III MODELO OPERATIVO DE LOS EQUIPOS (IEEE-448.2)

III.1 PROTOCOLOS.

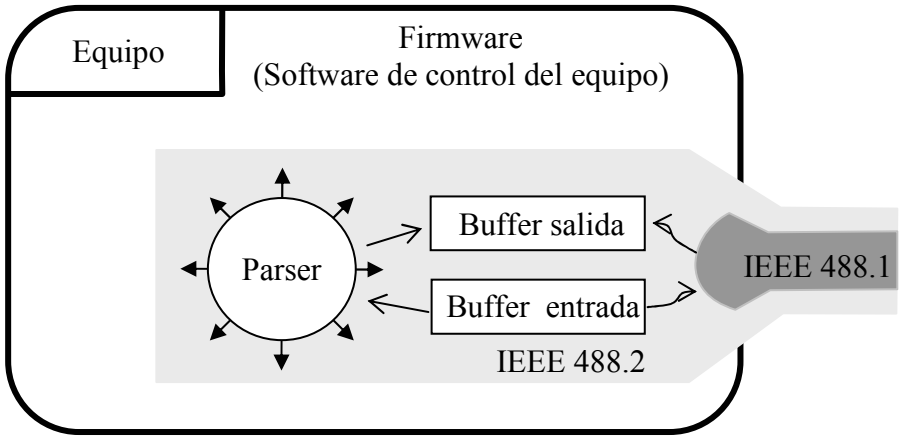
El estándar IEEE-488.2 define los modos de operación básicos de los equipos que lo satisfacen. Define los protocolos de intercambio de mensajes con el que el equipo y el controlador se comunican, así como, facilidades básicas de control del instrumento. Define:

a) Cuando los equipos están dispuestos a escuchar o a hablar a lo otros equipos.

b) Que ocurre cuando no se cumplen las normas establecidas en el protocolo.

Los equipos que satisfacen los modos operativos definidos en el estándar IEEE-488.2., necesariamente satisfacen los niveles de comunicación físicos definidos en el estándar IEEE-488.1. La implicación inversa no es requerida.

El modelo de intercambio de un equipo que satisface el estándar IEEE-488.2, incluye a los siguientes componentes:



Buffer de entrada: Es el área de memoria en la que las ordenes y requerimientos de entrada son almacenadas, antes de que sean interpretadas y ejecutadas por el Parser. El buffer de entrada, permite que el equipo controlador del instrumento, almacene en el buffer un string conteniendo una o varias ordenes que llevan un cierto tiempo ser ejecutada, y mientras esto ocurre pueda realizar otras operaciones con otros equipo.

Buffer de salida: Es el área de memoria en el que los datos de salida (mensajes de salida) son almacenadas en espera de que el controlador decida leerlas.

Parser: Es el controlador interno del equipo que interpreta los mensajes completos almacenados en el buffer de entrada, los ejecuta, y en caso de que sean de requerimiento, deposita en la cola de salida los datos que resulten.

Protocolo básico

- El equipo y el controlador se comunican intercambiando mensajes de ordenes y mensajes de respuesta.

- Los mensajes de órdenes son enviados por el controlador, y pueden ser de dos tipos:

  Ordenes de control: Que requieren un cambio de estado del equipo, pero que no requieren ninguna respuesta.

  Ordenes de requerimiento: Que solicitan información sobre el estado del equipo o sobre información que posee.

- El equipo solo habla (envía un mensaje de salida) como respuesta de una orden de requerimiento.

- El controlador solo admite un mensaje de salida (respuesta de una orden de requerimiento), y lo requiere antes de enviar un nuevo mensaje de ordenes. En caso contrario se genera una situación de bloqueo.

- La regla básica del protocolo es:

  "El equipo solo habla cuando está dispuesto a ello, y en ese caso, tiene que hablar antes de que se le ordene hacer una cosa nueva"

- Cuando el equipo es encendido, o cuando recibe una orden de inicialización "*CLS" el buffer de entrada y de salida son inicializada, y el parser es inicializado a la raíz de su árbol de ordenes.

- El equipo y el controlador se comunican intercambiando mensajes de ordenes y de respuestas completos. Esto significa que el controlador siempre debe terminar un mensaje de órdenes, antes de intentar leer una respuesta.

- Si el equipo envía un mensaje de respuesta, el controlador debe siempre leer de forma completa el mensaje, antes de que envíe un nuevo mensaje de ordenes al equipo.

- El controlador puede enviar un mensaje conteniendo múltiples ordenes de requerimientos. A esto se le denomina un "requerimiento compuesto". Los diferentes requerimientos dentro del mensaje deben estar separados por el delimitador ";". Los mensajes de respuesta son encolados en la cola de salida, separados entre sí también, por el delimitador ";".

- Los comandos son ejecutados en el orden en que han sido recibidos.

Protocolos de excepción:

Cuando un error ocurre en el intercambio de la información, este no termina en la forma normal, sino que sigue un protocolo de excepción:

a) Equipo direccionado para hablar sin nada en la cola:

- Si es consecuencia de que el equipo no haya recibido una orden de requerimiento, el equipo indicará un error de encolamiento, y no enviará ningún byte por el bus.
- Si es como consecuencia de que la orden de requerimiento previa no se ejecutó como consecuencia de un error, el equipo no indica ningún error de encolamiento, sino que el equipo espera a recibir el siguiente mensaje del controlador.

b) Equipo direccionado para hablar sin que ningún equipo escuche: En este caso, el equipo esperará o bien a que algún equipo escuche, o a que el controlador tome el control.

c) Error de orden: Se genera cuando se detecta un fallo de sintaxis o una orden no reconocible.

d) Error de ejecución: Se genera si un parámetro está fuera de rango, o si el equipo se encuentra en un estado que no permita la ejecución del comando requerido.

e) Error específico del equipo: Se produce cuando el equipo es incapaz de ejecutar una orden, como consecuencia de una razón estrictamente dependiente de él, y no del protocolo seguido por el bus.

f) Error de encolamiento: Se genera si no se sigue el protocolo de lectura de los datos de la cola de salida.

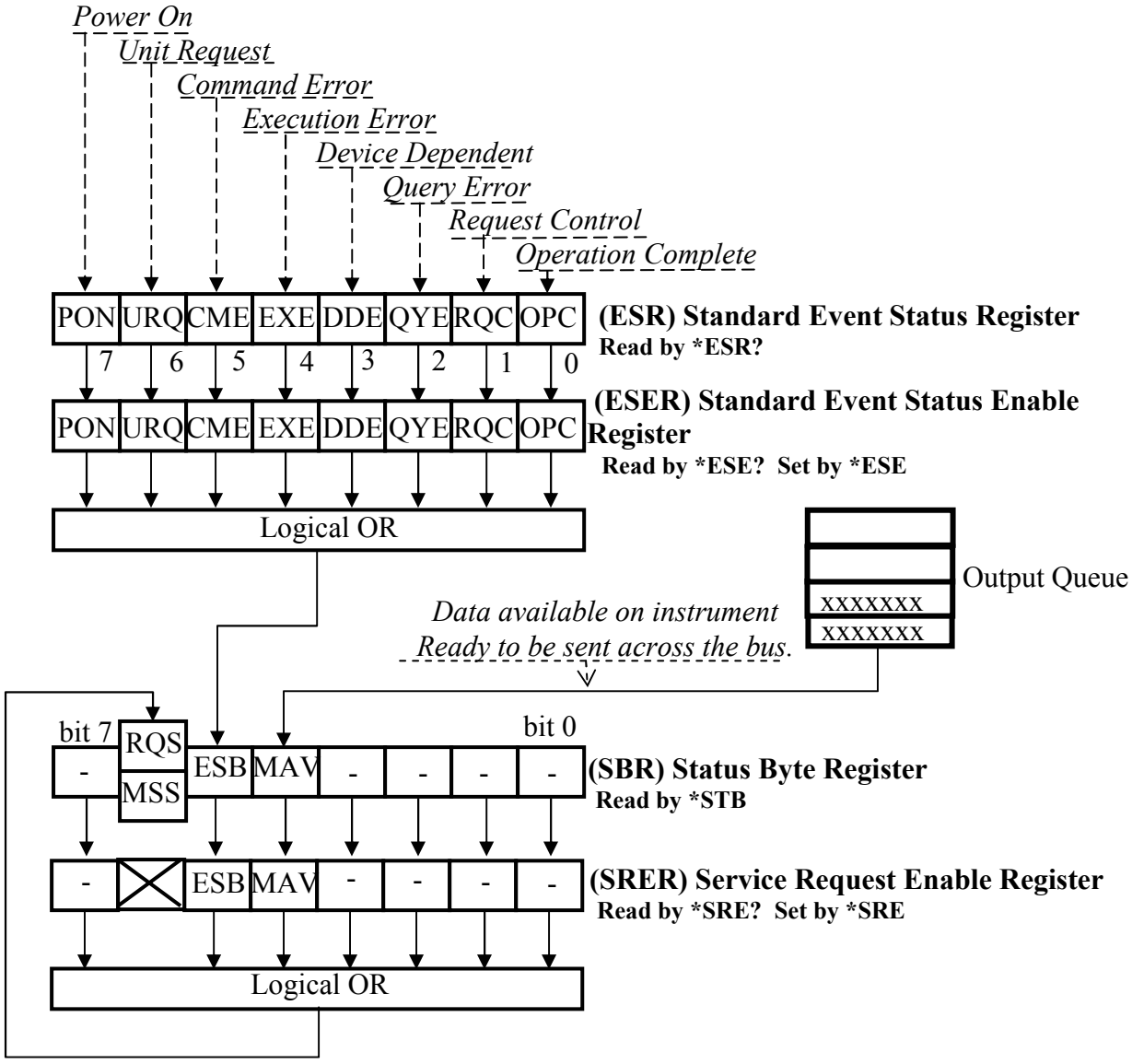
g) Condición inconclusa: Si el controlador intenta leer un mensaje de respuesta antes de que el programa haya concluido de ejecutar la orden que los genera. En este caso el Parser, se inicializa a si mismo, la respuesta ya elaborada es limpiada de la cola de salida, y ningún dato es transferido por el bus.

h) Condición interrumpida: Si el controlador no lee completamente el mensaje generado por un mensaje de requerimiento, y envía otro mensaje de orden, el equipo genera un error de encolamiento, y el segmento de mensaje de salida no leído es eliminado. La orden que interrumpe es inafectada.

i) Bloqueo de buffer: El equipo alcanza un estado de bloqueo si el buffer de entrada, está lleno y también resulta llena la cola e salida. Esta situación ocurre si un mensaje de orden muy largo que contiene órdenes de requerimiento ha sido enviado, y genera un mensaje de salida superior al que puede contener la cola. El controlador no puede terminar de enviar el mensaje de entrada porque no cabe, y el buffer de entrada no se vacía porque espera que su cola de salida sea vaciada para concluir la orden en ejecución. En este caso el equipo rompe el bloqueo, limpiando la cola de salida y generando un error de encolamiento.

III.2 Estatus de un equipo por el protocolo IEEE 488.2.

El estándar IEEE 488.2 ofrece un mecanismo estandarizado de presentar y mostrar el estado interno del equipo. A través de este mecanismo, se puede tener información de si el equipo tiene un dato dispuesto para transferir, así como si algún tipo de error ha ocurrido.



Status registers

El mecanismo se basa en cuatro registros:

Status Byte Register (SBR): Cada bit está asociado con un tipo de estado específico del instrumento. Cuando el estado cambia, el instrumento establece el correspondiente bit a 1. Se puede habilitar e inhibir el efecto de cada bit del SBR a efectos de requerir atención (bit RQS), con el correspondiente bit del registro SRER. Se puede determinar que eventos han ocurrido leyendo los establecidos en el registro SBR.

Bit	Label	Description
0-3	-	Instrument-specific summary messages.
4	MAV	The Message Available bit indicates if data is available in the Output Queue. MAV is 1 if the Output Queue contains data. MAV is 0 if the OutputQueue is empty.
5	ESB	The Event Status Bit indicates if one or more enabled events Have occurred. ESB is 1 if an enabled event occurs. ESB is 0 if no enabled events occur. You enable events with the Standar Event Status Enabled Register.
6	MSS	The Master Sumary Status sunarizes the ESB and MAV bits. MSS is 1 if either MAV or ESB is 1. MSS is 0 if both MAV and ESB are 0. This bits are obtained from *STB? Command.
	RQS	The ReQuest Service bit indicates that the instruments requests service from the GPIB controller. This bits is obtained by from a serial poll.
7	-	Instrument specific summary message.

Service Request Enable Register (SRER): Máscara de los bits correspondientes del registro SBR que va a determinar si se establece el Request Service (bit RQS).

Event Status Register (ESR): Cada bit está asociado con un tipo específico de evento. Cuando un evento ocurre, el instrumento establece el correspondiente bit a 1. Se puede habilitar o inhibir los eventos que van a ser proyectados sobre el bit ESB, a través de los correspondientes bits de máscara de registro ESER. Se puede leer el evento que ha ocurrido leyendo el contenido del registro ESR.

bit	Label	Description
0	OPC	The Operation Complete bit indicates that all commands have completed.
1	RQC	The Request Control bit is not used by most instruments.
2	QTE	The Query Error bit indicates that the instrument attempted to read an Empty output buffer, or that data in the output buffer was lost.
3	DDE	The Device Dependent Error bit indicates that a device error occurred (such as a self –test error).
4	EXE	The execution error bit indicates that an error occurred when the devices Was executing a command or query.
5	CME	The command Error bit indicates that a command syntax error occurred.
6	URQ	The User Request bit is not used by most instruments.
7	PON	The Power on bit indicates that the device is powered on.

Event Status Enable Register (ESER): Mascara de los bits del registro ESR que van a requerir el servicio a través del registro SBR (asertando el bit ESB).

Por ejemplo, si deseamos conocer cuando ocurre un error de ejecución, se debería establecer el bit 5 del registro ESER a fin de que el evento de error de interés (que establecerá el bit 5 del ESR a 1) sea reportado por el bit ESB del registro SBR.

III.3 ORDENES BASICAS

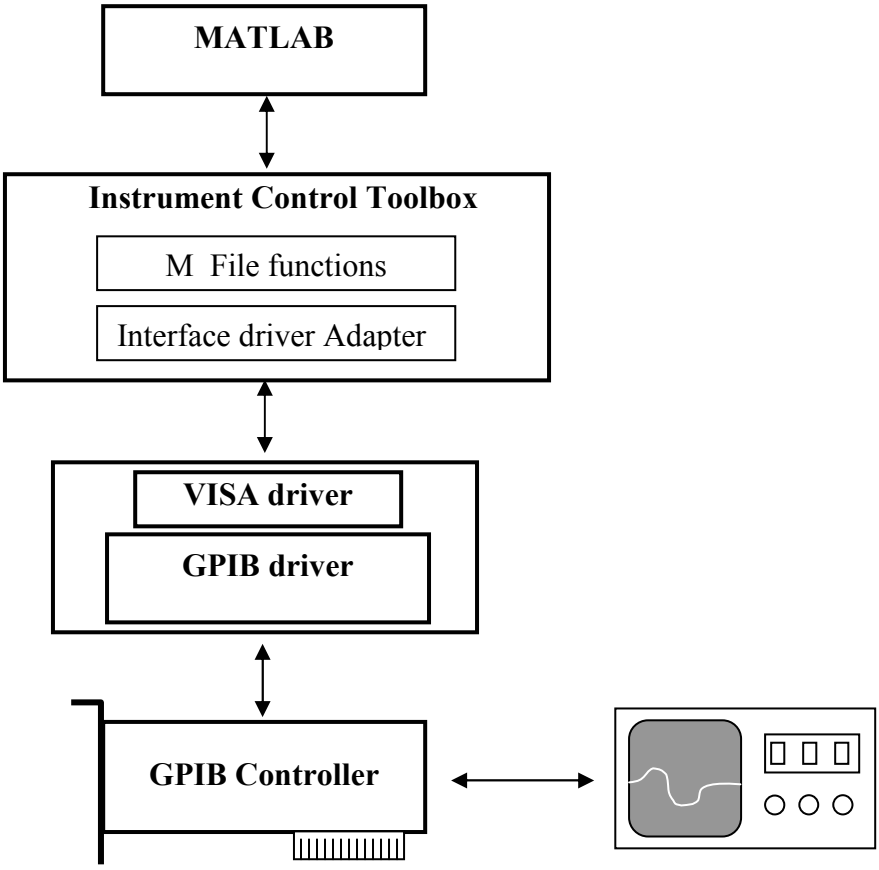
El estándar IEEE-488.2 un conjunto de ordenes básicas que realizan funciones comunes a todos los equipos, con independencia de su naturaleza.

Orden	Nombre de la orden	Función
*CLS	Clear Status Command	*-Despeja el registro de estado y los registros de incidencia.*
*ESE	Event Status Enable Command	*-Habilita bits del registro de habilitación de incidencias*
*ESE?	Event Status Enable Query	*-Interroga el registró de habilitación de Incidencias estándar.*
*ESR?	Event Status Register Query	*-Interroga el registro de Incidencias estándar.*
*IDN	Identification Query	*-Identifica tipo de instrumento y versión software.*
*LRN?	Learn Device Setup Query	*-Requiere el estado actual del equipo.*
*OPC	Operation Complete Command	*-Fija el bit de "Operación completa" del registro estándar.*
*OPC?	Operation Complete Query	*-Responde con "1" si se han ejecutado ordenes previas.*
*OPT?	Option Identification Query	*-Requiere la opción instalada en el equipo.*
*RCL	Recall Command	*-Restaura el estado del equipo del registro save/recall.*
*RST	Reset Command	*-Sitúa al equipo en el estado básico de referencia.*
*SAV	Save Command	*-Almacena el estado actual en un registro save/recall.*
*SRE	Service Request Enable Command	*- Habilita los bits del registro de habilitación de Byte de estado.*
*SRE?	Service Request Enable Query	*- Requiere el contenido del registro SER de habilitación del byte de estado*
*STB?	Read Status Byte Query	*- Requiere el estado del registro resumido del Byte de estado.*
*TRG	Trigger Command	*- Arranca o dispara la operación del equipo de forma remota.*
*TST?	Selft-Test Query	*-Requiere el resultado del autotest del equipo.*
*WAI	Wait-to-Continue Command	*- Espera a que se realicen todas las operaciones pendientes.*

IV INTERFACES CON EL BUS GPIB

IV.1 Componentes de una interfaz.

Para el control desde un computador de un sistema constituido por un conjunto de equipos todos ellos interconectados a través de un bus GPIB, se necesita incorporar al computador varios componentes hardware y software:



GPIB Controller : Es la tarjeta hardware incorporada al computador, con capacidad de controlar, leer y escribir sobre las líneas físicas que constituyen el bus GPIB.

Se utilizan dos tipos de tarjetas:

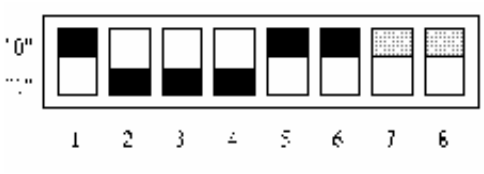
Tarjeta 82350 B PCI (Agilent): Es un controlador para PCI que soporta Plug&play.

Requerimientos mínimos del sistema: Windows 95/98 /Me/2000/XP, Pentium -200, 32 MB RAM, 50 MB de espacio libre de disco, PCI port .

Estandars que soporta: IEEE-488.1 y IEEE-488.2 compatible, SICL yVISA 2.2.

Tarjeta HP-82335 (Hewlett-Packard): Interface para bus ISA que no soporta Plug&Play ni la interfaz VISA.

Configuración: Antes de ser instaladas, esta tarjeta debe ser configuradas, a fin de que ocupen zonas de memoria independientes y de que no se solapen con las zonas de memoria utilizadas por otros periféricos. Esta configuración se realiza mediante una combinación adecuada en un conmutador tipo DIP interno que posee la tarjeta. Como se indica en la siguiente tabla, a cada combinación válida se le asocia un código que identifica a la tarjeta y que se denomina SELECT_CODE (Sel_code). Este código es la clave a través del que el driver identifica la interfaz sobre l que debe ejecutar las funciones que se programen. En nuestro caso, en cada computador existe una sola, interfaz, que siempre se configura con **SELECT_CODE = 7**.



b5	b6	N. Int.	Conflictos
0	0	3	**Com2**
0	1	4	Com
1	0	5	LPT2
1	1	7	LPT1

b1	b2	b3	b4	Sel_code	D.Memoria
0	0	0	1	3	CC000
0	1	0	0	4	D0000
0	1	0	1	5	D4000
0	1	1	0	6	D8000
0	1	1	1	7	DC000

GPIB Driver : Para poder acceder al hardware debe instalarse el driver correspondiente en el sistema operativo. Este puede ofrecer o no la interfaz VISA.

Para comprobar desde Matlab el hardware y los drivers disponibles se puede utilizar la función "**instrhwinfo**". Esta orden nos devuelve la lista de las interfaces soportadas por el entorno aunque no estén instaladas en el ordenador. Para obtener la información acerca de una determinada interfaz, debemos incluir como parámetro de la orden el identificador de dicha interfaz. La información que visualiza incluye los adaptadores instalados y la sintaxis para crear los objetos Matlab de control de la instrumentación.

En el laboratorio el driver con interfaz VISA es "GPIB0".

Interface Driver Adapter: Es un objeto Matlab que permite formular la comunicación con un instrumento, y por tanto permite su manejo, a través del entorno de programación Matlab. Representa el enlace a través del que se pasa la información entre Matlab y el driver correspondiente.
Cada objeto está asociado a un controlador y a un solo instrumento, por lo que será necesario crear uno por cada instrumento que forme parte del sistema.

IV.2 Fases de una sesión de comunicación con un instrumento.

Las fases que se siguen durante toda sesión de comunicación con un instrumento son las siguientes:

1. Creación de los objetos GPIB o VISA

Creación de GPIB object:

obj= gpib('vendor', boardindex, primaryaddress)

siendo: **obj**: Objeto Matlab GPIB.
'vendor': Nombre del vendedor del driver {'agilent', 'cec', 'iotech', 'keithley', 'mcc', 'ni'}
boardindex: El indice de la tarjeta GPIB
primaryaddress: El GPIB address primario {1 – 30}

Ejemplo: Creación de un objeto GPIB de un objeto Matlab para controlar un equipo con el address 15.

```
generador= gpib('agilent',7,15);
```

Creación de un VISA object:

obj= visa('vendor', rsrcname)

siendo: **obj**: Objeto Matlab VISA.
'vendor': Nombre del vendedor del driver {'agilent', 'ni', 'tec'}
rsrcname: El nombre del recurso VISA que controla el instrumento:

GPIB[board::primary_address::[secondary_address]::INSTR

Ejemplo: Creación de un objeto VISA para controlar por GPIB un generador que tiene asignado el address 15 en el bus.

```
generador= visa('agilent', GPIB0::15::INSTR)
```

2. Configuración de las propiedades del GPIB Object y VISA Object

Propiedades de GPIB Object y VISA Object:

Todo objeto GPIB o VISA tiene un conjunto de propiedades que pueden ser utilizadas, bien para leer su estado o bien para configurar el modo de comunicación con los instrumentos.

Write Properties
BytesToOutput: Indicate the number of bytes currently in the output buffer.
OutputBufferSize: Specify the size of the output buffer in bytes.
Timeout: the waiting time to complete a read or write operation.

TransferStatus: Indicate if an asynchronous read or write operation is in progress.
ValuesSent: Indicate the total number of values written to the instrument.

Read Properties
BytesAvailable: Indicate the number of bytes available in the input buffer.
InputBufferSize: Specify the size of the input buffer in bytes.
Timeout: Specify the waiting time to complete a read or write operation.
TransferStatus: Indicate if an asynchronous read or write operation is in progress.
ValuesReceived: Indicate the total number of values read from the instrument.

Callback Properties
BytesAvailableFcn: Specify the M-file callback function to execute when a specified number of bytes are available in the input buffer, or a terminator is read.
BytesAvailableFcnCount: Specify the number of bytes that must be available in the input buffer to generate a bytes-available event.
BytesAvailableFcnMode: Specify if the bytes-available event is generated after a specified number of bytes are available in the input buffer, or after a terminator is read.
ErrorFcn: Specify the M-file callback function to execute when an error event occurs.
OutputEmptyFcn Specify the M-file callback function to execute when the output buffer is empty.
TimerFcn: Specify the M-file callback function to execute when a predefined period of time passes.
TimerPeriod: Specify the period of time between timer events

Recording Properties
RecordDetail: Specify the amount of information saved to a record file
RecordMode: Specify whether data and event information are saved to one record file or to multiple record files.
RecordName: Specify the name of the record file.
RecordStatus: Indicate if data and event information are saved to a record file.

General Purpose Properties
ByteOrder: Specify the order in which the instrument stores bytes.
Name: Specify a descriptive name for the instrument object..
Status: Indicate if the instrument object is connected to the instrument.
Tag: Specify a label to associate with a instrument object .
Type: Indicate the object type.
UserData: Specify data that you want to associate with a instrument object.

Propiedades específicas
BoardIndex (GPIB y VISA): Specify the index number of the GPIB board.
BusManagementStatus (GPIB):Indicate the state of the GPIB bus management lines.
CompareBits (GPIB): Specify the number of bits that must match the EOS character to complete a read operation, or to assert the EOI line.
EOIMode (GPIB y VISA): Specify if the EOI line is asserted at the end of a write operation.
EOSCharCode (GPIB y VISA): Specify the EOS character
EOSMode (GPIB y VISA): Specify when the EOS character is written or read.
HandshakeStatus (GPIB y VISA): Indicate the state of the GPIB handshake lines.

RsrcName (VISA): Indicate the resource name for a VISA instrument.
PrimaryAddress (GPIB y VISA): Specify the primary address of the GPIB instrument.
SecondaryAddress (GPIB y VISA): Specify the secondary address of the GPIB instrument.

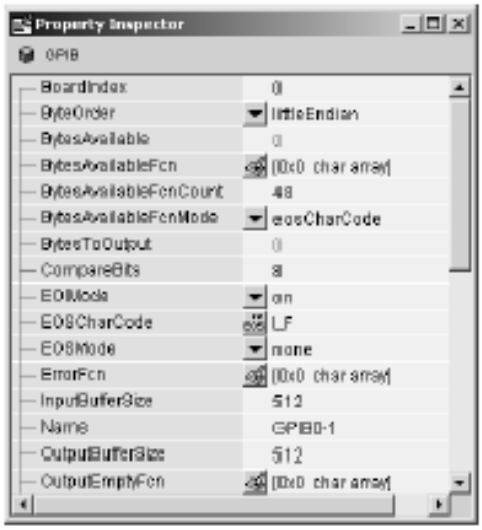
La propiedades de un objeto pueden leerse desde un programa Matlab mediante las sentencias:

Get(generador, 'BoardIndex') o generador.BoardIndex

Así mismo, las propiedades de un objeto pueden establecerse mediante las sentencias:

Set(generador, 'Timeout', 5.0) o generador.Timeout= 5.0

Matlab ofrece una aplicación para visualizar/cambiar las propiedades del un objeto GPIB o VISA de forma interactiva. Para ello seleccionar el objeto en el Workspace Browser e invocar la herramienta Explore-> Call Property Inspector:



Valores por defecto Todas la variables tienen definido un valor por defecto, por lo que si no son establecidas en el proceso de configuración, son estos valores los que permanecen.

Los valores por defecto de algunas propiedades importantes son:

ByteOrder: [**littleEndian** | bigEndian]
EOIMode: [**on** | off]
EOSMode: [**none** | read | write | read&write]

3. Conexión del objeto al instrumento

Una vez que se ha creado el objeto, debemos realizar la conexión del mismo con el instrumento al que corresponde. Este paso se realiza mediante la sentencia:

```
fopen(generador)
```

Esta sentencia cambia el estado de la propiedad generator.Status de 'closed' a 'open'.

Hay que tener cuidado con el momento en que realizamos la conexión, pues existen propiedades del objeto, tales como *InputBufferSize* or *OutputBufferSize*, que solo se pueden establecer antes de la llamada a fopen.

4. Comunicación con un instrumento VISA o GPIB.

Comunicar con un instrumento implica transferir o recibir mensajes ASCII o bloques de datos binarios con el instrumento. Esto se realiza a través de un conjunto de funciones que ofrece Matlab y que se aplican al objeto que constituye el adaptador del instrumento.

La comunicación puede realizarse:

Síncronamente: La función que invoca la comunicación se suspende hasta que esta concluye y luego retorna el control.

Asíncronamente: La función retorna inmediatamente el control al programa sin que su efecto haya concluido. Posteriormente se accede de nuevo al objeto para verificar el estado de aquella, o se responde mediante funciones "callback" a los eventos que proceden de los instrumentos. Esta forma de comunicación solo tiene sentido si se realiza un estrategia de programación concurrente. No la utilizaremos en este curso.

Operaciones de Escritura.

Las funciones que ofrece Matlab para enviar datos a un instrumento son:

Function Name	Description
binblockwrite	Write binblock data to the instrument.
fprintf	Write text to the instrument.
fwrite	Write binary data to the instrument.
stopasync	Stop asynchronous read and write operations.

Cuando se va a realizar una operación de escritura sobre un instrumento, el proceso conlleva dos fases:

- Los datos a escribir son enviados al buffer de salida.
- Los datos del buffer son enviados al instrumento.

Las propiedades del objeto que afecta a estas operaciones, son:

Property Name	Description
BytesToOutput	Indicate the number of bytes currently in the output buffer.
OutputBufferSize	Specify the size of the output buffer in bytes.
Timeout	Specify the waiting time to complete a read or write operation.
TransferStatus	Indicate if an asynchronous read or write operation is in progress.
ValuesSent	Indicate the total number of values written to the instrument.

Ejemplos de transferencia de información a un instrumento son:

```
% ================================================
% Operaciones previas que afectan a la operaciones de escritura:
% Creación del instrumento
generador=visa('agilent','GPIB0::10::0::INSTR');
% Configuración propiedades relacionadas con escritura
generador.Timeout=2.0;
generador.EOIMode='on';
generador.EOSMode='none'
% ================================================
% Operaciones de escritura:
fprintf(generador,'*RST');
fprintf(generador,'APPLY:Square 5 khz,5 vpp');
% ================================================
```

Operaciones de lectura

Las funciones que ofrece Matlab para leer datos de un instrumento son:

Function Name	Description
binblockread	Read binblock data from the instrument.
fgetl	Read one line of text from the instrument and discard the terminator.
fgets	Read one line of text from the instrument and include the terminator.
fread	Read binary data from the instrument.
fscanf	Read data from the instrument, and format as text.
readasync	Read data asynchronously from the instrument.
scanstr	Read data from the instrument, format as text, and parse
stopasync	Stop asynchronous read and write operations.

Cuando se va a realizar una operación de lectura sobre un instrumento, el proceso conlleva dos fases:

- Los datos leídos del instrumento son almacenados en el buffer de entrada.
- Los datos del buffer de entrada se almacenan en la variable MATLAB.

Las propiedades del objeto que afecta a estas operaciones son:

Property Name	Description
BytesAvailable	Indicate the number of bytes available in the input buffer.
InputBufferSize	Specify the size of the input buffer in bytes.
ReadAsyncMode	Specify whether an asynchronous read is continuous or manual (serial port, TCP/IP, UDP, and VISA-serial objects only).
Timeout	Specify the waiting time to complete a read or write operation.
TransferStatus	Indicate if an asynchronous read or write operation is in progress.
ValuesReceived	Indicate the total number of values read from the instrument.

Ejemplos de lectura de información desde un instrumento son:

```
% ================================================
% Operaciones previas que afectan a la operaciones de escritura:
oscilo=visa('agilent','GPIB0::5::0::INSTR');
oscilo.Timeout=2.0;
oscilo.EOIMode='on';
oscilo.EOSMode='none'
oscilo.inputBufferSize=4256;
% ================================================
% Operaciones de lectura.
% Lectura de un string.
fprintf(oscilo, 'MEASURE:VPP?');
Vpp= fscanf(oscilo);
% ================================================
% Lectura de un bloque de datos binarios.
fprintf(oscilo, 'DIGITIZE Channel1');
fprintf(oscilo, 'WAVEFORM:DATA?');
s= binblockread(oscilo);
% ================================================
```

5. Desconexión y destrucción de un GPIB object y VISA object:

Cuando se haya concluido la comunicación con un instrumento dentro de un programa, el adaptador debe ser cerrado:

```
fclose(generador)
```

Esta sentencia cambia el estado de la propiedad generator.Status de 'open' a 'closed'.

El objeto se destruye y la variable se elimina de memoria con las sentencias:

```
delete(generator)
clear generator
```

Importante:
Si se invocan las sentencias delete o clear antes de que se haya invocado fclose, las variables son eliminadas de memoria, pero permanece establecida la conexión con el instrumento (sin que este sea ya accesible mediante una variable Matlab), y en futuras conexiones se plantearan errores. Si tal cosa ocurre, se debe ejecutar (fuera de programa) la sentencia:

```
fclose(instrfind)
```

con lo que se cierran la conexión con todos los instrumentos.

IV.3 Documentación.

La información completa relativa al control de instrumentos desde Matlab se puede encontrar en **Instrument Control Toolbox** .