

## 4 Sistemas basados en el bus VME y VXI

### 4.1 Del VME al VXI

A finales de la década de los setenta, muchos fabricantes diseñaban y construían sus propios sistemas de instrumentación modulares o en tarjeta. Sin embargo, la mayor parte de estos equipos utilizaban arquitecturas propias y sólo algunas soportaban los equipos de más de un fabricante.

La aparición previa del estándar del bus VME para sistemas basados en microprocesador, junto con el deseo del Departamento de Defensa de los Estados Unidos de reducir el tamaño de sus equipos de prueba automáticos, propiciaron la aparición del bus VXI. Para conseguirlo recurrieron al bus VME, que en aquel momento, gracias a su estructura modular, y velocidad de transferencia de datos, tenía un éxito comercial importante. Era particularmente útil en el diseño de aplicaciones de test digital y procesado digital de señal.

No obstante, la limitación más importante del bus VME era que su diseño no estaba orientado al diseño de sistemas de instrumentación, en especial no incorporaba líneas analógicas ni de sincronización. Para solventar este problema se creó en abril de 1987 un comité de cinco empresas fabricantes de instrumentación (Colorado Data Systems, Hewlett Packard, Racal Dana, Tektronix y Wavetek). Cuatro meses después se definía un nuevo estándar abierto para instrumentación basado en el bus VME llamado VXI (VME bus **E**xtensions for **I**nstrumentation).

El bus VXI es una arquitectura abierta para todos los fabricantes de instrumentos modulares. Esta especificación pública de sistema permite la coexistencia y el funcionamiento de un amplio margen de instrumentos dentro del mismo bastidor. Mientras que el estándar IEEE-488 es principalmente un estándar de comunicaciones para facilitar la integración del sistema, el bus VXI es un estándar de sistema. En la norma VXI no se define únicamente el protocolo de comunicaciones y la conexión física de dichos instrumentos, se definen también aspectos mecánicos, eléctricos, de compatibilidad electromagnética y tipos de instrumentos.

El bus VXI está orientado a aplicaciones que requieren una capacidad de integración y velocidad que se encuentran fuera de las capacidades del bus IEEE-488. Sin embargo, la enorme popularidad de la norma IEEE-488 se tuvo en cuenta en la definición de la norma VXI y se utilizó como modelo para el protocolo de comunicaciones entre dispositivos e instrumentos.

---

## 4.2 Especificaciones generales de la norma VXI

Desde la primera definición de la norma en el año 1987 han ido apareciendo sucesivas versiones que incorporaban ligeras modificaciones sobre el borrador original, de las cuales la última versión ha sido la 1.4, correspondiente a abril de 1992, que se ha consolidado con el reconocimiento por parte del IEEE como estándar IEEE-1155. El consorcio original de cinco empresas se ha ampliado con nuevos miembros como Bruel & Kjaer, Fluke, GenRad, Keythley y National Instruments.

Para la definición de la norma se tomó como base el bus VME (IEEE-1014). Su estructura jerárquica, velocidad, flexibilidad y la existencia de multitud de tarjetas ya disponibles lo hacen ideal en este contexto. La primera consecuencia es la posibilidad de integrar dispositivos o sistemas VME ya existentes en futuras aplicaciones basadas en VXI.

La norma VXI define las características mecánicas, eléctricas, los protocolos y los procesos de inicialización así como las peculiaridades de los posibles interfaces IEEE-488 que puedan emplearse para la comunicación con elementos de control externos.

Dentro de la norma VXI se pueden distinguir un conjunto de nueve especificaciones que hacen referencia a diferentes aspectos del diseño de sistemas VXI y son:

- VXI-0 Visión general de las especificaciones del bus VXI. Revisión 1.0 mayo 1992.
- VXI-1 Especificación de un sistema basado en el bus VXI. Revisión 1.4, abril 1992.
- VXI-2 Especificación para dispositivos extendidos basados en registros y para dispositivos VXI de memoria extendidos basados en registros. Revisión 1.0, febrero 1991.
- VXI-3 Especificación de comandos de palabra serie para la identificación de la versión y el número de serie de dispositivos VXI. Revisión 1.0, febrero 1991.
- VXI-4 Especificación de mnemotécnicos comunes en la norma VXI. Revisión 1.0, junio 1991.
- VXI-5 Especificación de los comandos ASCII comunes del sistema. Revisión 1.0, junio 1991.
- VXI-6 Especificación de la interfase estándar para la extensión del bus VXI. Revisión 1.0, febrero 1991.
- VXI-7 Especificación del formato de datos de memoria compartida. Revisión 1.0, marzo 1992.
- VXI-9 Especificación del protocolo de memoria compartida para sistemas VXI. Revisión 1.0, mayo 1992.

Todas ellas están recogidas y publicadas dentro de un documento conocido como *VXIbus System Specifications Revision 1.4*. El núcleo principal de la norma, donde se comprende la configuración y el funcionamiento de un sistema VXI, es la especificación VXI-1.

## 4.3 Descripción de los buses VXI y VME

La norma VXI constituye una extensión de la norma VME. Por tanto, los dispositivos VME son un subconjunto dentro de los instrumentos VXI.

---

### 4.3.1 El bus VME

El bus VME es bus asíncrono formado por cuatro sub-buses: transferencia de datos, arbitraje, interrupciones y utilidades.

#### - Bus de transferencia de datos:

Es un bus asíncrono de datos que puede transferir palabras de 8, 16 o 32 bits entre módulos. Está compuesto de un conjunto de 32 líneas de dirección (A1-A32), 32 líneas de datos (D1-D32) y líneas de control.

El espacio de direcciones puede configurarse de tres maneras diferentes:

- Direccionamiento corto (A16) 64 kbyte
- Direccionamiento estándar (A24) 16 Mbyte
- Direccionamiento extendido (A32) 4 Gbyte

#### - Bus de arbitraje de transferencia de datos:

Permite transferir el control del bus de datos entre diferentes dispositivos maestros de una manera ordenada y garantizando que únicamente un dispositivo maestro controla el bus de datos en cualquier instante.

#### - Bus de interrupciones

Dispone de siete líneas de interrupción (IRQ1-IRQ7) con diferentes niveles de prioridad y un máximo de siete controladores de interrupción. Permite a los dispositivos realizar sus peticiones de interrupción y que éstas sean reconocidas por el módulo controlador del sistema.

#### - Bus de utilidades

Suministra la alimentación, las señales de inicialización del sistema, detección de fallos en la alimentación, señales de reloj y una línea auxiliar de transmisión de datos serie.

Dentro de la norma VME se definen tarjetas de dos dimensiones estándar diferentes (figura 4.1); Eurocard, tamaño A (100 mm x 160 mm), y doble Eurocard, tamaño B (233 mm x 160 mm), figura 4.1. Las señales descritas están disponibles en dos conectores denominados P1 y P2 de 96 contactos cada uno. En el primer conector se utilizan las 96 líneas y del segundo (P2) se utilizan las 32 líneas centrales del conector. Las tarjetas de tamaño A incluyen únicamente el conector P1, que es la configuración mínima de funcionamiento. El espacio de direcciones en la tarjeta de tamaño A puede ser el A24 (16 Mbytes) o A16 (64 kbytes) con palabras de 8 o 16 bits.

El tamaño B dispone de los dos conectores P1 y P2 y el espacio de direcciones puede ser el A16, A24 o A32 (4 Gbytes) con palabras de 8, 16 o 32 bits.

---

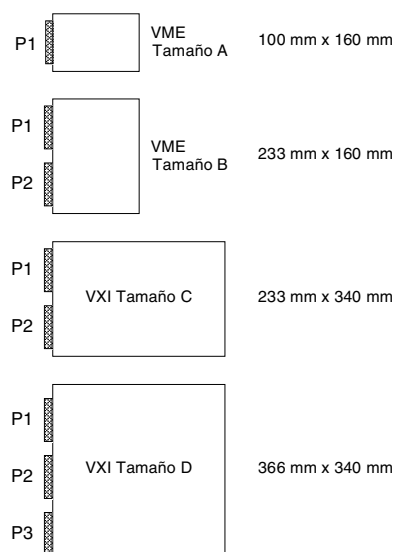


Fig. 4.1 Tarjetas estándar VXI y VME

#### 4.3.2 Extensión del bus VME, el bus VXI

La extensión del bus VME, por parte de la norma VXI, a un bus para instrumentación modular contempla diversos aspectos.

El primero de ellos es el mecánico. Se definen dos nuevos tamaños además de los ya incorporados en la norma VME, y son el C (233,35 mm x 340 mm) y el D (366,7 mm x 340 mm) a los que se les añade un nuevo conector P3. El segundo define los sub-buses que se han añadido utilizando las dos líneas de contactos del conector P2, que quedan libres en la norma VME, y el conector P3. Los nuevos sub-buses son:

##### Bus de reloj

Está formado por dos señales de reloj de 10 MHz y 100 MHz y una señal de sincronización de reloj para esta última. La señal de 10 MHz (CLK10) está disponible en el conector P2 y la señal de 100 MHz (CLK100) junto con la de sincronización (SYNC100) están ubicadas en el conector P3. CLK10, CLK100 y SYNC100 son señales diferenciales con niveles ECL y disponen de un amplificador en el bus para cada módulo para aumentar el aislamiento entre módulos y reducir el efecto de carga sobre la señal de reloj. El controlador del bus VXI, el módulo 0, puede generar estos relojes o pueden ser suministrados a partir de un generador de frecuencia patrón a través del panel frontal del módulo 0. La señal SYNC100 permite sincronizar diferentes módulos con un determinado flanco de subida de la señal CLK100.

##### Bus en estrella

Situado en el conector P3, está formado por doce pares de líneas (STARX<sub>n</sub> y STARY<sub>n</sub>) que unen respectivamente cada uno de los módulos con el módulo 0 como muestra la figura 4.2. Además son líneas bidireccionales diferenciales con niveles ECL.

El bus en estrella ofrece una vía de comunicación de alta velocidad entre módulos para procesos en los cuales la temporización es extremadamente crítica. La norma VXI fija un retardo máximo de 5 ns entre cualquier módulo y el módulo 0 y una desviación de 2 ns entre cualquier par de señales STARXn-STARXn.

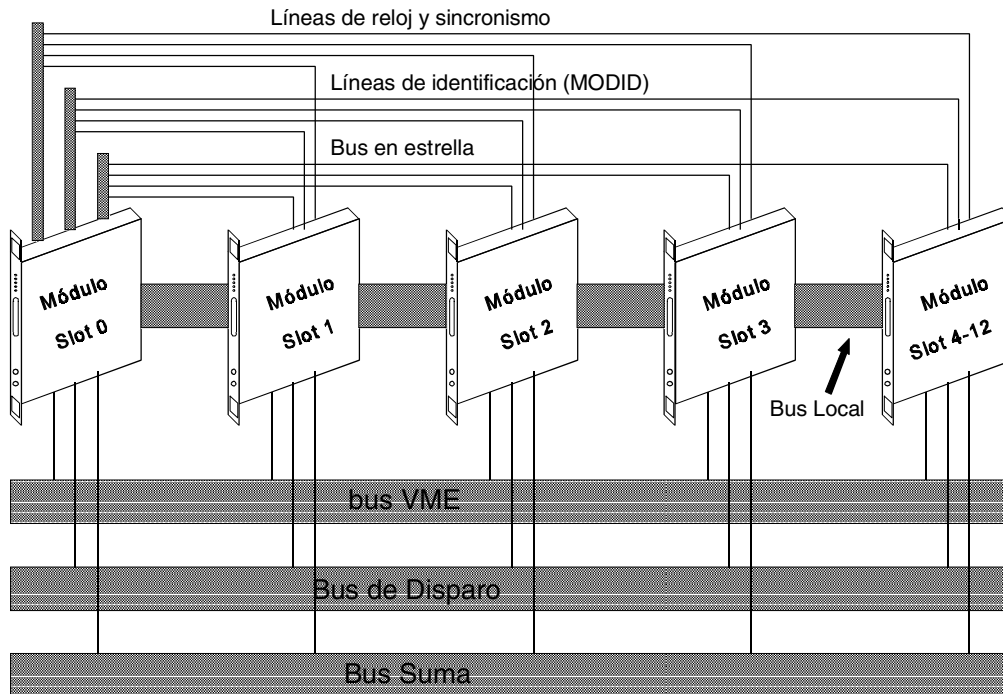


Fig. 4.2 Esquema de la estructura completa de un BUS VXI

### Bus de disparo

El bus de disparo puede subdividirse en 8 líneas de disparo con niveles TTL (TTLTRG\*) y 6 líneas con niveles ECL (ECLTRG). Todas las líneas TTLTRG\* y dos de las ECLTRG están situadas en el conector P2. Las cuatro restantes ECLTRG están en el conector P3.

Su utilización por un instrumento o grupo de ellos posibilita la implementación de complejos patrones de disparo y temporización. Se han definido diversos protocolos para la coordinación y comunicación entre módulos. Además de señales de disparo y reloj es posible el intercambio de información digital a través de estos buses mediante la agrupación de líneas con el fin de descargar y complementar al bus VME. La velocidad de transmisión en las líneas TTLTRG\* puede llegar a una frecuencia máxima de 12,5 MHz mientras que en las ECLTRG este límite llega a 62,5 MHz.

Las líneas TTLTRG\* son TTL en colector abierto y están terminadas en cada uno de los extremos del bus con una red pasiva. El nivel lógico alto, 5 V, viene fijado por las terminaciones del bus. Para la temporización de procesos mediante estas líneas la norma recomienda utilizar los flancos descendentes de las señales al tener un tiempo de bajada menor por la configuración de colector abierto.

La norma define seis protocolos para las señales TTLTRG\*:

- Disparo síncrono: una única línea transmite una señal de disparo que no requiere una validación por parte de ningún receptor. La velocidad de repetición máxima es de 12,5 MHz.
- Disparo semisíncrono: una única línea transmite una señal de disparo, nivel bajo, y múltiples receptores validan la señal poniendo la misma línea a nivel bajo.
- Disparo asíncrono: Se utilizan dos líneas. Hay una única fuente de disparo y un único receptor para validar la señal de disparo.
- Transmisión de reloj: Cualquier línea TTLTRG\* puede utilizarse para la transmisión de una señal de reloj desde 0 Hz a 12,5 MHz.
- Transmisión de datos: Una o más líneas TTLTRG\* pueden agruparse para la transmisión de datos en paralelo. Una de estas líneas se usa como reloj o disparo.
- Protocolo *Start/Stop*: Una línea TTLTRG\* es controlada por el módulo *Slot 0* y su estado significa el inicio o el final de la operación de otros módulos.

### **Bus local**

Está formado por 36+36 líneas (12+12 en el conector P2 y 24+24 en P3) que comunican cada módulo únicamente con los inmediatamente adyacentes y con el que se pueden realizar transferencias de hasta 200 MHz, proporcionando una velocidad de transmisión efectiva de 1 Gbyte/s. Su función es la de enlazar localmente diferentes tarjetas a gran velocidad, por ejemplo un módulo digitalizador y un módulo de procesado de señal.

Al no estar ligado al conjunto de los módulos, la definición de las señales transferidas se realiza según las necesidades de cada dispositivo, estando prevista la transferencia de señales digitales TTL y ECL y analógicas de bajo, medio y alto nivel (-42 V a 42 V).

### **Bus de suma analógica**

Está presente en el conector P2 y forma un nodo común a todos los módulos del sistema. Está terminado con una resistencia de 50  $\Omega$  conectada a la masa de señal en cada extremo del bus. La señal de suma analógica está situada en un extremo del conector P2, alejada de las señales digitales y apantallada por tres terminales de masa. Está previsto que cualquier módulo pueda enviar o recibir de esta línea. Cada módulo puede transmitir a esta línea mediante una fuente de corriente con una impedancia de salida mayor de 10 k $\Omega$  con una capacidad en paralelo menor de 4 pF. El receptor deberá tener una impedancia equivalente de entrada mayor de 10 k $\Omega$  con una capacidad en paralelo menor de 3 pF.

Un ejemplo de utilización es la generación de formas de onda complejas a partir de las salidas de diferentes módulos que se combinaría de forma aditiva en el bus suma. La ventaja de este método es que permite combinar diferentes módulos de generación de señal para crear una forma de onda compleja que, de otra forma, requeriría un instrumento de mayor coste.

---

### **Bus de identificación de módulos**

El bus de identificación de módulos está formado por 12 líneas de P2 (MODID) conectadas cada una de ellas a una de las 12 ranuras disponibles en el bus VXI para la conexión de módulos. Estas líneas se alimentan desde el módulo conectado en la ranura 0 y permiten la detección de la existencia de los dispositivos conectados al bus, incluso en el caso de fallo de los mismos, ya que se basa en la detección de un contacto a masa. Es de gran utilidad en el diagnóstico de fallos y en el proceso de autoconfiguración en los módulos que dispongan del conector P2.

### **Bus de alimentación**

El bus de distribución de la alimentación puede suministrar hasta 268 W a un único módulo que tenga los conectores P1, P2 y P3. Esta potencia se entrega en siete tensiones reguladas diferentes. El bus VXI añade con respecto a la norma VME tensiones de +24 V y -24 V para la alimentación de circuitos analógicos, y -5.2 V y -2 V para circuitos ECL.

En la figuras 4.2 y 4.3 puede verse una esquema de la estructura completa de sub-buses del bus VXI y la distribución de las diferentes señales en los conectores P1, P2 y P3.

## **4.4 Modos de funcionamiento y arquitecturas del bus VXI**

En el apartado anterior se han descrito los recursos físicos disponibles en la norma VXI que se basan en un bus específico. La norma también define las relaciones entre los componentes del sistema y las estructuras que permiten la realización de configuraciones efectivas.

Un sistema VXI puede estar constituido por un máximo de 256 dispositivos incluyendo uno o más subsistemas VXI. Cada subsistema incluye un módulo central o ranura 0 (SLOT0), que genera las señales de reloj, temporización y control del sistema, y hasta 12 instrumentos o módulos adicionales. Todo el subsistema se encuentra ubicado en un bastidor normalizado.

Los dispositivos son los elementos básicos de un sistema VXI; están formados generalmente por un solo módulo, pero pueden existir varios dispositivos en un solo módulo, o bien un dispositivo puede ocupar varios módulos. Cada uno de los 256 dispositivos que pueden existir en un sistema VXI tiene una dirección lógica asociada entre 0 y 255. A cada dispositivo se le adjudican 64 direcciones absolutas de memoria en el mapa de direcciones A16, de 64 kbytes. La dirección lógica del dispositivo define la dirección de este segmento de 64 bytes. Estos 64 bytes contienen los registros de configuración y los registros de comunicación del dispositivo.

### **4.4.1 Tipos de dispositivos en un sistema VXI**

En la norma VXI se ha buscado, ante todo, una gran flexibilidad que facilite su adaptación a los más variados entornos y aplicaciones. Se han definido cuatro tipos diferentes de dispositivos VXI: basados en mensajes, basados en registros, memoria y extendidos.

Los dispositivos de memoria proporcionan posiciones de almacenamiento de datos en bloques de memoria RAM o ROM. Permiten mover grandes cantidades de datos. Los dispositivos

---

Definición del conector P1 del bus VXI; Slot 0-12				Definición del conector P2 del bus VXI; Slot 0			
Número de PIN	Fila A Señal	Fila B Señal	Fila C Señal	Número de PIN	Fila A Señal	Fila B Señal	Fila C Señal
1	D00	BBSY*	D08	1	ECLTRG0	+5	CLK10+
2	D01	BCLR*	D09	2	-2V	GND	CLK10-
3	D02	ACFAIL*	D10	3	ECLTRG1	RSV1	GND
4	D03	BG0IN*	D11	4	GND	A24	-5,2V
5	D04	BG0OUT*	D12	5	MODID12	A25	LBUSC00
6	D05	BG1IN*	D13	6	MODID11	A26	LBUSC01
7	D06	BG1OUT*	D14	7	-5,2V	A27	GND
8	D07	BG2IN*	D15	8	MODID10	A28	LBUSC02
9	GND	BG2OUT*	GND	9	MODID09	A29	LBUSC03
10	SYSCLK	BG3IN*	SYSFAIL*	10	GND	A30	GND
11	GND	BG3OUT*	BERR*	11	MODID08	A31	LBUSC04
12	DS1*	BR0*	SYSRESET*	12	MODID07	GND	LBUSC05
13	DS0*	BR1*	LWORD*	13	-5,2V	+5V	-2V
14	WRITE*	BR2*	AM5	14	MODID06	D16	LBUSC06
15	GND	BR3*	A23	15	MODID05	D17	LBUSC07
16	DTACK*	AM0	A22	16	GND	D18	GND
17	GND	AM1	A21	17	MODID04	D19	LBUSC08
18	AS	AM2	A20	18	MODID03	D20	LBUSC09
19	GND	AM3	A19	19	-5,2V	D21	-5,2V
20	IACK*	GND	A18	20	MODID02	D22	LBUSC10
21	IACKIN*	SERCLK(1)	A17	21	MODID01	D23	LBUSC11
22	IACKOUT*	SERDAT*(1)	A16	22	GND	GND	GND
23	AM4	GND	A15	23	TTLTRG0*	D24	TTLTRG1*
24	A07	IRQ7*	A14	24	TTLTRG2*	D25	TTLTRG3*
25	A06	IRQ6*	A13	25	+5V	D26	GND
26	A05	IRQ5*	A12	26	TTLTRG4*	D27	TTLTRG5*
27	A04	IRQ4*	A11	27	TTLTRG6*	D28	TTLTRG7*
28	A03	IRQ3*	A10	28	GND	D29	GND
29	A02	IRQ2*	A9	29	RSV2	D30	RSV3
30	A01	IRQ1*	A8	30	MODID00	D31	GND
31	-12V	+5VSTDBY	+12V	31	GND	GND	+24V
32	+5V	+5	+5V	32	SUMBUS	+5V	-24V

Definición del conector P3 del bus VXI; Slot 0			
Número de PIN	Fila A Señal	Fila B Señal	Fila C Señal
1	ECLTRG2	+24V	+12V
2	GND	-24V	-12V
3	ECLTRG3	GND	RSV4
4	-2V	RSV5	+5V
5	ECLTRG4	-5,2V	RSV6
6	GND	RSV7	GND
7	ECLTRG5	+5V	-5,2V
8	-2V	GND	GND
9	STARY12+	+5V	STARX01+
10	STARY12-	STARX01-	STARX01-
11	STARX12+	STARX12-	STARX01+
12	STARY11+	GND	STARX02+
13	STARY11-	STARX02-	STARX02-
14	STARX11+	STARX11-	STARX02+
15	STARY10+	+5V	STARX03+
16	STARY10-	STARX03-	STARX03-
17	STARX10+	STARX10-	STARX03+
18	STARY09+	-2V	STARX04+
19	STARY09-	STARX04-	STARX04-
20	STARX09+	STARX09-	STARX04+
21	STARY08+	GND	STARX05+
22	STARY08-	STARX05-	STARX05-
23	STARX08+	STARX08-	STARX05+
24	STARY07+	+5V	STARX06+
25	STARY07-	STARX06-	STARX06-
26	STARX07+	STARX07-	STARX06+
27	GND	GND	GND
28	STARX+	-5,2V	STARY+
29	STARX-	GND	STARY-
30	GND	-5,2V	-5,2V
31	CLK100+	-2V	SYNC100+

Fig. 4.3 Distribución de señales en los conectores P1, P2 y P3 de un bus VXI

basados en registros disponen de una inteligencia limitada. Se comunican con su módulo controlador por medio de protocolos específicos definidos por el fabricante, en forma de escrituras directas en sus

registros de configuración o comunicación. Su principal desventaja es la dificultad de programación, y su mayor ventaja es la facilidad de diseño y coste reducido. Los dispositivos extendidos se encuentran reservados, y permiten la definición de nuevas subclases de dispositivos VXI en futuras ampliaciones de la norma.

Los dispositivos basados en mensajes, en contraste con los basados en registros, se comunican mediante mensajes de caracteres ASCII similares a los empleados por los instrumentos IEEE-488. El protocolo de comunicación está definido en la norma, denominado *Word Serial Protocol* (WSP), proporciona un soporte básico para la transmisión serie de los comandos de alto nivel. Actualmente se tiende a la estandarización de los comandos, de forma que dispositivos equivalentes de diferentes fabricantes puedan ser perfectamente intercambiables. Esta tendencia se recoge en la normativa SCPI (*Standard Commands for Programmable Instrumentation*), que se describe en el capítulo siguiente.

Un instrumento puede estar compuesto por más de un módulo. Los buses definidos en la norma son el soporte físico que permite el funcionamiento de los distintos módulos como un único instrumento. En cuanto al soporte lógico, software, es necesario que la acción de los componentes del instrumento sea coordinada adecuadamente y que la información se comparta de la forma más natural y transparente posible. La norma VXI contempla la constitución de estructuras jerárquicas basadas en dispositivos *Servants* y *Commanders* (ver figura 4.4). Varios dispositivos *Servants* dependen de un único dispositivo *Commander*. Un determinado *Commander* puede a su vez ser un *Servant* de otro. Los dispositivos *Servants* pueden ser de dos tipos, basados en registro o basados en mensajes. Sin embargo, los *Commander* ha de ser dispositivos basados en mensajes que permitan controlar a los *Servant*.

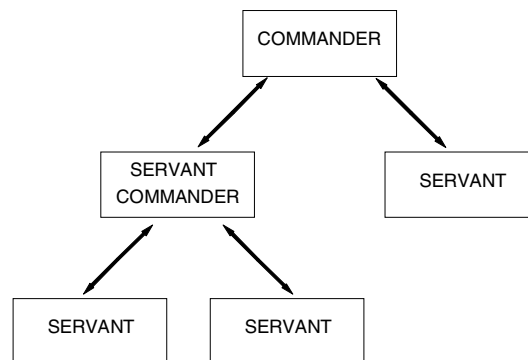


Fig. 4.4 Estructura e interrelaciones entre un 'comander' y un 'servant'

#### 4.4.2 Recursos del sistema

Como ya se ha comentado, la arquitectura del bus VXI ofrece un conjunto de recursos comunes al sistema. Estos recursos o funciones están centralizados en dos entidades conocidas como *Slot 0* y *Resource Manager*. Normalmente se implementan en un mismo módulo que, además, incluye la interfase de comunicación (IEEE-488, RS232 o MXI) o control adecuados.

El *Slot 0* debe generar las señales de reloj, CLK10, CLK100 y SYNC100 así como las señales del Bus de Identificación de Módulos (MODID) y comunicación STARXn, STARYn.

El *Resource Manager* tiene asignadas funciones de mas alto nivel. Es un dispositivo basado en mensajes con capacidad de *Commander* que debe estar situado en la dirección lógica 0 y que, al arrancar el sistema, realiza las siguientes funciones:

- Identificar todos los dispositivos VXI del sistema.
- Gestionar el autotest y la secuencia de inicialización del sistema.
- Configurar el mapa de direcciones A24 y A32 del sistema.
- Configurar las jerarquías *Commander/Servant* del sistema.
- Iniciar la operación normal del mismo.

#### 4.4.3 Protocolos de comunicación

La comunicación dentro de un sistema VXI se establece entre un *Commander* y cualquiera de sus *Servants* a iniciativa del primero. En el nivel superior de la estructura jerárquica se encuentra el controlador VXI o el dispositivo de comunicación entre el controlador externo y el bus VXI.

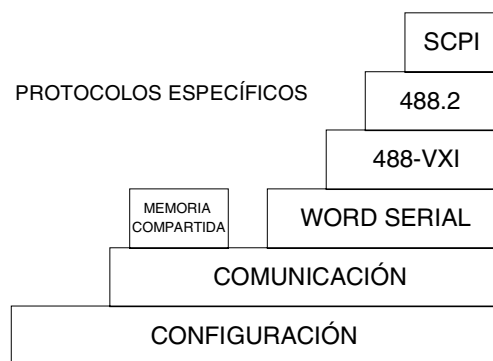


Fig. 4.5 Jerarquía de comunicación en un bus VXI

La comunicación con los dispositivos basados en registros, que sólo pueden actuar como *Servants*, no está especificada por la norma VXI y depende del fabricante del equipo. En este caso el sistema deberá incluir el dispositivo o los dispositivos *Commander* adecuados del fabricante para poder controlar el *Servant* basado en registros. La comunicación entre *Commanders* y *Servants* basados en mensajes está totalmente definida en la norma. El protocolo empleado es conocido como Word Serial. Permite el envío y la recepción de información secuencialmente de una forma similar a la usada en el entorno IEEE-488. Se trata de un protocolo asíncrono cuya velocidad se adapta al dispositivo más lento en cada momento. La longitud de estas palabras es de 16 bits, aunque existen variantes de 8, 32 o 48 bits. El protocolo hace uso del contenido de diversos registros en el *Servant* normalizados para la coordinación y configuración de las transmisiones.

Un protocolo alternativo es de memoria compartida. En este, zonas de memoria, del espacio de direcciones VME, pueden ser utilizadas por más de un dispositivo y usarse para la transferencia de grandes bloques de información. La memoria puede estar físicamente incluida en los propios dispositivos o estar en tarjetas de memoria conectadas a tal efecto al bus. En la figura 4.5 puede verse la estructura jerárquica de los diferentes niveles de comunicación para un sistema VXI.

#### 4.4.4 Arquitecturas de un sistema VXI

La norma VXI permite una gran variedad de configuraciones posibles. Las más típicas son:

- 1.- Un único controlador externo, *Host*, conectado al sistema a través del SLOT 0 y el *Resource Manager* mediante una interfase IEEE-488, RS232 o red local.
- 2.- Un único controlador interno que realiza además las funciones de SLOT0 y *Resource Manager*. Los controladores VXI existentes se basan, en su mayoría, en la arquitectura PC, con procesadores de la familia INTEL 80x86.
- 3.- Varios procesadores internos con control distribuido de sistemas.

La primera opción es la más frecuente. La disponibilidad de controladores adecuados, las velocidades de transferencia posibles y el hecho de que la mayoría de sistemas que incluyen equipamiento VXI sean mixtos al hacer uso de instrumentación IEEE-488, son las razones que avalan esta elección. En este caso el controlador se conecta a uno o varios subsistemas a través de un módulo de interfase adecuado presente en cada uno de ellos. Este dispositivo realiza además las funciones de SLOT 0 y *Resource Manager*. La norma especifica cómo ha de ser la traducción de protocolos entre el bus VXI y el IEEE-488 pero no define nada sobre los esquemas de direccionado, que quedan estos libres a elección del fabricante o diseñador. Las soluciones adoptadas hasta el momento han sido tres:

- a) Múltiples primarias: Cada instrumento VXI se identifica mediante una dirección primaria válida dentro del bus IEEE-488. Tiene como ventaja la similitud con la programación de instrumentación IEEE-488, pero el número de direcciones de dispositivos se limita a 30.
  - b) Múltiples secundarias: Las direcciones secundarias IEEE-488 se utilizan para extender el espacio de direccionamiento. En este caso una única dirección primaria se asignaría al bastidor y una dirección secundaria a cada dispositivo VXI.
  - c) Dirección incluida en los comandos, *Embedded Addressing*. Se envía con cada comando un identificador antepuesto al mismo indicando qué dispositivo lógico es el destinatario de la información. Se utiliza una única dirección primaria y el nombre del dispositivo queda a elección del programador.
-

## 5 Comandos de medida normalizados. SCPI, ADIF

La norma ANSI/IEEE 488.2 de 1987 (revisada en 1992) introdujo una estandarización de muchos aspectos que no estaban incluidos en la norma previa de 1975, como son: el formato de datos, el reporte de estados, el tratamiento de errores, la funcionalidad del controlador y algunos comandos básicos al que todos los instrumentos deben responder. Así, esta norma establece algunas especificaciones del *software* que no estaban incluidas en la norma original. Sin embargo, deja abierto el formato y tipo de comandos que hay que enviar a los instrumentos (ver la figura 3.6). La norma SCPI (*Standard Commands for Programmable Instruments*) aparece en 1991 para conseguir una estandarización de los comandos de control y el formato de los datos de los instrumentos. El objetivo es que, independientemente del fabricante, equipos que tienen la misma funcionalidad respondan de igual forma a un conjunto estándar de comandos.

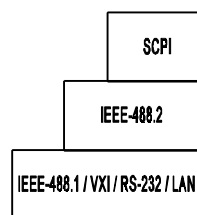


Fig. 5.1 Jerarquía de las normas para instrumentos controlables

La norma SCPI es el escalón más alto dentro de la jerarquía normativa para el control de sistemas de instrumentación. Tal como se puede ver en la figura 5.1, la norma SCPI se asienta sobre la IEEE-488.2 y esta, a su vez, se basa en la IEEE-488.1. A pesar de esta jerarquía los comandos y la estructura de datos basados en la norma SCPI pueden usarse, y se usan, en sistemas de instrumentación que no estén basados en IEEE-488, por ejemplo en sistemas basados en VXI, RS-232 o LAN.

El objetivo general de la norma SCPI es reducir los costes de desarrollo y mantenimiento de programas de control de sistemas de instrumentación para prueba automática. Este objetivo se consigue ya que el cumplimiento de la normativa:

- facilita el aprendizaje y uso de los comandos y los datos.
- facilita el desarrollo y mantenimiento de los programas.
- posibilita la sustitución de equipos con los mínimos cambios de software.

Para conseguir estos objetivos la norma establece la sintaxis y los formatos de los mensajes para que instrumentos con la misma funcionalidad o instrumentos del mismo tipo utilicen los mismos comandos. Por ejemplo, los comandos para medir una frecuencia utilizando frecuencímetros de distintos fabricantes serán los mismos. Además, la medida de la frecuencia con otro instrumento que

lo permita, por ejemplo un osciloscopio digital o un multímetro, también utilizarán los mismos comandos.

La norma establece tres tipos de compatibilidad entre instrumentos:

- 1.- Vertical: equipos de un mismo tipo tienen los mismos controles. Ejemplo: en dos multímetros distintos la selección de escala se realizará de la misma forma.
- 2.- Horizontal: equipos que realizan la misma medida, aunque sea de formas distintas, utilizarán los mismos comandos. Ejemplo: un osciloscopio que pueda medir períodos y un frecuencímetro-contador utilizarán los mismos comandos para medir el período de una señal.
- 3.- Funcional: dos equipos distintos que puedan realizar la misma función lo harán con los mismos comandos. Ejemplo: un analizador de espectros que pueda realizar barridos de frecuencia y un generador de señal serán funcionalmente compatibles si el barrido de frecuencia se programa con los mismos comandos.

Para conseguir un conjunto de comandos que puedan ser utilizados en cualquier instrumento, optimizando la compatibilidad, la norma define un modelo de instrumento universal. Este modelo es el que pasamos a ver de forma más detallada a continuación.

Para facilitar el aprendizaje de los nombres de los comandos, que provienen de abreviaciones de palabras inglesas, usaremos las denominaciones inglesas para las distintas partes de los instrumentos.

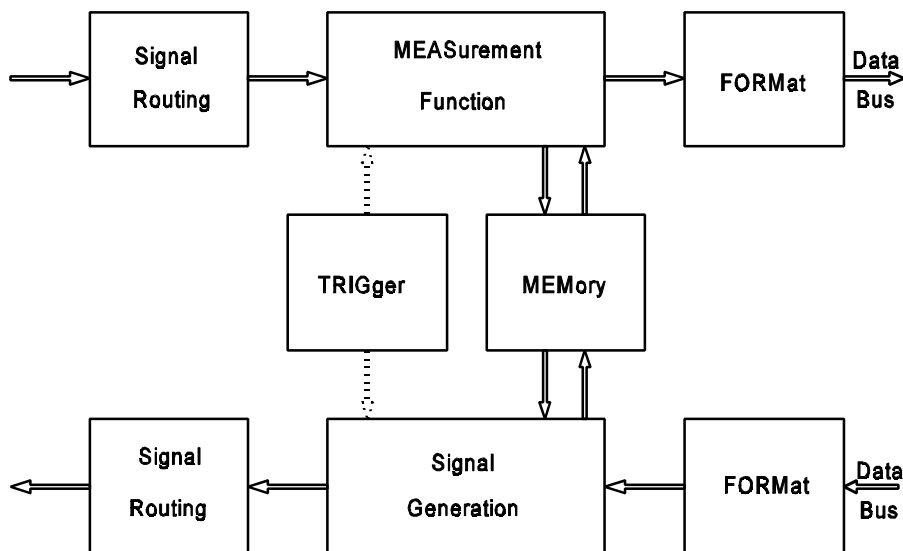


Fig. 5.2 Modelo de instrumento general definido en la norma SCPI

## 5.1 Modelo de instrumento según la norma SCPI

El objetivo de establecer un modelo general de instrumento es estandarizar un conjunto de bloques funcionales a los que se les asignará un conjunto de comandos para su programación. De esta forma cada instrumento concreto podrá ser descrito mediante un subconjunto de estos bloques

funcionales y su programación se realizará utilizando los comandos correspondientes a estos bloques. El modelo de instrumento es el representado en la figura 5.2. Este modelo describe el flujo de datos en un instrumento genérico; las líneas de trazo continuo representan flujo de datos y las líneas a trazos representan controles.

Cada instrumento específico se representa sólo por los bloques que lo constituyen. Por ejemplo, un multímetro digital puede tener sólo los bloques de función de medida, TRIGger y FORMat (figura 5.3). En cambio, para un generador de funciones sólo tendríamos: FORMat y Signal generation (figura 5.4)<sup>1</sup>.

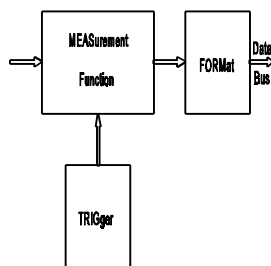


Fig. 5.3 Modelo para un multímetro digital

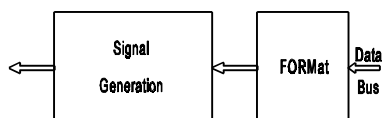


Fig. 5.4 Modelo para un generador de funciones

A continuación se describe cada uno de los bloques del modelo de instrumento definido.

#### - Signal ROUTing

Representa la posibilidad que tienen ciertos instrumentos para direccionar las señales de sus conectores de entrada a distintos circuitos internos. Es análogo al bloque de direccionamiento de señal visto en el apartado 5.2. Los comandos que controlan esta sección se denominan ROUTe.

#### - MEASurement Function

Este bloque es el que realiza la medida, entendida en su sentido más amplio como una transformación de una señal en un código que luego podrá ser procesado. Este bloque se subdivide a su vez en los siguientes (figura 5.5): INPut, SENSE y CALCulate.

La subdivisión del bloque de medida en estas tres partes no se ha incluido en el modelo de instrumento (figura 5.2) porque hay instrumentos que no serían 'horizontalmente compatibles' a este nivel pero sí lo son a nivel del bloque funcional de medida. Por ejemplo, la medida del valor eficaz de una senoide con un voltímetro y con un osciloscopio digital podrían ser horizontalmente compatibles utilizando instrucciones al nivel del bloque MEASurement (utilizarían los mismos comandos), pero los comandos para programar la medida a un nivel

<sup>1</sup> La parte de las palabras que está en mayúsculas coincide con el nombre de los comandos estándares abreviados para cada bloque.

más bajo serían distintos. En el voltímetro la medida se realiza con un comando del bloque SENSE, mientras que en el osciloscopio habría que calcular (utilizando comandos del bloque CALCulate) el valor eficaz a partir de las muestras adquiridas.

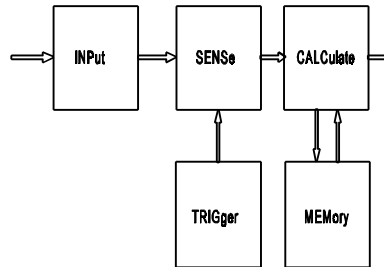


Fig. 5.5 Bloque de medida expandido

**- INPut**

Representa la etapa de adaptación de la señal de entrada, por ejemplo el acoplo AC, DC, GND de osciloscopios.

**- SENSE**

Es propiamente el bloque de medida, que pasa de una señal eléctrica a un código que luego puede ser procesado digitalmente.

**- CALCulate**

Su función es pasar los datos adquiridos por el bloque de medida a valores que sean más apropiados para una aplicación concreta. Por ejemplo, calcular períodos, frecuencias, tiempos de subida, etc. en un osciloscopio digital.

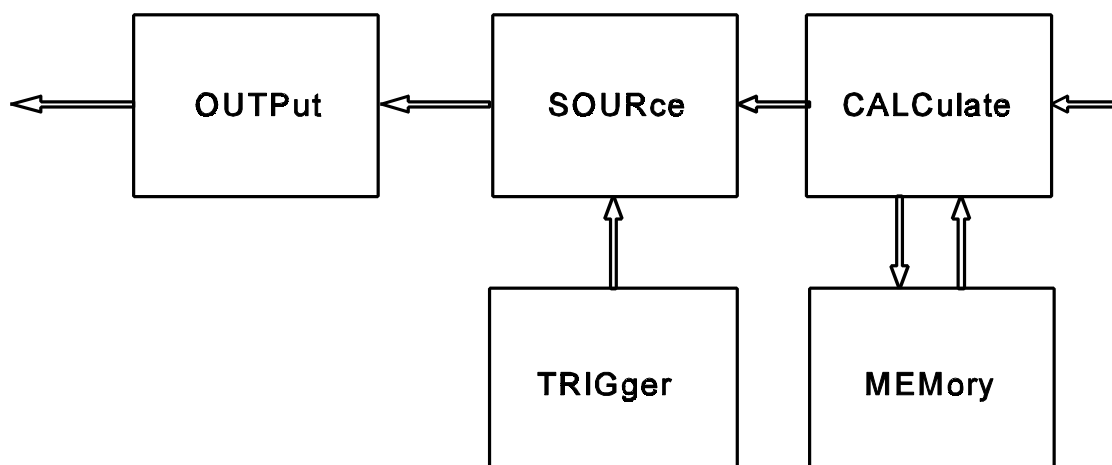


Fig. 5.6 Bloque de generación expandido

### - Signal Generation

Este bloque también está subdividido al igual que el de medida (figura 5.6). Su función es la de pasar datos a una señal eléctrica. Para ello se pueden distinguir los siguientes sub-bloques:

#### - CALCulate

Su función es modificar el flujo de datos recibido para generar la señal deseada a la salida.

#### - SOURce

La función de este bloque es determinar las características de la señal generada a partir del flujo de datos suministrado.

#### - OUTput

Es la parte que determina cómo se aplica la señal generada. Incluye las funciones de atenuación, filtrado, suma de tensiones continuas, etc.

#### - TRIGger

Su función es proveer al instrumento de medios para sincronizarse con eventos de las señales, tanto internas como externas.

#### - MEMory

Es el almacén de datos interno al instrumento. Según sea el caso pueden realizarse lecturas de datos, escrituras o guardarse parámetros de calibración internos.

#### - FORMat

Es el encargado de transformar el formato de los datos digitales internos al instrumento a otros que sean transferibles a través del bus de control.

## 5.2 Sintaxis y estilo

Los comandos en la norma SCPI se agrupan jerárquicamente en forma de árbol. En la raíz del árbol se encuentran los comandos que hacen referencia a los bloques que aparecen en el modelo de instrumento visto en la sección anterior. Cada uno de estos comandos se divide en un conjunto de ramas identificadas por palabras clave que a su vez identifican a funciones subordinadas al bloque raíz y así sucesivamente.

En la figura 5.7 se representa el árbol del bloque de INPut. La notación que se sigue es la siguiente:

- : indican el paso a un nivel jerárquico inferior. Para clarificar la notación también se ha utilizado indentación de los niveles inferiores.
  - [ ] palabras clave opcionales
  - <> encierran el tipo de parámetro
  - | separa parámetros opcionales (solo se puede poner uno de ellos)
  - ; separa comandos que están en la misma línea (no cambia el nivel del último comando)
  - , se usa para separar distintos parámetros dentro de un mismo nivel
  - ? indica que es un comando de consulta y que se espera una respuesta del equipo al que se envía. Es muy importante leer el dato solicitado; de no ser así al enviar otro comando se crea una situación de error en el instrumento.
-

KEYWORD	PARAMETER FORM	NOTES
INPut		
:ATTenuation	<numeric_value>	<tipo de variable>
:AUTO	<Boolean> ONCE	separa parámetros opcionales
:BIAS		
[:STATe]	<Boolean>	opcional
:VOLTage		
[:DC]	<numeric_value>	opcional
:AC	<numeric_value>	
:TYPE	CURRent VOLTage	
.		otras opciones
.		.
.		.
:COUPling	AC DC GROund	
:GAIN	<numeric_value>	
:AUTO	<Boolean> ONCE	
.		otras opciones
.		.
.		.

Fig. 5.7 Estructura jerárquica de comandos para el bloque de INPut de un instrumento.

Comandos SCPI correctos para este bloque son:

INPut:BIAS:STATe OFF

INP:COUPling ?

INP:ATT 20

INPut:ATTenuation:AUTO; INPut:BIAS:STATe ON; INPut:BIAS:VOLTage:DC 0.1 V

INP:ATT:AUTO; INP:BIAS:STAT ON; VOLT:DC 0.1 V (esta línea realiza la misma programación del instrumento que la anterior)

La sintaxis para los valores numéricos, las unidades y sus prefijos es la definida en los apartados 7.2 y 7.3 de la norma IEEE-488.2. Los parámetros numéricos pueden ser en cualquier tipo de notación: entera, decimal o científica. Entre los sufijos numéricos aceptados están: MA ( $10^6$ ), k o K ( $10^3$ ), m o M ( $10^{-3}$ ), u o U (por  $\mu$ :  $10^{-6}$ ). También pueden enviarse las palabras MAXimum y DEFault.

Los parámetros booleanos pueden escribirse como ON y OFF o como 0 y 1. Los instrumentos siempre responderán con 0 o 1.

Existen dos grupos de comandos definidos en la norma: comunes y específicos de instrumento. Los comandos comunes son los mismos comandos que se definen como obligatorios en la norma IEEE 488.2. Éstos sirven para funciones tales como: reinicialización, autocomprobación y operaciones de estado. Los comandos específicos de instrumento son los propios que define la norma SCPI. La sintaxis de cada grupo puede verse en la figura 5.8.

### 5.3 Comandos

Los comandos comunes definidos en la norma IEEE-488.2 que deben estar implementados para cumplir la norma SCPI son:

- \*CLS Clear Status Command
- \*ESE Standard Event Status Enable Command

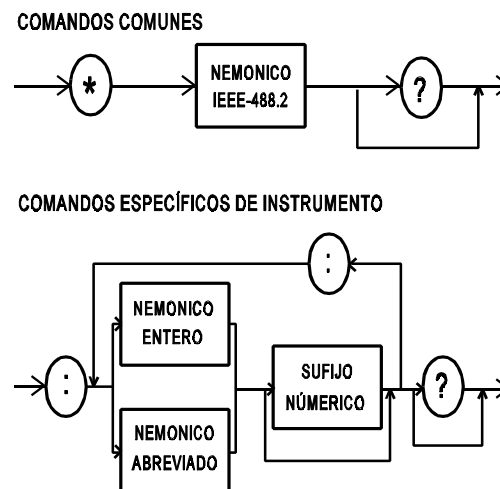


Fig. 5.8 Sintaxis para los comandos comunes y específicos.

- \*ESE? Standard Event Status Enable Query
- \*ESR? Standard Event Status Register Query
- \*IDN? Identification Query
- \*OPC Operation Complete Command
- \*OPC? Operation Complete Query
- \*RST Rest Command
- \*SRE Service Request Enable Command
- \*SRE? Service Request Enable Query
- \*STB? Read Status Byte Query
- \*TST? Self Test Query
- \*WAI Wait-to-Continue Command

Para una explicación de estos comandos véase el apartado 3.5.4.

Los comandos específicos definidos en la norma SCPI se dividen a su vez en dos grupos, los obligatorios y los opcionales. Los únicos bloques que son obligatorios son el de SYSTEM y el de STATus.

Los comandos específicos definidos en la norma SCPI son los siguientes:

- |               |              |          |              |
|---------------|--------------|----------|--------------|
| - MEASure     | - INPut      | - ROUTe  | - TRACelDATA |
| - CALCulate   | - INSTRument | - SENSE  | - TRIGger    |
| - CALibration | - MEMory     | - SOURce | - UNIT       |
| - DIAGnostic  | - MMEMory    | - STATus | - VXI        |
| - DISPlay     | - OUPut      | - SYSTem |              |
| - FORMat      | - PROGram    | - TEST   |              |

En la norma se estipulan todos los niveles inferiores posibles para cada uno de estos bloques. En un instrumento en particular sólo se utilizarán aquellos comandos que tengan una función definida. En la figura 5.9 pueden verse los comandos que son funcionales en un multímetro digital

para los bloques de medida y configuración. Se debe notar que si se envía un comando de medida, por ejemplo:

```
MEASure
:VOLTage:DC? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:VOLTage:DC:RATio? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:VOLTage:AC? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:CURRent:DC? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:CURRent:AC? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:RESistance? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:FRESistance? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:FREQuency? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:PERiod? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:CONTInuity?
:DIODE?
```

```
CONFigure
:VOLTage:DC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:VOLTage:DC:RATio {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:VOLTage:AC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:CURRent:DC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:CURRent:AC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:RESistance {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:FRESistance {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
:FREQuency {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}
```

Fig. 5.9 Ejemplo de comandos posibles en un multímetro digital para los bloques de medida y configuración

```
MEAS:VOLT:DC? 10, 0.003
```

el multímetro automáticamente realizará una medida y esperará a que se lea su registro de salida. En cambio, con la instrucción:

```
CONF:VOLT:DC 10, 0.003
```

solo se configurará el instrumento pero la medida no se realizará hasta que se envíe una instrucción adecuada, por ejemplo:

```
TRIG:SOUR EXT; READ?
```

en este caso el multímetro esperará a recibir un sincronismo de reloj externo para realizar la medida y ponerla en su registro de salida.

## 5.4 Formato de datos: ADIF

El objetivo de este formato de datos es el de disponer de un estándar para el intercambio de datos entre instrumentos o sistemas de forma automática. El formato ADIF está diseñado para que la información de cada bloque de datos sea autosuficiente. La estructura está diseñada para que sea muy flexible y extensible, y puede almacenar desde datos sencillos como un vector unidimensional a estructuras complejas multidimensionales.

El formato de datos está estructurado en bloques. De forma general, un conjunto de datos consiste en varios bloques de descripción y un bloque de datos. Los bloques de descripción contienen palabras claves que identifican sub-bloques en los que otras palabras clave seguidas de variables definen las características de la información contenida en el bloque de datos.

Los bloques definidos por la norma son 10; entre corchetes [] se incluyen los bloques que son opcionales:

ADIF: define el bloque general en el que se incluyen los demás, asignándole una etiqueta.

STD: define la versión de ADIF usada.

[REMark]: es un texto libre que contiene comentarios generales sobre los datos.

[IDENTify]: da nombre al bloque de datos, describe de dónde proviene y contiene identificadores como número de prueba, fecha y hora de captura, etc.

[ENCode]: especifica la codificación de todos los datos numéricos del bloque y, opcionalmente, su valor máximo y mínimo o su resolución.

DIMension: este bloque puede repetirse varias veces y especifica la estructura y el formato de cada bloque de datos. Los datos pueden ser implícitos, y en este caso se incluirán en el bloque DATA, o explícitos estando definidos en este bloque. Contiene otras características como son: escala, offset y unidades de medida.

[ORDer]: especifica el orden dentro del bloque de datos para cada una de las dimensiones implícitas que tenga. Los datos pueden estar ordenados por dimensiones (DIM) o de forma alternada con un dato de cada dimensión (TUPLE).

[TRACe]: agrupa lógicamente las variables en forma de *arrays*, superficies o conjuntos arbitrarios. Da nombres a las trazas y puede dar información sobre posibles simetrías de éstas.

[VIEW]: define relaciones entre los distintos grupos de trazas (TRACe), por ejemplo puede identificar a una dimensión como parte real y a otra como parte imaginaria de un número complejo.

DATA: contiene los datos propiamente dichos en el orden indicado por ORDER y en el número indicado por las dimensiones implícitas y su longitud. Normalmente incluye una comprobación de errores basada en una suma (CSUM)

El formato está estructurado en bloques; en la figura 5.10 puede verse un ejemplo. Cada nivel jerárquico está definido por un nombre de bloque seguido de un paréntesis que incluye a los bloques subordinados. Dentro de cada sub-bloque hay palabras clave seguidas de uno o más valores numéricos. Los valores numéricos están separados por comas. Esta estructura de datos es compatible con lo definido por la norma IEEE 488.2.

Todos los caracteres usados son ASCII de 7 bits (ANSI X3.4-1977). Los datos en el bloque DATA pueden ir codificados en otros códigos distintos de ASCII. La norma especifica 11 tipos distintos de codificación, desde números enteros en ASCII a bloques de datos en binario con distintas longitudes (8, 16, 32 o 64 bits) pasando por formatos especiales para cadenas de caracteres, tiempo y fecha.

El ejemplo de la figura 5.10 es bastante complejo; para mostrar el caso más simple podemos poner parte de la misma información de la siguiente forma:

---

```

ADIF = datos1 (
  STD (1.0)
  DIM = amplitud (
    TYPE EXP
    UNIT "V")
  DATA ( CURV ( VAL 233, -134, ... 389 ) ) )

ADIF = canal1 (
  STD (VERSion 1.0)
  REMARK (
    NOTE "array opcional de un máximo de 255 caracteres")
  IDENTify (
    NAME "Ejemplo de formato"
    DATE 1995-08-22
    TIME 13:15:10.55
    TEST (NUMBer "AD1", "3.1"))
  ENCode (
    NOTE "array opcional de caracteres"
    FORMat INT16
    HRANge 1000
    LRANge -1000)
  DIMension = amplitud (
    TYPE EXPLicit
    SCALe 3.2E-6
    OFFSet 1.0E-5
    UNITs "V")
  DIMension = tiempo (
    TYPE IMPLicit
    SCALe 1E-6
    SIZE 1024
    UNITs "S")
  ORDer (
    BY DIMension)
  TRACE = respuesta_impulsional (
    SYMMetry NONE
    INDEpendent (LABel tiempo)
    DEPEndent (LABel amplitud))
  DATA (
    CURVe (
      VALue 233, -134, ... 389
      CSUM 67)))

```

*Fig. 5.10 Ejemplo de formato de datos ADIF*

La estructura se ha simplificado mucho pero a costa de perder información (en este caso la escala temporal y otros datos identificativos), lo que está en contra de la filosofía básica de la norma, que es el contener los datos en una estructura que sea autosuficiente para reconstruir toda la información adquirida por los instrumentos de medida.

## **6 Instrumentación virtual. Programas de ayuda al diseño de sistemas de instrumentación**

### **6.1 Clases de instrumentos virtuales**

Un instrumento virtual es simplemente un conjunto de programas y equipos con una interfase gráfica que tiene la apariencia y el aspecto de un instrumento físico. El usuario puede manejar el instrumento a través del panel gráfico como si fuera un instrumento real.

Los instrumentos tradicionales son autocontenidos, tanto las capacidades de entrada salida como la interfase de usuario, botones, pulsadores, pantallas, etc., y otras características adicionales son fijas. Dentro de la caja del instrumento convertidores A/D, circuitos acondicionadores de señal, microprocesadores, memoria y un bus interno convierten señales del mundo real, las analizan, y las presentan como resultados al usuario. El fabricante define completamente la funcionalidad del instrumento, sin la posibilidad de ser cambiada por el usuario.

La tendencia actual en instrumentación es utilizar el ordenador como un elemento más. Los instrumentos virtuales se benefician de la arquitectura abierta de los estándares de ordenadores para ofrecer las capacidades de procesado, memoria y visualización; mientras que las tarjetas de adquisición, de bajo coste, y las tarjetas de interfase IEEE-488 y VXI conectadas en el bus del ordenador sirven de vehículo para las capacidades del instrumento virtual. La funcionalidad del instrumento la define el propio usuario y la potencia de procesado puede ser incluso mucho mayor que en los instrumentos convencionales.

El instrumento virtual simula cada una de las partes descritas anteriormente, apoyándose en elementos hardware accesibles por el ordenador (tarjetas de adquisición, instrumentos IEEE-488, VXI, RS-232).

Cuando se ejecuta un programa que funciona como instrumento virtual, el usuario ve en la pantalla de su ordenador un panel cuya función es idéntica a la de un instrumento físico, lo que facilita la visualización y el control del aparato. A partir de los datos reflejados en el panel frontal, el instrumento virtual debe actuar recogiendo o generando señales, como lo haría su homólogo físico.

El usuario, y no el fabricante, define la funcionalidad final del instrumento. El software es la clave en los instrumentos virtuales; ofrece al usuario las herramientas necesarias para construir instrumentos virtuales y expandir su funcionalidad ofreciendo una conectividad con las enormes posibilidades de los PC y las estaciones de trabajo, y otras aplicaciones. Esta flexibilidad que permite

---

el software puede, sin embargo, llevar un coste asociado mayor que el del instrumento tradicional. Si el usuario no dispone de las herramientas adecuadas de programación para el desarrollo de la aplicación, las horas invertidas en la realización de los programas encarecerán el valor real del instrumento final.

Las diferencias entre instrumentos tradicionales y virtuales se pueden resumir en la siguiente tabla:

Instrumentos tradicionales	➔	Instrumentos virtuales
Definido por el fabricante	➔	Definido por el usuario
Función específica, conectividad limitada	➔	Sistema orientado a la aplicación con conectividad a redes, periféricos y aplicaciones
El hardware es la clave	➔	El software es la clave
Caro	➔	Bajo coste, reutilizable
Cerrado, funcionalidad fija	➔	Abierto, funcionalidad flexible utilización de una tecnología familiar
Cambios lentos en la tecnología (5-10 años de ciclo de vida)	➔	Adaptación rápida a los cambios tecnológicos, (1-2 años de ciclo de vida)
Costes de desarrollo y mantenimiento grandes	➔	El software minimiza los costes de desarrollo y mantenimiento

Tal como se ha visto, un instrumento virtual está formado por tres elementos principales: adquisición de datos, análisis y presentación, tal como muestra la figura 6.1. Junto al gran crecimiento de la microinformática han ido apareciendo en el mercado diversas herramientas de programación para el desarrollo de sistemas de instrumentación virtual, que se centran en alguno o en todos los elementos del sistema. La elección del entorno de programación vendrá condicionada por la aplicación final del usuario.

El software que realiza el instrumento virtual y que se ejecuta sobre el controlador, en este caso el ordenador, ha evolucionado con el tiempo. Los primeros entornos de programación permitían el control de instrumentos o dispositivos externos al ordenador, un ejemplo es el Basic de la familia HP9000 (la mayor parte de los instrumentos de HP programables mediante el bus IEEE-488 todavía incluyen ejemplos de programación en Basic). Otros fabricantes de interfases para ordenador suministraban, primero, un conjunto de funciones (funciones de nivel 0 o de registro) que se dejaban residentes en memoria y a las que se accedía mediante interrupciones software. Posteriormente suministraban librerías de funciones (funciones de nivel 1) que se podían llamar desde lenguajes de alto nivel, ( C, Basic o Pascal) que ejecutaban las interrupciones software. Estas primeras herramientas facilitaban el control de la interfase de comunicaciones con instrumentos externos y eran independientes del instrumento a controlar. No obstante, éstas no incluían utilidades para el análisis de datos ni la presentación de los mismos. La creación de instrumentos virtuales, con

interfases gráficas de usuario, requerían un gran esfuerzo de programación y adolecían de flexibilidad de adaptación o modificaciones.

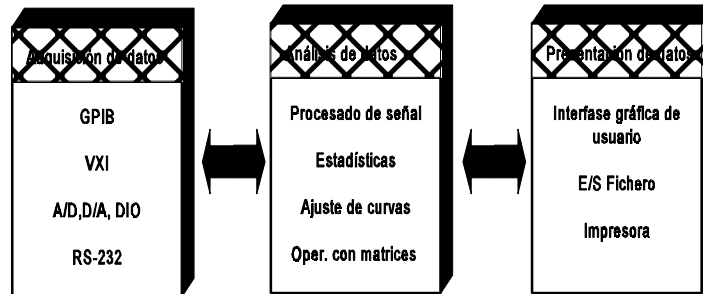


Fig. 6.1 Partes integrantes de un instrumento virtual

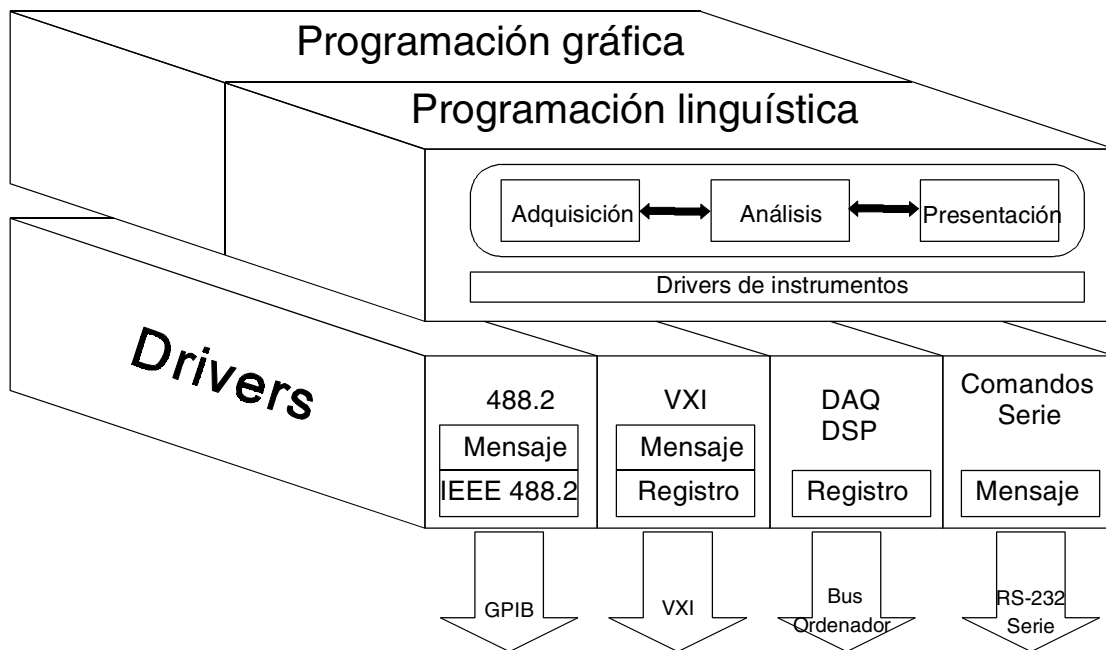


Fig. 6.2 Arquitectura del entorno de programación para instrumentación virtual

El siguiente paso fue crear librerías de funciones específicas para un determinado instrumento o conjunto de instrumentos de un fabricante. Evitan la necesidad de aprender los comandos o las instrucciones del instrumento por parte del programador. Cada una de estas funciones (de nivel 2) suele hacer uso de una serie más o menos compleja de funciones de nivel 1 y 0. A este conjunto de funciones se le denomina *driver de instrumento* y pueden encontrarse en forma de librerías enlazables a un lenguaje de programación (C, Basic, Pascal), o bien en formato DLL (*Dyamic Link Library*) para

MS-Windows de manera que se pueden llamar desde cualquier entorno de programación (figura 6.2).

Al igual que las funciones de nivel 1 y 0, las de nivel 2 siguen abarcando únicamente la adquisición de datos, bien sea a través de instrumentos convencionales provistos de interfase GPIB, RS-232 o VXI, o con tarjetas de adquisición de datos incorporadas al bus del propio ordenador.

No obstante, los mismos fabricantes de software ofrecen entornos programación propios para el desarrollo de aplicaciones que permiten acceder a estas librerías de una forma más fácil. Estos entornos ya incluyen herramientas para el análisis y presentación de datos más o menos elaboradas.

## 6.2 Lenguajes de control para sistemas de instrumentación

Los entornos de programación para el control de sistemas de instrumentación virtual pueden clasificarse en diversas categorías o clases según el grado de flexibilidad y facilidad de uso.

El primer grupo lo comprenden aquellos entornos que han sido desarrollados para el control de un instrumento específico o tarjeta de adquisición de datos. Permiten, mediante una interfase de menús desplegables, configurar y programar el dispositivo para la adquisición de una señal y su visualización. Un ejemplo de estos entornos es el software **DAQ-Ware** de National Instruments que ofrece con sus tarjetas de adquisición de datos. Permite configurar el número de canales, la ganancia y la frecuencia de muestreo de sus tarjetas y visualizar en pantalla las señales adquiridas así como salvarlas a fichero.

En el segundo grupo están los entornos de programación lingüísticos. El acceso a las funciones de nivel 2, para la adquisición de datos, se hace a través de un terminado lenguaje de programación, este puede ser estándar (C, BASIC, etc.) o propio del entorno. Además se dispone de librerías de funciones para el análisis y la presentación de datos. Algunos de estos entornos incorporan una interfase gráfica de menús desplegables que permite el acceso a estas funciones para facilitar la generación del programa. Otro parámetro importante es el modo de ejecución de la aplicación final. Algunos entornos permiten únicamente la ejecución de la aplicación dentro del mismo. Tienen como ventaja las facilidades de depuración que incluyen y la simplificación de la interfase con el hardware del controlador. Sin embargo, la velocidad de ejecución es mucho menor y requiere el pago, en la mayoría de los casos, de una licencia *run-time* para su distribución o venta.

Algunos de los entornos más utilizados son:

**LabWindows y LabWindows/CVI (National Instruments):** Son entornos de programación propios en C y Basic, con menús de ayuda para la generación de código de forma interactiva, para aplicaciones MS-DOS y MS-Windows, respectivamente, de test, medida y control. Incluyen compiladores de ANSI-C, *linker*, *debugger* y librerías para adquisición de datos, análisis y presentación. Se pueden incorporar ficheros fuente externos, módulos objeto y DLL (en la versión MS-Windows). Ambas versiones pueden generar aplicaciones ejecutables fuera del entorno aunque

---

en la versión MS-DOS es necesario un compilador C estándar externo al entorno, no suministrado con el paquete. Una característica importante de estos paquetes de programación es la inclusión de librerías de instrumentos, *drivers*, dentro del mismo entorno y que el fabricante suministra de forma gratuita para una gran diversidad de fabricantes de instrumentos controlables: IEEE-488, VXI, RS-232 y CAMAC.

**VisuaLab (IOtech):** Es una extensión del entorno Visual Basic para el desarrollo de aplicaciones de adquisición de datos.

**Asyst (Keithley-MetraByte):** Entorno de programación en un lenguaje propio para MS-DOS. Aplicaciones de adquisición de datos y análisis. Incluye librerías de análisis y *drivers* para tarjetas de adquisición de varios fabricantes. Los programas corren únicamente dentro del entorno.

**HPITG II (Hewlett-Packard):** Es un entorno de programación con una interfase gráfica que permite mediante menús generar secuencias sencillas de medida y análisis. Estas pueden grabarse en un fichero y convertirse posteriormente a una secuencia de llamadas de funciones en C para incorporar dentro de la aplicación final. Incluye únicamente librerías para instrumentos de Hewlett-Packard, aunque dispone de un instrumento genérico que permite programar cualquier instrumento con interfase IEEE-488.

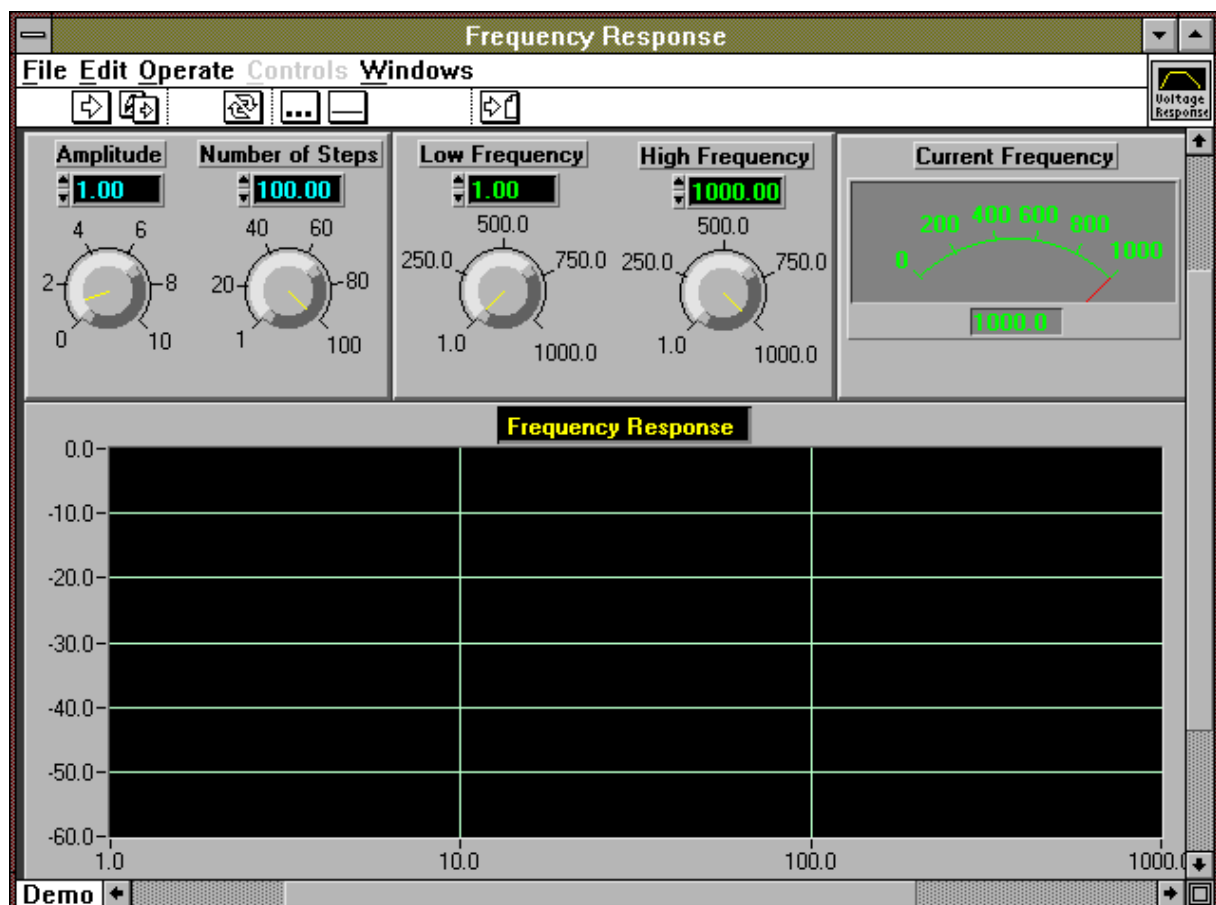


Fig. 6.4 Panel frontal de un instrumento virtual creado con labVIEW

Los entornos de programación gráficos forman el tercer grupo. Su aparición en el mercado es más reciente. El desarrollo de aplicaciones es totalmente diferente. Permiten crear al usuario soluciones completas uniendo iconos de una forma totalmente gráfica y según una estructura jerárquica; véase la figura 6.5. Los bloques son módulos software preprogramados que aparecen como iconos en la pantalla. Algunos iconos son estándares para cualquier aplicación, pero otros corresponden a un hardware específico del sistema de medida. Pulsando sobre cualquier icono se abre una ventana de diálogo que contiene un listado de propiedades para su funcionamiento. Una de las características más importantes de la programación gráfica es que se dispone de iconos para crear interfases de usuario muy similares a las de cualquier instrumento convencional. Al igual que los lenguajes de programación clásicos se dispone de múltiples tipos de datos y estructuras de programación (bucles, condiciones, E/S, etc.) incluyendo algunos entornos un compilador gráfico para aumentar la velocidad de ejecución.

El ciclo de aprendizaje y programación se reduce drásticamente al ser entornos que imitan una forma de programación muy parecida a un diagrama de flujo o bloques. Esta simplificación en la forma de programación lleva asociada algunas limitaciones. La velocidad final de la aplicación será mucho menor que su versión en lenguaje C y la utilización de muchos elementos gráficos e iconos requiere grandes cantidades de memoria y potencia de cálculo.

Existen en el mercado diversos paquetes de programación gráfica; no todos son iguales y deben hacerse algunas distinciones. Una diferencia básica es que algunos, como LabVIEW, son entornos que funcionan por sí solos y otros, como el Visual DAS de Keithley, son adaptaciones de entornos de programación de uso común como el Visual Basic. Otra diferencia importante a considerar a la hora de la adquisición del paquete de programación son los *drivers* para dispositivos, instrumentos, tarjetas de adquisición, etc., que incluyen. Algunos paquetes únicamente cubren los del propio fabricante, por lo que obligan prácticamente a utilizar su hardware.

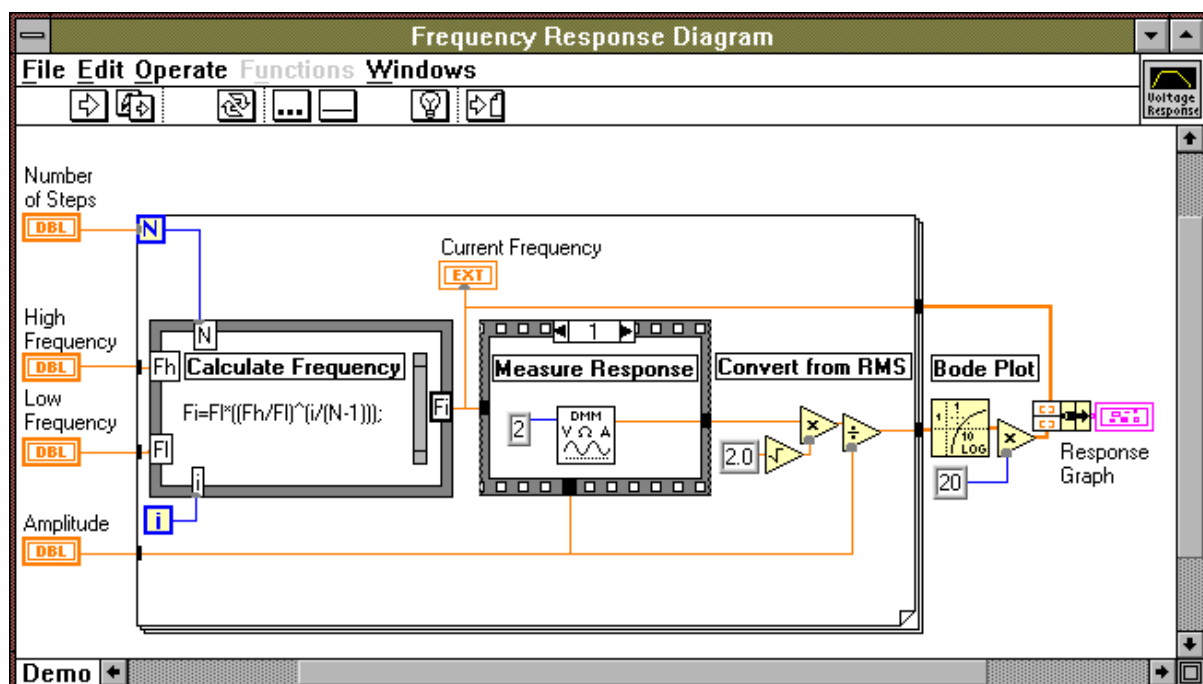


Fig. 6.5 Ejemplo de un programa gráfico para un instrumento virtual (LabVIEW)

Uno de los primeros y más conocido de los entornos de programación gráfica es el **LabView (National Instruments)**, que apareció en el año 1986. Las primeras versiones funcionaban sólo sobre Macintosh, que eran entonces los únicos que disponían de memoria suficiente, 1 Mbyte, y de un sistema operativo avanzado. Actualmente está disponible en versiones Macintosh, PC MS-Windows y para estaciones de trabajo Sun SPARC.

El usuario dispone de una completa gama de librerías de iconos para la manipulación de datos, control de flujo, interfase de usuario (botones, gráficos, menús, etc.), tarjetas de adquisición de datos del propio fabricante y *drivers* de la mayoría de instrumentos controlables (IEEE-488, VXI, RS-232, CAMAC) disponibles en el mercado. También pueden incluirse rutinas en lenguaje C como iconos y llamadas a funciones de librerías externas, por ejemplo DLL de MS-Windows.

El editor gráfico incluye un compilador para aumentar la velocidad de la aplicación. La aplicación final corre dentro del entorno y es necesario el pago de una licencia *run-time* para aplicaciones comerciales.

**VEE (Hewlett-Packard):** Es un entorno de programación gráfica propio que funciona sobre diferentes plataformas: Macintosh, PC (MS-Windows, Windows NT), y estaciones de trabajo HP y SUN.

Dispone de librerías para la manipulación de datos, control de flujo, interfases de usuario y *drivers* de instrumento controlables de HP (IEEE-488, VXI, RS-232). No dispone, sin embargo, de librerías de *drivers* para el control de tarjetas de adquisición. También permite el uso de librerías externas. El editor gráfico no incluye compilador y existe una versión *run-time* para la aplicación final.

**DT VEE (Data Translation):** Es un entorno propio, disponible únicamente en la versión PC (MS-Windows). Está orientado a la programación de sus tarjetas de adquisición y presentación de datos. No dispone de *drivers* para instrumentos controlables.

---