

An Introduction to the Instrument and Industrial Control Protocol

Introduction

Manufacturers of electronic instrumentation and manufacturers of industrial control devices know that such devices are frequently connected to computers for automated test and control. There is an obvious need to connect these devices using industry standard I/O. Using industry standard I/O has several advantages:

1. Cost. Standard I/O connectors and cables are cheaper than proprietary or low-volume connectors and cables.
2. Ease of use. Users simply plug standard, familiar cables into connectors that already exist on computers. Users do not have to open up a computer to install a host bus adapter. More importantly, users do not have to deal with hardware or software configuration for the host bus adapter.

It is anticipated that the IEEE 1394 high-speed serial bus will become an industry standard I/O on many computers. The 1394 interface is already on some PC's and workstations, and more manufacturers plan on providing 1394 to meet I/O needs for consumer electronics. In addition to the advantages listed above, 1394 equipped computers will offer:

1. High speed communications. Speeds up to 400 Mbps are allowed with IEEE 1394.
2. Self-configured addressing. Users do not have to set address switches. There is no potential for address conflicts.
3. A tiered-star topology, allowing up to 63 devices to be connected.

The attractiveness of 1394 for electronic instruments and industrial control devices led to the formation of the Instrumentation and Industrial Control Working Group (II-WG) within the 1394 Trade Association.

The II-WG examined existing and work-in-progress 1394 protocols and concluded that a new, lightweight protocol was needed to most efficiently meet communications requirements for instruments.

The II-WG currently has a draft specification for this new protocol called the Instrument and Industrial Control Protocol, or IICP. IICP is not yet an approved standard. However, much of the document is now stable. This paper serves as an introduction to IICP.

To view the IICP draft specification or to get involved in the II-WG requires membership in the 1394 Trade Association. The 1394 Trade Association web site, <http://www.1394ta.org> explains how to become a member. Once the IICP draft specifications are approved within the II-WG, they will be circulated to wider audiences.

IICP communications model

IICP is a baseline, mid-level protocol. Figure 1 shows a layered communications model with IICP and a higher level protocol, IICP488, layered above IICP. IICP488 defines how to send GPIB (IEEE488.1 and IEEE488.2) messages using IICP, and is also being defined by the II-WG.

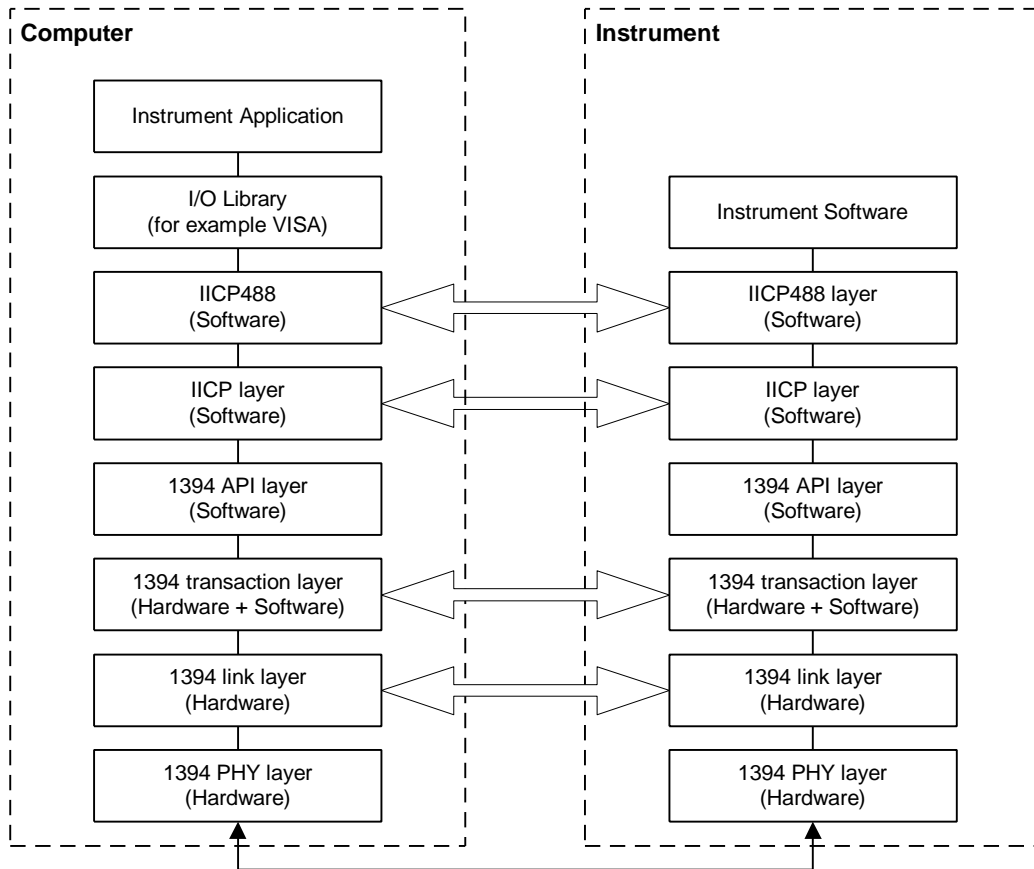


Figure 1 -- IICP488 communication model

Starting from the bottom, the IEEE 1394-1995 specification defines the 1394 PHY, link, and transaction layers.

A 1394 API layer may be used in some implementations to provide basic services needed by IICP and other possible protocols.

The IICP layer implements the services needed to create a connection between two IICP devices, and then to transfer data through the connection.

The IICP488 layer implements the services for GPIB-like communications, such as sending a message, sending a device clear, or setting local operation.

Instrument vendors typically provide an I/O library so an application can be written to communicate to instruments. VISA (Virtual Instrument Software Architecture) is an instrument I/O library supplied by several vendors.

IICP connections

A computer with an IICP connection to an instrument is shown in Figure 2.

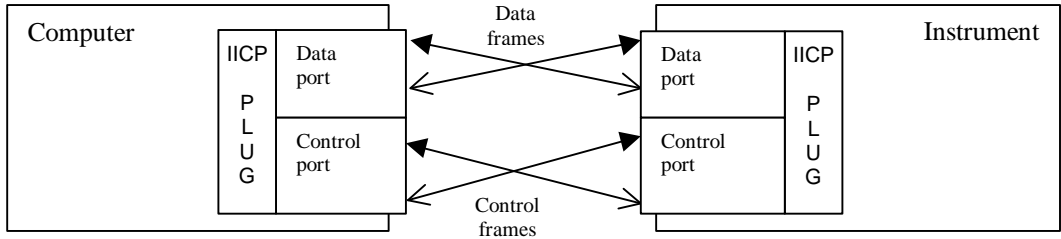


Figure 2 -- Typical IICP connection

Figure 2 shows an IICP connection. An IICP connection consists of an IICP plug on the computer and an IICP plug on the instrument. A plug contains two ports. A higher level protocol layered above the IICP layer determines the actual use of the ports. One port would typically be used to send and receive data frames, and one port would be used to send and receive control frames. A frame is logical, contiguous data. An example of a data frame if the higher level protocol is IICP488 is a SCPI message or response. An example of a control frame in IICP488 would be an SRQ packet.

The duplex send and receive functions (producer and consumer) in each port provide a mechanism for a producer to send a message and a consumer to receive the message and then send a response message.

Each plug port consists of device memory mapped to 1394 space. The exact plug architecture is still subject to change, but will likely be as shown in Figure 3. The plug registers provide a flow control mechanism. A consumer uses flow control to pace the frames from the producer.

ConsumerMode
ProducerLimits
ProducerMode
PageTableElement 0
PageTableElement 1
...
PageTableElement N-1

Figure 3 -- IICP plug port

A frame transfer from a producer to consumer is enabled when a consumer writes to the ProducerLimits, ProducerMode, and PageTableElement registers of the plug on the connected node.

The ProducerLimits register lets the consumer specify the maximum sized payload in a single 1394-write block transaction.

The ProducerMode register has 3 mode bits. The producer mode is usually set to "SEND", which means the consumer is ready to receive frames from the producer. The ProducerMode register also contains the number of PageTableElement registers the producer may use when transferring a frame.

The PageTableElement registers consist of a 16-bit length and 48-bit address of consumer buffers that the consumer has mapped to 1394 space. The PageTableElement registers provide a scatter/gather mechanism useful with some 1394 link implementations. Some link implementations conform to the Open Host Controller Interface specification, or OHCI. OHCI defines a "physical DMA" mechanism in which the OHCI link performs read and write operations directly to system memory, without processor intervention.

Consumer buffers in virtual memory are typically described as a scatter/gather list in physical memory. For optimum transfer speeds, the PageTableElement registers would be programmed with a scatter/gather list that corresponds to a consumer buffer in physical space.

The producer, after seeing the ProducerLimits, ProducerMode, and PageTableElement registers programmed, now sends a frame to the consumer. The frame may be sent using multiple 1394 write block transactions. When the frame is completely sent, or when the consumer buffers are completely filled, the producer writes to the ConsumerMode register on the connected node. The producer writes a field in the ConsumerMode register to indicate how many bytes were transferred. A mode field in the ConsumerMode register lets the producer indicate “LAST”, if all of a frame has been sent, or “MORE” if only a partial frame has been sent.

The producer then waits until the consumer is again ready for more data. When again ready, the consumer again writes to the ProducerLimits, ProducerMode, and PageTableElement registers as necessary.

Small-frame transfers

There are many situations where a computer may repeatedly send small frames to a device. An example is when a computer is sending some kind of query to an instrument to get measurement results.

For small frames, there is no need for the ProducerMode and PageTableElement updates, followed by a ConsumerMode update for every frame. Instead, the plug register updates only need occur when the next small frame does not fit completely into the consumer buffer. This allows frame transfers using just one 1394 transaction per frame. This is more efficient than the three 1394 transactions described earlier, in which ProducerMode, PageTableElement, and ConsumerMode registers are updated for each frame.

Connection establishment

An IICP connection manager establishes IICP plug connections. Any node may act as an IICP connection manager. A connection is established by first locking connection registers and then sending a sequence of connection request packets. The connection request packets contain the information needed for one device to learn the address of the plug created on the connected node. By first locking a connection register, an IICP connection client is never burdened with processing simultaneous connection requests from multiple threads or multiple controllers.

The connection manager is responsible for the maintenance of the connection. If a 1394 bus reset occurs, all plug activity is immediately stopped. This is because node addresses may have changed due to a new device having been added to the bus. The connection manager must enumerate the bus, find new node addresses, and then update plug information before plug activity can restart. Once reactivated, plug activity resumes automatically. 1394 bus resets are completely handled by the IICP layer. Higher level protocols and applications are oblivious to 1394 bus resets and continue on with whatever prior activities were being performed.

A typical system would have a computer running the test system application acting as a connection manager. Only the computer would typically enumerate the bus to discover 1394 IICP devices.

Discovery of IICP devices

Discovery refers to the process that software goes through to understand the protocol that a particular device understands. The IEEE 1394-1995 and IEEE 1212 specifications define the use of a configuration ROM for this purpose. Every device configuration ROM is mapped to a specific 48-bit address, FFFF F000 0000₁₆.

IICP defines the requirements of the unit directory within the configuration ROM. Note that a device can have multiple unit directories to support multiple 1394 protocols. An example would be a device that communicates to a computer using IICP and to an attached printer using the SBP-2 protocol.

Software implementation of IICP plug connections

The first step in the software implementation of IICP is to decide on a source for a 1394 API. A 1394 API should provide mechanisms to send compare and swap lock transactions and responses, write quadlet transactions, and write block transactions. The API should also provide mechanisms to map buffers and plug registers to 1394 space and receive notification when these are accessed by the connected node. Note that since IICP does not currently use any isochronous transactions, the API requirements are fewer.

For a PC, there are at least two choices:

1. Use a 1394 API from a third party.
2. Develop a 1394 API that is layered above the Microsoft™ 1394 bus class driver.

For non-PC platforms, there are also two choices:

1. Use a 1394 API from a third party. There are third party 1394 API's available for many popular RTOS's.
2. Develop a 1394 API.

Once the source for a 1394 API has been decided on, the IICP and higher layers must be implemented. The IICP document leaves many implementation details unspecified, but the following is offered as a possible set of services provided by an IICP layer.

IICP Service	Comment
IICP_control()	Initializes the IICP layer and allows higher layer to register callback functions for when the following occur: <ul style="list-style-type: none">- 1394 bus reset- a new connection has been established by another node- a control frame has been received- a connection has been closed by another node
IICP_open()	Creates a connection between two devices.
IICP_write()	Writes data to a consumer.
IICP_read()	Consumes data from a producer.
IICP_close()	Tears down a connection.
IICP_controlPortWrite()	Write a control frame to a consumer

Figure 4 -- IICP layer services

Implementing IICP488 should be straightforward, with the IICP services shown in Figure 4.

GPIB using IICP

The details of GPIB communication over 1394 are contained in a separate II-WG document, IICP488. The IICP488 document builds on the baseline IICP document and specifies use of the plug ports.

The data port is used for sending and receiving GPIB messages. An example data port message would be the IEEE 488.2 identification query “*IDN?”. A device receiving this message would send a response message, consisting of the IEEE 488.2 formatted identification string. Data port messages are pure data – there is nothing extra for software to parse or strip off. For instruments currently equipped with GPIB, this allows the adoption of the 1394 interface with less effort. Once an IICP488 data message is received, it is acted upon the same as if it came over GPIB. The messages remain the same – they are just sent over using different transaction and physical layers.

IICP488 specifies use of the plug control port for sending and receiving control messages. These messages are equivalent to the out-of-band GPIB signals. Control messages have a field in the 1394 payload that indicates the particular operation to perform. Example control messages the GPIB-like SRQ or the GPIB selected device clear.

IICP488.2 defines an SRQ packet that communicates the occurrence of an interrupt, along with the status byte. A controller no longer has to do a serial poll to find the interrupting device or the cause of the interrupt.

IICP memory mapped connections

The II-WG recognized that many low cost, simple devices may use memory-mapped I/O. 1394 defines a 48-bit address space, and a memory-mapped I/O device dedicates some of this space for device-dependent functions. Memory-mapped I/O may be suitable for simple digital to analog converters or for various kinds of sensors.

IICP does define a simple interrupt mechanism that may be used for these simple devices. This may, in some cases, alleviate the need for polling by a controller.

Future II-WG activities

In the short term, the II-WG plans to finish work on the IICP and IICP488 specifications.

In the long term, there is a need to define the operation of 1394 to GPIB bridge devices. Such devices will allow the use of legacy GPIB instruments with computers equipped with the 1394 interface.