15
24

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"PAQUETE PARA LECTURA Y DECODIFICACION DE ACELEROGRAMAS DIGITALES TIPO KINEMETRICS"

TESIS PROFESIONAL

INGENIERIA EN COMPUTACION

MARTHA ALICIA FUENTES MARILES

DIRECTOR DE TESIS

ING. ENRIQUE MENA SANDOVAL

México, D. F. 1988

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E PAGINA

PAQUETE PARA LECTURA Y DECODIFICACION

DE ACELEROGRAMAS DIGITALES TIPO

KINEMETRICS.

RESUMEN.

En diferentes zonas sísmicas como las costas de Guerrero,Oaxaca y Michoacán, el Instituto de Ingeniería opera acelerógrafos digitales marca KINEMETRICS, de los cuales es necesario obtener los acelerogramas para su posterior procesamiento. En el Instituto de Ingeniería de la UNAM se ha implementado un sistema de programas en torno a un equipo de cómputo PRIME P-550 MK-II para el proceso de acelerogramas que se han registrado en diversos tipos de acelerógrafos.

La información que registran los aparatos KINEMETRICS en cassettes se ha de recuperar en la computadora por medio de un paquete de lectura, que fue objeto de estudio.

Este paquete de lectura está formado principalmente de dos programas escritos en FORTRAN: Uno llamado DSP que controla el equipo de lectura de los cassettes digitales KINEMETRICS y captura la información contenida en los mismos, como son las señales de aceleración, tiempo y auxiliar, y el otro llamado PLOT que muestra una gráfica de las 3 componentes de la señal de aceleración, para lo cual requiere una decodificación de las señales contenidas en el archivo creado con el programa anterior.

Finalmente, se desarrolló el programa LINEAL para almacenar la información de manera tal que permitiera el uso de programas para adecuación de las señales, como parte previa al sistema de procesamiento TERRE.

1. ANTECEDENTES.

-Introducción.

Dada la actividad sísmica de nuestro país, se tienen instalados en zonas estratégicas diversos instrumentos de medición de movimientos del terreno, entre los cuales se encuentran los acelerógrafos. Mediante el uso de los acelerógrafos se obtienen registros de la aceleración del terreno en función del tiempo.

Existen acelerógrafos digitales y analógicos. Los acelerógrafos digitales registran las aceleraciones del terreno en cinta magnética digital tipo cassette, mientras los analógicos lo hacen en película fotográfica de 70 mm, cinta magnética o papel fotosensible .

Dentro de los acelerógrafos digitales se tienen del tipo DSA-1 y del PDR-1, de la marca KINEMETRICS.

El DSA-1 es un acelerógrafo triaxial que registra movimientos fuertes en una cinta magnética de 4 canales. Convierte las tres salidas analógicas de 3 acelerómetros a valores digitales proporcionales al movimiento del terreno. El instrumento se encuentra apagado hasta que un sismo origine que un disparador actúe la electrónica y la grabación de la cinta, lo cual ocurre en menos de 0.1 s. El acelerógrafo opera hasta después de 10 s.

(ajustable) de la última señal de disparo. Registra un sismo o varios de hasta 20 minutos, teniéndose al final de cada evento un espacio de 750 ms. Las señales analógicas de los acelerómetros están conectadas a un módulo de conversión analógica/digital y se multiplexan para formar una muestra. Cada muestreo y conversión se hace 200 veces por segundo. La salida del conversor son palabras de 12 bits.

Las salidas digitales quedan en 4 latches (3 para datos y 1 de paridad) que están conectados a los controladores de la cabeza de la cinta que se graba en 4 canales paralelos a una densidad de 1280 bpi. Contiene un acelerómetro para cada componente de la aceleración: longitudinal, transversal y vertical.

El PDR-1 es un acelerógrafo que graba eventos digitales en 3 canales, además del de paridad. A diferencia del anterior, la sección de entrada analógica cuenta con amplificadores de ganancia variable, haciéndose el muestreo a 100 o 200 veces por segundo. Se considera ideal para grabado digital de temblores locales o microtemblores.

Para obtener la información generada por los acelerógrafos digitales se requiere de un conjunto de programas de computadora que permita leer y almacenar los datos grabados; al recuperar los datos del cassette en una computadora se tiene la posibilidad de que la información contenida acerca de la aceleración del terreno sea procesada, permitiendo hacer un análisis más rápido con menos errores, y evitando el proceso de digitización a mano.

Es por ello el objeto de la presente tesis, que se ha dividido en varios capítulos: En el primer capítulo se hace una breve descripción del equipo de de lectura y de cómputo PRIME-550 que se tiene en el Instituto de Ingeniería para realizar el proceso de los diversos registros obtenidos de los acelerógrafos digitales.

En el capítulo 2 se mencionan los objetivos y descripción general del programa de computadora que hace posible la lectura de los cassettes digitales KINEMETRICS y su almacenamiento en la computadora PRIME.

En el capítulo 3 se hace una descripción de la sección de control de la lectora o unidad de reproducción de cassettes. Este control se lleva a cabo en forma externa, es decir, haciendo uso de comandos desde una terminal de computadora.

El formato de grabación del cassette se describe en el capitulo 4, así como la manera en que se realiza la decodificación de las señales de aceleración, tiempo y datos auxiliares (número de instrumento en que se grabó el cassette y la tasa de muestreo).

En el capítulo 5 se describe el almacenamiento del acelerograma en un archivo de datos en la computadora PRIME, y por último, en el capítulo 6 se presenta un sencillo manual de usuario que permita ejecutar los programas de lectura y/o graficación del evento grabado en los cassettes.

-Equipo de cómputo PRIME-550

Para hacer uso del paquete de lectura se cuenta con el equipo de cómputo PRIME modelo P-550 MK-II con las siguientes características:

Memoria principal	2 Mbytes
Memoria caché	2 Kbytes
Longitud de palabra	32 bits
Longitud de página	1024 palabras
Longitud de segmento	64 páginas
Acceso a memoria caché	80 nanosegundos
Unidad de disco	2 de 80 Mbytes c/u
Unidad de cinta	1 de 9 canales, 800/1600 bpi
Número de líneas	8 terminales y 4 de entrada de datos
Tipo de terminales	5 TEKTRONIX 4010 y 3 PRIME PT-45
Número máximo de líneas	15, con la configuración actual
Número máximo de tareas	128 simultáneas
Tamaño máximo de programa	32 Mbytes
Impresora	ATI-II
Graficador	VERSATEC, HARDCOPY
Sistema operativo	PRIMOS, Rev 18.3
Forma de trabajo	Tiempo compartido
Lenguajes	FORTRAN, FORTRAN-77, BASIC, CPL, PMA
Utilerías	MIDAS, DIPLOT, IRVING, DEBUGGER

En la configuración actual se tiene un procesador central con 2Mbytes de memoria principal a la que están conectados: un

controlador de disco que maneja dos unidades de disco de 80 Mbytes cada una, un controlador de cinta magnética para una unidad de cinta, una interfaz de propósito general para un graficador VERSATEC V-80, un controlador para el sistema al que están conectadas la consola del operador y la impresora, y un controlador de líneas asíncronas el cual maneja las líneas para las terminales y las líneas de propósito general (las líneas de envío o recepción de datos a otros dispositivos como la lectora de cassettes).

El diagrama de la configuración del sistema se muestra en la Figura 1.

El sistema operativo de PRIME se conoce como PRIMOS (Prime's Operating System), está escrito en PMA (Prime Macro Assembler) y en FORTRAN , conteniendo el código que maneja el acceso al sistema, el control de las entradas/salidas, el acceso interactivo por terminal, las tareas no interactivas, el sistema de archivos, el sistema de comunicaciones, los lenguajes y las utilerías.

De manera general, las "rutinas" del sistema, accesibles al usuario, se clasifican en cuatro:

1) Subrutinas del sistema operativo.- Aquellas que permiten interactuar con el equipo: editar, compilar y correr un programa, además de subrutinas para el manejo de errores y entradas/salidas a disco y cinta.

2) Subrutinas de aplicaciones y matrices para FORTRAN estándar.-

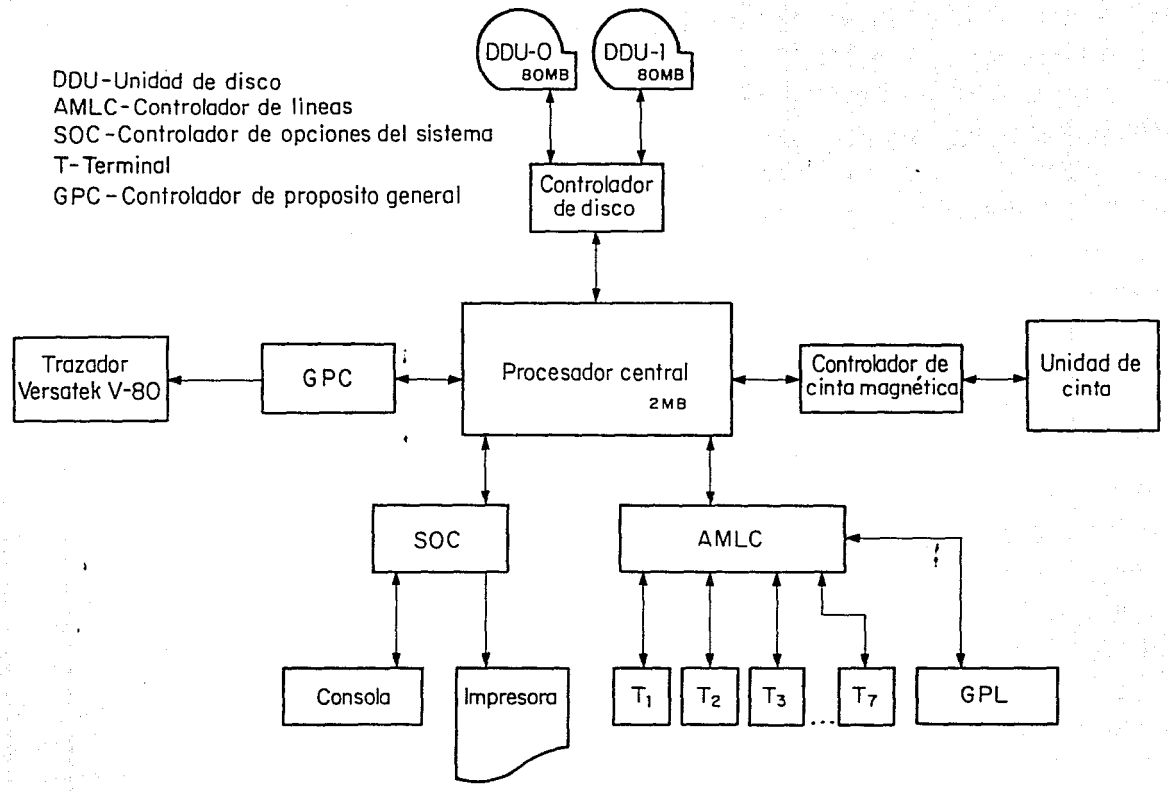


Fig 1 Configuración del sistema PRIME 550 MK-II

Contienen las funciones ANSI-estándar para FORTRAN, funciones lógicas, rutinas para operaciones aritméticas y para operaciones con matrices, rutinas de servicio y para ordenamientos.

3) Subrutinas de entrada/salida.- Rutinas básicas que realizan entrada/salida entre la computadora PRIME y los discos, terminales y dispositivos periféricos conectados al sistema. Estas subrutinas forman un grupo llamado IOCS (Input/Output Control System).

4) Subrutinas de comunicación y tiempo real.- Cuentan con controladores síncronos y asíncronos para el movimiento de datos de líneas de comunicación, y también con semáforos y temporizadores para sincronizar la ejecución de algunos programas con otros usuarios. Dentro de estas subrutinas, se tiene la T$AMLC que es la que controla la transferencia de información entre el DSP-3 o lectora de cassettes y la línea asignada AMLC (Asynchronous Multiline Controller).

Las unidades de disco están configuradas en 4 discos lógicos:

M183A1 Reside el sistema operativo y paginación (partición 460 y 20061 respectivamente).

TRAB1 Disco de trabajo (partición 10460).

TRAB2 Disco de trabajo (partición 462).

TRAB3 Disco de trabajo (partición 10463).

Los archivos PRIMOS en disco tienen una estructura de árbol, donde el MFD (Master File Directory) es la raíz o tronco del árbol,

y los nodos son los archivos y los directorios UFD (User File Directory). (Fig. 2).

El sistema puede manejar archivos de los siguientes tipos:

SAM	Archivo de acceso secuencial.
DAM	Archivo de acceso directo.
SEGSAM	Archivo segmentado de acceso secuencial.
SEGDAM	Archivo segmentado de acceso directo.
UFD	Directorio o Subdirectorio.

-Equipo de lectura

El DSP-3 (DIGITAL PLAYBACK SYSTEM) es una interfaz entre una computadora y cassettes digitales DSA-1 y PDR-1 que se usa para leerlos o reproducirlos. Está basado en un microprocesador con una memoria de 64Kbytes que permite reproducir los cassettes de manera consistente a la velocidad de transmisión.

El DSP-3 está diseñado para transferir los datos grabados en estos cassettes a una computadora, a través de una interfaz serie RS-232C, y acepta también instrucciones desde la computadora para controlar su operación.

Tiene 4 salidas analógicas para enviarse a un dispositivo de despliegue o graficación, y dos digitales (una serie RS-232C y una

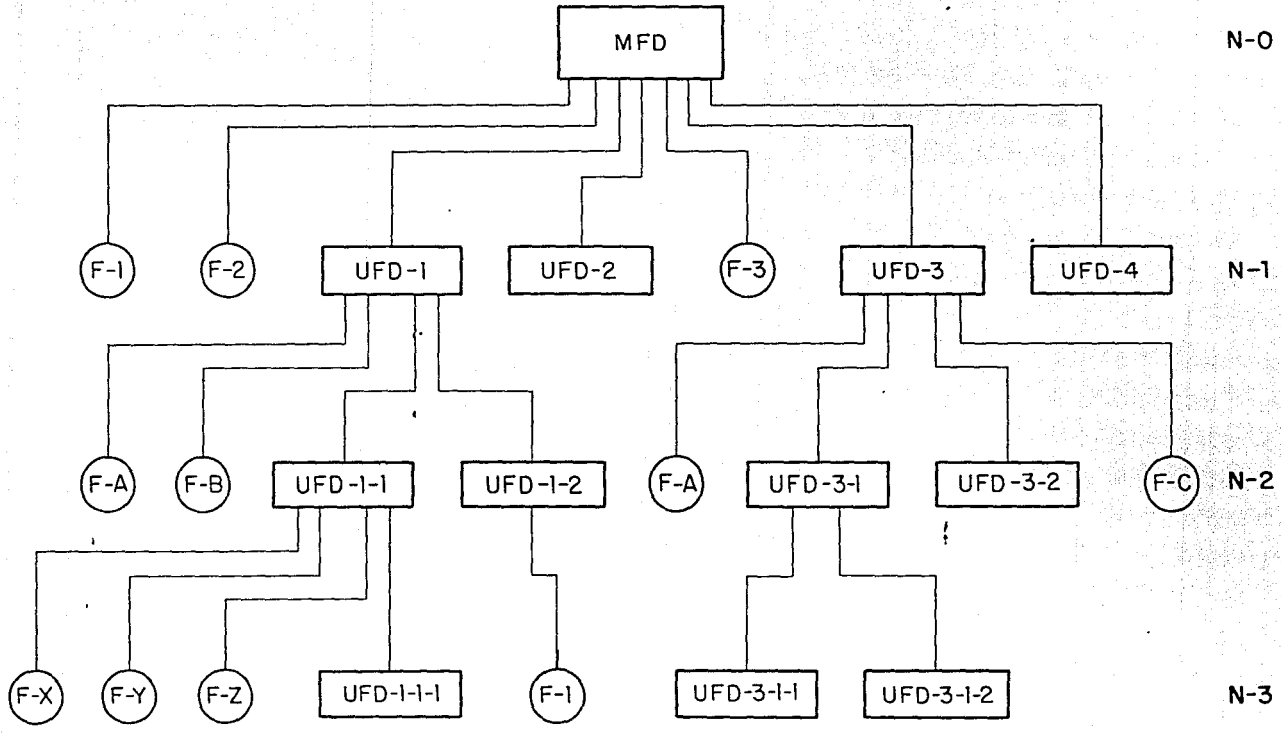


Fig 2 Estructura de archivos PRIMOS

paralela de 8 bits). La velocidad de transmisión se puede seleccionar entre 150 y 9600 bauds, así como el formato de salida binario o ASCII. El tablero del DSP-3 se muestra en la Fig. 3.

Hay dos modos de control para el DSP-3: manual o externo, que se selecciona con un interruptor. En el control manual se activan los botones para el manejo y salida de los datos, ignorándose cualquier comando desde la computadora para el control del cassette. En el control externo, las acciones se ejecutan bajo un comando de la computadora, deshabilitándose los controles manuales a excepción de STOP-TAPE. (Fig.3).

Al frente tiene luces indicadoras del estado y ubicación de los datos, así como indicadores numéricos que muestran el tiempo de reproducción y el número del evento leído.

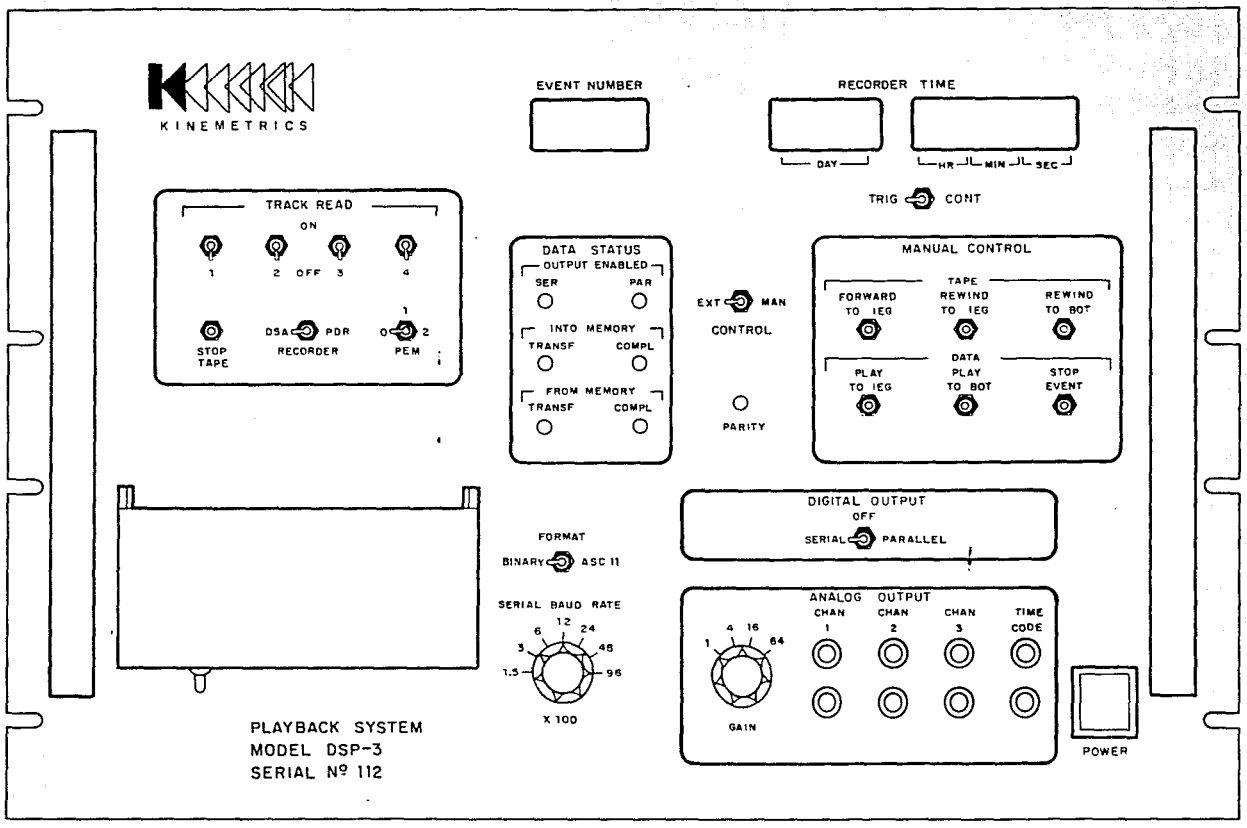


Fig 3 Tablero del DSP-3

2. OBJETIVOS Y DESCRIPCION GENERAL DEL PAQUETE.

-Objetivos

El paquete de lectura permite, por medio de una terminal de la computadora PRIME:

-Controlar el equipo de lectura de cassettes digitales con formato KINEMETRICS.

El programa de interfaz llamado #DSP permite enviar comandos desde la terminal para el control del DSP-3 (Digital Playback System).

-Capturar la información contenida en los cassettes.

El comando P (Play to Ieg) del programa de interfaz, solicita el nombre de un archivo donde quedará almacenada la información.

-Decodificar las señales de aceleración y tiempo.

Se tienen en un arreglo valores entre 0 y 4095 correspondientes a la aceleración para cada canal.

-Almacenar en la computadora la información del cassette.

Se crea un archivo donde guarda la información completa del cassette: el número de serie del instrumento donde fue grabado, la tasa de muestreo de las señales, los valores de

aceleración en sus tres componentes, además de la fecha y hora en que se registró el evento.

-Graficar la señal de aceleración en sus tres componentes (longitudinal, vertical y transversal).

-Descripción General

El paquete de lectura para acelerógrafos digitales KINEMETRICS es un conjunto de programas, incluyendo los archivos de comandos necesarios para su instalación y operación, para alcanzar los objetivos mencionados. Consta de dos programas principales: la interfaz entre la reproductora y la computadora (DSP) y el de graficación de datos (TEKPLOT). De manera general, el programa de interfaz, emplea una línea asignable para transferir información del cassette a la computadora PRIME. Esta transferencia se lleva a cabo empleando un buffer que regula la recepción de los datos y que constantemente se prueba para verificar si está listo para recibir más, si se ha llenado o si está vacío, mandando las señales de control correspondientes. El programa de graficación de datos solicita al usuario el nombre del archivo que desea graficar, y dibuja punto por punto, dentro de un marco que se traza previamente, en la parte superior la señal del canal 1, en la parte media la señal del canal 2 y debajo de ésta la señal del canal 3; la señal de reloj Omega se muestra en la parte más baja del marco de graficación.

A.INSTALACION.

Para la instalación del paquete DSP en el sistema PRIME es necesario que se carguen en un directorio (UFD) los siguientes archivos: DSP.FTN, DOSSUB.PMA y C_DSP.SEG. Para que quede instalado el programa de interfaz debe ejecutarse el archivo C_DSP.SEG. Estos archivos forman el programa de interfaz (#DSP) que al ejecutarse asigna una línea para la lectora (DSP-3), desplegando un menú de comandos para el manejo del cassette (rebobinarlo al inicio, reproducir un evento, reproducir varios eventos hasta el final del cassette, detener el proceso de reproducción). Los comandos de lectura y envío de datos al programa solicitan al usuario un nombre de archivo que contendrá los datos leídos.

Para instalar el programa que permita la graficación de datos en una terminal TEKTRONIX, es necesario cargar los siguientes archivos: TEK.FTN, PLOTDEFS, FILDEF y C_TEK.SEG. Estos archivos forman el programa de graficación (#PLOT), que solicita al usuario el nombre del archivo de datos que se va a graficar. Para que quede instalado el programa de graficación debe ejecutarse el archivo C_PLOT.SEG.

B.OPERACION.

Para ejecutar el programa de interfaz se deben realizar las siguientes tareas en el DSP-3:

-Conectar la línea física de transferencia, en la parte posterior de la lectora (conector SERIAL)

-Encender el equipo de lectura DSP-3

-Introducir el cassette a reproducir o leer

-Colocar los interruptores en las siguientes posiciones:

control EXT, digital output SERIAL, format BINARY, serial baud rate 96, recorder DSA o PDR (según sea el caso)

Hecho todo lo anterior, encender la terminal, entrar al sistema y al directorio donde se ha instalado el paquete de lectura y teclear SEG #DSP.

El programa pregunta el número de línea de comunicación asignada al DSP-3 o lectora, siendo usualmente la número 12; al quedar establecida la comunicación, se muestra al usuario un menú de instrucciones para el control del cassette. Cuando el usuario desea leer un evento, teclea al comando P (Play to leg), entonces el programa pide un nombre de archivo para almacenar la información. Este archivo se crea en un subdirectorio llamado .RAWDATA.

Para ejecutar el programa de graficación de datos se debe teclear SEG #PLOT. Al ejecutar dicho programa se obtiene una gráfica del archivo leído, la cual se escala automáticamente, desplegando las 3 trazas correspondientes a las componentes longitudinal (canal 1), vertical (canal 2) y transversal (canal 3), además de los pulsos de la señal de tiempo o reloj. La gráfica obtenida es de aceleración (en volts) contra tiempo (en segundos), la cual debera escalarse de acuerdo con la sensibilidad real de cada sensor (gals/volt).

-Alcances y limitaciones.

Es necesario asignar una línea de comunicación, si no es aceptada se termina el programa.

El DSP-3 debe enviar un caracter de reconocimiento antes de iniciar la transferencia de datos. Se tiene un tiempo de espera para ello, después del cual si no se envía se termina el programa.

Si se teclea un comando que NO existe en el menú, se despliega un mensaje de inválido y regresa al menú. Se presenta de nuevo el menú cuando se ha llevado a cabo algún comando.

Se verifica la existencia del subdirectorio .RAWDATA antes de preguntar por un nombre de archivo, si no existe lo crea una y solo una vez.

Abre un archivo para cada evento. Si se da un nombre de archivo igual a uno existente, pregunta si almacena los datos en ese mismo; si no pregunta por un nuevo nombre de archivo. Si no se puede abrir el archivo se pregunta por un nuevo nombre.

La lectura del evento se hace ventana por ventana (64 muestras en la que se incluye la sincronía). Escribe en el archivo cada 256 palabras de 16 bits (64 muestras X 4 canales).

Antes de desasignar la línea de comunicación o interrumpir la lectura de un evento se cierra el archivo que se ha abierto.

No es posible interrumpir con CTRL-P la ejecución del programa de lectura. (CTRL-P significa BREAK en PRIME).

En la lectura continua (comando A), después de cada evento se hace una pausa de 2.5 s. y se agrega un número al nombre base del archivo, para distinguir cada evento.

3. CONTROL DE LA UNIDAD DE REPRODUCCION.

-Descripción de la sección de control.

Existen dos categorias de instrucciones que pueden ser enviadas al DSP-3 desde la computadora:

Instrucciones de Control de Datos e Instrucciones de Control de Cinta (o cassette).

SECCION DE CONTROL

INSTRUCCIONES DE CONTROL DE DATOS	INSTRUCCIONES DE CONTROL DE CINTA
CTRL-Q	S STOP TAPE
CTRL-S	F FORWARD TO IEG
	R REWIND TO IEG
	B REWIND TO BOT
	P PLAY TO IEG
	A PLAY TO EOT
	E STOP EVENT

CTRL-Q	Dice al DSP-3 que la computadora está lista para recibir datos. El DSP-3 enviará datos hasta el fin de un evento o hasta que reciba una instrucción CTRL-S.
CTRL-S	Dice al DSP-3 que la computadora NO está lista para recibir datos. El DSP-3 detendrá el envío de datos, hasta que reciba la instrucción CTRL-Q.
S (STOP TAPE)	Interruptor que detiene la cinta en forma inmediata. Detiene la salida de datos, restablece los circuitos de control y limpia la memoria.
F (FORWARD TO IEG)	Avanza la cinta hasta el próximo espacio entre eventos (inter-event gap).
R (REWIND TO IEG)	Regresa la cinta al principio del evento inmediato anterior.
B (REWIND TO BOT)	Regresa la cinta hasta su inicio.(Begin of tape).
P (PLAY TO IEG)	Inicia lectura de datos, que se almacenan en la memoria de la lectora para después transmitirse a través de la salida serie. La entrada de datos a memoria del DSP-3 es

más rápida que su salida, por lo que después de un tiempo la memoria llega a llenarse. Cuando esto ocurre, la cinta se detiene y regresa al inicio del evento. Cuando la memoria se ha vaciado, la cinta avanza automáticamente hasta donde localice el punto en que se detuvo, y entonces continúa introduciendo datos a memoria. Esto se repite las veces necesarias hasta completar la lectura de un evento.

A (PLAY TO EOT) Avanza la cinta y se leen los datos. Al encontrar el fin de un evento se detiene la cinta brevemente y automáticamente lee el siguiente evento. Este procedimiento continúa hasta llegar al fin de la cinta.

E (STOP EVENT) Detiene la lectura de un evento. La cinta avanza al inicio del próximo evento y se detiene. Los datos listos en memoria se transfieren, pero no se introducen nuevos datos.

El DSP-3 no iniciará el envío de datos a la computadora hasta que reciba un caracter CTRL-Q de la misma; este carácter puede recibirse después de que el DSP-3 ha iniciado la lectura o

reproducción de un evento. Cuando el interruptor de control está en EXTernal, el DSP-3 responderá a las instrucciones de control enviadas desde la computadora y NO a los botones del tablero, excepto el botón de STOP-TAPE que siempre funciona.

La comunicación se lleva a cabo mediante una interfaz serie que transfiere caracteres de 8 bits entre el DSP-3 y la computadora. Además de los 8 bits de datos se envía un bit alto para indicar el inicio del dato y 2 bits bajos de stop:

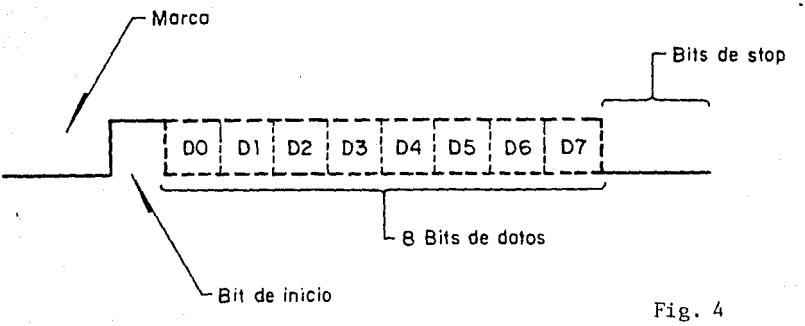


Fig. 4

La conexión eléctrica para la interfaz serie, en la configuración estándar, necesita 2 líneas para la señal (RXD y TXD) y una línea de tierra (Fig. 5).

Conector serie en la parte posterior del DSP-3 (tipo DB-25):

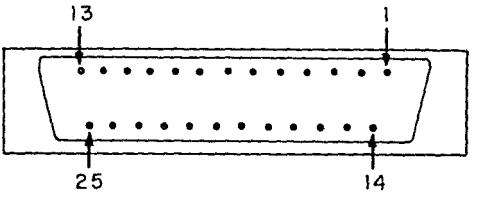


Fig. 5

No. PIN	DESCRIPCION	DIRECCION DE LA SEÑAL
2	TXD	Entrada al DSP-3

3	RXD	Salida
7	GRD	Tierra

-Posicionamiento del cassette

Para colocar el cassette digital en el DSP-3, se ejecutan los siguientes pasos:

-Levantar la cubierta de plástico.

-Insertar la parte superior del cassette en la parte alta del espacio del transporte.

-Bajar el expulsor y acomodar el cassette en el riel.

-Soltar el expulsor y empujar el cassette.

-Bajar la cubierta de plástico.

Para retirar el cassette digital del DSP-3, se ejecutan los siguientes pasos:

-Levantar la cubierta de plástico.

-Empujar hacia abajo el expulsor del cassette.

-Retirar el cassette.

-Soltar el expulsor.

-Bajar la cubierta de plástico.

-Transferencia de datos

La comunicación entre computadora y DSP-3, incluyendo la

transferencia de datos, se lleva a cabo por medio de la rutina T$AMLC contenida en el sistema PRIME.

Dicha rutina realiza el movimiento de datos con el uso del buffer de la línea AMLC que ha sido asignada al DSP-3 o unidad de reproducción, además nos permite conocer el estado de la línea en cuanto al buffer de entrada/salida.

Sintaxis de la llamada a la rutina T$AMLC:

```
CALL T$AMLC (line, user-buf-addr, char-count, key, stat-vec,
             char-pos-arg, errcode)
```

line — Número de la línea AMLC.

user-buf-addr — Dirección del buffer que llama (arreglo de datos).

char-count — Número de caracteres a mover.

key — Funciones:

1 Introduce caracteres al user-buf-addr.

2 Introduce char-count caracteres o hasta que encuentre .NL.

3 Saca caracteres.

4 Número de caracteres en el buffer de entrada (stat-vec(1)).

6 Introduce todos los caracteres posibles en

el buffer de entrada.

8 Limpia el buffer de entrada.

10 Limpia el buffer de entrada y el de salida.

stat-vec — Dos palabras que forman el vector de estado de los datos del buffer.

char-pos-arg — Posición inicial dentro del buffer direccionado por user-buf-addr.
Si se omite este argumento, por default la posición inicial es el 1er. carácter. No se actualiza dentro de T$AMLC.

errcode — Argumento opcional que nos regresa un código de error, sin que se impriman los mensajes de error en la terminal. Si no hay error, errcode es igual a 0 (cero).

-Subprogramas.

El control de la unidad de reproducción por medio de una terminal en el sistema PRIME se lleva a cabo al correr el programa #DSP correspondiente al programa fuente DSP.FTN, que hace uso de diversas rutinas propias del sistema, además de las siguientes subrutinas:

ASAMLC	Asigna un número de línea AMLC (RS-232 serie).
FLUSH	Limpia el buffer de la línea AMLC.
WAITPR	Espera un tiempo definido a que se reciba un carácter en el buffer de la línea AMLC.
GETUFD	Verifica la existencia del subdirectorio .RAWDATA en el actual directorio. En caso negativo, lo crea.
MAKNAM	Genera un archivo dentro del subdirectorio .RAWDATA, de acuerdo con el nombre seleccionado por el usuario.
GETEVE	Lee del DSP-3 un evento y lo salva en un archivo.
UNAMLC	Desasigna la línea AMLC, cierra archivos y sale al sistema.
READIN	Lee una ventana de datos (64 muestras).

Rutinas propias del sistema:

BREAK$	Inhibe o habilita CTRL-P (interrumpe un programa).
CLOS$A	Cierra un archivo.
CNIN$	Lee de la terminal un número específico de caracteres.
CNSIG$	Indica el estado de cierta condición que se ha programado.
CREA$$	Crea un subdirectorio.
DOSSUB	Resuelve referencias a direcciones entre programa principal y subrutinas externas.
DUPLX$	Controla la manera en que el sistema operativo tratará a la terminal (full o half duplex).

MKON$F	Crea una unidad nueva para manejar cierta condición.
PRWF$$	Lee/escribe/posiciona/trunca algún archivo.
SLEEP$	Suspende durante un tiempo definido la ejecución de un proceso.
SRCH$$	Abre/cierra/borra/prueba la existencia de algún archivo.
TIOCT	Lee un número octal desde terminal.
TNOU	Imprime en la terminal ciertos caracteres y hace un cambio de renglón.
TNOUA	Imprime en la terminal ciertos caracteres, sin cambiar renglón.
TODEC	Imprime en la terminal un número decimal, sin cambiar renglón .
TOHEX	Imprime en la terminal un número hexadecimal, sin cambiar renglón .
TONL	Hace un cambio de renglón.
TOOCT	Imprime en la terminal un número octal sin cambiar renglón.
TRNC$A	Trunca un archivo.
T$AMLC	Rutina para el control asíncrono de movimiento de datos.

El programa completo se presenta en el apéndice A1 y en la Fig. 6 se muestra un diagrama simplificado de #DSP.

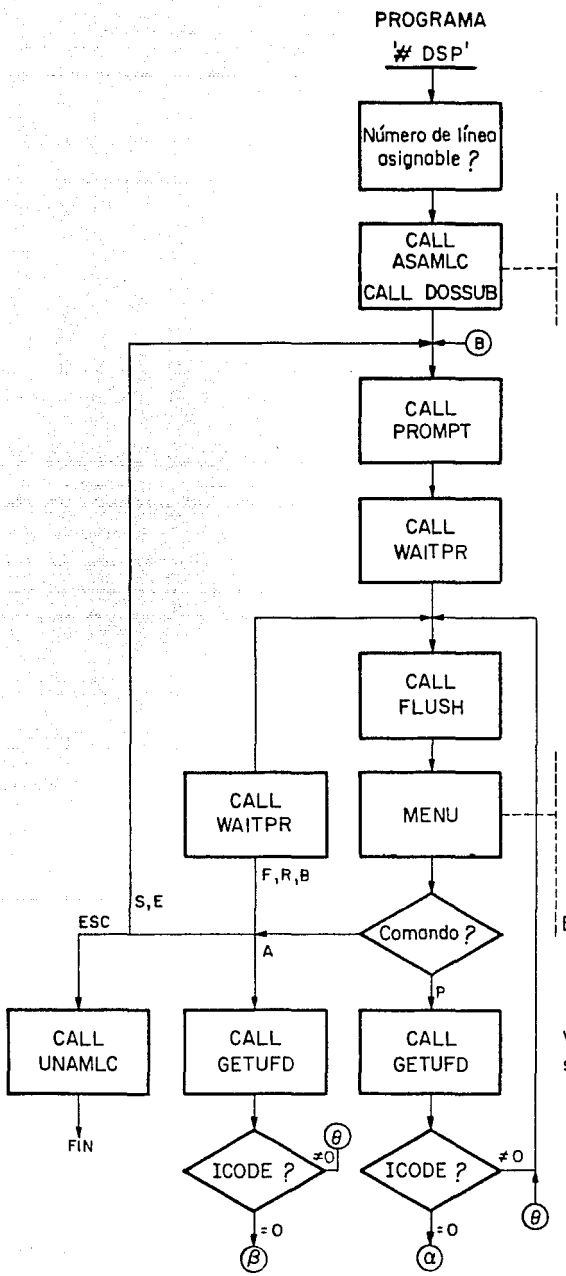
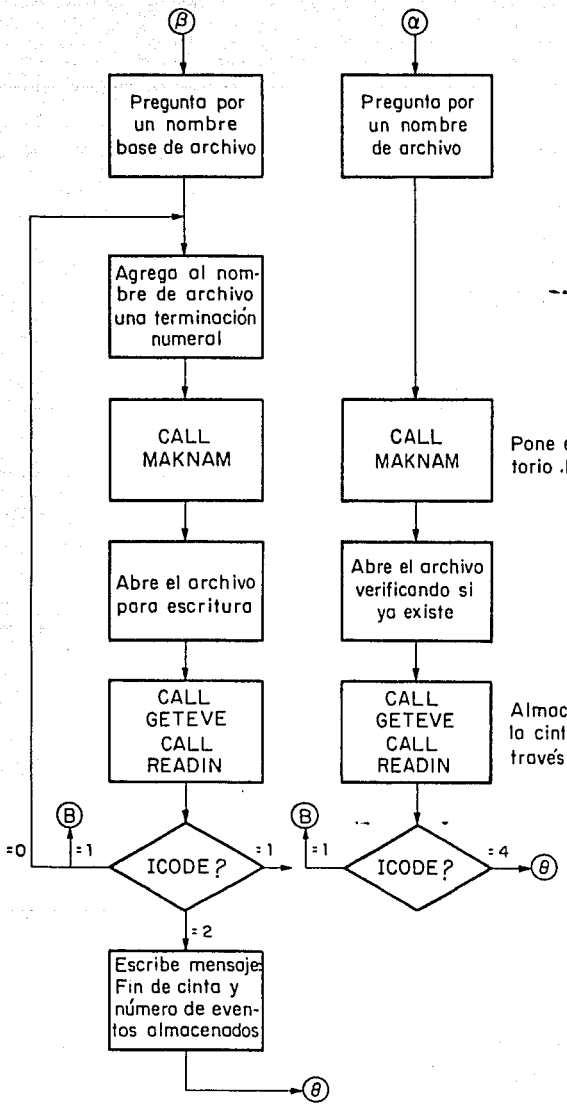

PROGRAMA
'# DSP'
Número de línea asignable ?
CALL ASAMLC
CALL DOSSUB
Intenta asignar una línea para comunicar el DSP-3 con la computadora.
Envía un mensaje del número de línea que se asignó y si no fue posible sale del programa
B
CALL PROMPT
Envía un caracter de inicio al DSP-3.
CALL WAITPR
Espera un caracter de respuesta del DSP-3.
CALL FLUSH
Limpia los buffers de entrada/salida.
CALL WAITPR
MENU
S = Stop
F = Forward to IEG
R = Rewind to IEG
B = Rewind to beginning
P = Play to IEG
A = Play to EOT
E = Stop event
ESC = Unassign seriol line and quit program
F,R,B
S,E
ESC
Comando ?
A
P
CALL UNAMLC
CALL GETUFD
CALL GETUFD
Verifica la existencia del subdirectorio.RAWDATA, si no existe lo crea en el actual directorio
FIN
ICODE ?
≠0
θ
=0
β
ICODE ?
≠0
=0
α
θ

Fig 6


β
α
Pregunta por un nombre base de archivo
Pregunto por un nombre de archivo
Agrega al nombre de archivo una terminación numeral
CALL MAKNAM
CALL MAKNAM
Pone el archivo dentro de subdirectorio .RAWDATA
Abre el archivo para escritura
Abre el archivo verificando si ya existe
CALL GETEVE CALL READIN
CALL GETEVE CALL READIN
Almacena en un archivo un evento, leyendo la cinta con la subrutina READIN a través de un buffer
B
B
=0
=1
ICODE ?
=1
=1
ICODE ?
=4
θ
=2
Escribe mensaje: Fin de cinta y número de eventos almacenados
θ

4. DECODIFICACION DE LAS SEÑALES DE ACELERACION, TIEMPO Y AUXILIAR.

-Descripción del formato de grabación del cassette.

El formato de los datos que envía el DSP-3 a la computadora y su secuencia se muestra en la Figura 7. Como la muestra de 16 bits, es muy grande para transferirse en una sola vez a través de la interfaz serie de 8 bits, se separa en dos segmentos como se muestra en la Figura 8.

Los datos que contienen los cassettes fueron grabados en muestras de 16 bits para cada uno de los cuatro canales: canal 1 (longitudinal), canal 2 (vertical), canal 3 (transversal) y canal 4 de paridad (impar). Cada 64 muestras se graban ceros en todos los canales para sincronía (Frame Sync), por lo que en el momento de graficación estos ceros se sustituyen al repetir la muestra anterior.

El formato de grabación de los datos (muestras de 16 bits) depende del tipo de acelerógrafo, el cual se describe a continuación:

-Para los cassettes provenientes de aparatos DSA-1:

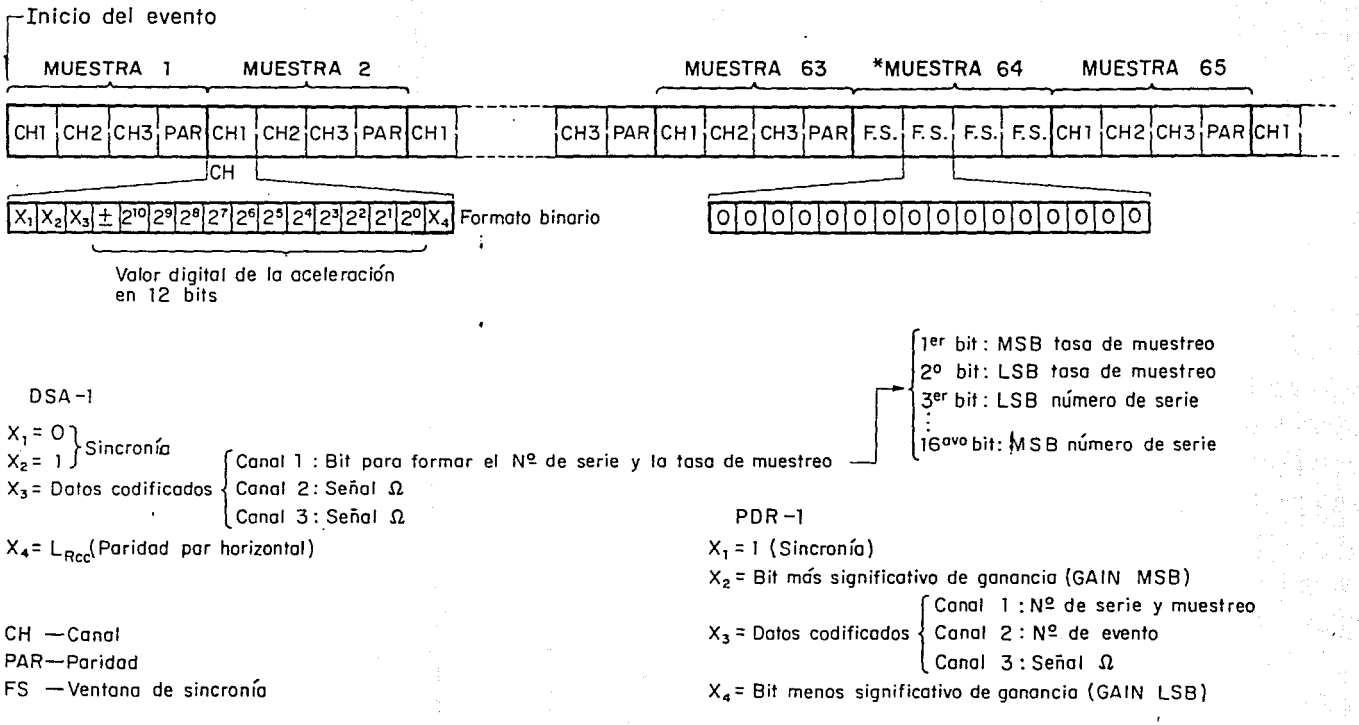


Fig 7 Secuencia de datos

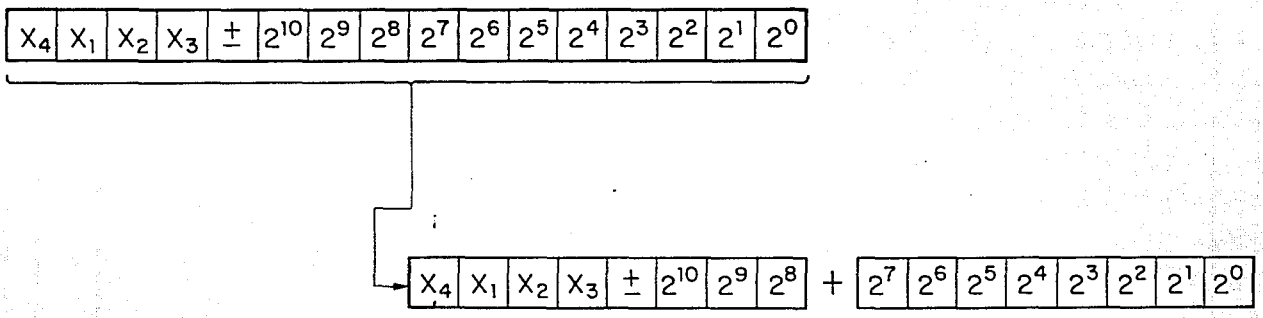


Fig 8 Formato segmentado en 2 Bytes

2 bits de sincronía "01" en todos los canales.

1 bit para datos codificados según el canal:

Canal 1: a) Tasa de Muestreo formada con las 2 primeras muestras después de la ventana de sincronía (Frame Sync). Se repite cada 64 muestras.

Bit MSB	Bit LSB	Tasa de muestreo (muestras/seg)
0	0	50
0	1	100
1	0	200
1	1	400

b) Número de serie del instrumento, para lo cual requiere a lo más 14 muestras a partir de la 3ra. muestra. (Se repite cada 64 muestras).

Canal 2: Reloj interno de 2 Hz.

Canal 3: Para un código de tiempo opcional, usualmente es un reloj Omega (internacional).

12 bits correspondientes a los valores de aceleración:

VOLTS	BINARIO	HEXADECIMAL	ENTERO
+2.500	0000 0000 0000	000	0
+1.250	0100 0000 0000	400	1024
0.000	1000 0000 0000	800	2048
-1.250	1100 0000 0000	C00	3072
-2.498	1111 1111 1111	FFF	4095

1 bit de LRCC (Longitudinal Redundancy Check Character) para provocar un número par de bits "1" en cada palabra de muestra.

-Para los cassettes provenientes de aparatos PDR-1:

1 bit "1" de sincronía.

2 bits que involucran la ganancia del amplificador para cada canal:

GMSB	GLSB	Ganancia
0	0	1
0	1	4
1	0	16
1	1	64

1 bit para datos codificados:

Canal 1: a) Tasa de muestreo formada con las 2 primeras muestras después de cada ventana de sincronía (Frame Sync). Se repite cada 64 muestras.

Bit MSB	Bit LSB	Tasa de muestreo (muestreo/seg).
0	1	100
1	0	200

b) Número de serie del instrumento (14 muestras a partir de la 3ra. muestra). Se repite cada 64 muestras.

Canal 2: Número de evento.

Canal 3: Señal de reloj opcional (Reloj omega).

12 bits que contienen los valores de aceleración:

VOLTS	BINARIO	HEXADECIMAL	ENTERO
+2.500	0000 0000 0000	000	0
+1.250	0100 0000 0000	400	1024

0.000	1000 0000 0000	800	2048
-1.250	1100 0000 0000	C00	3072
-2.498	1111 1111 1111	FFF	4095

-Decodificación de las señales de aceleración

Los valores de aceleración, como se mencionó en el formato de grabación del cassette, están formados con 12 bits. En la subrutina READIN del programa TEKPLOT se lee cada muestra para cada canal, se toman los 12 bits que corresponden al valor de aceleración (se hace un corrimiento a la derecha a las muestras de los 3 canales), después se convierten en un número real cuyo valor se guarda en un arreglo de dos dimensiones en donde el ler. índice es el número de la muestra y el segundo índice es el número del canal. Como debe recordarse, la muestra 64 está formada con ceros (para sincronía), lo cual no es un valor de aceleración, por lo que se repite el valor de la muestra anterior con el fin de que la gráfica se reproduzca con los valores de aceleración para todas las muestras.

Para obtener la gráfica de la señal de aceleración para cada canal, se despliega punto a punto el arreglo de valores de aceleración desde la la. muestra hasta el total de muestras leídas, cuyo valor determina el escalamiento horizontal con sus respectivas marcas (en segundos), además de adecuar en la escala vertical las señales de aceleración conforme a los valores que contiene el arreglo.

-Decodificación de la señal de tiempo.

La señal de tiempo que se graba en los cassettes KINEMETRICS es una señal internacional conocida como reloj Omega que contiene fecha (número de día) y hora (hora, minutos y segundos), de manera que permite conocer el tiempo en que ocurrió un evento registrado en el cassette.

Esta señal de tiempo está codificada en BCD (Binary Code Decimal), consta de 50 pulsos que incluyen el inicio/fin de la señal, segundos, minutos, horas, día y número de serie del aparato que envió la señal al acelerógrafo. Cada inicio y fin de la señal implica que transcurrieron 10 segundos, que es la duración de una señal de reloj completa.

Para decodificar la señal de reloj se localiza en la gráfica de la señal omega un pulso ancho de inicio (del tamaño de 2 pulsos) a partir del cual se agrupan los siguientes pulsos de 4 en 4 de acuerdo al flanco de bajada: en los primeros 8 pulsos se tienen las unidades y decenas que conforman los segundos, en los siguientes 8 se tienen las unidades y decenas para el número de minutos, posteriormente aparecen los 8 pulsos correspondientes a las horas; con los 12 pulsos siguientes se obtienen las unidades, decenas y centenas que conforman el día, y los últimos 12 bits corresponden a las unidades, decenas y centenas del número de serie del Omega face.

En toda la señal Omega se presentan primero los bits menos significativos y posteriormente los más significativos.

En la Fig. 9 se muestra el formato de la señal Omega.

-Decodificación de la señal con datos auxiliares.

La señal de datos auxiliares está involucrada en las 16 primeras muestras después de la ventana de sincronía como datos codificados en el canal 1; éstos corresponden a la tasa de muestreo de las señales grabadas y al número de serie del aparato o acelerógrafo que registró el evento, como se ha mencionado en el formato de grabación del cassette.

La decodificación de la tasa de muestreo se hace con la subrutina GETSR, del programa TEKPLOT, que selecciona las 2 primeras palabras del canal 1 después de la ventana de sincronía y guarda en una variable el valor obtenido, dependiendo de los valores del bit más significativo (está en la la. muestra) y del menos significativo (está en la 2a. muestra).

La decodificación del número de serie del aparato se hace con la subrutina GETSN, del programa TEKPLOT, que selecciona las siguientes 14 palabras después de las 2 de la tasa de muestreo, y guarda en una variable el valor entero decimal que resulta de acuerdo a los valores de los bits de datos codificados, es decir, se convierten de un número binario al número decimal correspondiente.

-Subprogramas de esta sección.

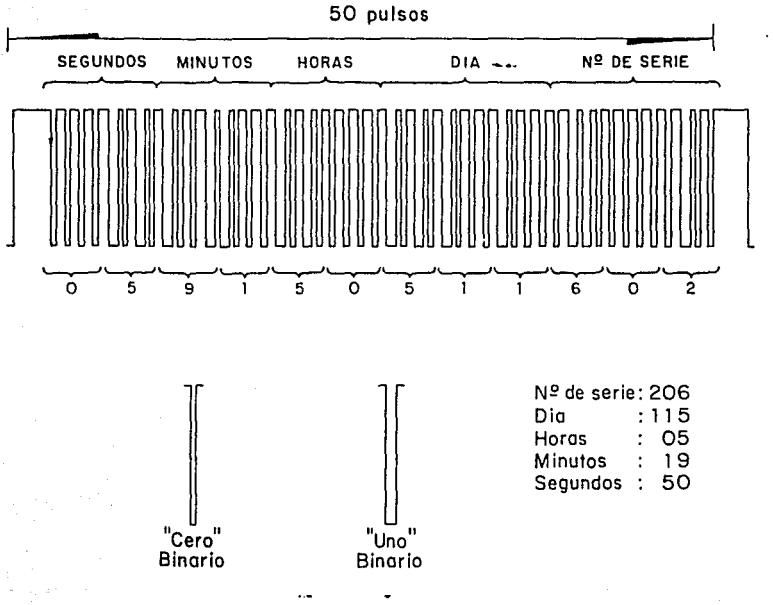


Fig 9 Formato de la señal de tiempo

La decodificación de las señales de aceleración, tiempo y auxiliar se hace al ejecutarse el programa #PLOT, que obtiene, como se ha mencionado con anterioridad, una gráfica de las tres componentes de aceleración (longitudinal, vertical y transversal), muestra los pulsos de la señal de tiempo (reloj Omega) y, al pie del marco de graficación, escribe los datos que corresponden al número de serie del acelerógrafo, periodo de muestreo de las señales, duración del evento , número de puntos o muestras graficadas y el nombre del archivo. Este programa corresponde al programa fuente TEKPLOT.FTN, que hace uso de diversas rutinas propias del sistema, y de las siguientes subrutinas:

ANMOD	Pone a la terminal en modo alfanumérico.
BOXIT	Dibuja un marco para la gráfica.
CLABEL	Dibuja los números y letras de la escala vertical.
DRWCHN	Dibuja la señal de un canal.
DRWEVT	Abre las rutinas de graficación y llama a otras rutinas para dibujar la gráfica de un evento.
GETSNO	Obtiene el número de serie del aparato que registró el evento.
GETSR	Obtiene la tasa de muestreo de las señales.
GMOD	Pone la terminal en modo gráfico.
MAKNAM	Convierte el nombre de un archivo a un nombre dentro del subdirectorio .RAWDATA.
NEWPAG	Borra la pantalla de la terminal TEKTRONIX.

NUMBER	Dibuja un número.
PLOT	Dibuja un punto de coordenadas en la pantalla.
PLOTS	Inicia las rutinas de graficación.
PLTTEK	Pone los bits apropiados a la terminal de graficación.
PSTAT	Dibuja en el pie del marco de graficación los datos de la gráfica.
READIN	Lee por ventanas de datos hasta fin de archivo y lo cierra.
SCALIT	Calcula los factores de escala.
SYMBOL	Dibuja una cadena de caracteres.
XTICKS	Dibuja las cotas horizontales de la gráfica.
YLABEL	Ubica los puntos donde se etiqueta la escala vertical.
YTICK	Dibuja las cotas verticales de la gráfica.

Rutinas propias del sistema:

CNVB$A	Convierte un número en una cadena de caracteres.
ENCD$A	Convierte un carácter o número en una cadena de caracteres de un tamaño determinado.
JSTR$A	Justifica a la derecha/izquierda una cadena de caracteres.
MSTR$A	Coloca en cierta posición una cadena de caracteres.
MSUB$A	Agrega una cadena de caracteres a otra.
PRWF$$	Lee/escribe/posiciona/trunca algún archivo.
SLEEP$	Suspende un tiempo definido la ejecución de un

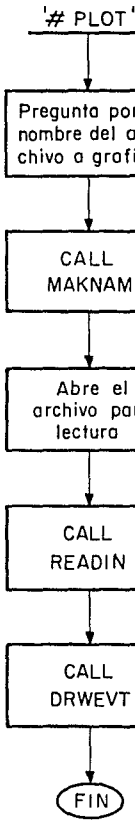
proceso.

TNOUA Imprime ciertos caracteres sin cambiar de renglón.

El programa completo se presenta en el apéndice A2 y en la Fig. 10 se muestra el diagrama simplificado de #PLOT.

La salida del programa de graficación se presenta en la Fig.11.

PROGRAMA



Pone el archivo (RAWDATA > archivo) dentro del subdirectorio. RAWDATA

Lee ventanas de datos (cada 256 palabras) hasta fin de archivo

Dibuja la gráfica con los 3 señales de aceleración además de los parámetros (Nº de serie, periodo de muestreo, tipo de aparato, longitud del evento)

Fig 10

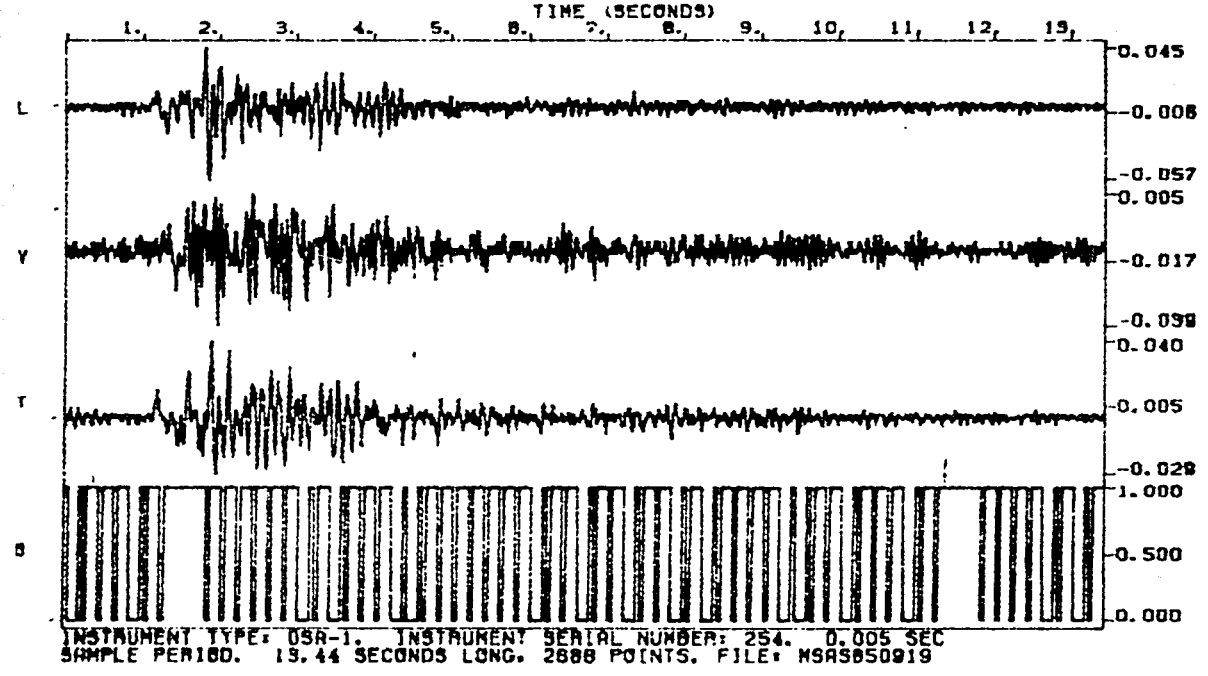

TIME (SECONDS)
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
L
Y
T
B
0.045
0.008
-0.057
0.005
-0.017
-0.038
0.040
0.005
-0.028
1.000
0.500
0.000
INSTRUMENT TYPE: DSR-1. INSTRUMENT SERIAL NUMBER: 254. 0.005 SEC
SAMPLE PERIOD. 13.44 SECONDS LONG. 2688 POINTS. FILE: MSAS850919

Fig 11

5. ALMACENAMIENTO DE LOS ACELEROGRAMAS.

-Descripción del formato de almacenamiento.

Cuando se ejecuta el programa #DSP, los datos transferidos del DSP-3 se almacenan en un archivo que se crea en un subdirectorio llamado .RAWDATA, este archivo tiene el nombre que el usuario ha elegido al inicio del comando Play to Ieg.

Los datos se almacenan en palabras de 16 bits, de manera consecutiva tal y como fueron grabados en los cassettes.

Cuando se ejecuta el programa #PLOT, se abre el archivo que contiene los datos leídos previamente y se leen cada 256 palabras, es decir, con cada frame de muestras (64 muestras por frame) se va llenando en un vector de dos índices con los valores de cada canal, sin considerar los valores correspondientes al canal 4 que corresponden a la paridad vertical.

El archivo que contiene el acelerograma (las tres componentes de aceleración) tiene almacenadas las muestras de sincronía (muestras de valor cero) entre las muestras de valores de aceleración. Para hacer compatible dicho archivo, al hacer uso de otros programas para el procesamiento de las señales,fue necesario desarrollar un programa de computadora que sustituyera las muestras de sincronía por valores de aceleración. Al ejecutar este programa

llamado #LINEAL1 , se leen las muestras del archivo creado con el programa #DSP, se hace una interpolación lineal cada 64 muestras y se almacenan en un nuevo archivo que el usuario nombra.

La interpolación lineal es de la forma:

$$y_i = (y_{i+1} - y_{i-1})/2 + y_{i-1}$$

donde y_i es el valor de aceleración para cada muestra 64 que está en el arreglo de dos dimensiones, y_{i+1} , y_{i-1} son los valores de aceleración de la muestra posterior y anterior a la de sincronía respectivamente.

Ya que se ha llevado a cabo la sustitución de los valores, el archivo estará listo para tomarlo como de datos de entrada a otros programas que llevan a cabo el procesamiento de las señales de aceleración (cálculo de espectros de respuesta, filtrado, integración, etc.) previa adecuación de las mismas con el programa #SELECT que permite eliminar pulsos de calibración y glitches entre otras cosas.

-Subprogramas de esta sección.

Para hacer compatible la información almacenada en el archivo que se crea al leer el cassette para el uso del progrma #SELECT fue necesario sustituir las muestras de sincronía por valores de aceleración y colocar los 12 bits de aceleración en el extremo derecho de cada palabra del archivo mediante el programa LINEAL.FTN en el cual se usaron rutinas propias del sistema y las siguientes

subrutinas:

READIN	Lee una ventana de datos (64 muestras).
MAKNAM	Genera un archivo dentro del subdirectorio .RAWDATA.

Rutinas propias del sistema:

TONL	Hace un cambio de renglón.
PRWF$$	Lee/escribe/posiciona/trunca algún archivo.
CLOS$A	Cierra un archivo.

El programa LINEAL.FTN se presenta en el apéndice A3.

6. MANUAL DE USUARIO.

-Lectura de datos.

Para iniciar la lectura de eventos de un cassette, se requiere colocar el cassette en el DSP-3 verificando que esté rebobinado en su inicio, colocando los interruptores del DSP-3 en las siguientes posiciones:

INTERRUPTORES	CASSETTE DSA-1	CASSETTE PDR-1
CONTROL	EXT	EXT
DIGITAL OUTPUT	SERIAL	SERIAL
FORMAT	BINARY	BINARY
RECORDER	DSA	PDR
SERIAL BAUD RATE	96	96

- Encender el DSP-3.

- Entrar al sistema PRIME:
  a) Teclear LOGIN y a continuación la clave de acceso autorizada.
  b) Entrar al directorio donde se instaló el programa #DSP.

- Teclear SEG #DSP (programa de transferencia de datos entre

DSP-3 y computadora PRIME). Aparecerá un mensaje de bienvenida.

- Responder con un 12 a la solicitud de número de línea asignada a la lectora o DSP- 3.

- Oprimir el interruptor STOP-TAPE en el DSP-3 (es un reset que inicia la comunicación).

- Teclear la letra correspondiente al comando deseado según el menú que se presenta, y que se muestra a continuación:

Commands Available:

S : Stop
F : Forward to IEG
R : Rewind to IEG
B : Rewind to Beginning
P : Play to IEG
A : Play to EOT
E : Stop Event
T : Toggle to ascii input
ESC : Unassign line and quit program

Durante la transferencia (con el comando Play to Ieg), aparece el mensaje del número de frame que se está leyendo con los valores

de sincronía de c/u, es decir 16 ceros que corresponden a los 4 canales. Al terminar de leer un evento, aparece de nuevo el menú de comandos.

Para leer un solo evento teclear P.

Para leer un cassette completo teclear A.

Para terminar teclear ESC.

-Limitaciones

Se solicita un nombre de archivo para cada evento leído.

Para salir del programa si la terminal no tiene la tecla ESC, apagar el DSP-3 y enviar cualquier comando. Como el DSP-3 no responderá el programa desasigna la línea de comunicación y sale al sistema PRIME.

-Graficación.

Para obtener la gráfica de los archivos que contienen los datos leídos de los cassettes se requiere:

- Entrar al sistema:
  a) Teclear LOGIN y a continuación la clave de acceso autorizada.
  b) Entrar al directorio donde se instaló el programa #PLOT.

- Teclear SEG #PLOT.

- Teclear el nombre del archivo que desea graficar. Dibuja la gráfica y termina el programa.

-Limitaciones.

Es necesario ejecutar el programa #PLOT para cada evento que se desee.

No es posible interrumpir la graficación ya que la terminal está en modo gráfico.

La señal de reloj se dibuja en la gráfica y su decodificación debe realizarla el usuario, observándola en la terminal.

Las gráficas de las señales de aceleración se muestran tal y como se grabaron, es decir, incluyen pulsos de calibración del aparato, quedando distribuidas en el marco de la gráfica.

-Compatibilidad.

Para hacer compatible el archivo creado con #DSP para usarlo con el programa #SELECT que reside en el sistema, se ejecuta el programa #LINEAL1, para lo cual se requiere:

- Entrar al sistema:

a) Teclear LOGIN y a continuación la clave de acceso autorizada.

b) Entrar al directorio donde se instaló el programa #LINEAL1, que debe corresponder al mismo donde está #DSP ya que la búsqueda del archivo se hace en el subdirectorio .RAWDATA que se ha creado ahí.

- Teclear SEG #LINEAL1.

- Teclear el nombre del archivo de datos originales. (Archivo que se crea al ejecutar #DSP).

- Teclear el nuevo nombre de archivo que contendrá los valores de aceleración sin incluir los ceros de sincronía y que será el archivo de entrada al ejecutar #SELECT.

-Limitaciones.

Es necesario ejecutar el programa #LINEAL1 siempre que se desee hacer uso del programa #SELECT.

El programa #LINEAL1 debe permanecer en el directorio donde se hace la lectura del cassette ya que buscará el archivo de datos originales en el subdirectorio .RAWDATA.

7. REFERENCIAS Y BIBLIOGRAFIA

FORTRAN.
PRIME Computer, Inc.

Load and Seg Reference Guide PDR3524.
PRIME Computer, Inc.

Operating Instructions for model DSP-3 Digital Playback System.
KINEMETRICS, INC.
May 1982.

Primos Commands Reference Guide FDR3108-101A.
PRIME Computer, Inc.

Primos Subroutines Reference Guide PDR3621. Revision A.
PRIME Computer, Inc. 1980.

STRONG MOTION ACCELEROGRAPH SYSTEM.
Training Manual.
University of California San Diego.
November 1984.

PROGRAMA DSP.FTN

APENDICE 1

```
C PROGRAM DSP.FTN
C Program Design: Kenneth Shapiro
C Purpose: Communicate with DSP-3 computer. Transfer files to Prime.
C Modified for Kinemetrics Inc. July 1982
C
C Revision A. 9-6-84 to add ability to use DSP-3 block protocol and prompts.
C
      SUBROUTINE MAIN
$INSERT SYSCOM>A$KEYS
$INSERT SYSCOM>KEYS.F
      PARAMETER NLINES=1
      INTEGER*2 ASCII,BRK$,CNTLO,COM(11),EVNUM,FILNAM(16),FNAME(32)
      INTEGER*2 I,IC,ICODE,INBUF(2ØØØ),IUNIT,J1,LINE,LINLST(NLINES)
      INTEGER*2 LINOCT,NLEN,PARITY,PDR,STATV(8),T$OUTP
      INTEGER*4 BUF(4,64)
      PARAMETER T$OUTI=3, CARRET=LS(:215,8)+RT('  ',8)
      PARAMETER NCMDS=11
      COMMON /BREAKV/BRK$
      COMMON /COMBUF/INBUF,BUF,ASCII,PARITY,PDR
      COMMON /LINCOM/LINE
      EXTERNAL BREAK
C Here is where the line number is set.
      DATA LINLST/:12/
C This is the command table
      DATA  COM/'S  ','F  ','R  ','B  ','P  ','A  ','E  ',:1154ØØ,'T  ',
     +'C  ','D  '/
C Display signon message
      CALL TON
      CALL TNOU ('Welcome to DSP, the interface to the DSP-3.',43)
      CALL TNOU ('Copyright Kinemetrics Inc. 1984 ',32)
      CALL TNOU ('Revision A.1',12)
C Setup some variables
C Sets default value for data transfer type.  Should be one for ascii, Ø for
C binary.
      ASCII=Ø
      BRK$=2
      CNTLO=LS(:221,Ø)
C Sets default value for parity checking.  Should be 1 for yes, Ø for no.
      PARITY=Ø
C Sets default value for parity. should be one for PDR-1, two for DSP-1
      PDR=1
C create new on-unit for QUITS condition. This causes a CNTL-P to
C be trapped and handled by the program.
      CALL MKONSF ('QUITS',5,BREAK)
      CALL ENCARS (.FALSE.)     /* Make sure breaks are enabled
      CALL TON
      CALL TNOUA('Dame el numero de linea asignada (EN OCTAL):  ',46)
      CALL TINOCT(NUM)
      LINLST(1)=NUM
      CALL TOF
C Try to assign the line, exit if unsuccesful
1Ø    CALL ASNLNC (LINE,LINLST,NLINES,ICODE)
      IF (ICODE.NE.Ø) CALL EXIT
      LINOCT=LS(LINE/8,3)+RT(LINE,3)+'ØØ'
      WRITE (1,2Ø) LINOCT
2Ø    FORMAT ('You will be communicating over line :',A2,'.')
C reset DSP-3 by simulating a 'S' command (will return to line 3Ø when done
25    I=1
      GO TO 11ØØ
C turn on prompt mode.
3Ø    CALL SLEEPS (IFIX(L(5ØØØ)))
      CALL PROMPT (ICODE)
      IF (ICODE.EQ.Ø) GO TO 1ØØ
4Ø    CALL TNOU ('The DSP-3 does not respond.',27)
      GO TO 16ØØ
C Command Loop
C Sets break key to no function
1ØØ   BRK$=Ø
      CALL FLUSH
      WRITE (1,11Ø)
11Ø   FORMAT (/'Commands Available:'/'S: Stop'/'F: Forward to IEG'/
     +    'R: Rewind to IEG'/'B: Rewind to Beginning'/
     +    'P: Play to IEG'/'A: Play to EOT'/'E: Stop Event')
      IF (ASCII.EQ.1) WRITE (1,12Ø)
```

```
120   FORMAT ('T: Toggle to binary input')
      IF (ASCII.EQ.0) WRITE (1,130)
130   FORMAT ('T: Toggle to ascii input')
      IF (PARITY.EQ.0) WRITE (1,140)
140   FORMAT ('C: Toggle parity check on')
      IF (PARITY.EQ.1) WRITE (1,150)
150   FORMAT ('C: Toggle parity check off')
      IF ((PARITY.EQ.1).AND.(PDP.EQ.0)) WRITE (1,160)
160   FORMAT ('D: Toggle to PDP-1 parity')
      IF ((PARITY.EQ.1).AND.(PDP.EQ.1)) WRITE (1,170)
170   FORMAT ('D: Toggle to DGA-1 parity')
      WRITE (1,180)
180   FORMAT ('<ESC>: Unassign serial line and quit program.')
      CALL TNOUA ('Command:  ',9)
      CALL CNINS (IC,1,ICODE)
C Check for valid command, jump to processing routine if valid.
      DO 190 I=1,NCMDS
      IF (LT(IC,8).NE.LT(COM(I),8)) GO TO 190
      GO TO (1100,1200,1300,1400,1500,1600,1100,1800,1900,2000,2100),I
C              S    F    R    B    P    A    E  ESC  T    C    D
190   CONTINUE
      CALL TONL
      CALL TNOU ('Unrecognized command.  ',23)
      GO TO 100
C processing for 'Stop' and 'rtop Event' commands.
1100  CONTINUE
      CALL TSAMLC (LINE,LOC(COM(I)),1,TSOUTP.STATV,1,ICODE)
      CALL SLEEPS (INTL(1000))
      GO TO 30
C processing for 'Forward to IEG', 'rewind to Beginning of tape' and 'Rewind to
C IEG' commands.
1200  CONTINUE
      CALL TSAMLC (LINE,LOC(COM(I)),1,TSOUTP,STATV,1,ICODE)
      ITIME=0
1300  CALL WAITPR (ICODE)
      IF (ICODE.EQ.0) GO TO 1000
      IF (ITIME.GT.40) GO TO 70
      ITIME=ITIME+1
      GO TO 1300
C processing for the 'Play' to IEG command
1500  CONTINUE
      CALL GETUFD (ICODE)
      IF (ICODE.NE.0) GO TO 1000
C Clear file name variable to spaces
      DO 1510 II=1,10
1510  FILNAM(II)='  '
1520  CALL TONL
      CALL TNOUA ('Input filename for data output:  ',32)
C Get file name
      READ (1,1530,END=1520) FILNAM
1530  FORMAT (10A1)
      IF (FILNAM(1).EQ.'  ')GO TO 1000
C Convert file name to a tree-name and display it
      CALL MK_NAM (FILNAM,FNAME)
      WRITE (1,1540)FNAME
1540  FORMAT(A9,2)
C Use RMS system subroutine to open file on file unit 1 (Fortran unit 5)
      IUNIT=1
      IF (.NOT.OPENVSA(ASSDEV0+ASSNAME,FNAME,64,IUNIT,ASVNEW,0,0))GOTO 1520
C file open, now read a file, subroutine returns with file closed.
      CALL GETEVE(ICODE,IUNIT)
C if break key was pressed, prompt mode must be reset.
      IF (ICODE.EQ.1) GO TO 30
      IF (ICODE.EQ.4) GO TO 100
C otherwise go and get prompt.
      GOTO 25
C Multiple Play  command.
1600  CALL GETUFD(ICODE)
      IF (ICODE.NE.0) GO TO 1000
      CALL TONL
      CALL TNOUA('Input base file name:  ',22)
      READ(1,1530,ERR=1600)FILNAM
      IF (FILNAM(1).EQ.'  ') GO TO 100
      DO 1620 I=1,29
      IF (FILNAM(I).EQ.'  ') GO TO 1630
```

```
162Ø  CONTINUE
      GO TO 16ØØ
163Ø  NLEN=1
      EVNUM=Ø
      IUNIT=1
164Ø  CONTINUE
C Create a file name of the form aaaaaaa.nnn  where aaaaaa is the
C first 13 characters of the base file name and nnn is the event number.
      EVNUM=EVNUM+1
      FILNAM(NLEN)='.'
      FILNAM(NLEN+1)=LS(EVNUM/1ØØ+176,8)
      FILNAM(NLEN+2)=LS(MOD(EVNUM,1ØØ)/1Ø+176,8)
      FILNAM(NLEN+3)=LS(MOD(EVNUM,1Ø)+176,8)
C convert file name to a treename
      CALL MAKNAM(FILNAM,FNAME)
      WRITE(1,154Ø)FNAME
C after the first event don't bother user about duplicate files
      IF (EVNUM.GT.1) GO TO 165Ø
C open file, warn user if file exists, return to command loop on error
      IF (.NOT.OPNVSA('ASFDWE-ASSAME',FNAME,64,IUNIT,ASVNEW,Ø,Ø))GOTO 169Ø
      GO TO 166Ø
C open file, no warning to user this time
165Ø  IF (.NOT.OPENSA('ASFDWE-ASSAME',FNAME,64,IUNIT))GOTO 169Ø
C read event file, returns with file closed
166Ø  CALL GETEVE(ICODE,IUNIT)
C if break key was pressed restart program
      IF (ICODE.EQ.1) GO TO 2Ø
C test for problem in getting data. ICODE will be 2 if a E-O-T occured
      IF (ICODE .EQ. 2)GOTO 167Ø
      CALL SLEEPS(INT2(25ØØ))
      GOTO 164Ø
C process end of tape condition.
167Ø  CONTINUE
      WRITE(1,168Ø)EVNUM
168Ø  FORMAT('End of tape.',I4,' events written.')
      GOTO 1ØØ
C no file error message
169Ø  CONTINUE
      CALL TROU ('Unable to open file.',2Ø)
      GOTO 1ØØ
C exit to primos <ESC> key pressed
17ØØ  CONTINUE
      CALL ASAMLC (ICODE)
      CALL EXIT
      GOTO 1Ø
C toggle from ascii to binary or from binary to ascii.  Ascii if ASCII=1.
19ØØ  IF (ASCII.EQ.Ø) GO TO 191Ø
      ASCII=Ø
      GO TO 1ØØ
191Ø  ASCII=1
      GO TO 1ØØ
C toggle from parity check to no parity check. Parity check if PARITY=1.
2ØØØ  IF (PARITY.EQ.Ø) GO TO 2Ø1Ø
      PARITY=Ø
      GO TO 1ØØ
2Ø1Ø  PARITY=1
      GO TO 1ØØ
C Toggle between PDR-1 parity and DSA-1 parity
21ØØ  IF (PDR.EQ.Ø) GO TO 211Ø
      PDR=Ø
      GO TO 1ØØ
211Ø  PDR=1
      GO TO 1ØØ
      END
C
      SUBROUTINE ASAMLC(LINE,LINLST,NLINES,ICODE)
C
C Assign AMLC (RS-232 serial) line number. Try to assign each line in
C LINLST array until either we run out of lines or we successfully assign
C a line.
      INTEGER*2 I,II,ICODE,LINE,LINLST(1),NLINES,OUTLIN(12)
C Enable the desired baud rate by removing the C$ from the appr. line
C 2213 = 3ØØ, 2313 = 12ØØ and 2413 = 96ØØ baud

C$    DATA OUTLIN/'  AS AMLC TRAN  nn 2213 '/
C$    DATA OUTLIN/'  AS AMLC TRAN  nn 2313 '/
```

```
      DATA OUTLIN/'  AS AMLC TRAN nn 2412 '/
C
      DO 10 I=1,NLINES
      LINE=LINEST(I)
      II=LS(LINE/8,6)+RT(LINE,3)+'60'X
      OUTLIN(9)=II
      OUTLIN(1)=23
      CALL TNOUA('Attempting to assign line ',27)
      CALL TOOCT(LINE)
      CALL TNOUA(' -- ',4)
C     CALL TNOU(OUTLIN(2),23)
C  This is a 'magic' subroutine that can do just about anything in the
C  operating system.
C
      CALL DOSSUB(OUTLIN,ICODE)
C
      IF(ICODE .EQ. 0)RETURN
10    CONTINUE
      CALL TNOU('All lines in use--sorry. ',24)
      RETURN
      END
C
C----This subroutine is executed when BREAK is hit.
C   (CP is a dummy argument passed by operating system.)
      SUBROUTINE BREAK (CP)
      INTEGER*2 ASCII,BRKS,COUNT,INBUFF(200)
      INTEGER*2 LINE,PARITY,PDR,STATV(8),STOP
      INTEGER*4 BUF(4,64),CP
      COMMON /BREAKV/IPKS
      COMMON /COMBUF/INBUF,BUF,ASCII,PARITY,PDR
      COMMON /LINCOM/LINE
      DATA STOP/'S '/
C---Check for calling level 2 is menu level.
      IF (BRKS.GE.1) RETURN
C---Initialize loop counter.
      COUNT=0
C---Send 'S' to stop PSF-3
      CALL TSAMLC (LINE,LOC(STOP),1,3,STATV,1,ICODE)
      CALL TONL
      CALL TNOU ('Interrupted.  Returning to menu.',30)
C wait for serial lines to clear
      CALL SLEEPS (INTL(100))
C clear out the amlc buffer
      CALL FLUSH
C set break flag to one to indicate that this has been done
30    BRKS=1
      RETURN
      END

C Subroutine to unassign line, close files and exit to system
      SUBROUTINE UNAMLC(CP)
$INSERT SYSCOM>KEYS
      INTEGER*2 ICODE,II,LINE,OUTLIN(7)
      INTEGER*4 CP
      COMMON /LINCOM/LINE
      DATA OUTLIN/'  UN AMLC nn  '/
C Send a couple of nulls to clean up terminal in case <ESC> was pressed
      CALL TNOUA ('',2)
      OUTLIN(1)=12
      II=LS('FF'X,8)+RT(LINE,3)+'80'X
      OUTLIN(6)=II
      WRITE(1,10)II
10    FORMAT('Unassigning line ',A2,'.  Goodbye.')
C Send command to operating system to unassign line
      CALL DOSCUE(OUTLIN,'COM')
C      CALL EXIT$(0,0)
C
C   Close any outstanding data files.
C
      IF (UNIT$A(1)) CALL CLOSA(1)
      CALL SLEEPS(INTL(200))
      CALL CNSIG$(ICODE)   /* Signal a true QUIT condition (exits to primos).
      RETURN
      END
C
C this subroutine will read one frame (64 samples) of data, ascii or binary.
```

```
C
      SUBROUTINE READIN (LINE,BUF,ICODE)
C
$INSERT SYSCOM>ASYKEYS
      INTEGER*2 BLOCK,BRKT,BUF(1),CNTLO,COUNT,ICODE
      INTEGER*2 II,LINE,MAX,NUM,PROMPT,STATV(8)
      INTEGER*4 TIMOUT
      COMMON /BREAKV/BRKS
C setup data constants.
      CNTLO=LS!:22),0
      COUNT=0
      TIMOUT=INTL(180)
      NUM=450
      MAX=1024                          /* number of characters to read
      PROMPT=0
C ICODE=1 if data is binary format.
      IF (ICODE.EQ.0) MAX=512
C clear error code.
      ICODE=0
C Check for break key pressed.
10    IF (BRKS.EQ.1).RETURN
C check to see if any characters have arrived.
      CALL TSAMLC(LINE,LOC(BUF),0,4,STATV,1,ICODE)
      IF (STATV(1).NE.0) GO TO 20
C no character yet so try again.
      CALL SLEEPS(TIMOUT)
      GO TO 10
C read the block prefix character.
20    CALL TSAMLC(LINE,LOC(BUF),1,1,STATV,1,ICODE)
C got the block prefix, now process it.
      BLOCK=AND(LS(BUF(1),8),:177)
C     CALL TNOUA (BUF,1)      /* display block character for debugging.
C test for prompt.
      IF (BLOCK.EQ.:70) GO TO 80
C test for valid prefix.
      IF ((BLOCK.LT.:110).OR.(BLOCK.GT.:117)) GO TO 70
C if it's an H then end the frame.
      IF (BLOCK.EQ.:110) GO TO 30
C it's not an H so check for end of tape.
      IF (AND(BLOCK,1).NE.0) GO TO 60
C now test for end of event.
      IF (AND(BLOCK,2).NE.0) GO TO 50
C if a prompt has been received, go to prompt exit point
      IF (PROMPT.EQ.1) GO TO 10
C no data is going to arrive so send another block request and start over
C Request data transfer.
      CALL TSAMLC(LINE,LOC(CNTLO),1,2,STATV,1,ICODE)
      GO TO 10
C start of data gathering loop, find out how many characters in buffer
30    CALL TSAMLC(LINE,0,0,4,STATV,1,ICODE)
C Check for break key pressed.
      IF (BRKS.EQ.1) RETURN
C see if the buffer is full.
      IF (STATV(1).GE.MAX) GO TO 40
C there is a full buffer on its way so wait a while and try again.
      COUNT=COUNT+1
      CALL SLEEPS (TIMOUT)
C this shouldn't take more than ten seconds.
      IF (COUNT.LE.5) GO TO 30
C buffer is not full, so give up
      GO TO 50
C get the frame of data and return.
40    CALL TSAMLC(LINE,LOC(BUF),MAX,0,STATV,1,ICODE)
C Request data transfer for next block of data.
      CALL TSAMLC(LINE,LOC(CNTLO),1,3,STATV,1,ICODE)
      RETURN
C   (Jump here when end-of-event character (Z) has been detected)
50    CONTINUE
      ICODE=1
55    CALL TONL
      CALL TNOU('End-of-event.',13)
      RETURN
C Jump here if end of tape is detected.
60    ICODE=2
      CALL TONL
      CALL TNOU ('End-of-tape.',12)
      RETURN
```

```
C Jump here in the unlikely event that a bad block character is detected.
70      ICODE=3
        CALL TONE
        CALL TNOU ('Bad block character detected.',29)
        RETURN
C Jump here if a prompt is detected.
80      PROMPT=1
        GO TO 10
C exit point when prompt is found.
90      ICODE=4
        GO TO 55
        END
C
C  Function: Translate ASCII data into binary for storage, one word
C  at a time.
C
        INTEGER*2 FUNCTION TRANS(IDAT1)
        INTEGER*2 I,II,JDAT,T(31)
        INTEGER*4 IDAT,IDAT1
        DATA T/10,11,12,13,14,15,16*0,1,2,3,4,5,6,7,8,9,6*0/
C Zero the accumulator word
        JDAT=0
C Save input characters for safety
        IDAT=IDAT1
C Translate the four hex nibbles into binary nibbles and assemble into buffer
        DO 10 I=0,3
        II=T(AND(RS(IDAT,I*8),:07))
        JDAT=OR(JDAT,LS(II,I*4))
10      CONTINUE
C   One of the following two statements is used depending on an internal
C   format selection switch in the DSP-3.
C5      TRANS=JDAT
        TRANS=OR(LS(JDAT,1),RS(JDAT,15))
        RETURN
        END
C
C This subroutine converts a file name to a path name.
C
        SUBROUTINE MAKNAM(FILNAM,FNAME)
        INTEGER*2 I,ILTH,FILNAM(16),FNAME(32),K1
        INTEGER*2 K2,K3,K6,K7,FNAME(29),PLTH
        DATA FNAME/'* >  R A W D A T A > 1 2 3 4 5 6 7 8 . 0 0 0'/
C Setup variables
        PLTH=11
C Clear output name buffer
        DO 10  I=1,32
        FNAME(I)=:120F24
10      CONTINUE
C Clear unused portion of dummy pathname
        DO 30 I=1,12
        FNAME(I+PLTH)=:120F24
20      CONTINUE
C Determine length of file name
        DO 30 ILTH=1,16
        IF (FILNAM(ILTH).EQ.' ') GO TO 40
30      CONTINUE
C Move file name into proper area of dummy pathname
40      DO 50 K1=1,ILTH
        FNAME(K1+PLTH)=FILNAM(K1)
50      CONTINUE
C Setup variables for packing loop
        K2=1
        K6=PLTH+ILTH+1
        K3=K6/2
C Loop to pack dummy pathname (1 char/cell) into output pathname (2 char/cell)
        DO 60 K1=1,K3
        FNAME(K1)=OR(AND(FNAME(K2),:177400),RS(FNAME(K2+1),8))
        K2=K2+2
60      CONTINUE
        RETURN
        END
C
C Read in a single event from the DSP-3, save in a file open on IUNIT.
C Return with file closed.  If end of tape timeout occurs return with ICODE=1
C
```

```
      SUBROUTINE GETEVE(ICODE,IUNIT)
$INSERT SYSCOM>KEYS.F
$INSERT SYSCOM>3ASKEYS
      INTEGER*2 ASCII,BRKS,CNTLO,DAT(4,64),ERRORS,FRMCNT,I
      INTEGER*2 ICHAR,ICODE,INBUF(2###),IRNW,IUNIT,J,JCHAR
      INTEGER*2 JJ,LINE,PARITY,PDR,STATV(8),TRANS,T$OUTP
      INTEGER*4 BUF(4,64)
      PARAMETER T$OUTP=3
      COMMON /BREAKV/BRKS
      COMMON /LINCOM/LINE
      COMMON /COMBUF/INBUF,BUF,ASCII,PARITY,PDR
      DATA ICHAR,JCHAR/'1P',:#442#/
C Setup variables and constants
      BRKS=#
      CNTLO=LS(:221,F)
      FRMCNT=#
C send request to enter block mode.
5     CALL TSAMLC(LINE,LOC(ICHAR),1,T$OUTP,STATV,1,ICODE)
      CALL SLEEPS (INSL(1###))
C clear AMLC buffers.
7     CALL TSAMLC (LINE,LOC(INBUF),1,1#,STATV,T,ICODE)
C send request to play an event.
      CALL TSAMLC(LINE,LOC(ICHAR),1,T$OUTP,STATV,2,ICODE)
C Wait for DSE-3 to digest last command and get started.
      CALL SLEEPS (INSL(2###))
C now send a block request to get things started.
      CALL TSAMLC (LINE,LOC(CNTLO),1,T$OUTP,STATV,1,ICODE)
C Start of frame reading loop set ICODE for ascii or binary transfer
1#    ICODE=ASCII
C Read one frame of data
      CALL READIN (LINE,BUF,ICODE)
C  Quit now if break key has been pressed.
      IF (BRKS.NE.#) GO TO 17#
C Check for error condition
      IF (ICODE.EQ.1) GO TO 2##   /* E-O-F CHARACTER
      IF (ICODE.EQ.2) GO TO 3##   /* E-O-T CHARACTER
      IF (ICODE.EQ.4) GO TO 4##   /* PROMPT CHARACTER
      IF (ICODE.NE.#) GO TO 5##   /* Read too many characters
C Increment frame counter (only exists for users benifit).
      FRMCNT=FRMCNT+1
C Test for data type
      IF (ASCII.EQ.1) GO TO 8#
C Binary data.  Packed 4 bytes per cell.  Unpack into new array.
      DO 2# J=#,31
C Get the first four words
      DO 2#  I=#,1
      DAT (2*I+1,2*J+1)=AND(RSH(BUF(I+1,J+1),16),:177777)
2#    DAT (2*I+2,2*J+1)=AND(BUF(I+1,J+1),:177777)
C Now get the next four words
      DO 3#  I=#,1
      DAT (2*I+1,2*J+2)=AND(RSH(BUF(I+3,J+1),16),:177777)
3#    DAT (2*I+2,2*J+2)=AND(BUF(I+3,J+1),:177777)
C   Next 4 lines needed depending on an internal format selection switch
C   in the DSE-3.
      DO 4# I=1,4
      DO 4# K=1,2
      L=2*J+K
C This is a 1 bit left rotate.
      DAT (I,L)=OR(LS(DAT(I,L),1),RS(DAT(I,L),15))
4#    CONTINUE
      GO TO 7#
C  Translate from ASCII to binary (using TRANS function)
8#      DO 6# J=1,64
        DO 6# I=1,4
6#      DAT(I,J)=TRANS(BUF(I,J))
C Time to check the parity here
7#    IF (PARITY.EQ.#  GO TO 11#
      ERRORS=#
C First see what kind of parity to check for
      DO 10# I=1,63
C This is proper parity checking for DSA-1, last word should be :37777
      IF (PDR.EQ.#)J=XOR(DAT(I,1),DAT(2,I),DAT(3,I),DAT(4,I),:37777)
C This is proper parity checking for PDR-1, last word should be :37777
      IF (PDR.EQ.1)J=XOR(DAT(I,1),DAT(2,I),DAT(3,I),DAT(4,I),:37777)
C If parity is right, J should be #
      IF (J.EQ.#) GO TO 12#
```

```
C Parity error found, increment error counter
      ERRORS=ERRORS+1
C If this is PSA-1 data also check for LRCC error
      IF (PDP.EQ.1) GO TO 1ØØ
C Outer loop is to eliminate track four
      DO 9Ø J=1,3
C First reset the LRCC counter
      LRCC=Ø
C Now determine what the LRCC is
      DO 8Ø K=2,15
8Ø    LRCC=XOR(LRCC,AND(RS(DAT(J,I),K),1))
C Test for correct LRCC, increment error counter if lrcc is bad
9Ø    IF (LRCC.NE.Ø) ERRORS=ERRORS+1
1ØØ   CONTINUE
C Send a <CR> to the terminal
11Ø   CALL TNOUA (LS(:215,8),1)
C Now skip a few spaces
      CALL TNOUA ('          ',1Ø)
C Display the frame number
      CALL TODEC (FRMCNT)
      CALL TNOUA ('--',2)
C Now display the frame sync characters.
      DO 12Ø I=1,4
      CALL TOHEX (DAT(I,64))
12Ø   CALL TNOUA ('    ',4)
C Display parity errors if required.
      IF (PARITY.EQ.Ø) GO TO 13Ø
      CALL TODEC (ERRORS)
      CALL TNOUA (' Parity errors.',15)
13Ø   CALL TNOUA (LS(:215,8),1)
      CALL TNOUA ('Read frame',1Ø)
      IF (ERRORS.NE.Ø) CALL TONL
C    Proper sync is indicated by 4 zero words at end of frame.
      DO 14Ø I=1,4
      IF(DAT(I,64) .EQ. Ø)GOTO 14Ø
      CALL TONL
      CALL TNOU ('FRAME SYNC ERROR',16)
      GO TO 15Ø
14Ø   CONTINUE
         CALL TONL
C     Write out buffer. (This program will write out
C     the data even if sync was incorrect.)
C
C        DO 145 I=1,4
C        DO 145 J=1,64
C        CALL TOHEX(DAT(I,J))
C145     CALL TNOUA('    ',4)
C333     FORMAT(4(2X,I6))
15Ø   CALL PRWF$$(KSWRIT+KSPOSR,IUNIT,LOC(DAT),256,INTL(Ø),IRNW,ICODE)
      GOTO 1Ø
C     Exit point for end of tape.
16Ø   ICODE=2
      GO TO 18Ø
C     Exit point for break key pressed.
17Ø   ICODE=1
      CALL TNOU ('Break detected.',15)
18Ø   CALL TERMSA(IUNIT)
      CALL CLOSSA(IUNIT)
      RETURN
19Ø   ICODE=3
      GO TO 18Ø
2ØØ   ICODE=Ø      /*  (Signal successful event)
      GO TO 18Ø
21Ø   ICODE=4
      GO TO 18Ø
      END
C Subroutine to check for the existance of subufd .RAWDATA in current ufd
      SUBROUTINE GETUFD (ICODE)
$INSERT SYSCOM>KEYS.F
$INSERT SYSCOM>A$KEYS
      INTEGER*2 ICODE,LENGTH,FILNAM(4),UNIT,BUFFER(16),TYPE
      DATA FILNAM/'.RAWDATA'/
      UNIT=1
      CALL TONL
      CALL SRCH$$(K$EXST,'.RAWDATA',8,UNIT,TYPE,ICODE)
      IF (ICODE.NE.Ø) GO TO 1Ø
```

```
      IF (TYPE.EQ.4) RETURN
      CALL TRON ('File .RAWDATA not a UFD.',24)
      ICODE=1
      RETURN
1Ø    IF (SYSNOSAL'UFD .RAWDATA does not exist. add',32,A$NDEF))GOTO2Ø
      ICODE=1
      RETURN
2Ø    CALL CREATF('.RAWDATA',Ø,Ø,Ø,ICODE)
      RETURN
      END
C sends start prompt mode to DSP-3 and waits for prompt to return
      SUBROUTINE PROMPT (ICODE)
      INTEGER STATV(8)
      COMMON /LINCOM/LINE
      ICHAR='D'
      ITIME=Ø
      CALL TSAMLC(LINE,LOC(ICHAR),1,2,STATV,1,ICODE)
1Ø    CALL WAITPR (ICODE)
      IF (ICODE.EQ.Ø) RETURN
      IF (ITIME.GE.3Ø) RETURN
      ITIME=ITIME+1
      GO TO 1Ø
      END
C waits until a character is received and tests it for a prompt '>'
      SUBROUTINE WAITPR (ICODE)
      COMMON /LINCOM/LINE
      INTEGER STATV(8)
      ICODE=1
      ITIME=Ø
C find out how many characters in AMLC buffer
1Ø    CALL TSAMLC(LINE,LOC(ICHAR),Ø,4,STATV,1,JCODE)

C if there's something there go and get it.
      IF (STATV(1).GT.Ø) GO TO 2Ø
C after one second give up
      IF (ITIME.GE.1Ø) GO TO 3Ø
C hasn't been one second yet so bump counter and wait a while
      ITIME=ITIME+1
      CALL SLEEPS(INTL(1ØØ))
      GO TO 1Ø
C read character in buffer
2Ø    CALL TSAMLC(LINE,LOC(ICHAR),1,2,STATV,1,JCODE)
C if it's a prompt set error code to zero and get out
      IF (ICHAR.EQ.'>') GO TO 25
C if it's not a prompt there is a problem so set error code to two and quit
      ICODE=2
      GO TO 3Ø
25    ICODE=Ø
3Ø    RETURN
      END
C subroutine to clear out the amlc buffer
      SUBROUTINE FLUSH
      COMMON /LINCOM/LINE
      INTEGER STATV(8)
      CALL TSAMLC(LINE,LOC(ICHAR),Ø,8,STATV,1,JCODE)
      RETURN
      END
```

PROGRAMA TEKPLOT.FTN

APENDICE 2

```
      SUBROUTINE MAIN
$INSERT SYSCOM>A$KEYS
$INSERT SYSCOM>KEYS.F
$INSERT FILDEF
$INSERT PLOTDEFS
      INTEGER*2 FLAG
       LOGICAL LOG
C
       CALL TONL
       CALL TONL
       WRITE(1,33)
33      FORMAT(//,25X,'TEKPLOT.FTN')
       WRITE(1,44)
44      FORMAT(//,10X,'GRAFICA DE DATOS LEIDOS CON EL DSP-3',////)
C init BOX COMMON area
C
1      CONTINUE
10     CONTINUE
       BOXBOT= 0.0
       BOXTOP=7.0
       BOXL=-0.50
       BOXR= 9.5
       BOXHT=BOXTOP-BOXBOT
       BOXLTH=BOXR-BOXL
       BOXORG(1)=BOXL
       BOXORG(2)=BOXBOT
C set character size
       CSIZE=.129
       VFS=5.
       IWCNT=0
       SCONT=0
       CALL RDTK$$ (1,INFO,FILNAM,16,ICODE)
       IF (INFO(1).EQ.1) GO TO 40
20     CALL TNOUA ('  NOMBRE DEL ARCHIVO A GRAFICAR:  ',34)
       READ(1,30)FILNAM
30     FORMAT(16A2)
40     IF (NLEN1A(FILNAM,32).EQ.0) GO TO 20
       CALL MAKNAM (FILNAM,FNAME)
       IF(.NOT. OPENSA(A$READ,FNAME,32,1))GOTO 10000
       FRAMES=0                              /* Reset frame counter
       IWCNT=1                               /* Reset sample counter
         FIN=0
         INI=1
       CALL READIN(ICODE)        /* Get the file here
       CALL TONL
C
       IF (ICODE.NE.1) GO TO 10000
       NCHAN=4
       NSAMP=IWCNT
       IF(NSAMP.EQ.0)  WRITE(1,234) FNAME
234    FORMAT(2X,'NO TIENE DATOS EL ARCHIVO: ',16A2,/)
       IF(NSAMP.EQ.0) GO TO 10000
       ISTART=1
       WRITE(1,5)
5      FORMAT(2X,'Cuantos segundos deseas graficar (MAX=40)')
       READ(1,7) LENG
7      FORMAT(I4)
       IF(LENG.EQ.0) LENG=40
2       FIN=FIN+LENG
       SAM=SRATE/100
       NFRAMS=LENG*SAM
       IF(NFRAMS.GT.NSAMP) NFRAMS=NSAMP
       SFIN=FIN*SAM
       ISTART=INI
       INI=SFIN
4       IWCNT=NFRAMS
       IF(IWCNT.GT.NSAMP) IWCNT=NSAMP
       IF(IWCNT.GE.NSAMP) WRITE(1,345)
345    FORMAT(2X,'SE GRAFICARA LA ULTIMA PARTE')
C       CALL SLEEP$  (2000)
       IF(IWCNT.EQ.0) GO TO 10000
       ISTPOS=1
       CALL DRWEVT (VSCALE,ISTPOS,TRACE,STAT,ISTART,NFRAMS)     /* Draw it here
C
30000  CONTINUE
       READ(1,120) FLAG
```

```
12Ø     FORMAT(I2)
        CALL OPCION(FLAG)
        GO TO (1,2,4,4),FLAG
1ØØØØ  CALL EXIT
        GO TO 1Ø
        END
C
        SUBROUTINE OPCION(FLAG)
        INTEGER*4 OPT
        INTEGER*2 FLAG
        LOGICAL LOG
$INSERT SYSCOM>KEYS.F
$INSERT SYSCOM>ASKEYS
$INSERT PLOTDEFS
C
        CALL TONL
1ØØ     CALL TONL
        LOG=PROMPTA('OPCION (13=MENU)',16,ASDEC,OPT)
1Ø1     GO TO (1,1,13,13,13,13,13,13,13,1Ø,13,13,13),OPT
C
1       FLAG=INTS(OPT)
        RETURN
C
1Ø      FLAG=4
        RETURN
C
13      CALL NEWPAG
        CALL TONL
        CALL TONL
        CALL TONL
        CALL MENU
        GO TO 1ØØ
        END
C
        SUBROUTINE MENU
        WRITE(1,11Ø)
11Ø     FORMAT(6X,'<Ø>',5X,'TERMINA EL PROGRAMA')
        WRITE(1,111)
111     FORMAT(6X,'<1>',5X,'GRAFICA OTRO ARCHIVO')
        WRITE(1,222)
222     FORMAT(6X,'<2>',5X,'GRAFICA EL SIGUIENTE TRAMO')
        WRITE(1,1Ø1Ø)
1Ø1Ø    FORMAT(6X,'<1Ø>',4X,'DESPLIEGA LA GRAFICA ORIGINAL')
        WRITE(1,1313)
1313    FORMAT(6X,'<13>',4X,'DESPLIEGA ESTA LISTA')
        RETURN
        END
C
        SUBROUTINE BOXIT
$INSERT PLOTDEFS
        CALL PLOT (BOXL,BOXBOT,3)
        CALL PLOT (BOXR,BOXBOT,2)
        CALL PLOT (BOXR,BOXTOP,2)
        CALL PLOT (BOXL,BOXTOP,2)
        CALL PLOT (BOXL,BOXBOT,2)
        RETURN
        END
C
C
C
        SUBROUTINE XTICKS (XPNTS)
$INSERT FILDEF
$INSERT PLOTDEFS
C  Purpose: draw tick marks at bottom of box. Scale factor used
C  is based on the number of seconds of data in the record.
        INTEGER*2 SCLNUM(4)
        DATA SCLNUM/1,2,5,1Ø/
        ASRATE=SRATE/1ØØ.
        IF(SRATE .EQ. Ø)ASRATE=1ØØ.    /*  SET DEFAULT TO 1ØØ SAMP/SEC
        ASEC=NSAMP/ASRATE
        ITICK=INTS(ASEC/BOXLTH+.5)
C  Calculate scale interval ISC such that 1Ø to 15 ticks will be needed
C  and ISC = 1,2,5,1Ø,2Ø,3Ø,... etc.
        DO 1Ø I=1,4
1Ø      IF(ITICK .LE. SCLNUM(I))GOTO 3Ø
        ISC=INT(ITICK/BOXLTH+.5)*1Ø
```

```
      GOTO 40
30    ISC=SCLNUM(I)
40    CONTINUE
C  Draw the tick marks.
      YPOS=BOXTOP
      DO 100 I=1,15
      XPOS=((I-1)*(ISC*ASPATE)*XPNTS)+BOXL
      IF(XPOS .GT. BOXR)GOTO 200
      CALL PLOT(XPOS,YPOS,3)
      Y1=YPOS-.08
      CALL PLOT(XPOS,Y1,2)
      X1=INT(ALOG10((I-1)*ISC+.0001))*.055+.20
      CALL PLOT(XPOS-X1,YPOS+.26,3)
C  Label the tick marks
      ISEC=(I-1)*ISC
      VSEC=FLOAT(ISEC)
      IF (ISEC.EQ.0) GO TO 623
      CALL NUMBER(XPOS-X1,YPOS+.10,CSIZE,VSEC,0.0,0)
623   CONTINUE
100   CONTINUE
200   CONTINUE
C  Label X-axis
      CALL SYMBOL(BOXLTH/2-1.0,YPOS+.30,CSIZE,'TIME (SECONDS)',0.0,14)
      RETURN
      END
C
C
C
      SUBROUTINE YLABEL(VSCALE,TRACE)
$INSERT PLOTDEFS
      DO 450 ICTR=1,NCHAN
500   CHAN=ICTR
      YSPOT=(BOXHT+(BOXHT/NCHAN/2.))-(ICTR*(BOXHT/NCHAN))+BOXBOT
      CALL CLABEL(VSCALE,TRACE,YSPOT)
450   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE CLABEL(VSCALE,TRACE,YSPOT)
$INSERT PLOTDEFS
      DATA YLBL /
     & ' ','V',' ','T',
     & ' ',' ','FLG',' ','2-V',
     & 'S','A',' ','LTA',' ','> TH',
     & 'F','LG',' ','3-T',' ','STA',
     & 'L','TA',' ','> TH',' ','FLG',
     & 'NOTE',' ','REQ',' ','----',
     & '----','----'/
      TICK=.10
      NBL=3
400   CONTINUE
      CALL YTICK (BOXR,YSPOT,TICK,-.05,5)   /* Ind. center of plot
      VOLT=TRACE(3,CHAN)-TRACE(4,CHAN)      /* Compute p-p voltage
CC    CALL NUMBER  (BOXR+.15,YSPOT-.05,CSIZE,VOLT,0,NDEC)  /* Disp.
C
      VOLT=(TRACE(3,CHAN)+TRACE(4,CHAN))/2  /* Compute average voltage
      CALL NUMBER  (BOXR+.15,YSPOT-.15,CSIZE,VOLT,0,NDEC)
C
110   VOLT=TRACE(2,CHAN)
      CALL YTICK (BOXR,YSPOT-.2,.0003,-.05)   /* SAY AVERAGE
CC    CALL NUMBER (BOXR+.15,YSPOT-.2,CSIZE,VOLT,0,NDEC)
C
120   Y=(TRACE(3,CHAN)*TRACE(1,CHAN))/(VFS/BOXHT)-TRACE(5,CHAN)
      IF (TRACE(3,CHAN).GT..0001.OR.TRACE(3,CHAN).LT.1.0)
     & CALL YTICK(BOXR,Y,TICK,-.2)   /* SAY MOST POSITIVE
      IF (TRACE(3,CHAN).GT..0001.OR.TRACE(3,CHAN).LT.1.0)
     & CALL NUMBER (BOXR+.15,Y-.1,CSIZE,TRACE(3,CHAN),0,NDEC)
C
130   Y=(TRACE(4,CHAN)*TRACE(1,CHAN))/(VFS/BOXHT)+TRACE(5,CHAN)
      CALL YTICK(BOXR,Y,TICK,.10)
      CALL NUMBER (BOXR+.15,Y,CSIZE,TRACE(4,CHAN),0,NDEC)     /* Max DC neg.
C
      Y=0./(VFS/BOXHT)+TRACE(5,CHAN)
      IF (TRACE(3,CHAN).LT.0) GO TO 440
```

```
      IF (ITEM(14,CHAN).GE.0) GO TO 440
      CALL YTICK(BOXL-.6,BOXBOT,-1)     /* Ind. zero point and id ch. no.
440   CALL SYMBOL (BOXL-.6,YSPOT,.15,ILBL(CHAN),0.,4)
C
500   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE YTICK(XORG,YPOS,XOVER,YUPDN)
      CALL PLOT (XORG,YPOS,3)
      CALL PLOT (XORG-XOVER,YPOS,2)
      CALL PLOT (XORG-XOVER+.05,YPOS+YUPDN,3)
      RETURN
      END
C
C
C
      SUBROUTINE DRWEVT (VSCALE,ISTPOS,TRACE,STAT,ISTART,NFRAMS)
$INSERT SYSCOM>ASKEYS
$INSERT SYSCOM>KEYS.I
$INSERT SYSCOM>ERRD.I
$INSERT FILDEF
$INSERT PLOTDEFS
500   CONTINUE
      CALL ATTDEV (0,7,1,100)   /* The PTYLIB wants the logical device
C                               /* to be the same as the file unit
C      CALL VTLINE (1,'
```

```
',1,ICODE)   /* Eject page (014 here
      NFRAMS=ISTART+NFRAMS-1
      CALL PLOTS(0,0,99)    /* Init plot library
      CALL BOXIT   /* Draw a box !
C
      DO 450 ICTR=1,NCHAN
      CHAN=ICTR
504   YORG=((BOXHT+(BOXHT/NCHAN/2.))-((ICTR*(BOXHT/NCHAN)))+BOXBOT
510   CALL SCALIT (ISTPOS,VSCALE,TRACE)
520   CALL DRWCHN(VSCALE,ISTPOS,TRACE,ISTART,NFRAMS)
450   CONTINUE
C       CALL ELITE(ISTART,NFRAMS,NUSTRT,NUEND)
C       CALL MARCO(ISTART,NFRAMS)
      XLT=BOXLFT/INCH
      CALL PLOT(XLT,0.,3)
      CALL YLABEL(VSCALE,TRACE)
C       CALL FLOO
      CALL PSTAT (TRACE,STAT,ISTMAS,IFPOS)
C       CALL MARCO(ISTART,NFRAMS)
      CALL PLOT (0,0,-999)
      CALL TONL
      CALL TONL
C       WRITE(1,20) ISTART,NFRAMS,XI,YI,XF,YF
20    FORMAT(2X,2(2X,IE),2X,4(1X,F8.2))
      RETURN
      END
```

```
C
C
C
C Trace plotting subr.
C
C IWCNT= total points.Plot area is 0-10. Each point is .01. 1000 points
C
      SUBROUTINE DRWCHN(VSCALE,ISTPOS,TRACE,ISTART,NFRAMS)
$INSERT SYSCOM>A$KEYS
$INSERT SYSCOM>KEYS.F
$INSERT FILDEF
$INSERT PLOTDEFS
400   CONTINUE
      XSCALE=IWCNT/BOXLTH
      IPEN=3
      XPOS=BOXL
      YPOS=EVENT(1,CHAN)
      YPOS=(YPOS*VSCALE)/(VFS/BOXHT)+YORG
      IF (YPOS.GT.BOXTOP) YPOS=BOXTOP
      IF (YPOS.LT.BOXBOT) YPOS=BOXBOT
      CALL PLOT (XPOS,YPOS,IPEN)
      IPEN=2
C      NFRAMS=ISTART+NFRAMS-1
      DO 410 SCCNT=ISTART,NFRAMS
         XPOS=((SCCNT-ISTART)/XSCALE)+BOXL
         YPOS=EVENT(SCCNT,CHAN)
         YPOS=(YPOS*VSCALE)/(VFS/BOXHT)+YORG
         IF (YPOS.GT.BOXTOP) YPOS=BOXTOP
         IF (YPOS.LT.BOXBOT) YPOS=BOXBOT
         CALL PLOT(XPOS,YPOS,IPEN)
         IPEN=2
410   CONTINUE
      RETURN
      END
C
C
C
C SCALIT ... A SUB TO COMPUTE THE SCALE FACTORS FOR AUTO SCALE ETC
      SUBROUTINE SCALIT (ISTPOS,VSCALE,TRACE)
$INSERT SYSCOM>A$KEYS
$INSERT SYSCOM>KEYS.F
$INSERT FILDEF
$INSERT PLOTDEFS
      VTOT=0.
      LCOUNT=IWCNT-1
      DO 100 ICOUNT=0,LCOUNT
      VSAMP=EVENT(ICOUNT-ISTPOS,CHAN)
      IF (ICOUNT.GT.1) GO TO 20
      VMAX=VSAMP
      VMIN=VSAMP
20    IF (VSAMP.GT.VMAX) VMAX=VSAMP
      IF (VSAMP.LT.VMIN) VMIN=VSAMP
      VTOT=VTOT+VSAMP
100   CONTINUE
      VAVE=VTOT/IWCNT
      VPEAK1=VMAX-VMIN
      IF (VPEAK1.EQ.0) VPEAK1=.000001
      IF (VPEAK1.LT.0) VPEAK1=VPEAK1*-1
      VSCALE=.90*(VFS/NCHAN/VPEAK1)
      VPEAK2=(VMAX+VMIN)/2
      YOS=(VPEAK2*VSCALE)/(VFS/BOXHT)
      YORG=YORG-YOS
110   CONTINUE
      IF (CHAN.EQ.5.OR.CHAN.EQ.10.OR.CHAN.EQ.15.OR.CHAN.EQ.16
     & .OR.CHAN.EQ.17
     & VSCALE=VSCALE*.5
      TRACE (1,CHAN)=VSCALE    /* Scale factor
      TRACE (2,CHAN)=VAVE      /* Average voltage
      TRACE (3,CHAN)=VMAX      /* Most positive
      TRACE (4,CHAN)=VMIN      /* Most negative
      TRACE (5,CHAN)=YORG      /* Plot origin (y)
      TRACE (6,CHAN)=YOS       /* Offset factor
      TRACE(7,CHAN)=0.0        /* Left over from TRED et all
      RETURN
      END
C
```

```
C
C This subroutine converts a file name to a path name.
C
      SUBROUTINE MAKNAM(FILNAM,FNAME)
$INSERT SYSCOM>ASKEYS
$INSERT SYSCOM>KEYS.F
      INTEGER*2 I,ILTH,FILNAM(16),FNAME(32),K1
      INTEGER*2 K2,K3,K6,K7,FNAME(2Ø),PLTH
      DATA FNAME/'*>.RAWDATA>6778899ØØ1122334455667788990Ø'/
C Set length of dummy pathname
      PLTH=11
C Determine length of file name
      ILTH=NLEN$A(FILNAM,32)
C Move dummy pathname into output name buffer and pad remainder with blanks
      CALL MSTR$A (FNAME,PLTH,FNAME,64)
C Move file name into proper area of dummy pathname
      CALL MSUB$A (FILNAM,32,1,ILTH,FNAME,64,PLTH+1,64)
      RETURN
      END
C
C
      SUBROUTINE READIN (ICODE)
$INSERT SYSCOM>ASKEYS
$INSERT SYSCOM>KEYS.F
$INSERT FILDEF
$INSERT PLOTDEFS
      INTEGER*2 NUM(1ØØØØ)
      PDR=Ø
1Ø    CONTINUE
C Read in one frame of data
5Ø    CALL PRWFSS (KSREAD+KSPOSR,1,LOC(INBUF),256,INTL(Ø),IWORDS,ICODE)
C Make sure we really got something
      IF (ICODE.NE.Ø) GO TO 1ØØ
C Get the sample rate from the data
      FRAMES=FRAMES+1
      CALL GETSR
      CALL GETSNO
      SAMPER=1ØØØ./SRATE
C Now translate the data portion of the frame
      DO 7Ø I=1,63
      DO 6Ø J=1,3
C First get the sample into a useful form (signed integer).
C Now make it a floating point number between -1 and 1
      IF(J.EQ.3) EVENT(IWCNT,J)=FLOAT(AND(RS(INBUF(J,I),13),:1))*1.Ø
      OMEGA(IWCNT)=AND(RS(INBUF(J,I),13),:1)
C         EVENT(IWCNT,J)=FLOAT(2ØØ48-AND(RS(INBUF(J,I),1),:7777))/2Ø48.*2.Ø
CC        EVENT(IWCNT,J)=AND(RS(INBUF(J,I),1),:7777)
      EVENT(IWCNT,J)=FLOAT(2ØØ8-AND(RS(INBUF(J,I),1),:7777))/2Ø48.*2.5
C       EVENT(IWCNT,J)=FLOAT(2ØØ48-AND(INBUF(J,1),:7777))/2Ø48.*2.5
C       EVENT(IWCNT,J)=FLOAT(2ØØ8-AND(RS(INBUF(J,I),1),:7777))
C Test for PDR=1 type data
      IF (RS(INBUF(J,I),15).EQ.Ø) GO TO 6Ø
C This sample is PDR=1 data so gain ranging must be accounted for
      PDR=1
      K=Ø
      IF (AND(INBUF(J,I),1).NE.Ø) K=K+2
      IF (AND(INBUF(J,I),1ØØØ).NE.Ø) K=K+4
      EVENT(IWCNT,J)=EVENT(IWCNT,J)/2**K
       IF(J.EQ.2) NUM(IWCNT)=AND(INBUF(J,I),8192)
6Ø    CONTINUE
      IWCNT=IWCNT+1
7Ø    CONTINUE
      DO 8Ø I=1,4
8Ø    EVENT(IWCNT,I)=EVENT(IWCNT-1,I)
      IWCNT=IWCNT+1
C
      GO TO 5Ø
C
1ØØ   CALL CLOSSA(1)    /* CLOSE THE FILES
      ICODE=1     /* SET FOR EOF
      IWCNT=IWCNT-1
      NSAMP=IWCNT
C          WRITE(1,8787) (NUM(I),I=1,256)
8787       FORMAT(2X,'EVENTO NUMERO: ',8(2X,I5))
      RETURN
      END
```

```
C
C
C
C PSTAT... a subr. to print the trace parameters
      SUBROUTINE PSTAT (TRACE,STAT,ISTPOS,IFPOS)
$INSERT SYSCOM>ASKEYS
$INSERT SYSCOM>KEYS.F
$INSERT FILDEF
$INSERT PLOTDEFS
      INTEGER*2 CLINE1(4Ø),CLINE2(4Ø),NUMBER(1Ø)
        DATA CLINE1/
C       '1122334455667788990011223344556677889900
     & 'INSTRUMENT TYPE:  PDR-1.  INSTRUMENT SERI',
     & 'AL NUMBER:                                '/
      LENGTH=6Ø
C       CALL MSUBSA('INSTRUMENT TYPE:',16,1,16,CLINE1,LENGHT,1,LENGTH)
C If this is a DSA tape the instrument type must be changed
      IF (PDR.EQ.1) GO TO 1Ø
      CLINE1(9)='  D'
      CLINE1(1Ø)='SA'
C       CALL MSUBSA('INSTRUMENT SERIAL NUMBER:',25,1,25,CLINE1,LENGTH,
C     + 26,5Ø)
C Convert the instrument type to a string
1Ø    CALL CNVBSA(ASCII,CU,INTL(SERIAL),NUMBER,2Ø)
C Now left justify that string . . .
      CALL JSTRSA(ASLEFT,NUMBER,2Ø)
C and tack it onto the end of the comment line string.
      CALL MSUBSA(NUMBER,2Ø,1,2Ø,CLINE1,LENGTH,52,LENGTH)
C Now add a ',' to make it look pretty.
      CALL MSUBSA(',',1,1,1,CLINE1,LENGTH,1+NLENSA(CLINE1,LENGTH)
     +,LENGTH)
C Convert the sample rate into a left justified string and add it too.
      CALL ENCDSA(NUMBER,2Ø,2,DELE(S/MFED))
      CALL JSTRSA(ASLEFT,NUMBER,2Ø)
      CALL MSUBSA(NUMBER,2Ø,1,2Ø,CLINE1,LENGTH,3+NLENSA(CLINE1,LENGTH)
     +,LENGTH)
      CALL MSUBSA('SEC',3,1,3,CLINE1,LENGTH,2+NLENSA(CLINE1,LENGTH)
     +,LENGTH)
C Now send the line off to the plotter
      CALL SYMBOL (BOXL,BOXBOT-.05-CSIZE,CSIZE,CLINE1,Ø.,6Ø)
      TIME=IWORT*SAMPER    /* Compute length of event
      CALL ENCDSA(NUMBER,2Ø,2,DELE(TIME))  /* Convert to a string
      CALL JSTRSA(ASLEFT,NUMBER,2Ø)   /* Left justify that string
      CALL MSTRSA('SAMPLE PERIOD:',14,CLINE1,LENGTH)
      CALL MSUBSA(NUMBER,2Ø,1,2Ø,CLINE1,LENGTH,3-NLENSA(CLINE1,LENGTH),
     +LENGTH) /* Move it to the output line
      CALL MSUBSA('SECONDS LONG:',13,1,13,CLINE1,LENGTH
     +,2+NLENSA(CLINE1,LENGTH),LENGTH)
C Convert number of samples to a string and append it to the output line
      CALL CNVBSA(ASCII,CU,INTL(IWORT),NUMBER,2Ø)
      CALL JSTRSA(ASLEFT,NUMBER,2Ø)
      CALL MSUBSA(NUMBER,2Ø,1,2Ø,CLINE1,LENGTH,2+NLENSA(CLINE1,LENGTH)
     +,LENGTH)
      CALL MSUBSA('POINTS.',7,1,7,CLINE1,LENGTH
     +,2+NLENSA(CLINE1,LENGTH),LENGTH)
        CALL MSUBSA('FILE:',5,1,5,CLINE1,LENGTH,2+NLENSA(CLINE1,LENGTH)
     +   ,LENGTH)
      CALL MSUBSA(FILNAM,32,1,32,CLINE1,LENGTH,2+NLENSA(CLINE1,LENGTH)
     +,LENGTH)
C       CALL MSUBSA('TIME:',5,1,5,CLINE1,LENGTH,2+NLENSA(CLINE1,LENGTH)
C     +  ,LENGTH)
C Send this one to the plotter, but a little farther down.
      CALL SYMBOL (BOXL,BOXBOT-.11-2*CSIZE,CSIZE,CLINE1,Ø.,8Ø)
      RETURN
      END
      SUBROUTINE GETSNO
$INSERT FILDEF
$INSERT PLOTDEFS
      INTEGER*4 SERNR, BINPOS
      BINPOS=1
      SERIAL=Ø
      DO 1ØØ I=1,14
      SERNR=INTL(AND(INBUF(1,I+2),8192)/8192)
      SERNR=SERNR*BINPOS
      SERIAL=SERIAL+SERNR
      BINPOS=BINPOS*2
```

```
100  CONTINUE
     RETURN
     END
C
C
     SUBROUTINE GETSR
$INSERT FILDEF
$INSERT PLOTDEFS
     INTEGER*2 SR1, SR2
     SR1=AND(INBUF(1,1),8192)/8192
     SR2=AND(INBUF(1,2),8192)/8192
     SRATE=INTL(2*SR1+SR2)
     SRATE=2**SRATE
     SRATE=SRATE*50*1000
     RETURN
     END

C
C*****************************************************************
C
C                              SUBROUTINE
C
C     TITLE:NEWPAG
C
C     APPLICABLE PROGRAMS:ANY TEK. TERM PROGRAM
C
C     PURPOSE:ERASE TEK. SCREEN AND DELAY AS APPROP.
C
C     COMMENTS:DELAY MAY BE TUNED AS NECESSARY
C
C*****************************************************************
C
      SUBROUTINE NEWPAG
C     CLEARS THE PAGE, HOMES THE CURSOR.
C
      DATA NEW/:1150;4/
      CALL TROJA(NEW,2)
      CALL SLEEPS(INTL(1200))
      RETURN
      END

C
C*****************************************************************
C
C                              SUBROUTINE
C
C     TITLE:PLOTS
C
C     APPLICABLE PROGRAMS:ANY TEK PROGRAMS
C
C     PURPOSE:INIT PLOT ROUTINES
C
C     COMMENTS:CALL PRIOR TO PLOT CALLS AND AFTER PLOT CALLS
C
C*****************************************************************

      SUBROUTINE PLOTS(IDUM1,IDUM2,IDUM3)
      INTEGER*4 IXN,IYN,IXORG,IYORG,LASTX,LASTY
      COMMON /PLTERM/ ANGL,SNANG,CSANG,
     * XX,YY,FAC,IXN,IYN,IXORG,IYORG,LASTX,LASTY

      COMMON/LINSCM/ISIZE,SIZE
C
      IXORG=60
      IYORG=72
      IXN=0
      IYN =0
      IP=0
      FAC=1.
      CSANG=1.
      SNANG=0.
      SIZE=1.
      ISIZE=0
      CALL GMOD
      CALL NEWPAG
      RETURN
      END
```

```
C            SUBROUTINE PLOT
C      THIS PLOT INTERFACES TO THE TEXTRONIX 4010
C      IT SHOULD BE REASONABLY COMPATIBLE WITH
C      THE ZETA ROUTINES.
C      IT IS NOW PART OF TEKLB1.
C
C
      SUBROUTINE PLOT(X,Y,IPEN)
      INTEGER*4 IXN,IYN,IXORG,IYORG,LASTX,LASTY
      COMMON /PLTPRM/ ANGL,SNANG,CSANG,XX,YY,FAC,IXN,IYN,
     *  IXORG,IYORG,LASTX,LASTY
      COMMON /ZYYZ/ OLDX,OLDY
      INTEGER*4 IP,IPTXMN,IPTXMX
      COMMON /PSPLOT/IP,IPTXMN,IPTXMX
      PARAMETER NHORZ=1023
      PARAMETER NVERT=779
      DATA GS/'116635/
      IF (IPEN .EQ. 999)GOTO 285
      OLDX=X
      OLDY=Y
      X1=FAC*(X*CSANG+Y*SNANG)
      Y1=FAC*(Y*CSANG-X*SNANG)
      IF (IPEN .EQ. 0) GO TO 100
      IPA=MAX0(MIN0(IABS(IPEN),3),1)
      IF (IPA-2)101,102,103
  101 CONTINUE
  102 IX=X1*(0.95653+SIGN(.5,X1)
      IY=Y1*(0.95652+SIGN(.5,Y1)
      IXO=IX+1.4826*IXORG
      IYO=IY+1.2355*IYORG
      IF (IXO .LT. 0)IXO=0
      IF (IXO .GT. NHORZ)IXO=NHORZ
      IF (IYO .LT. 0)IYO=0
      IF (IYO .GT. NVERT)IYO=NVERT
      CALL PLTTEK(IXO,IYO)
      IXN=INTL(FLOAT(IX)*.674/860)
      IYN=INTL(FLOAT(IY)*.805/864)
      IF (IPEN .GT. 0) RETURN
      IXORG=IXN+IXORG
      IYORG=IYN+IYORG
CS    IF (IXORG.LT. 0) IXORG=0
CS    IF (IXORG .GT. NHORZ)IXORG=NHORZ
CS    IF (IYORG .LT. 0) IYORG=0
CS    IF (IYORG .GT. NVERT)IYORG=NVERT
      IXN=0
      IYN=0
      RETURN
  103 CALL TNOUA(GS,1)
      GO TO 101
  100 X=IXN/0.
      Y=IYN/77.
      RETURN
  285 CALL TNOUA(GS,1)        /* FINISHED PLOTTING.
      CALL PLTTEK(0,22)
      CALL ANMOD
      RETURN
      END
C
C
C**********************************************************************
C
C                              SUBROUTINE
C
C     TITLE:
C
C     APPLICABLE PROGRAMS:
C
C     PURPOSE:
C
C     COMMENTS:
C
C**********************************************************************
C
      SUBROUTINE PLTTEK(IX,IY)
C
C     THIS SUBROUTINE ACTUALLY SETS THE PROPER BITS AND SENDS THE
C     CHARACTERS TO THE 4010.
```

```
C
      INTEGER JOUT(2)
      DATA IXMASK/:12Ø3ØØ/,IYMASK/:12Ø34Ø/
      DATA IHMASK/:174Ø/,ILMASK/:37/
      JX=OR(LS(AND(IHMASK,IX),3),AND(ILMASK,IX))
      JOUT(2)=OR(JX,IYMASK)
      JY=OR(LS(AND(IHMASK,IY),3),AND(ILMASK,IY))
      JOUT(1)=OR(JY,IYMASK)
      CALL TNOJA(JOUT,4)
      RETURN
      END
      SUBROUTINE GMOD
C      TURNS ON THE 4Ø1Ø'S GRAPHIC DISPLAY MODE.
C
      DATA IGS/:116635/
      CALL TNOJA(IGS,1)
      RETURN
      END
C
C
C***********************************************************************
C
C                              SUBROUTINE
C
C     TITLE:
C
C     APPLICABLE PROGRAMS:
C
C     PURPOSE:
C
C     COMMENTS:
C
C***********************************************************************
C
      SUBROUTINE ANMOD
C     SETS ALPHANUMERIC MODE.
C
      DATA MOD/:117637/
      CALL TNOJA(MOD,1)
      RETURN
      END
      SUBROUTINE SYMBOL(X,Y,H1,A,T,N)
      INTEGER *2  A(1),AA,MESBUF(/8Ø)
      INTEGER*2 SC(1[illegible]),CC(111),C(381)
      INTEGER*2 C1(1Ø4),C2(96),C3(99),C4(82)
      EQUIVALENCE (C(1),C1(1)),(C(1Ø5),C2(1)),(C(2Ø1),C3(1)),
     1(C(3ØØ),C4(1))
      INTEGER*2 IA2(2),NCVT(197)
      INTEGER*4 IA4
      COMMON/PTXSYM/C,NCVT,SC
      EQUIVALENCE (IA4,IA2(1))
C
      DATA JIM/3/,XX/Ø./,YY/Ø./
      DATA XZ,YZ,OLDX,OLDY /Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø/
      DATA SC/
     18145,12649,6155,7657,766Ø,7166,7169,6663,
     18173,7673,14676,13657,6185,4159,6188,3151,3734/
      DATA CC/
     &Ø,8115,5242,Ø,8744,821Ø,2749,Ø,2749,875Ø,6754,
     &3757,13[illegible],1135Ø,3303,1,7231,8726,11ØØ3,11533,
     &14ØØ2,1Ø721,43[illegible],424Ø,4238,11111,5114,6134,2615,
     &5Ø3Ø,2Ø15,9ØØ3,35Ø6,8Ø11,13Ø2Ø,6Ø65,9596,11Ø41,6Ø28,
     &17Ø1Ø,1,1Ø4,11131,13131,3629,5374,3126,13137,15214,
     &1Ø1[illegible]1,13146,8ØØ,7546,7516,6Ø17,12598,6Ø91,5Ø6Ø,
     &52[illegible],75[illegible],3516,1Ø33,45[illegible],11ØØ1,756Ø,11ØØ2,1Ø58Ø,
     &12538,4[illegible]1,656Ø,3Ø78,5Ø79,5Ø76,5Ø74,8593,4Ø15,2577,
     &4Ø14,311Ø,2373,[illegible]774,6273,12282,12783,1Ø789,7794,9298,
     &76Ø3,14379,1231Ø,9319,5923,Ø,Ø,5328,1483Ø,6126,6126,
     &47Ø3,8837,5194,8841,11845,5379,5351,5876,5853,927Ø,2362,9765,
     &6259,17Ø64/
      DATA C1/
     32547,4637,17Ø6,11Ø,3Ø41,464Ø,2227,3616,
     3272Ø,1Ø3Ø,617,3746,44Ø1,4Ø,47Ø7,4Ø,
     37,47Ø7,434,617,3746,4534,1434,4341,
     33Ø1Ø,1Ø3,14Ø5,6Ø7,4746,111Ø,1,117Ø,
     32227,2Ø7Ø,232,4334,14Ø5,1646,3717,6Ø5,
```

```
31424,4341,3Ø1Ø,1Ø6,1737,4641,3ØØØ,737,
34645,34Ø4,3443,414Ø,3Ø31,417Ø,47,7Ø17,
316Ø6,717,2621,1Ø3Ø,2Ø27,1737,2621,3ØØØ,
342Ø2,373Ø,2Ø4Ø,7,11Ø,3Ø41,47Ø7,272Ø,
31133,2Ø25,725,47ØØ,7Ø4Ø,72Ø,474Ø,24ØØ,
3737,4645,34Ø4,3443,4Ø47,23Ø7,7Ø,4Ø25,
3347,4ØØ7,7,444,474Ø,24,1434,244Ø,
37Ø4,3443,413Ø,1ØØ1,617,3746,7Ø33,5344/
 DATA C2/
33313,4Ø6,1737,4641,3Ø1Ø,125,41Ø5,2345,
3123,2125,23Ø3,437Ø,45Ø1,7ØØ1,417Ø,3Ø1Ø,
33,4346,3717,6Ø3,434Ø,244,67Ø,141,
37Ø42,446,2414,1525,247Ø,2111,1722,211Ø,
31Ø21,3Ø1Ø,7Ø22,2434,4546,3717,643,314,
312Ø3,2224,4ØØ,4Ø44,2422,2414,3Ø1,1Ø2Ø,
34143,3424,2224,141,2422,42Ø2,222Ø,2422,
344ØØ,22Ø4,4Ø22,44,44Ø,2224,22Ø,4224,
3222Ø,24Ø2,4224,2244,447,4Ø,22,472,
34422,2Ø22,4433,13Ø4,1311,11,314Ø,3133,
32244,44Ø,22,1Ø21,5Ø24,641Ø,2262,2226,
32527,213,2Ø27,472Ø,22Ø2,3121,2615,3625/
 DATA C3/
32623,1333,141,7Ø43,37Ø,141Ø,7Ø3Ø,34Ø1,
32541,141,7Ø22,2Ø24,444,3111,2Ø5,1636,
34542,3224,2514,1322,3341,3Ø1Ø,1Ø2,3627,
31615,4Ø25,2717,257Ø,4547,3745,1Ø21,3Ø1Ø,
37Ø22,2722,2627,15Ø7,1715,716,117Ø,4736,
3314Ø,363,4442,6Ø44,47Ø,242,7Ø35,11ØØ,6ØØ7,
367Ø1,1ØØØ,2130,4757,6612,4253,5475,1514,3Ø54,
3112,224Ø,5Ø61,7ØØ3,1424,4252,631Ø,2Ø31,
33344,3537,2Ø16,5Ø4Ø,3132,2435,2748,56ØØ,
31412,2131,4244,4251,1115,545,3531,2Ø36,7Ø25,
34554,5241,2112,1425,7ØØ2,633Ø,357Ø,1424,
32231,4244,5411,547Ø,1424,4151,1512,7132/
 DATA C4/
33532,4152,551Ø,227Ø,1625,2351,5745,
33433,2212,3Ø4,1125,3433,4252,4636,2534,
34456,5241,3122,1334,1352,7Ø55,3524,
32231,5114,2534,127Ø,3445,547Ø,5Ø1Ø,
33417,5725,3534,1425,7Ø22,3221,2122,7ØØ3,
35316,2736,4756,1Ø36,3Ø22,427Ø,2646,7Ø37,
33Ø22,4227,2444,7Ø26,4Ø77,373Ø,1355,7Ø15,
35132,2Ø72,423Ø,2Ø36,7ØØ4,4525,1412,2141,52ØØ,
[illegible],343,4153,5247,3324,2615,4Ø3,[illegible],
[illegible],[illegible],6453,[illegible],[illegible],7ØØ,4[illegible],
[illegible],27[illegible],324,[illegible]/
 DATA NCVT/96,97,97,18,1ØØ,1Ø1,1Ø2,1Ø3,1Ø4,1Ø5,1Ø6,1Ø7,115,
*11Ø,23,114,116,111,12Ø,22,21,113,112,18,29,19,121,119,
*122,26,127,3Ø,2Ø,33,34,35,30,37,38,39,4Ø,41,42,43,44,45,46,
*47,48,49,5Ø,51,52,53,54,55,56,57,58,59,6Ø,61,62,63,64,65,66,
*67,68,69,7Ø,71,72,73,74,75,76,77,78,79,8Ø,81,82,83,84,85,86,87,88,
*89,9Ø,91,92,93,14,95,25,65,66,67,68,69,7Ø,71,72,73,74,75,76,77,78,
*79,8Ø,81,82,83,84,85,86,87,88,89,9Ø,123,124,125,126,23,31,27,117,
*118,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,
*65,66,67,68,69,7Ø,71,72,73,74,75,76,77,78,79,8Ø,81,82,83,84,85,
*86,87,88,89,9Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø/
 R=T*Ø.Ø1745329
 IF(H1.NE.Ø.)H=H
 D=ABS(H)/7.
 B=D*COS(R)
 S=D*SIN(R)
 B1=B
 S1=S
 XZ=X
 YZ=Y
 IF(X.EQ.999.)XZ=XX
 IF(Y.EQ.999.)YZ=YY
 OLDX=XZ
 OLDY=YZ
 XZS=XZ
 YZS=YZ
 KI=3
 HX2=7.
 HY2=7.
 NL=Ø
 ISC=Ø
```

```
      IF(N)1,40,40
    1 CONTINUE
C     IA2(1)=A(1)
C     IA2(2)=A(2)
C     AA=IABS(IA4)
      AA=IABS(A(1))
      NLT=0
      IF(N+1)10,25,25
10    KI=IABS(N)
      IF(KI-1)25,12,12
12    CALL PLOT(XZ,YZ,KI)
      KI=KI-1
25    CH=2.
      IF(AA.EQ.15)CH=3.
      D=ABS(H)/4.
      B=D*COS(R)
      S=D*SIN(R)
      ISC=1
      XZ=XZ+CH*(S-B)
      YZ=YZ-CH*(S+B)
      NCW=SC(AA+1)
      GOTO 75
30      IA=AA-16
      NCW=CC(IA)
      GOTO 75
40    NLT=N
      NNN=(N+3)/4
      NNNN=N
      IF(R.GT.0.)GOTO 42
      NNN=N
      NNNN=N*4
 42   CONTINUE
      DECODE(NNNN,100,A)MESBUF
100   FORMAT(80A1)
      DO 101 I=1,NNNN
      MESBUF(I)=RS(MESBUF(I),8)
      MESBUF(I)=MOD(MESBUF(I),128)
101   CONTINUE
      IF(F)44,200,50
44    DO 46 I=1,N
      IJ=I*4-3
      MESBUF(I)=MESBUF(IJ)
  46  CONTINUE
      GOTO 50
40    MESBUF(1)=A(1)
      NLT=1
50    IF(NL.EQ.NLT)GO TO 200
      NL=NL-1
C     NCC=MOD(MESBUF(NL),128)
      NCC=MESBUF(NL)
      IF(NCC.LE.196) GO TO 300
      NCC=32
 300  NCC=NCVT(NCC+1)
      IF(NCC-16)55,55,60
55    NCW=SC(NCC)
      GO TO 75
60    NCC=NCC-16
      NCW=CC(NCC)
C     IF(NCW.EQ.0)GO TO 210
75    NM=NCW/1000
      NC=MOD(NCW,1000)
      LH=0
      IF(NC.LT.500)GOTO 80
      NC=NC-500
      LH=1
80    NMM=0
      KJ=0
90    IF(NMM.EQ.NM)GOTO 150
      NMM=NMM+1
      K=2
      IF(LH.EQ.1)GOTO 100
      J=C(NC)/100
      GOTO 110
100   NNNNC=C(NC)
      J=MOD(NNNNC,100)
110   JX=J/10
```

```
      IF(JX.LT.7)GO TO 112
      KJ=1
      GOTO 115
112   JY=MOD(J,10)
      XC=XZ+FLOAT(JX)*B-FLOAT(JY)*S
      YC=YZ+FLOAT(JX)*S+FLOAT(JY)*B
      K=2
      IF(NMM.EQ.1)K=KI
      IF(PJ.EQ.1)K=3
      KJ=0
      CALL PLOT(XC,YC,K)
115   IF(LH.EQ.0)GOTO 120
      LH=0
      NC=NC+1
      GOTO 90
120   LH=1
      GOTO 90
150   OLDX=OLDX+HX2*B1
      OLDY=OLDY+HY2*S1
      XZ=OLDX
      YZ=OLDY
      GOTO 50
200   IF(ISC)220,220,210
210   XZ=XZS
      YZ=YZS
220   CALL PLOT(XZ,YZ,JIM)
C*****************************************

C
C           PUT CODE TO SET XX AND YY (FOR 999 MODE) HERE.

      XX=XZ
      YY=YZ
      RETURN
      END
      SUBROUTINE NUMBER (X,Y,HT,FP,ANG,NDEC)
C
C           NUMBER ROUTINE.

      INTEGER ILAB(80)
      DIMENSION FP(1)
      INTEGER*2 FORM1(5),FORM2(4),NC,ND
      INTEGER*2 M(1),N(1),MN(1)
      DATA FORM1/'(F','M','.','M',')'/
      DATA FORM2/'(F','N','.','0',')'/
C
C           MAKE THE FORMAT STATEMENT, THEN USE IT TO ENCODE.

      NC=1
      IF (ABS(FP(1)).LE.0.0) GO TO 10
      NC=MAX0(1,INT(ALOG10(ABS(FP(1))+0.5))+1)
   10 CONTINUE
C
C           NC IS THE NO. OF DIGITS BEFORE THE DECIMAL POINT.

      IF (FP(1).LT.0) NC=NC+1
      ND=IABS(NDEC)
      IF (NDEC.LT.0) GO TO 100
C
C           FLOATING POINT FORMAT.

      ENCODE (2,2,M) ND
    2 FORMAT (I2)
      FORM1(4)=M(1)
      NCD=NC+ND+1
      ENCODE (2,2,MN) NCD
      FORM1(2)=MN(1)
      ENCODE (NCD,FORM1,ILAB) FP(1)
      GO TO 200
C
C           INTEGER FORMAT.

  100 CONTINUE
      NCD=ND+NC
      ENCODE (2,2,N) NCD
      FORM2(2)=N(1)
      NCD=NCD-1
      ENCODE (NCD,FORM2,ILAB) FP(1)
C
C           CALL SYMBOL WITH THE HOLLERITH ARRAY.
C
  200 CONTINUE
      CALL SYMBOL (X,Y,HT,ILAB,ANG,NCD)
      RETURN
      END
```

PROGRAMA LINEAL.FTN

APENDICE 3

```
C       PROGRAMA QUE SUSTITUYE VALORES CEROS DE
C       SINCRONIA DE ARCHIVOS LEIDOS CON #DSP
C       POR VALORES DE ACELERACION PARA DEJARLOS
C       COMO ARCHIVOS DE ENTRADA PARA #SELECT.

        SUBROUTINE MAIN
        INTEGER*2 FINAME(16),OFILE(32),FNAME(32)
        INTEGER*2 EVENT(131072,3)
        INTEGER*2 INBUF(4,64),KOUT(3)
        INTEGER*2 NSAMP,IWCNT,IFRAM,LINEAL
$INSERT SYSCOM>A$KEYS
$INSERT SYSCOM>KEYS.F
        COMMON/COMBUF/INBUF,EVENT
        COMMON/SAM/IWCNT,NSAMP
        CALL TONL
        WRITE(1,5)
5       FORMAT(10X,'PROGRAMA PARA QUITAR LOS CEROS DE SINCRONIA')
        WRITE(1,7)
7       FORMAT(15X,'DE ARCHIVOS LEIDOS CON #DSP.')
        CALL TONL
        CALL TONL
10      WRITE(1,1000)
1000    FORMAT('DAME EL NOMBRE DEL ARCHIVO A LEER :')
        READ(1,1010)FINAME
1010    FORMAT(16A2)
        IF (NLEN$A(FINAME,32).EQ.0) GO TO 10
        CALL MAKNAM(FINAME,FNAME)
        IF(.NOT. OPEN$A(A$READ,FNAME,32,1)) GO TO 10000
        FRAMES=0                              /* Reset frame counter
        IWCNT=1                               /* Reset sample counter
        CALL READIN(ICODE)          /* Get the file here
        CALL TONL
        IF (ICODE.NE.1) GO TO 10000
        NCHAN=3
        NSAMP=IWCNT
        DO 700 L=1,IWCNT
        IF(MOD(L,64).NE.0) GO TO 700
        DO 700 I=1,3
         LINEAL=(EVENT(IWCNT+1,I)-EVENT(IWCNT-1,I))/2+EVENT(IWCNT-1,I)
         EVENT(IWCNT,I)=LINEAL
700     CONTINUE
        CALL TONL
        WRITE(1,900) NSAMP
900     FORMAT(2X,'NUMERO DE MUESTRAS LEIDAS:',I8)
        CALL TONL
        CALL TONL
80      WRITE(1,85)
85      FORMAT('DAME EL NOMBRE DEL ARCHIVO DE SALIDA:')
        READ(1,1010)(OFILE(I),I=1,16)
        IF(.NOT. OPNV$A(A$WRIT,OFILE,16,1,A$VNEW,0,0)) GO TO 80
        IFRAM=0
        DO 600 K=1,IWCNT
        KOUT(1)=EVENT(K,1)
        KOUT(2)=EVENT(K,2)
        KOUT(3)=EVENT(K,3)
        CALL PRWF$$(K$WRIT,1,LOC(KOUT),3,
     +                  INTL(0),IRNW,ICODE)
        IFRAM=IFRAM+1
        IF(ICODE.NE.0) WRITE(1,500) OFILE,ICODE,IFRAM
500     FORMAT(2X,'ERROR AL ESCRIBIR EN EL ARCHIVO :',16A2,2X,
     +                  'ICODE=',I5,'NUMERO DE MUESTRAS: ',I8)
        IF(ICODE.NE.0) GO TO 9000
600     CONTINUE
        CALL TONL
        WRITE(1,550) IFRAM
550     FORMAT(2X,'NUMERO DE MUESTRAS ESCRITAS:  ',I8)
C
9000    CALL CLOS$A(1)
10000   CALL EXIT
        GO TO 10
        END
C
```

```
C
      SUBROUTINE READIN (ICODE)
      INTEGER*2 INBUF(4,64),EVENT(131072,3)
$INSERT SYSCOM>A$KEYS
$INSERT SYSCOM>KEYS.F
      COMMON/COMBUF/INBUF,EVENT
      COMMON/SAM/IWCNT,NSAMP
      PDR=0
10    CONTINUE
C Read in one frame of data
50    CALL PRWF$$ (K$READ+K$POSR,1,LOC(INBUF),256,INTL(0),IWORDS,ICODE)
C Make sure we really got something
      IF (ICODE.NE.0) GO TO 100
C Get the sample rate from the data
      FRAMES=FRAMES+1
C      CALL GETSR
C      SAMPER=1000./SRATE
C Now translate the data portion of the frame
      DO 70 I=1,63
      DO 60 J=1,3
C First get the sample into a useful form (signed integer).
C Now make it a floating point number between -1 and 1
C       EVENT(IWCNT,J)= AND(RS(INBUF(J,I),1),:7777)
C        EVENT(IWCNT,J)= AND(INBUF(J,I),:177777)
       EVENT(IWCNT,J)= AND(RS(INBUF(J,I),1),:177777)
C Test for PDR-1 type data
      IF (RS(INBUF(J,I),15).EQ.0) GO TO 60
C This sample is PDR-1 data so gain ranging must be accounted for
      PDR=1
      K=0
      IF (AND(INBUF(J,I),1).NE.0) K=K+2
      IF (AND(INBUF(J,I),16384).NE.0) K=K+4
      EVENT(IWCNT,J)=EVENT(IWCNT,J)/2**K
60    CONTINUE
      IWCNT=IWCNT+1
70    CONTINUE
      DO 80 I=1,3
80    EVENT(IWCNT,I)=EVENT(IWCNT-1,I)
      IWCNT=IWCNT+1
C
      GO TO 50
C
100   CALL CLOS$A(1)    /* CLOSE THE FILES
      ICODE=1     /* SET FOR EOF
      IWCNT=IWCNT-1
      NSAMP=IWCNT
      RETURN
      END
C
       SUBROUTINE MAKNAM(FINAME,FNAME)
$INSERT SYSCOM>A$KEYS
$INSERT SYSCOM>KEYS.F
       INTEGER*2 I,ILTH,FINAME(16),FNAME(32),K1
       INTEGER*2 K2,K3,K6,K7,PNAME(20),PLTH
       DATA PNAME/'*>.RAWDATA>6778899001122334455667788990 0'/
       PLTH=11
       ILTH=NLEN$A(FINAME,32)
       CALL MSTR$A (PNAME,PLTH,FNAME,64)
       CALL MSUB$A (FINAME,32,1,ILTH,FNAME,64,PLTH+1,64)
       RETURN
       END
```