33
2-7

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DIGITAL DE ADQUISICION DE DATOS PARA LA RED DE TELEMETRIA SISMEX

T E S I S

QUE PRESENTAN

PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

BERTHA LOPEZ NAJERA

R[illegible] RE

Director de Tesis: M. en I. Roberto Quaas Weppen

México, D. F. 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

I. INTRODUCCION

México se encuentra dentro de una zona de gran actividad sísmica en el mundo. Los eventos ocurridos han provocado mucho interés por el estudio y análisis de estos fenómenos y sus consecuencias, necesitandose para esto una amplia infraestructura de registro y procesamiento.

En 1973 fué instalado en el Instituto de Ingeniería un sistema de telemetría sísmica, SISMEX, el cual ha funcionado ininterrumpidamente desde entonces.

El sistema SISMEX recibe señales analógicas vía radio de las estaciones sismológicas actualmente establecidas en la ciudad de México, y estados aledaños como Guerrero, Puebla y Edo. de México.

Actualmente este sistema obtiene los eventos por medio de unos registradores de tambor. Una vez que se desean analizar estas señales, es necesario retirar el papel de los tambores y utilizar métodos gráficos para obtener su información, por lo cual los resultados de un sismo no se obtienen ni con la rapidez, ni con la exactitud deseada.

Por lo anterior se planteó la conveniencia de ampliar SISMEX con un sistema digital basado en una computadora PC, que permite la evaluación de la información de una manera más oportuna y rápida, siendo el objetivo de la presente tesis el desarrollo y análisis de este sistema digital de adquisición de datos.

La utilización de una PC da la posibilidad de utilizar métodos de análisis más sofisticados, y obtener así información referente al evento sísmico, como su localización, tiempo de ocurrencia, tiempo de arribo a la ciudad de México, duración, magnitudes, gráficas, etc.

El sistema digital esta formado por una tarjeta cuya tarea principal es la adquisición de datos y detección de un evento sísmico en tiempo real y una computadora PC que comunica al sistema con el usuario y procesa los datos a través de un programa en lenguaje de alto nivel.

La tarjeta, instalada en una de las ranuras de expansión de la computadora PC, es controlada por un microprocesador 8088 y efectúa la transferencia de datos a través de una memoria de conmutación que permite procesos en tiempo real.

El programa que rige a la tarjeta de adquisición cuenta con un algoritmo de detección de eventos sísmicos que, al momento de detectar alguno, envía una interrupción a la PC para que ésta almacene los datos obtenidos en disco duro y posteriormente los procese. El manejo de la interrupción permite a la PC realizar otras actividades independientes al sistema.

El sistema se introduce en el capítulo II, mediante una descripción breve de la red telemetrica de SISMEX. En los capítulos III y IV se plantea la filosofía general del diseño de la tarjeta adquisidora, el hardware, la justificación del uso de cada uno de los componentes del sistema y la lógica de comunicación entre la tarjeta adquisidora y la computadora PC. En los capítulos V y VI se presenta el software desarrollado: la programación del adquisidor y de la PC, respectivamente. Finalmente en el capítulo VII se dan algunos resultados preliminares de las pruebas efectuadas al sistema.

Como referencia se presentan en los anexos los listados completos de todos los programas, diagramas de tiempo y material de consulta .

II. RED DE TELEMETRIA SISMICA, SISMEX

En este capítulo se da una visión general del sistema de telemetría (SISMEX), de la cual se obtienen las señales que son utilizadas por el sistema de adquisición de datos.

El Sistema de información sismo-telemétrica de México, SISMEX, es un proyecto que fué desarrollado por el Intituto de Ingeniería de la UNAM en colaboración con la UNESCO en 1973, con el objeto de registrar y monitorear en tiempo real la actividad sísmica en el D.F. y estados aledaños.

Desde 1973 hasta la fecha, el sistema ha operado de manera ininterrumpida y ha sido de gran importancia para la detección, localización de temblores y en general estudios sismológicos diversos.

La red consta de 9 estaciones remotas de campo que envían en forma continua las señales sísmicas captadas por sismómetros verticales de alta ganancia a un puesto central de registro situado en el Instituto de Ingeniería.

Estas señales moduladas en FM, se multiplexan junto con las de otras estaciones y se transmiten mediante enlaces radiofónicos al puesto central.

En la figura 2.1 se muestra la ubicación de las estaciones de la red sismológica y los enlaces telemétricos existentes. En la tabla 2.1 se da la localización de las estaciones.

Una vez recibida la información, se filtra y demodula, con el fin de obtener las señales originales en forma de sismogramas que son graficados en papel mediante registradores de tambor.

Dado que la información de la red es recibida en el puesto central en tiempo real, se tiene allí una referencia común de tiempo que registra la fecha y hora precisa de ocurrencia de los temblores en forma de pulsos ó marcas cada minuto y hora superpuestas al sismograma. Cada 24 horas se cambian las hojas con los registros para ser analizadas.

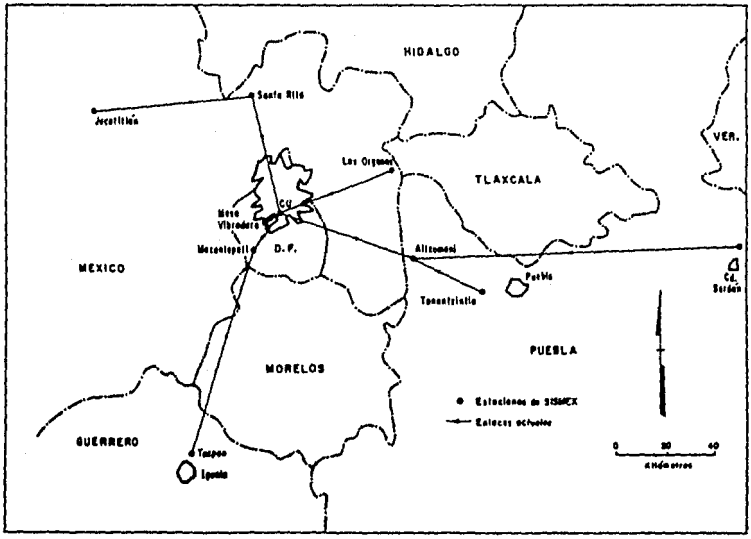


Figura 2.1 Ubicación de las estaciones de la red de telemetría.

ESTACION	UBICACION	LOCALIZACION LAT.N.	LONG.N	ALTITUD	TIPO
Sta.Rita	Edo.de México	19.796	99.258	2725m	T,R
C.U.	D.F.	19.326	99.182	2275m	PCR,T
Tonantzintla	Puebla	19.021	98.308	2205m	T
C.Tuxpan	Guerrero	18.376	99.468	1750m	T
Cd.Serdán	Puebla	18.988	97.376	2985m	T
Jocotitlán	Edo.de México	19.734	99.762	3900m	T
Los Organos	Edo.de México	19.592	98.724	2850m	T
Altzomoni	Puebla	19.143	98.655	3900m	R,T
Mezontepec	D.F.	19.188	99.228	3450m	T
Mesa Vibradora	D.F.	19.326	99.182	2275m	T

T = estación transmisora
R = estación retransmisora
PCR = puesto central de registro

Tabla 2.1 Estaciones telemétricas de SISMEX.

De una manera muy general, cada estación de campo está compuesta por los bloques mostrados en la fig. 2.2. Para garantizar una prolongada autonomía de operación independiente de la energía eléctrica comercial, cada estación se alimenta por medio de celdas solares y baterías en flotación.

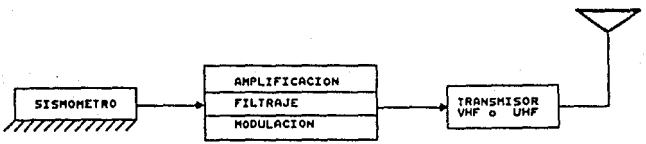


Fig. 2.2 Estación sismológica telemétrica.

En la figura 2.3 se muestra el proceso y equipo de recepción en el puesto central de registro. Un sistema de suministro de energía mantiene en operación al sistema en caso de falla de la energía comercial.

Como base de tiempo se recibe la señal codificada WWV, que a su vez sincroniza un reloj universal.

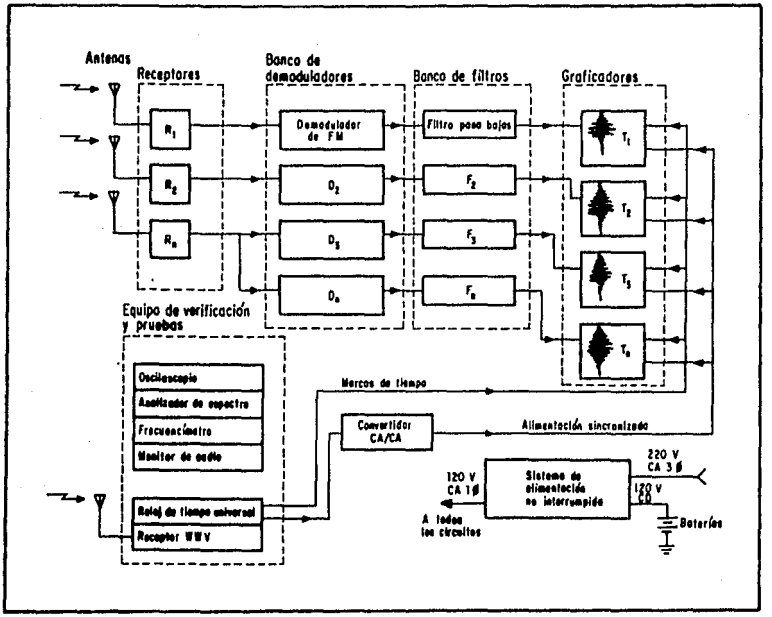


Figura 2.3 Proceso de recepción.

III. DESCRIPCION DEL SISTEMA

3.1 Consideraciones iniciales de diseño y arquitectura general del sistema.

El objetivo del sistema es la detección y almacenamiento en disco de los eventos sísmicos registrados por la red SISMEX. En este capítulo se describe en general el funcionamiento del sistema desarrollado, la filosofía de diseño y el planteamiento de las especificaciones y restricciones iniciales.

Las entradas al sistema son las señales demoduladas que llegan a los registradores de tambor en el puesto central de registro.

Las características de las señales de SISMEX son las siguientes :

- **12 canales de registro**
- **Banda de frecuencia de las señales sísmica:0-30Hz**
- **Nivel de salida de cada canal: ± 1 Vp**
- **Rango dinámico: 40 dB**

A partir de estos datos iniciales se plantearon las siguientes especificaciones:

El sistema de adquisición debía ser capaz de procesar por lo menos 12 señales analógicas. Con el objeto de permitir una futura expansión del sistema se escogieron 16.

Debido a que la frecuencia máxima de las señales sísmicas transmitidas es de 20 Hz, cada canal debía ser muestreado por lo menos al doble de esta frecuencia para cumplir con el teorema de Nyquist. El muestreo se fijó a **40 muestras por segundo**, lo que da un intervalo de muestreo ó tiempo de adquisición de **25 mseg.**

Dado que el rango dinámico de la red telemétrica es de 40 dB, se escogió un conversor analógico-digital de 8 bits, es decir, con una resolución de 48 dB.

Para adecuar las señales analógicas de salida de SISMEX al sistema de adquisición hubo que considerar el acondicionamento de las mismas a los niveles de voltaje requeridos por el conversor A/D.

Por facilidad de manejo, costo y disponibilidad se consideró inicialmente emplear para el sistema una computadora PC. Sin embargo surgió la pregunta si un procesador podría realizar simultáneamente las 4 tareas básicas que son: adquisición de datos, procesamiento de un algoritmo de detección, memoria de preevento y almacenamiento en tiempo real en disco.

De estas cuatro fases, la del almacenamiento en tiempo real se consideró la más pesada por lo cual se comenzó por evaluarla primero.

Se desarrollaron diversos programas para medir los tiempos de escritura a disco, tanto flexible, como duro. Estas mediciones se hicieron con la ayuda de un osciloscopio y empleando las interrupciones del sistema operativo para el acceso a disco. Los programas desarrollados se presentan en el anexo A.

Los resultados de las mediciones fueron :

Para disco flexible:

1. Tiempo promedio de escritura de 512 bytes a una pista y sector determinado: 200 mseg.

2. Tiempo promedio de movimiento de las cabezas de una pista a otra contigua: 200 mseg.

3. Tiempo promedio de escritura de una pista completa (512 bytes x 9 sectores = 4608 bytes), más el cambio a otra pista contigua : 2000 mseg.

Para disco duro:

1. Tiempo promedio de escritura de 512 bytes a una pista y sector fijos : 17 mseg.

2. Tiempo promedio de movimiento de las cabezas de una pista a otra contigua : 35 mseg.

3. Tiempo promedio de escritura de un cilindro ó pista (512 bytes x 17 sectores = 8704 bytes), más el cambio a otro contiguo: 324 mseg.

Comparando estas mediciones con el tiempo de muestreo especificado de 25 mseg, y a pesar de que el disco duro es aproximadamente diez veces más rápido que el flexible, se concluyó que los tiempos de acceso y escritura a disco eran considerables y no era factible entonces ejecutar la tarea de almacenamiento, la de adquisición y las del algoritmo de detección y memoria de preevento en forma simultánea con un solo procesador. Además se tenía la restricción de que los procesos a disco no son interrumpibles.

Por la anterior restricción se resolvió el problema de otra manera y se llegó a la configuración del sistema mostrada en figura 3.1: un módulo independiente para adquisición de datos, una computadora PC para coordinar el proceso y almacenar la información, y una memoria conmutable para el intercambio de datos entre el adquisidor y la PC.

A su vez cada módulo quedó integrado por los siguientes bloques:

Adquisidor de datos

- Multiplexor y conversor A/D para 16 canales
- Microprocesador
- Memoria conmutable
- Conmutador
- Programa de control de adquisición y algoritmo de disparo

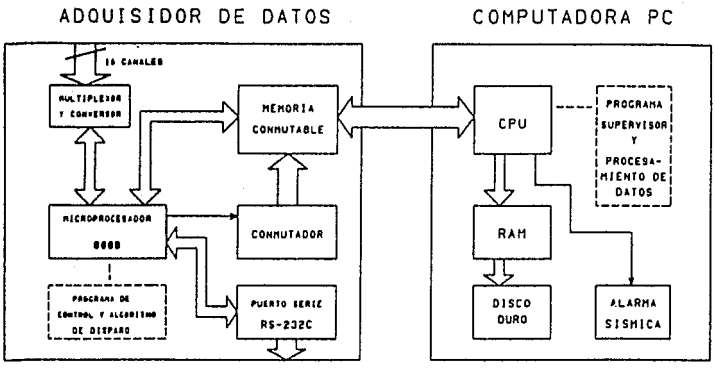


Figura 3.1 Diagrama de bloques simplificado del sistema digital de adquisición de datos.

Computadora PC

- CPU
- Memoria RAM
- Disco duro
- Programa de control y procesamiento de datos
- Dispositivo para una alarma sísmica

Por su bajo costo, facilidad de manejo y compatibilidad con la computadora PC, se seleccionó para el MPA el microprocesador Intel 8088 (mismo que utiliza la PC). Esto permite además utilizar el mismo reloj de la PC para ambos.

Para fines del control del proceso del sistema, se designó a la computadora PC como maestro y al adquisidor como esclavo.

El modo de operación del sistema es el siguiente:

El microprocesador de adquisición, o adquisidor de datos, que se abreviará como MPA, bajo control de un programa autónomo, convierte 16 canales analógicos a razón de 40 muestras/seg, y almacena transitoriamente los datos en la memoria conmutable que comparte con la PC.

El mismo microprocesador evalúa los datos mediante un algoritmo para detección de eventos. En caso de detectarse y confirmarse un sismo en cualquiera de las estaciones remotas, el sistema se dispara, generando una interrupción a la PC para que transfiera el bloque de datos almacenados en la memoria conmutable a su memoria RAM y de allí al disco duro.

Transferidos los datos de la memoria conmutable, el adquisidor continúa la recolección de datos mientras la PC almacena en disco la información del bloque anterior. Este procedimiento se repite hasta que la señal sísmica disminuye por debajo del umbral de disparo, dando por terminado el evento.

El control de la memoria de conmutación es el punto clave del diseño. Debido a que se llevan a cabo dos procesos asíncronos en forma simultánea, el de adquisición y almacenamiento temporal [adquisidor] y el de transferencia a RAM y vaciado a disco [PC], se dividió la memoria en dos "buffers" conmutables RAM1 y RAM2 como se muestra en la figura 3.2.

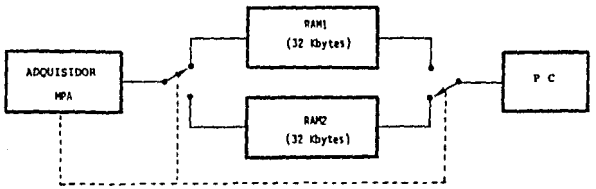


Figura 3.2 Memoria conmutable.

Mientras el adquisidor almacena sus datos en una de las memorias, la PC vacía el otro a su memoria RAM y de allí a disco.

Con lo anterior se logran ejecutar ambas tareas simultáneamente y se resuelve el problema del tiempo relativamente grande de acceso al disco.

Hay que aclarar que un registro sísmico puede durar hasta varios minutos. Dado que la capacidad de la memoria conmutable es pequeña, para un evento se tendrán que conmutar las memorias varias veces dependiendo de la duración del sismo.

El tamaño de la memoria conmutable se escogió con una capacidad tal, que puede almacenar temporalmente 30 segundos de datos de preevento información previa al inicio del sismo y los datos correspondientes al tiempo de propagación de la onda sísmica, desde la estación más lejana de la red, hasta el puesto central en el Instituto de Ingeniería en C.U. Actualmente la estación mas lejana se encuentra a 240 km en línea recta de CU. Esto se ilustra en el diagrama de la figura 3.3.

Sabiendo que la velocidad de propagación de la onda sísmica (onda p) es de **6 a 8 [km/seg]**, implica que tardará aproximadamente **40 segundos** (240 [km] / 6 [km/seg]), desde el momento en el que es detectada en la estación remota hasta su arribo a C.U.. Si a este tiempo se le suman los 30 segundos de preevento, dan 70 seg de datos que deben ser almacenados. Esto es:

30	+	40	=	70 seg.
(tiempo de preevento)	+	(tiempo de propagación)		(tiempo total de registro en memoria)

Esto equivale en términos de capacidad de memoria a 44,800 bytes, ya que:

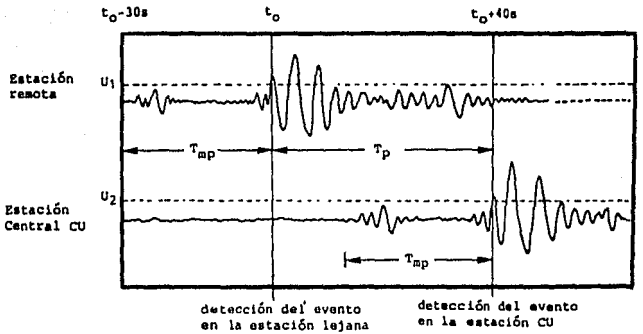
$$40 \frac{\text{muestras}}{\text{seg·canal}} \times 1 \frac{\text{byte}}{\text{muestra}} \times 16 \text{ canales} \times 70\text{seg} = 44{,}800 \text{ bytes}$$

Ajustando este valor a la capacidad de memorias comerciales, da 64 Kbytes (65 536 bytes). Se escogieron por tanto memorias de 32 Kbytes para RAM1 y RAM2.

Cabe aclarar que debido al tiempo de muestreo utilizado de 25 mseg, cada memoria de **32 Kbytes** tarda en llenarse **51.2 segundos**, mientras que a la PC le toma 80 ms transferir los **32 Kbytes** de la memoria conmutable a su RAM y de allí 1.6 segundos para pasarlos a disco duro. Con ello se garantiza que no habrá problemas en cuanto a tiempos de acceso a disco mientras el MPA conmuta memorias.

Basado en el anterior cálculo, durante el tiempo en que el adquisidor muestrea y llena la memoria conmutable, la PC puede efectuar holgadamente diversas tareas en línea tales como estimación de los tiempos de arribo, localización de eventos en un mapa, e incluso graficación en pantalla de la información registrada.

Dado que el MPA tiene el control del proceso de adquisición de los datos en tiempo real, también se le asignó la tarea de ejecutar el algoritmo de detección. Ya que esta tarea implica programación a nivel de lenguaje de máquina, se decidió, en esta primer etapa de desarrollo, diseñar el algoritmo de una manera sencilla sin complejos cálculos numéricos. Sin embargo, para permitir un algoritmo de disparo y alarma más sofisticado, se prevee utilizar un puerto de salida por



Distancia est. remota-CU: 240 km
Velocidad de propagación: 6 km/s
Tiempo de propagación T_p: 40 s
Tiempo de preevento T_{mp}: 30 s
Tiempo total: 70 s

No. de canales: 16
Velocidad de muestreo: 40 M/s
Longitud de palabra: 8 bits/M
Capacidad del buffer (mínima):
70s x 40M/s x 16 can.= 44,800 bytes
Capacidad seleccionada: 64K bytes
(distancia a la est. remota=430 km)

Figura 3.3 Tiempos estimados de propagación para el cálculo en la capacidad de almacenamiento de las memorias de conmutación.

donde se envíen datos al exterior, con lo cual se podría tener en otra PC un programa de disparo más elaborado, usando un lenguaje de alto nivel.

Hasta aquí se ha descrito en forma muy general la filosofía de diseño, modo de operación y especificaciones del sistema. A continuación se detallarán la estructura del "hardware" desarrollado y su funcionamiento.

3.2 Interconexión PC y MPA.

El sistema de adquisición se ensambló sobre una tarjeta de circuito impreso conectada en una ranura interna de la PC. La interconexión se hizo como muestra la figura 3.4.

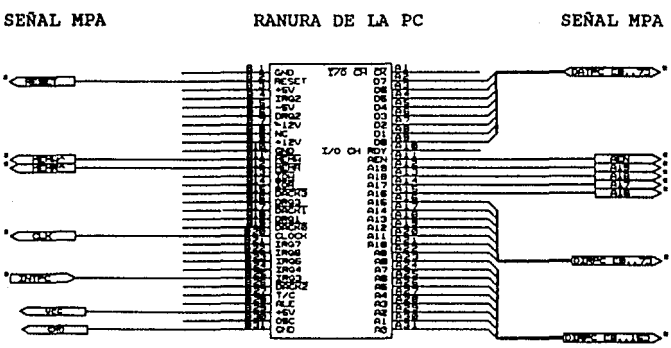


Figura 3.4 Ranura y conexiones del MPA y PC.

Las señales usadas derivadas de la ranura de la PC son:

- RESET: Se emplea la misma señal de la PC.
- CLK: Señal de reloj de la PC al MPA (8 MHz).
- $\overline{\text{MEMR}}$ y $\overline{\text{MEMW}}$: Las líneas de petición a lectura y escritura de memoria de la PC al MPA.
- DIRPC [0..15]: El bus de direcciones de la PC al MPA.
- DATPC [0..7]: El bus de datos de la PC.
- INTPC: La línea de interrupción IRQ3 del MPA a PC.
- AEN: Línea del control que indica la presencia de direcciones del DMA (control de acceso directo a memoria) en el bus.
- A16..A19: Líneas de direcciones para la decodificación.
- +5V y GND: Voltaje de alimentación 5V y tierra.

3.3 Mapa de memoria de la PC.

A través de la memoria de conmutación se comunican el MPA y la PC, y por tanto ambos deben tener acceso a ella. Se buscó que este buffer fuera parte de la configuración de memoria de la PC. Para ello fue necesario buscar una zona dentro del mapa de memoria de la PC que no fuera ocupada, encontrándose los bloques de dirección física **D0000H** y **E0000H**, que están destinadas para ampliación de memoria por el usuario.

Por lo anterior la PC, al generar la dirección **D0000H** en su bus de direcciones selecciona una de las memorias de conmutación. De esta manera la RAM1 ó la RAM2 son parte del mapa de memoria de la PC, sin que ésta pueda distinguir a cual de ellas está teniendo acceso, ya que el control de conmutación lo tiene el MPA. El mapa de memoria de la PC se muestra en la tabla 3.1.

Para facilitar la intercomunicación y control entre la PC y el MPA se añadió un puerto (latch) llamado **Puerto de control PC, PPC.** A este puerto se le asignó la otra dirección, **E0000H**, del mapa de memoria libre de la PC. A través del PPC, la PC envía códigos de control al MPA, que lo decodifica y ejecuta.

MAPA DE MEMORIA DE PC

RANGO DE DIRECCIONAMIENTO (HEXADECIMAL)	DISPOSITIVO SELECCIONADO	FUNCION
00000 - 003FF	RAM *	Vectores de interrupción del CPU 8088.
00400 - 0047F	RAM *	Reservado para variables del ROM BIOS.
00480 - 0050F	RAM *	Reservado para memoria del monitor.
00150 - 1FFFF	RAM *	Area del usuario para programa/sistema operativo.
20000 - 9FFFF	RAM *	Disponible para tarjetas de expansión de memoria.
A0000 - AFFFF	NO INSTALADO	Reservado para EGA.
B0000 - B7FFF	I/O	Reservado para tarjetas de expansión de video.
B8000 - BFFFF	RAM *	Buffer para el despliegue de video.
C0000 - C7FFF	NO INSTALADO	Reservado para el BIOS de EGA.
C8000 - C9FFF	-	Controlador de disco duro.
D0000 - DFFFF	RAM1/RAM2	Memoria de conmutación compartida con el MPA.
E0000 - EFFFF	PUERTO	Puerto de control del sistema de adquisición.
FE000 - FFFFF	ROM	Memoria permanente de PC.

* memoria RAM propia de la PC

Tabla 3.1 Descripción del mapa de memoria de la PC.

3.4 Configuración del microprocesador de adquisición.

El diagrama general de bloques del adquisidor se muestra en la figura 3.5. A continuación se describirá con detalle los distintos elementos que lo integran.

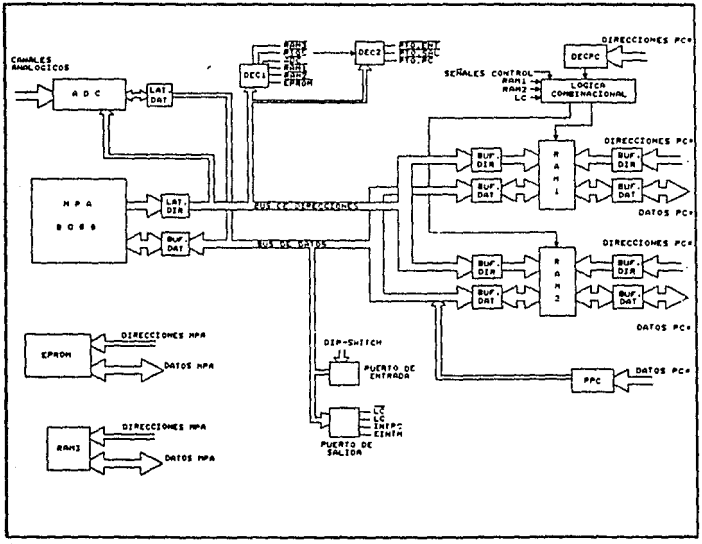


Figura 3.5 Diagrama de bloques del sistema.

El circuito integrado 8088 de INTEL (U2) es un microprocesador de 16-bits, es decir, su unidad aritmetica y lógica, registros internos y la mayoría de sus instrucciones, están diseñadas para trabajar con palabras binarias de 16-bits. Su diagrama se muestra en la figura 3.6.

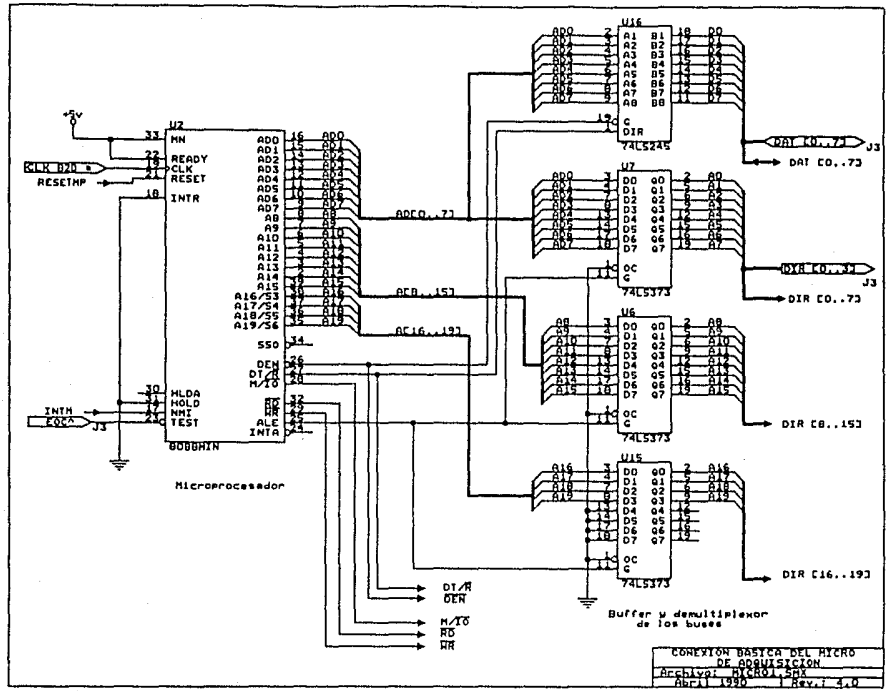


Figura 3.6 Diagrama del microprocesador.

El canal de datos y el de direcciones se encuentran multiplexados. Su bus de datos es de 8 bits y el de direcciones de 20 bits, pudiendo direccionar 1,048,576 bytes. Dicho multiplexaje se da de la siguiente manera:

Las líneas del bus de datos AD0 a AD7, como se observa en la figura 3.6, son usadas al inicio de un ciclo de máquina, para enviar direcciones y más tarde enviar o recibir datos. La señal **ALE** es activada por el 8088 cuando las direcciones están presentes en el bus multiplexado y con ellas se habilitan los latches externos U7, U6, y U15.

Después de seleccionar las líneas de dirección AD0 - AD7, el microprocesador prepara estas líneas en modo de entrada o salida para recibir o enviar datos a través del buffer bidireccional U16. Este proceso se controla con la señal $\overline{\text{DEN}}$, que habilita al buffer, y la señal DT/$\overline{\text{R}}$, que controla dicho direccionamiento (transmitir/recibir).

U16 es un buffer bidireccional que aisla el bus de datos del microprocesador y proporciona una mayor capacidad de corriente para manejar todos los dispositivos conectados en paralelo a este bus de datos (DAT[0..7]).

Otras de las señales propias del MPA utilizadas en el sistema son:

- Las señales de control de buses para lectura, escritura y localidades de memoria (**RD**, **$\overline{\text{WR}}$** y **M/$\overline{\text{IO}}$**).

- En esta aplicación el 8088 se encuentra operando en **"modo mínimo"**, ya que los canales no son compartidos con otros microprocesadores.

- El reloj del MPA (**CLK**) de 8 Mhz se toma del slot de la PC.

- La interrupción utilizada es la no-mascarable **NMI**, la cual será activada por un reloj de 40 Hz (señal IMI), y se usa para el muestreo y conversión de los 16 canales.

  La base de tiempo utilizada para muestrear los datos es generada por medio de un oscilador controlado por cristal trabajando a **4MHz**. A través de una cadena de contadores de década se divide la frecuencia hasta obtener la señal de 40Hz. El circuito correspondiente se analizará más adelante en la decripción del conversor A/D (sección 3.4.5).

La señal IMI, activada cada 25mseg (40Hz), interrumpe al MPA a través del buffer U33, que se muestra en la figura 3.7, mismo que es habilitado por la señal $\overline{\text{EINTM}}$, enviada a través del puerto de salida (U29), que se describirá en la sección 3.4.3.

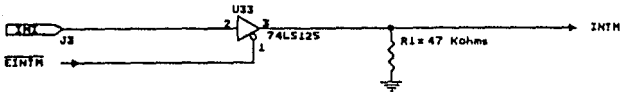


Figura 3.7 Buffer habilitador de interrupción para el MPA.

La inicialización del sistema se lleva a efecto poniendo en estado alto la terminal de **RESETMP**, ya sea externamente al encender el sistema, o manualmente por medio de un interruptor. Para ello se diseñó el circuito mostrado en la figura 3.8, basado en un circuito LM555 alambrado como un multivibrador monoestable, el cual genera un pulso de 2 μseg (el mínimo requerido es de 4 ciclos de reloj) al oprimirse el switch SW. La señal RESET B2* es la señal proveniente de la PC.

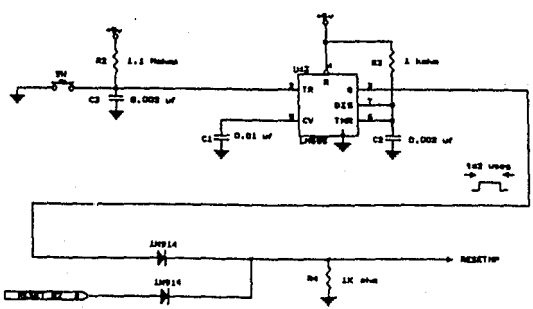


Figura 3.8 Circuito de RESET.

3.4.1 Mapa de memoria del MPA.

Físicamente, la memoria del MPA está organizada por bloques seleccionados por las líneas de direcciones del micropocesador.

Ciertas localidades de memoria están asignadas para operaciones específicas del CPU. Por ejemplo las localidades desde la dirección **FFFF0H** hasta la **FFFFFH**, están reservadas, entre otras funciones, para el vector de inicio del programa después de activarse la señal de RESET. También las localidades **00000H** a **003FFH** son asignadas para almacenar vectores de interrupción y variables de programación.

Debido a lo anterior, el mapa de memoria se escogió en la forma mostrada en la tabla 3.2. Con base en esta tabla, se diseñó la lógica de decodificación de la figura 3.9. Como se puede ver de ésta figura, las señales $\overline{\text{RAM3}}$, $\overline{\text{PTOS}}$, $\overline{\text{ADC}}$, $\overline{\text{RAM1}}$, $\overline{\text{RAM2}}$ Y $\overline{\text{EPROM}}$, activas en estado bajo, son generadas por el decodificador U25 que utiliza las direcciones altas A16 - A19.

Asimismo existe un subdireccionamiento para los puertos por medio de U42 que genera las señales PTO.SAL, PTO.ENT y PTO.PC que habilitarán a los respectivos dispositivos.

En la tabla 3.3 se describe la función de cada una de las señales de selección y se indica el espacio de memoria asignado.

3.4.2 Memorias RAM3 y EPROM.

En la figura 3.10 se presenta el diagrama de las memorias RAM3 y EPROM del MPA.

La RAM3 (U1), tiene una capacidad de almacenamiento de 2 Kbytes y es utilizada por el MPA como "stack" y para variables del programa. Por otro lado, la EPROM (U30) es de 4 Kbytes y en ella reside el programa del MPA.

MAPA DE MEMORIA DE LA TARJETA DE ADQUISICION

LINEAS DE DIRECCION DEL 8088																				DISPOSITIVO SELECCIONADO	RANGO DE DIRECCIONAMIENTO	SUBSELECCION	
																						DISPOSITIVO SELECCIONADO	RANGO DE DIRECCIONAMIENTO
A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0				
0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RAM3	00000 - 1FFFF		
0	0	1	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	PTOS	20000 - 3FFFF	PTO.ENT	20000 - 23FFF
			0	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*			PTO.SAL	24000 - 27FFF
			0	1	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*			PTO.PC	28000 - 2BFFF
			0	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*			NC	
			1	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*			NC	
			1	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*			NC	
			1	1	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*			NC	
			1	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*			NC	
0	1	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	ADC	40000 - 5FFFF		
0	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	NC	60000 - 7FFFF		
1	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	NC	80000 - 9FFFF		
1	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RAM1	A0000 - BFFFF		
1	1	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RAM2	C0000 - DFFFF		
1	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	EPROM	E0000 - FFFFF		

NOTA: * - don't care
NC - sin conectar

Tabla 3.2 Mapa de memoria del MPA.

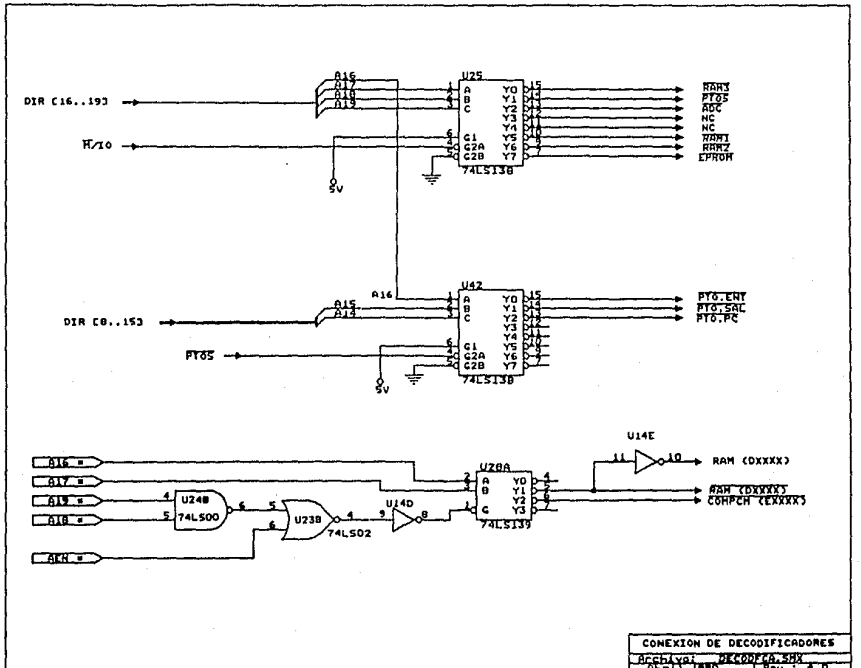


Figura 3.9 Decodificadores del sistema de adquisición.

DESCRIPCION DEL MAPA DE MEMORIA DE LA TARJETA DE ADQUISICION

Dirección	Señal	Descripción
0000 - 1FFFF	RAM3	Memoria utilizada para almacenar vectores de interrupción y variables de programación.
20000 - 23FFF	PTO.ENT	Espacio asignado al puerto de entrada en la tarjeta de adquisición.
24000 - 27FFF	PTO.SAL	Espacio asignado al puerto de salida en la tarjeta de adquisición. Por medio de este puerto se controla la lógica de conmutación de las memorias RAM1 y RAM2, así como las interrupciones de la PC y el MPA.
28000 - 2BFFF	PTO.PC	Espacio asignado al puerto que comunica la PC con el MPA por medio de códigos de control.
2C000 - 3FFFF	NC	Espacio disponible para expasión de puertos.
40000 - 5FFFF	ADC	Espacio asignado para la conversión A/D.
60000 - 7FFFF	NC	Espacio disponible para expansión de memoria.
80000 - 9FFFF	NC	Espacio disponible para expansión de memoria.
A0000 - BFFFF	RAM1	Espacio asignado a la memoria RAM1 conmutable del MPA. En esta memoria se almacena transitoriamente la información convertida.
C0000 - DFFFF	RAM2	Espacio asignado a la memoria RAM2 conmutable del MPA. En esta memoria se almacena transitoriamente la información convertida.
E0000 - FFFFF	EPROM	Espacio definido para los vectores y el programa del microprocesador.

Tabla 3.3 Descripción del mapa de memoria del MPA.

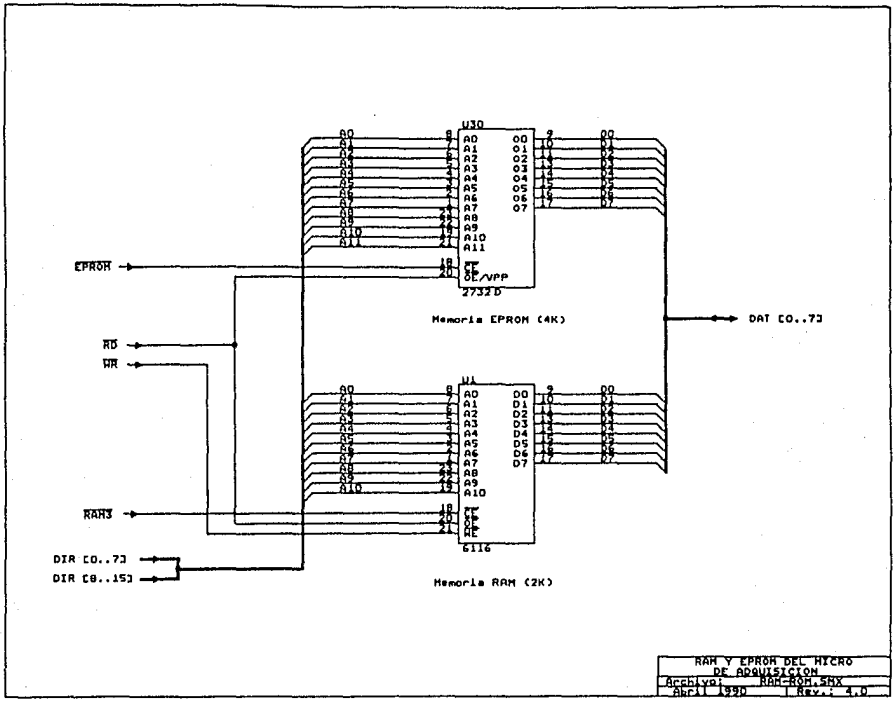


Figura 3.10 Conexión de RAM3 y EPROM del MPA.

3.4.3 Puertos.

Los puertos con los que cuenta el sistema, y los cuales se encuentran ilustrados en la figura 3.11 son:

- Puerto de entrada
- Puerto de salida
- Puerto PC

El **puerto de entrada** es utilizado para transferir externamente información al adquisidor, siendo usado para calibración y rutinas de prueba. El **puerto de salida** es un dipositivo mediante el cual el MPA envía datos para el control de la adquisición, conmutación de memorias e interrupción de la PC. Por último el **puerto de la PC** es la vía de comunicación entre la PC y el MPA.

A continuación se explica cada uno de estos dispositivos.

- **Puerto de entrada**

El puerto de entrada U5 es un flip-flop octal tipo D con salidas de tres estados que se conectan al bus de datos [0..7]. Las entradas se conectan directamente a un grupo de 8 interruptores SW1 utilizados para programar manualmente ciertas operaciones del MPA.

El circuito es activado con el flanco de subida de un pulso generado por una lógica combinacional con U23A y U14A a partir de las señales $\overline{RD}$ y $\overline{PTO.ENT}$. Las resistencias R5 a R12 mantienen las entradas del puerto en alto cuando los interruptores están abiertos.

- **Puerto de salida**

Este puerto, identificado como U29 en la figura 3.11, es también un flip-flop octal tipo D con salidas de tres estados, activado por un flanco de subida. Este flanco es generado a partir de las señales $\overline{WR}$, y $\overline{PTO.SAL}$, junto con el flanco del reloj de la PC.

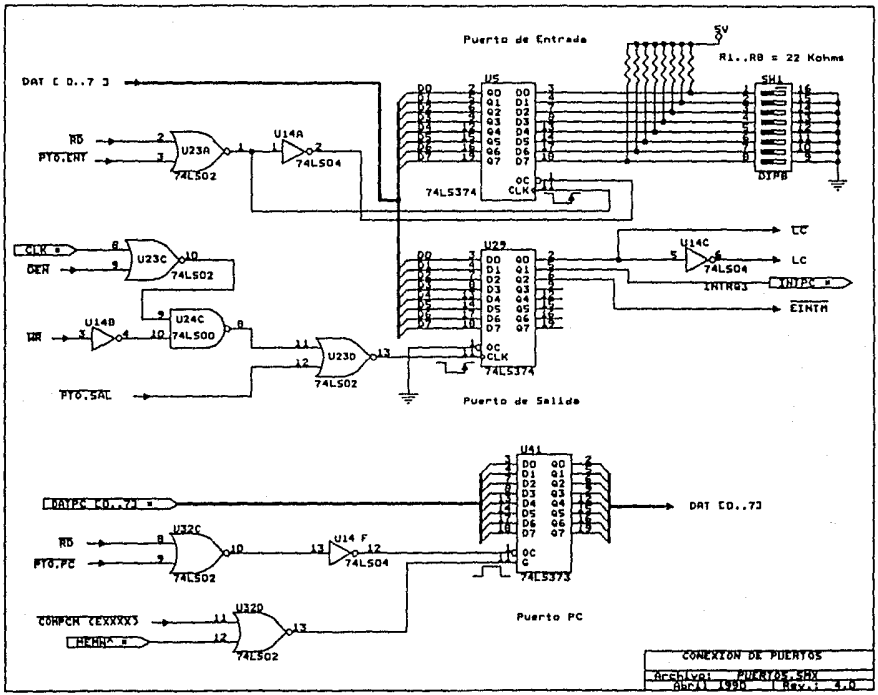


Figura 3.11 Puertos del adquisidor.

Las entradas provienen del canal de datos del MPA y sus salidas generan las líneas de control LC y $\overline{LC}$ para la conmutación de memorias. Además genera dos líneas de interrupción, una para la PC (INTPC) y la otra para el MPA ($\overline{EINTM}$).

Es importante mencionar que las líneas de control LC y $\overline{LC}$ van conectadas a 14 circuitos integrados, por lo que, para evitar una sobrecarga del circuito, se colocó a la salida del puerto el buffer U44 como se muestra en la figura 3.12. Por lo anterior $\overline{LC}$ se dividió en $\overline{LC1}$, $\overline{LC2}$ y $\overline{LC3}$, mientras que LC en LC1 y LC2. Cabe hacer notar que se dió esta asignación con el único propósito de distinguir las líneas que llegan a los diferentes circuitos de la tarjeta.

Del diagrama se puede observar que la línea de salida **1Y1** se conecta a las entradas **1A2,1A3 y 1A4** del mismo integrado. Esto se hizo para dar un retardo por medio del buffer a la señal $\overline{LC}$ igual que LC, la cual pasa por un inversor después del puerto de salida.

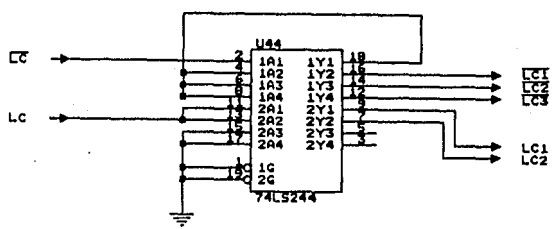


Figura 3.12 Buffer para las señales LC y $\overline{LC}$.

· Puerto de comunicación con la PC.

Identificado como U41 en la figura 3.11, este puerto es un flip-flop tipo D de tres estados.

A través de este puerto de entrada la PC se comunica con el MPA con el fín de coordinar las tareas de ambos procesadores mediante el envío de códigos de comando.

Con las señales **MEMW** y **COMPCM(EXXXX)** la PC genera un pulso que almacena en la memoria el código deseado. Este código a su vez es transmitido al canal de datos del adquisidor, al habilitarse la salida del puerto con las señales **RD** y **PTO.PC**.

La señal COMPCM(EXXXX) y RAM(DXXXX) son generadas por el decodificador U28A de la figura 3.9, utilizando las direcciones A16 y A17 de la PC. Las otras lineas de dirección A18 y A19, junto con la señal AEN se combinan mediante una lógica con U24B, U23B y U14B para generar la señal de habilitación de U28A. Cuando AEN se encuentra en estado alto, es decir, durante las operaciones de DMA de la PC, se deshabilita el decodificador.

3.4.4 Memoria Conmutable.

Como ya se mencionó al inicio del capítulo, el objetivo principal de esta memoria es el almacenamiento temporal de los datos convertidos por el MPA para que posteriormente la PC los transfiera de allí a su memoria RAM y luego a disco.

En la figura 3.13 se muestra nuevamente un diagrama de bloques de la memoria conmutable y su interconexión con el MPA y la PC.

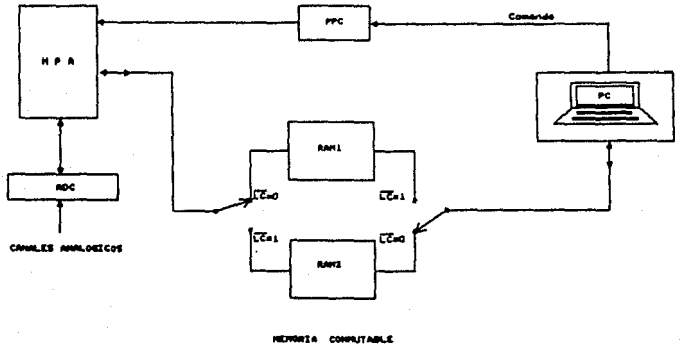


Figura 3.13 Diagrama general de la comunicación entre el MPA y la PC . Memoria de conmutación y puerto de control.

La conmutación de las memorias se realiza a través de la línea de control (LC) proveniente del puerto de salida y está bajo control del microprocesador MPA.

Dependiendo del estado lógico de la señal LC, la PC y el MPA tendrán acceso a la al memoria RAM1 ó RAM2, respectivamente.

Con LC en estado bajo, el MPA tiene acceso a RAM1 y la PC a RAM2. Por el contrario, si LC se encuentra en estado alto, la PC tendrá acceso a RAM1 y el MPA a RAM2.

La tabla mostrada en la figura 3.14 indica, de una manera general, la función de esta señal LC.

LC = 0		
Para el MPA		
datos y direcciones:	habilitados	deshabilitados
	RAM 1	RAM 2
Para la PC		
datos y direcciones:	habilitados	deshabilitados
	RAM 2	RAM 1

LC = 1		
Para el MPA		
datos y direcciones:	habilitados	deshabilitados
	RAM 2	RAM 1
Para la PC		
datos y direcciones:	habilitados	deshabilitados
	RAM 1	RAM 2

Figura 3.14 Control de conmutación a partir de la señal LC.

Para evitar que los dos microprocesadores, el de la PC y el MPA, tengan acceso simultáneo a la misma memoria, con la línea ($\overline{LC1}$) se conmutan las señales de escritura, lectura y el habilitador de cada memoria por medio de multiplexores. Las primeras se seleccionan mediante U27, y las segundas mediante U26 (figura 3.15).

La tabla 3.4 muestra la tabla de verdad para el multiplexor U27, para el cual la línea de selección es ($\overline{LC1}$), y las entradas de datos son $\overline{RD}$, $\overline{WR}$ (señales activadas por el MPA), $\overline{MEMR}$ y $\overline{MEMW}$ (señales habilitadas por la PC).

Cabe hacer notar que en un ciclo de lectura o escritura, las señales $\overline{RD}$ y $\overline{WR}$ se encontrarán siempre en estado lógico contrario, al igual que $\overline{MEMR}$ y $\overline{MEMW}$, siendo independientes los ciclos del MPA con los de la PC.

De la tabla de verdad para el multiplexor U26, tabla 3.5, se observa que las memorias RAM1 ó RAM2 se pueden habilitar de las siguientes dos maneras:

a) Mediante las señales $\overline{RAM1}$ ó $\overline{RAM2}$, provenientes del decodificador U25.

b) Mediante la línea $\overline{RAM(DXXXX)}$ controlada por la PC.

La conexión de la conmutación de buses para la RAM1 y RAM2, así como los circuitos de las memorias mismas, se muestra en las figuras 3.16 y 3.17 respectivamente.

En el diagrama de la RAM 1 se muestran 4 buffers para direcciones y 2 para datos, de los cuales son asignados al MPA U8,U9 (direcciones) y U19 (datos), mientras que a la PC, U13, U12 (direcciones) y U20 (datos).

La lógica para la conmutación y el acceso de los buses de datos del MPA y PC a las memorias RAM1 ó RAM2, se realiza mediante U31C, U31D y U24A, U24D, respectivamente.

De manera análoga para el diagrama de la RAM2 son asignados al MPA U17, U18 (direcciones) y U10 (datos), mientras que a la PC, U21, U22 (direcciones) y U11 (datos).

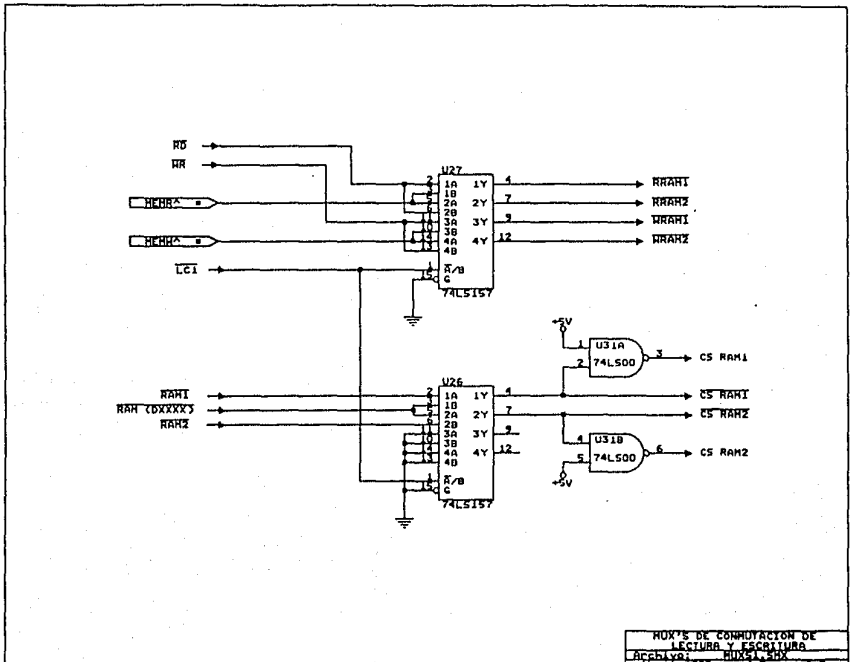


Figura 3.15 Conexión de los multiplexores de conmutación del MPA.

$\overline{LC}$	$\overline{RD}$	$\overline{MEMR}$	$\overline{WR}$	$\overline{MEMW}$	SEÑAL ACTIVA (L)	TERMINAL HABILITADA
L	L	*	H	*	$\overline{RRAM1}$	$\overline{OE}$ de RAM1
L	*	L	*	H	$\overline{RRAM2}$	$\overline{OE}$ de RAM2
L	H	*	L	*	$\overline{WRAM1}$	$\overline{WE}$ de RAM1
L	*	H	*	L	$\overline{WRAM2}$	$\overline{WE}$ de RAM2
H	L	*	H	*	$\overline{RRAM2}$	$\overline{OE}$ de RAM2
H	*	L	*	H	$\overline{RRAM1}$	$\overline{OE}$ de RAM1
H	H	*	L	*	$\overline{WRAM2}$	$\overline{WE}$ de RAM2
H	*	H	*	L	$\overline{WRAM1}$	$\overline{WE}$ de RAM1

* Cualquier estado.

Tabla 3.4 Tabla de verdad del multiplexor U27.

$\overline{LC}$	$\overline{RAM1}$	RAM(DXXXX)	$\overline{RAM2}$	SEÑAL ACTIVA (L)	TERMINAL HABILITADA
L	L	*	*	$\overline{CSRAM1}$	Habilita terminal $\overline{CE}$ de RAM1
L	*	L	*	$\overline{CSRAM2}$	Habilita terminal $\overline{CE}$ de RAM2
H	*	L	*	$\overline{CSRAM1}$	Habilita terminal $\overline{CE}$ de RAM1
H	*	*	L	$\overline{CSRAM2}$	Habilita terminal $\overline{CE}$ de RAM2

* Cualquier estado.

Tabla 3.5 Tabla de verdad del multiplexor U26.

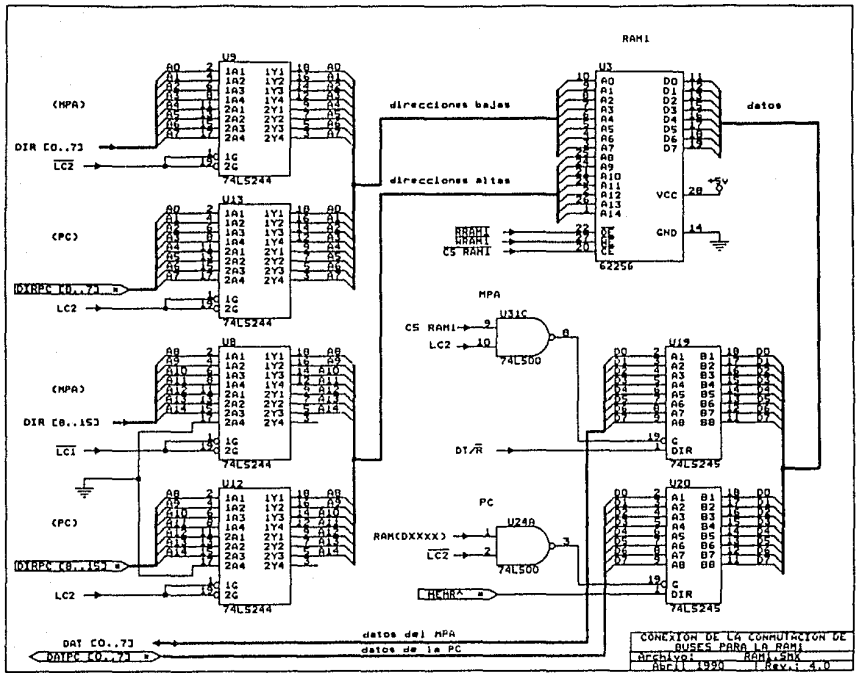


Figura 3.16 Conexión de RAM1. Bus de datos y direcciones.

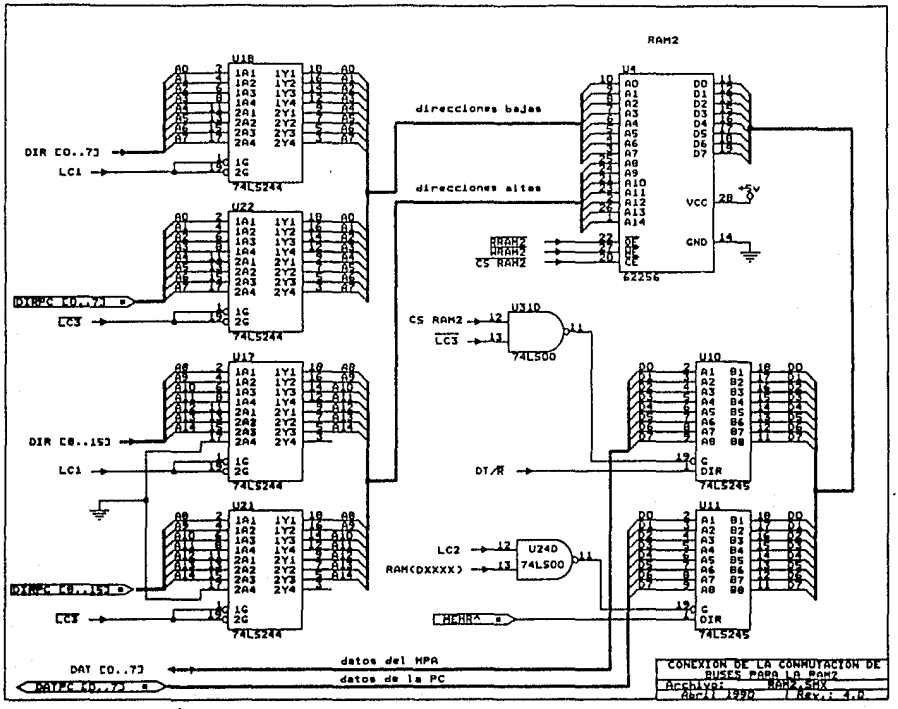


Figura 3.17 Conexión de RAM2. Bus de datos y direcciones.

3.4.5 Conversor analógico-digital.

Como ya se explicó, el sistema fué diseñado para procesar **16** canales, 12 para las estaciones que se encuentran operando actualmente y 4 mas para una futura expansión. Por lo anterior se seleccionó un conversor A/D de 16 canales analógicos multiplexados, uno por cada estación.

El circuito integrado seleccionado fué el conversor **ADC0816** de National, de 8 bits.

Entre sus carterísticas más importantes están:

- **Multiplexor analógico integrado para 16 canales de entrada.**
- **Palabras de 8 bits.**
- **Error total no ajustable (+-) 1/2 bit.**
- **Tiempo de conversión 100 microseg.**
- **Registro de salida de tres estados.**
- **Compatible con TTL y CMOS.**
- **Niveles de entrada de 0 a 5 volts.**
- **Voltaje de alimentación 5 VDC.**

El circuito completo del conversor A/D diseñado se muestra en la figura 3.18.

La señal de reloj de **400 kHz** se derivó de un oscilador de cristal de **4 MHz** (U34) y un divisor entre 10 (U35).

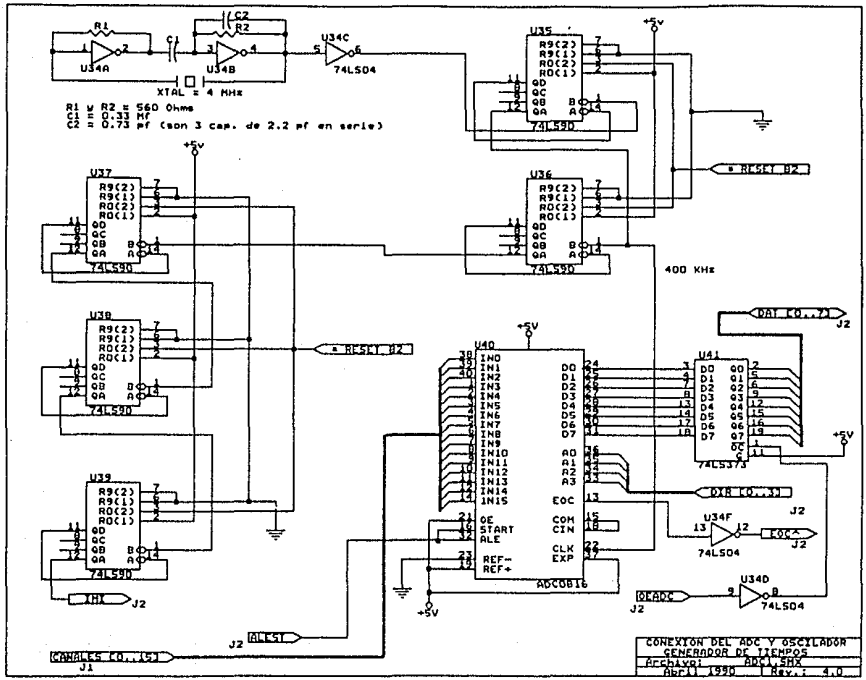


Figura 3.18 Conexión del ADC y oscilador generador de tiempos.

El modo de operación del conversor es el siguiente :

Las señales de los 16 canales analógicos van directamente a las entradas del conversor (U40). El canal a convertirse se selecciona mediante las direcciones bajas DIR[0..3] del MPA.

Tanto la selección del canal como el inicio de un ciclo de conversión se efectúa a partir de la señal ALEST, generada a través del circuito de la figura 3.19 con las señales $\overline{WR}$ y $\overline{ADC}$.

El flanco de subida de ALEST carga la dirección del canal y el flanco de bajada inicia la conversión.

Durante el proceso la señal $\overline{EOC}$ de fin de conversión, normalmente alta, pasa a un estado bajo. Esta señal se invierte con U34F y se manda a la entrada TEST del microprocesador. Con ello se mantiene al CPU en estado de espera (WAIT) durante todo el tiempo que la señal EOC está en estado bajo.

Una vez terminada la conversión, el procesador genera la señal OEADC a partir de la señal de lectura $\overline{RD}$ y de selección $\overline{ADC}$ con el circuito de la figura 3.19. Con el flanco negativo de la señal OEADC invertida por U34D, se habilita el latch U41 para enviar al bus de datos el dato convertido.

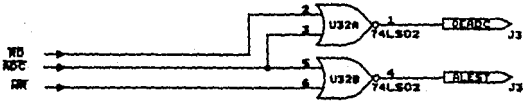
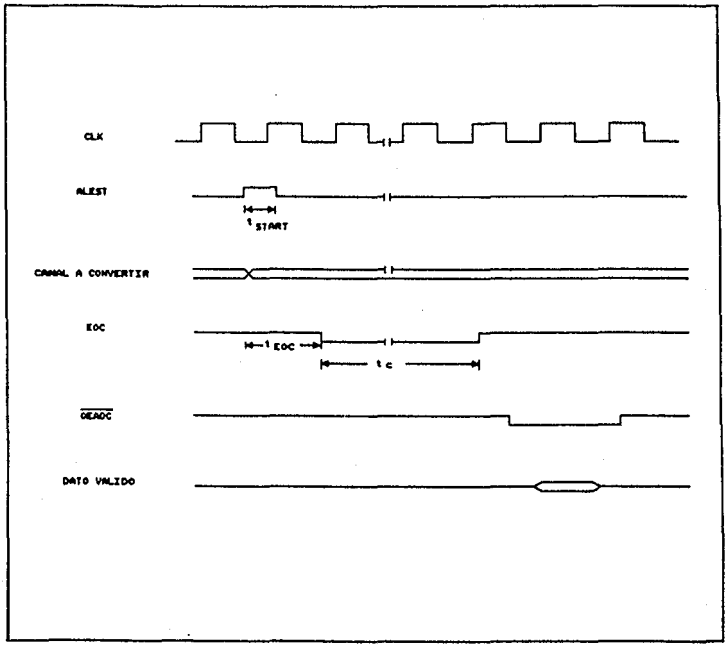


Figura 3.19 Codificación de las señales de conversión en el ADC.

En la figura 3.20, se presenta el diagrama de tiempos de un ciclo de conversión.

Como se puede ver en la figura, existe un pequeño retraso entre la señal de inicio de conversión (ALEST) y de fin de conversión (EOC), debido al tiempo de respuesta del ADC.

Asimismo se puede apreciar que la señal OEADC es activada en el momento en el que el MPA desee leer el dato convertido; lo cual ocurre cuando la señal de fin de conversión EOC toma el nivel alto, indicando que el dato está listo en el buffer de salida.



Donde:

tEOC = 10.8 μseg (retardo)
tc = 160 μseg
tSTART = 0.3 μseg

Figura 3.20 Diagrama de tiempos en la conversión de un dato en un canal.

3.4.6 Acondicionamiento de las señales analógicas de SISMEX.

· Amplificación y filtraje.

Las señales analógicas provenientes de SISMEX tienen un nivel (+-) 1 volt. Sin embargo el conversor A/D requiere que las señales de entrada estén en el rango de 0 a 5 volts. Por tal razón fué necesario acondicionar la señal de cada canal mediante un circuito que se muestra en la figura 3.21.

Consta de un amplificador operacional U45 para acoplar los dos circuitos mediante este dispositivo de alta impedancia. Posteriormente la señal se limpia de altas frecuencias mediante un filtro paso bajas con corte a 15 Hz, U46.

La señal bipolar a la salida del filtro finalmente se amplifica con U47 y se le suma 2.5 VDC para obtener una señal de salida de 0 a 5 volts con su punto central a 2.5 volts. El ajuste del nivel de DC se hace mediante P1 y con P2 se da el valor de amplificación deseado.

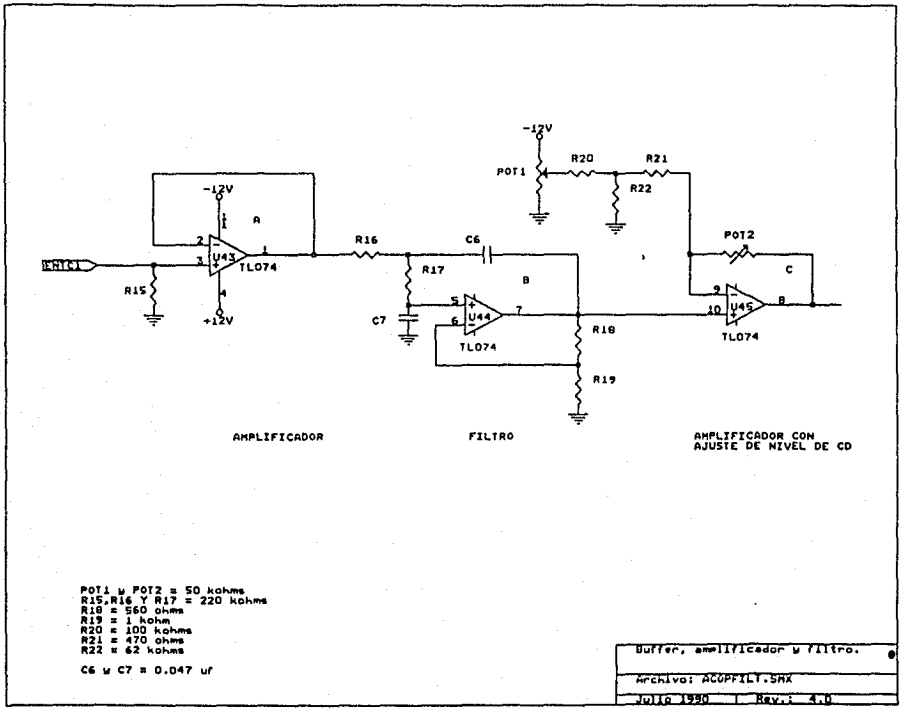


Figura 3.21 Acondicionamiento de la señal analógica de SISMEX.

IV. INTERCOMUNICACION DEL MPA Y LA PC

Debido a que el sistema de adquisición de datos está gobernado por un microprocesador, el cual es completamente independiente del microprocesador de la PC, fué necesario establecer un protocolo de comunicación entre ambos sistemas.

Se recordará del capítulo anterior que las memorias conmutables son el único medio de comunicación entre la PC y el sistema de adquisición, por lo que se añadió un puerto de control (PPC), a través del cual la PC, como maestro, puede mandar códigos de control al microprocesador de adquisición para especificar ciertas funciones a realizar (figura 3.13). Con la inclusión de este puerto se logra la subordinación del sistema de adquisición a la PC.

El diagrama general de flujo del programa del MPA se presenta en la figura 4.1.

En un principio, al encender el sistema, el MPA debe esperar a que la PC termine su proceso de inicialización ("boot"), tiempo que utilizará para borrar las memorias de conmutación y pasar luego a un estado de espera de recepción de un código de inicio (CI) de la PC. Este código se da a través de la localidad 00H de RAM1 ó RAM2.

Cuando la PC termina la inicialización escribe un código diferente de 00H en esa localidad.

Inicializado el sistema el MPA espera a que la PC le mande un código de control (CC) a través del PPC con la tarea ó función específica a realizar.

Al recibir el código de control, el MPA deberá contestar a la PC que ya lo recibió y entendió. Lo hace mediante un código de respuesta (CR) que lo manda a través de la localidad 0001H de las memorias de conmutación (único medio para que el sistema de adquisición puede enviar datos a la PC).

A su vez la PC, al recibir el CR y con el fin de sincronizarse con el MPA, limpia el PPC con un código CC=00H. Este comando, para el MPA, es el inicio de ejecución de la tarea ordenada por la PC. El MPA entra

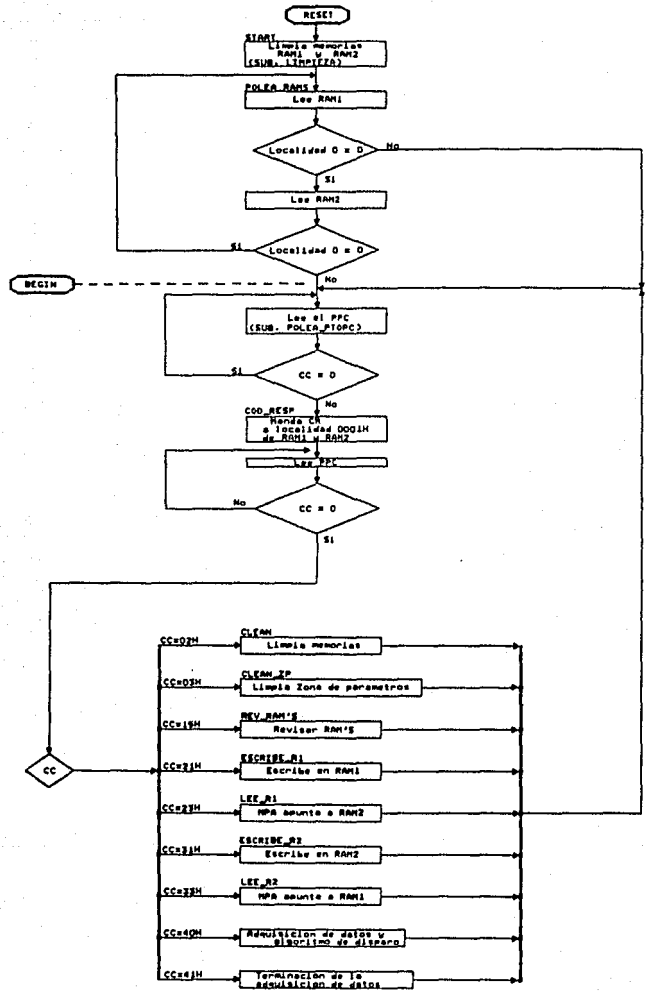


Figura 4.1 Diagrama de flujo del programa del MPA y menú de tareas.

entonces a una tabla para reconocer y ejecutar la tarea correspondiente al código recibido.

Una vez ejecutada la tarea, el MPA indica a la PC la terminación colocando en la localidad 02H de RAM1 ó RAM2 un código de terminación (CT).

Al principio de cada una de las memorias de conmutación RAM1 y RAM2, se estableció una zona de 128 bytes para comunicación entre ambos procesadores. En ella se almacenan los códigos de control, de respuesta y de terminación, así como datos y parámetros para comunicación entre el MPA y la PC.

En la tabla 4.1 se describen las 128 localidades de la zona de parámetros de RAM1 y RAM2.

LOCALIDAD (Hexadecimal)	NOMBRE CODIGO	DESCRIPCION
00	CI	Código de inicialización. Reservado para inicio, después del boot de la PC.
01	CR	Código de Respuesta. Reservado para que el MPA conteste a la PC que recibió el CC.
02	CT	Código de Terminación.
03	LR1	Localidad número 1 para datos de respuesta del MPA.
04	LR2	Localidad número 2 para datos de respuesta del MPA.
.	.	.
.	.	.
.	.	.
13	LR16	Localidad número 16 para datos de respuesta del MPA.
14	LD1	Localidad de la variable 1 para ejecución de rutinas del MPA.
15	LD2	Localidad de la variable 2 para ejecución de rutinas del MPA.
.	.	.
.	.	.
.	.	.
17	LD4	Localidad de la variable 4 para ejecución de rutinas del MPA.
18-7F		Otros datos.

Tabla 4.1 Primeras 128 localidades de las zonas de parámetros de RAM1 y RAM2.

Los códigos más importantes del proceso descrito son:

a) Código de control (CC)

Código enviado por la PC al MPA a través del puerto PPC con la tarea a procesar. Los principales códigos y tareas definidas para el sistema se presentan en la tabla 4.2.

b) Código de Inicio (CI)

Esta localidad únicamente cumplirá la función de transmitir al MPA el código de inicio después de terminada la inicialización de la PC que tarda aproximadamente 7 segundos. La localidad asignada para este código, tanto en la RAM1 como en la RAM2, es la 0000H.

Los valores de CI pueden ser:

00	**El MPA se mantiene revisando la localidad 00H de RAM1 Y RAM2. No efectúa otra función.**
$\neq$ cero	**El MPA pasa a esperar un código de control de la PC en el puerto PPC.**

c) Código de Respuesta (CR)

Este código lo transmitirá el MPA a través de la localidad 0001H y con el indicará a la PC que el código de control que le fué enviado, lo entendió y está listo para ejecutar la rutina.

El valor del CR será el mismo que el código de control que le fué enviado por el PPC.

d) Código de Terminación (CT)

El Código de terminación es una señal del MPA enviado a la PC con el que le indica que ha terminado la tarea encomendada. Utiliza para ello la localidad 02H.

Los valores de CT son :

A0	Término de ejecución, sin datos de respuesta.
A1	Término de ejecución, con 1 dato de respuesta en la localidad 03H.
A2	Termino de ejecución, con 2 datos de respuesta en la localidad 03H y 04H.
A3	Termino de ejecución, con 3 datos de respuesta en la localidad 03H, 04H y 05H.
.	.
.	.
.	.
AF	Término de ejecución, con datos en las localidades 03H,04H.....y 13H.

CODIGO (Hex)	TAREA
01	Inicio del reconocimiento del PPC después del boot de la PC.
02	Puesta a cero de todas las localidades de RAM1 y RAM2.
03	Puesta a cero de todas las localidades de la zona de parámetros.
15	Prueba de lectura-escritura de las memorias RAM1 y RAM2.
21	Escritura en RAM1 de un caracter determinado número de veces.
23	Posiciona al MPA hacia RAM2.
31	Escritura en RAM2 de un caracter determinado número de veces.
33	Posiciona al MPA hacia RAM1.
40	Adquisición continua de datos de 16 canales a 40 muestras/seg, incluyendo ejecución del algoritmo de detección.
41	Suspende la adquisición de datos y espera un código de control $\neq 0$.
45	Adquisición de datos de 16 canales hasta llenar una memoria RAM1 ó RAM2.

Tabla 4.2 Códigos de control (CC) y sus funciones.

V. PROGRAMACION DEL MICROPROCESADOR DE ADQUISICION

En este capítulo se dará una breve descripción de las rutinas que ejecuta el MPA. Los diagramas de flujo correspondientes a cada una de ellas se presentan en el desarrollo de los algoritmos utilizados. Los listados completos de los programas a nivel ensamblador se presentan en el anexo C.

Básicamente, se tienen dos tipos de rutinas: Las primeras son rutinas de servicio, enfocadas a pruebas y diagnósticos de fallas del buffer de conmutación. El otro tipo de rutinas son de adquisición de datos, incluyendo la ejecución del algoritmo de detección.

A continuación se describen las rutinas hasta ahora diseñadas. El número de la rutina corresponde al código de comando CC.

5.1 Rutinas de servicio.

Rutina 02 (CC=02H)

Esta rutina borra las dos memorias de conmutación de 32 Kbytes cada una.

El código de terminación de esta rutina es CT=A0H. Su diagrama de flujo se muestra en la figura 5.1.

Rutina 03 (CC=03H)

Efectúa el borrado únicamente de la zona de parámetros de las dos memorias de conmutación. Su código de terminación también es CT=A0H.

El diagrama de flujo de esta rutina se da en la figura 5.2.

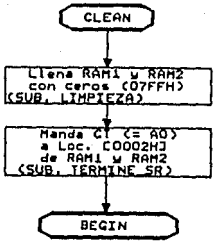


Figura 5.1 **Diagrama de flujo de la rutina 02H: borrado de las memorias de conmutación.**

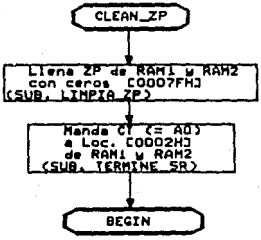


Figura 5.2 **Diagrama de flujo de la rutina 03H: borrado de la zona de parámetros en las memorias de conmutación.**

Rutina 15 (CC=15H)

Esta rutina se encarga de verificar que ambas memorias RAM1 y RAM2 operen correctamente.

Se basa en escribir y leer un número hexadecimal en cada localidad de RAM1. Al terminar los 32 Kbytes continúa con la revisión de RAM2. El código de terminación de esta rutina es A2H.

En la localidad [0003] se almacena el resultado de la revisión de RAM1 y en [0004] la de RAM2.

Los distintos resultados pueden ser:

Resultado	Descripción
1F	Memoria RAM1 correcta.
1B	Memoria RAM1 con problemas en alguna localidad.
2F	Memoria RAM2 correcta.
2B	Memoria RAM2 con problemas en alguna localidad.

El diagrama de flujo de esta rutina se muestra en las figuras 5.3a y 5.3b.

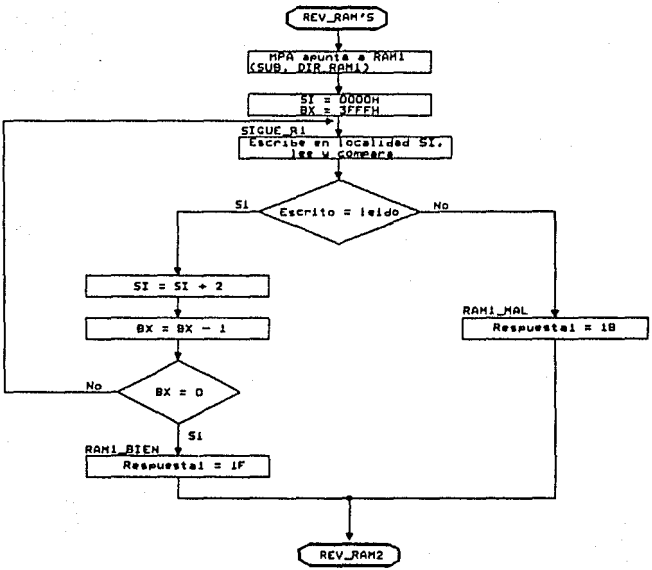


Figura 5.3a Diagrama de flujo de la rutina 15H: revisión de memorias RAM1 y RAM2.

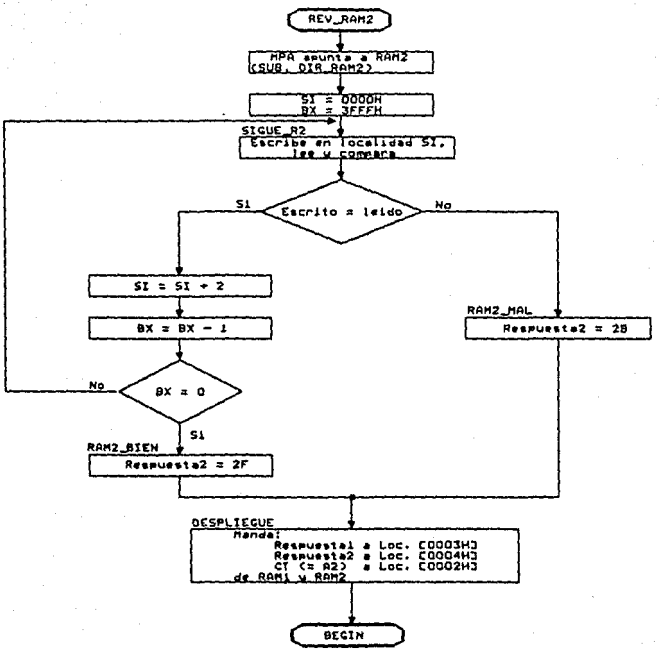


Figura 5.3b Diagrama de flujo de la rutina 15H: revisión de memorias RAM1 y RAM2 (continuación).

Rutina 21 (CC=21H)

Esta rutina escribe específicamente en la RAM1 un determinado caracter un número de veces.

Los tres parámetros, el número hexadecimal a escribir, el número de veces y la localidad de inicio, son definidos por el usuario y enviados desde la PC a las localidades [0020], [0022], y [0024] de RAM1 respectivamente.

El código de terminación de esta rutina es AOH.

El diagrama de flujo correspondiente se muestra en la figura 5.4.

Rutina 31 (CC=31H)

Esta rutina realiza la misma tarea que la rutina 21, pero almacenando el dato en RAM2.

De igual forma los tres parámetros serán tomados de las localidades [0020], [0022] y [0024]. El código de terminación también es AOH.

El diagrama de flujo se muestra en la figura 5.5.

Rutina 23 (CC=23H)

Esta rutina permite a la PC desplegar en pantalla el contenido de las localidades de RAM1. El MPA solamente apunta hacia RAM2, permitiendo a la PC acceso a RAM1.

Su código de terminación es AOH; su diagrama correspondiente se muestra en la figura 5.6.

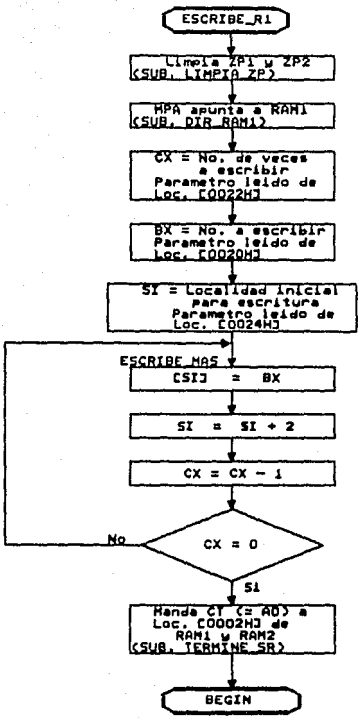


Figura 5.4 Diagrama de flujo de la rutina 21H: escritura a RAM1.

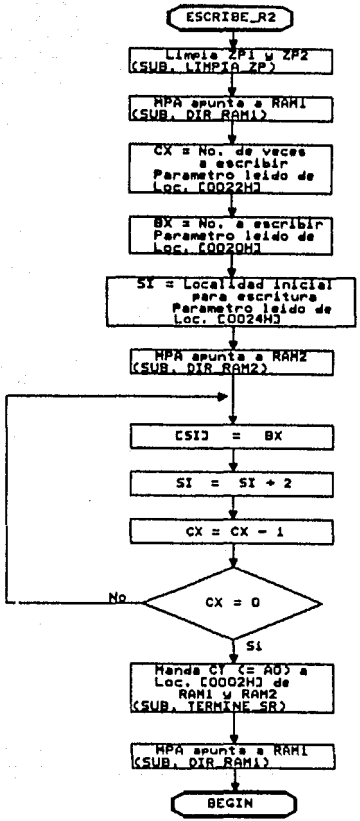


Figura 5.5 **Diagrama de flujo de la rutina 31H: escritura a RAM2.**

Rutina 33 CC=33H)

De forma similar a la rutina 23, el MPA apunta hacia RAM1, para que la PC tenga acceso a RAM2, y pueda visualizar su contenido.

Su código de terminación será de igual forma A0H; su diagrama se muestra en la figura 5.7.

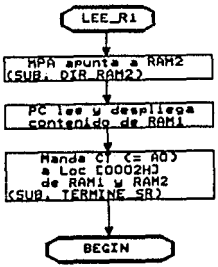


Figura 5.6 Diagrama de flujo de la rutina 23H: conmuta MPA hacia RAM2.

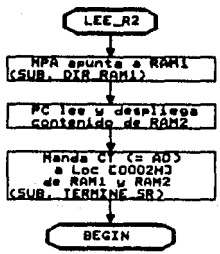


Figura 5.7 Diagrama de flujo de la rutina 33H: conmuta MPA hacia RAM1.

5.2 Rutinas de adquisición de datos.

Rutina 40 (CC=40H)

Esta rutina efectúa cada 25 milisegundos la adquisición y conversión analógica/digital de los 16 canales en forma consecutiva. Simultáneamente procesa los datos de acuerdo con el algoritmo de disparo.

Cada muestra convertida de 8 bits es un byte. Los 16 bytes correspondientes a los 16 canales convertidos se almacenan inicialmente en la RAM1. Al llenarse ésta se conmuta a RAM2 y se continúa el almacenamiento en RAM2. Así sucesivamente se llenan alternadamente ambas memorias.

El número de muestras (de 16 canales) que se almacenan en cada memoria de 32 Kbytes es:

$$\frac{32768(\text{bytes}) - 12(\text{bytes, zona de parámetros})}{16\ (\text{bytes/muestra})} = 2040 \text{ muestras}$$

El diagrama de flujo del proceso de conversión se muestra en la figura 5.8.

Rutina 41 (CC=41H)

La utilización de esta rutina es para finalizar el proceso de adquisición de datos, inicializar apuntadores y esperar alguna nueva tarea asignada por la PC.

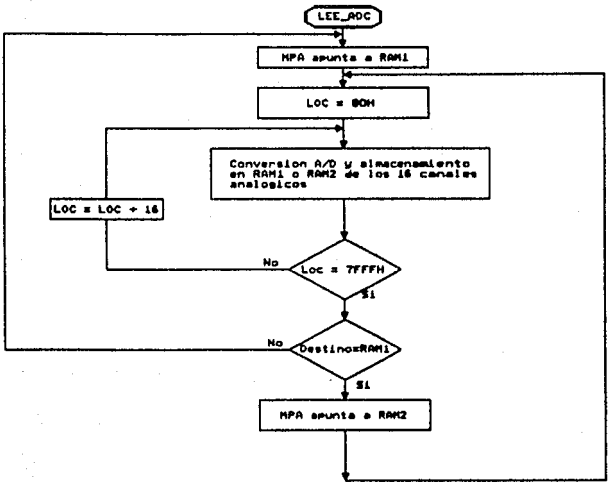


Figura 5.8 Diagrama de flujo de la rutina 40H: adquisición de datos.

5.3 Algoritmo de disparo.

Para poder determinar si la información recibida por el sistema corresponde a un sismo de determinada intensidad y proceder a registrar los datos en disco, se desarrolló un sencillo algoritmo de disparo. Se basa en los siguientes pasos:

- Rectificación de las señales bipolares para obtener sus valores absolutos.
- Calcular el promedio de 64 muestras y comparar este valor con el umbral escogido.
- En caso de rebasar la señal el umbral, pasar al estado de disparo y generar una interrupción a la PC para registrar los datos en disco.

Este proceso se lleva a cabo para cada uno de los 16 canales. El diagrama de flujo del algoritmo se muestra en la figura 5.9.

5.3.1 Rectificación.

El método de rectificación de los datos digitales se realiza de la siguiente manera:

El conversor toma una muestra mi que puede variar entre 00H y FFH:

si mi >= 80H entonces mi = mi-80H

si mi < 80H entonces mi = 80H-mi

5.3.2 Comparación del promedio de 64 muestras con un umbral.

Para sacar el promedio de los datos adquiridos se efectúa la sumatoria de 64 muestras m(i) consecutivas, dividiendo el resultado entre 64; esto para cada canal.

De esta forma el promedio P(j) por canal es:

$$P(j) = \frac{\sum_{i=0}^{64} m(i)}{64} \quad ; j=1,2....16$$

Una vez obtenido el promedio de cada canal, se compara con el umbral definido por el usuario a través de la PC.

si:

P(j) >=	Umbral entonces	Sistema en disparo.
P(j) <	Umbral entonces	Inicializa el cálculo de un nuevo promedio P(j).

5.3.3 Estado de disparo.

En el instante en que el sistema detecta un sismo, entra en estado de disparo. El MPA interrumpe a la PC y le transfiere información de los apuntadores de inicio y la RAM en la que el evento comenzó. La PC activará entonces una alarma auditiva (beeps) indicando la detección del sismo. El MPA mientras tanto continúa la adquisición de datos.

Al momento de dispararse el sistema se inicia un conteo de 20 segundos. Cada vez que la señal rebasa el umbral este contador se pone nuevamente a cero, es decir el sistema se redispara. Si transcurridos 20 segundos la señal no alcanza sobrepasar el nivel de disparo, se asume que el sismo ha terminado. La PC recibirá entonces los apuntadores finales de los datos almacenados en la memoria de conmutación para que pueda concluir el registro del evento.

La información de la RAM y de los apuntadores que el MPA transfiere a la PC a través de la zona de parámetros se registra de la siguiente manera:

Localidad	Parámetro.
21H	Apuntador inicial (parte baja).
22H	Apuntador inicial (parte alta).
24H	RAM donde inició el sismo.
28H	Canal donde se registro el disparo.
40H	Apuntador final (parte baja).
41H	Apuntador final (parte alta).
42H	RAM donde finalizó el sismo.

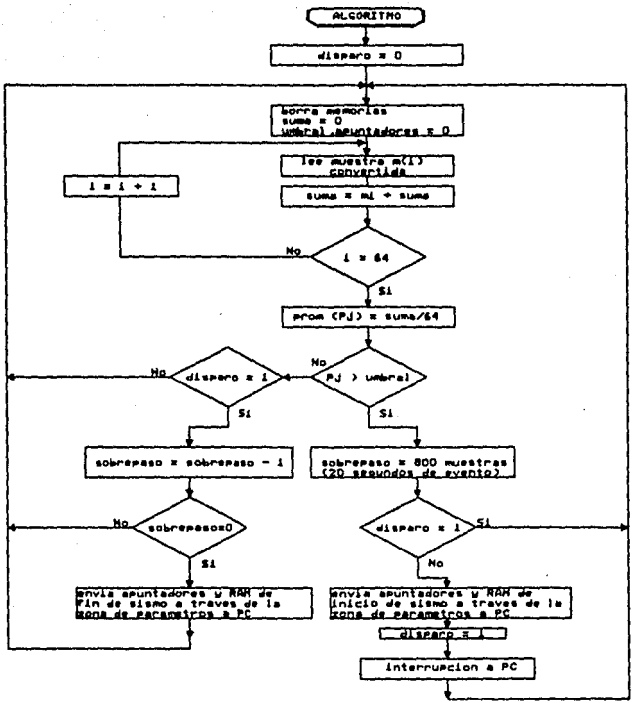


Figura 5.9 Diagrama de flujo del algoritmo de disparo para un canal j.

VI. PROGRAMACION DE LA PC

Como se recordará de los capítulos anteriores, a la computadora PC se le designó maestro y al procesador de adquisición esclavo. Esta jerarquía se estableció a través de la programación y control del sistema de adquisición.

La computadora PC ejecuta un programa principal mediante el cual interactúa con el usuario, controla la operación del procesador de adquisición de datos, se encarga del almacenamiento en disco de la información y posteriormente de su procesamiento.

La programación de la PC se realizó en lenguaje PASCAL V.5. Se eligió por las ventajas que brinda en cuanto al manejo de gráficos e interacción con el hardware de PC. Cabe mencionar que sólo aquellas partes que necesitaban interactuar de una forma más directa con el hardware del sistema adquisidor, se escribieron en lenguaje ensamblador 8088, como fué el caso de la rutina de interrupción.

En este capítulo se describen los programas y funciones que realiza la PC: la comunicación con el usuario en base a una estructura de menús, el procesamiento de datos, como graficación y desplegado, el control del adquisidor y las tareas de almacenamiento a través de la rutina de interrupción.

Para ejecutar el programa se invoca con el nombre de "ADQUISIDOR" desde el sistema operativo. Un archivo de comandos "BATCH" se encarga de ejecutar al programa ADQUISIDOR.EXE cuyo listado fuente en lenguaje PASCAL se encuentra en el archivo ADQUISIDOR.PAS.

El listado de los programas y algunos diagramas de flujo de ciertas rutinas se presentan en los anexos V y VI.

6.1 Descripción general del programa.

El programa está compuesto fundamentalmente de tres partes (figura 6.1): inicialización (A), programa principal (B) y rutina de interrupción (C).

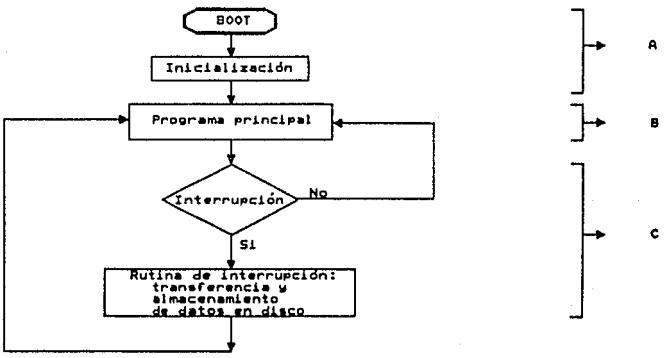


Figura 6.1 Esquema general de la programación.

A) Inicialización.

La inicialización empieza desde que se enciende la PC y se efectúa el "boot". Al terminar esta rutina comienza a inicializar al MPA a través del puerto PPC y prepara también la rutina de interrupción.

B) Programa principal.

Está estructurado con base en menús, iniciando con un menú principal. Las cinco primeras opciones del menú principal son rutinas que ejecuta escencialmente el

MPA, recibiendo la orden correspondiente de la PC, mientras que las tres últimas son independientes del adquisidor.

C) Rutina de interrupción.

La interrupción de la PC se efectúa cuando el procesador de adquisición detecta un disparo en el sistema, es decir, detecta un movimiento sísmico que satisface el algoritmo de disparo. En ese momento la PC transfiere los datos de RAM1 y/o RAM2 del sistema adquisidor a su RAM, para después almacenarlos en disco duro.

En la figura 6.2 se presenta el diagrama de flujo que describe las partes A y B del programa general. La parte C se explicará más adelante en el inciso 6.3.2.

En la ejecución de las opciones 1 a 5 del menú principal la PC envía un código de comando al MPA por el Puerto PPC (localidad E000:0000), indicándole qué opción desea realizar el usuario. Estos códigos son:

Código de Comando	Función que realiza el MPA.
2	Borrado total de RAMS.
3	Borrado de zona de parámetros.
15	Revisión de RAMS.
21	Escritura en RAM1.
31	Escritura en RAM2.
23	Lectura de RAM1.
33	Lectura de RAM2.
40	Inicia adquisición de datos.
41	Termina adquisición de datos.

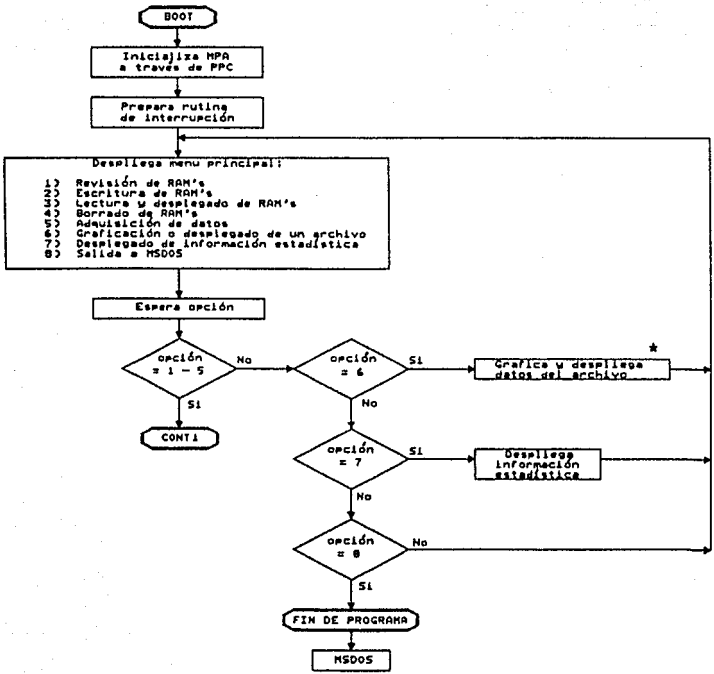
Tabla 6.1 Códigos de comando.

La PC espera entonces el código de respuesta (CR) que le envía el MPA a través de la RAM de conmutación (localidad D000:0001). Se realiza luego un "poleo" hasta que el

código de comando sea igual al código de respuesta recibido. De no ser así, la PC envía un mensaje de error. Sin embargo, si el código de comando es igual al de respuesta, la PC ordena al MPA que puede realizar la función indicada, enviándole por el PPC un valor de cero. Después de concluída la tarea la PC recibe el código de terminación (localidad D000:0002 de la RAM).

Con el código de terminación y el de respuesta, la PC indica mediante mensajes desplegados en pantalla, que la opción solicitada por el usuario ha sido realizada y regresa nuevamente al menú principal.

Las opciones 6 y 7 son realizadas únicamente por la PC y se describirán más adelante.



* Explicación detallada en el anexo V (figura 1).

Figura 6.2 Diagrama de flujo general de las partes A y B.

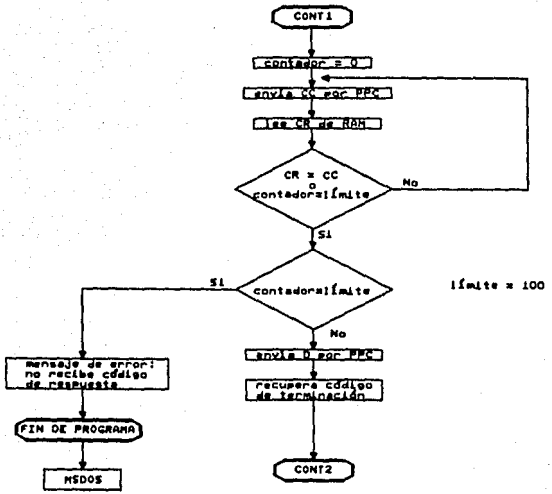


Figura 6.2 Diagrama de flujo general de las partes A y B (continuación).

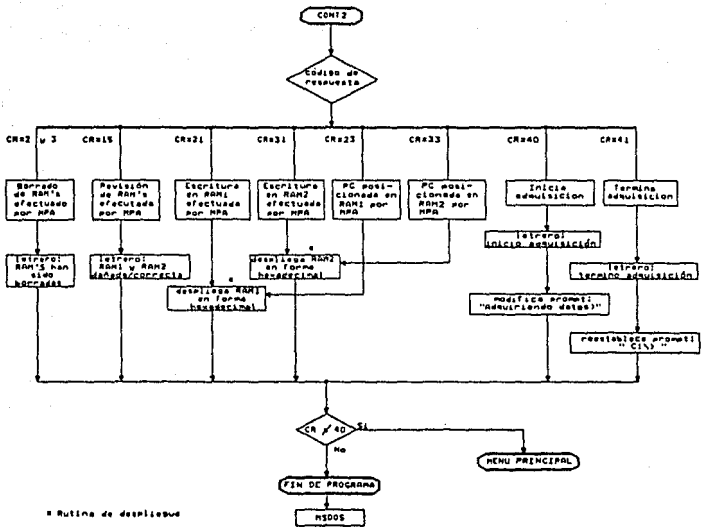


Figura 6.2 Diagrama de flujo general de las partes A y B (continuación).

6.2 Menú principal.

En la figura 6.3 se muestra el menú principal a través del cual se ejecutan las distintas tareas hasta la fecha desarrolladas.

La mayoría corresponden a rutinas de prueba y verificación de las memorias de conmutación. Se escribieron con la finalidad de contar con utilerías durante la fase de diseño del sistema que permitieran verificar la correcta operación de las RAMS y la comunicación e iteracción del MPA y la PC.

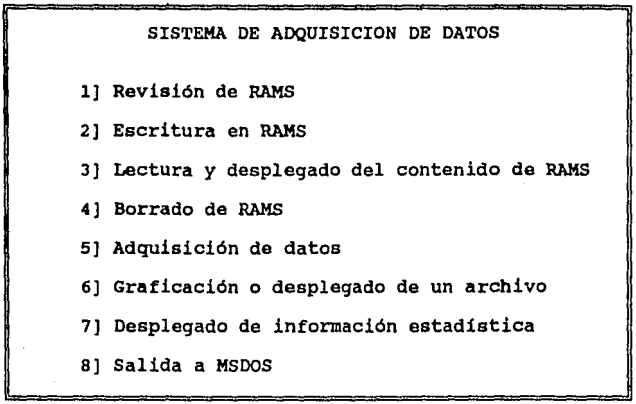


Figura 6.3 Menú principal de operación.

Una vez ejecutada la tarea seleccionada, el programa regresa al menú principal, excepto en la opción 5 que al terminar, lleva al sistema al MSDOS.

A continuación se describe detalladamente cada una de las opciones del menú principal:

6.2.1 Revisión de RAMS.

Con esta opción la PC revisa el correcto funcionamiento de las memorias de conmutación RAM1 y RAM2 de la tarjeta de adquisición. Ejecutada la tarea se despliega el resultado de la revisión con el siguiente mensaje:

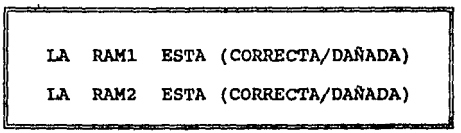


Figura 6.4 Desplegado del resultado de revisión de memorias.

6.2.2 Escritura en RAMS.

Con esta opción aparece un submenú para que el usuario escoja la memoria a la cual desea escribir (figura 6.5). El operador deberá entonces definir el dato que desea escribir, el número de veces y a partir de qué localidad.

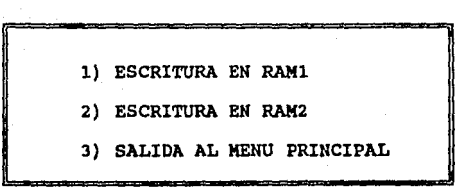


Figura 6.5 Menú de escritura en memorias.

Ejecutada la tarea la PC despliega los 128 bytes de la zona de parámetros de la RAM a la cual escribió, como se muestra en la figura 6.6.

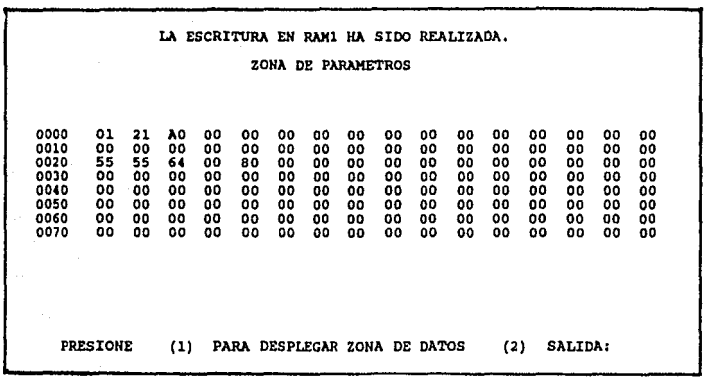


Figura 6.6 Desplegado de la zona de parámetros.

Como se puede observar, se presenta otro submenú, con las opciones:

1. **Desplegar la zona de datos de la RAM.**
2. **Salir al menú principal.**

Aunque físicamente la zona de datos de la RAM inicia en la localidad **80H**, para el usuario ésta se considera como la **00H**.

El desplegado de la zona de datos se realiza por pantallas, cada una despliega **256 bytes**. La primer pantalla se tiene desde la localidad 0000H hasta la 00FFH, la segunda, desde la localidad 0100H a la 01FFH y así sucesivamente, hasta llegar a la localidad 7F7FH (32 Kbytes de memoria en RAM menos 128 bytes de la zona de parámetros).

El usuario tiene la opción de elegir una localidad de memoria específica en cualquier momento y poder desplegar la pantalla correspondiente.

En la figura 6.7 se muestra un ejemplo de la forma como se despliega la información.

```
                    LA ESCRITURA EN RAM1 HA SIDO REALIZADA.


 0000  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0010  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0020  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0030  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0040  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0050  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0060  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0070  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0080  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 0090  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 00A0  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 00B0  55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
 00C0  55 55 55 55 55 55 55 55 55 00 00 00 00 00 00 00
 00D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

PRESIONE   (1) SEGUIR EN ZONA DATOS    (2) ZONA PARAMETROS    (3) SALIDA :
```

Figura 6.7 Desplegado de información.

El diagrama de flujo que explica detalladamente la rutina de desplegado de datos se presenta en el ANEXO V.

6.2.3 Lectura y despliegue de las RAMS.

Esta tarea inicia presentando un submenú a través del cual el usuario determina qué RAM desea desplegar (figura 6.8) y decidir a partir de qué localidad se desean visualizar los datos.

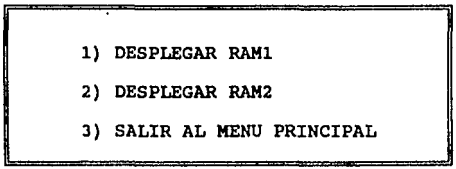


Figura 6.8 Menú de despliegue de memorias.

El desplegado se realiza de igual forma que en la opción 1 y 2 del inciso anterior.

6.2.4 Borrado de las RAMS.

Esta subrutina llena de ceros todas las localidades de memoria RAM1 y RAM2.

La opción inicia con un submenú como se muestra en la figura 6.9.

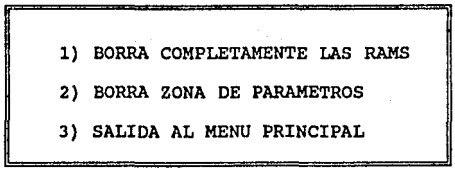


Figura 6.9 Menú para borrado de memorias.

Cuando la tarea ha sido terminada, aparece un mensaje indicando su finalización.

6.2.5 Adquisición de datos.

Con esta opción se dan al MPA los comandos para el proceso de adquisición de datos (figura 6.10). Como su nombre lo indica, esta es la rutina principal del sistema y la que normalmente se procesará.

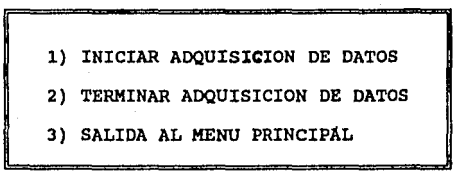


Figura 6.10 Menú de adquisición.

Si se escoge la opción de inicio de adquisición de datos, el programa pide al usuario el valor del umbral de disparo deseado, necesario para el algoritmo de detección de un evento sísmico .

Una vez que la PC ordena al MPA el inicio de la adquisición de datos, sale del programa principal y regresa al MSDOS. Para indicar al usuario de que la adquisición está en proceso, el prompt "C:/>" usual del sistema operativo aparecerá ahora como "ADQUIRIENDO DATOS>".

Cuando el sistema se encuentra en adquisición de datos y se invoca nuevamente al programa, sólo se pueden ejecutar las opciones 5,6,7 y 8 del menú principal. Dado que la PC se encuentra en MSDOS, se pueden ejecutar cualquier programa deseado o comandos del sistema operativo. Sin embargo, al momento de detectarse un sismo, la PC será interrumpida por el MPA para transferir los datos del evento a disco y esto provocará que la rutina en proceso quede truncada.

Si se desea terminar la adquisición de datos, será necesario entrar de nuevo al menú principal, e indicarle con la opción 5 la terminación de adquisición de datos. Un letrero mostrará que se ha suspendido el proceso.

6.2.6 Graficación o desplegado de un archivo.

Al invocar esta opción se despliega el siguiente menú (figura 6.11):

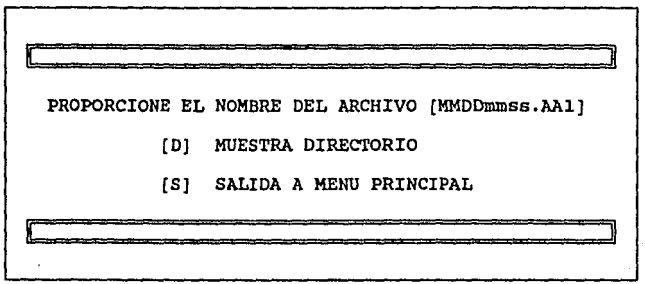


Figura 6.11 Menú de petición de archivo.

Seleccionando un archivo, se tiene una nueva opción como se muestra en la figura 6.12.

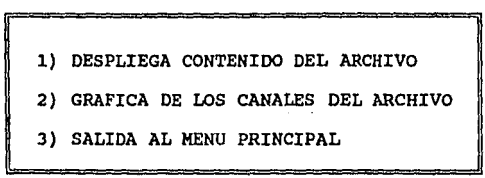


Figura 6.12 Menú de manejo de Archivo

La opción de despliegue del contenido de un archivo permite desplegar numéricamente los datos en forma hexadecimal o directamente en volts. El desplegado se realiza en la misma forma que en la escritura ó lectura a RAMS, desplegando el archivo en bloques de 32 Kbytes. Cabe mencionar, que para el caso de despliegue en volts, el usuario indica que canales, de los 16 que se tienen, desea ver, teniendo la opción de despelgar hasta 4 canales en una pantalla.

En la figura 6.13 se da un ejemplo de desplegado de un archivo en volts.

DESPLEGADO DEL ARCHIVO: "07262046.901".
(DATOS EN VOLTS).

LOCALIDAD	CANAL 2	CANAL 3	CANAL 7	CANAL 16
0000	0.0236	0.0945	0.0236	0.0236
0010	0.0236	0.1102	0.0236	0.0236
0020	0.0236	0.1339	0.0236	0.0236
0030	0.0236	0.1417	0.0236	0.0236
0040	0.0236	0.1339	0.0236	0.0236
0050	0.0236	0.1260	0.0236	0.0236
0060	0.0236	0.1102	0.0236	0.0236
0070	0.0236	0.0866	0.0236	0.0236
0080	0.0236	0.0630	0.0236	0.0236
0090	0.0236	0.0315	0.0236	0.0236
00A0	0.0236	0.0000	0.0236	0.0236
00B0	0.0236	0.0000	0.0236	0.0236
00C0	0.0236	-0.0079	0.0236	0.0236
00D0	0.0236	-0.0157	0.0236	0.0236
00E0	0.0236	-0.0157	0.0236	0.0236
00F0	0.0236	0.0000	0.0236	0.0236

PRESIONE (1) SEGUIR EN ZONA DATOS (2) ZONA PARAMETROS (3) SALIDA :

Figura 6.13 Desplegado de un archivo en volts.

Al iniciar la graficación de un evento, los datos completos de los canales que se desean graficar se leen de disco duro y se colocan en memoria, mientras esto ocurre, aparece en la pantalla la fecha y hora del evento, así como el canal y estación en donde se detectó.

La pantalla utilizada en este sistema tiene una resolución de 200 x 640 pixeles, 2 colores (CGA).

La graficación puede hacerse de cualquiera de los 16 canales, teniéndose la opción de graficar hasta 4 canales en una sola pantalla, con una decimación desde 1 hasta 5, y una amplitud de 1 a 2.

Cada muestra almacenada tiene un rango de valores de 00H a FFH y corresponden a -1 y 1 volt, respectivamente.

El convertidor analógico digital entrega los datos digitales de 16 canales y los almacena en las RAMS de la tarjeta de adquisición de manera tal, que los datos del canal 1 quedan almacenados en las localidades 0000, 0010, 0020, ..., los datos del canal 2 se encuentran en las localidades 0001, 0011, 0021, ..., el canal 3 en la 0002, 0012, 0022, etc., y así hasta llegar al canal 16, que quedará en las localidades 000F, 001F, 002F, etc. De manera semejante quedan almacenados en la memoria de la PC al leerse el archivo.

El procedimiento de graficación realiza un barrido de todas las localidades, graficando solo aquellos canales que el usuario escogió y con la decimación y amplitud seleccionadas.

En la graficación de las señales se presenta un par de ejes coordenados: el eje Y representa la amplitud en volts, el eje X el tiempo.

La graficación de la señal se efectúa leyendo los datos previamente almacenados en memoria y transformandolos a la escala de la pantalla, obteniendo así los valores en el eje Y. Por ejemplo al graficar un canal con una amplitud de 2 se tiene que 255 cuentas, valor máximo de una muestra, corresponden a 127 pixeles en pantalla, es decir 2 cuentas/pixel. En este caso el intervalo de graficación en pixeles es de 37 (255 cuentas) a 164 (0 cuentas).

El eje X se incrementa en uno con cada muestra, cada punto en el eje X equivale a 0.025 segundos (velocidad de muestreo = 40 muestras/seg).

El valor en pixeles de una muestra para un solo canal y con un valor de amplitud= 2 es:

Posición en el eje Y = 164 - TRUNC (Valor de la señal/ 2).

El valor de 164 es el máximo valor que puede tomar una señal en pixeles para el eje Y y corresponde a un valor de 0 volts de la señal.

La graficación de 2,3 y 4 canales por pantalla se grafica de forma semejante cambiando simplemente las constantes de la función.

Los diagramas de flujo detallados de las rutinas de graficación y desplegado de datos se presentan en el anexo V.

A continuación se muestran 2 ejemplos de datos gráficados en pantalla (figura 6.14 a y b). La señal corresponde a una función de prueba.

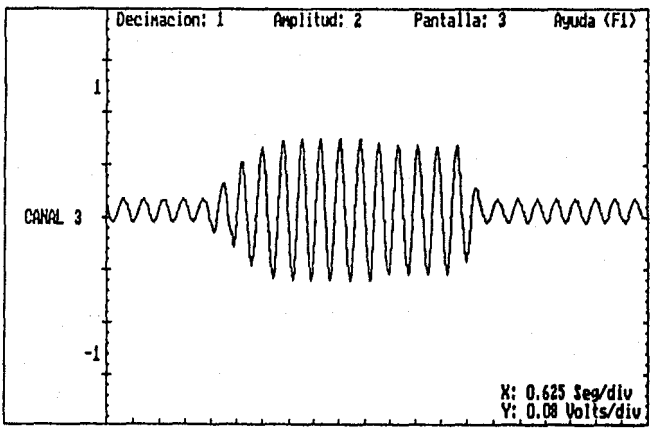


Figura 6.14 a) Grafica en pantalla de 1 canal con decimación 1.

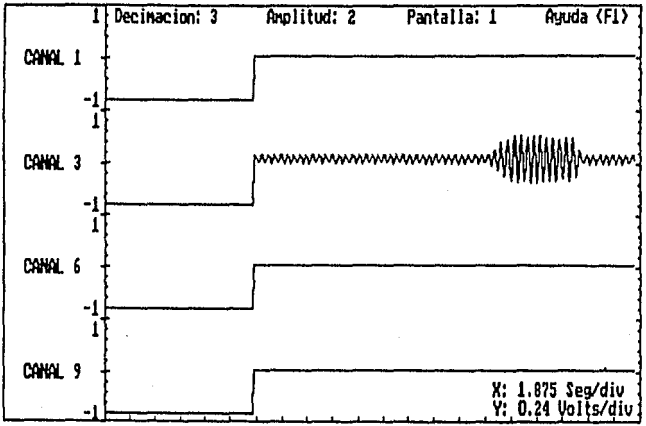


Figura 6.14 b) Grafica de 4 canales con decimación 3.

Al terminar la graficación aparecerá el menú de la figura 6.7.

6.2.7 Despliegue de información estadística.

Con esta opción se presenta un submenú como el mostrado en la figura 6.15.

1) DESPLIEGA EVENTOS OCURRIDOS EN UNA ESTACION.

2) DESPLIEGA EVENTOS OCURRIDOS EN UNA FECHA.

3) DESPLIEGA TODOS LOS EVENTOS.

4) SALIDA AL MENU PRINCIPAL

Figura 6.15 Menú de despliegue de información.

La opción 1 pide al usuario el nombre de la estación de la cual desea conocer los eventos ocurridos; en seguida el programa listará archivo, fecha y hora de inicio de todos los eventos ocurridos en esa estación.

La opción 2 pide al usuario la fecha de la cual desea conocer los eventos ocurridos. El programa listará archivo, estación y hora de inicio de los eventos, además dibuja un mapa de la República Méxicana en donde se indica la localización de las estaciones que registraron los eventos en esa fecha.

Con la opción 3 aparecerá un listado de los archivos generados, la estación en donde ocurrió el disparo y la fecha y hora de inicio del evento como se muestra en la figura 6.16.

DIRECTORIO DE DATOS:

ARCHIVO	FECHA	HORA	ESTACION
07092102.901	09/ Jul./ 1990	21:02:03.02	SANTA RITA, EDO. MEX.
07170025.901	17/ Jul./ 1990	00:25:04.17	SANTA RITA, EDO. MEX.
07231731.901	23/ Jul./ 1990	17:31:22.08	SANTA RITA, EDO. MEX.
07231732.901	23/ Jul./ 1990	17:32:11.18	TONANTZINTLA, PUE.
08011848.901	01/ Ago./ 1990	18:47:40.72	TONANTZINTLA, PUE.
01010226.801	01/ Ene./ 1980	02:26:23.67	TONANTZINTLA, PUE.
01010229.801	01/ Ene./ 1980	02:28:51.42	TONANTZINTLA, PUE.
01010041.801	01/ Ene./ 1980	00:40:45.76	SANTA RITA, EDO. MEX.
07230001.901	23/ Jul./ 1990	00:01:28.19	TONANTZINTLA, PUE.
07200005.901	20/ Jul./ 1990	00:04:41.15	TONANTZINTLA, PUE.
07230008.901	23/ Jul./ 1990	00:07:33.78	SANTA RITA, EDO. MEX.
07230500.901	23/ Jul./ 1990	05:00:27.52	TONANTZINTLA, PUE.
07231315.901	23/ Jul./ 1990	13:15:13.16	TONANTZINTLA, PUE.
07231321.901	23/ Jul./ 1990	13:21:14.51	TONANTZINTLA, PUE.
07231335.901	23/ Jul./ 1990	13:35:15.42	TONANTZINTLA, PUE.
07231339.901	23/ Jul./ 1990	13:39:22.20	TONANTZINTLA, PUE.
03271350.901	27/ Mar./ 1990	13:50:23.89	TONANTZINTLA, PUE.
01010000.801	01/ Ene./ 1980	00:00:24.87	TONANTZINTLA, PUE.

PRESIONE <ENTER> PARA SEGUIR.

Figura 6.16 Ejemplo de despliegue de información estadística.

6.3 Transferencia y almacenamiento de datos en disco.

Cuando el MPA se encuentra adquiriendo datos, la computadora puede realizar otras tareas. Sin embargo, si el algoritmo de disparo del MPA detecta un evento, la computadora PC será interrumpida y ejecutará la rutina de interrupción.

La rutina de interrupción tiene como objetivo principal la transferencia y almacenamiento de datos en disco duro. Transfiere los datos del evento almacenados por el MPA en la memoria de conmutación a la RAM de la PC. Enseguida crea un archivo ASCII y almacena los datos de RAM en disco duro .

6.3.1 Proceso de interrupción.

Para poder realizar la rutina de interrupción fué necesario analizar con detalle el sistema de interrupciones que maneja la computadora PC y determinar el tipo de interrupción más conveniente. Una descripción completa del sistema de interrupciones de la PC se da en el ANEXO IV.

El microprocesador 8088, al igual que la mayoría de los microprocesadores, tiene dos entradas de interrupción disponibles: una entrada de interrupción mascarable (INTR) y otra de interrupción no mascarable (NMI). como se muestra en la fig 6.17.

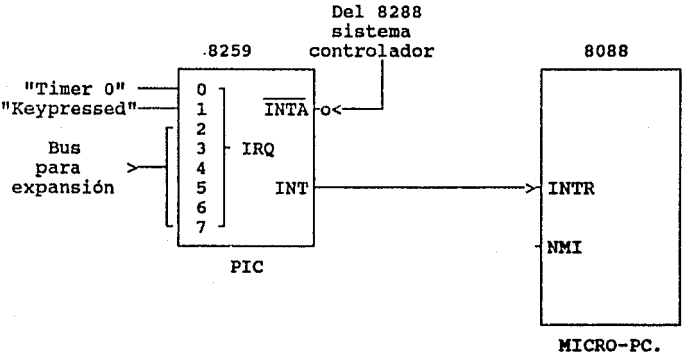


Figura 6.17 Sistema de interrupción de PC.

La interrupción INTR es generada por el circuito 8259, un controlador de interrupciones programable (PIC). Este permite un máximo de 8 interrupciones mascarables. Sin embargo, como se muestra en la tabla 6.2, no todos estos niveles de interrupción están disponibles para el usuario.

	Nivel de Interrupción		Uso
PRIORIDAD	Tienen el más alto nivel de prioridad. No disponibles al usuario.	NMI	RAM que maneja paridad de teclado Entrada/Salida del canal paridad. Coprocesador numérico 8087.
		IRQ0	Sistema de tiempos.
		IRQ1	Teclado.
	Disponibles en el bus del sistema. Disponibles al usuario.	IRQ2	Reservado.
		IRQ3	Puerto serie COM2.
		IRQ4	Puerto serie COM1.
		IRQ5	Controlador de disco duro.
		IRQ6	Controlador de disco flexible.
		IRQ7	Puerto paralelo LPT1.

Tabla 6.2 Tabla de interrupciones de la PC.

La interrupción IRQ0 es la de mayor prioridad e IRQ7 la de menor.

La interrupción escogida para el sistema fué la IRQ3, ya que es una de alta prioridad y corresponde al puerto serie COM2 no empleado por esta PC.

Físicamente la interrupción se efectúa cuando la entrada IRQ3 del 8259 recibe un pulso enviado por el MPA.

Los primeros 1024 bytes de memoria RAM de la PC están dedicados a guardar el vector de las 256 interrupciones que tiene la PC. Cada una de las interrupciones tiene asignadas 4 localidades de memoria para su vector.

Las interrupciones IRQ del 8259 se encuentran dentro de esta zona de memoria de la PC. Cada IRQ tiene asignadas 4 localidades para almacenar el código de segmento CS y el apuntador de dirección IP que definen la dirección de inicio de la rutina de interrupción.

En la tabla 6.3 se muestran las localidades utilizadas por las IRQ y se enmarca IRQ3, que es la interrupción utilizada en el sistema.

Interrupción IRQ.	Direcciones para almacenar el IP.		Direcciones para almacenar el CS.	
IRQ0	0020H	0021H	0022H	0023H
IRQ1	0024H	0025H	0026H	0027H
IRQ2	0028H	0029H	002AH	002BH
IRQ3	002CH	002DH	002EH	002FH
IRQ4	0030H	0031H	0032H	0033H
IRQ5	0034H	0035H	0036H	0037H
IRQ6	0038H	0039H	003AH	003BH
IRQ7	003CH	003DH	003EH	003FH

Tabla 6.3 Localidades del vector de interrupciones utilizadas por el dispositivo 8259.

Al principio de la ejecución del programa principal es necesario programar al 8259 a través de los puertos 0020H y 0021H, enviandole palabras de control de inicialización (ICW's) y después, dentro de la rutina de interrupción se envían al 8259 las palabras de control de operación (OCW's). Con ambos tipos de palabras de control se define su modo de operación.

En la tabla 6.4 se resumen los valores de las palabras ICW y OCW necesarios en nuestro sistema para programar al 8259.

Se envia por el puerto:	ICW	byte D7 -- Do	Valor Hexadecimal
0020H	ICW1	00010011	13
0021H	ICW2	00001000	08
0021H	ICW3	--------	--
0021H	ICW4	00001001	09
0021H	OCW1	11110111	F7
0020H	OCW2	01100011	63
0020H	OCW3	00001011	0B

Tabla 6.4. Palabras de inicialización y control en la interrupción.

Los pasos consecutivos del proceso de interrupción son:

1. Un ciclo de interrupción es iniciado por un flanco de subida en la entrada IRQ3 del 8259.

2. El 8259 examina la prioridad de la interrupción y la coloca en una cola de interrupciones.

3. La salida INT del 8259 se activa en alto, y con ello la entrada INTR del micro 8088.

4. El 8088 reconoce la interrupción y se activa la señal INTA del 8288 que bloquea la cola de prioridades del 8259.

5. El 8288 pulsa el INTA por segunda vez, que provoca que el 8259 coloque sobre el bus de datos el valor de 2CH que corresponde a la interrupción de IRQ3.

6. El 8088 lee el tipo de interrupción y calcula la dirección del vector de interrupción correspondiente (CS, IP).

7. El 8088 guarda en el stack los contenidos actuales del IP y CS, salva los valores de los registros, el estado de las banderas y carga los nuevo valores del CS e IP calculados.

8. El 8088 ejecuta el procedimiento o rutina de interrupción.

6.3.2 Rutina de interrupción: transferencia y almacenamiento de datos en disco.

La rutina de interrupción es un programa escrito en ensamblador 8088.

El programa principal inicializa la rutina de interrupción en los siguientes pasos:

1. Habilita la bandera de interrupción para que el 8088 esté preparado para recibir un llamado de interrupción.

2. Carga el programa de la rutina de interrupción en memoria de la PC. La dirección en la que se instala el programa es cualquier localidad disponible en en memoria y representa la dirección de inicio de la rutina de interrupción (CS:IP).

3. Coloca en las localidades 0000:002C y 0000:002E (tabla 6.3) de memoria de la PC los valores del IP y CS, respectivamente.

4. Envía por los puertos 0020H y 0021H las palabras de inicialización del 8259 (ICW's de la tabla 6.4).

Con lo anterior la rutina de interrupción queda lista para ejecutarse en cualquier momento.

La interrupción de la PC se efectúa si el MPA se encuentra adquiriendo datos y su algoritmo de disparo detecta un evento. A continuación se muestra un diagrama de flujo general de la rutina de interrupción (figura 6.18).

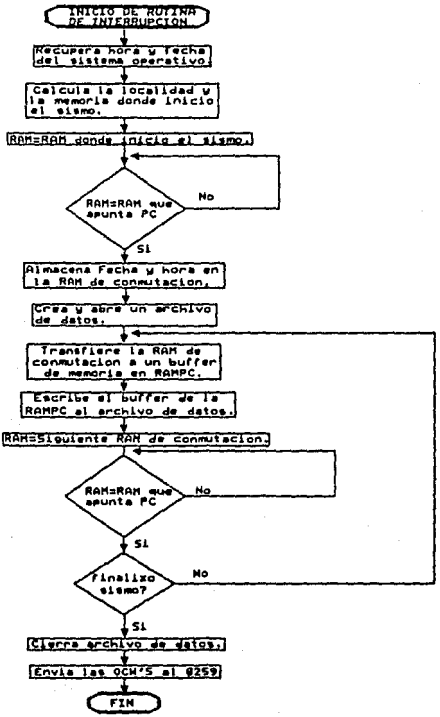


Figura 6.18 Diagrama de flujo general de la rutina de interrupción.

El listado del programa de la rutina de interrupción, así como el diagrama de flujo detallado de la misma se presentan en los anexos VI y V, respectivamente.

La rutina de interrupción inicia obteniendo el tiempo del sistema operativo, para tener la hora y fecha en que un evento disparó al sistema. Utiliza la funcion 2CH y 2AH de la interrupción 21H de MSDOS para recuperar el tiempo del sistema operativo.

El nombre del archivo de datos se forma con el mes, el día, la hora y los minutos de la fecha del disparo. La extensión contiene los últimos dos dígitos del año y una constante con valor de 1 que representa la ocurrencia del evento (MMDDhhmm.AA1).

Luego dentro de la rutina de interrupción se recuperan ciertos parámetros que el MPA envía en localidades de ambas RAMS de conmutación (tabla 6.5). La PC accesa a las RAMS de conmutación por la dirección D000:****.

Localidades en RAMS de conmutación	Contenido	Significado
000FH	D1	La PC apunta a RAM1 de conmutación, MPA a RAM2
000FH	D2	La PC apunta a RAM2 de conmutación, MPA a RAM1
0021H y 0022H	Cualquier valor	Localidad dentro de la RAM de conmutación donde el algoritmo de disparo detectó evento.
0024H	A0	Algoritmo de disparo detectó evento en RAM1.
0024H	C0	Algoritmo de disparo detectó evento en RAM2.
0028H	De 00 a 1E	Canal que detectó primero el evento. 00=canal 1, 02=canal 2, ..., 1E=canal 16.

Tabla 6.5 Parámetros que envía el MPA a la PC a través de las RAMS de conmutación.

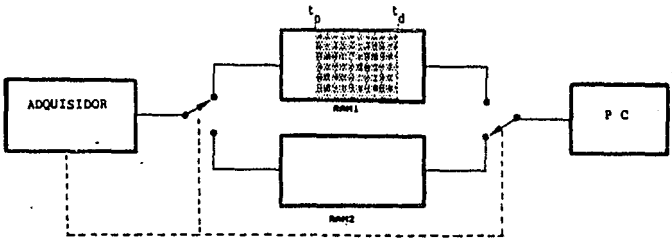
Con el valor obtenido en la localidad 0028H se conoce el número de canal que detectó el evento y se determina el nombre de la estación en donde ocurrió.

La fecha, hora, estación y canal de disparo del evento, junto con el nombre del archivo donde se almacenarán los datos son desplegados en la pantalla hasta que termine la rutina de interrupción.

Cuando un evento dispara al sistema, es porque el algoritmo de disparo detectó que se sobrepasó el umbral. Sin embargo, el sobrepaso del nivel de umbral no indica el inicio del evento.

El inicio del evento se encuentra dentro de los 30 segundos previos al disparo, entonces, se deben restar 30 segundos de preevento al momento en que ocurrió el disparo.

Es necesario considerar que un evento que dispara al sistema, puede estar distribuido en una o más RAMS de conmutación, por ello el inicio del evento puede quedar ó no en la misma RAM de conmutación en que ocurrió el disparo (figura 6.19 a y b).

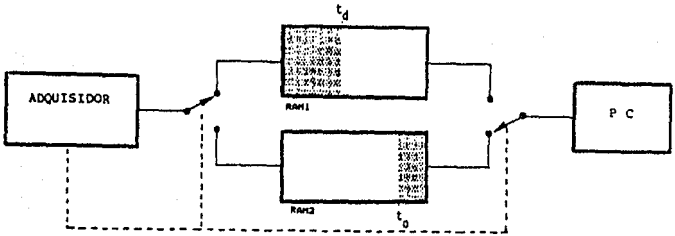


t_o = tiempo de inicio del sismo

t_d = tiempo de disparo del sistema

$t_d - t_o$ = tiempo de preevento (30 seg)

Figura 6.19 a) Posición del inicio de un sismo y los datos de preevento en una memoria RAM de conmutación.



t_o = tiempo de inicio del sistema

t_d = tiempo de disparo del sistema

$t_d - t_o$ = tiempo de preevento (30 seg)

Figura 6.19 b) Posición del inicio de un sismo y los datos de preevento en ambas memorias RAMS de conmutación.

La localidad de disparo dentro de la RAM de conmutación nos determina la localidad de inicio del evento. Existen cuatro casos críticos de la localización del inicio de un evento dentro de las RAMS de conmutación como se muestra en la figura 6.20.

Se debe tener presente que el MPA llena cada una de las RAMS de conmutación con 51.2 segundos de información, ya que este tiempo es el que tarda el MPA en llenar una RAM de conmutación a la velocidad de muestreo de 40 Hz.

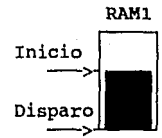
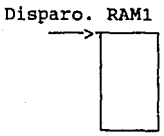
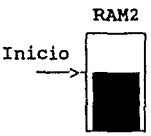
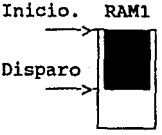
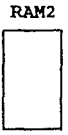
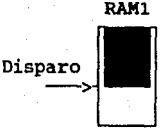
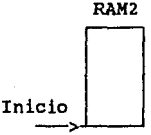
Direccionada por el MPA.	Direccionada por al PC.	Comentario.
		El MPA terminó RAM1 y direccionará RAM2 y la PC a RAM1. La PC tiene 51.2 seg (tiempo que tarda el MPA en llenar RAM2) para almacenar los datos de RAM1 en la memoria de la PC, luego al archivo.
		La PC está apuntando a RAM2, que es donde inició el even- Tendrá los 51.2 seg que tarda el MPA en llenar RAM1. Por lo tanto la PC tiene que transferir los datos de RAM2 a su memoria y luego al archivo.
		Al MPA le faltan 21.2 seg. en llenar RAM1. La PC tiene que esperar este tiempo antes de direccionar a RAM1 y transferir los datos a su memoria y luego al archivo.
		Al MPA le faltan 21.2 seg. para llenar RAM1, mismos que tiene la PC para guardar RAM2 en su memoria y luego al archivo.

Figura 6.20. Casos críticos de la localización del inicio de un sismo en las RAMS (continuación).

Cada RAM de conmutación tiene una capacidad de 32,767 bytes, que corresponden a 51.2 segundos de información. 30 segundos de información que equivalen a la memoria de preevento corresponden por tanto a 19,200 bytes.

De la tabla 6.5, con los valores obtenidos en las localidades 000FH (RAM a la que apunta la PC), 0024H (RAM de disparo), 0021H y 0022H (localidad de disparo), la rutina de interrupción realiza un cálculo para determinar en qué RAM de conmutación inició el evento.

Para el cálculo de la localidad de inicio del evento, se restan 19,328 bytes a la localidad de disparo. Los 19,200 bytes de memoria de preevento más 128 bytes de memoria RAM de conmutación asignados a la zona de parámetros.

Al calcular la localidad de inicio se pueden tener dos casos: el primero es que la localidad de disparo tenga un valor mayor a 19,328 bytes, esto significa que la localidad de inicio del evento se encuentra en la misma RAM de conmutación en donde se encuentra la localidad de disparo:

Loc. inicio del evento = Loc. disparo -19,328.

En el segundo caso la localidad de disparo tiene un valor menor a 19,328. Esto significa que el inicio del evento se localiza en la otra RAM de conmutación de donde se encuentra la localidad de disparo. La localidad de inicio del evento se calcula restando a una RAM de 32,767 bytes el valor que se excedió de la localidad de disparo:

Loc. inicio del evento = 32,767 + (Loc. disparo - 19,328).

Si se conocen la RAM de disparo, la RAM de conmutación a la que apunta la PC (de la tabla 6.5) y la localidad de inicio del evento, entonces, se puede conocer la RAM en donde inició el evento.

La rutina de interrupción crea y abre el archivo de datos en disco duro utilizando las funciones 3CH y 3DH, respectivamente, de la interrupción 21H de MSDOS.

En el momento que la rutina de interrupción comprueba que la PC apunta a la RAM de conmutación que contiene la localidad de inicio del evento, escribe dentro de la zona de parámetros de la RAM de conmutación (localidad 70H a la 78H) la fecha y hora del disparo, obtenida del sistema operativo al inicio de la rutina. Enseguida la rutina de

interrupción realiza la transferencia de toda la RAM de conmutación a la RAMPC.

Los 32 Kbytes de la RAM de conmutación que se almacenaron en RAMPC se escriben dentro del archivo de datos en disco duro, utilizando la función 40H de la interrupción 21H.

El tiempo requerido para transferir una RAM de conmutación a la RAMPC (32 Kbytes) es de 80 msegundos, y de RAMPC a un archivo en disco duro es de 1.6 segundos. Estos tiempos se obtuvieron con los programas TRANSFIERE_1.ASM y GENERA_ARCHIVO.ASM que se presentan en el anexo I.

La rutina de interrupción espera la señal que envía el MPA, a través de las RAMS de conmutación (tabla 6.6), indicando que el evento ha finalizado. Sí no la recibe, espera que la PC apunte a la siguiente RAM de conmutación para continuar transfiriendo datos a RAMPC y almacenandolos en el archivo en bloques de 32 Kbytes (figura 6.18).

Localidades en RAMS de conmutación	Contenido	Significado
0040H y 0041H	Cualquier valor	Localidad dentro de la RAM de conmutación donde terminó el evento
0042H	A0	Finalizó evento en RAM1
0042H	C0	Finalizó evento en RAM2

Tabla 6.6 Parámetros que indican el final del evento.

La rutina de interrupción termina cuando recibe la señal de que el evento finalizó. Luego ejecuta lo siguiente:

- Cierra el archivo de datos con la función 3EH de la interrupción 21H de MSDOS.
- Deja a la rutina de interrupción residente en memoria de la PC, utilizando la función 31H de la interrupción 21H de MSDOS .
- Envía las palabras de operación del 8259 (OCW's de la tabla 6.4) a través de los puertos 0020H y 0021H
- Sale de la rutina de interrupción y regresa a la rutina o programa de origen.

VII. PRUEBAS Y RESULTADOS DEL SISTEMA DE ADQUISICION

En este capítulo se presentan algunas pruebas realizadas al sistema de adqusición con el objetivo de mostrar de manera general algunas características importantes de su funcionamiento básico.

Se presentan tres pruebas:

- Comparación de una señal de entrada al sistema y la obtenida por el sistema de adquisición.
- Verificación de la línea que activa la interrupción a PC al detectarse un sobrepaso del umbral en una señal de entrada.
- Un evento sísmico real como entrada al sistema.

Para cada caso se presentan las gráficas obtenidas, tanto con el osciloscopio para las señales analógicas de entrada, como en la pantalla de la computadora para la información digital adquirida.

7.1 Comparación de una señal de entrada al sistema con la obtenida por el sistema de adquisición de datos.

El objetivo de esta prueba fué comparar una señal de entrada al sistema y la reproducida por la PC en pantalla después de haber efectuado la adquisición.

En la figura 7.1a se muestra la señal de entrada al sistema obtenida mediante un generador de funciones. En la figura 7.1b se tiene la señal captada por la PC a partir del punto indicado por el cursor en la figura 7.1a.

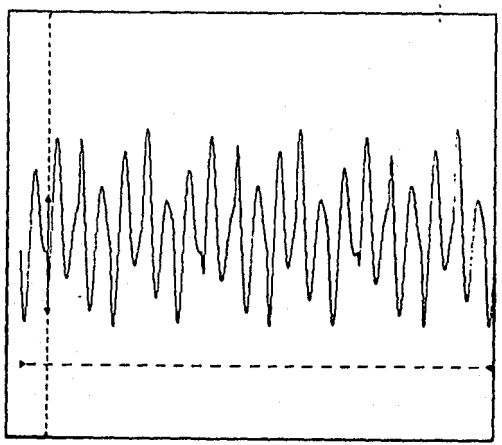


Figura 7.1 a) Señal de entrada al sistema de adquisición.

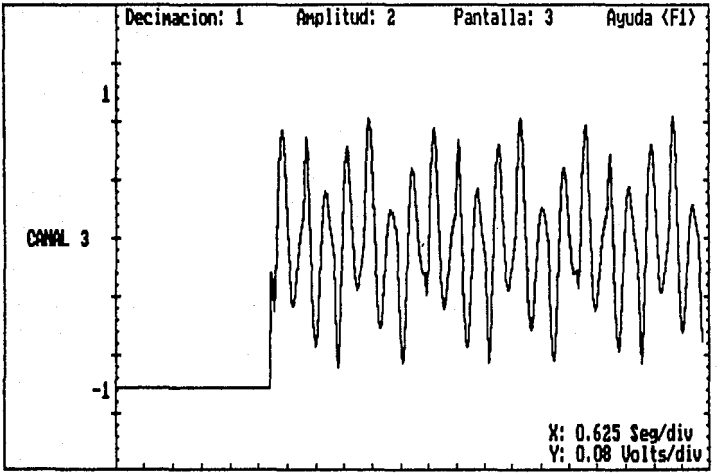


Figura 7.1 b) Gráfica de la señal de entrada después de la adquisición.

7.2 Verificación de la línea que activa la interrupción a PC al detectarse un sobrepaso de la señal de entrada por encima del umbral seleccionado.

Para esta prueba el umbral se fijó en 2 volts (p-p)(40% de la amplitud máxima que alcanza la señal de entrada). El voltaje se escogió con el objetivo de distinguir fácilmente el disparo en la gráfica.

La señal de entrada tuvo las siguientes características:

Señal: senoidal
Amplitud: de 0 a 5 v(p-p)
Frecuencia: 2 Hz
Voltaje de offset: 2.5 v

En la parte superior de la figura 7.2 se muestra la señal que el MPA manda a la PC para generar la interrupción. El flanco de subida indica el punto en que la señal de entrada rebasa el umbral definido.

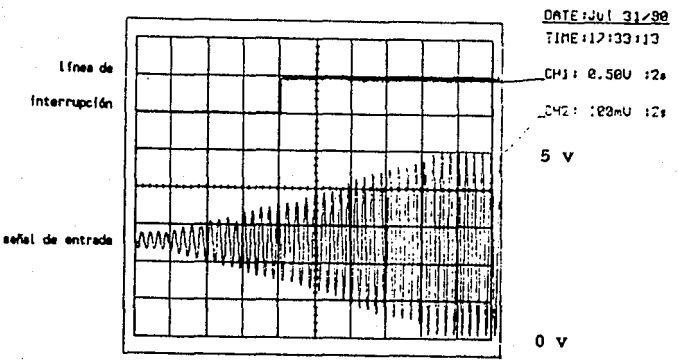


Figura 7.2 Línea de interrupción a PC y señal de entrada.

7.3 Evento sísmico real detectado por el sistema.

El objetivo de esta prueba fué mostrar el funcionamiento del sistema con el registro de un evento sísmico real.

La entrada al sistema es parte del sismo ocurrido el 19 de septiembre de 1985, cuya gráfica en osciloscopio se muestra en la figura 7.3. En la figura 7.4 se observa este evento en la pantalla de PC después de haber sido detectado y desplegado.

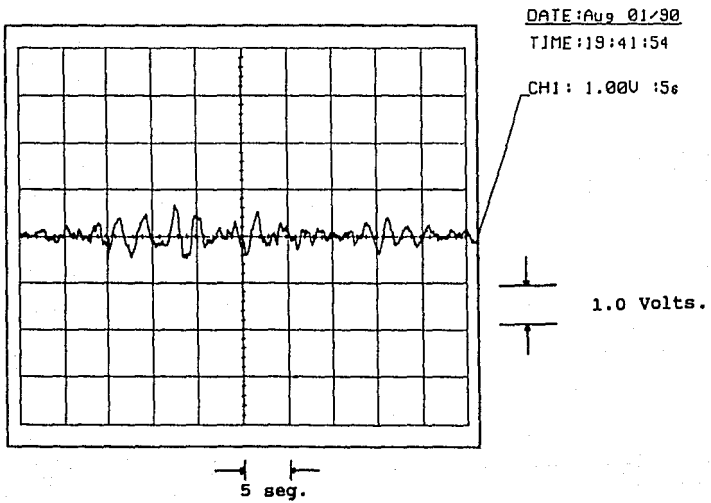


Figura 7.3 Parte de la señal del sismo del 19 de septiembre de 1985 vista en el osciloscopio.

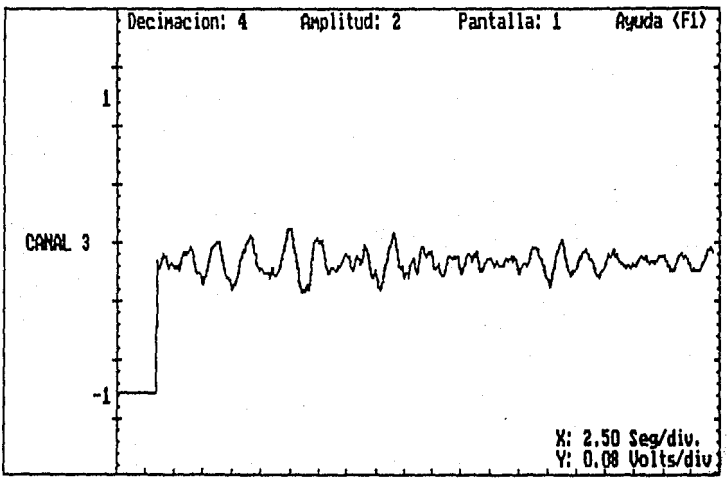


Figura 7.4 Gráfica del sismo de la figura 7.3 registrado con el sistema y desplegado en la pantalla de la PC.

VIII. CONCLUSIONES

Con la elaboración de este trabajo se concluye la primera etapa del sistema digital de adquisición de datos sísmicos, la cual comprende el diseño del hardware y el desarrollo del software básico. Con esto se dejan las bases para futuras expansiones del sistema, como podrían ser: desarrollo de rutinas de graficación y análisis, más sofisticadas, además de la depuración y ampliación de las rutinas de adquisición.

Las diversas pruebas realizadas al sistema han dado resultados satisfactorios, aunque falta someterlo a pruebas más exhaustivas y probarlo ya integrado a la red telemétrica SISMEX.

A pesar de que el método utilizado en la adquisición de los datos, empleando un conversor A/D de alta integración y un microprocesador bajo control de una PC, es convencional; el sistema de registro y transferencia de los mismos, basado en una memoria de conmutación, constituye una técnica novedosa que podría ser útil para procesos en tiempo real similares. Además, el adquisidor desarrollado puede utilizarse para otras aplicaciones de registro e inclusive podría convertirse en un sistema universal de adquisición de uso múltiple, modificando ligeramente la programación del microprocesador y de la PC.

Con este sistema se espera mejorar sustancialmente la infraestructura de registro de SISMEX, permitiendo la captura y evaluación de la información de manera más eficiente y sobre todo oportuna.

En caso de presentarse un sismo con características similares a las de los temblores ocurridos en 1985, durante los cuales se registraron, además de los dos eventos principales, más de 1000 réplicas en los días posteriores, un sistema como el desarrollado hubiera sido de gran ayuda, tanto para la evaluación inmediata de la información como su posterior análisis con técnicas numéricas apropiadas.

Asimismo podría utilizarse como un sistema de alertamiento temprano, el cual, basado en un algoritmo de detección más sofisticado, pudiese proporcionar una señal

de alerta, en caso de un temblor importante, con una anticipación de hasta 30 segundos aprovechando el tiempo de propagación de la señal sísmica desde la estación más lejana de la red. Esta señal podría emplearse a su vez para activar por control remoto las estaciones acelerográficas existentes en el Valle de México, de tal manera que estas se encuentren registrando en el momento de arribo de la onda sísmica.

Además de la utilidad práctica que representa el sistema, como infraestructura de registro sísmico del Instituto de Ingeniería, el desarrollo del trabajo dejó una valiosa experiencia dado que requirió, no sólo de ingenio en su diseño electrónico, sino de una importante labor de programación, tanto a nivel de lenguaje de máquina, como en lenguaje de alto nivel.

IX. BIBLIOGRAFIA.

- Torres M ., Mena E., "El sistema de información sismotelemétrico de México, SISMEX. Avances hasta 1987, Instituto de Ingeniería, UNAM, agosto 1983.

- Alvarez T., Cajiga G., Fortoul A., "Manuales de operación y mantenimiento de las instalaciones de SISMEX", Instituto de Ingeniería, UNAM.

- Douglas V. Hall, "Microprocesors and interfacing, programming and hardware", McGraw Hill, Singapore, 1986.

- Byton W. Putman, "Microcomputer hardware, operation, and trouble shooting with IBM PC applications", Prentice Hall, New Jersey, 1988.

- Lewis C. Eggebrecht, "Interfacing to the IBM personal computer".

- Kai Hwang/Fayé A. Briggs, "Arquitectura de Computadoras y procesamiento paralelo", McGraw Hill, 1987.

- Thomas Sheldon, "PC-DOS y MS-DOS", McGraw Hill, 1987.

- "MPC operations guide", Columbia data products, 1984.

- Ray Duncan, "Programmer's Quick Reference, Series IBM, ROM BIOS", Microsoft, Press, 1984.

- Paul Bildstein, "Filtros Activos", Paraninfo, Marcombo, 1977.

- "Linear Data Book", National Semiconductor Corporation, 1984.

- "Logic Data Book", Volumen II, National Semiconductor Corporation, 1984.

- "Manual de semiconductores de silicio", Texas Instruments, 1985.

- R. Mejía, B. López, C. Pérez, R. Quaas, "Sistema digital de adquisición de datos para la red de telemetría SISMEX", VI Congreso Nacional de Instrumentación, Guanajuato, 1990.

A N E X O I.

Listados de los programas de pruebas a disco.

A N E X O I.

Se presentan las rutinas desarrolladas para medir los tiempos de acceso, escritura a disco, así como el tiempo de transferencia de datos de la memoria conmutable a RAM de la PC.

Para medir los tiempos se hace uso del puerto PPC del sistema de adquisición. La tarea que se desea medir se repite cíclicamente (loop); al término de cada ciclo se envía un señal al PPC. Esta señal se mide luego con el osciloscopio.

El puerto PPC es direccionado por la PC a través de la localidad E000. Estas líneas de direccionamiento de la PC entran a un decodificador de la tarjeta de adquisición, que activa el puerto PPC. La señal medida es la terminal 6 del circuito U28.

1) Rutina MIDE_TRACK1.ASM:

Con este programa se mide el tiempo que tarda el sistema de la PC en guardar 512 bytes (mínima información almacenable en disco) en una pista y sector definidos del disco flexible.

Se emplea la interrupción 13H del BIOS.

MIDE_TRACK1.ASM

```
code segment
assume cs:code, ds:code

inicio: mov ax,E000h    ; Direccionamos el puerto PPC a través
        mov ds,ax       ; del decodificador, enviando un valor
        mov si,0000h    ; de 0 por la localidad E000:0000.
        mov bl,00h
        mov [si],bl
        mov ax,2000h    ; Localidad de memoria PC a copiar a
        mov es,ax       ; disco flexible, el segmento se coloca
        mov bx,0000h    ; en el registro ES y el offset en el BX.
        mov al,01h      ; Para transferir un sector.
        mov dl,00h      ; Escribir al drive A.
        mov dh,01h      ; Cabeza 1.
        mov ch,25h      ; Track 25h = 37.
        mov cl,08h      ; Sector 8.
        mov ah,03h      ; funcion 03, realiza la escritura a disco
```

```
        int 13h         ; de un buffer de memoria PC.
        jb fin          ; Si falla la escritura se va a fin.
        mov ah,01h      ; Con esta función espera un teclazo para
        int 16h         ; terminar con el loop.
        jz inicio
   fin: mov ax,4c00h    ; Sale a MSDOS
        int 21h

        code ends
        end inicio
```

El resultado obtenido fué:

T1 = 200 mseg

2) Rutina MIDE_TRACK2.ASM:

Este programa es sólo una variante del programa anterior. Con él se mide el tiempo que tarda el sistema de la PC en guardar 512 bytes (mínima información almacenable en disco) en una pista y sector definidos del disco duro.

Se emplea la interrupción 13H del BIOS.

MIDE_TRACK2.ASM

```
        code segment
        assume cs:code, ds:code

inicio: mov ax,E000h    ; Direccionamos el puerto PPC a través
        mov ds,ax       ; del decodificador, enviando un valor
        mov si,0000h    ; de 0 por la localidad E000:0000.
        mov bl,00h
        mov [si],bl
        mov ax,2000h    ; Localidad de memoria PC a copiar a
        mov es,ax       ; disco duro, el segmento se coloca
        mov bx,0000h    ; en el registro ES y el offset en el BX.
        mov al,01h      ; Para transferir un sector.
        mov dl,80h      ; Escribir en disco duro.
        mov dh,01h      ; Cabeza 1.
        mov ch,FEh      ; Track FEh = 254, track=cilindro.
        mov cl,0Ah      ; Sector A.
        mov ah,03h      ; funcion 03, realiza la escritura a disco
        int 13h         ; de un buffer de memoria PC.
        jb fin          ; Si falla la escritura se va a fin.
        mov ah,01h      ; Con esta función espera un teclazo para
        int 16h         ; terminar con el loop.
        jz inicio
   fin: mov ax,4c00h    ; Sale a MSDOS
        int 21h
```

```
code ends
end inicio
```

El resultado obtenido fué:

T2 = 17 mseg.

Se aprecia que este tiempo es sensiblemente menor que el equivalente T1 en el disco floppy.

3) Rutina MIDE_HEAD1.ASM:

Con este programa se mide el tiempo que tarda la cabeza del controlador de disco flexible en moverse de una pista a otra contigua.

Se emplea la interrupción 13H del BIOS.

Del mismo modo que en los programas anteriores, por medio del osciloscopio se pueden conocer los tiempos de escritura a dos pistas contiguas.

Si con el programa MIDE_TRACK1.ASM se midió el tiempo de escritura en un pista del disco flexible; entonces al escribir en dos pistas contiguas, se deduce cual es el tiempo que tarda la cabeza en moverse de un pista a otra.

MIDE_HEAD1.ASM

```
code segment
assume cs:code, ds:code

inicio: mov ax,E000h    ; Direccionamos el puerto PPC a través
        mov ds,ax       ; del decodificador, enviando un valor
        mov si,0000h    ; de 0 por la localidad E000:0000.
        mov bl,00h
        mov [si],bl
        mov ax,2000h    ; Localidad de memoria PC a copiar a
        mov es,ax       ; disco flexible, el segmento se coloca
        mov bx,0000h    ; en el registro ES y el offset en el BX.
        mov al,01h      ; Para transferir un sector.
        mov dl,00h      ; Escribir al drive A.
        mov dh,01h      ; Cabeza 1.
        mov ch,25h      ; Track 25h = 37.
        mov cl,08h      ; Sector 8.
        mov ah,03h      ; funcion 03, realiza la escritura a disco
        int 13h         ; de un buffer de memoria PC.
        jb fin          ; Si falla la interrupción se va a fin.
        mov al,01       ; Transfiere un sector.
        mov ch,26h      ; Track 26h = 38.
```

```
     mov ah,03       ; Funcion 03, realiza la escritura a disco
     int 13h         ; de un buffer de memoria PC.
     jb fin          ; Si falla la interrupción se va a fin.
     mov ah,01h      ; Con esta función espera un teclazo para
     int 16h         ; terminar con el loop.
     jz inicio
fin: mov ax,4c00h    ; Sale a MSDOS
     int 21h
     code ends
     end inicio
```

El resultado obtenido fué:

T3 = 600 mseg.

Sí:

T(2 pistas) = 2 * T(1 pista) + **T(mov.cabeza)**

donde:

T(2 pistas) : tiempo total que tarda en escribir a dos pistas.

T(1 pista) : tiempo para escribir a una pista.

T(mov. cabeza): tiempo que tarda en desplazar la cabeza del disco de una pista a otra.

entonces:

T(2 pistas) = **600 mseg.**
T(1 pista) = **200 mseg.** (Del programa anterior)
T(mov.cabeza) = **200 mseg.**

El controlador de disco flexible tarda **200 mseg.** en mover la cabeza de una pista a otra contigua.

4) Rutina MIDE_HEAD2.ASM:

Con este programa se mide el tiempo que tarda la cabeza del controlador de disco duro en moverse de una pista a otra contigua. Este programa varía solo en algunas instrucciones del programa anterior, pues realiza las mismas funciones, solo que con disco duro.

Se emplea la interrupción 13H del BIOS.

MIDE_HEAD2.ASM

```
        code segment
        assume cs:code, ds:code

inicio: mov ax,E000h    ; Direccionamos el puerto PPC a través
        mov ds,ax       ; del decodificador, enviando un valor
        mov si,0000h    ; de 0 por la localidad E000:0000.
        mov bl,00h
        mov [si],bl
        mov ax,2000h    ; Localidad de memoria PC a copiar a
        mov es,ax       ; disco duro, el segmento se coloca
        mov bx,0000h    ; en el registro ES y el offset en el BX.
        mov al,01h      ; Para transferir un sector.
        mov dl,80h      ; Escribir en disco duro.
        mov dh,01h      ; Cabeza 1.
        mov ch,FEh      ; Track FEh = 254, track=cilindro.
        mov cl,0Ah      ; Sector A.
        mov ah,03h      ; funcion 03, realiza la escritura a disco
        int 13h         ; de un buffer de memoria PC.
        jb fin          ; Si falla la interrupción se va a fin.
        mov al,01       ; Transfiere un sector.
        mov ch,FFh      ; Track 26h = 38.
        mov ah,03       ; Funcion 03, realiza la escritura a disco
        int 13h         ; duro de un buffer de memoria PC.
        jb fin          ; Si falla la interrupción se va a fin.
        mov ah,01h      ; Con esta función espera un teclazo para
        int 16h         ; terminar con el loop.
        jz inicio
   fin: mov ax,4c00h    ; Sale a MSDOS
        int 21h

        code ends
        end inicio
```

El resultado obtenido fué:

T4 = 70 mseg.

Si:

T(2 pistas) = 2 * T(1 pista) + T(mov.cabeza)

donde:

T(2 pistas) : tiempo total que tarda en escribir a dos pistas.

T(1 pista) : tiempo para escribir a una pista.

T(mov. cabeza): tiempo que tarda en desplazar la cabeza del disco de una pista a otra.

entonces:

T(2 pistas) = 70 **mseg.**
T(1 pista) = 17 **mseg.** (Del programa anterior)
T(mov.cabeza) = 36 **mseg.**

El controlador de disco duro tarda entre 35 y 36 **mseg**, pues los valores son lo más aproximados posibles y tienen variantes de diezmilésimas de segundo.

5) Rutina TRANSFIERE_1.ASM:

Con este programa se mide el tiempo que le toma a la PC en transferir 512, 1024, y 2048 bytes de la memoria de conmutación (RAM1, RAM2) de la tarjeta de adquisición a la localidad 2000:0000 de memoria RAM de la PC.

TRANSFIERE_1.ASM

```
code segment
     assume cs:code, ds:code

inicio: mov ax,E000h    ; Direccionamos el puerto PPC a través
        mov ds,ax       ; del decodificador, enviando un valor
        mov si,0000h    ; de 0 por la localidad E000:0000.
        mov bl,00h
        mov [si],bl
        cld             ; Se limpia bandera de direccion para el
                          uso de movsb.
        mov ax,D000h    ; Dirección donde encontramos los valores
                          de la RAM de conmutación de la tarjeta.
                          de adquisición, esta es nuestra
                          dirección fuente.
        mov ds,ax
        mov si,0000h    ; Desde la localidad 0 (fuente)
        mov ax,2000h    ; Direccion destino.
        mov es,ax
        mov di,0000h    ; Desde la localidad 0
```

```
        mov cx,512      ; Se transfieren 512 bytes, este valor se
                          variará para realizar distintas
                          mediciones.
        rep movsb
        mov ah,01h      ; Funcion que espera un teclazo para
                          terminar el loop.
        int 16h
        jz inicio
        mov ax,4c00h    ; Sale a MSDOS
        int 21h

    code ends
        end inicio
```

Los valores que se obtuvieron fueron:

512 bytes se transfirieron en: 1.2 mseg.
1024 bytes se transfirieron en: 2.4 mseg.
2048 bytes se transfirieron en: 4.8 mseg.

6) Rutina TRANSFIERE_2.ASM:

Este programa es una variante del programa anterior. En él se efectúa una transferencia de un bloque de 32 Kbytes de la RAM de conmutación en la tarjeta de adquisición a un buffer relocalizable de la RAM de la PC, utilizando la instrucción de MOVSB.

TRANSFIERE_2.ASM:

```
    code segment
         assume cs:code, ds:code

inicio: mov ax,E000h    ; Direccionamos el puerto PPC a través
        mov ds,ax       ; del decodificador, enviando un valor
        mov si,0000h    ; de 0 por la localidad E000:0000.
        mov bl,00h
        mov [si],bl
        cld             ; Limpia bandera de direccion.
        mov ax,D000     ; Direccion de RAM de adquisición.
        mov ds,ax       ; Esta direccion fuente es D000.
        mov si,0000     ; Offset de direccion fuente.
        mov ax, seg buffer;  Direccion destino.
        mov es, ax      ; Segmento de localidad dentro de RAM PC.
```

```
        mov di, offset buffer; Offset de direccion destino.
        mov cx, 32767  ; No. de bytes a escribir de fuente
                         destino.
    rep mov sb         ; Instrucción que realiza transferencia.

    fin: mov ah,01h    ; funcion que espera un telcazo para
                         terminar
        int 16h        ;
        jz inicio      ;
        mov ax,4c00h   ; Sale a MSDOS
        int 21h

buffer  db 32767 dup (?)

    code ends
        end inicio
```

En este programa el resultado fué:

T5 = 80 mseg.

7) Rutina GENERA_ARCHIVO.ASM:

Con este programa se realiza la escritura de un bloque de 32 Kbytes tomados de la memoria de conmutación a disco duro empleando las funciones de creación, apertura, escritura y cerrado de un archivo ASCII de la interrupción 21H de MSDOS.

Cabe mencionar que para esta escritura es necesario hacer una transferencia previa de la RAM de adquisición, al buffer de memoria PC, esto es, se necesita agregar el tiempo que tarda una transferencia de RAM de adquisición a la RAM PC.

GENERA_ARCHIVO.ASM:

```
    code segment
        assume cs:code, ds:code

inicio: mov ax,E000h   ; Direccionamos el puerto PPC a través
        mov ds,ax      ; del decodificador, enviando un valor
        mov si,0000h   ; de 0 por la localidad E000:0000.
        mov bl,00h
```

```
        mov [si],bl
        xor ax,ax
        mov ah,3ch      ; funcion para crear archivo ASCII.
        xor cx,cx       ; limpia cx para archivo atributo normal.
        mov dx,seg archivo; direccion del segmento de archivo.
        mov ds,dx       ; segmento en ds.
        mov dx,offset archivo; offset del archivo.
        int 21h         ; interrupción que crea archivo.
        jnc sigue       ;
        jmp error       ; si no se realizó la creación.
 sigue: mov fhandle,ax  ; recupera el file handle.
        mov ah,3d       ; funcion para abrir un archivo ASCII.
        mov al,2        ; con atributo de lectura/escritura.
        mov dx,seg archivo;
        mov ds,dx       ;
        mov dx,offset archivo;
        int 21h         ;
        jnc siguel      ;
        jmp error       ; si no pudo abrirse el archivo.
siguel: mov fhandle,ax  ; recupera el file handle.
        mov ah,40       ; funcion para escribir en un archivo.
        mov bx,fhandle  ;
        mov cx,32767    ; numero de bytes a escribir.
        mov dx,seg buffer; direccion del buffer de memoria de
                          donde va a leer.
        mov ds,dx       ;
        mov dx,offset buffer;
        int 21h         ;

        jc error        ; si no pudo escribirse en el archivo.
        cmp ax,32767    ;
        jne lleno       ; si no es 32 767 es que disco esta lleno.
 lleno: nop             ; esta etiqueta despues se implementa.
        mov ah,3e       ; funcion para cerrar un archivo.
        mov bx,fhandle  ;
        int 21h         ; interrupcion para cerrar un archivo.
        jc error
        jmp fin
 error: xor bx,bx       ;
        mov ah,14h      ; realiza un sonido.
   fin: mov ah,01h      ; funcion que espera un telcazo para
                          terminar
        int 16h         ;
        jz inicio       ;
        mov ax,4c00h    ; Sale a MSDOS
        int 21h

archivo db 'c:\archivo1.dat',0
fhandle dw 0
buffer  db 32767 dup (?)
```

```
code ends
     end inicio
```

La creación, apertura, escritura y cerrado de un archivo se efectúo en:

T6 = 1.6 segundos.

La escritura unicamente de un bloque de 32 Kbytes se realiza en un tiempo menor a este, pues no tendría que crearse, abrirse y cerrarse el archivo cada vez que se efectúe la escritura.

A N E X O II.

Diagrama de tiempos y hoja de datos de la RAM 62256.

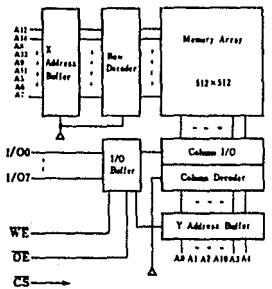
HM62256LP Series, HM62256LFP Series

32768-word × 8-bit High Speed CMOS Static RAM

■ FEATURES

- High Speed: Fast Access Time 85/100/120/150ns (max)
- Low Power Standby and Low Power Operation;
  Standby: 200 μW (typ.), 10 μW (typ.) (L-version)
  Operation: 40mW (typ.) (f = 1MHz)
- Single 5V Supply
- Completely Static RAM; No Clock or Timing Strobe Required
- Equal Access and Cycle Time
- Common Data Input and Output, Three-state Output
- Directly TTL Compatible: All Input and Output
- Standard 28 Pin Package Configuration
- Capability of Battery Backup Operation

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

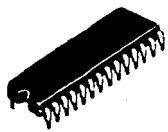
Item	Symbol	Rating	Unit
Voltage on any pin with relative to GND	V_T	−0.5* to +7.0	V
Power Dissipation	P_T	1.0	W
Operating Temperature	T_{opr}	0 to +70	°C
Storage Temperature	T_{stg}	−55 to +125	°C
Temperature Under Bias	T_{bias}	−10 to +85	°C

*Pulse Width 50ns: −3.0V

■ TYPES OF PRODUCTS

Type No	Access	Package
HM62256P-8	85 ns	600 mil 28 pin plastic DIP
HM62256P-10	100 ns	
HM62256P-12	120 ns	
HM62256P-15	150 ns	
HM62256LP-8	85 ns	
HM62256LP-10	100 ns	
HM62256LP-12	120 ns	
HM62256LP-15	150 ns	
HM62256FP-8	85 ns	28 pin plastic SOP
HM62256FP-10	100 ns	
HM62256FP-12	120 ns	
HM62256FP-15	150 ns	
HM62256LFP-8	85 ns	
HM62256LFP-10	100 ns	
HM62256LFP-12	120 ns	
HM62256LFP-15	150 ns	

HM62256LP Series

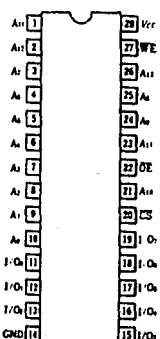


(DP-28)

HM62256LFP Series

(FP-28D)

■ PIN ARRANGEMENT



(Top View)

■ TRUTH TABLE

$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	Mode	V_{CC} Current	I/O Pin	Reference Cycle
H	x	x	Not Selected	I_{SB}, I_{SB1}	High Z	–
L	L	H	Read	I_{CC}	Dout	Read Cycle No. 1~3
L	H	L	Write	I_{CC}	Din	Write Cycle No. 1
L	L	L	Write	I_{CC}	Din	Write Cycle No. 2

X means H or L

■ RECOMMENDED DC OPERATING CONDITIONS (T_a = 0 to +70°C)

Item	Symbol	min.	typ.	max.	Unit
Supply Voltage	V_{CC}	4.5	5.0	5.5	V
	GND	0	0	0	V
Input Voltage	V_{IH}	2.2	–	6.0	V
	V_{IL}	−3.0*	–	0.8	V

*Pulse Width, 50ns. DC: V_{IL} min = −0.5V

■ DC AND OPERATING CHARACTERISTICS (V_{CC}=5V± 10%, GND=0V, T_a=0 to +70°C)

Item		Symbol	Test Condition	min.	typ.*	max.	Unit
Input Leakage Current		$\|I_{LI}\|$	V_{IN}=GND to V_{CC}	–	–	2	μA
Output Leakage Current		$\|I_{LO}\|$	$\overline{CS}$=V_{IH} or $\overline{OE}$=V_{IH}, $V_{I/O}$=GND to V_{CC}	–	–	2	μA
Operating Power Supply Current		I_{CC}	$\overline{CS}$=V_{IL}, $I_{I/O}$=0mA	–	8	15	mA
Average Operating Power Supply Current	HM62256LP/LFP-8	I_{CC1}	Min. Cycle, duty=100%, $I_{I/O}$=0mA	–	50	70	mA
	HM62256LP/LFP-10			–	40	70	
	HM62256LP/LFP-12			–	35	70	
	HM62256LP/LFP-15			–	33	70	
		I_{CC2}	$\overline{CS}$=V_{IL}, V_{IH}=V_{CC}, V_{IL}=0V, $I_{I/O}$=0mA, f=1MHz	–	8	15	mA
Standby Power Supply Current		I_{SB}	$\overline{CS}$=V_{IH}	–	0.5	3	mA
		I_{SB1}	$\overline{CS}$ ≥ V_{CC} −0.2V	–	2	100	μA
Output Voltage		V_{OL}	I_{OL}=2.1mA	–	–	0.4	V
		V_{OH}	I_{OH}=−1.0mA	2.4	–	–	V

Notes: *: Typical values are at V_{CC}=5.0V, T_a=25°C and specified loading.

■ CAPACITANCE (T_a=25°C, f=1MHz)

Item	Symbol	Test Condition	typ.	max.	Unit
Input Capacitance	C_{in}	V_{IN}=0V	–	6	pF
Input/Output Capacitance	$C_{I/O}$	$V_{I/O}$=0V	–	8	pF

Note) This parameter is sampled and not 100% tested.

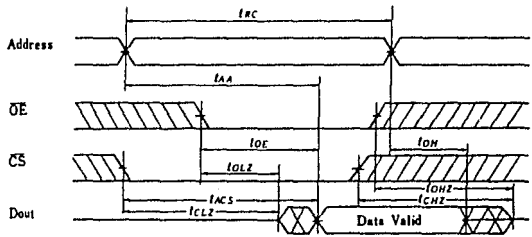
■ AC CHARACTERISTICS (V_{CC}=5V± 10%, T_a=0 to +70°C unless otherwise noted)

- **AC Test Conditions**
  - Input pulse levels: 0.8V to 2.4V
  - Input rise and fall times: 5ns
  - Input and Output timing reference levels: 1.5V
  - Output load: 1TTL Gate and C_L = 100pF (Including scope and jig)

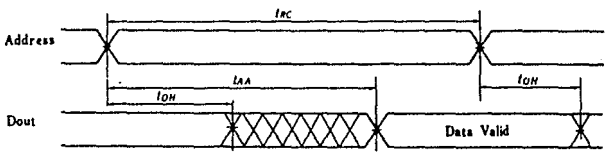
- **Read Cycle**

Item	Symbol	HM62256LP LFP-8		HM62256LP/LFP-10		HM62256LP/LFP-12		HM62256LP-LFP-15		Unit
		min	max.	min.	max.	min.	max.	min	max.	
Read Cycle Time	t_{RC}	85	–	100	–	120	–	150	–	ns
Address Access Time	t_{AA}	–	85	–	100	–	120	–	150	ns
Chip Select Access Time	t_{ACS}	–	85	–	100	–	120	–	150	ns
Output Enable to Output Valid	t_{OE}	–	45	–	50	–	60	–	70	ns
Output Hold from Address Change	t_{OH}	5	–	10	–	10	–	10	–	ns
Chip Selection to Output in Low Z	t_{CLZ}	10	–	10	–	10	–	10	–	ns
Output Enable to Output in Low Z	t_{OLZ}	5	–	5	–	5	–	5	–	ns
Chip Deselection to Output in High Z	t_{CHZ}	0	30	0	35	0	40	0	50	ns
Output Disable to Output in High Z	t_{OHZ}	0	30	0	35	0	40	0	50	ns

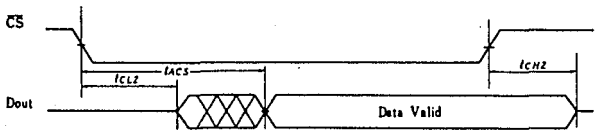
- **Timing Waveform of Read Cycle No. 1**[1]



- **Timing Waveform of Read Cycle No. 2**[1][3][4]



• Timing Waveform of Read Cycle No. 3[1][3][4]

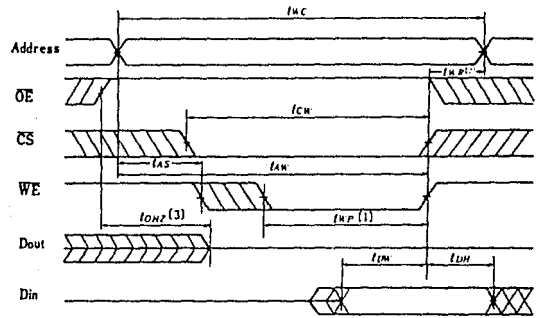


Notes) 1. WE is High for Read Cycle.
2. Device is continuously selected, $\overline{CS} = V_{IL}$.
3. Address Valid prior to or coincident with CS transition Low.
4. $\overline{OE} = V_{IL}$.

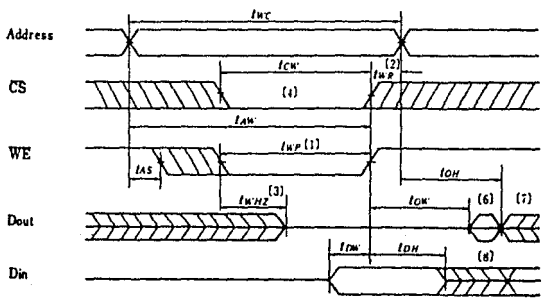
• Write Cycle

Item	Symbol	HM62256-8		HM62256-10		HM62256-12		HM62256-15		Unit
		min	max	min	max	min	max	min	max	
Write Cycle Time	t_{WC}	85	—	100	—	120	—	150	—	ns
Chip Selection to End of Write	t_{CW}	75	—	80	—	85	—	100	—	ns
Address Valid to End of Write	t_{AW}	75	—	80	—	85	—	100	—	ns
Address Setup Time	t_{AS}	0	—	0	—	0	—	0	—	ns
Write Pulse Width	t_{WP}	60	—	60	—	70	—	90	—	ns
Write Recovery Time	t_{WR}	10	—	0	—	0	—	0	—	ns
Write to Output in High Z	t_{WHZ}	0	30	0	35	0	40	0	50	ns
Data to Write Time Overlap	t_{DW}	40	—	40	—	50	—	60	—	ns
Data Hold from Write Time	t_{DH}	0	—	0	—	0	—	0	—	ns
Output Disable to Output in High Z	t_{OHZ}	0	30	0	35	0	40	0	50	ns
Output Active from End of Write	t_{OW}	5	—	5	—	5	—	5	—	ns

- Timing Waveform of Write Cycle No. 1 ($\overline{OE}$ Clock)



- Timing Waveform of Write Cycle No. 2[8] ($\overline{OE}$ Low Fixed)



Notes: 1. A write occurs during the overlap (t_{WP}) of a low $\overline{CS}$ and a low $\overline{WE}$.
2. t_{WR} is measured from the earlier of $\overline{CS}$ or $\overline{WE}$ going high to the end of write cycle.
3. During this period, I/O pins are in the output state. The input signals out of phase must not be applied
4. If the $\overline{CS}$ low transition occurs simultaneously with the $\overline{WE}$ low transition or after the $\overline{WE}$ low transition, outputs remain in a high impedance state.
5. $\overline{OE}$ is continuously low. ($\overline{OE} = V_{IL}$)
6. Dout is in the same phase of written data of this write cycle.
7. Dout is the read data of next address.
8. If $\overline{CS}$ is low during this period, I/O pins are in the output state. The input signals out of phase must not be applied to I/O Pins.

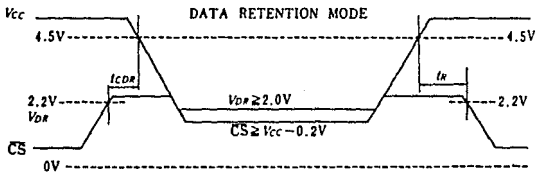
■ LOW V_{CC} DATA RETENTION CHARACTERISTICS (T_A = 0 to 70°C)

These characteristics are guaranteed only for L version.

Item	Symbol	Test Conditions	min.	typ.	max	Unit
V_{CC} for Data Retention	V_{DR}	$\overline{CS} \geq V_{CC} - 0.2V$	2.0	–	–	V
Data Retention Current	I_{CCDR}	V_{CC} = 3.0V, $\overline{CS} \geq 2.8V$	–	–	50	µA
Chip Deselect to Data Retention Time	t_{CDR}	See Retention Waveform	0	–	–	ns
Operation Recovery Time	t_R		t_{RC}*	–	–	ns

* t_{RC} = Read Cycle Time

● Low V_{CC} Data Retention Waveform



NOTE) In data retention mode, $\overline{CS}$ controls the address, $\overline{WE}$, $\overline{OE}$ and D_{IN} Buffers. V_{IN} for these inputs can be in high impedance state in data retention mode.

A N E X O III.

Listado del programa del MPA.

```
CODE_HERE SEGMENT WORD PUBLIC
    ASSUME CS:CODE_HERE, DS:CODE_HERE

        NUESTRA     DB      00
        RAM         DW      0000
        ALARMA      DB      00
        CANAL       DW      0000
        CUENTA      DW      0000



    START:  CALL INICIA         ;Dirección rutina NMI,
                                ;direccionc stack
            CALL LIMPIEZA       ;Rutina de limpieza al
                                ;inicializarse el sistema
            CALL LIMPIA_RAM3    ;Rutina de limpieza sumatorias
                                ;RAM 3 (700H a 800H)

;-------------------------------TERMINACION DEL "BOOT"

  POLEA_RAMS:  CALL DIR_RAM1    ;MPA apunta a RAM1
               MOV SI,0000H
               MOV DX,[SI]      ;Carga registro con
                                ;contenido de Loc. 00H
               CMP DX,+00H
               JNZ BBB          ;Si no es cero indica
                                ;terminación "BOOT"
               CALL DIR_RAM2    ;MPA apunta a RAM2
               MOV SI,0000H
               MOV DX,[SI]      ;Carga registro con contenido
                                ;de Loc. 00H
               CMP DX,+00H
               JZ POLEA_RAMS    ;Si es cero sigue revisando
;-------------------------------ESPERA DE CODIGO DE COMANDO

        BBB:   CALL DIR_RAM2    ;Asegura datos en RAM1 [0070]
      BEGIN:   CALL POLEA_PTOPC
;-------------------------------CODIGO DE RESPUESTA

   COD_RESP:   CALL DIR_RAM1    ;MPA apunta a RAM1
               MOV CX,0000H
               MOV SI,0001H
               MOV [SI],DI      ;Manda CR a Loc. 01H
               CALL DIR_RAM2    ;MPA apunta a RAM2
               MOV SI,0001
               MOV [SI],DI      ;Manda CR a Loc. 01H
               MOV AX,2800H
               MOV DS,AX        ;Carage DS con la dirección
                                ;de PPC
               MOV CX,0000H
               MOV BH,00
               MOV SI,0000
    EWAITT:    MOV CL,[SI]      ;Carga el CC leído de PPC
               CMP CL,00
               JNZ EWAITT       ;Espera que PPC sea limpiado
                                ;por la PC

;-------------------------------ELECCION DE RUTINAS
      TABLA:   CMP DI,0002H
               JZ CLEAN         ;Limpia memorias 1 y 2
               CMP DI,03H
               JZ CLEAN_ZP      ;Limpia Zona de parámetros
               CMP DI,0015H
               JZ REV_RAMS      ;Revisa RAM's
               CMP DI,0021H
               JZ ESC_R1
               CMP DI,0023H
               JZ L_R1
```

```
                CMP DI,0031H
                JZ ESC_R2
                CMP DI,0033H
                JZ L_R2
                CMP DI,0040H
                JZ ADC
                JMP BBB
        ESC_R1: JMP ESCRIBE_R1    ;Escritura en RAM1
          L_R1: JMP LEE_R1        ;MPA apunta a RAM2
        ESC_R2: JMP ESCRIBE_R2    ;Escritura en RAM2
          L_R2: JMP LEE_R2        ;MPA apunta a RAM1
           ADC: CALL ETIQUETAS
                CALL DIR_RAM1
                MOV SI,0030H
                MOV BL,[SI]
                MOV AX,0000H
                MOV DS,AX
                MOV BYTE PTR [DS:0500H],BL
                CALL LIMPIA_RAM3
                CALL ADQUIS_ADC
                CALL PASS
                JMP BEGIN
;---------------------------------------------------------
     *    CLEAN: CALL LIMPIEZA     ;Limpia memorias 1 y 2
                CALL TERMINE_SR    ;Manda CT
                JMP BEGIN
;---------------------------------------------------------
       CLEAN_ZP: CALL LIMPIA_ZP    ;Limpia zona de parámetros
                CALL TERMINE_SR    ;manda CT
                JMP BEGIN
;---------------------------------------------------------
      REV_RAMS: CALL DIR_RAM1     ;MPA apunta a RAM1
                MOV SI,0000H
                MOV CX,0000H
                MOV BX,3FFFH      ;Carga capacidad de memoria
      SIGUE_R1: MOV [SI],SI       ;Escribe en localidad
                MOV AX,[SI]       ;Lee de localidad
                CMP AX,SI         ;Compara lo escrito y leido
                JNZ RAM1_MAL
                INC SI
                INC SI
                DEC BX
                JNZ SIGUE_R1
      RAM1_BIEN: MOV CL,1FH       ;carga respuesta
       REV_RAM2: CALL DIR_RAM2
                MOV SI,0000H
                MOV BX,3FFFH
      SIGUE_R2: MOV [SI],SI
                MOV AX,[SI]
                CMP AX,SI
                JNZ RAM2_MAL
                INC SI
                INC SI
                DEC BX
                JNZ SIGUE_R2
      RAM2_BIEN: MOV CH,2FH       ;carga respuesta
                JMP DESPLIEGUE
       RAM1_MAL: MOV CL,1BH       ;carga respuesta
                JMP REV_RAM2
       RAM2_MAL: MOV CH,2BH       ;carga respuesta
     DESPLIEGUE: CALL DIR_RAM1    ;MPA apunta a RAM1
                MOV SI,0003H      ;* manda respuesta de RAM1
                MOV [SI],CL       ;a loc 03H
                MOV SI,0004H      ;* manda respuesta de RAM2
                MOV [SI],CH       ;a loc 04H
                MOV DH,00A2H
                MOV SI,0002H
                MOV [SI],DH       ;* manda CT a loc 02H
                CALL DIR_RAM2     ;MPA apunta a RAM2
                MOV SI,0003H
```

```
                MOV [SI],CL         ;*  Ident
                MOV SI,0004H
                MOV [SI],CH
                MOV DH,00A2H
                MOV SI,0002H
                MOV [SI],DH
                JMP BEGIN

;-----------------------------------ESCRITURA Y LECTURA A RAM'S

                                    ;ESCRITURA EN RAM1

     ESCRIBE_R1: CALL DIR_RAM1      ;MPA apunta a RAM1
                 MOV SI,0022H
                 MOV DI,0020H
                 MOV CX,[SI]        ;recoge # de veces a escribir
                                    ;de loc 22H
                 MOV BX,[DI]        ;recoge # a escribir de loc 20H
                 MOV DI,0024H       ;recoge loc a partir de la cual
                 MOV SI,[DI]        ;se inicia a escribir, de loc 24H
    ESCRIBE_MAS: MOV [SI],BX
                 INC SI
                 INC SI
                 DEC CX
                 JNZ ESCRIBE_MAS
                 CALL TERMINE_SR    ;manda código de terminación
                 JMP BEGIN

;-----------------------------------DESPLIEGUE DE RAM1
                                     ;MPA POSICIONADO EN RAM2

        LEE_R1:  MOV AX,2400H       ;direcciona  PTO.SAL
                 MOV DS,AX
                 MOV DL,05H         ; LC = 1
                 MOV SI,0000
                 MOV [SI],DL        ;manda LC (MPA apunta a RAM2)
                 CALL TERMINE_SR    ;manda CT
                 JMP BEGIN

;-----------------------------------ESCRITURA EN RAM2

     ESCRIBE_R2: CALL DIR_RAM1      ;MPA apunta a RAM1
                 MOV SI,0000
                 MOV DI,0000
                 MOV BX,0000
                 MOV CX,0000
                 MOV SI,0022H
                 MOV DI,0020H
                 MOV CX,[SI]        ;recoge # de veces a escribir
                 MOV BX,[DI]        ;recoge # a escribir
                 MOV DI,0024H       ;recoge loc a partir de la cual
                 MOV SI,[DI]        ;se inicia a escribir, de loc 24H
                 CALL DIR_RAM2      ;MPA apunta a RAM2
           AQUI: MOV [SI],BX
                 INC SI
                 INC SI
                 DEC CX
                 JNZ AQUI
                 CALL TERMINE_SR    ;manda código de terminación
                 CALL DIR_RAM1
                 JMP BEGIN
;-----------------------------------DESPLIEGUE DE RAM2
                                     ;MPA POSICIONADO EN RAM1

        LEE_R2:  CALL TERMINE_SR    ;manda código de terminación
                 MOV AX,2400H       ;direcciona PTO.SAL
                 MOV DS,AX
                 MOV DL,04          ; LC = 0
                 MOV SI,0000
                 MOV [SI],DL        ;manda LC (MPA apunta a RAM1)
                 JMP BEGIN
```

```
;-------------------------------------------------------
;          S U B R U T I N A S


       DIR_RAM1  PROC NEAR
                 PUSH AX
                 MOV AX,2400H                  ;direcciona PTO.SAL
                 MOV DS,AX
                 MOV BYTE PTR [DS:0000H],04H   ;LC=0
                 MOV AX,2222H                  ;direcciona RAM1
                 MOV DS,AX
                 POP AX
                 RET
       DIR_RAM1  ENDP
;-------------------------------------------------------
       DIR_RAM2  PROC NEAR
                 PUSH AX
                 MOV AX,2400H                  ;direcciona PTO.SAL
                 MOV DS,AX
                 MOV BYTE PTR [DS:0000H],05H   ;LC=1
                 MOV AX,3333H                  ;direcciona RAM2
                 MOV DS,AX
                 POP AX
                 RET
       DIR_RAM2  ENDP
;-------------------------------------------------------
    POLEA_PTOPC  PROC NEAR
                 MOV AX,2800H        ;direcciona PTO.PC
                 MOV DS,AX
                 MOV BH,00
                 MOV SI,0000
           POPC: MOV BL,[SI]         ;poleo PPC
                 MOV DI,BX
                 CMP DI,00           ;espera CC diferente de cero
                 JZ POPC
                 RET
     POLEA_PTOPC ENDP
;-------------------------------------------------------
       LIMPIEZA  PROC NEAR
                 CALL DIR_RAM1       ;direcciona RAM1
                 MOV BX,7FFFH        ;limpia toda la memoria1
                 MOV SI,0000
                 MOV CL,00
            MORE: MOV [SI],CL
                 INC SI
                 DEC BX
                 JNZ MORE
                 CALL DIR_RAM2       ;direcciona RAM2
                 MOV BX,7FFFH        ;limpia toda la memoria2
                 MOV CL,00
                 MOV SI,0000
             MAS: MOV [SI],CL
                 INC SI
                 DEC BX
                 JNZ MAS
                 RET
       LIMPIEZA  ENDP
;-------------------------------------------------------
     LIMPIA_RAM3 PROC NEAR
                 PUSH DS
                 PUSH SI
                 MOV AX,0000H
                 MOV DS,AX
                 MOV SI,0700H
           JR3:  MOV WORD PTR [DS:SI],0000H
                 CMP SI,0720H
                 JGE LSR3
                 INC SI
                 INC SI
                 JMP JR3
```

```
          LSR3:  POP SI
                 POP DS
                 RET
     LIMPIA_RAM3 ENDP
;-------------------------------------------------
       LIMPIA_ZP PROC NEAR
                 CALL DIR_RAM1      ;direcciona RAM1
                 MOV BX,0040H       ;limpia Zona de parámetros1 (128 bytes)
                 MOV SI,0000H
                 MOV CX,0000H
          BRINCA: MOV [SI],CX
                 INC SI
                 INC SI
                 DEC BX
                 JNZ BRINCA
                 CALL DIR_RAM2      ;direcciona RAM2
                 MOV BX,0040H       ;limpia Zona de parámetros2 (128 bytes)
                 MOV CX,0000
                 MOV SI,0000H
          SALTA: MOV [SI],CX
                 INC SI
                 INC SI
                 DEC BX
                 JNZ SALTA
                 RET
       LIMPIA_ZP ENDP

;-------------------------------------------------
      TERMINE_SR PROC NEAR
                 CALL DIR_RAM1      ;direcciona RAM1
                 MOV BYTE PTR [DS:0002H],00ADH
                 CALL DIR_RAM2      ;direcciona RAM2
                 MOV BYTE PTR [DS:0002H],00A0H
                 RET
      TERMINE_SR ENDP
;-------------------------------------------------
      ADQUIS_ADC PROC  NEAR
                 PUSH AX
                 PUSH BX
                 PUSH CX
                 PUSH DX
                 PUSH SI

                 MOV AX,0000H           ;       INICIO:
                 MOV DS,AX              ;
                 MOV ALARMA,00H         ;
                 MOV MUESTRA,64
                 MOV CUENTA,0800

                 MOV BX,5555H
                 MOV WORD PTR [DS:000AH],BX
                 MOV WORD PTR [DS:0008H],OFFSET LEE_ADC

     LLENA_RAM1: CALL VARIA_RAM1

                 MOV SI,0000H
                 MOV DI,0080H
                 CALL RAM1_INT
           LASDO: MOV DX,0720H
             OGO: MOV SI,0000H
                 MOV DS,4000H
                 DEC DX
                 NLT
                 MOV AX,0000H
                 MOV DS,AX
                 CMP ALARMA,01H
                 JZ DDEE
                 JMP FETT
            DDEE: DEC CUENTA
                 JZ  ACABA
```

```
          FETT:  CALL SUMAS
                 DEC MUESTRA
                 CMP MUESTRA,00
                 JZ CHEQQ
          IGOR:  MOV AX,0000H
                 MOV DS,AX
                 CMP RAM,3333H
                 JZ RARA1

                 CMP DX,0000H
                 JE LLENA_RAM2
                 JMP OFFO
          ACABA: CALL FINAL_POINT
                 JMP IGOR
          RARA1: CMP DX,0000H
                 JE LLENA_RAM1

           OFFO: JMP OGO
          CHEQQ: CALL CHEQ_ALARM
          ALTOO: MOV AX,2800H              FIN ADQ ?
                 MOV+DS,AX
                 MOV SI,0000
                 MOV AL,[SI]
                 CMP AL,41H
                 JZ SSTTOP
          OGORO: CALL LIMPIA_RAM3
                 JMP OGO
      LLENA_RAM2: CALL VARIA_RAM2
                 CALL RAM2_INT
                 MOV DI,0080H
                 JMP LASDO
          MOMM1: MOV AX,2400H
                 MOV DS,AX
                 MOV BYTE PTR [DS:0000H],05H

          SSTTOP: POP SI
                 POP DX
                 POP CX
                 POP BX
                 POP AX
                 RET
       ADQUIS_ADC ENDP
;-------------------------------------------------------
       CHEQ_ALARM PROC NEAR
                 PUSH BX
                 PUSH CX
                 PUSH DX
                 PUSH SI
                 CALL ETIQUETAS
                 MOV AX,0000
                 MOV DS,AX
                 MOV MUESTRA,64
                 MOV BH,0064        ;dividendo
                 MOV SI,500H
                 MOV BL,[SI]        ;umbral
                 CMP ALARMA,01H
                 JZ CHAN
                 JMP NORM
            CHAN: MOV SI,CANAL
                 JMP HAYH
            NORM: MOV SI,0700H
            HAYH: MOV AX,[SI]
                 DIV BH
                 CMP AL,BL
                 JGE ENERGRAM1
                 CMP ALARMA,01H
                 JZ FIUU
                 INC SI
                 INC SI
                 CMP SI,071EH
```

```
                JZ FIUU
                JMP MAYH

      EMERGRAM1: MOV CUENTA,800
                CMP ALARMA,01H
                JZ FIUU
                MOV DX,RAM
                MOV ALARMA,01H
                MOV CX,SI
                MOV CANAL,CX
                CALL RAM2_INT
                MOV AX,3333H
                MOV DS,AX
                MOV BYTE PTR [DS:0020H],00A1H  ;ALARMA
                MOV SI,0021H
                MOV [SI],DI                    ;APUNTADOR INIC
                MOV SI,0023H
                MOV [SI],DX                    ;RAM DE DISPARO
                MOV SI,0028H
                MOV [SI],CX                    ;CANAL DISPARADO
                CALL RAM1_INT
                MOV AX,2222H
                MOV DS,AX
                MOV BYTE PTR [DS:0020H],00A1H
                MOV SI,0021H
                MOV [SI],DI
                MOV SI,0023H
                MOV [SI],DX
                MOV SI,0028H
                MOV [SI],CX
           SISI: CMP RAM,2222H
                JZ AAA
                CALL INT2_PC
                JMP FIRT
            AAA: CALL INT1_PC
                JMP FIRT
           FIUU: CMP RAM,2222H
                JZ CCC
                CALL RAM2_INT
                JMP FIRT
            CCC: CALL RAM1_INT
           FIRT: POP SI
                POP DX
                POP CX
                POP BX
                RET
      CHEQ_ALARM ENDP
;-------------------------------------------------------
          INICIA PROC NEAR
                MOV AX,1000H          ; SS = 1000H
                MOV SS,AX
                MOV AX,0400H
                MOV SP,AX
                RET
          INICIA ENDP
;-------------------------------------------------------
   ETIQUETAS PROC NEAR
             CALL DIR_RAM2
             MOV BYTE PTR [DS:000FH],00D2H
             CALL DIR_RAM1
             MOV BYTE PTR [DS:000FH],00D1H
             RET
   ETIQUETAS ENDP
;-------------------------------------------------------
            LEE_ADC PROC NEAR
                    PUSHF

                    CLD
                    MOV CX,0010H
                AP: NOP
```

```
          II:  MOV BYTE PTR [DS:SI],00
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               NOP
               WAIT
               NOP
               NOP
               MOVSB
               NOP
               NOP
               NOP
               NOP
               LOOP II

               POPF
               IRET
     LEE_ADC ENDP

;-------------------------------------------------------
        SUMAS  PROC NEAR
               PUSH DX

               MOV BH,00        ;contador de 16 canales
               MOV SI,DI
               PUSH DI
               MOV AX,0000H
               MOV DS,AX
               MOV AX,0700H
               MOV DI,AX
               MOV AX,RAM
               MOV DS,AX
               SUB SI,0010H
      PROCH:   MOV DX,[SI]

     ;.................RECTIFICACION.....

               MOV CL,80H       ;CARGA '0' DIGITAL
               CMP DL,80H
               JBE SS1
               SUB DL,CL
               MOV CL,80H
               JMP SS3
        SS1:   SUB CL,DL
               MOV DL,CL
               MOV CL,80H

        SS3:   CMP DH,80H
               JBE SS2
               SUB DH,CL
               MOV CL,80H
               JMP SS4
        SS2:   SUB CL,DH
               MOV DH,CL
```

```
;.....SLECTOR DE CANAL A SUMAR

      SS4:  MOV AX,0000H
            MOV DS,AX
;.....SUMAS EN RAM3

            MOV AH,0000H
            MOV AL,DL
            ADD [DI],AX
            INC DI
            INC DI
            MOV AH,0000H
            MOV AL,DH
            ADD [DI],AX
            INC DI
            INC DI

 TOUT_CHAN: CMP BH,07H
            JZ FI
            INC BH
            INC BI
            INC SI
            MOV AX,RAM
            MOV DS,AX
            JMP PROCH

       FI:  POP DI
            POP DX
            RET
     SUMAS  ENDP
;-------------------------------------------------------
 FINAL_POINT PROC NEAR
            MOV AX,0000H
            MOV DS,AX
            MOV DX,RAM
            MOV ALARMA,00H
            MOV CANAL,0000H
            MOV CUENTA,800
            CALL RAM2_INT
            MOV AX,3333H
            MOV DS,AX
            MOV BYTE PTR [DS:0030],00A2H
            MOV SI,0031H
            MOV [SI],DI      ;APUNTADOR FINAL
            CALL RAM1_INT
            MOV AX,2222H
            MOV DS,AX
            MOV BYTE PTR [DS:0030H],00A2H
            MOV SI,0031H
            MOV [BI],DI
            RET
 FINAL_POINT ENDP
;-------------------------------------------------------
  VARIA_RAM2 PROC NEAR
            MOV AX,0000H
            MOV BH,AX
            MOV RAM,3333H
            MOV AX,3333H
            MOV ES,AX
            RET
  VARIA_RAM2 ENDP
;-------------------------------------------------------
  VARIA_RAM1 PROC NEAR
            MOV AX,0000H
            MOV DS,AX
            MOV RAM,2222H
            MOV AX,2222H
            MOV ES,AX
            RET
  VARIA_RAM1 ENDP
```

```
;-------------------------------------------------------------

        RAM1_INT  PROC NEAR
                  PUSH DS
                  MOV AX,2400H
                  MOV DS,AX
                  MOV BYTE PTR [DS:0000],00H
                  POP DS
                  RET
        RAM1_INT  ENDP
;-------------------------------------------------------------
        RAM2_INT  PROC NEAR
                  PUSH DS
                  MOV AX,2400H
                  MOV DS,AX
                  MOV BYTE PTR [DS:0000],01H
                  POP DS
                  RET
        RAM2_INT  ENDP
;-------------------------------------------------------------
         INT1_PC  PROC NEAR
                  PUSH DS
                  MOV AX,2400H
                  MOV DS,AX
                  MOV BYTE PTR [DS:0000],02H
                  POP DS
                  RET
         INT1_PC  ENDP
;-------------------------------------------------------------
         INT2_PC  PROC NEAR
                  PUSH DS
                  MOV AX,2400H
                  MOV DS,AX
                  MOV BYTE PTR [DS:0000],03H
                  POP DS
                  RET
         INT2_PC  ENDP
;-------------------------------------------------------------
            PASS  PROC NEAR
                  CALL DIR_RAM1
                  MOV BYTE PTR [DS:000CH],00F1H
                  CALL DIR_RAM2
                  MOV BYTE PTR [DS:000CH],00F2H
                  RET
             PASS ENDP
;-------------------------------------------------------------
  CODE_HERE   ENDS
              END START
```

A N E X O IV.

Sistema de interrupciones de la PC.

A N E X O IV.

Se explicará el sistema de interrupciones de la computadora PC utilizado en la implementación de la rutina de interrupción, a la cuál se hace referencia en el capítulo VI.

1. Sistema de interrupciones de la computadora PC.

Una interrupción es generada en una computadora cuando un dispositivo requiere los servicios del microprocesador, pues de lo contrario, si no se utiliza la interrupción, el microprocesador periódicamente preguntaría a todos sus dispositivos, si es que requieren de su servicio y esto significaría un desperdicio del microprocesador.

Para hacer más eficiente el sistema de interrupciones dentro de una computadora, existe un manejador (sistema de procesamiento de interrupciones de una computadora PC), el cual se muestra en la figura 1.

La interrupción no mascarable es la de más alta prioridad y no puede ser ignorada. Se genera por medio de flip-flops D y compuertas AND en la mayoría de los diseños de microprocesadores. En el caso del microprocesador 8088 se tienen dos entradas de interrupción disponibles: una entrada de interrupción mascarable (INTR) y otra de interrupción no mascarable (NMI).

La interrupción INTR está a cargo del dispositivo 8259, llamado también controlador de interrupciones programable (PIC). El PIC es un sofisticado dispositivo que permite un máximo de 8 fuentes de requerimiento de interrupción (IRQ0 - IRQ7). Cuando dos dispositivos diferentes requieren al mismo tiempo del servicio de interrupción, el PIC pasará el requerimiento al microporcesador, de aquel dispositivo al cual se le asignó la más alta prioridad y atenderá al segundo hasta haber terminado con el primer dispositivo. En la PC, el PIC es programado para dar a IRQ0 la más alta prioridad y a IRQ7 la más baja.

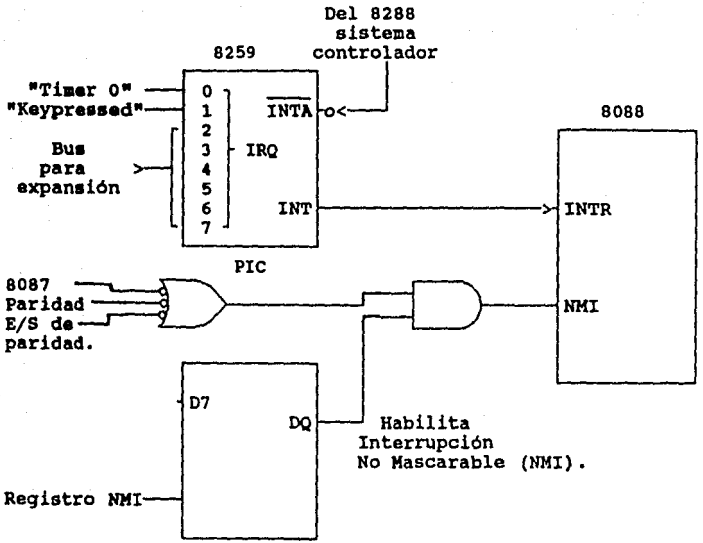


Figura 1. Sistema de interrupción de PC.

Las interrupciones IRQ0 e IRQ1 no están disponibles al usuario. La primera es utilizada en el "refresco de memoria" cada 100 mseg. e IRQ1 se efectúa cada vez que se presiona una tecla.

Las siguientes 6 interrupciones (IRQ2 - IRQ7) de entrada al PIC son utilizadas por las ranuras de expansión. La tabla 1 indica cómo son asignadas generalmente (más no exclusivamente) estas interrupciones.

^ ^	Nivel de Interrupción		Uso
P R I O R I D A D ^ ^	Tienen el más alto nivel de prioridad. No disponibles al usuario.	NMI	RAM que maneja paridad de teclado
		IRQ0	Entrada/Salida del canal paridad. Coprocesador numérico 8087. Sistema de tiempos.
		IRQ1	Teclado.
	Disponibles en el bus del sistema. Disponibles al usuario.	IRQ2	Reservado.
		IRQ3	Puerto serie COM2.
		IRQ4	Puerto serie COM1.
		IRQ5	Controlador de disco duro.
		IRQ6	Controlador de disco flexible.
		IRQ7	Puerto paralelo LPT1.

Tabla 1. Tabla de interrupciones de la PC.

Al utilizar una interrupción se debe asegurar que ésta no esté asignada anteriormente. Esto no resultará un peligro para el hardware, pero causará problemas de funcionamiento al software.

El 8259 (PIC) consta de un bus de datos de 8 bits, un bit de direcciones y comandos de entrada como se ilustra en la figura 2.

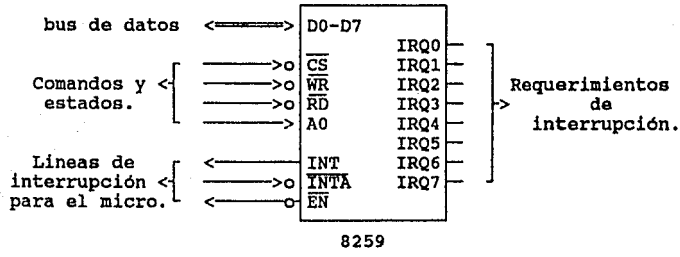


Figura 2. Controlador de interrupciones programable.

La salida INT del 8259 maneja la entrada INTR del micro 8088 y la salida $\overline{\text{INTA}}$ del 8288 maneja la entrada $\overline{\text{INTA}}$ del 8259. La señal $\overline{\text{EN}}$ es una salida usada en algunos sistemas para habilitar buffers del bus de datos para que el microprocesador pueda leer un vector de interrupciones desde el 8259, pero en este caso la PC no utiliza esta señal.

El PIC o dispositivo 8259 está formado internamente por tres registros: el registro de requerimiento de interrupción (IRR), que es el que recibe las entradas IRQ, el registro de interrupciones no mascarables (IMR), en el cual se habilita o deshabilitan las interrupciones IRQ como mascarables o no, y el registro de servicio de interrupción (ISR) que contiene el tipo de interrupción que se está atendiendo en un momento dado.

El primer Kbyte de memoria RAM de la PC esta dedicado a guardar el vector de interrupciones. Son 256 interrupciones las que forman el vector de interrupciones, cada una de ellas tiene reservados 4 bytes en los que se almacena el código de segmento (CS) y el apuntador de instrucción (IP) como se ilustra (figura 3).

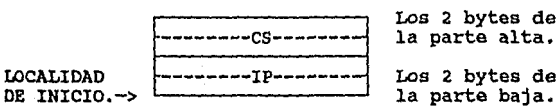
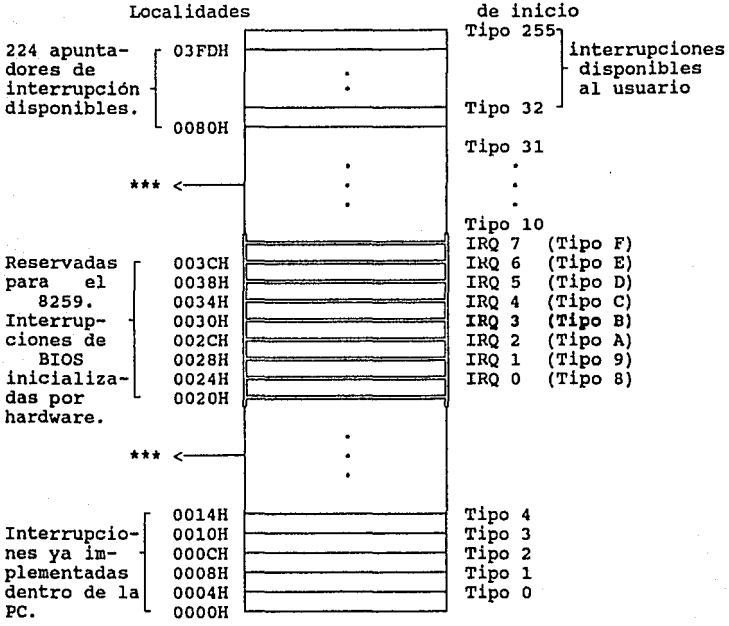


Figura 3. Distribución de localidades para CS e IP de los vectores de interrupción.

Estos dos parámetros indican al micro el inicio de la rutina de interrupción. En el vector de interrupciones se tienen reservados 32 bytes para almacenar las direcciones de CS e IP de las 8 interrupciones IRQ (figura 4).



*** Son zonas de apuntadores a interrupción del BIOS, inicializadas por software.

Figura 4. Vector de Interrupciones.

En esta forma el 8259 producirá una interrupción tipo 8 cuando su entrada IRQ0 se active. La entrada IRQ1 producirá una interrupción tipo 9, la IRQ2 una interrupción tipo A y así sucesivamente.

El 8259 se programa por medio de los puertos 0020H y 0021H de la PC. Cuando se enciende el sistema y corre el programa de inicio de la PC se inicializa el 8259 por primera vez. Después se programa por medio de los puertos, enviándole palabras de control de inicialización (ICW's) y palabras de control de operación (OCW's).

Con las palabras ICW's y OCW's se le indica al 8259 la forma en que operará, programándose para que trabaje en modo simple o en modo cascada, en caso de trabajar con arquitecturas que contengan más de un 8259. Se conectan en cascada un máximo de 8 controladores para incrementar el número de entradas de interrupción a 64 (la PC XT trabaja con un solo 8259).

También se le indica al 8259 como responder al activarse una de sus entradas puediendo ser por flanco o por nivel alto. En la palabra ICW2 se señala al 8259 que la entrada IRQ0 corresponde a la interrupción tipo 8 del vector de interrupciones asumiendo que las IRQ's consecutivas serán asignadas respectivamente a los siguientes tipos de interrupción.

La palabra ICW3 sólo se programa en sistemas que trabajen con varios 8259 en cascada.

La palabra ICW4 es usada para programar varias opciones del 8259; el bit menos significativo indica que se trabaja con un micro 8085 o con un 8088; otro bit es usado para definir, cómo una entrada de interrupción reaccionará a múltiples requerimientos. Considerando el caso de que la entrada IRQ3 se activa por flanco y el 8088 ejecuta una rutina de interrupción, un bit indica si se debe esperar un final de interrupción (EOI) al 8259 o tiene una terminación normal con una instrucción de regreso de interrupción.

A continuación se muestra el significado del valor para las palabras ICW's y las OCW's (Figura 5, a y b):

Se envía por el puerto:	ICW	byte	Contenido:
0020H	ICW1	xxx1lisf	l=1 disparo por nivel l=0 disparo por flanco i=1 intervalo de 4. i=0 intervalo de 8. s=1 trabaja un solo 8259 s=0 trabaja 8259 en cascada. f=1 si se envía ICW4. f=0 si no se envía ICW4. x no importa su valor.
0021H	ICW2	ttttxxx	ttttt=1 hasta 31, si es 1 indica que IRQ0 será la interrupción tipo 8, si es 2 IRQ0=tipo 9, etc. x no importa su valor.
0021H	ICW3	--------	No se envía cuando se usa solo un 8259.
0021H	ICW4	000sbbeu	s=1 modo de anidamiento lleno especial. s=0 modo de anidamiento lleno no especial.

Figura 5. a) Significado del valor que pueden tomar las palabras ICW del 8259.

Se envía por el puerto:	OCW	byte	Contenido:
0021H	OCW1	mmmmmmmm	m=1 No mascarable. m=0 mascarable. bit menos significativo corresponde a IRQ0, el siguiente corresponde a IRQ1, así sucesivamente con los 7 bits restantes.
0020H	OCW2	eee00111	eee=000 rota en auto EOI. eee=001 EOI no específico eee=010 no operación. eee=011 EOI específico. eee=100 rotando en auto EOI "reset". eee=101 rotando sobre EOI no específico. eee=110 enciende prioridad usando 111. eee=111 rota sobre una EOI específica.
0020H	OCW3	0mm01prr	mm=2 apaga mascara especial. mm=3 enciende mascara especial. p=1 polea. p=0 no polea. rr=2 registro del 8259 que tiene la interrupción en servicio toma del registro de entrada la próxima interrupción. rr=3 toma del registro servicio la próxima interrupción.

EOI (End Of Interruption), Fin de interrupción.

Figura 5. b) Significado del valor que pueden tomar las palabras OCW del 8259.

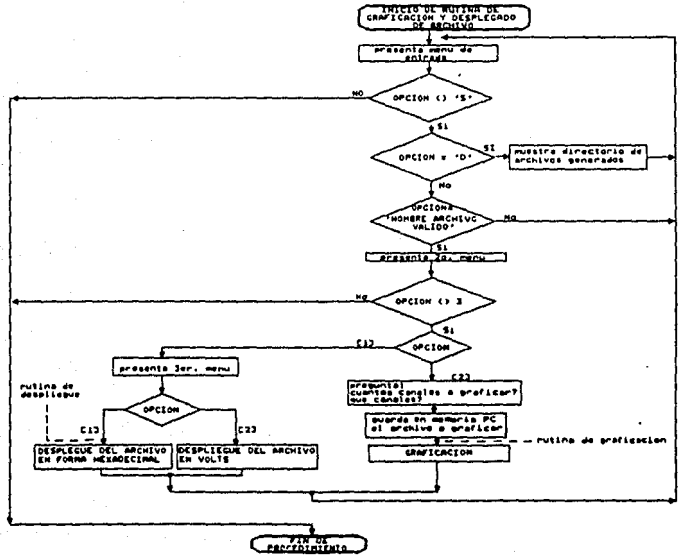
A N E X O V.

Diagramas de flujo para el desplegado y graficación de datos y la rutina de interrupción.

A N E X O V.

A continuación se presentan los diagramas de flujo de algunos procedimientos del programa principal y el diagrama de flujo de la rutina de interrupción:

1. Diagrama de flujo de la rutina de graficación y desplegado de un archivo.



NOTA:

Los diagramas de flujo de la rutina de despliegue y de la rutina de graficación se presentan más adelante.

Figura 1. Diagrama de flujo de la rutina de graficación y desplegado de un archivo.

2. Diagrama de flujo del procedimiento de desplegado de datos.

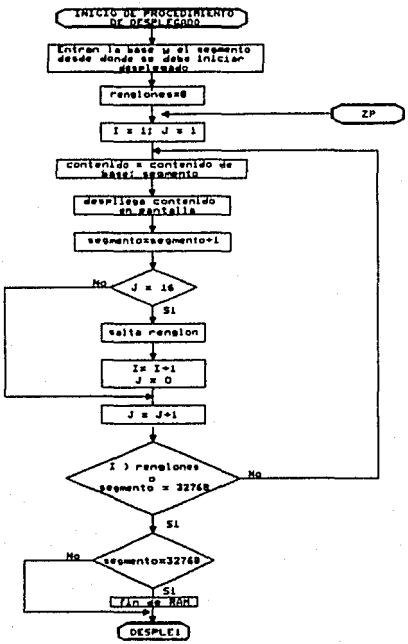


Figura 2. Diagrama de flujo del procedimiento de desplegado de datos.

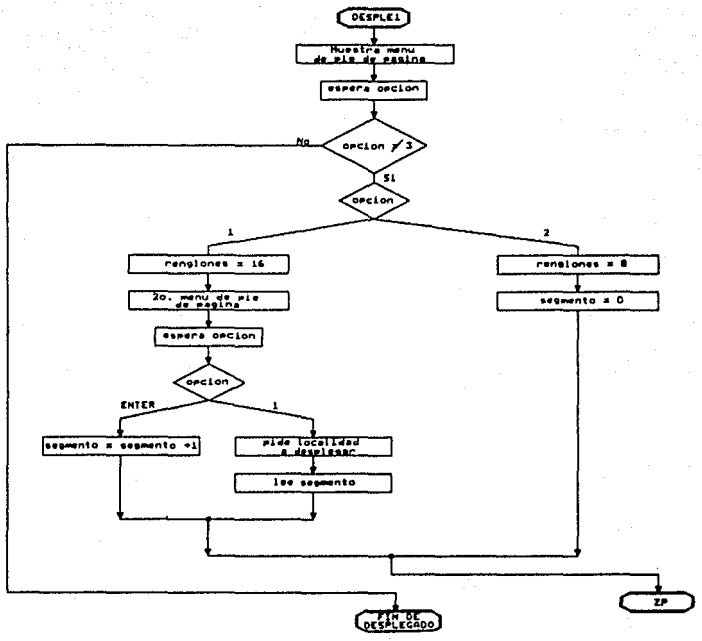
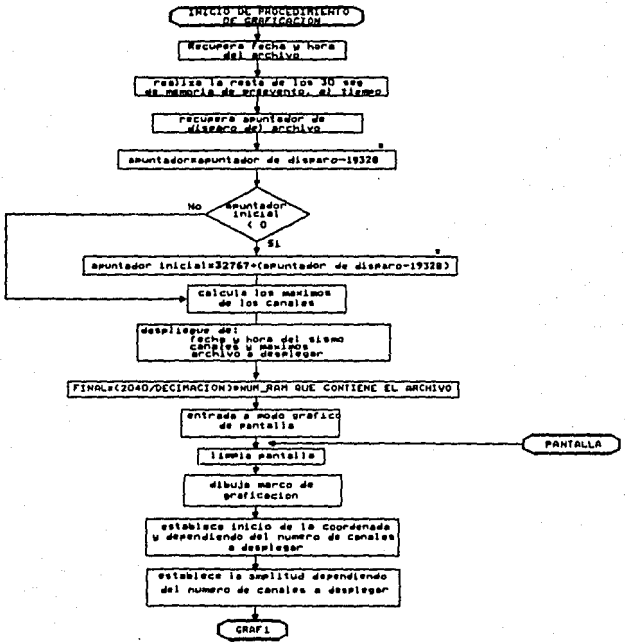


Figura 2. Diagrama de flujo del procedimiento de desplegado de datos (continuación).

3. Diagrama de flujo del procedimiento de graficación de un archivo.



NOTA:
El valor de 19,328 bytes se explica en el Capítulo VI, iniciso 6.3.2

Figura 3. Diagrama de flujo del procedimiento de graficacion de un archivo.

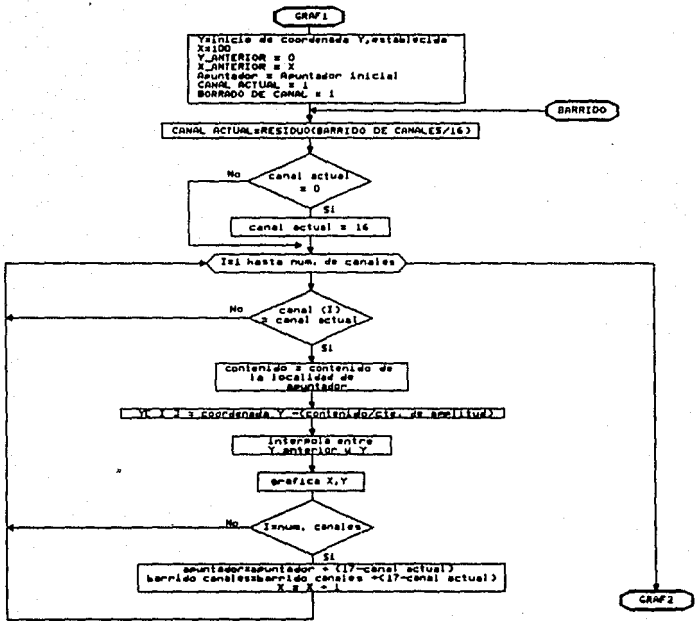


Figura 3. Diagrama de flujo del procedimiento de graficacion de un archivo (continuación).

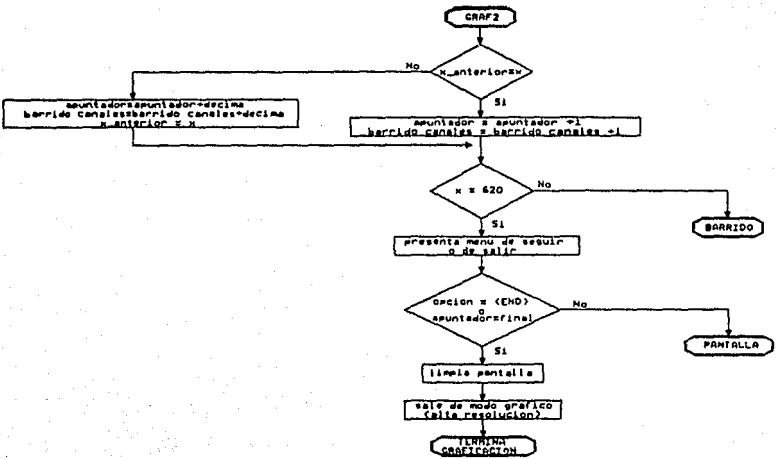
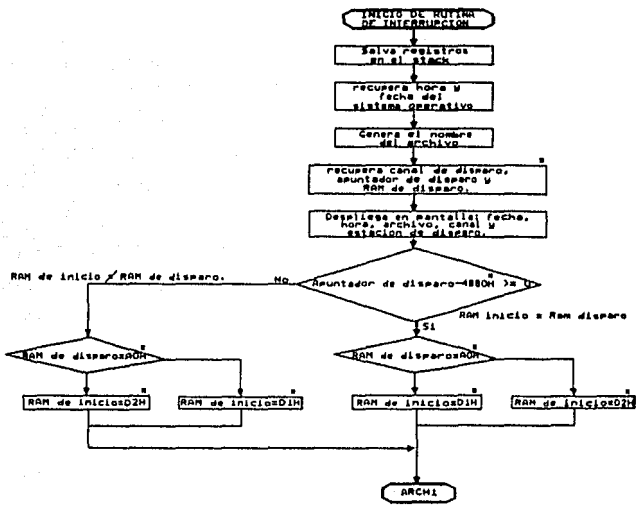


Figura 3. Diagrama de flujo del procedimiento de graficacion de un archivo (continuación).

4. Diagrama de flujo de la rutina de interrupción.



NOTA:

- Los términos RAM de disparo, RAM de inicio, Canal de disparo y Apuntador de disparo, así como los valores D1H, D2H y A0H se explican en la tabla 6.5 del capítulo VI.

- El valor 4B80H es equivalente número hexadecimal de 19,328 bytes (19,200 + 128) que se explican en el capítulo VI, iniciso 6.3.2.

Figura 3. Diagrama de flujo de la rutina de interrupción.

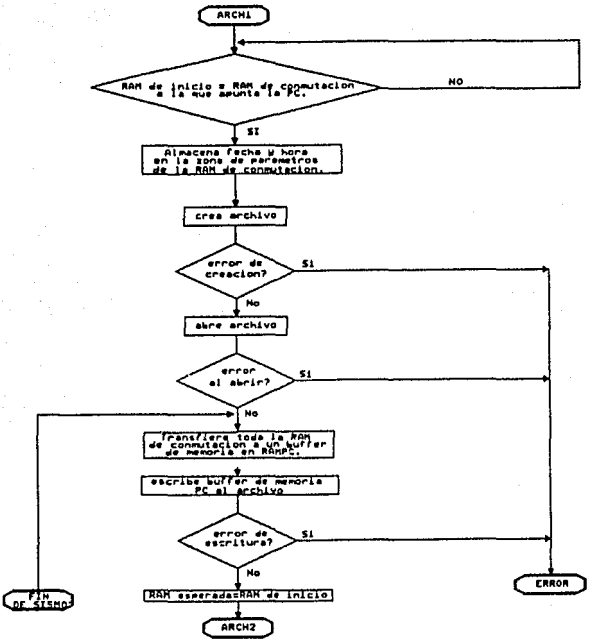


Figura 3. Diagrama de flujo de la rutina de interrupción (continuación).

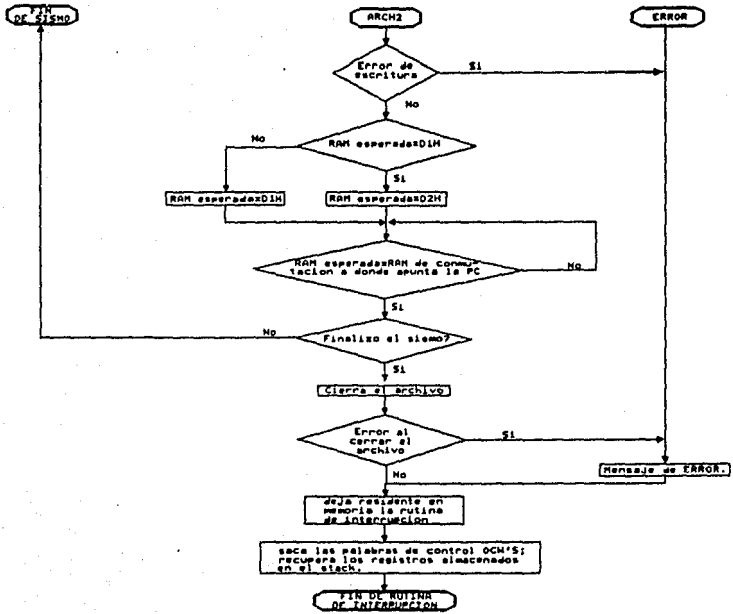


Figura 3. Diagrama de flujo de la rutina de interrupción (continuación).

A N E X O VI.

Listado del programa principal de la PC
y rutina de interrupción.

```
{                  Programa de adquisición de datos sísmicos.                }
{                                                                            }
{                  Para:  Instituto de Ingeniería, U.N.A.M.                  }
{                  Fecha de actualización:  25 de julio de 1990.             }


{$L-}PROGRAM TESIS(INPUT,OUTPUT);
USES
               CRT,DOS,PRINTER,GRAPH;          { Declaracion de unidades especiales para turbo5. }
CONST
                  TIEMPO=500;                  { Tiempo que se le da al CR (código de respuesta) }
                                               { para que iguale al CC (código de comando) que se}
                                               { ve en el programa principal.                    }
VAR
                    I,J,K: WORD;               { Variables generales)
                   II,JJ: INTEGER;
                       X: INTEGER;
                       Z: LONGINT;

                   OPCION: CHAR;               { Variables de respuesta }
                  OPCION1: CHAR;
                  OPCION2: CHAR;
                RESPUESTA: CHAR;
                  LOC_DEC: WORD;               { Variables utilizadas en procesos de }
                 CONTIENE: WORD;               { conversion HEX_DEC, DEC_HEX y desplegado }
                 LOC_HEXS: STRING;             { DESPLEGA y DESP_ARCH.           }
                 LOC_HEXO: STRING;
                  LOC_DES: STRING;
                  LOC_HEX: STRING[4];
                      AUX: STRING[4];
               LOCALIDAD: STRING[4];
                POSICION: INTEGER;
                LONGITUD: INTEGER;

              Z_PARAMETRO: BOOLEAN;            { Variables utilizadas en procesos de }
                RENGLONES: INTEGER;            { desplegado DESPLEGA, SIGUE_DESP,    }
                  LETRERO: STRING[40];         { DESP_ARCH.                          }

                     HORA: BYTE;               { Variables utilizadas en los procedimientos}
                   MINUTO: BYTE;               { de manejo de tiempo RESTA_30SEG, ESTADISTICA}
                  SEGUNDO: BYTE;
                   DECSEG: BYTE;
                      MES: BYTE;
                      DIA: BYTE;
                      AHO: WORD;
             NOMBRE_MES: STRING[10];

            ARCH_EXTERNO: STRING[15];          { Variables generales en el manejo de archivos}
             APUNTA_ARCH: WORD;                { en los procesos ESTADISTICA y GRAFICACION.  }

                  CUENTA: BYTE;                { Variables utilizadas en los procedimientos  }
                  DECIMA: BYTE;                { de graficacion: GRAFICACION, GRAFICA1,      }
                AMPLITUD: BYTE;                { RELLENA, DATOS_SISMO, ESCOGE_CANALES.       }
                  UMBRAL: INTEGER;
                CONTADOR: INTEGER;
              NO_CANALES: INTEGER;
                   COLOR: WORD;
                   TOTAL: WORD;
             INICIO_SISMO: WORD;
                       Y: ARRAY [1..4] OF INTEGER;
                   Y_ANT: ARRAY [1..4] OF INTEGER;
                   CANAL: ARRAY [1..4] OF INTEGER;
                    COOR: ARRAY [1..4] OF INTEGER;
                ESTACION: ARRAY [1..16] OF STRING[23];
              DISTANCIA: ARRAY [1..16] OF STRING[9];
                NEGATIVO: ARRAY [1..4] OF BOOLEAN;
```

```
                    MUESTRA: ARRAY [1..4] OF WORD;
                     MAXIMO: ARRAY [1..4] OF REAL;

                        CR: INTEGER;            { Variables utilizadas en el programa principal. }
                        CC: INTEGER;
                        CT: INTEGER;
                     RESP1: INTEGER;
                    NUMERO: WORD;
                     VECES: WORD;


{  ..........................  PROCEDIMIENTO  HEX_DEC ..............................}
{                                                                                    }
{  Procedimiento que convierte un numero hexadecimal a un numero decimal, el         }
{  numero en hexadecimal se asigna a la variable LOC_HEX, el resultado se da        }
{  en la variable LOC_DEC.                                                           }

PROCEDURE HEX_DEC;
VAR
    DIGITO1,DIGITO2,DIGITO3,DIGITO4: WORD;

BEGIN
    DIGITO1:=0;DIGITO2:=0;DIGITO3:=0;DIGITO4:=0;
    LONGITUD:=LENGTH(LOC_HEX);
    FOR I:=1 TO LONGITUD DO
        AUX[I]:=LOC_HEX[I];
    I:=1;
    FOR J:=5-LONGITUD TO 4 DO
    BEGIN
        LOC_HEX[J]:=AUX[I];
        I:=I+1;
    END;
    FOR I:=1 TO 4-LONGITUD DO
        LOC_HEX[I]:='0';
    FOR I:=1 TO 4 DO
    BEGIN
        IF LOC_HEX[I] IN ['0'..'9'] THEN
            J:=ORD(LOC_HEX[I])-48
        ELSE
            CASE LOC_HEX[I] OF
                'A','a': J:=10;
                'B','b': J:=11;
                'C','c': J:=12;
                'D','d': J:=13;
                'E','e': J:=14;
                'F','f': J:=15
              ELSE
            END;
        IF I=1 THEN DIGITO1:=J*4096;
        IF I=2 THEN DIGITO2:=J*256;
        IF I=3 THEN DIGITO3:=J*16;
        IF I=4 THEN DIGITO4:=J;
    END;
    LOC_DEC:=DIGITO1+DIGITO2+DIGITO3+DIGITO4;
END;

{  ..........................  PROCEDIMIENTO  DEC_HEX ..............................}
{                                                                                    }
{  Procedimiento que convierte un numero decimal a un numero hexadecimal, el         }
{  numero en decimal se asigna a la variable LOC_DEC, el resultado se da en la       }
{  variable LOC_HEX.                                                                 }

PROCEDURE DEC_HEX;
VAR
                DIGITO: ARRAY [1..4] OF INTEGER;
```

```
BEGIN
    I:=4;
    J:=LOC_DEC;
    LOC_HEX:='';
    REPEAT
        DIGITO[I]:=((J) MOD (16));
        I:=I-1;
        J:= TRUNC(J/16);
    UNTIL J=0;
    FOR J:=1 TO I DO
        DIGITO[J]:=0;
    FOR J:=1 TO 4 DO
        IF DIGITO[J]<10 THEN
            LOC_HEX[J]:=CHR(48+DIGITO[J])
        ELSE
            LOC_HEX[J]:=CHR(55+DIGITO[J]);
    AUX:='';
    IF POSICION=2 THEN
    BEGIN
        FOR I:=POSICION+1 TO 4 DO
            AUX:=CONCAT(AUX,LOC_HEX[I]);
        LOC_HEX:=AUX;
    END;
    IF POSICION=4 THEN
    BEGIN
        FOR I:=1 TO 4 DO
            AUX:=CONCAT(AUX,LOC_HEX[I]);
        LOC_HEX:=AUX;
    END
END;

{  ..........  PROCEDIMIENTO  FORMATO  ..........}
{                                                 }
{  Procedimiento que da el formato de despliegado de las localidades de las RAMS. }

PROCEDURE FORMATO;
BEGIN
     WRITE('  ');
     LOC_HEX:=LOC_HEXO;
     HEX_DEC;
     IF Z_PARAMETRO=FALSE THEN
        LOC_DEC:=LOC_DEC-128;
     POSICION:=4;
     DEC_HEX;
     WRITE(LOC_HEX,'   ');
END;

{  ..........  PROCEDIMIENTO  INICIO  ..........}
{                                               }
{  Procedimiento que coloca en la variable LOC_DES el valor de la localidad     }
{  desde donde va a comenzar a desplegar la zona de datos de las RAMS.          }

PROCEDURE INICIO;
BEGIN
    LOCALIDAD:=LOC_HEX;
    HEX_DEC;
    Z:=256;
    WHILE LOC_DEC>=Z DO
        Z:=Z+256;
    LOC_DEC:=Z-128;
    LOC_HEX:='';
    LOC_DES:='';
    POSICION:=4;
    DEC_HEX;
    LOC_DES:=LOC_HEX;
    LOC_HEX:=LOCALIDAD;
```

```
    HEX_DEC;
END;

PROCEDURE NOMBRE_ARCH;FORWARD;

PROCEDURE ARCH_BUFFER;FORWARD;

{  ..........  PROCEDIMIENTO  SEGUIR_DES  ..........}
{                                                    }
{  Procedimiento que permite seguir desplegando los datos contenidos en las RAMS.}

PROCEDURE SEGUIR_DES;
BEGIN
    REPEAT
        GOTOXY(1,24);
        WRITE('PRESIONE   (1) SEGUIR EN ZONA DATOS    (2) ZONA PARAMETROS    (3) SALIDA : ');
        OPCION:=READKEY;
    UNTIL OPCION IN ['1','2','3'];
    IF OPCION='1' THEN
    BEGIN
        GOTOXY(1,24);
        WRITE(' PRESIONE   (1) DESPLEGAR OTRA LOCALIDAD   O   <ENTER> LA SIGUIENTE LOCALIDAD:');
        RESPUESTA:=READKEY;
        IF RESPUESTA='1' THEN
        BEGIN
            REPEAT
                GOTOXY(1,24);
                WRITE(' ':78);
                GOTOXY(1,24);
                WRITE('DE LA LOCALIDAD  (HEXADECIMAL < 7F7F)  A DESPLEGAR: ');
                READLN(LOC_HEX);
                INICIO;
                LOC_HEXD:=LOC_DES;
                GOTOXY(1,24);
                IF LOC_DEC>=32639 THEN
                    WRITE(' ':78);
            UNTIL LOC_DEC<32639;
        END;
    END;
END;

{  ..........  PROCEDIMIENTO  DA_OPCION  ..........}
{                                                  }
{  Procedimiento que pide la opcion:  seguir desplegando los datos contenidos en }
{  las RAMS, desplegar la zona de parametros de las RAMS o salir a menu principal}

PROCEDURE DA_OPCION;
BEGIN
    IF Z_PARAMETRO THEN
    BEGIN
        REPEAT
            GOTOXY(1,24);
            WRITE(' ':5,'PRESIONE    (1)  PARA DESPLEGAR ZONA DE DATOS    (2)  SALIDA: ');
            OPCION:=READKEY;
        UNTIL OPCION IN ['2','1'];
        IF OPCION='2' THEN
            OPCION:='3'
    END
    ELSE
        SEGUIR_DES;
END;

{  ..........  PROCEDIMIENTO  ESCOGE_OPCION  ..........}
{                                                      }
{  Procedimiento que da los parametros a utilizar en el procedimiento de DESPLEGA}
{  dependiendo de la opcion que se eligio.                                       }
```

```
PROCEDURE ESCOGE_OPCION;
BEGIN
    CASE OPCION OF
        '1': BEGIN
                IF Z_PARAMETRO THEN
                    LOC_HEXO:=LOC_DES;
                Z_PARAMETRO:=FALSE;
                RENGLONES:=16;
             END;
        '2': BEGIN
                LOC_HEXO:='0000';
                Z_PARAMETRO:=TRUE;
                RENGLONES:=8;
             END
        ELSE
    END;                                        { Fin del case de opcion.
}
END;

{  ......................  PROCEDIMIENTO  PASA_PARAM  ......................}
{                                                                           }
{  Procedimiento que obtiene el contenido de las RAMS de la tarjeta adquisidora  }

{$F+}
PROCEDURE PASA_PARAM;
BEGIN
    INLINE


                        { code_here segment                                 }
                        {       assume cs:code_here, ds:code_here           }
                        { pasa_param proc near                              }
    ($8C/$D8/           {       mov ax,ds       }
     $50/               {       push ax         }
     $B8/$00/$22/       {       mov ax,2200h    }
     $8E/$D8/           {       mov ds,ax       }
     $8E/$00/$00/       {       mov si,0000h    }
     $8A/$04/           {       mov al,[si]     }
     $8E/$01/$00/       {       mov si,0001h    }
     $8A/$24/           {       mov ah,[si]     }
     $8E/$02/$00/       {       mov si,0002h    }
     $8A/$1C/           {       mov bl,[si]     }
     $8E/$03/$00/       {       mov si,0003h    }
     $8A/$3C/           {       mov bh,[si]     }
     $8E/$D8/           {       mov ds,ax       }
     $8B/$F3/           {       mov si,bx       }
     $8A/$0C/           {       mov cl,[si]     }
     $B8/$00/$22/       {       mov ax,2200h    }
     $8E/$D8/           {       mov ds,ax       }
     $8E/$04/$00/       {       mov si,0004h    }
     $88/$0C/           {       mov [si],cl     }
     $58/               {       pop ax          }
     $8E/$D8 );         {       mov ds,ax       }
                        {       ret             }
                        {pasa_param endp        }
                        {code_here ends         }
                        {       end             }
END;
{$F-}

{  ......................  PROCEDIMIENTO  DESPLEGA  ......................}
{                                                                           }
{  Procedimiento que despliega las localidades de memoria de las RAMS y es  }
{  despues de que se hace una escritura en ella, o en la lectura y desplegado  }
{  de las mismas.                                                           }
```

```
PROCEDURE DESPLEGA;
BEGIN
    LOC_HEX$:='';
    LOC_HEX$:='0000';
    LOC_HEXO:='';
    LOC_HEXO:='0000';
    RENGLONES:=8;
    Z_PARAMETRO:=TRUE;
    REPEAT                                  ( lectura de cualquiera de las RAM"S.              )
        CLRSCR;
        WRITELN;
        WRITELN(' ':15,LETRERO);
        IF Z_PARAMETRO THEN
        BEGIN
            WRITELN;
            WRITELN(' ':15,'          ZONA DE PARAMETROS');
            WRITELN;
        END;
        WRITELN(' ':160);
        FORMATO;
        JJ:=1;
        II:=1;
        REPEAT
           MEMW[$2200:$0000]:=53248;                    ( Pasando el 0000 como data segment.    )
            LOC_HEX:=LOC_HEXO;
            HEX_DEC;
            MEMW[$2200:$0002]:=LOC_DEC;
            PASA_PARAM;
            CONTIENE:=MEM[$2200:$0004];
            POSICION:=2;
            LOC_DEC:=CONTIENE;
            DEC_HEX;
            WRITE(LOC_HEX,' ');
            LOC_HEX:=LOC_HEXO;
            HEX_DEC;
            LOC_DEC:=LOC_DEC+1;
            POSICION:=4;
            DEC_HEX;
            LOC_HEXO:=LOC_HEX;
            IF JJ=16 THEN
            BEGIN
                WRITELN;
                IF (II<RENGLONES) AND (LOC_HEXO<>'8000') THEN FORMATO;
                II:=II+1;
                JJ:=0;
            END;
            JJ:=JJ+1;
        UNTIL (II>RENGLONES) OR (LOC_HEXO='8000') ;
        IF LOC_HEXO='8000' THEN
        BEGIN
            OPCION:='3';
            WRITELN(' ':400);
            WRITELN(' ':16,'TERMINO DE MOSTRAR TODA LA ZONA DE DATOS.');
            SEGUIR_DES;
        END
        ELSE
            DA_OPCION;
        ESCOGE_OPCION;
    UNTIL OPCION='3';
END;

{  ..................... PROCEDIMIENTO LOCALIDAD_A_DESP .....................}
{                                                                           }
{  Procedimiento que pide la localidad desde donde se desea comenzar a desplegar }
{  la zona de datos de las RAMS.                                            }
```

```
PROCEDURE LOCALIDAD_A_DESP;
BEGIN
    CLRSCR;
    WRITELN;
    WRITELN('                                                                    ');
    WRITELN('                                                                    ');
    WRITELN(' ':1500);
    WRITELN('                                                                    ');
    WRITELN('                                                                    ');
    REPEAT
        GOTOXY(8,12);
        WRITE('DESDE QUE LOCALIDAD [HEXADECIMAL  < 7F7F ] DESEA DESPLEGAR:  ');
        READLN(LOC_HEX);
        INICIO;
        GOTOXY(8,12);
        IF LOC_DEC>=32639 THEN
            WRITE(' ':78);
  UNTIL LOC_DEC<32639;
END;

{   ..........................  PROCEDIMIENTO  CUADRO  ..............................}
{                                                                                    }
{   Procedimiento que coloca un cuadro en pantalla.                                  }

PROCEDURE CUADRO;
BEGIN
    WRITELN(' ':320);
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    WRITELN(' ':10,'                                                              ');
    GOTOXY(20,11);
END;

{   ......................  PROCEDIMIENTO TABLA_ESTACIONES ........................}
{                                                                                    }
{   Procedimiento que contiene el nombre y distancia de todas las estaciones de      }
{   se resiven las señales sismicas.                                                 }

PROCEDURE TABLA_ESTACIONES;
BEGIN
    ESTACION[1]:=' SANTA RITA, EDO. MEX. ';
    ESTACION[2]:=' CD. SERDAN, PUE.      ';
    ESTACION[3]:=' TOMANTZINTLA, PUE.    ';
    ESTACION[4]:=' LOS ORGANOS, EDO. MEX.';
    ESTACION[5]:=' ALTZOMONI, PUE.       ';
    ESTACION[6]:=' MEZONTEPETL, D.F.     ';
    ESTACION[7]:=' TUXPAN, GRO.          ';
    ESTACION[8]:=' JOCOTITLAN, PUE.      ';
    ESTACION[9]:=' MESA VIBRADORA, C.U.  ';
    ESTACION[10]:=' NO ASIGNADA.          ';
    ESTACION[11]:=' NO ASIGNADA.          ';
    ESTACION[12]:=' NO ASIGNADA.          ';
    ESTACION[13]:=' NO ASIGNADA.          ';
    ESTACION[14]:=' NO ASIGNADA.          ';
    ESTACION[15]:=' NO ASIGNADA.          ';
    ESTACION[16]:=' NO ASIGNADA.          ';
    DISTANCIA[1]:=' 47.2 Km.  ';
```

```
    DISTANCIA[2]:=' 188.0 Km. ';
    DISTANCIA[3]:=' 86.0 Km.  ';
    DISTANCIA[4]:=' 48.0 Km.  ';
    DISTANCIA[5]:=' 56.4 Km.  ';
    DISTANCIA[6]:=' 18.4 Km.  ';
    DISTANCIA[7]:=' 101.6 Km. ';
    DISTANCIA[8]:=' 172.8 Km. ';
    DISTANCIA[9]:=' 7.2 Km.   ';
    DISTANCIA[10]:='NO ASIGNADA.';
    DISTANCIA[11]:='NO ASIGNADA.';
    DISTANCIA[12]:='NO ASIGNADA.';
    DISTANCIA[13]:='NO ASIGNADA.';
    DISTANCIA[14]:='NO ASIGNADA.';
    DISTANCIA[15]:='NO ASIGNADA.';
    DISTANCIA[16]:='NO ASIGNADA.';
END;

{$I INTERRUP}
{$I ARCHIVO}
{$I ESTADISTICA}

{  .....................    PROGRAMA PRINCIPAL  ..........................}
{                                                                         }
{  El programa realiza la comunicación con la tarjeta adquisidora, y comunica a  }
{  esta con el usuario, para que este realize tareas de adquisicion de datos y   }
{  autodiagnostico, asi como el procesamiento de datos sismicos obtenidos.       }

BEGIN
    TEXTBACKGROUND(0);
    CLRSCR;
    SOUND(220);
    DELAY(200);
    NOSOUND;
    MEM[$E000:$0000]:=0;                    { Limpia la primera localidad del PPC.          }
    MEM[$0000:$0000]:=1;                    { Coloca un 1 a la primer localidad de la RAM a la }
                                            { que este apuntando en ese  momento para indicar  }
                                            { que ya termino el boot y puede comenzar el MPA.  }
    INTERR;
    REPEAT
        REPEAT
            REPEAT
                CLRSCR;
                WRITELN;
                WRITELN('                                                                ');
                WRITELN('                                                                ');
                WRITELN('                                                                ');
                WRITELN('                                                                ');
                WRITELN('                          M E N U:                              ');
                WRITELN('                                                                ');
                WRITELN('                1.-  REVISION DE RAM'S.                         ');
                WRITELN('                                                                ');
                WRITELN('                2.-  ESCRITURA EN RAM'S.                        ');
                WRITELN('                                                                ');
                WRITELN('                3.-  LECTURA Y DESPLEGADO DE EL CONTENIDO DE LAS RAM'S.');
                WRITELN('                                                                ');
                WRITELN('                4.-  BORRADO DE LAS RAM'S.                      ');
                WRITELN('                                                                ');
                WRITELN('                5.-  ADQUISICION DE DATOS.                      ');
                WRITELN('                                                                ');
                WRITELN('                6.-  GRAFICACION  O  DESPLEGADO DE UN ARCHIVO.  ');
                WRITELN('                                                                ');
                WRITELN('                7.-  DESPLEGADO DE INFORMACION ESTADISTICA.     ');
                WRITELN('                                                                ');
                WRITELN('                8.-  SALIR A MSDOS.                             ');
                WRITELN('                                                                ');
                WRITELN('                                                                ');
                WRITELN;
                WRITE  ('                              OPCION: ');
```

```
    OPCION1:=READKEY;
UNTIL OPCION1 IN ['1','2','3','4','5','6','7','8'];
CASE OPCION1 OF
  '1': BEGIN                              ( Revision de RAM'S.                    )
         CC:=$15;                  ( Codigo que debe tener el CC para hacer revision de RAM'S. )
         OPCION2:='1';
       END;
  '2': BEGIN                              ( Escritura en RAM'S.                   )
         REPEAT
           CLRSCR;
           WRITELN(' ':320);
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':12,'                      M E N U:                      ');
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':12,'                1.- ESCRITURA EN RAM1.              ');
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':12,'                2.- ESCRITURA EN RAM2.              ');
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':12,'                3.- SALIDA A MENU PRINCIPAL.        ');
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':12,'                                                    ');
           WRITELN(' ':160);
           WRITE(' ':20,'OPCION: ');
           OPCION2:=READKEY;
         UNTIL OPCION2 IN ['1','2','3'];
         IF OPCION2<>'3' THEN
         BEGIN
           CLRSCR;
           WRITELN;
           WRITELN('                                                           ');
           WRITELN('                                                           ');
           WRITELN(' ':1500);
           WRITELN('                                                           ');
           WRITELN('                                                           ');
           GOTOXY(4,10);
           WRITE('EL NUMERO [ HEXADECIMAL <= FFFF ] QUE DESEA ESCRIBIR EN LA RAM',OPCION2,' ES: ');
           READ(LOC_HEX);
           REPEAT
             GOTOXY(4,12);
             WRITE('CUANTAS VECES [ <= 16320 ] DESEA ESCRIBIR EL NUMERO ',LOC_HEX,': ');
             READLN(VECES);
             GOTOXY(4,12);
             IF VECES>16320 THEN
               WRITE(' ':78);
           UNTIL VECES<=16320;
           HEX_DEC;
           NUMERO:=LOC_DEC;
           MEMW[$0000:$0020]:=NUMERO; ( Pasa el numero que se desea escribir a la locali-)
                                      ( dad 0020 de la RAM.                   )
           REPEAT
             GOTOXY(4,14);
             WRITE('EN QUE LOCALIDAD [ < 7F7F ] DESEA SE COMIENCE A ESCRIBIR: ');
             READLN(LOC_HEX);
             INICIO;
             GOTOXY(4,14);
             IF LOC_DEC>=32639 THEN
               WRITE(' ':78);
           UNTIL LOC_DEC<32639;
           Z:=LOC_DEC*128+VECES*2;
           IF Z<=32768 THEN      ( Verifica que veces ha escribir no pase Capacid. de RAM.)
              MEMW[$0000:$0022]:=VECES (Pasa veces a escribir a localidad 0022 de la RAM)
           ELSE
             MEMW[$0000:$0022]:=TRUNC((32640-LOC_DEC)/2);
           LOC_DEC:=LOC_DEC*128;
           MEMW[$0000:$0024]:=LOC_DEC;
```

```
            CASE OPCION2 OF
               '1': CC:=$21;     ( Codigo que debe tener el CC para escritura en RAM1. )
               '2': CC:=$31;     ( Codigo que debe tener el CC para escritura en RAM2. )
                ELSE
            END;
         END;
      END;
'3': BEGIN                        ( Desplegado y lectura de RAM'S.                    )
         REPEAT
            CLRSCR;
            WRITELN(' ':320);
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':12,'                          M E N U:                        ');
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':12,'   1.- LECTURA Y DESPLEGADO DEL CONTENIDO EN RAM1.        ');
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':12,'   2.- LECTURA Y DESPLEGADO DEL CONTENIDO EN RAM2.        ');
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':12,'   3.- SALIDA A MENU PRINCIPAL.                          ');
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':12,'                                                          ');
            WRITELN(' ':160);
            WRITE(' ':20,'OPCION: ');
            OPCION2:=READKEY;
         UNTIL OPCION2 IN ['1','2','3'];
         IF OPCION2<>'3' THEN
         BEGIN
            LOCALIDAD_A_DESP;
            CASE OPCION2 OF
               '1': CC:=$23;     ( Codigo que debe tener el CC para desplegado de RAM1. )
               '2': CC:=$33;     ( Codigo que debe tener el CC para desplegado de RAM2. )
                ELSE
            END;
         END;
      END;
'4': BEGIN                           ( Limpia las RAM'S, su contenido es ceros.       )
         REPEAT
            CLRSCR;
            WRITELN(' ':320);
            WRITELN(' ':10,'                                                          ');
            WRITELN(' ':10,'                                                          ');
            WRITELN(' ':10,'                          M E N U:                        ');
            WRITELN(' ':10,'                                                          ');
            WRITELN(' ':10,'   1.- BORRA SOLO LA ZONA DE PARAMETROS DE LAS RAM'S.     ');
            WRITELN(' ':10,'                                                          ');
            WRITELN(' ':10,'   2.- BORRA COMPLETAMENTE LAS RAM'S.                     ');
            WRITELN(' ':10,'                                                          ');
            WRITELN(' ':10,'   3.- SALIDA A MENU PRINCIPAL.                          ');
            WRITELN(' ':10,'                                                          ');
            WRITELN(' ':10,'                                                          ');
            WRITELN(' ':160);
            WRITE(' ':20,'OPCION: ');
            OPCION2:=READKEY;
         UNTIL OPCION2 IN ['1','2','3'];
         CASE OPCION2 OF
            '1': CC:=$3;(Codigo que debe tener CC para limpiar zona de parametros de RAMS)
            '2': CC:=$2;      ( Cogigo que debe tener CC para limpiar completamente RAMS.)
             ELSE
         END;
      END;
'5': BEGIN                           ( Adquisicion de datos.                          )
         REPEAT
            CLRSCR;
            WRITELN(' ':320);
            WRITELN(' ':12,'                                                          ');
```

```
                WRITELN(' ':12,'
                WRITELN(' ':12,'                          M E N U:                                 ');
                WRITELN(' ':12,'                                                                   ');
                WRITELN(' ':12,'      1.- INICIAR ADQUISICION DE DATOS.                            ');
                WRITELN(' ':12,'                                                                   ');
                WRITELN(' ':12,'      2.- TERMINAR ADQUISICION DE DATOS.                           ');
                WRITELN(' ':12,'                                                                   ');
                WRITELN(' ':12,'      3.- SALIDA A MENU PRINCIPAL.                                 ');
                WRITELN(' ':12,'                                                                   ');
                WRITELN(' ':12,'                                                                   ');
                WRITELN(' ':160);
                WRITE(' ':20,'OPCION: ');
                OPCION2:=READKEY;
            UNTIL OPCION2 IN ['1','2','3'];
            IF OPCION2='1' THEN
            BEGIN
                CLRSCR;
                CUADRO;
                WRITE('PROPORCIONE EL UMBRAL DE DISPARO (1-3):  ');
                READLN(UMBRAL);
                MEM[$0000:$0030]:=UMBRAL;
            END;
            IF OPCION2<>'3' THEN
                CASE OPCION2 OF
                   '1': CC:=$40;     { Codigo que debe tener el CC para adquirir datos.     }
                   '2': CC:=$41;     { Codigo que debe tener el CC para ya no adquirir datos}
                    ELSE
                END;
        END;
    '6': BEGIN
            MANEJO_ARCHIVO;
        END;
    '7': BEGIN
            ESTADISTICA;
        END;
    '8': BEGIN
            CLRSCR;
           CC:=$2;                          { Para salirse a MSDOS limpia completamente RAM'S. }
            OPCION2:='1';
        END;
     ELSE
  END;                                      { Fin del case general de opciones del menu.       }
UNTIL OPCION2<>'3';
CONTADOR:=0;
REPEAT
  MEM[$0000:$0000]:=1;                      { Asegura que termino boot y puede iniciar.        }
  MEM[$E000:$0000]:=CC;                     { La PC pone en PPC el CC (opcion a realizar),  el }
                                            { micro (MPA) la localiza en el registro DI.       }
  CR:=MEM[$0000:$0001];                     { Lee la PC de la RAM a la que este apuntando  en  }
                                            { ese momento el  CR  (codigo de respuesta),  este }
                                            { debe ser igual al CC.                            }
   CONTADOR:=CONTADOR+1;
UNTIL (CR=CC) OR (CONTADOR=TIEMPO);         { Este en un "loop" hasta que CR=CC.               }
IF OPCION1<>'8' THEN                  { Si la opcion no es la de salida entra y sigue proceso. }
 BEGIN
  IF CONTADOR=TIEMPO THEN             { Se salio del "loop" porque el tiempo se agoto y CR<>CC. }
   BEGIN
      CLRSCR;
      SOUND(220);
      CUADRO;
      WRITE('ERROR, NO RECIBE EL CODIGO DE RESPUESTA');
      DELAY(200);
      NOSOUND;
      DELAY(3000);
   END
  ELSE                                      { Si no entonces CC=CR.                            }
```

```
BEGIN
   MEM[$E000:$0000]:=0;                    { Pone en el puerto PPC un  0 para que  comienze a }
                                           { a ejecutar la opcion pedida.                     }
    CLRSCR;
    WRITELN(' ':560);
   CT:=MEM[$D000:$0002];                   { Lee  la  PC el  CT (codigo de terminacion) de la }
                                           { localidad 02 de la RAM a la que este apuntando en}
                                           { ese momento y este CT puede ser: A0, A1, A2, etc.}
   Z:=CT-160;                            { La variable Z tiene el numero de respuestas que trae. }
    IF (Z>16) THEN
    BEGIN
        CLRSCR;
        SOUND(220);
        CUADRO;
        WRITE('ERROR, CODIGO DE TERMINACION INCORRECTO!');
        DELAY(200);
        NOSOUND;
    END
    ELSE
    BEGIN
        CASE CR OF
         $2,$3: BEGIN                  { Respuesta a la opcion de borra RAM'S.              }
                    CLRSCR;
                    WRITELN(' ':560);
                    WRITELN(' ':10,'                                                    ');
                    WRITELN(' ':10,'                                                    ');
                    WRITELN(' ':10,'        LAS RAM'S HAN SIDO LIMPIADAS CON 0'S        ');
                    WRITELN(' ':10,'                                                    ');
                    WRITELN(' ':10,'                                                    ');
                    DELAY(2000);
                END;
           $15: BEGIN                  { Respuesta a la opcion de revisa RAM'S.            }
                    CLRSCR;
                    WRITELN(' ':560);
                   RESP1:=MEM[$0000:$0003];(Localidad donde encontramos 1F o 1B segun Edo.RAM1)
                    WRITELN(' ':12,'                                                    ');
                    WRITELN(' ':12,'                                                    ');
                    IF RESP1<>$1F THEN
                        WRITELN(' ':12,'          LA RAM1 ESTA CORRECTA.              ')
                    ELSE
                        WRITELN(' ':12,'          LA RAM1 ESTA DANADA.                ');
                    WRITELN(' ':12,'                                                    ');
                   RESP1:=MEM[$0000:$0004];(Localidad donde encontramos 2F o 2B segun Edo.RAM2)
                    IF RESP1<>$2F THEN
                        WRITELN(' ':12,'          LA RAM2 ESTA CORRECTA.              ')
                    ELSE
                        WRITELN(' ':12,'          LA RAM2 ESTA DANADA.                ');
                    WRITELN(' ':12,'                                                    ');
                    WRITELN(' ':12,'                                                    ');
                    SOUND(620);
                    DELAY(200);
                    NOSOUND;
                    WRITELN(' ':320);
                    WRITELN(' ':12,'          ESPERE UN MOMENTO...');
                    DELAY(2000);
                END;
           $21: BEGIN                    { Despliega la escritura realizada en RAM1.       }
                    CLRSCR;
                    WRITELN(' ':400);
                    LETRERO:=' LA ESCRITURA EN RAM1 HA SIDO REALIZADA.';
                    DELAY(1000);
                    DESPLEGA;
                END;
           $31: BEGIN                    { Despliega la escritura realizada en RAM2.       }
                    CLRSCR;
                    WRITELN(' ':400);
```

```
                    LETRERO:=' LA ESCRITURA EN RAM2 HA SIDO REALIZADA.';
                    DELAY(1000);
                    DESPLEGA;
                END;
            $23: BEGIN                          { Despliega lo que lee de la RAM1.          }
                    CLRSCR;
                    WRITELN;
                    LETRERO:='     LECTURA Y DESPLEGADO DE LA RAM1:   ';
                    DESPLEGA;
                END;
            $33: BEGIN                          { Despliega lo que lee de la RAM2.          }
                    CLRSCR;
                    WRITELN;
                    LETRERO:='     LECTURA Y DESPLEGADO DE LA RAM2:   ';
                    DESPLEGA;
                END;
             $40: BEGIN
                    CLRSCR;
                    CUADRO;
                    WRITE('    INICIO DE ADQUISICION DE DATOS.');
                    DELAY(2000);
                END;
             $41: BEGIN
                    CLRSCR;
                    CUADRO;
                    WRITE('     TERMINO ADQUISICION DE DATOS.');
                    DELAY(2000);
                END;
              ELSE
            END;
        END;                                    { Fin del else del error en codigo de terminacion. }
    END;                                        { Fin del if then else CONTADOR=TIEMPO.            }
    CLRSCR;
    RESPUESTA:='S';
  END
  ELSE                                          { Else si la opcion1='6' quiere salir a MSDOS.     }
    RESPUESTA:='N';                             { Fin del if de opcion1='6'.                       }
  CLRSCR;
  IF (CC=$40) AND (CR=$40) THEN
   BEGIN
      RESPUESTA:='N';
      SWAPVECTORS;
      EXEC('\COMMAND.COM','/C PROMPT ADQUIRIENDO DATOS$G');
      SWAPVECTORS;
  END;
UNTIL (RESPUESTA='N');
INTERR;
END.
```

(A CONTINUACION: $I ARCHIVO)

```
{  ..........  PROCEDIMIENTO  MARCO  ..........}
{                                               }
{  Procedimiento que coloca un marco en pantalla.}

PROCEDURE MARCO;
BEGIN
    CLRSCR;
    WRITELN('╔═══════════════════════════════════════════════════════════════════════════╗');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('║                                                                           ║');
    WRITELN('╚═══════════════════════════════════════════════════════════════════════════╝');
END;

{  ..........  PROCEDIMIENTO PREGUNTA_CANAL  ..........}
{                                                       }
{  Procedimiento que pregunta cuantos y que canales se desea desplegar o graficar)

PROCEDURE PREGUNTA_CANAL;
VAR
    CANAL_SISMO: BYTE;
BEGIN
    CLRSCR;
    WRITELN;
    WRITELN('╔═══════════════════════════════════════════════════════════════════════════╗');
    WRITELN('╚═══════════════════════════════════════════════════════════════════════════╝');
    WRITELN(' ':1500);
    WRITELN('╔═══════════════════════════════════════════════════════════════════════════╗');
    WRITELN('╚═══════════════════════════════════════════════════════════════════════════╝');
    REPEAT
        CANAL_SISMO:=MEM[$8000:$003C];
        CANAL_SISMO:=TRUNC(CANAL_SISMO/2)+1;
        GOTOXY(14,6);
        WRITE('EL CANAL ',CANAL_SISMO,' DETECTO PRIMERO EL SISMO.');
        GOTOXY(14,8);
        WRITE(' ':75);
        GOTOXY(14,8);
        WRITE('CUANTOS CANALES DESEA ',LETRERO,' (1,2,3,4)?  ');
        READ(NO_CANALES);
    UNTIL NO_CANALES IN [1,2,3,4];
    I:=1;
    J:=0;
    REPEAT
        REPEAT
            GOTOXY(14,J+10);
            WRITE(' ':75);
            GOTOXY(14,J+10);
```

```
            IF NO_CANALES=1 THEN
                WRITE('EL CANAL A ',LETRERO,' DE [1 - 16] ES: ')
            ELSE
                WRITE('EL CANAL A ',LETRERO,' NO. ',I,' ES [1 - 16]: ');
            READ(CANAL[I]);
        UNTIL  (CANAL[I] IN [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]);
        J:=J+2;
        I:=I+1;
    UNTIL I>NO_CANALES;
    LETRERO:='T';
END;

{  ..........................  PROCEDIMIENTO  HEX_DEC  ..............................}
{                                                                                    }
{  Procedimiento que ordena los canales a desplegar o graficar en forma              }
{  decreciente.                                                                      }

PROCEDURE ORDENA_CANAL;
VAR
        N: BYTE;
    TEMPORAL: BYTE;

BEGIN
    N:=1;
    REPEAT
        FOR I:=1 TO (NO_CANALES-1) DO
            IF CANAL[I]>CANAL[I+1] THEN
            BEGIN
                TEMPORAL:=CANAL[I];
                CANAL[I]:=CANAL[I+1];
                CANAL[I+1]:=TEMPORAL;
            END;
        N:=N+1;
    UNTIL N>(NO_CANALES-1);
END;

{  ..........................  PROCEDIMIENTO  NOMBRE_ARCH  ..........................}
{                                                                                    }
{  Procedimiento que envia a un buffer de memoria el nombre del archivo con el       }
{  se va he trabajar.                                                                }

PROCEDURE NOMBRE_ARCH;
BEGIN
    MEM[$8000:$0000]:=$64;
    MEM[$8000:$0001]:=$3A;
    MEM[$8000:$0002]:=$5C;
    MEM[$8000:$0003]:=ORD(ARCH_EXTERNO[1]);
    MEM[$8000:$0004]:=ORD(ARCH_EXTERNO[2]);
    MEM[$8000:$0005]:=ORD(ARCH_EXTERNO[3]);
    MEM[$8000:$0006]:=ORD(ARCH_EXTERNO[4]);
    MEM[$8000:$0007]:=ORD(ARCH_EXTERNO[5]);
    MEM[$8000:$0008]:=ORD(ARCH_EXTERNO[6]);
    MEM[$8000:$0009]:=ORD(ARCH_EXTERNO[7]);
    MEM[$8000:$000A]:=ORD(ARCH_EXTERNO[8]);
    MEM[$8000:$000B]:=$2E;
    MEM[$8000:$000C]:=ORD(ARCH_EXTERNO[10]);
    MEM[$8000:$000D]:=ORD(ARCH_EXTERNO[11]);
    MEM[$8000:$000E]:=ORD(ARCH_EXTERNO[12]);
    MEM[$8000:$000F]:=$00;
    MEMW[$8000:$0010]:=$0000;
    MEMW[$8000:$0012]:=APUNTA_ARCH;
END;

{  ..........................  PROCEDIMIENTO  ARCH_BUFFER  ..........................}
{                                                                                    }
{  Procedimiento que almacena en un buffer de memoria el archivo con el que se va}
```

```
{   a trabajar.                                                    }

($F+)
PROCEDURE ARCH_BUFFER;
BEGIN
    INLINE
    ($1E/                {       push ds                 }
     $8C/$00/            {       mov ax,ss               }
     $89/$E3/            {       mov bx,sp               }
     $89/$E9/            {       mov cs,bp               }
     $50/                {       push ax                 }
     $53/                {       push bx                 }
     $51/                {       push cx                 }
     $B4/$3D/            {       mov ah,3d               }
     $B0/$02/            {       mov al,02               }
     $BA/$00/$80/        {       mov dx,8000             }
     $8E/$DA/            {       mov ds,dx               }
     $BA/$00/$00/        {       mov dx,0000             }
     $CD/$21/            {       int 21                  }
     $73/$02/            {       jnb sigue               }
     $EB/$3D/            {       jmp error               }
     $A3/$10/$00/        {sigue: mov [0010],ax           }
     $B8/$00/$42/        {       mov ax,4200             }
     $8B/$1E/$10/$00/    {       mov bx,[0010]           }
     $B9/$00/$00/        {       mov cx,0000             }
     $BB/$16/$12/$00/    {       mov dx,[0012]           }
     $CD/$21/            {       int 21                  }
     $72/$28/            {       jb error                }
     $B4/$3F/            {       mov ah,3f               }
     $8B/$1E/$10/$00/    {       mov bx,[0010]           }
     $B9/$FF/$7F/        {       mov cx,7fff             }
     $BA/$00/$80/        {       mov dx,8000             }
     $8E/$DA/            {       mov ds,dx               }
     $BA/$14/$00/        {       mov dx,0014             }
     $CD/$21/            {       int 21                  }
     $72/$13/            {       jb error                }
     $B4/$3E/            {       mov ah,3e               }
     $BA/$00/$80/        {       mov dx,8000             }
     $8E/$DA/            {       mov ds,dx               }
     $8B/$1E/$10/$00/    {       mov bx,[0010]           }
     $CD/$21/            {       int 21                  }
     $72/$04/            {       jb error                }
     $B3/$00/            {       mov bl,00               }
     $EB/$02/            {       jmp fin                 }
     $B3/$01/            {error: mov bl,01               }
     $88/$1E/$10/$00/    {  fin: mov [0010],bl           }
     $59/                {       pop cx                  }
     $5B/                {       pop bx                  }
     $58/                {       pop ax                  }
     $8E/$D0/            {       mov ss,ax               }
     $89/$DC/            {       mov sp,bx               }
     $89/$CD/            {       mov bp,cx               }
     $1F );              {       pop ds                  }
END;
($F-)

{   ..........................  PROCEDIMIENTO  DESP_ARCH  ..............................}
{                                                                                       }
{   Procedimiento que despliega los datos de un archivo en forma hexadecimal            }
{   o en volts.                                                                         }

PROCEDURE DESP_ARCH;
VAR
    KK: INTEGER;
    CANAL_VOLT: REAL;
```

```
BEGIN
    KK:=0;
    LOC_HEXO:='0000';
    RENGLONES:=8;
    Z_PARAMETRO:=TRUE;
    NOMBRE_ARCH;
    ARCH_BUFFER;
    REPEAT
        CLRSCR;
        WRITELN;
        WRITE(' ':16,'DESPLEGADO DEL ARCHIVO: ',ARCH_EXTERNO,'.');
        IF Z_PARAMETRO THEN
        BEGIN
            WRITELN;
            WRITELN(' ':15,'          ZONA DE PARAMETROS: ',CUENTA);
        END;
        IF OPCION2='2' THEN
        BEGIN
            WRITELN;
            WRITELN(' ':26,'(DATOS EN VOLTS).');
            CASE NO_CANALES OF
             1: WRITELN('LOCALIDAD',' ':5,'CANAL ',CANAL[1]);
             2: WRITELN('LOCALIDAD',' ':5,'CANAL ',CANAL[1],' ':8,'CANAL ',CANAL[2]);
             3: WRITELN('LOCALIDAD',' ':5,'CANAL ',CANAL[1],' ':8,'CANAL ',CANAL[2],' ':8,'CANAL ',CANAL[3])

             4: WRITELN('LOCALIDAD',' ':5,'CANAL ',CANAL[1],' ':8,'CANAL ',CANAL[2],
                ' ':8,'CANAL ',CANAL[3],' ':8,'CANAL ',CANAL[4]);
            END;
            WRITELN;
        END
        ELSE
            WRITELN(' ':160);
        FORMATO;
        JJ:=1;
        II:=1;
        REPEAT
            LOC_HEX:=LOC_HEXO;
            HEX_DEC;
            LOC_DEC:=LOC_DEC+20;
            CONTIENE:=MEM[$8000:LOC_DEC];
            LOC_DEC:=LOC_DEC-20;
            POSICION:=2;
            KK:=LOC_DEC;
            IF OPCION2='1' THEN
            BEGIN
                LOC_DEC:=CONTIENE;
                DEC_HEX;
                WRITE(LOC_HEX,'  ');
            END;
            IF OPCION2='2' THEN
            BEGIN
                K:=(LOC_DEC+1) MOD 16;
                IF K=0 THEN K:=16;
                FOR I:=1 TO NO_CANALES DO
                IF CANAL[I]<K THEN
                BEGIN
                    IF (CONTIENE=128) OR (CONTIENE=127) THEN
                    BEGIN
                        CANAL_VOLT:=0;
                        WRITE('      ',CANAL_VOLT:2:4,'  ');
                    END;
                    IF CONTIENE>128 THEN
                    BEGIN
                        CANAL_VOLT:=(CONTIENE-128)*0.007874;
                        WRITE('      ',CANAL_VOLT:2:4,'  ');
                    END;
                    IF CONTIENE<127 THEN
```

```
                BEGIN
                    CANAL_VOLT:=(127-CONTIENE)*0.007874;
                    WRITE('     -',CANAL_VOLT:2:4,'   ');
                END;
            END;
        END;
        LOC_DEC:=KK+1;
        POSICION:=4;
        DEC_HEX;
        LOC_HEXO:=LOC_HEX;
        IF JJ=16 THEN
        BEGIN
            WRITELN;
            IF (II<RENGLONES) AND (LOC_DEC<>32768) THEN FORMATO;
            II:=II+1;
            JJ:=0;
        END;
        JJ:=JJ+1;
    UNTIL (II>RENGLONES) OR (LOC_DEC=32768);
    IF LOC_DEC=32768 THEN
    BEGIN
        OPCION:='3';
        WRITELN(' ':400);
        WRITELN(' ':16,'TERMINO DE MOSTRAR ',CUENTA,'a. RAM DEL ARCHIVO');
        SEGUIR_DES;
    END
    ELSE
        DA_OPCION;
    ESCOGE_OPCION;
UNTIL OPCION='3';
END;

{   .....................  PROCEDIMIENTO  COLOCA_NOMBRE_CANAL  ..................}
{                                                                               }
{   Procedimiento que coloca los letreros de los canales a graficar.            }

PROCEDURE COLOCA_NOMBRE_CANAL;
BEGIN
    OUTTEXTXY(20,J,'CANAL ');
    CASE CANAL[I] OF
    1:  OUTTEXTXY(70,J,'1');
    2:  OUTTEXTXY(70,J,'2');
    3:  OUTTEXTXY(70,J,'3');
    4:  OUTTEXTXY(70,J,'4');
    5:  OUTTEXTXY(70,J,'5');
    6:  OUTTEXTXY(70,J,'6');
    7:  OUTTEXTXY(70,J,'7');
    8:  OUTTEXTXY(70,J,'8');
    9:  OUTTEXTXY(70,J,'9');
    10: OUTTEXTXY(70,J,'10');
    11: OUTTEXTXY(70,J,'11');
    12: OUTTEXTXY(70,J,'12');
    13: OUTTEXTXY(70,J,'13');
    14: OUTTEXTXY(70,J,'14');
    15: OUTTEXTXY(70,J,'15');
    16: OUTTEXTXY(70,J,'16')
    ELSE
    END;
END;

{   ........................  PROCEDIMIENTO  INTERPOLA  ..........................}
{                                                                               }
{   Procedimiento que interpola los valores del eje Y al realizar una graficacion.}

PROCEDURE INTERPOLA;
VAR
```

```
    X_ANT,DIF,A,B: INTEGER;

BEGIN
    X_ANT:=X-1;
    IF Y_ANT[I]=-1 THEN
       Y_ANT[I]:=Y[I];
    DIF:=(ABS(Y[I]-Y_ANT[I]))-1;
    IF (DIF>0) THEN
        IF (Y_ANT[I]<Y[I]) THEN
        BEGIN
            IF ODD(DIF) THEN
                B:=0
            ELSE
                B:=1;
            A:=TRUNC(DIF DIV 2) + B;
            FOR B:=1 TO A DO
                PUTPIXEL(X_ANT,Y_ANT[I]+B,COLOR);
            FOR B:=B TO DIF DO
                PUTPIXEL(X,Y_ANT[I]+B,COLOR);
        END
        ELSE
        BEGIN
            IF ODD(DIF) THEN
                B:=1
            ELSE
                B:=0;
            A:=TRUNC(DIF DIV 2) + B;
            FOR B:=1 TO A DO
                PUTPIXEL(X_ANT,Y_ANT[I]-B,COLOR);
            FOR B:=B TO DIF DO
                PUTPIXEL(X,Y_ANT[I]-B,COLOR);
        END;
END;

{  ..................  PROCEDIMIENTO  SONIDO_ERROR  ..................}
{                                                                      }
{  Procedimiento envia un sonido cuando se presiona una tecla equivocada en la  }
{  graficación de un archivo.                                          }

PROCEDURE SONIDO_ERROR;
BEGIN
    SOUND(220);
    DELAY(50);
    NOSOUND;
END;

{  ..................  PROCEDIMIENTO  GRAFICA1 ..................}
{                                                                  }
{  Procedimiento que grafica el numero de canales que se indicaron, a la amplitud}
{  y decimacion dada.                                              }

PROCEDURE GRAFICA1;
VAR
             MANEJADOR,MODO: INTEGER;
                   PANTALLA: INTEGER;
                 INICIACION: WORD;
               PUNTO_CINCO: BOOLEAN;

BEGIN
    CLRSCR;
    TOTAL:=2040*NO_CANALES*CUENTA;
    INICIACION:=INICIO_SISMO;
    PUNTO_CINCO:=FALSE;
    PANTALLA:=1;
    DETECTGRAPH(MANEJADOR,MODO);
    IF (MANEJADOR = EGA) OR (MANEJADOR = EGA64) THEN
```

```
BEGIN
   MANEJADOR:= CGA;
   MODO:=CGAHi;
END;
INITGRAPH(MANEJADOR,MODO,'');
IF GRAPHRESULT <> grOk THEN
    HALT(1);
REPEAT
    CLEARDEVICE;
    LINE(0,0,625,0);
    LINE(625,0,625,199);
    LINE(625,199,0,199);
    LINE(0,199,0,0);
    LINE(100,0,100,199);
    MOVETO(0,0);
    COLOR:=GETMAXCOLOR;
    I:=0;
    REPEAT
        FOR J:=100 TO 102 DO
            PUTPIXEL(J,I,COLOR);
        FOR J:=623 TO 625 DO
            PUTPIXEL(J,I,COLOR);
        CONTADOR:=I MOD 25;
        IF CONTADOR=0 THEN
        BEGIN
            FOR J:=96 TO 104 DO
                PUTPIXEL(J,I,COLOR);
            FOR J:=621 TO 625 DO
                PUTPIXEL(J,I,COLOR);
        END;
        I:=I+5;
    UNTIL I=200;
    I:=125;
    REPEAT
        FOR J:=197 TO 199 DO
            PUTPIXEL(1,J,COLOR);
        I:=I+25;
    UNTIL I=650;
    OUTTEXTXY(535,2,'Ayuda <F1>');
    IF PUNTO_CINCO THEN
        CASE PANTALLA OF
            1: OUTTEXTXY(400,2,'Pantalla: 1.5');
            2: OUTTEXTXY(400,2,'Pantalla: 2.5');
            3: OUTTEXTXY(400,2,'Pantalla: 3.5');
            4: OUTTEXTXY(400,2,'Pantalla: 4.5');
            5: OUTTEXTXY(400,2,'Pantalla: 5.5');
            6: OUTTEXTXY(400,2,'Pantalla: 6.5');
            7: OUTTEXTXY(400,2,'Pantalla: 7.5');
            8: OUTTEXTXY(400,2,'Pantalla: 8.5');
            9: OUTTEXTXY(400,2,'Pantalla: 9.5');
        END
    ELSE
        CASE PANTALLA OF
            1: OUTTEXTXY(400,2,'Pantalla: 1');
            2: OUTTEXTXY(400,2,'Pantalla: 2');
            3: OUTTEXTXY(400,2,'Pantalla: 3');
            4: OUTTEXTXY(400,2,'Pantalla: 4');
            5: OUTTEXTXY(400,2,'Pantalla: 5');
            6: OUTTEXTXY(400,2,'Pantalla: 6');
            7: OUTTEXTXY(400,2,'Pantalla: 7');
            8: OUTTEXTXY(400,2,'Pantalla: 8');
            9: OUTTEXTXY(400,2,'Pantalla: 9');
        END;
    IF AMPLITUD=1 THEN
        OUTTEXTXY(260,2,'Amplitud: 1')
    ELSE
```

```
    OUTTEXTXY(260,2,'Amplitud: 2');
CASE DECIMA OF
    1:BEGIN
          OUTTEXTXY(483,180,'X: 0.625 Seg/div');
          OUTTEXTXY(110,2,'Decimacion: 1');
      END;
    2:BEGIN
          OUTTEXTXY(483,180,'X: 1.25 Seg/div.');
          OUTTEXTXY(110,2,'Decimacion: 2');
      END;
    3:BEGIN
          OUTTEXTXY(483,180,'X: 1.875 Seg/div');
          OUTTEXTXY(110,2,'Decimacion: 3');
      END;
    4:BEGIN
          OUTTEXTXY(483,180,'X: 2.50 Seg/div.');
          OUTTEXTXY(110,2,'Decimacion: 4');
      END;
    5:BEGIN
          OUTTEXTXY(483,180,'X: 3.125 Seg/div');
          OUTTEXTXY(110,2,'Decimacion: 5');
      END;
END;
IF OPCION=#59 THEN
BEGIN
    OUTTEXTXY(8,2,'          ');
    OUTTEXTXY(15,10,'Pantalla ');
    OUTTEXTXY(10,18,'siguiente.');
    PUTPIXEL(39,27,COLOR);
    PUTPIXEL(39,28,COLOR); PUTPIXEL(40,28,COLOR);
    PUTPIXEL(39,29,COLOR); PUTPIXEL(40,29,COLOR); PUTPIXEL(41,29,COLOR);
    FOR I:=28 TO 43 DO
        PUTPIXEL(I,30,COLOR);
    PUTPIXEL(39,31,COLOR); PUTPIXEL(40,31,COLOR); PUTPIXEL(41,31,COLOR);
    PUTPIXEL(39,32,COLOR); PUTPIXEL(40,32,COLOR);
    PUTPIXEL(39,33,COLOR);
    OUTTEXTXY(8,34,'          ');
    OUTTEXTXY(15,42,'Pantalla');
    OUTTEXTXY(14,50,'anterior.');
    PUTPIXEL(32,59,COLOR);
    PUTPIXEL(31,60,COLOR); PUTPIXEL(32,60,COLOR);
    PUTPIXEL(30,61,COLOR); PUTPIXEL(31,61,COLOR); PUTPIXEL(32,61,COLOR);
    FOR I:=28 TO 43 DO
        PUTPIXEL(I,62,COLOR);
    PUTPIXEL(30,63,COLOR); PUTPIXEL(31,63,COLOR); PUTPIXEL(32,63,COLOR);
    PUTPIXEL(31,64,COLOR); PUTPIXEL(32,64,COLOR);
    PUTPIXEL(32,65,COLOR);
    OUTTEXTXY(8,66,'          ');
    OUTTEXTXY(11,74,' Mayor  ');
    OUTTEXTXY(6,82,'decimacion.');
    FOR I:=37 TO 39 DO
        PUTPIXEL(I,93,COLOR);
    FOR I:=36 TO 40 DO
        PUTPIXEL(I,94,COLOR);
    FOR I:=35 TO 41 DO
        PUTPIXEL(I,95,COLOR);
    FOR I:=34 TO 42 DO
        PUTPIXEL(I,96,COLOR);
    FOR I:=91 TO 104 DO
        PUTPIXEL(38,I,COLOR);
    OUTTEXTXY(8,106,'          ');
    OUTTEXTXY(11,114,' Menor  ');
    OUTTEXTXY(6,122,'decimacion.');
    FOR I:=37 TO 39 DO
        PUTPIXEL(I,142,COLOR);
    FOR I:=36 TO 40 DO
```

```
        PUTPIXEL(1,141,COLOR);
    FOR I:=35 TO 41 DO
        PUTPIXEL(1,140,COLOR);
    FOR I:=34 TO 42 DO
        PUTPIXEL(1,139,COLOR);
    FOR I:=131 TO 144 DO
        PUTPIXEL(38,I,COLOR);
    OUTTEXTXY(15,149,' Cambia ');
    OUTTEXTXY(16,157,'Amplitud');
    OUTTEXTXY(16,165,' <Home> ');
    OUTTEXTXY(11,179,'  Salir  ');
    OUTTEXTXY(10,187,'  <End>  ');
END;
CASE NO_CANALES OF
   1:BEGIN
        J:=97;
        I:=1;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        IF AMPLITUD=1 THEN
        BEGIN
            COOR[1]:=142;
            K:=3;
            OUTTEXTXY(80,54,' 1');
            OUTTEXTXY(80,140,'-1');
            OUTTEXTXY(483,189,'Y: 0.12 Volts/div');
        END
        ELSE
        BEGIN
            COOR[1]:=164;
            K:=2;
            OUTTEXTXY(80,35,' 1');
            OUTTEXTXY(80,162,'-1');
            OUTTEXTXY(483,189,'Y: 0.08 Volts/div');
        END;
     END;
   2:BEGIN
        J:=47;
        I:=1;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        J:=147;
        I:=2;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        IF AMPLITUD=1 THEN
        BEGIN
            COOR[1]:=82;
            COOR[2]:=182;
            K:=4;
            OUTTEXTXY(483,189,'Y: 0.16 Volts/div');
            OUTTEXTXY(80,16,' 1');
            OUTTEXTXY(80,80,'-1');
            OUTTEXTXY(80,116,' 1');
            OUTTEXTXY(80,180,'-1');
        END
        ELSE
        BEGIN
            COOR[1]:=92;
            COOR[2]:=192;
            K:=3;
            OUTTEXTXY(483,189,'Y: 0.12 Volts/div');
            OUTTEXTXY(80,4,' 1');
            OUTTEXTXY(80,90,'-1');
            OUTTEXTXY(80,104,' 1');
            OUTTEXTXY(80,190,'-1');
```

```
        END;

    END;
3:BEGIN
        J:=47;
        I:=1;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        J:=97;
        I:=2;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        J:=147;
        I:=3;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        IF AMPLITUD=1 THEN
        BEGIN
            COOR[1]:=66;
            COOR[2]:=116;
            COOR[3]:=166;
            K:=8;
            OUTTEXTXY(483,189,'Y: 0.32 Volts/div');
            OUTTEXTXY(80,31,' 1');
            OUTTEXTXY(80,63,'-1');
            OUTTEXTXY(80,81,' 1');
            OUTTEXTXY(80,113,'-1');
            OUTTEXTXY(80,131,' 1');
            OUTTEXTXY(80,163,'-1');
        END
        ELSE
        BEGIN
            COOR[1]:=72;
            COOR[2]:=122;
            COOR[3]:=172;
            K:=6;
            OUTTEXTXY(483,189,'Y: 0.24 Volts/div');
            OUTTEXTXY(80,26,' 1');
            OUTTEXTXY(80,68,'-1');
            OUTTEXTXY(80,76,' 1');
            OUTTEXTXY(80,118,'-1');
            OUTTEXTXY(80,126,' 1');
            OUTTEXTXY(80,168,'-1');
        END;
    END;
4:BEGIN
        J:=22;
        I:=1;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        J:=72;
        I:=2;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        J:=122;
        I:=3;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        J:=172;
        I:=4;
        IF OPCION<>#59 THEN
            COLOCA_NOMBRE_CANAL;
        IF AMPLITUD=1 THEN
        BEGIN
            COOR[1]:=41;
            COOR[2]:=91;
```

```
            COOR[3]:=141;
            COOR[4]:=191;
            K:=8;
            OUTTEXTXY(483,189,'Y: 0.32 Volts/div');
            OUTTEXTXY(80,7,' 1');
            OUTTEXTXY(80,38,'-1');
            OUTTEXTXY(80,57,' 1');
            OUTTEXTXY(80,88,'-1');
            OUTTEXTXY(80,107,' 1');
            OUTTEXTXY(80,138,'-1');
            OUTTEXTXY(80,157,' 1');
            OUTTEXTXY(80,188,'-1');
        END
        ELSE
        BEGIN
            COOR[1]:=45;
            COOR[2]:=95;
            COOR[3]:=145;
            COOR[4]:=195;
            K:=6;
            OUTTEXTXY(483,189,'Y: 0.24 Volts/div');
            OUTTEXTXY(80,2,' 1');
            OUTTEXTXY(80,42,'-1');
            OUTTEXTXY(80,52,' 1');
            OUTTEXTXY(80,92,'-1');
            OUTTEXTXY(80,102,' 1');
            OUTTEXTXY(80,142,'-1');
            OUTTEXTXY(80,152,' 1');
            OUTTEXTXY(80,192,'-1');
        END;
    END
    ELSE
END;        (fin del case)

X:=100;
Y[1]:=0;Y[2]:=0;Y[3]:=0;Y[4]:=0;
Y_ANT[1]:=-1;
Y_ANT[2]:=-1;
Y_ANT[3]:=-1;
Y_ANT[4]:=-1;
REPEAT
    FOR I:=1 TO NO_CANALES DO
    BEGIN
        CONTIENE:=MEM[$9000:INICIO_SISMO];
        Y[I]:=COOR[I]-TRUNC(CONTIENE/K);
        INTERPOLA;
        Y_ANT[I]:=Y[I];
        PUTPIXEL(X,Y[I],COLOR);
        INICIO_SISMO:=INICIO_SISMO+1;
    END;
    X:=X+1;
    INICIO_SISMO:=(INICIO_SISMO-NO_CANALES)+(DECIMA*NO_CANALES);
UNTIL (X=620) OR (INICIO_SISMO>=TOTAL);
I:=420;                                 (Avance de pantalla)
OPCION:=READKEY;
IF OPCION = #0 THEN
BEGIN
    OPCION:=READKEY;
    CASE OPCION OF
        #77: BEGIN
                PANTALLA:=PANTALLA+1;
             END;
        #75: IF PANTALLA=1 THEN
             BEGIN
                SONIDO_ERROR;
                INICIO_SISMO:=INICIO_SISMO-NO_CANALES*TRUNC((X+1)/2)*DECIMA;
```

```
                END
                ELSE
                BEGIN
                    PANTALLA:=PANTALLA-1;
                    INICIO_SISMO:=INICIO_SISMO-(NO_CANALES*(X+1)*DECIMA);
                END;
            #71: BEGIN
                    INICIO_SISMO:=INICIO_SISMO-NO_CANALES*TRUNC((X+1)/2)*DECIMA;
                    IF AMPLITUD=1 THEN
                       AMPLITUD:=2
                    ELSE
                       AMPLITUD:=1;
                END;
            #72: IF DECIMA=5 THEN
                BEGIN
                    SONIDO_ERROR;
                    INICIO_SISMO:=INICIO_SISMO-NO_CANALES*TRUNC((X+1)/2)*DECIMA;
                END
                ELSE
                BEGIN
                    INICIO_SISMO:=INICIACION;
                    DECIMA:=DECIMA+1;
                    PANTALLA:=1;
                END;
             #80: IF DECIMA=1 THEN
                BEGIN
                    SONIDO_ERROR;
                    INICIO_SISMO:=INICIO_SISMO-NO_CANALES*TRUNC((X+1)/2)*DECIMA;
                END
                ELSE
                BEGIN
                    INICIO_SISMO:=INICIACION;
                    DECIMA:=DECIMA-1;
                    PANTALLA:=1;
                END;
            #116: BEGIN
                    INICIO_SISMO:=INICIO_SISMO-NO_CANALES*TRUNC((X+1)/4)*DECIMA;
                    IF PUNTO_CINCO THEN
                    BEGIN
                       PANTALLA:=PANTALLA+1;
                       PUNTO_CINCO:=FALSE;
                    END
                    ELSE
                       PUNTO_CINCO:=TRUE;
                END;
             #59: INICIO_SISMO:=INICIO_SISMO-NO_CANALES*TRUNC((X+1)/2)*DECIMA;
              ELSE
          END;
      END
      ELSE
          INICIO_SISMO:=INICIO_SISMO-NO_CANALES*TRUNC((X+1)/2)*DECIMA;
   UNTIL (INICIO_SISMO>=TOTAL) OR (OPCION=#79);
   CLEARDEVICE;
   CLOSEGRAPH;
   CLRSCR;
END;

{   ..................... PROCEDIMIENTO  RESTA 30SEG .....................}
{                                                                         }
{   Procedimiento que le resta 30 segundos al tiempo que se obtiene dentro del    }
{   archivo de datos, el tiempo resultante es el tiempo de inicio del evento.    }

PROCEDURE RESTA_30SEG;
VAR
   MESES,MESES_BIS: ARRAY[1..12] OF BYTE;
         BISIESTO: BYTE;
```

```
BEGIN
    MESES[1]:=31;    MESES_BIS[1]:=31;
    MESES[2]:=28;    MESES_BIS[2]:=29;
    MESES[3]:=31;    MESES_BIS[3]:=31;
    MESES[4]:=30;    MESES_BIS[4]:=30;
    MESES[5]:=31;    MESES_BIS[5]:=31;
    MESES[6]:=30;    MESES_BIS[6]:=30;
    MESES[7]:=31;    MESES_BIS[7]:=31;
    MESES[8]:=31;    MESES_BIS[8]:=31;
    MESES[9]:=30;    MESES_BIS[9]:=30;
    MESES[10]:=31;   MESES_BIS[10]:=31;
    MESES[11]:=30;   MESES_BIS[11]:=30;
    MESES[12]:=31;   MESES_BIS[12]:=31;
    IF SEGUNDO>=30 THEN
        SEGUNDO:=SEGUNDO-30
    ELSE
        IF MINUTO>=1 THEN
        BEGIN
            MINUTO:=MINUTO-1;
            SEGUNDO:=SEGUNDO+30;
        END
        ELSE
            IF HORA>=1 THEN
            BEGIN
                HORA:=HORA-1;
                MINUTO:=59;
                SEGUNDO:=SEGUNDO+30;
            END
            ELSE
                IF DIA>1 THEN
                BEGIN
                    DIA:=DIA-1;
                    HORA:=23;
                    MINUTO:=59;
                    SEGUNDO:=SEGUNDO+30;
                END
                ELSE
                    IF MES>1 THEN
                    BEGIN
                        MES:=MES-1;
                        BISIESTO:=ANO MOD 4;
                        IF BISIESTO=0 THEN
                            DIA:=MESES_BIS[MES]
                        ELSE
                            DIA:=MESES[MES];
                        HORA:=23;
                        MINUTO:=59;
                        SEGUNDO:=SEGUNDO+30;
                    END
                    ELSE
                    BEGIN
                        ANO:=ANO-1;
                        MES:=12;
                        DIA:=31;
                        HORA:=23;
                        MINUTO:=59;
                        SEGUNDO:=SEGUNDO+30;
                    END;
END;

{   ..................... PROCEDIMIENTO  DATOS_SISMO .....................}
{                                                                        }
{   Procedimiento que despliega en pantalla los datos del sismo a graficar.   }

PROCEDURE DATOS_SISMO;
```

```
BEGIN
    MARCO;
    CASE NO_CANALES OF
      1: J:=9;
      2: J:=7;
      3: J:=6;
      4: J:=4;
    END;
    GOTOXY(2,2);
    WRITE('                              DATOS DEL SISMO A GRAFICAR:        ');
    GOTOXY(15,J);
    WRITE('ARCHIVO:  ',ARCH_EXTERNO);
    GOTOXY(15,J+1);
    WRITE('FECHA DEL SISMO:  ',DIA,' de ',NOMBRE_MES,' de ',ANO,'.');
    GOTOXY(15,J+2);
    WRITE('HORA DEL SISMO:  ');
    IF HORA>9 THEN
        WRITE(HORA,':')
    ELSE
        WRITE('0',HORA,':');
    IF MINUTO>9 THEN
        WRITE(MINUTO,':')
    ELSE
        WRITE('0',MINUTO,':');
    IF SEGUNDO>9 THEN
        WRITE(SEGUNDO,'.')
    ELSE
        WRITE('0',SEGUNDO,'.');
    IF DECSEG>9 THEN
        WRITE(DECSEG)
    ELSE
        WRITE('0',DECSEG);
    FOR I:=1 TO NO_CANALES DO
    BEGIN
        GOTOXY(15,4+J);
        WRITE('CANAL: ',CANAL[I]);
        GOTOXY(15,5+J);
        WRITE('ESTACION: ',ESTACION[CANAL[I]]);
        GOTOXY(15,6+J);
        WRITE('DISTANCIA: ',DISTANCIA[CANAL[I]]);
        J:=J+4;
    END;
    GOTOXY(58,23);
    WRITE('ESPERE UN MOMENTO.');
END;

{   ..................  PROCEDIMIENTO GUARDA_BUFFER  ....................}
{                                                                        }
{   Procedimiento que guarda dentro de un buffer de memoria los canales del evento  }
{   a graficarse.                                                        }

PROCEDURE GUARDA_BUFFER;
VAR
    SEGMENTO: WORD;
    NUM_RAMS: BYTE;
    ULTIMA_X: INTEGER;

BEGIN
    NUM_RAMS:=1;
    SEGMENTO:=0;
    REPEAT
        JJ:=1;
        II:=1;
        X:=100;
        ULTIMA_X:=X;
        LOC_DEC:=148;   {128+20}
```

```
        REPEAT
            JJ:=II MOD 16;
            IF JJ=0 THEN JJ:=16;
            FOR I:=1 TO NO_CANALES DO
            IF CANAL[I]=JJ THEN
            BEGIN
                CONTIENE:=MEM[$8000:LOC_DEC];
                MEM[$9000:SEGMENTO]:=CONTIENE;
                SEGMENTO:=SEGMENTO+1;
                IF I=NO_CANALES THEN
                BEGIN
                    X:=X+1;
                    LOC_DEC:=LOC_DEC+(17-JJ);
                    II:=II+(17-JJ);
                END;
            END;
            IF ULTIMA_X=X THEN
            BEGIN
                LOC_DEC:=LOC_DEC+1;
                II:=II+1;
            END
            ELSE
                ULTIMA_X:=X;
            TOTAL:=2040*NO_CANALES*NUM_RAMS;
        UNTIL SEGMENTO>=TOTAL;
        NUM_RAMS:=NUM_RAMS+1;
        IF NUM_RAMS<=CUENTA THEN
        BEGIN
            APUNTA_ARCH:=APUNTA_ARCH+32767;
            NOMBRE_ARCH;
            ARCH_BUFFER;
        END;
    UNTIL NUM_RAMS>CUENTA;
END;

{  ..................  PROCEDIMIENTO  NOMBRE_DE_MES  ..................}
{                                                                       }
{  Procedimiento que obtiene el nombre del mes para un numero dado.     }

PROCEDURE NOMBRE_DE_MES;
BEGIN
    NOMBRE_MES:='';
    CASE MES OF
       1: NOMBRE_MES:='Enero';
       2: NOMBRE_MES:='Febrero';
       3: NOMBRE_MES:='Marzo';
       4: NOMBRE_MES:='Abril';
       5: NOMBRE_MES:='Mayo';
       6: NOMBRE_MES:='Junio';
       7: NOMBRE_MES:='Julio';
       8: NOMBRE_MES:='Agosto';
       9: NOMBRE_MES:='Septiembre';
      10: NOMBRE_MES:='Octubre';
      11: NOMBRE_MES:='Noviembre';
      12: NOMBRE_MES:='Diciembre';
    END;
END;

{  ..................  PROCEDIMIENTO ALMACENA_GRAFICA  ..................}
{                                                                       }
{  Procedimiento que llama a otros procedimientos, para almacenar, graficar y  }
{  desplegar en pantalla los datos de un evento contenidos en un archivo.      }

PROCEDURE ALMACENA_GRAFICA;
VAR
                        RAM_INICIAL: BYTE;
```

```
                LOC_1_SISMO,AP_INICIAL: WORD;
                        DIFERENCIA: INTEGER;

BEGIN
    TABLA_ESTACIONES;
    DECSEG:=MEM[$8000:$0084];        {localidad 0070+14}
    SEGUNDO:=MEM[$8000:$0085];
    MINUTO:=MEM[$8000:$0086];
    HORA:=MEM[$8000:$0087];
    DIA:=MEM[$8000:$0088];
    MES:=MEM[$8000:$0089];
    ANO:=MEMW[$8000:$008C];
    RESTA_30SEG;
    NOMBRE_DE_MES;
    AP_INICIAL:=MEMW[$8000:$0035];   {21+14}
    DIFERENCIA:=AP_INICIAL-$4880;
    IF DIFERENCIA>=0 THEN
        LOC_1_SISMO:=DIFERENCIA
    ELSE
        LOC_1_SISMO:=$7FFF+DIFERENCIA;
    LOC_1_SISMO:=LOC_1_SISMO-(LOC_1_SISMO MOD 16);
    INICIO_SISMO:=NO_CANALES*(TRUNC(LOC_1_SISMO/16));
    DATOS_SISMO;
    GUARDA_BUFFER;
    GOTOXY(47,23);
    WRITE('PRESIONE <ENTER> PARA SEGUIR.');
    OPCION:=READKEY;
    GRAFICA1;
END;

PROCEDURE ARCH1_BUFFER;FORWARD;

{  ..................  PROCEDIMIENTO MANEJO_ARCHIVO  ..................}
{                                                                       }
{  Procedimiento que llama a un archivo de datos para desplegarlo o graficarlo.  }

PROCEDURE MANEJO_ARCHIVO;
VAR
        DIRINFO: SEARCHREC;
        BANDERA: BOOLEAN;
        CUENTA1: BYTE;
        CUENTA2: BYTE;

BEGIN
    REPEAT
        REPEAT
            CLRSCR;
            WRITELN;
            WRITELN('[redacted]');
            WRITELN('[redacted]');
            WRITELN(' ':1500);
            WRITELN('[redacted]');
            WRITELN('[redacted]');
            GOTOXY(60,10);
            WRITE('                ');
            GOTOXY(10,10);
            WRITE('PROPORCIONE EL NOMBRE DEL ARCHIVO (MMDDmmss.DAT): ');
            GOTOXY(20,12);
            WRITE('[D]   MUESTRA DIRECTORIO.');
            GOTOXY(20,14);
            WRITE('[S]   SALIDA A MENU PRINCIPAL.');
            GOTOXY(60,10);
            ARCH_EXTERNO := '';
            READLN(ARCH_EXTERNO);
            OPCION2:=ARCH_EXTERNO[1];
            BANDERA:=TRUE;
```

```
CASE OPCION2 OF
    '0','1': BEGIN
                LOC_HEX:=CONCAT('00',COPY(ARCH_EXTERNO,1,2));
                HEX_DEC;
                IF ((LOC_DEC>=1) AND (LOC_DEC<=18)) THEN
                    BANDERA:=TRUE
                ELSE
                    BANDERA:=FALSE;
                IF BANDERA THEN
                BEGIN
                    LOC_HEX:=CONCAT('00',COPY(ARCH_EXTERNO,3,2));
                    HEX_DEC;
                    IF ((LOC_DEC>=1) AND (LOC_DEC<=49)) THEN
                        BANDERA:=TRUE
                    ELSE
                        BANDERA:=FALSE;
                    IF BANDERA THEN
                    BEGIN
                        LOC_HEX:=CONCAT('00',COPY(ARCH_EXTERNO,5,2));
                        HEX_DEC;
                        IF ((LOC_DEC>=0) AND (LOC_DEC<=89)) THEN
                            BANDERA:=TRUE
                        ELSE
                            BANDERA:=FALSE;
                        IF BANDERA THEN
                        BEGIN
                            LOC_HEX:=CONCAT('00',COPY(ARCH_EXTERNO,5,2));
                            HEX_DEC;
                            IF ((LOC_DEC>=0) AND (LOC_DEC<=89)) THEN
                                BANDERA:=TRUE
                            ELSE
                                BANDERA:=FALSE;
                            IF BANDERA THEN
                            BEGIN
                                LOC_HEX:=COPY(ARCH_EXTERNO,9,1);
                                IF ((LOC_HEX='.') AND (COPY(ARCH_EXTERNO,12,1)='1')) THEN
                                    BANDERA:=TRUE
                                ELSE
                                    BANDERA:=FALSE;
                            END;
                        END;
                    END;
                END;
                IF BANDERA THEN
                BEGIN
                    CLRSCR;
                    APUNTA_ARCH:=0;
                    NOMBRE_ARCH;
                    ARCH1_BUFFER;
                    CONTIENE:=MEM[$8000:$0010];
                    IF CONTIENE=1 THEN
                    BEGIN
                        OPCION2:='3';
                        CUADRO;
                        WRITE('       EL ARCHIVO NO SE ENCUENTRA. ');
                        SOUND(220);
                        DELAY(200);
                        NOSOUND;
                        DELAY(2000);
                    END;
                    GOTOXY(25,10);
                    WRITE(' ':35);
                END
                ELSE
                    OPCION2:='3';
            END;
```

```
        'D','d': BEGIN
                    MARCO;
                    GOTOXY(27,2);
                    WRITE('DIRECTORIO DE DATOS: ');
                    I:=4;
                    J:=6;
                    FINDFIRST('D:\*.*',ARCHIVE,DIRINFO);
                    WHILE DOSERROR = 0 DO
                    BEGIN
                        GOTOXY(J,I);
                        WRITE(DIRINFO.NAME,'    ',DIRINFO.SIZE,' Bytes.');
                        FINDNEXT(DIRINFO);
                        I:=I+1;
                        IF I=23 THEN
                        BEGIN
                            J:=J+36;
                            IF J=78 THEN
                            BEGIN
                                DELAY(2000);
                                MARCO;
                                GOTOXY(27,2);
                                WRITE('DIRECTORIO DE DATOS: ');
                                I:=4;
                                J:=6;
                            END;
                            I:=4;
                        END;
                    END;
                    GOTOXY(48,23);
                    WRITE('PRESIONE <ENTER> PARA SALIR.');
                    OPCION2:=READKEY;
                    DELAY(100);
                    OPCION2:='3';
                END;
        'S','s': BEGIN
                    OPCION2:='4';
                END
         ELSE    BEGIN
                    OPCION2:='3';
                END;
    END;
UNTIL OPCION2<>'3';
IF OPCION2='4' THEN OPCION2:='3';
IF OPCION2<>'3' THEN
BEGIN
    REPEAT
        CLRSCR;
        WRITELN(' ':320);
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':7,'                                                                    ');
        WRITELN(' ':160);
        WRITE(' ':20,'OPCION: ');
        OPCION2:=READKEY;
    UNTIL OPCION2 IN ['1','2','3'];
    IF OPCION2='1' THEN
```

```
                                M E N U:


        1.- DESPLIEGA CONTENIDO DE LOS 16 CANALES DEL ARCHIVO.

        2.- GRAFICA DE LOS CANALES DEL ARCHIVO.

        3.- SALIDA A MENU PRINCIPAL.
```

```
BEGIN
    LOC_HEX:='0000';
    INICIO;
    REPEAT
        CLRSCR;
        WRITELN(' ':320);
        WRITELN(' ':6,'
        WRITELN(' ':6,'
        WRITELN(' ':6,'                          M E N U:
        WRITELN(' ':6,'
        WRITELN(' ':6,'
        WRITELN(' ':6,'
        WRITELN(' ':6,'                1.- DESPLEGADO EN HEXADECIMAL.
        WRITELN(' ':6,'
        WRITELN(' ':6,'                2.- DESPLEGADO EN VOLTS.
        WRITELN(' ':6,'
        WRITELN(' ':6,'
        WRITELN(' ':6,'
        WRITELN(' ':160);
        WRITE(' ':20,'OPCION: ');
        OPCION2:=READKEY;
    UNTIL OPCION2 IN ['1','2'];
    IF (OPCION2='1') OR (OPCION2='2') THEN
    BEGIN
        CUENTA1:=0;
        LETRERO:='';
        LETRERO:='D:\'+COPY(ARCH_EXTERNO,1,12);
        FINDFIRST(LETRERO,ARCHIVE,DIRINFO);
        LETRERO:='';
        REPEAT
            CUENTA1:=CUENTA1+1;
            CLRSCR;
            CUENTA:=TRUNC(DIRINFO.SIZE/32765);
            CUENTA2:=CUENTA;
            WRITELN;
            WRITELN('
'');
            WRITELN('
'');
            WRITELN(' ':1500);
            WRITELN('
'');
            WRITELN('
'');
            GOTOXY(22,12);
            WRITE('EL ARCHIVO ESTA FORMADO POR  ',CUENTA,' RAM''S. ');
            GOTOXY(15,14);
            WRITE('CUAL RAM DESEA DESPLEGAR [1-',CUENTA,'] ó [0] PARA SALIR: ');
            READLN(CUENTA);
            IF CUENTA<>0 THEN
            BEGIN
                CASE CUENTA OF
                    1: APUNTA_ARCH:=0;
                    2: APUNTA_ARCH:=32767;
                    3: APUNTA_ARCH:=65535;
                END;
                IF OPCION2='2' THEN
                BEGIN
                    LETRERO:='DESPLEGAR';
                    PREGUNTA_CANAL;
                    ORDENA_CANAL;
                    LOCALIDAD_A_DESP;
                    CLRSCR;
                END;
                LOCALIDAD_A_DESP;
                DESP_ARCH;
            END;
        UNTIL (CUENTA1=CUENTA2) OR (CUENTA=0);
    END;
```

```
            OPCION2:='1';
        END;
        IF OPCION2='2' THEN
        BEGIN
            LETRERO:=' GRAFICAR';
            PREGUNTA_CANAL;
            ORDENA_CANAL;
            GOTOXY(14,J+10);
            REPEAT
                WRITE('DE LA DECIMACION [1,2,3,4,5]:  ');
                READ(DECIMA);
            UNTIL DECIMA IN [1,2,3,4,5];
            GOTOXY(14,J+12);
            REPEAT
                WRITE('DE LA AMPLITUD [1,2]:  ');
                READ(AMPLITUD);
            UNTIL AMPLITUD IN [1,2];
            LETRERO:='';
            LETRERO:='D:\'+COPY(ARCH_EXTERNO,1,12);
            FINDFIRST(LETRERO,ARCHIVE,DIRINFO);
            LETRERO:='';
            CUENTA:=TRUNC(DIRINFO.SIZE/32765);
            ALMACENA_GRAFICA;
        END;
    END;
UNTIL OPCION2='3';
END;
```

(A CONTINUACION: $I ESTADISTICA)

```
{  ..................  PROCEDIMIENTO ARCH1_BUFFER  ..........................}
{                                                                             }
{  Procedimiento que lee de el archivo solo la zona de datos que contiene el  }
{  tiempo de inicio del evento, el canal donde disparo, el apuntador inicial, }
{  y la RAM de conmutación inicial.  Estos datos los almacena en memoria.     }

($F+)
PROCEDURE ARCH1_BUFFER;
BEGIN
    INLINE
    ($1E/                {       push ds                 }
     $8C/$D0/            {       mov ax,ss               }
     $89/$E3/            {       mov bx,sp               }
     $89/$E9/            {       mov cs,bp               }
     $50/                {       push ax                 }
     $53/                {       push bx                 }
     $51/                {       push cx                 }
     $B4/$3D/            {       mov ah,3d               }
     $B0/$02/            {       mov al,02               }
     $BA/$00/$80/        {       mov dx,8000             }
     $8E/$DA/            {       mov ds,dx               }
     $BA/$00/$00/        {       mov dx,0000             }
     $CD/$21/            {       int 21                  }
     $73/$02/            {       jnb sigue               }
     $EB/$26/            {       jmp error               }
     $A3/$10/$00/        {sigue: mov [0010],ax           }
     $B4/$3F/            {       mov ah,3f               }
     $BB/$1E/$10/$00/    {       mov bx,[0010]           }
     $B9/$7F/$00/        {       mov cx,7fff             }
     $BA/$00/$80/        {       mov dx,8000             }
     $8E/$DA/            {       mov ds,dx               }
     $BA/$14/$00/        {       mov dx,0014             }
     $CD/$21/            {       int 21                  }
     $72/$0E/            {       jb error                }
     $B4/$3E/            {       mov ah,3e               }
     $8B/$1E/$10/$00/    {       mov bx,[0010]           }
     $CD/$21/            {       int 21                  }
     $72/$04/            {       jb error                }
     $B3/$00/            {       mov bl,00               }
     $EB/$02/            {       jmp fin                 }
     $B3/$01/            {error: mov bl,01               }
     $88/$1E/$10/$00/    {  fin: mov [0010],bl           }
     $59/                {       pop cx                  }
     $5B/                {       pop bx                  }
     $58/                {       pop ax                  }
     $8E/$D0/            {       mov ss,ax               }
     $89/$DC/            {       mov sp,bx               }
     $89/$CD/            {       mov bp,cx               }
     $1F );              {       pop ds                  }
END;
($F-)

{  ..................  PROCEDIMIENTO OBTIENE_FECHA  .........................}
{                                                                             }
{  Procedimiento que obtiene la fecha de inicio del evento del archivo de datos. }

PROCEDURE OBTIENE_FECHA;
BEGIN
    DECSEG:=MEM[$8000:$0084];        {localidad 0070+14}
    SEGUNDO:=MEM[$8000:$0085];
    MINUTO:=MEM[$8000:$0086];
    HORA:=MEM[$8000:$0087];
    DIA:=MEM[$8000:$0088];
    MES:=MEM[$8000:$0089];
    ANO:=MEMW[$8000:$008C];
```

```
END;

{  ..................... PROCEDIMIENTO DESPLIEGA_FECHA .....................}
{                                                                           }
{  Procedimiento que despliega en pantalla la fecha del evento contenido en el  }
{  archivo leido.                                                           }

PROCEDURE DESPLIEGA_FECHA;
BEGIN
    IF HORA>9 THEN
        WRITE(HORA,':')
    ELSE
        WRITE('0',HORA,':');
    IF MINUTO>9 THEN
        WRITE(MINUTO,':')
    ELSE
        WRITE('0',MINUTO,':');
    IF SEGUNDO>9 THEN
        WRITE(SEGUNDO,'.')
    ELSE
        WRITE('0',SEGUNDO,'.');
    IF DECSEG>9 THEN
        WRITE(DECSEG)
    ELSE
        WRITE('0',DECSEG);
END;

{  ..................... PROCEDIMIENTO ESTADISTICA ..........................}
{                                                                           }
{  Procedimiento que lee todo el directorio de datos y obtiene de cada uno de los}
{  archivos que lo forman los datos principales del evento que contienen como  }
{  son el tiempo de inicio del evento, el canal y estacion de disparo.      }

PROCEDURE ESTADISTICA;
VAR
      DIRINFO1:SEARCHREC;
    CANAL_DISP: BYTE;
      HORA_ANT: BYTE;
    MINUTO_ANT: BYTE;
   SEGUNDO_ANT: BYTE;
    DECSEG_ANT: BYTE;
       DIA_ANT: BYTE;
       MES_ANT: BYTE;
       AHO_ANT: WORD;
        COLUMN: BYTE;
  NOMBRE_FECHA: STRING[8];
          BIEN: BOOLEAN;

BEGIN
    REPEAT
        CLRSCR;
        WRITELN(' ':320);
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':12,'                    M E N U:                        ');
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':12,'    1.- DESPLIEGA EVENTOS OCURRIDOS EN UNA ESTACION.');
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':12,'    2.- DESPLIEGA EVENTOS OCURRIDOS EN UNA FECHA.   ');
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':12,'    3.- DESPLIEGA TODOS LOS EVENTOS.                ');
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':12,'    4.- SALIDA A MENU PRINCIPAL.                    ');
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':12,'                                                    ');
        WRITELN(' ':160);
```

```
    WRITE(' ':20,'OPCION: ');
    OPCION2:=READKEY;
UNTIL OPCION2 IN ['1','2','3','4'];
IF OPCION2<>'4' THEN
BEGIN
    MARCO;
    GOTOXY(27,2);
    WRITE('DIRECTORIO DE DATOS: ');
    J:=5;
    HORA_ANT:=0;
    MINUTO_ANT:=0;
    SEGUNDO_ANT:=0;
    DECSEG_ANT:=0;
    CASE OPCION2 OF
        '1': BEGIN
                I:=1;
                K:=4;
                GOTOXY(27,2);
                WRITE('E S T A C I O N E S: ');
                TABLA_ESTACIONES;
                REPEAT
                    GOTOXY(10,K);
                    WRITE(I,'. ',ESTACION[I]);
                    K:=K+2;
                    I:=I+1;
                UNTIL I=10;
                GOTOXY(10,23);
                WRITE('PROPORCIONE EL NUMERO DE LA ESTACION: ');
                OPCION2:=READKEY;
                K:=ORD(OPCION2)-48;
                I:=2;
                REPEAT
                    GOTOXY(4,I);
                    WRITE(' ':72,'');
                    I:=I+1;
                UNTIL I=24;
                GOTOXY(15,2);
                WRITE('INFORMACION DE LA ESTACION: ',ESTACION[K]);
                GOTOXY(4,4);
                WRITE('  ARCHIVO',' ':13,'FECHA',' ':11,'HORA');
                FINDFIRST('D:\*.*',ARCHIVE,DIRINFO1);
                WHILE DOSERROR = 0 DO
                BEGIN
                    ARCH_EXTERNO:=' ';
                    ARCH_EXTERNO:=DIRINFO1.NAME;
                    NOMBRE_ARCH;
                    ARCH1_BUFFER;
                    CANAL_DISP:=MEM[$8000:$003C];
                    I:=TRUNC(CANAL_DISP/2)+1;
                    IF I=K THEN
                    BEGIN
                        OBTIENE_FECHA;
                        RESTA_30SEG;
                        NOMBRE_DE_MES;
                        GOTOXY(4,J);
                        WRITE(ARCH_EXTERNO);
                        GOTOXY(21,J);
                        IF ((HORA=HORA_ANT) AND (MINUTO=MINUTO_ANT)) AND
                         ((SEGUNDO=SEGUNDO_ANT) AND (DECSEG=DECSEG_ANT)) THEN
                           WRITE ('     ARCHIVO VACIO.')
                        ELSE
                        BEGIN
                            NOMBRE_MES:=COPY(NOMBRE_MES,1,3);
                            IF DIA>9 THEN
                                WRITE(DIA,'/ ',NOMBRE_MES,'./ ',AHO)
                            ELSE
```

```
                    WRITE('0',DIA,'/ ',NOMBRE_MES,'./ ',AHO);
                  GOTOXY(39,J);
                  DESPLIEGA_FECHA;
                END;
                J:=J+1;
            END;
            HORA_ANT:=HORA;
            MINUTO_ANT:=MINUTO;
            SEGUNDO_ANT:=SEGUNDO;
            DECSEG_ANT:=DECSEG;
            FINDNEXT(DIRINFO1);
            IF (J>23) AND (DOSERROR = 0) THEN
            BEGIN
                GOTOXY(47,23);
                WRITE('PRESIONE <ENTER> PARA SEGUIR.');
                OPCION2:=READKEY;
                MARCO;
                GOTOXY(27,2);
                WRITE('DIRECTORIO DE DATOS: ');
                J:=5;
                GOTOXY(4,4);
                WRITE('  ARCHIVO',' ':13,'FECHA',' ':11,'HORA');
            END;
        END;
    END;
'2': BEGIN
        COLUMN:=5;
        GOTOXY(27,2);
        WRITE('                                    ');
        REPEAT
            BIEN:=TRUE;
            GOTOXY(15,11);
            WRITE('PROPORCIONE LA FECHA (DD/MM/AA):                ');
            GOTOXY(48,11);
            READ(NOMBRE_FECHA);
            IF (NOMBRE_FECHA[3]='/') AND (NOMBRE_FECHA[6]='/') THEN
            BEGIN
                DIA_ANT:=((ORD(NOMBRE_FECHA[1])-48)*10)+(ORD(NOMBRE_FECHA[2])-48);
                MES_ANT:=((ORD(NOMBRE_FECHA[4])-48)*10)+(ORD(NOMBRE_FECHA[5])-48);
                AHO_ANT:=((ORD(NOMBRE_FECHA[7])-48)*10)+(ORD(NOMBRE_FECHA[8])-48);
                AHO_ANT:=1900+AHO_ANT;
            END
            ELSE
                BIEN:=FALSE;
            IF (NOMBRE_FECHA[3]='/') AND (NOMBRE_FECHA[5]='/') THEN
            BEGIN
                DIA_ANT:=((ORD(NOMBRE_FECHA[1])-48)*10)+(ORD(NOMBRE_FECHA[2])-48);
                MES_ANT:=ORD(NOMBRE_FECHA[4])-48;
                AHO_ANT:=((ORD(NOMBRE_FECHA[6])-48)*10)+(ORD(NOMBRE_FECHA[7])-48);
                AHO_ANT:=1900+AHO_ANT;
            END
            ELSE
                BIEN:=FALSE;
            IF (NOMBRE_FECHA[2]='/') AND (NOMBRE_FECHA[4]='/') THEN
            BEGIN
                DIA_ANT:=ORD(NOMBRE_FECHA[1])-48;
                MES_ANT:=ORD(NOMBRE_FECHA[3])-48;
                AHO_ANT:=((ORD(NOMBRE_FECHA[5])-48)*10)+(ORD(NOMBRE_FECHA[6])-48);
                AHO_ANT:=1900+AHO_ANT;
            END
            ELSE
                BIEN:=FALSE;
            IF (NOMBRE_FECHA[2]='/') AND (NOMBRE_FECHA[5]='/') THEN
            BEGIN
                DIA_ANT:=ORD(NOMBRE_FECHA[1])-48;
                MES_ANT:=((ORD(NOMBRE_FECHA[3])-48)*10)+(ORD(NOMBRE_FECHA[4])-48);
```

```
            AHO_ANT:=((ORD(NOMBRE_FECHA[6])-48)*10)+(ORD(NOMBRE_FECHA[7])-48);
             AHO_ANT:=1900+AHO_ANT;
        END
        ELSE
            BIEN:=FALSE;
        IF (((DIA_ANT<=31) AND (DIA_ANT>0)) AND ((MES_ANT<=12) AND (MES_ANT>0)))
           AND (AHO_ANT>=1980) THEN
            BIEN:=TRUE
        ELSE
            BIEN:=FALSE;
    UNTIL BIEN;
    GOTOXY(15,11);
    WRITE('                                                              ');
    MES:=MES_ANT;
    NOMBRE_DE_MES;
    GOTOXY(10,2);
    WRITE('EN LA FECHA:  ',DIA_ANT,' de ',NOMBRE_MES,' de ',AHO_ANT,'.  LOS EVENTOS SON:');
    GOTOXY(4,4);
    WRITE('  ARCHIVO',' ':11,'HORA',' ':19,'ESTACION');
    FINDFIRST('D:\*.*',ARCHIVE,DIRINFO1);
    WHILE DOSERROR = 0 DO
    BEGIN
        ARCH_EXTERNO:=' ';
        ARCH_EXTERNO:=DIRINFO1.NAME;
        NOMBRE_ARCH;
        ARCH1_BUFFER;
        OBTIENE_FECHA;
        RESTA_30SEG;
        IF (((DIA_ANT=DIA) AND (MES_ANT=MES)) AND (AHO_ANT=AHO)) THEN
        BEGIN
            NOMBRE_DE_MES;
            CANAL_DISP:=MEM[$8000:$003C];
            K:=TRUNC(CANAL_DISP/2)+1;
            TABLA_ESTACIONES;
            GOTOXY(4,COLUMN);
            WRITE(ARCH_EXTERNO);
            GOTOXY(21,COLUMN);
            IF ((HORA=HORA_ANT) AND (MINUTO=MINUTO_ANT)) AND
              ((SEGUNDO=SEGUNDO_ANT) AND (DECSEG=DECSEG_ANT)) THEN
                WRITE ('     ARCHIVO VACIO.')
            ELSE
            BEGIN
                DESPLIEGA_FECHA;
                GOTOXY(43,COLUMN);
                WRITE(ESTACION[K]);
            END;
            COLUMN:=COLUMN+1;
        END;
        HORA_ANT:=HORA;
        MINUTO_ANT:=MINUTO;
        SEGUNDO_ANT:=SEGUNDO;
        DECSEG_ANT:=DECSEG;
        FINDNEXT(DIRINFO1);
        IF (COLUMN=23) AND (DOSERROR = 0) THEN
        BEGIN
            GOTOXY(47,23);
            WRITE('PRESIONE <ENTER> PARA SEGUIR.');
            OPCION2:=READKEY;
            MARCO;
            COLUMN:=5;
            GOTOXY(4,4);
            WRITE('  ARCHIVO',' ':11,'HORA',' ':19,'ESTACION');
        END;
    END;
END;
'3': BEGIN
```

```
                GOTOXY(4,4);
                WRITE('  ARCHIVO',' ':13,'FECHA',' ':11,'HORA',' ':13,'ESTACION');
                FINDFIRST('D:\*.*',ARCHIVE,DIRINFO1);
                WHILE DOSERROR = 0 DO
                BEGIN
                    ARCH_EXTERNO:=' ';
                    ARCH_EXTERNO:=DIRINFO1.NAME;
                    NOMBRE_ARCH;
                    ARCH1_BUFFER;
                    OBTIENE_FECHA;
                    RESTA_30SEG;
                    NOMBRE_DE_MES;
                    CANAL_DISP:=MEM[$8000:$003C];
                    I:=TRUNC(CANAL_DISP/2)+1;
                    TABLA_ESTACIONES;
                    GOTOXY(4,J);
                    WRITE(ARCH_EXTERNO);
                    GOTOXY(21,J);
                    IF ((HORA=HORA_ANT) AND (MINUTO=MINUTO_ANT)) AND
                     ((SEGUNDO=SEGUNDO_ANT) AND (DECSEG=DECSEG_ANT)) THEN
                        WRITE ('     ARCHIVO VACIO.')
                    ELSE
                    BEGIN
                        NOMBRE_MES:=COPY(NOMBRE_MES,1,3);
                        IF DIA>9 THEN
                            WRITE(DIA,'/ ',NOMBRE_MES,'./ ',AHO)
                        ELSE
                            WRITE('0',DIA,'/ ',NOMBRE_MES,'./ ',AHO);
                        GOTOXY(39,J);
                        DESPLIEGA_FECHA;
                        GOTOXY(53,J);
                        WRITE(ESTACION[I]);
                    END;
                    HORA_ANT:=HORA;
                    MINUTO_ANT:=MINUTO;
                    SEGUNDO_ANT:=SEGUNDO;
                    DECSEG_ANT:=DECSEG;
                    J:=J+1;
                    FINDNEXT(DIRINFO1);
                    IF (J=23) AND (DOSERROR = 0) THEN
                    BEGIN
                        GOTOXY(47,23);
                        WRITE('PRESIONE <ENTER> PARA SEGUIR.');
                        OPCION2:=READKEY;
                        MARCO;
                        GOTOXY(27,2);
                        WRITE('DIRECTORIO DE DATOS: ');
                        J:=5;
                        GOTOXY(4,4);
                        WRITE('  ARCHIVO',' ':13,'FECHA',' ':11,'HORA',' ':13,'ESTACION');
                    END;
                END;
            END;
        ELSE
    END;
    GOTOXY(48,23);
    WRITE('PRESIONE <ENTER> PARA SALIR.');
    OPCION2:=READKEY;
    END;
    OPCION2:='3';
END;
```

RUTINA DE INTERRUPCION.

```
posicion macro a,b     ; Macro que posiciona el cursor en una
        push ax        ; determinada coordenada de la pantalla.
        push bx        ; Entran como parametros: a=renglon,
        push dx        ; b=columna.
        mov bx,0
        mov ah,02
        mov dh,a
        mov dl,b
        int 10h
        pop dx
        pop bx
        pop ax
        endm

 mensaje macro texto_x ; Macro que imprime en pantalla una
        push ax        ; cadena de caracteres previamente de-
        push dx        ; clarada en el data segment.
        push ds        ; Entra como parametro la variable de-
        mov dx,seg texto_x    ; clarada como string.
        mov ds,dx
        mov dx,offset texto_x
        mov ah,09
        int 21h
        pop ds
        pop dx
        pop ax
        endm



conviert macro digito,divisor ; Macro que descompone un numero en
        mov cx,ax      ; sus digitos colocandolos en las va-
        mov dx,0       ; riables digito0 - digito4.
        mov bx,divisor ; El numero a descomponer entra en el
        div bx         ; registro ax.
        mov digito,al
        mov ax,dx
        endm

   code segment
        assume cs:code, ds:code
inicio: push ax        ; Todos esto es para salvar el ambiente
        push bx        ; que encuentra antes de entrar a la interrupcion.
        push cx
        push dx
        push si
        push dl
        push ds
        push es
        push bp
        mov bp,sp
        mov ax,code    ; Pasa inicializa data segment.
        mov ds,ax
        call clrscr    ; Llama a subrutina que limpia pantalla.
        posicion 5,29  ; Posiciona cursor en renglon=5, columna=29.
        mensaje sismo  ; Coloca un mensaje  'S I S M O'.
        call sonido    ; Subrutina que envia un sonido de beep.
        posicion 8,14
        mensaje disparo; Envia un mensaje 'TIEMPO DE DISPARO: '.
        mov ah,2ch     ; Funcion de int 21 que recupera tiempo del
        int 21h        ; sistema operativo y lo pone en Regs.
        mov ax,0
        mov hora,ch    ; Recupera la hora del registro ch.
        mov minuto,cl  ; Recupera los minutos del registro cl.
        mov segundo,dh ; Recupera los segundos del registro dh.
        mov cien,dl    ; Recupera las centesimas de Seg. en Reg. dl.
        mov al,hora    ; Pasa hora al registro al para imprimir.
        mov modo,0     ; Variable que indica imprimir 2 digitos.
        call imprime   ; Subrutina de impresion de un numero
        mensaje puntos ; Envia mensaje ' : '.
        mov ax,0
        mov al,minuto
```

```
        call imprime   ; Subrutina que imprime en este caso los minutos.
        mensaje puntos ; Envia mensaje ' : '.
        mov ax,0
        mov al,segundo
        call imprime   ; Subrutina que imprime en este caso segundos.
        mensaje pun    ; Envia mensaje ' . '
        mov ax,0
        mov al,cien
        call imprime   ; Subrutina que imprime en este caso centesimas.
        call sonido    ; Subrutina que envia un sonido de beep.
        posicion 10,14 ; Posiciona cursor en renglon=10, Col.=14.
        mensaje fecha  ; Envia un mensaje 'FECHA DE HOY: '
        mov ah,2ah     ; Funcion que obtiene tiempo del Sist. Op.
        int 21h
        mov ax,0
        mov dia,dl     ; Recupera el dia del registro dl.
        mov mes,dh     ; Recupera el mes del registro dh.
        mov ano,cx     ; Recupera el ano del registro cx.
        mov modo,0     ; Variable para imprimir solo 2 digitos.
        mov al,dia
        call imprime   ; Subrutina que imprime lo que tenga al.
        mensaje de     ; Envia un mensaje ' de '
        xor ax,ax
        mov al,mes
        cmp al,01h
        jne cnt1
        jmp ene
cnt1:   cmp al,02h
        jne cnt2
        jmp feb
cnt2:   cmp al,03h
        jne cnt3
        jmp mar
cnt3:   cmp al,04h
        jne cnt4
        jmp abr
cnt4:   cmp al,05h
        jne cnt5
        jmp may
cnt5:   cmp al,06h
        jne cnt6
        jmp jun
cnt6:   cmp al,07h
        jne cnt7
        jmp jul
cnt7:   cmp al,08h
        jne cnt8
        jmp ago
cnt8:   cmp al,09h
        jne cnt9
        jmp sep
cnt9:   cmp al,0ah
        jne cnt10
        jmp oct
cnt10:  cmp al,0bh
        jne cnt11
        jmp nov
cnt11:  cmp al,0ch
        jne uno
        jmp dic
 ene:   mensaje enero
 uno:   jmp prosig
 feb:   mensaje febrero
        jmp prosig
 mar:   mensaje marzo
        jmp prosig
 abr:   mensaje abril
        jmp prosig
 may:   mensaje mayo
        jmp prosig
 jun:   mensaje junio
        jmp prosig
```

```
    jul: mensaje julio
         jmp prosig
    ago: mensaje agost
         jmp prosig
    sep: mensaje septi
         jmp prosig
    oct: mensaje octub
         jmp prosig
    nov: mensaje novie
         jmp prosig
    dic: mensaje dicie
prosig: mensaje de      ; Envia un mesaje ' de '.
        mov ano,cx
        mov ax,ano      ; Pasa a reg. ax el ano para imprimirlo.
        mov modo,1      ; Variable para que imprime 4 digitos.
        call imprime    ; Subrutina que imprime un numero.
        call sonido     ; Subrutina que genera un sonido de beep
        posicion 12,14 ; Posiciona el cursor en renglon=12, columna=14.
        mensaje genero ; Envia mensaje 'GENERO EL ARCHIVO: '.
        xor ax,ax
        mov al,mes
        mov modo,0      ; Variable para que imprima 2 digitos.
        call imprime    ; Forma el nombre del archivo con el mes,
        mov bl,digito1 ; Aqui va a dar nombre al archivo con el mes.
        or bl,30h
        mov si,2000h
        mov [si],bl
        mov bl,digito0
        or bl,30h
        mov si,2001h
        mov [si],bl     ; termina aqui de darle el mes al nombre.
        mov al,dia      ; el dia, la hora y los minutos.
        call imprime    ;          "
        mov bl,digito1 ; Aqui va a dar nombre al archivo con el dia.
        or bl,30h
        mov si,2000h
        mov [si],bl
        mov bl,digito0
        or bl,30h
        mov si,2001h
        mov [si],bl     ; termina aqui de darle el dia al nombre.
        mov al,hora     ;
        call imprime    ;
        mov bl,digito1 ; Aqui va a dar nombre al archivo con la hora.
        or bl,30h
        mov si,2000h
        mov [si],bl
        mov bl,digito0
        or bl,30h
        mov si,2001h
        mov [si],bl     ; termina aqui de darle la hora al nombre.
        mov al,minuto   ;
        call imprime    ;
        mov bl,digito1 ; Aqui va a dar nombre al archivo con el minuto.
        or bl,30h
        mov si,2000h
        mov [si],bl
        mov bl,digito0
        or bl,30h
        mov si,2001h
        mov [si],bl     ; termina aqui de darle el minuto al nombre.
        mensaje pun     ; coloca un mensaje de '. '
        xor ax,ax
        mov ax,ano      ; Aqui va a dar el ano al nombre
        call imprime    ;          "
        mov bl,digito1 ; Aqui va a dar nombre al archivo con el ano.
        or bl,30h
        mov si,2000h
        mov [si],bl
        mov bl,digito0
        or bl,30h
        mov si,2001h
```

```
        mov [si],bl     ; termina aqui de darle el ano al nombre.
        mensaje ext     ;                   "
        call sonido     ; Subrutina que envia un sonido de beep.
        push ds
        call d
        xor bx,bx
        mov si,0028h    ; Obtiene el canal que sobre paso el umbral
        mov bl,[si]     ; o en donde se detecto primero el sismo de la
        pop ds
        posicion 14,14 ; Posiciona el cursor en renglon=14 columna=14.
        cmp bl,00h      ; Loc. D000:0028h y compara para ver cual fue.
        jnz c2
        jmp canal1      ; Si bl=00 fue el canal 1
    c2: cmp bl,02h
        jnz c3
        jmp canal2      ; Si bl=02 fue el canal 2
    c3: cmp bl,04h
        jnz c4
        jmp canal3      ; Si bl=04 fue el canal 3
    c4: cmp bl,06h
        jnz c5
        jmp canal4      ; Si bl=06 fue el canal 4
    c5: cmp bl,08h
        jnz c6
        jmp canal5      ; Si bl=08 fue el canal 5
    c6: cmp bl,0ah
        jnz c7
        jmp canal6      ; Si bl=0a fue el canal 6
    c7: cmp bl,0ch
        jnz c8
        jmp canal7      ; Si bl=0c fue el canal 7
    c8: cmp bl,0eh
        jnz c9
        jmp canal8      ; Si bl=0e fue el canal 8
    c9: cmp bl,10h
        jz canal9       ; Si bl=10 fue el canal 9
        jmp c10
canal9: mensaje est9    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can9    ; Se coloca mensaje 'CANAL: 9'
        jmp c10
canal8: mensaje est8    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can8    ; Se coloca mensaje 'CANAL: 8'
        jmp c10
canal7: mensaje est7    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can7    ; Se coloca mensaje 'CANAL: 7'
        jmp c10
canal6: mensaje est6    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can6    ; Se coloca mensaje 'CANAL: 6'
        jmp c10
canal5: mensaje est5    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can5    ; Se coloca mensaje 'CANAL: 5'
        jmp c10
canal4: mensaje est4    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can4    ; Se coloca mensaje 'CANAL: 4'
        jmp c10
canal3: mensaje est3    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can3    ; Se coloca mensaje 'CANAL: 3'
        jmp c10
canal2: mensaje est2    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can2    ; Se coloca mensaje 'CANAL: 2'
        jmp c10
canal1: mensaje est1    ; Se coloca mensaje con el nombre de la estacion.
        posicion 16,14
        mensaje can1    ; Se coloca mensaje 'CANAL: 1'
```

```
  c10: call sonido
       push ds
       call d          ; Cambia ds por D000
       xor bx,bx
       mov si,0021H    ; Recupera apuntador de disparo.
       mov bx,[si]
       sub bx,04880H   ; Le resta 4880H y ve en que RAM inicio el evento.
       cmp bx,00
       jb menor        ; Inicio evento en la otra RAM de donde disparo.
       xor bx,bx
       mov si,0024H    ; Recupera RAM donde inicio el evento.
       mov bl,[si]
       cmp bl,0A0H     ; Disparo  e inicio el evento en la RAM1 ?.
       jz valor1       ; Disparo e inicio en RAM1.
valor2: pop ds
       mov modo,002H   ; Inicio en RAM2, esperara que PC apunte a RAM2.
       jmp contin
valor1: pop ds
       mov modo,001H   ; Inicio en RAM1, esperara que PC apunte a RAM1.
       jmp contin
 menor: xor bx,bx
       mov si,0024H    ; Recupera RAM donde incio el evento.
       mov bl,[si]
       cmp bl,0A0H     ; Disparo evento en RAM1, e inicio en RAM2 ?.
       jz valor2       ; Disparo en RAM1, pero inicio en RAM2.
       jmp valor1      ; Disparo en RAM2, pero inicio en RAM1.
contin: call polea
 etiq4: call cambia    ; cambia valor de la variable modo, espera otra RAM.
       mov bl,cien     ; Coloca el dato de centesimas de segundo
       push ds
       call d          ; Cambia ds con D000
       mov si,0070h    ; en la loc. D000:0070 de la zona de
       mov [si],bl     ; parametros.
       pop ds
       mov bl,segundo  ; Coloca el dato de segundos
       push ds
       call d          ; Cambia ds con D000
       mov si,0071h    ; en la Loc. D000:0071 de la zona de
       mov [si],bl     ; parametros.
       pop ds
       mov bl,minuto   ; Coloca el dato de minutos
       push ds
       call d          ; Cambia ds con D000
       mov si,0072h    ; en la Loc. D000:0072 de la zona de
       mov [si],bl     ; parametros.
       pop ds
       mov bl,hora     ; Coloca el dato de hora
       push ds
       call d          ; Cambia ds con D000
       mov si,0073h    ; en la Loc. D000:0073 de la zona de
       mov [si],bl     ; parametros.
       pop ds
       mov bl,dia      ; Coloca el dato de dia
       push ds
       call d          ; Cambia ds con D000
       mov si,0074h    ; en la Loc. D000:0074 de la zona de
       mov [si],bl     ; parametros.
       pop ds
       mov bl,mes      ; Coloca el dato de mes
       push ds
       call d          ; Cambia ds con D000
       mov si,0075h    ; en la Loc. D000:0075 de la zona de
       mov [si],bl     ; parametros.
       pop ds
       xor bx,bx       ; Coloca el ano en un registro de una
       mov bx,ano      ; palabra y lo pone en la Loc. D000:0078
       push ds
       call d          ; Cambia ds con D000
       mov si,0078h    ; de la zona de parametros.
       mov [si],bx
       pop ds
       call transf     ; Transfiere de D000:0 a buffer.
```

```
        push ds
        call d
        mov si,0006h    ; Recupera de 0000:000F la ram donde se
        mov bl,[si]     ; encuentra, para ver si sigue en la misma RAM
        pop ds
        cmp bl,modo     ; donde estaba antes de transferir RAMtarjeta al buffer
        jne cnt13
        jmp etiq4       ; si existe un cambio de RAM1-RAM2 o vice. Lo repite.
cnt13:  xor ax,ax
        mov ah,3ch      ; funcion para crear archivo ascii.
        xor cx,cx       ; limpia cx para archivo atributo normal.
        mov dx,seg archivo ; direccion del segmento del archivo.
        mov ds,dx       ; segmento en ds.
        mov dx,offset archivo ; offset del archivo.
        int 21h         ; interrupcion que crea archivo.
        jnc sigue
        jmp error
sigue:  mov fhandle,ax ; recupera el file handle que esta en ax.
        mov ah,3dh      ; funcion para abrir un archivo ascii.
        mov al,2        ; con atributo de lectura/escritura.
        mov dx,seg archivo
        mov ds,dx
        mov dx,offset archivo
        int 21h         ; interrupcion que abre archivo.
        jnc sigue1
        jmp error
sigue1: mov fhandle,ax ; recupera el file handle.
        call escri      ; Escribe 32K del buffer dentro del archivo (1 vez).
        jc error1
        cmp ax,32767
        je etiq3
error1: jmp error       ; si no es 32767 es que disco estaba lleno.
 etiq3: call polea      ; Espera que la otra RAM aparezca.
        call transf     ; Transfiere de 0000:0 a buffer.
        mov bl,modo
        call cambia     ; Cambia la variable modo.
        call escri      ; Escribe 32K del buffer dentro del archivo (2 vez).
        jc error2
        cmp ax,32767
        je etiq7        ; Si encuentra que el disco esta lleno.
error2: jmp error
 etiq7: push ds         ; Recupera el apuntador enviado en la localidad
        call d          ; 0042H, en el que se indica en que RAM finaliza
        xor bx,bx       ; el evento, si este apuntador tiene el valor de
        mov si,0042H    ; de A0H (RAM1) o de C0H (RAM2) es que el evento
        mov bl,[si]     ; ha finalizado y se prosigue a terminar la
        pop ds          ; rutina de interrupcion.
        cmp bl,0A0H
        je etii0
        cmp bl,0C0H
        je etii0
        jmp etiq3
 etii0: mov ah,3eh      ; cerrar un archivo.
        mov bx,fhandle
        int 21h         ; interrupcion para cerrar archivo.
        jc error
        jmp fin
 error: xor bx,bx
        mov ah,14h      ; hace beep.
        int 10h
        posicion 12,14
        mensaje falla
   fin: mov ah,31h      ; funcion para que un proceso quede residente en Mem.
        mov al,1        ; Codigo de regreso para el comando padre.
        mov dx,1000h    ; Parrafos reservados que es igual a 40 Kbytes.
        int 21h         ; Interrupcion que lo ejecuta
        mov dx,0021h    ; Coloca en dx puerto 21h
        in al,dx        ; Trae el contenido de este que contiene registro
        and al,0f7h     ; de las IRQ's no mascarables incrementa a IRQ3 como
        out dx,al       ; no mascarable y vuelve a enviarla al puerto (OCW1).
        mov al,063h     ; Envia por el puerto 20h la OCW2.
        mov dx,0020h
```

```
        out dx,al
        mov al,0bh      ; Envia por el puerto la OCW3.
        out dx,al
        mov sp,bp       ; Recupera el ambiente que encontro antes de entrar
        pop bp          ; a la interrupcion.
        pop es
        pop ds
        pop di
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
        iret            ; Regreso de la interrupcion.
;        mov ax,4c00h   ; Funcion de salida a MSDOS.
;        int 21h


      d proc near
        mov ax,1111h    ; Direccion donde encontramos los valores de RAM.
        mov ds,ax
        ret
      d endp

  cambia proc near      ; Cambia la variable modo.
        cmp bl,0d1h
        je etiq
        mov modo,0d1h   ; Se encuentra en RAM2 espera RAM1
        jmp etiq1
   etiq: mov modo,0d2h  ; Se encuentra en RAM1 espera RAM2
  etiq1: ret
  cambia endp

  transf proc near
        push ds
        cld             ; Bandera de direccion limpia para uso de movsb.
        call d          ; Esta direccion fuente es  D000.
        mov si,0000h    ; Offset de direccion fuente.
        mov ax,seg buffer; Direccion destino.
        mov es,ax       ; segmento de localidad dentro RAM de PC.
        mov di,offset buffer; offset de direccion destino.
        mov cx,32767    ; No. de bytes a escribir de fuente a destino.
     rep movsb          ; instruccion que realiza el cambio.
        pop ds
        ret
  transf endp

   polea proc near
  etiq2: push ds
        call d          ; Cambiar por d000
        mov si,000fh    ; D1 si estoy en RAM1 o D2 en RAM2, estoy
        mov bl,[si]     ; esperando el cambio a la otra RAM.
        pop ds
        cmp modo,bl
        jnz etiq2
        ret
   polea endp

   escri proc near
        xor ax,ax       ; repite para grabar la otra RAM .
        mov ah,40h      ; Empieza a graber la 2a. RAM con la fun. 40h
        mov bx,fhandle  ; Se le da el file handle
        mov cx,32767    ; numero de bytes a escribir
        mov dx,seg buffer ; direccion del buffer de memoria donde va a leer.
        mov ds,dx
        mov dx,offset buffer
        int 21h         ; Se ejecuta con la interrupcion 21 H.
        ret
   escri endp

  sonido proc near      ; Genera un sonido
        push ax
```

```
        push dx
        mov dl,7
        mov ah,2
        int 21h
        pop dx
        pop ax
        ret
 sonido endp

 clrscr proc near       ; Limpia la pantalla y escoge el modo video
        int 11h         ; con el que va a trabajar (monocromatico o
        and al,30h      ; a color).
        cmp al,30h
        mov ax,0b000h
        je seg_on
        mov ax,0b800h
seg_on: mov es,ax
        mov di,0h
        mov cx,4000
        mov ax,0720h
        rep stosw
        ret
 clrscr endp

hex_bcd proc near       ; Convierte un numero hexadecimal en bcd.
        push bx         ; El numero hexa debe entrar en el reg.
        push cx         ; AX, el numero en bcd sale en las varia-
        push dx         ; bles digito0 - digito3.
        conviert digito3,03e8h
        conviert digito2,0064h
        conviert digito1,000ah
        conviert digito0,0001h
        pop dx
        pop cx
        pop bx
        ret
hex_bcd endp

   envia proc near      ;procedimietno que imprime en pantalla
         or bl,30h      ;digito a digito una cifra numerica
         mov al,bl      ;Entra como parametro las varibles
         mov ah,14      ;digito 0 - digito 4.
         mov bx,0
         int 10h
         ret
   envia endp

imprime proc near       ; Imprime un numero en la pantalla
        push ax         ; este numero entra en el reg. AX.
        push bx
        xor bx,bx
        call hex_bcd
        cmp modo,0
        je salta
        mov bl,digito3
        call envia
        mov bl,digito2
        call envia
 salta: mov bl,digito1
        call envia
        mov bl,digito0
        call envia
        pop bx
        pop ax
        ret
imprime endp



archivo db 'c:\01234567.dat',0
fhandle dw 0
```

```
   fecha db 'FECHA DE HOY:   ','$'
disparo db 'HORA DE DISPARO:  ','$'
 genero db 'SE GENERO EL ARCHIVO:   ','$'
  sismo db ' S I S M O','$'
     de db ' de ','$'
 puntos db ':','$'
    pun db '.','$'
    ext db '1','$'
  enero db 'Enero','$'
febrero db 'Febrero','$'
  marzo db 'Marzo','$'
  abril db 'Abril','$'
   mayo db 'Mayo','$'
  junio db 'Junio','$'
  julio db 'Julio','$'
  agost db 'Agosto','$'
  septi db 'Septiembre','$'
  octub db 'Octubre','$'
  novie db 'Noviembre','$'
  dicie db 'Diciembre','$'
digito0 db ?
digito1 db ?
digito2 db ?
digito3 db ?
   modo db ?
    ano dw ?
    mes db ?
    dia db ?
   hora db ?
 minuto db ?
segundo db ?
   cien db ?
   est1 db 'ESTACION: Santa Rita, Edo. Mex.    DISTANCIA: 47.2 Km.','$'
   est2 db 'ESTACION: Cd. Serdan, Pue.         DISTANCIA: 188.0 Km.','$'
   est3 db 'ESTACION: Tonantzintla, Pue.       DISTANCIA: 86.0 Km.','$'
   est4 db 'ESTACION: Los Organos, Edo. Mex.   DISTANCIA: 48.0 Km.','$'
   est5 db 'ESTACION: Altzomoni, Pue.          DISTANCIA: 56.4 Km.','$'
   est6 db 'ESTACION: Nezontepetl, D.F.        DISTANCIA: 18.4 Km.','$'
   est7 db 'ESTACION: Tuxpan, Gro.             DISTANCIA: 101.6 Km.','$'
   est8 db 'ESTACION: Jocotitlan, Pue.         DISTANCIA: 172.8 Km.','$'
   est9 db 'ESTACION: Mesa Vibradora, C.U.     DISTANCIA: 7.2Km.','$'
   can1 db 'CANAL: 1','$'
   can2 db 'CANAL: 2','$'
   can3 db 'CANAL: 3','$'
   can4 db 'CANAL: 4','$'
   can5 db 'CANAL: 5','$'
   can6 db 'CANAL: 6','$'
   can7 db 'CANAL: 7','$'
   can8 db 'CANAL: 8','$'
   can9 db 'CANAL: 9','$'
  falla db 'ERROR: EL ARCHIVO NO PUDO GENERARSE.','$'
buffer db 32767 dup (?)


   code ends
        end inicio
```