. Most, if not all, of these tools default to use of termination method (2) because it is the most efficient and eliminates potential data conflicts. Some, but not all, of the tools can be customized to allow the use of termination methods (1) and (3).

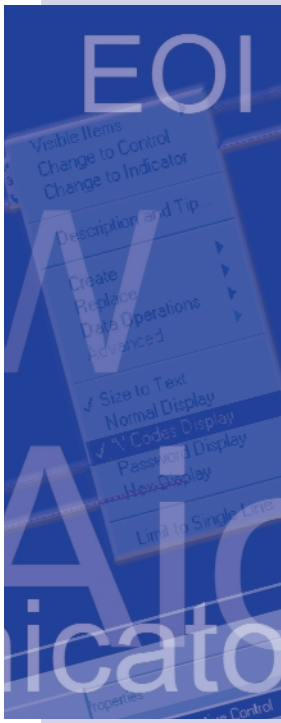
Non-Compliance

The fundamental problem that people have in controlling a few Anritsu instruments arises from the fact that those instruments do not fully comply with the IEEE-488.2 standard. In particular, the instruments ignore the EOI line. Consequently, they are unable to receive a command terminated using method (2), the most commonly used method. Such non-compliant instruments are, however, able to reliably receive commands using methods (1) and (3) because in both cases a line feed is transmitted.

A typical user will connect their computer to an Anritsu instrument along with maybe a few other instruments, start up a GPIB utility such as National Instruments' MAX (Measurement and Automation Explorer), and attempt to communicate with the instruments. Most often everything works exactly as expected. Unfortunately, a few Anritsu instruments will not work as expected. The utility will recognize that the instrument is on the bus at a particular address and the instrument will go into Remote mode and accept characters. But the instrument will never execute the command.

A non-compliant instrument will properly receive and store away every character of the command and will continue to do so until it receives the terminating line feed that it is expecting. The transmitter, on the other hand, sends all the characters, asserting EOI on the last character, but because EOI was ignored by the non-compliant instrument, the instrument ends up waiting forever for a line feed and the customer ends up frustrated.

The typical user of our instruments understands the GPIB bus enough to know that ASCII characters are transmitted back and forth, but they often don't understand the concept of termination methods. Even if the user does understand termination methods, the Remote Control Manuals for the few non-compliant instruments still may not be helpful.



The manual for every Anritsu instrument designed in the past decade will claim that it is fully compliant with the IEEE-488.2 standard and in most cases, that is correct. Some non-compliant instruments will claim they are fully compliant with IEEE-488.2, but a few pages deeper in the manual will describe how the instrument really only accepts termination methods (1) and (3). Some non-compliant instruments will claim they are fully compliant with IEEE-488.2 and proceed to describe how all three termination methods are accepted, but actual use of the instrument shows that method (2) is not accepted.

The following are a few instruments currently known to be non-compliant: MF9619, MP1555, MP1763, MP1764, MN63, and the MP1570.

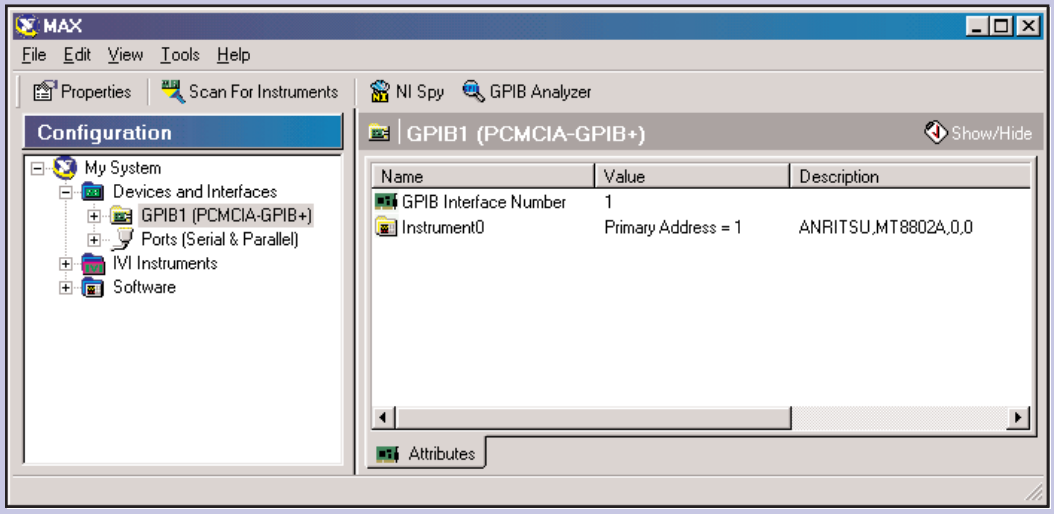
The natural question of, "WHY are these instruments non-compliant?" is beyond the scope of this paper and would be better addressed to the Anritsu divisions responsible for designing the instruments.

Solution

While frustrating at first, these few non-compliant instruments are perfectly reliable and will flawlessly receive, execute, and respond to GPIB commands as long as the commands are terminated with a line feed. The manner in which you append the line feed depends on the tool you are using for communication.

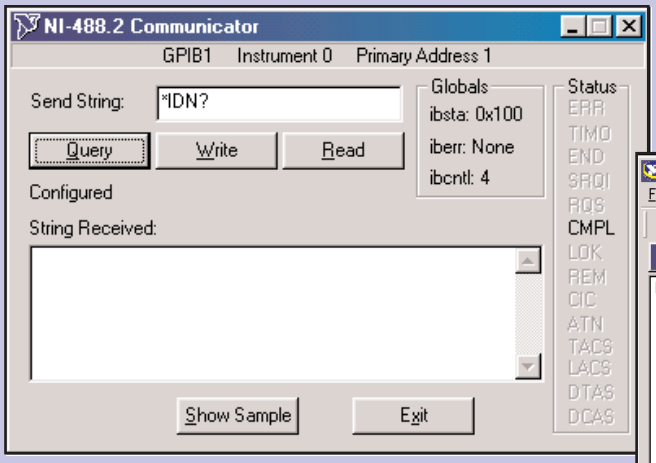
NI-488.2 Communicator

National Instruments is the largest manufacturer of GPIB cards in the US. Every card comes with a software utility called MAX (Measurement & Automation Explorer) as shown in Figure 1.

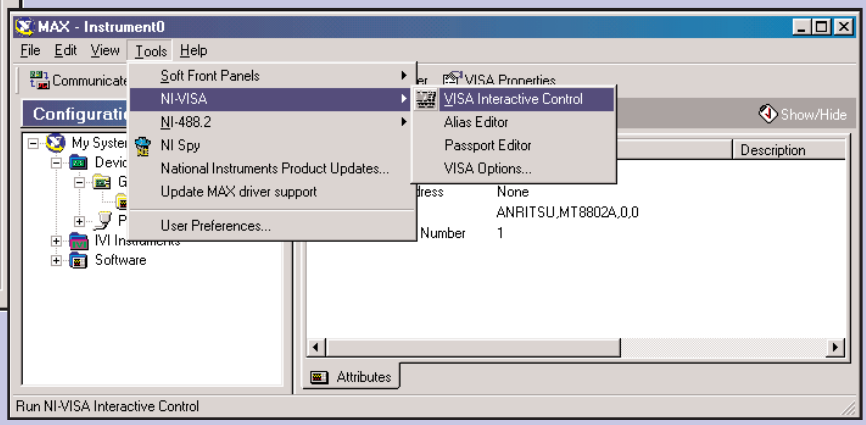


(Figure 1)

(Figure 2)



(Figure 3)



The upper left corner of MAX has a button labeled "Communicate with Instrument". When this button is clicked, the "NI-488.2 Communicator" application pops up as shown in Figure 2. This is the utility our customers most commonly use to manually control our instruments. For any device that fully complies with IEEE-488.2, this is a very easy utility to send and receive GPIB commands.

The termination method employed by NI-488.2 Communicator is EOI-only, method (2), thus Communicator will not communicate with our few non-compliant instruments. Furthermore, there is no way to force Communicator to send a line feed.

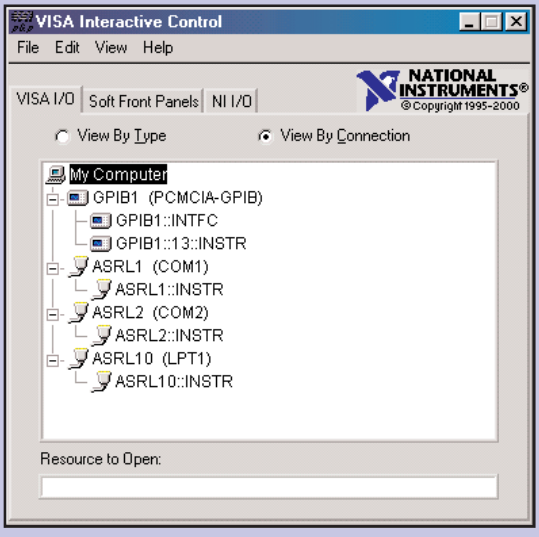
(In January 2001, National Instruments acknowledged that this was a bug and promised that some later version of Communicator would allow line feeds.)

Fortunately, MAX includes another GPIB tool which can send line feeds.

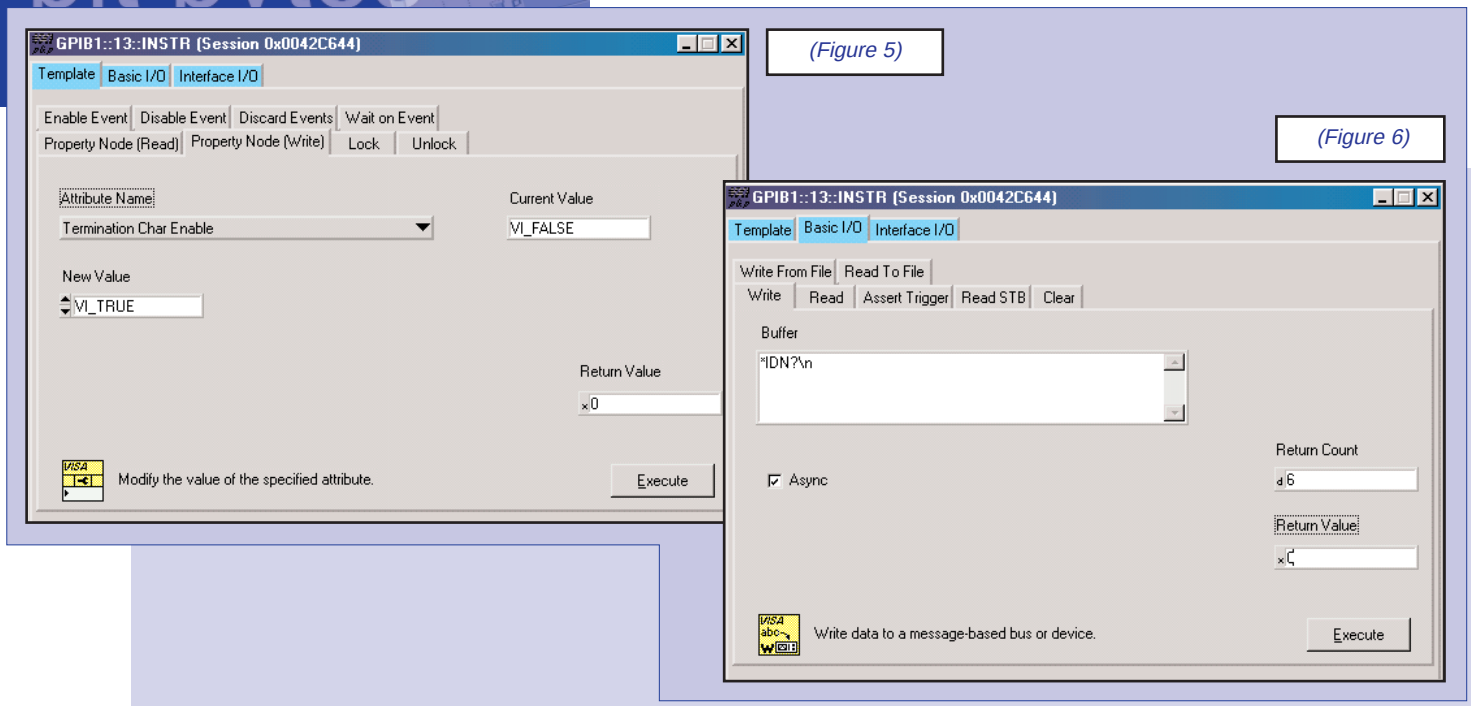
VISAic

VISA (Virtual Instrumentation Systems Architecture) is a driver software architecture developed by National Instruments to unify instrumentation software. VISAic (VISA Interactive Control) is a predecessor to NI-488.2 Communicator and can be invoked from MAX as shown in Figure 3.

The first screen to appear when VISAic is invoked will be similar to Figure 4. VISAic automatically scans your entire computer for all forms of I/O, typically finding a couple RS-232 ports, a printer port, and any instruments that may be connected to your GPIB bus. In the following example, one GPIB device is found and that device has an address of 13.



(Figure 4)



(Figure 5)

(Figure 6)

To communicate with a particular device, such as the one shown in Figure 4 with a GPIB address of 13, the user double-clicks the "GPIB1::13::INSTR" line. After doing so, the screen as shown in Figure 5 appears.

The blue "Template" tab exposes all of the attributes of the GPIB interface, only one of which must be changed to deal with Anritsu's few non-compliant instruments. The viSetAttribute tab contains a pull-down list of Attributes that can be selected, including the VI_ATTR_TERMCHAR_EN attribute. This attribute defaults to VI_FALSE and must be changed to VI_TRUE.

Our non-compliant instruments are equally non-compliant whether transmitting or receiving on the GPIB bus. The instruments ignore the EOI when receiving a command, thus the computer must append the line feed termination character. Similarly, when the instrument is responding to a query, the instrument does not assert EOI on the last character and instead appends a new line character (sometimes a carriage return character precedes the terminating line feed). In its default mode, VISAic presumes the instrument will be ending its transmissions with EOI.

By setting the VI_ATTR_TERMCHAR_EN attribute to true, you are enabling the termination of character reception when a particular termination character is received.

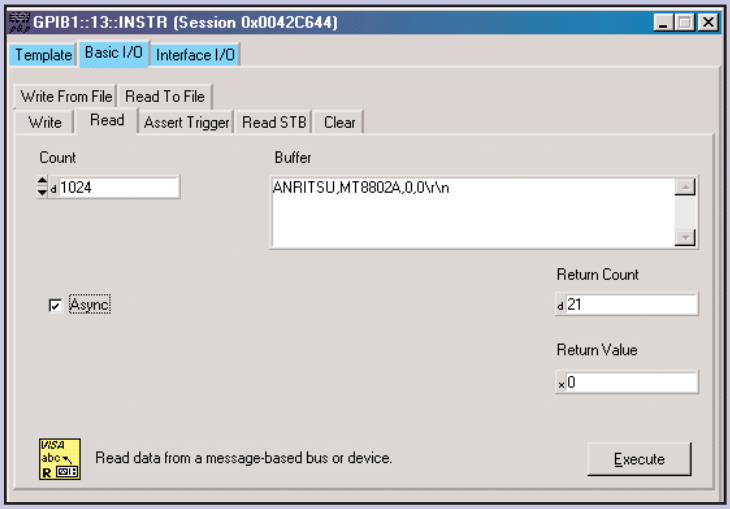
One other attribute, VI_ATTR_TERMCHAR, defines the EOS character, but since it defaults to the desired line feed (0x0A), it need not be changed.

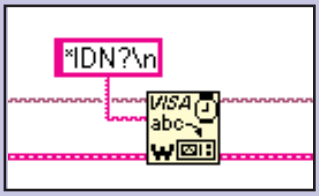
After the attributes are properly set, click on the blue "Basic I/O" tab, and then select the viWrite tab. The VISAic screen will appear as Figure 6.

In the input text field named Buffer, type in the command you wish to transmit to the instrument. A very common command to use is "*IDN?" which is defined in the IEEE-488.2 standard as an identification request. When the instrument receives this command, it should respond with a character string identifying the instrument's manufacturer and model number. The screenshot above shows the crucial command termination character.

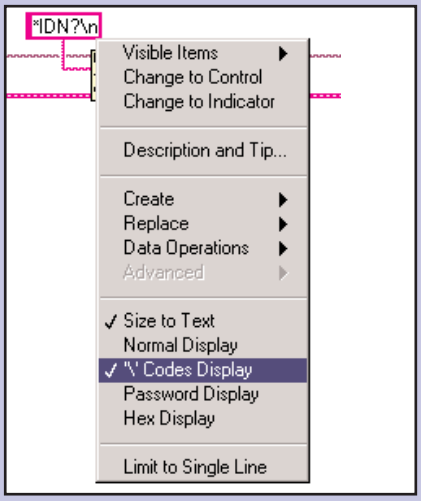
Following the "*IDN?" characters are the two characters "\n". Veteran C++ programmers will recognize these two characters, backslash followed by lower case "n", as meaning new line.

(Figure 7)





(Figure 8)



(Figure 9)

When VISAic transmits the characters entered into the Buffer field, it interprets the two "\n" characters as a single new line (0x0A) character. *(C++ and VISAic also interpret "\t" as a tab character and "\r" as a carriage return character.)* After entering the desired command and new line into Buffer, the string is transmitted to the instrument by clicking on the Execute button. The screenshot shown previously was saved immediately after clicking on Execute and the Return Count field shows the number of characters transmitted. Note that 6, not 7, characters were transmitted, indicating that the "\n" pair was indeed transmitted as a single new line character.

To read the response from the instrument, click on the viRead tab and then click the Execute button. The screen will appear as Figure 7 with the identification string from the connected instrument.

Note that the string received from the instrument is shown to have a "\n" (new line) as the final character.

LabVIEW

When sending commands to a non-compliant instrument using the LabVIEW language, the line feed character is appended to the command string as "\n" as shown in the Figure 8 graphic.

LabVIEW string constants default to "Normal Display" mode in which the "\n" pair would be transmitted as two characters.

If you right-click on the string constant, as shown in Figure 9, a pop-up menu will allow you to change the string from "Normal Display" to "\' Codes Display" in which the "\n" pair is interpreted as the single desired new line character.

Summary

The difficulties communicating with a few non-compliant Anritsu instruments can very easily be resolved by simply appending a new line character to every command and by allowing responses from the instrument to be terminated by a new line. Once these two rules are understood and followed, the non-compliant instruments are just as reliable and communicative as our compliant instruments.

Anritsu
Discover What's Possible™

ANRITSU COMPANY
1155 East Collins Boulevard
Richardson, TX 75081

http://www.us.anritsu.com

SALES & SUPPORT

UNITED STATES

Tel: 1-800-ANRITSU
Fax: 972-671-1877

CANADA

Tel: 1-800-ANRITSU
Fax: 613-828-5400

SOUTH AMERICA

Tel: 55-21-527-6922
Fax: 55-21-537-1456

JAPAN

Tel: 81-3-3446-1111
Fax: 81-3-3442-0235

ASIA-PACIFIC

Tel: 65-282-2400
Fax: 65-282-2533

EUROPE

Tel: 44-(0)-1582-433433
Fax: 44-(0)-1582-731303

June 2001
P/N : 80202-00157
Printed in USA