UTILIZACION SIMPLIFICADA DEL BUS IEEE 488 PARA CONTROL DE INSTRUMENTACION

José Antonio Rodríguez, Juan José Balza

El presente artículo tiene como objetivo tratar algunos de los problemas que aparecen en el montaje y control de un sistema de adquisición de datos, basado en el bus IEEE 488, y que pueden resolverse a través de la confección de un software adecuado.

José Antonio Rodríguez Mondejar y Juan José Balza Arrabal pertenecen al Departamento de Electrotécnia y Sistemas de la ETS Ingenieros Industriales del ICAI

SUMARIO

1. INTRODUCCION

El bus IEEE 488 o GPIB nació como un bus para control de instrumentación. Fue diseñado por la compañía *Hewlett-Packard* (bus HP-IB). Su amplia aceptación entre los fabricantes de instrumentación dio lugar a que fuera asumido por la IEEE, en 1975, originando la norma IEEE 488, revisada posteriormente en 1978; otro tanto hizo la IEC, recogiéndolo en la norma IEC625-1. Aunque ambas normas utilizan un conector distinto, los buses son totalmente compatibles entre sí.

El IEEE 488 es una interfase que permite la interconexión simultánea de instrumentos sobre una línea común de comunicaciones (bus), incluyendo un ordenador/controlador; éste dirige las operaciones de los instrumentos y recoge los resultados.

Las características más importantes de este bus son:

- La transmisión entre aparatos interconectados es digital. Se realiza en formato byte serie y bit en paralelo.
- El bus es bidireccional.
- El número máximo de aparatos conectados al bus es de 15.
- La transmisión es asíncrona. La velocidad de transmisión se adapta automáticamente a la del instrumento con menor velocidad de procesamiento.
- La máxima velocidad de transferencia es 1 Mb/s.

La norma define, dentro del sistema interfase, conforme a especificaciones funcionales, eléctricas y mecánicas, los siguientes niveles:

1. Bus físico que conecta los instrumentos, incluyendo conectores, líneas de señal y niveles de tensión.
2. Funciones de posible uso por las interfases de los instrumentos y del ordenador, es decir, las capacidades y niveles de capacidad de las interfases.
3. Mensajes de control y protocolos de transmisión.

La norma no define los mensajes específicos de los dispositivos conectados al bus.

Entre las ventajas más sobresalientes de este bus frente a otros buses, como los basados en tarjetas de adquisición de datos, destacan:

- Utilización de instrumentos especialmente diseñados para realizar un tipo de medida, sin necesidad de sobrecargar el controlador del sistema.
- Adaptabilidad futura. Los instrumentos que hay conectados al bus pueden ser sustituidos por otros más modernos conforme aparezcan en el mercado.
- Configuración dinámica. Se puede aumentar o disminuir el número de aparatos unidos al bus, así como la utilización de estos aparatos en otras medidas.

Tiene el inconveniente de un control más indirecto de lo que se quiere medir, sobre todo, si se quiere hacer una modificación *on-line*. Otro problema es la distancia máxima entre aparatos (entre 2 y 4 metros).

2. ESTRUCTURA DEL SISTEMA

Un sistema de adquisición de datos basado en este bus consta de tres partes:

- Ordenador/Controlador. Normalmente es un ordena-

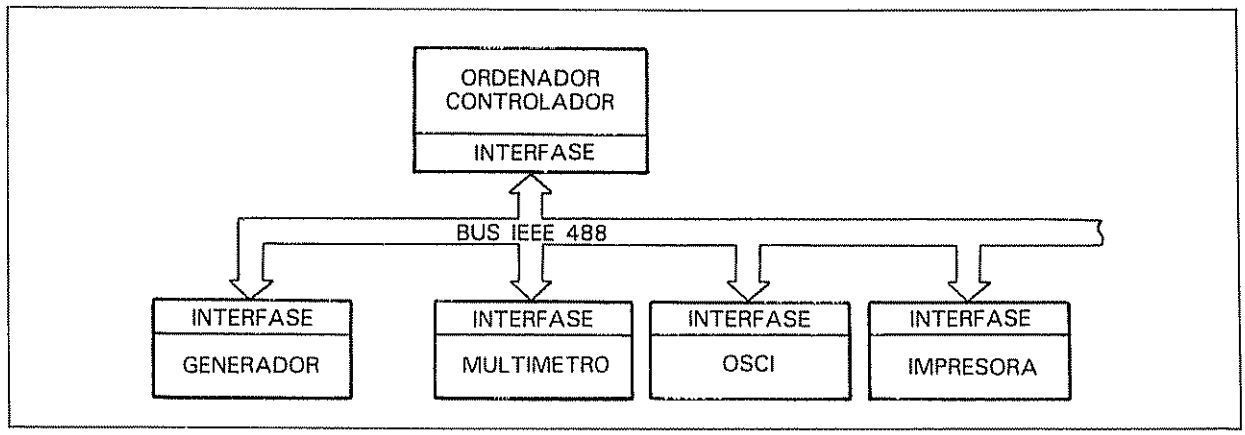

Fig 1 – Disposición típica de un sistema de adquisición de datos basado en bus IEEE 488

dor tipo PC, al cual se le ha instalado una tarjeta controladora de bus IEEE 488
- Instrumentación diversa, con sus respectivas interfases IEEE 488
- Cable físico (bus) de interconexión entre interfases formado por 24 líneas, distribuidas en 8 líneas de datos, 3 líneas de control de transferencia de datos, 5 líneas de manejo general del bus y 8 líneas de retorno

Funcionalmente, los aparatos quedan divididos dentro del sistema como:

- Controlador del bus (ordenador)
- *Talkers*: todo aquel dispositivo que es capaz de enviar información al bus.
- *Listeners*: aquellos dispositivos que pueden recibir información desde el bus

La figura 1 muestra una disposición típica de este sistema

3. CONFIGURACION DEL SISTEMA

A la hora de configurar y montar el sistema de adquisición de datos se siguen los siguientes pasos:

1 Definición del sistema a controlar: tipos de datos que se van a adquirir, rapidez de adquisición, simultaneidad, forma de procesamiento, posibles ampliaciones, etc
2 Elección de los instrumentos adecuados
3 Elección del ordenador con la suficiente capacidad de procesamiento y almacenamiento, y con tarjeta interfase GPIB instalada.
4 Preparación del software necesario para gestión del bus, adquisición de datos y su posterior procesamiento.

Los fabricantes de tarjetas IEEE 488 suministran con ellas un software para su manejo. Estos paquetes adolecen de los siguientes problemas: no cubren los diferentes tratamientos que demandan los instrumentos que actualmente hay en el mercado; están escritos en un lenguaje difícil de manejar como ensamblador o lento como el Basic; tienen un difícil interfase con el resto del programa destinado al tratamiento de los datos adquiridos: no están pensados para funcionar de forma concurrente.

Una posible solución es la compra de software que lleve implementada tanto la parte de adquisición como la de procesamiento. Estos paquetes son caros, lentos, y en general son paquetes cerrados, de difícil ampliación. La otra solución es diseñar el software que necesita el sistema, de tal forma que sea fácil su ampliación y la incorporación de nuevo instrumental al bus. El inconveniente de esta solución está en la necesidad de un personal muy cualificado. Se puede obviar este problema mediante la utilización de un lenguaje de alto nivel que permita trabajar de una forma fácil y estructurada tanto a alto como a bajo nivel. Dentro de este grupo de lenguajes destaca el Modula-2.

3.1. Características del MODULA-2

Este lenguaje fue diseñado por el Profesor N. Wirth como una evolución del Pascal. Frente a este último, incorpora software moderno pensado para ingeniería, como por ejemplo concurrencia, tratamiento de interrupciones y sus prioridades, manejo a bajo nivel, manejo completo de la memoria en forma dinámica, etc. Es un lenguaje totalmente estructurado, prueba de ello es la no existencia de la sentencia GO TO. La aportación más importante es la introducción del concepto de módulo. Existen dos tipos de módulos:

- *Módulo Programa*, que contiene el programa principal.
- *Módulos Librería*, con los procedimientos estándar que proporciona el fabricante del compilador, mas los procedimientos diseñados por el usuario. A su vez cada Módulo Librería está dividido en dos partes:

* *Modulo Definición* para definir la interface entre modulos, es decir, qué procedimientos, variables, tipos de variable tiene ese módulo que puedan ser utilizados por otros módulos
* *Módulo Implementación*, donde se recogen las sentencias necesarias para realizar las funciones definidas en el módulo definición

Esta estructura permite repartir de una forma adecuada la confección de un paquete de software entre un grupo de programadores. Basta con definir qué módulos hay que preparar y cuál es la interfase entre ellos. También permite variar la implementación de los módulos sin necesidad de compilar todo el paquete. Sólo hay que compilar la implementación del módulo modificado y encadenar el Módulo Programa (esta última operación es muy rápida). Cuando se compila, si un módulo necesita de otros módulos, basta con la parte de definición de estos.

Un Módulo Librería es inicializado previamente antes de ejecutarse cualquier procedimiento de dicho módulo. Antes de ejecutarse el Módulo Programa, que es el único que tiene cuerpo ejecutable, se inicializan los Módulos Librería.

El lenguaje permite la utilización de librerías en ensamblador, con lo cual, podemos utilizar las rutinas en

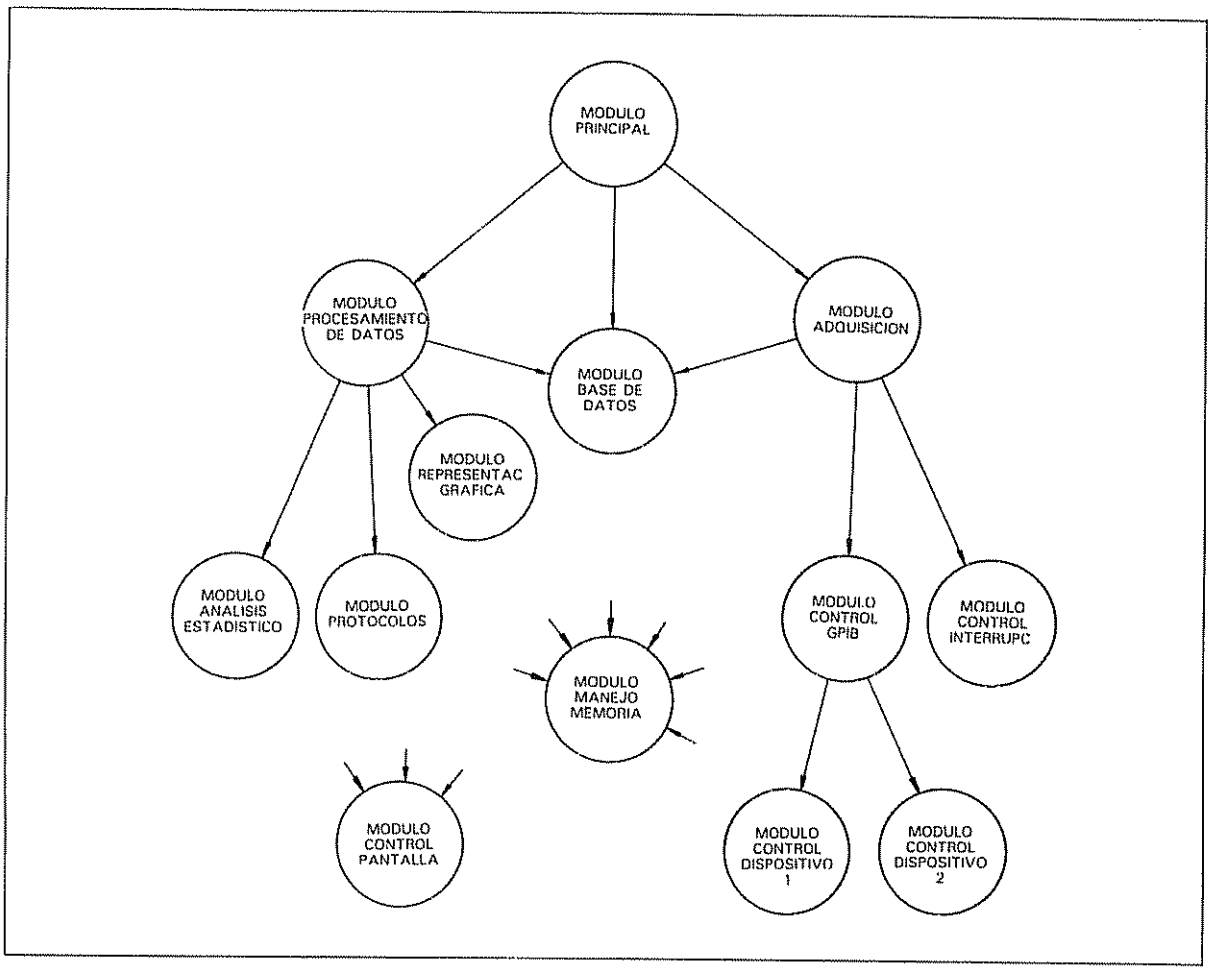

Fig. 2 – Estructura de programación modular para adquisición y tratamiento de datos en laboratorio

lenguaje máquina, que suministran los fabricantes de tarjetas GPIB

Otra ventaja del Modula-2 es la utilización del *Type Procedure*. El nombre de un procedimiento puede ser asignado a una variable, de tal forma que cada vez que aparezca dicha variable en el programa, se ejecuta dicho procedimiento. De igual forma se pueden construir matrices de procedimiento. Como es una variable, puede variar a lo largo del programa el procedimiento asignado. El único inconveniente está en que si no se programa con cuidado y con comentarios el programa, puede resultar indescifrable

La figura 2 muestra una posible estructura modular para el caso de desarrollo de un paquete de software de adquisición y tratamiento de datos, generados por la instrumentación de un laboratorio

El Modula-2 adolece de ser un lenguaje muy joven y poco conocido

3.2. Tarjetas de interfase IEEE-488

Esta tarjetas llevan incorporadas en el hardware todas las funciones definidas por la norma:

- *Talker (T)* permite a un dispositivo la capacidad de enviar datos a través del bus
- *Listener (L)*: con ella un dispositivo puede recibir datos a través del bus. Estos datos son almacenados en un buffer de la tarjeta (uno o varios registros)
- *Aceptor Handshale y Source Handshake*. SH y AH son las funciones encargadas de enviar los mensajes necesarios a través de las líneas de control de transferencia de datos (NRFD, NDAC y DAV) para poder realizar la transmisión de un byte de información entre un emisor *(talker)* y un receptor *(listener)*.
- *Service Request (SR)*: permite a un dispositivo la capacidad de pedir un servicio de atención al controlador.
- *Parallel Poll (PP)*. la función PP confiere a un dispositivo la capacidad de responder a un chequeo en paralelo por parte del controlador. A través de este chequeo, el controlador determina qué instrumento le ha pedido un servicio de atención (SR)
- *Device Clear (DC)*. paso a un estado definido de la tarjeta
- *Device Trigger (DT)*: capacidad de reconocer un orden de disparo a través del bus
- *Controller (C)*: la función C se encarga de la inicialización, dirección y gestión del bus.

La figura 3 detalla el esquema típico de esta tarjeta

Para el bus del PC, la tarjeta es un número limitado de registros consecutivo, que tienen una dirección dentro del bus y que pueden ser leídos o modificados.

Estas tarjetas pueden estar conectadas al bus del PC de dos formas:

- *I/O Mapped*. La tarjeta está conectada al mapa de entrada/salida del procesador (8088/8086/80286) Normalmente se manejan en este modo. El Modula-2 incorpora procedimientos que simulan las instrucciones en lenguaje máquina *in/out*.
- *Memory Mapped* Los registros de la tarjeta ocupan una zona de memoria. Con ello se consigue mayor velocidad, ya que una instrucción *Read/Write* es más

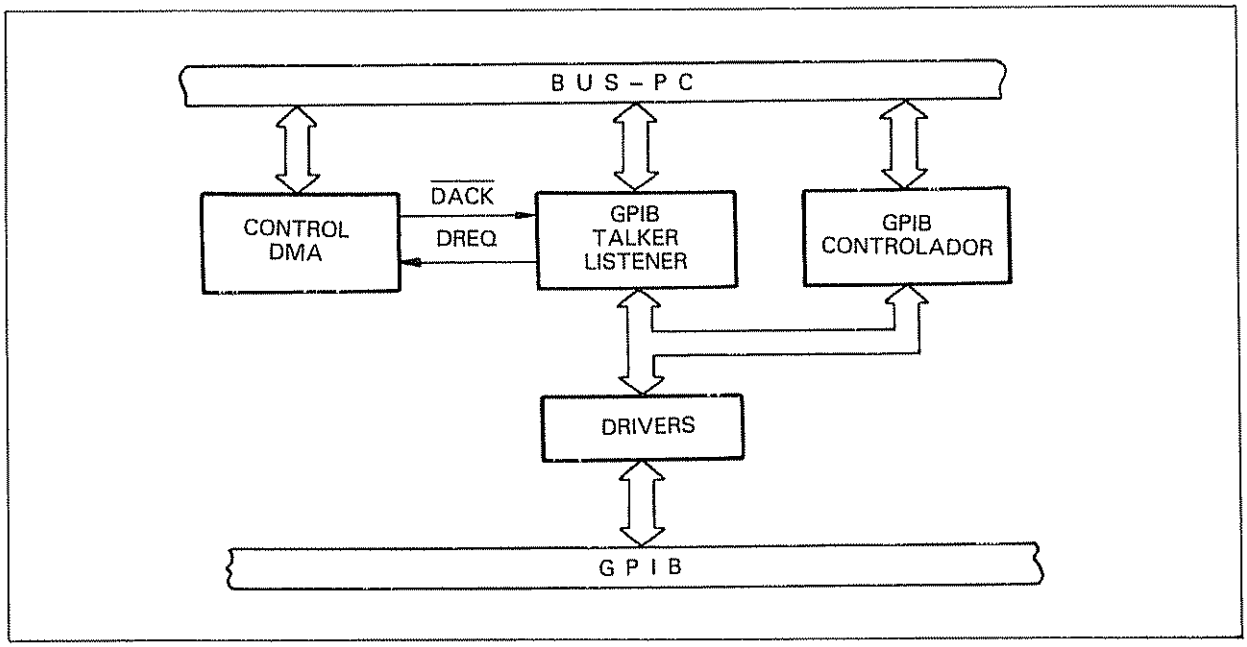

Fig 3 – Estructura hardware típica de una tarjeta de interfase IEEE 488

rápida que *in/out*. Tiene el inconveniente de la necesidad de acotar esa zona de memoria para que no sea utilizada indebidamente. El Modula permite crear variables con posiciones fijas en la memoria; de esta forma no sólo se acota esa zona, sino que a la vez se puede manejar a través de esa variable.

Para manejar la tarjeta se utilizan dos técnicas:

a) Mediante chequeo repetido de sus registros. Se escribe en el registro correspondiente el código de la operación que se quiere realizar; a continuación se realiza un chequeo repetido de la tarjeta para comprobar que se ha realizado la operación pedida, por ejemplo la lectura de un dato que hay en el bus (este dato se almacena en un registro de la tarjeta para que después sea leído por el procesador)
b) Bajo interrupción. No se necesita un chequeo continuo de la tarjeta para ver si se ha realizado una operación cuando se efectúa ésta, la tarjeta provoca una interrupción que es recogida por la correspondiente línea de interrupción del bus. Esto desencadena la ejecución de un programa conforme al tipo de interrupción. Este sistema ahorra tiempo de CPU pero es más difícil de programar.

4. TECNICAS DE PROGRAMACION

4.1. Secuencial.

Es la forma más usada. Consta de los siguientes pasos:

1. Inicialización de la tarjeta interfase del ordenador.
2. Direccionamiento del instrumento que va a realizar una determinada operación como receptor *(listener)*.
3. Direccionamiento de la tarjeta GPIB del ordenador como emisor *(talker)*.
4. Envío desde el ordenador, a través del bus GPIB del código de las operaciones que debe realizar el instrumento.
5. Desconfiguración del instrumento como receptor y el ordenador como emisor.
6. Direccionamiento del ordenador como receptor.
7. Direccionamiento del instrumento como emisor. A partir de este momento, el instrumento envía los datos al ordenador según el correspondiente *handshake* normalizado.
8. Recogida de datos por parte del ordenador.
9. Desconfiguración del instrumento como emisor y del ordenador como receptor.
10. Procesamiento de los datos.
11. Volver al paso 2 si se necesita otra medida con diferente configuración del instrumento o al paso 7, si no necesita ser cambiada dicha configuración.

Cada paso necesita un chequeo continuo por parte del ordenador para comprobar que la operación ha sido realizada.

4.2. Simulación de multiproceso.

Consiste en aprovechar los tiempos de chequeo para procesamiento de datos ya adquiridos.

El programa debe ser capaz de recoger simultáneamente los datos que se le envían a través de uno o varios buses GPIB, de otras tarjetas como un *port* serie y la información de las teclas pulsadas por el operador; además, debe estar dando información de todo lo que está ocurriendo a través de la pantalla, la impresora, grabación en disco, más una posible interferencia en el proceso que se está midiendo.

Una solución sencilla a este problema es, partiendo de la existencia de módulos que necesitan entrada por teclado, la siguiente:

Hay un programa principal con un único bucle, que comprueba si se ha pulsado una tecla. Si es cierto se pasa el control al módulo que necesita dicha tecla. Si no se ha pulsado, se da el control a módulos que no necesitan teclas, encargados de las labores de procesamiento de datos, representación, impresión, etc. Para evitar la impresión de un tiempo de respuesta demasiado largo del ordenador al teclado, conviene repartir en varios ciclos de chequeo aquellas tareas que no necesitan entradas por teclados.

Los módulos que utilizan el teclado deben estar diseñados para que en el momento que necesiten una tecla cedan el control al módulo administrador de tareas. Estos módulos deben guardar la posición de ejecución en la que estaban cuando necesitaron esa tecla, para que cuando se les devuelva el control

(después de haber pulsado la tecla), sigan su modo normal de ejecución

Los datos que manejan los módulos están siendo recogidos o enviados bajo interrupción. Los módulos de procesamiento de datos no envía o recogen los datos, los toman o depositan en *buffers* de tipo circular, con dos punteros: uno indica la posición del último dato recogido o enviado al bus, y otro la del último dato procesado. Cuando ocurre una interrupción, según un programa preestablecido, estos datos se envían o recogen del bus GPIB. A cada llamada de interrupción se ejecutarían los distintos pasos vistos en la programación secuencial. Estas interrupciones pueden ser generadas mediante:

- Programación de la tarjeta interfase GPIB para que realice una interrupción cuando completa la operación de adquirir un dato, enviarlo, etc. Puede ser útil cuando se necesita rapidez.
- Utilización de la interrupción de reloj que proporciona el MS-Dos. Para llevar el control del tiempo, el Sistema Operativo Dos utiliza la interrupción que proporciona uno de los temporizadores de la placa base del PC (S253). Esta interrupción se ejecuta cada 1/18.2 segundos. Se puede cambiar el programa que se ejecuta cada vez que ocurre esta interrupción. En el recuadro adjunto se da un programa escrito en Módula, como Módulo Librería, que permite realizar esta operación.

Al ejecutar el procedimiento Arranca Reloj, el procedimiento Interrupción se ejecuta cada 1/18.2 seg., y a su vez ejecuta el procedimiento Adquiere, del módulo Adquisión de datos. Este es el módulo encargado de las relaciones del sistema con el bus GPIB. Para parar la adquisición sólo hay que ejecutar el procedimiento ParaReloj. El procedimiento Interrupción incrementa la variable Tictac, que puede ser utilizada por otros módulos para el control de tiempos, sin necesidad de hacer llamadas al BIOS. La única limitación de este programa está en que no se pueden llamar a otros servicios del DOS dentro de esta interrupción; sin embargo, se admiten las llamadas a servicios del BIOS.

La ejecución de los programas bajo control del reloj permite organizar la adquisición de una forma ordenada, lo más parecido a la forma secuencial.

4.3. Soluciones propuestas para problemas típicos

4.3.1 Bus en modo dato (línea ATN del bus=0)

La norma IEEE no habla del formato de los datos ni de la longitud de la cadena. Estos pueden estar en código ASCII o en binario, si el mensaje puede ser finalizado mediante la activación de la línea EOI o enviando los códigos CR o LF. Las cadenas de información pueden superar la capacidad de los *buffers* diseñados por el fabricante. Para atacar estos problemas lo más fácil es tener una librería de procedimientos o comandos primitivos; estos son los comandos más básicos para manejo de la tarjeta (enviar un byte, recibir un byte, desactivar la línea ATN, desconfigurar los aparatos que actúan como *listeners*, desconfigurar los aparatos que actúan como *talkers*, enviar el mensaje EOI, etc.). A base de estos comandos primitivos se elaboran procedimientos más sofisticados que tratan convenientemente las peculiaridades de cada instrumento. Antes de elaborar estos procedimientos, se debe hacer un estudio detenido de cómo envía y recibe los mensajes cada instrumento conectado al bus.

Hay instrumentos que después de recibir un mensaje de operación, mandan varios mensajes de respuesta y no admiten interferencias entre estos mensajes y otro mensaje de operación.

MODULO DE DEFINICION

```
DEFINITION MODULE Reloj;

EXPORT QUALIFIED TicTac, ParaReloj, ArrancaReloj;

PROCEDURE ArrancaReloj;
    (*Redirecciona el programa que se ejecuta al ocurrir la
interrupción del reloj (1/18 seg) *)

PROCEDURE ParaReloj;
  (*Vuelve la interrupción a su estado normal*)

VAR TicTac: CARDINAL;

END Reloj.
```

MODULO DE IMPLEMENTACION

```
IMPLEMENTACION MODULE Reloj;

FROM SYSTEM    IMPORT  BYTE,WORD,ADDRESS,
                       ADR,CODE,SWI;
FROM Devices   IMPORT  GetDevicesStatus,
                       SetDeviceStatus,
                       SaveInterruptVector,
                       RestoreInterruptVector;

FROM Adquisición IMPORT Adquiere;
CONST
  Int1=1CH;
  (* Interrupcion 'Timer Tick'*)
  Int2=66H;
  (* Interrupción nueva utilizada para llamar a la que
     antes estaba en 1CH *)

VAR
  oldIntV1,oldIntV2:ADDRESS;
  (* Vectores de las dos interrupciones utilizadas (1CH
     y 66H) *)
PROCEDURE ArrancaReloj;
BEGIN
  SaveInterruptVector(Int1,oldIntV1);
  SaveInterruptVector(Int2,oldIntV2);
  RestoreInterruptVector(Int2,oldIntV1);
  RestoreInterruptVector(Int1,ADDRESS(Interrupción));
END ArrancaReloj;
PROCEDURE ParaReloj;
BEGIN
  RestoreInterruptVector(Int1,oldIntV1);
  RestoreInterruptVector(Int2,oldIntV2);
END ParaReloj;
(*$T-*)
(*$R-*)
(*$S-*)
PROCEDURE Interrupcion;
BEGIN
  (* El compilador genera: PUSH BP
  SUB SP,[Espacio para variables locales] *)
  CODE (50H,51H,52H,53H,56H,57H,1EH,06H);
  (* PUSH AX,CX,DX,BX,SI,DI,DS,ES.
  Salva todos los registros *)
  Adquiere;
  INC(TicTac);
  SWI(Int2);
  CODE(07H,1FH,5FH,5EH,5BH,5AH,59H,58H);
                    (* POP ES,DS,DI,SI,BX,DX,CX,AX *)
  CODE(89H,0ECH);   (* MOV SP,BP *)
  CODE(5DH);        (* POP BP *)
  CODE(0CFH);       (* IRET *)
END Interrupción;
(*$T=*)
(*$R=*)
(*$S=*)
END Reloj.
```

Programa escrito en Modula-2 que permite cambiar el programa en ejecución al recibir una interrupción de reloj de MS-Dos

4.3.2. Tratamiento de timeouts

La norma no especifica la duración del *handshake* entre dos interfases. Hay instrumentos que para realizar la labor encomendada a través del bus necesitan un tiempo largo, mientras que otros son muy rápidos. Por otro lado, el bus no puede estar esperando indefinidamente que un instrumento envíe una respuesta. Es necesario establecer un tiempo límite o *timeout*, que variará de un instrumento a otro. Para controlar este tiempo se puede utilizar una variable como TicTac del programa ejemplo del apartado 4.2. Hay fabricantes que permiten programar este *timeout* en la tarjeta, pero suelen dar como correcto, al cabo de este tiempo, la operación de lectura o escritura.

4.3.3 Instrumentos Only-Talker

Son instrumentos que sólo permiten ser direccionados como emisor y necesitan direcciones secundarias para determinar la acción a realizar.

4.3.4. Medida sincronizada

La norma IEEE 488 permite, mediante el comando GET, enviar una orden de disparo a varios instrumentos; de esta forma se pueden iniciar operaciones simultáneas en varios dispositivos conectados al bus GPIB. Esto es casi imposible en la práctica ya que cada instrumento tiene una rapidez de respuesta diferente frente al disparo. La forma de operar con este comando es configurar los instrumentos que lo acepten para realizar la medida con disparo a través del bus (los que no lo acepten, se pueden hacer por hardware); a continuación enviar el comando GET. Los instrumentos realizarán sus correspondientes medidas. La última fase es la lectura por parte del ordenador de los datos que ha almacenado cada uno de los instrumentos de medida.

5. BIBLIOGRAFIA

- ANSI/IEEE Std 488-1978. «IEEE Standard Digital Interface for Programmable Instrumentation».
- «Intel Microsystem Components Handbook» Intel Corporation. Santa Clara, 1984.
- «The IBM Personal Computer Technical Reference Manual». IBM Corporation, Boca Raton, 1981.
- «PC-MATE IEEE 488 *Interface Software Package*. Tecmar Incorporated. Cleveland, 1983.
- Michael Hyman. *Memory resident, utilities, interrupts and disk management with MSPCDOS*. Management Information Source, Inc, 1986.
- Niklaus Wirth. *Programming in Modula-2*. Springer-Verlag, 1982.
- Stuart B. Greenfield. *Invitation to Modula-2*. Petrocelli-Books, 1985.
- Yu-Cheng Liu, Glen A. Gibson. *Microcomputer Systems: The 8086/8088 Family, Architecture, Programming, and Design*. Prentice-Hall, 1984.

Ponencia del tercer Congreso Internacional de Metrología Industrial (Metromática '87) □