1394TA IICP488
Draft Specification for IEEE-488 Communications Using the Instrument & Industrial Control Protocol over IEEE 1394

Draft 1.00RC2 (release candidate)
August 2, 1999

Sponsored by:
Instrumentation and Industrial Control Working Group (II-WG) of the 1394 Trade Association

Approved for Release by:
This document has not yet been approved for release by either the II-WG or the 1394 Trade Association

Abstract:
This document describes the use of the 1394 Trade Association Instrument and Industrial Protocol (IICP) to communicate IEEE 488.1 and IEEE 488.2 messages and command/control sequences using the IEEE 1394 bus.

Keywords: protocol, instrument, 1394, GPIB, 488, 488.2, 488.1, SCPI

1394 Trade Association
2350 Mission College Blvd. , Suite 350 , Santa Clara, CA, 95054
http://www.1394TA.org

1394 Trade Association Specifications are developed with Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394 computer products. Participants in working groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is approved and issued is subject to change, brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association initiates action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
Regency Plaza Suite 350
2350 Mission College Blvd.
Santa Clara, California 95054, USA

Introduction

There is a need to connect electronic instrumentation and control devices to the high-speed, industry standard IEEE1394 interface. This document describes the operational details of a controller and device using the 1394 bus to communicate with a paradigm similar to GPIB (IEEE 488.1 and IEEE 488.2).

Committee Membership

Chairman: Andreas Schloissnik
Company: 3A International
Email: aschloissnik@3a.com
Phone: (602) 437-1751

Co-Chairman: Gary Sakmar
Company: Keithley Instruments
Email: gsakmar@keithley.com
Phone: (440) 542-8016

Editor: Andy Purcell
Company: Hewlett-Packard
Email: andyp@lvld.hp.com
Phone: (970) 679-5976

II-WG Reflector: '1394-ii@1394TA.org'

Table of contents

List of Figures

List of tables

1. Overview

1.1 Scope

This specification was developed with the following goals:

- Define how to send and receive GPIB data mode messages.
- Define how to send and receive GPIB command mode messages.
- Minimize the effort of adapting an instrument currently equipped with the GPIB interface to being a device capable of using 1394.
- Minimize the effort to make existing GPIB applications work with 1394 equipped instruments.

2. Reference and related documents

- 1394TA Instrument and Industrial Control Protocol (IICP)
- IEEE Std. 1394-1995, IEEE Standard for a High Performance Serial Bus.
- ANSI/IEEE Std. 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation.
- ANSI/IEEE Std. 488.2-1992, IEEE Standard Codes, Formats, Protocols, and Common Commands for use with ANSI/IEEE Std 488.1-1987.
- VXI-11, TCP/IP Instrument Protocol and Interface Mapping Specifications. Published by the VXI Consortium.
- SCPI-1994. SCPI Consortium, 8380 Hercules Drive, Suite P3, La Mesa, CA 91942

3. Document definitions and notation

3.1 Word usage – shall, should, may, can

The word ***shall*** is used to indicate mandatory requirements strictly to be followed in order to conform to the specification and from which no deviation is permitted.

The word ***should*** is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited.

The word ***may*** is used to indicate a course of action permissible within the limits of the specification.

The word ***can*** is used for statements of possibility and capability, whether material, physical, or casual.

3.2 Terminology

3.2.1 GPIB:

General Purpose Interface Bus. This may refer to IEEE 488.1 systems or IEEE 488.2 systems.

3.2.2 IEEE Std 488.1:

A standard published by the IEEE in 1987. This standard defines the electrical, mechanical, and signaling protocols for an interface bus. The 488.1 standard did not define contents of messages, syntax or data structures.

3.2.3 IEEE Std. 488.2:

A standard published by the IEEE in 1992 that is a superset of IEEE 488.1. It defines contents of messages and further defines the status-reporting model.

3.2.4 SCPI

Standard Commands for Programmable Instruments. The SCPI consortium, still active at this time, attempts to define common commands and methods for programming instruments.

3.2.5 SRQ

Service Request. In IEEE 488, a device uses the SRQ line to indicate the need for attention. Typically, SRQ indicates data is ready to be transmitted and/or an error condition exists.

3.3 Explanation of figures

3.3.1 1394 packets with data payload

There are figures in the document that show 1394 packet data payload contents. Most figures that contain a data payload are shown in short form, without the 1394 header and without the data CRC at the end of the data. To illustrate, the long form of showing a block write is shown below.

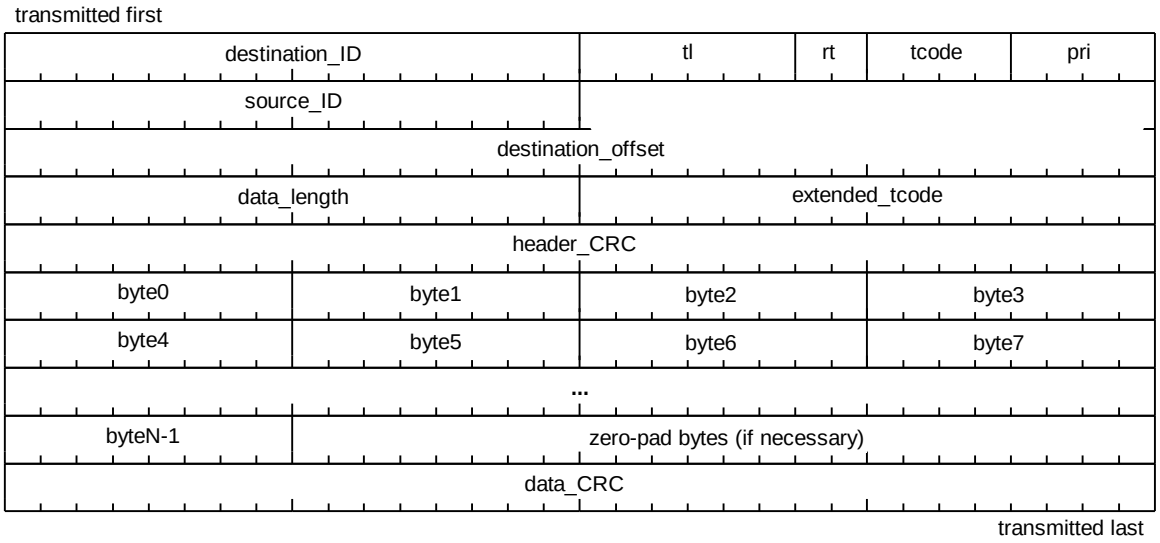

Figure 1 -- 1394 block write, including header and CRC's

The equivalent short form is shown below.

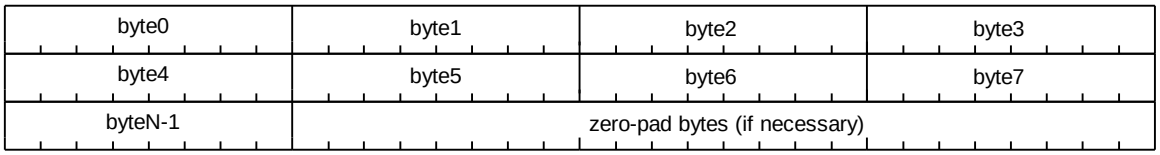

Figure 2 -- 1394 block write (short form)

3.3.2 Reserved fields

Some locations in a packet may be marked as ***Reserved*** or ***res***. These fields are reserved and shall be zero-valued. An implementation is not required to check for 0-valued reserved fields.

4. Configuration ROM for IICP488 devices

A device that uses IICP488 shall have a configuration ROM structure as specified in the IICP document.

4.1 IICP488 IICP_capabilities register

IICP488 makes use of the required IICP_capabilities register in the configuration ROM as shown below.

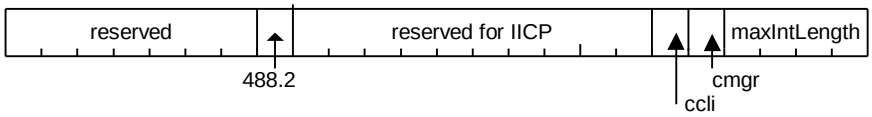

Figure 3 -- IICP488 IICP_capabilities

- The ***488.2***-bit shall be one-valued if the device implements the syntax, data structures, common commands and queries defined in IEEE 488.2
- ***ccli, cmgr, maxIntLength*** are as defined in IICP.

4.2 IICP488 configuration ROM values

Configuration ROM variable for a unit implementing IICP488	Configuration ROM MACRO	Value (Hex)
unit_spec_id	1394TA_SPEC_ID	$00A02D_{16}$
command_set_spec_id	1394TA_SPEC_ID	$00A02D_{16}$
unit_sw_version	IICP_UNIT_SW_VERSION	$4B661F_{16}$
command_set	IICP488_COMMAND_SET	$C27F10_{16}$
command_set_details	IIC488_COMMAND_SET_DETAILS	The value is the version of the IICP488 specification on the title page of this document.

Table 1 – IICP488 configuration ROM constants

Note: there is no configuration ROM entry to indicate if a device communicates using SCPI or some other language.

4.3 Interrupt_enable_reg_offset, Interrupt_handlr_reg_offset entries

IICP488 devices may optionally contain an Interrupt_enable_reg_offset entry and an Interrupt_handlr_reg_offset entry. The GPIB-like SRQ packets defined in IICP488 shall not use this mechanism. A device may have some device-dependent reason for using the alternative interrupt mechanism.

5. Connection model (informative)

The diagram below shows one possible communication model between a controller and an instrument using IICP488.

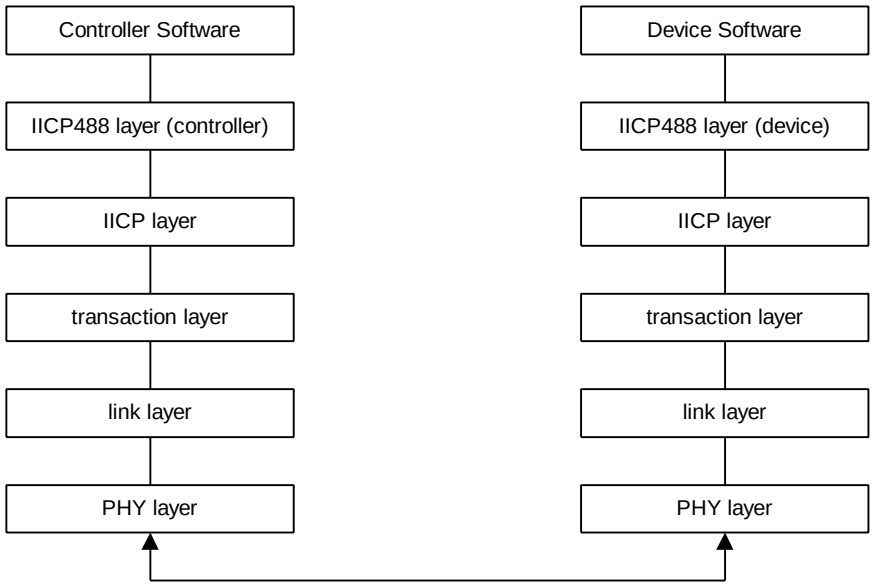

Figure 4 -- IICP488 communications model

5.1 IICP layer

The IICP layer implements the 1394TA Instrument and Industrial Control Protocol. It provides services used by the IICP488 layer to send frames using the data port and the control port of plugs.

5.2 IICP488 layer

The IICP488 layer implements the IICP488 services defined in this document. The IICP488 protocol defines the use of the data port and control port in an IICP plug.

An IICP plug schematic, as used in IICP488, is shown below. Refer to the IICP document for more details.

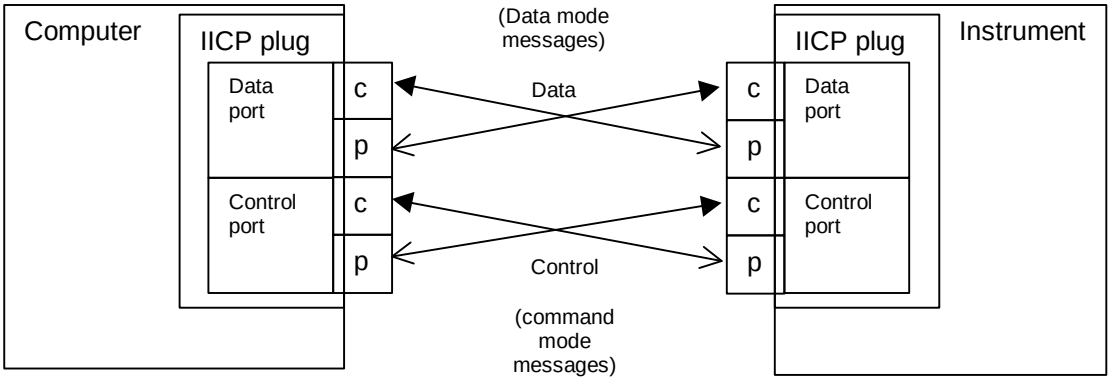

Figure 5 -- IICP488 plug schematic

- p = producer
- c = consumer

5.2.1 IICP plug data port use in IICP488

The data port is used to send IEEE 488.1 remote device dependent messages (IEEE 488.1 ATN false). These messages are sometimes called data mode messages, program messages, or in-band messages.

The instrument sends data mode response messages, when appropriate, also using the data port. The IICP service (IICP write data frame request) is used to send data mode messages and responses.

An example data mode message is the IEEE 488.2 defined identification query, ASCII-coded "*IDN?". An example data mode response would, in this case, be the IEEE 488.2 required ASCII-coded Manufacturer, Model, Serial number, and Firmware level or equivalent.

All SCPI messages and responses shall be sent using the data port.

5.2.1.1 Data mode messages and responses

Data mode messages and responses are sent using the data port. The contents of data mode messages are as specified in the relevant IEEE 488.1, IEEE 488.2, SCPI, and other applicable reference documents. In general, the format is a pure stream of data bytes, as shown below.

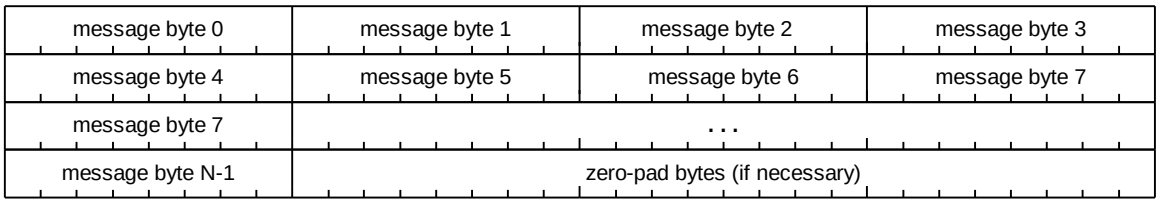

Figure 6 – IICP488 data port message packets

- ***message byte0 … message byte N-1*** is the data mode message.
- ***zero-pad bytes*** are required by 1394 since payloads must be a multiple of four in size. Some implementations may zero-pad automatically.

5.2.2 IICP plug control port use in IICP488

The computer sends IEEE 488.1 remote interface messages (IEEE 488.1 ATN true) using the control port. These messages are sometimes called command mode messages or out-of-band messages.

The instrument sends command mode response messages also using the control port. The IICP service (IICP write control frame) is used to send command mode messages and responses.

Command mode messages are used for reading the status byte, setting of remote/local, triggering, sending of SRQ, and selected device clear. Most command mode messages require a command mode response. For command mode message requests that require a command mode message response, the response shall be sent within 1 second of receiving the request.

There shall be at most one outstanding command mode message (no command mode response received yet) with the following exception - a selected device clear (SDC) message may be sent even if a command mode message response has not been received. A selected device clear shall not be sent if there is an outstanding SDC.

5.2.2.1 Command mode messages

Command mode messages are sent using the control port and shall have the format shown below.

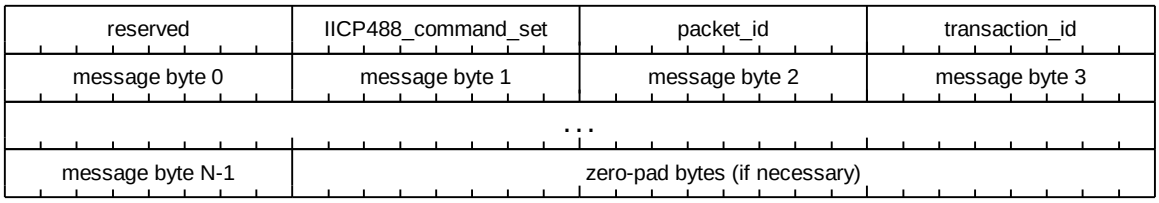

Figure 7 -- IICP488 command mode messages

- ***IICP488_command_set*** specifies the command set to be used to interpret the following packet_id. For the packet_id values shown in Table 2, the IICP488_command_set field shall be set to 1.
- ***packet_id*** identifies the packet. See Table 2.
- ***transaction_id*** is a label associated with this command. This label shall be part of the command mode response packet.
- ***message byte 0 … message byte N-1*** are whatever additional bytes that are defined for the particular command. The message bytes shall contain at most one command mode message. Multiple command mode messages shall not be embedded in one large message.
- ***zero-pad bytes*** are as described in Figure 6.

5.2.2.2 Command mode message responses

Command mode response messages are sent using the control port and shall have the format shown below.

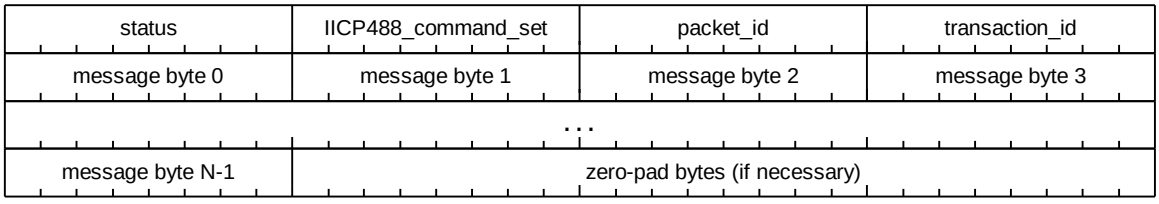

Figure 8 – IICP488 command mode responses

- ***status*** specifies the success or failure in processing the request. See Table 3.

- ***IICP488_command_set*** specifies the command set to be used to interpret the following packet_id. For the packet_id values shown in Table 2, the IICP488_command_set field shall be set to 1_{16}.
- ***packet_id*** identifies the packet as a particular command mode response packet. See Table 2.
- ***transaction_id*** is the transaction label of the request causing this response.
- ***message byte 0 … message byte N-1*** is whatever the response bytes are defined to be for the particular request.
- ***zero-pad bytes*** are as described in Figure 6.

5.3 Small frame messages

Data mode and control mode messages that fit the criteria for small frames may be sent as a small frame, if the consumer has enabled the producer to send small frames.

The message content shall include the complete logical message, from the first character through the last character. For example, "*IDN?", if sent as a small frame, shall consist of 5 bytes: '*', 'I', 'D', 'N', '?'. A new-line character may be added after the '?', but is not required. It is not permissible to send a subset of a message as a small frame and then to send another subset of the message as a subsequent small frame.

6. IICP488 required services

No particular IICP488 API is defined in this document. This specification only provides details on the actual 1394 packets sent when performing the required services listed below. All of these services make use of lower level IICP services, described in the IICP specification. See the IICP chapter on IICP services.

6.1 IICP488 open

The IICP488 open service makes use of lower level IICP layer service (IICP open) to create an IICP plug for IICP488 use. The IICP connection manager, as part of a connection sequence, sends a connection request packet (CREQ1) to each of the two devices involved in the connection. For complete details on the connection manager activities in creating a connection, see the IICP specification. A CREQ1 packet is shown below.

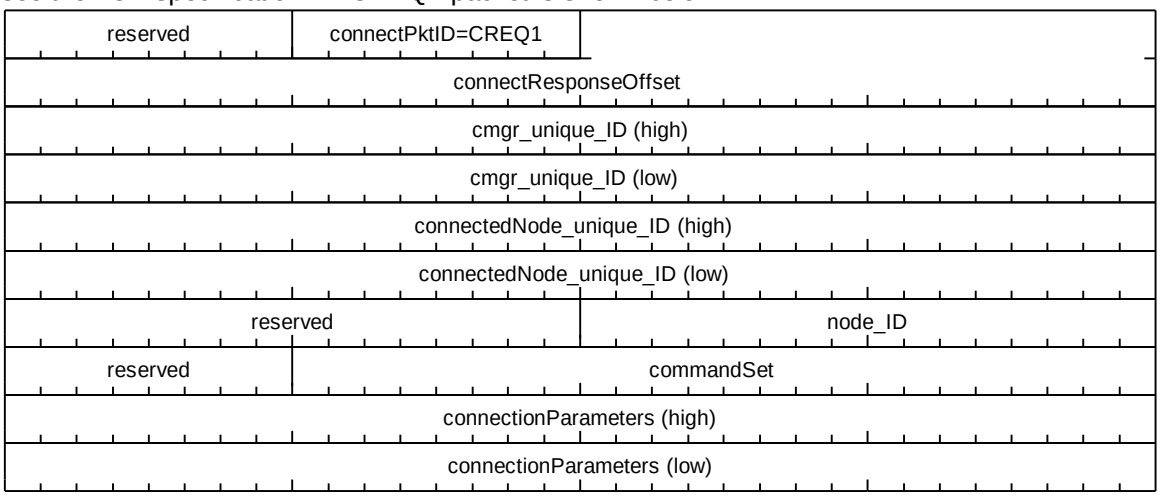

Figure 9 – IICP488 connection request (CREQ1) packet

The fields are defined in the IICP document. For IICP488, ***connectionParameters*** is defined as shown below.

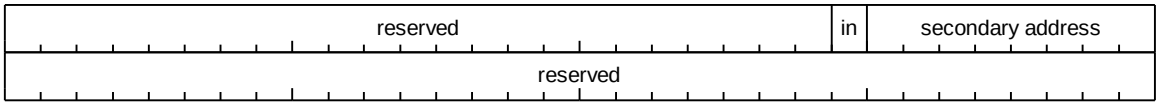

Figure 10 -- IICP488 connection parameters

- The ***in***-bit (initiate-bit) is one-valued if the receiver of the packet is to act as a controller as defined in IEEE 488.2, section 3.1. The ***in***-bit is zero-valued if the receiver of the packet is to act as a device (again, see IEEE 488.2 section 3.1) and passively listen for incoming messages and then send response messages. This enables an IICP488 device to know implicitly when frame contents are received whether it is a request or a response. The ***in***-bit shall be one-valued for only one of the devices involved in the connection. The CREQ1 packet sent to the other device shall have the ***in***-bit set to 0.
- ***secondary address*** identifies a possible sub-device. Secondary addresses are used as in IEEE 488.2, section 5.2. A secondary address value of FF_{16} shall be used for the device itself, not a sub-device.

IICP488 implementations shall create plugs when requested, so long as resources are available. A connection manager may create any number of connections between the same two nodes. Each connection is independent and autonomous. Only one connection shall be made to an IEEE 488.2 device when device is as defined in IEEE 488.2, section 3.1.

6.2 IICP488 close

This service explicitly closes a communication channel to an IICP488 instrument. The plug created during the IICP488 open service shall be stopped and the resources freed. This service makes use of a lower level IICP layer service (IICP close).

Note: a plug may be implicitly closed if a bus reset occurs and the IICP layer fails in an attempt to reactivate the connection.

6.3 IICP488 write

This service is used to write IEEE 488.1 device dependent messages through an IICP plug data port to an IICP488 device. The format of the packets was shown earlier, in Figure 6. This service makes use of a lower level IICP layer service (IICP write data frame).

6.4 IICP488 read

Reads response messages from an IICP488 device through an IICP plug data port. This service makes use of a lower level IICP layer service (IICP read data frame). The service will block until the connected node producer satisfies the read request.

6.5 IICP488 read status byte (IEEE 488.1 serial poll)

This service corresponds to the IEEE 488.1 serial poll mechanism and reads the status byte from an IICP488 device. This is accomplished using the control port. The format for the control READSTB command packet is as shown in Figure 7. In this case, ***packet_id*** is READSTB and there are no additional message bytes.

IICP488 devices shall return a command mode response packet after receiving a READSTB command. The command mode response packet is shown below.

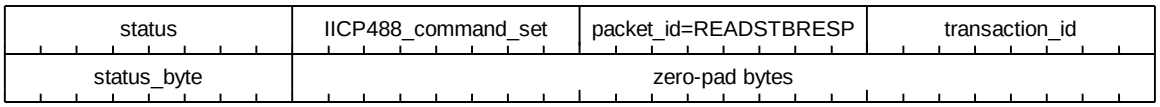

Figure 11 – IICP488 read status byte (READSTBRESP) response packet

- ***status, IICP488_command_set, packet_id, transaction_id*** were previously defined.
- ***status byte*** is the status byte. The status byte content shall be as specified in IEEE 488.2 if the 488.2-bit in the IICP_capabilities register in the configuration ROM is set to 1. The status byte content shall be as specified in IEEE 488.1 if the 488.2-bit in the IICP_capabilities register in the configuration ROM is set to 0.

6.6 IICP 488 trigger mechanisms

NOTE: A hardware method for triggering is preferred over the software triggering mechanisms defined below. The software triggering mechanisms are provided only if no acceptable hardware triggering mechanisms exist.

6.7 Triggering of a single device

This service corresponds to the IEEE 488.1 Group Execute Trigger (GET) sent to a single listener.

The format for a trigger command mode message is as shown in Figure 7, with ***packet_id*** = TRG.

The device being triggered shall return a TRGRESP response packet when triggered. The format for the TRGRESP response packet is as shown in Figure 8. A trigger response packet shall have ***packet_id*** = TRGRESP and no additional message bytes.

6.8 Triggering of a set of devices

An IICP implementation shall provide a service that corresponds to the IEEE 488.1 Group Execute Trigger of more than one listener.

6.8.1 Sending multiple TRG trigger packets

An IICP implementation may choose to send individual trigger packets as discussed in section 6.7. With this strategy, the synchronization of the trigger degrades with the number of devices. The figure below shows this mechanism.

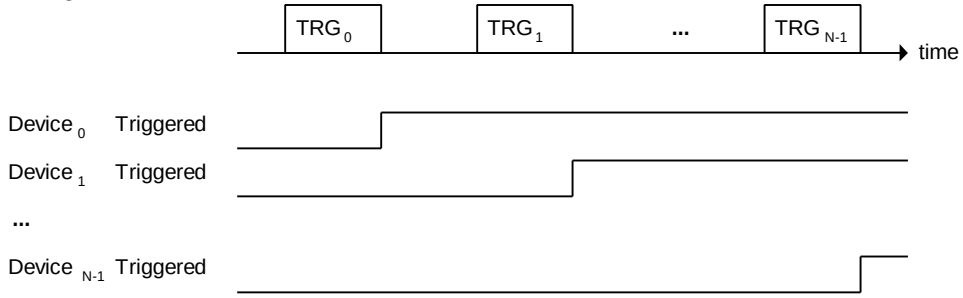

Figure 12 – Triggering multiple devices with multiple TRG packets

6.8.2 Triggering a set of devices using asynchronous stream packet

An IICP implementation may choose to trigger multiple devices by sending a single asynchronous stream trigger packet. Asynchronous stream triggers are accomplished with an asynchronous stream packet, defined in the P1394a document. This triggering mechanism allows a device to send one packet and trigger many devices. The device setting up an asynchronous stream trigger shall check that all nodes to be triggered have the Bus_Info_Block ***isc***-bit set, indicating the node is isochronous capable. If any node does not have the ***isc***-bit set, this mechanism shall not be used. The figure below illustrates this mechanism.

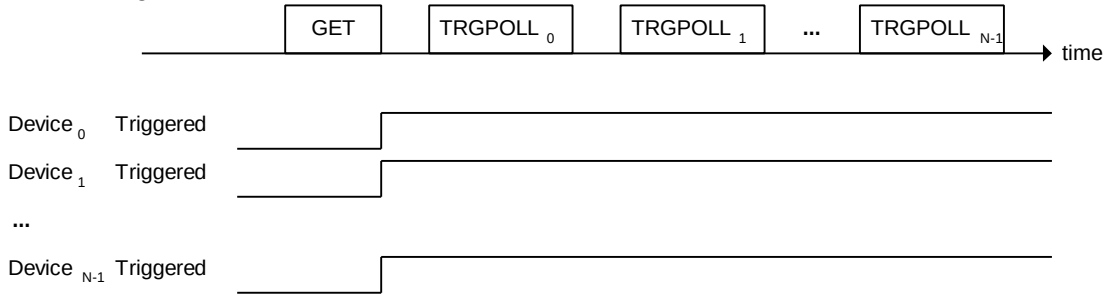

Figure 13 -- Triggering multiple devices with asynchronous stream trigger

6.8.2.1 Setting up an asynchronous stream trigger

To set up a device to accept an asynchronous stream trigger, a control request packet is sent to the device using the control port. The format of the packet is shown below.

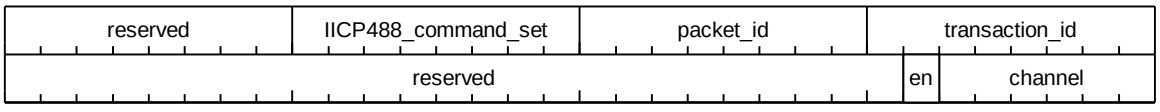

Figure 14 – Asynchronous stream trigger set up (GETCFG) request packet

- ***IICP488_command_set*** identifies this packet as an IICP488 packet.
- ***packet_id*** is set to GETCFG.

- ***transaction_id*** is as defined previously in association with Figure 7.
- The ***en***-bit is one-valued if the device is to enable receiving the asynchronous stream packet. The ***en***-bit is zero-valued if the device is to disable receiving the asynchronous stream packet.
- ***channel*** specifies the isochronous channel that will be used to communicate the asynchronous stream trigger event.

After receiving the GETCFG control packet, the device shall return a command mode response packet when ready to be triggered. The format for the response packet is as shown in Figure 8. A GETCFG response packet shall have ***packet_id*** = GETCFGRESP and no additional message bytes.

If any of the devices to be triggered returns a GETCFGRESP packet with status != SUCCESS, the device sending the trigger shall not use the asynchronous stream packet triggering mechanism. Any previous devices configured for an asynchronous stream trigger packet shall be sent a control packet to disable the anticipated trigger. The format for the packet is as shown in Figure 14, with the en-bit set to zero. After receiving the GETCFG packet to disable the trigger, the device sends a GETCFGRESP packet.

A selected device clear, SDC, shall disable any triggers set up via GETCFG packets.

6.8.2.2 Sending the asynchronous stream trigger packet

The format for an asynchronous stream trigger packet is shown below.

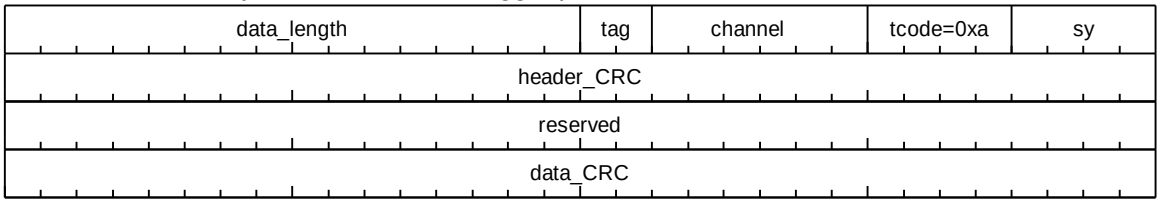

Figure 15 -- Asynchronous stream trigger (GET) packet

- The ***data_length, tag, channel, tcode,*** and ***sy*** fields are as specified in the IEEE 1394-1995 standard. The data_length shall be 4. The tag value shall be 0. The sy value shall be 0. The device sending the asynchronous stream packet shall allocate the ***channel*** from the isochronous resource manager as described in IEEE1394-1995, 8.4.3.2.

Once triggered, the device shall ignore asynchronous stream triggers with the same channel number until it is again set up to receive an asynchronous stream trigger.

6.8.2.3 Trigger polling

It is possible for a device to be busy and not receive the asynchronous stream trigger packet. Therefore, after sending the asynchronous stream trigger, the sender of the asynchronous stream GET packet shall poll each device that was to have triggered. The format of the poll request is as shown below.

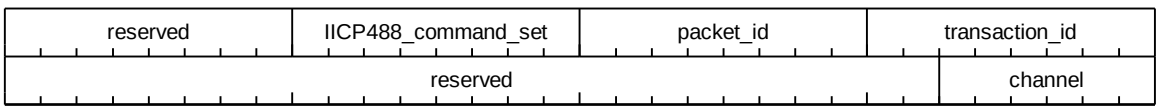

Figure 16 -- TRGPOLL packet

- ***IICP488_command_set*** identifies this packet as an IICP488 packet.
- ***packet_id*** is set to TRGPOLL.
- ***transaction_id*** is as defined previously in association with Figure 7.

- ***channel*** specifies the isochronous channel that was used to communicate the asynchronous stream trigger event.

After receiving the TRGPOLL control packet, the device shall trigger, if it for some reason did not trigger when the asynchronous stream packet was broadcast. A TRGPOLLRESP response packet shall be sent as shown in Figure 8. A TRGPOLLRESP response packet shall have ***packet_id*** = TRGPOLLRESP and no additional message bytes.

6.9 IICP488 selected device clear

Performs the equivalent of a GPIB selected device clear on an IICP488 instrument. This is sent using the control port. The format for the selected device clear command packet is shown in Figure 7. The ***packet_id*** = SDC, and there are no additional message bytes.

An instrument shall return a command mode response packet after receiving and processing to completion a selected device clear command. The response packet is as shown in Figure 8. The ***packet_id*** = SDCRESP, and there are no additional message bytes.

Guidelines for device behavior when an SDC packet is received is described in IEEE 488.2, section 5.8.

6.9.1 Selected device clear operation for controller sending SDC

6.9.1.1 Controller acting as producer and sending SDC

A controller sending an SDC packet shall coordinate the sending of SDC such that the SDC is sent when the controller has completed sending a data frame and has not yet started sending a new data frame. An SDC shall not be sent in the middle of a large data frame transfer. Sending an SDC shall block both data frame and control transfers until the SDCRESP packet has been received.

6.9.1.2 Controller acting as consumer and sending SDC

Once an SDC is sent, the controller shall discard all received data frames and control frames until the SDCRESP packet is received.

6.9.2 Selected device clear operation for device receiving SDC

6.9.2.1 Device acting as consumer and receiving SDC

A device receiving an SDC packet should discard all data frames and control frames received prior to the SDC packet. An SDCRESP packet shall be sent when all received data frames and control frames have been processed or discarded.

6.9.2.2 Device acting as producer and receiving SDC

A device receiving an SDC packet that has not yet started sending a data or control frame shall discard the frame. All frames to be sent, except the SDCRESP for this SDC, shall be discarded.

Some implementations may have frames queued by lower level 1394 drivers, and an implementation may not be able to prevent transmission of those frames. Once an SDC is received, an implementation shall not add to this queue.

If an SDC is received and only part of a large frame is transferred, the device acting as a producer shall write to the connected node LargeFrameConsumer register. The mode shall be set to TRUNC and the count shall be set to the number of bytes sent.

If an SDC is received and a complete large frame has been transferred and the LargeFrameConsumer register has not yet been updated, the device acting as a producer shall write to the connected node LargeFrameConsumer register. The mode shall be set to LAST and the count shall be set to the number of bytes sent.

An implementation shall send an SDCRESP packet after all frame content has either been discarded or sent and if a large frame was in process, the LargeFrameConsumer register updated.

6.9.3 SDC and TRGPOLL packets

If a device receives an SDC and the device has been previously set up to trigger with an asynchronous stream trigger, the device shall disable the trigger. The controller will again need to set up the asynchronous stream trigger.

6.10 IICP488 remote

This service places an IICP488 device into remote-only mode. Front panel (local) operations are disabled. This is sent over the control port. The format for the REMOTE command packet is shown below.

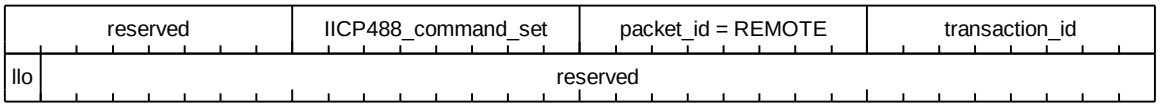

Figure 17 – REMOTE message

- ***IICP488_command_set*** identifies this packet as an IICP488 packet.
- ***packet_id*** is set to REMOTE.
- ***transaction_id*** is as defined previously in association with Figure 7.
- ***llo*** is one-valued if the “return to local” key, if it exists on the implementation, is to be disabled. If zero, the “return to local” key is enabled. The “return to local” key is a front panel key on some implementations that allow a user to regain local control.

An instrument shall return a command mode response packet after receiving and processing to completion a remote command. The response packet is as shown in Figure 8. The ***packet_id*** = REMOTERESP, and there are no additional message bytes.

6.11 IICP488 local

This service enables front panel (local) operations on an IICP488 instrument. This is sent over the control port. The format for the LOCAL command packet is as shown in Figure 7. The ***packet_id*** = LOCAL, and there are no additional message bytes.

An instrument shall return a command mode response packet after receiving and processing to completion a LOCAL command. The response packet is as shown in Figure 8. The ***packet_id*** = LOCALRESP, and there are no additional message bytes.

6.12 IICP488 ioctl

This service sends an IEEE 488.1 interface message through the control port to an IICP488 instrument. This command is used to cause the connected device to perform device-dependent functions. The packet is sent with the format shown below.

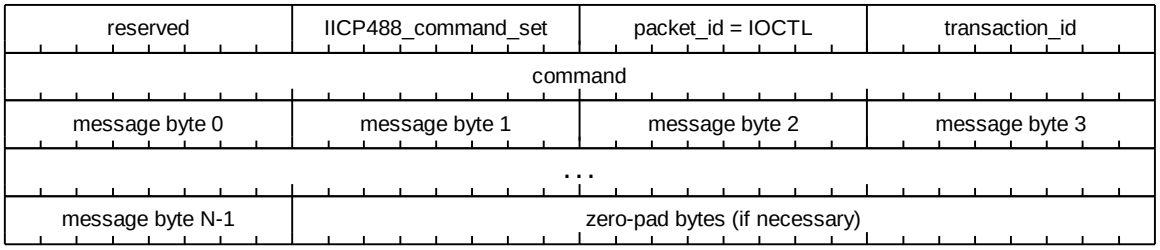

Figure 18 – IICP488 IOCTL request packet

- ***IICP488_command_set, packet_id, and transaction_id*** are as previously defined.
- ***command*** is the device-dependent command to be performed.
- ***message byte 0 … message byte N*** is any additional message bytes associated with ***command***.

A response shall be sent with the packet format shown in Figure 8, with packet_id = IOCTL_RESP.

7. GPIB interrupts (SRQ) and status reporting model

In IEEE 488.1, an interrupting device uses the SRQ line to communicate that an interrupt has occurred. The controller then enters into a serial poll algorithm to find out which GPIB device is causing the interrupt and the reason for the interrupt.

In IICP488, there is no need for such an algorithm. An interrupting device communicates the interrupt by sending a packet. Since a packet must be sent, the causes for the interrupt may as well be sent also, in the same packet, at the same time. There is no need for a serial polling algorithm.

Interrupts are enabled for IEEE 488.2 devices via programming of the Service Request Enable Register. The Service Request Enable Register is programmed to a value that allows an interrupt condition to be enabled.

Interrupts will be sent using the control port. The status_byte associated with the interrupt shall be included in the packet. The figure below shows an IICP488 SRQ packet.

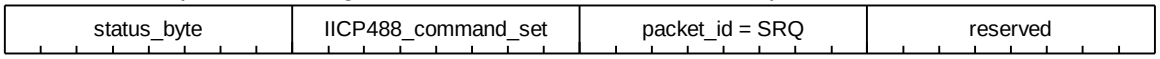

Figure 19 – SRQ packet

- ***status_byte*** shall be the status byte as defined in IEEE 488.2 if the 488.2-bit in the IICP_capabilities register of the device sending this packet is set to 1. The status_byte shall be as defined in IEEE 488.1 if the 488.2-bit in the IICP_capabilities register of the device sending this packet is set to 0.
- ***IICP488_ommand_set, packet_id*** are as defined previously.

After sending an SRQ control mode message the device shall behave as if the controller had performed a serial poll. The intent is to circumvent any requirement of controllers to do the equivalent of a serial poll. The act of sending an SRQ shall be equivalent to a GPIB SRQ event and a GPIB serial poll event.

For efficiency, the controller sends no control mode response packet to an SRQ packet.

7.1 MAV behavior

This clause is for devices that set the 488.2-bit in the IICP_capabilities register in the configuration ROM to one, and describes behavior associated with the IEEE 488.2 defined MAV bit (bit 4) in the status byte. See IEEE 488.2, section 11.2.1.2 for reference.

MAV shall be set TRUE when the first byte of message data becomes available to send. After being set TRUE, MAV shall remain TRUE until the frame is completely sent. If the frame is sent in large frame mode, MAV shall transition to FALSE when the last frame content is sent and before the LargeFrameConsumer register is updated. In IICP488, MAV is not allowed to toggle TRUE to FALSE to TRUE. In IICP488 systems, there is no need to toggle the MAV bit for polling or interrupt purposes.

8. IICP488 control mode message packet_id command values

packet_id	Decimal Value	Comment
GETCFG	1	Group execute trigger configuration packet.
IOCTL	2	Device-dependent control request packet.
LOCAL	3	Local packet.
READSTB	4	Read status byte
REMOTE	5	Remote packet.
SDC	6	Selected device clear
SRQ	7	Service request packet. No command mode message response packet is sent.
TRG	8	Immediate trigger packet.
TRGPOLL	9	Trigger poll packet.
GETCFGRESP	129	Group execute trigger configuration response packet.
IOCTLRESP	130	Device-dependent IOCTL control response packet.
LOCALRESP	131	Local response packet.
READSTBRESP	132	Read status byte response packet
REMOTERESP	133	Remote response packet.
SDCRESP	134	Selected device clear response packet.
TRGRESP	136	Immediate trigger response packet.
TRGPOLLRESP	137	Trigger poll response packet.
-	0, 10-128, 135, 138-255	Reserved

Table 2 -- IICP488 packet_id values

9. IICP488 control mode message response status values

status macro	Decimal Value	Comment
SUCCESS	0	The control request was processed successfully.
PARM	1	The control request failed due to a bad parameter in the request.
VENDOR_UNIQUE_0	128	Vendor unique status value.
VENDOR_UNIQUE_1	129	Vendor unique status value.
...	...	...
VENDOR_UNIQUE_15	143	Vendor unique status value.
-	2-127, 144-254	Reserved
FAIL	255	The control request failed for unknown reason.

Table 3 – IICP488 status values