UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y CONSTRUCCION DE UN SECUENCIADOR MIDI BASICO

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

JORGE ARTURO RAMIREZ ARAMBURO

DIRECTOR DE TESIS:
ING. JOSE ANTONIO ARREDONDO GARZA

MEXICO D. F. 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CAPITULO I
(INTRODUCCION)

UNA PEQUENA HISTORIA DE SINTETIZADORES I-1
QUE ES UN SECUENCIADOR MIDI I-4

CAPITULO II
(MENSAJES MIDI)

QUE ES EL MIDI II-1
CANALES MIDI II-3
MODOS DE RECEPCION II-4
CLASIFICACION DE LOS MENSAJES MIDI II-7
MENSAJES DE CANAL II-8
MENSAJES DE VOZ II-8
INICIO DE NOTA (NOTE ON) II-8
FIN DE NOTA (NOTE OFF) II-8
PRESION DE TECLA (POLYPHONIC KEY PRESSURE) II-9
PRESION DE CANAL (CHANNEL PRESSURE) II-9
CAMBIO DE PROGRAMA (PROGRAM CHANGE) II-10
CAMBIO DE CONTROL (CONTROL CHANGE) II-10
CAMBIO DE AFINACION (PITCH BEND CHANGE) II-11
MENSJES DE MODO II-12
CONTROL LOCAL (LOCAL CONTROL) II-12
APAGAR TODAS LAS NOTAS (ALL NOTES OFF) II-12
MODO OMNI DESACTIVADO (OMNI MODE OFF) II-12
MODO OMNI ACTIVADO (OMNI MODE ON) II-13
MODO MONO ACTIVADO (MONO MODE ON) II-13
MODO POLY ACTIVADO (POLY MODE ON) II-13
MENSAJES DEL SISTEMA II-13
MENSAJES DE TIEMPO REAL II-14
RELOJ (TIMING CLOCK) II-14
INICIO (START) II-15
DETENERSE (STOP) II-15
CONTINUAR (CONTINUE) II-15
CENSADO DE ACTIVIDAD (ACTIVE SENSING) II-16
REINICIO DEL SISTEMA (SYSTEM RESET) II-16
MENSAJES COMUNES DEL SISTEMA II-16
SELECCION DE CANCION (SONG SELECT) II-16
POSICION DEL PUNTERO DE CANCION (SONG POSITION PIONTER) II-17
SOLICITUD DE AFINACION (TUNE REQUEST) II-17
EOX (END OF EXCLUSIVE) II-17
MENSAJES DE SISTEMA EXCLUSIVO II-17
TRANSMITIENDO EVENTOS SIMULTANEOS II-18
CORRIMIENTO DEL ESTADO (RUNNING STATUS) II-18
TIPOS DE CONEXIONES II-19
CONEXION EN SERIE (DAISY CHAINING) II-19
RED EN ESTRELLA II-20
RED EN ANILLO II-21

CAPITULO III
(DESCRIPCION DEL HARDWARE)

CIRCUITO DE RELOJ	III-1
MEMORIA ROM	III-2
DISPLAY	III-4
TECLADO	III-8
PERIFERICOS	III-13
INTERRUPCIONES	III-17
MEMORIA RAM	III-19

CAPITULO IV
(DESCRIPCION DEL SOFTWARE)

DESCRIPCION GENERAL	IV-1
INICIALIZACION	IV-2
MENU PRINCIPAL	IV-4
GRABACION	IV-5
REPRODUCCION	IV-7
ESCRITURA A MDR	IV-8
LECTURA DE MDR	IV-10
AJUSTE DE TIEMPO	IV-12
APAGADO DE TODAS LAS NOTAS	IV-13
SUBRUTINAS	IV-14
EXCEPCIONES	IV-15
REFRESCO DEL DISPLAY	IV-16
MENSAJES	IV-17
VERSION COMPILADA DEL SOFTWARE	IV-19

CONCLUSIONES

APENDICE A
(TABLAS DE ESPECIFICACION MIDI 1.0)

TABLA I	(RESUMEN DE BYTES DE ESTATUS)	A-1
TABLA II	(MENSAJES DE VOZ)	A-2
TABLA III	(NUMERO DE CONTROLES)	A-3
TABLA IIIA	(NUMERO DE PARAMETROS REGISTRADOS)	A-4
TABLA IV	(MENSAJES DE MODO)	A-4
TABLA V	(MENSAJES COMUNES)	A-5
TABLA VI	(MENSAJES DE TIEMPO REAL)	A-5
TABLA VII	(MENSAJES DE SISTEMA EXCLUSIVO)	A-6
TABLA VIII	(NUMEROS DE IDENTIFICACION DE SYSEX)	A-7
TABLA IX	(NUMEROS DE IDENTIFICACION DE FABRICANTES)	A-8

APENDICE B
(DIAGRAMA)

BIBLIOGRAFIA

CAPITULO I

UNA PEQUEÑA HISTORIA DE LOS SINTETIZADORES

Los sintetizadores aparecen prácticamente al mismo tiempo que las bocinas electrónicas. El primer instrumento que produce música a través de un bocina sin la necesidad de pregrabar el sonido, aparece en 1906. En los 20's los avances en la tecnología permiten producir dos sintetizadores populares, el Theremin y el Ondes Martenot, y además diversos instrumentos experimentales. Compositores de esa epoca como Edgard Varese, Darius Milhaud, Henry Cowell y Olivier Messiaen, empiezan a usar estos instrumentos en su música. Muchos compositores de música para películas intrigados por la espectacularidad de la música sintetizada usan el Theremin y el Ondes Martenot en las partituras para cine.

Desde los 30's hasta los 50's los sintetizadores continúan desarrollandose. La mayor parte de estos nuevos instrumentos se encuentran en estudios de grabación. Muchos de esos dispositivos experimentales fueron desarrollados por compañías como RCA y los laboratorios Bell. El público en general desconocía los sintetizadores, los escuchaba ocacionalmente en películas. En 1968 un joven compositor llamado Walter Carlos (ahora Wendy Carlos) grabó un álbum que cambiaría todo esto.

Switched On Bach, un álbum con música de Johann Sebastian Bach tocada en un sintetizador Moog, esto fue un tremendo suceso. S.O.B. (así llamado por los aficionados) salía al aire en las estaciones de radio alrededor del mundo y daba la oportunidad al público de escuchar el sonido de un sintetizador. Para muchos, esto inicia un gran interes por los nuevos instrumentos. No mucho después de esto, los sintetizadores dejan de hacer efectos raros de fondo en el cine y empiezan a ejecutar melodías.

Por los 70's, los sintetizadores se han hecho un camino en las bandas de rock. Los tecladistas tienen en frente un banco de switches y jacks, muchos cables se enmarañan y parece un spaghetti color negro. Con los músicos de rock como clientes potenciales, nuevos fabricantes empiezan a aparecer, el precio empieza a bajar y los sintetizadores se mueven de las escuelas y estudios de grabación a ensayos de garage en todo el mundo.

El primer sintetizador comercial no era muy confiable, debido a que los componentes electrónicos internos eran muy sensibles a vibraciones y cambios de temperatura. La afinación de un sintetizador era usualmente inestable y difícil de controlar, y como los instrumentos se calentaban mucho existían cambios impredecibles de afinación.

Con la sustitución de componentes discretos por circuitos integrados, los fabricantes pueden hacer sintetizadores más confiables y bajar el costo al mismo tiempo. Poner un microprocesador a un sintetizador antes de esto era prácticamente imposible. El poder de un microprocesador permite la lectura del teclado, para controlar a los circuitos que generan el sonido y ayudar al músico a crear sonidos sintéticos sin la necesidad de usar una maraña de cables y switches.

Como los sintetizadores bajan de precio a un nivel aceptable, muchos músicos empiezan a tener varios. Generalmente cada sintetizador era una sola unidad, con su propio teclado, su propio panel de controles y su propio rango de sonidos. Un músico tenía un sonido llamativo del sintetizador para música de rock, y otro suave para otro tipo de música, pero en general, los sintetizadores tenían calidad solo para un estilo de sonido. Para tener todos estos sonidos durante

un concierto, los tecladistas debian tener todos sus sintetizadores en escena y saltar de teclado a teclado para obtener un nuevo sonido.

Muchos sintetizadores no se escuchaban bien por si solos como una guitarra o un piano. Los músicos usaban todos los trucos para hacer esto posible; algunos propietarios de sintetizadores encuentran que tocando dos sintetizadores al unisono se puede lograr este sonido. Si dos sintetizadores se escuchan bien, ? por que no usar tres o cuatro o más para producir un mejor sonido? en conclusión para crear un sonido suficientemente rico en armónicas como para escucharse bien por si solo, se necesitaba un banco de sintetizadores.

El problema de tocar varios sintetizadores al unisono es que los humanos solo tenemos dos manos. La solución era obvia, tener un teclado maestro y desde el tocar todos los demás sintetizadores. Sin embargo, esta no era una solución particularmente fácil de implementar. Cada sintetizador tenia su propio sistema, el teclado en un sintetizador controlaba su propio circuito de generación de sonido.

Un tecladista rico podia darse el lujo de contratar un técnico con conocimientos para realambrar los teclados de los sintetizadores, y lograr que trabajaran todos juntos. Otros tecladistas desarrollaban técnicas para tocar sus sintetizadores como instrumentos separados. Como mucha gente deseaba que sus sintetizadores trabajaran juntos, los departamentos de servicio de los fabricantes, recibian muchas llamadas preguntando por datos técnicos.

En 1981, tres hombres que trabajaban con fabricantes de sintetizadores -- Dave Smith de Sequential Circuits, I. Kakehashi de la corporación Roland, y Tom Oberheim de Oberheim Electronics, discuten la posibilidad de crear un estandar para el control de sintetizadores que las diferentes compañias pudieran adoptar sin dificultad. Después de esto, Dave Smith toma tiempo para escribir una propuesta para el estandar, llamado Universal Sinthesizer Interface (USI). En noviembre la propuesta es hace pública en una reunion de la Sociedad de Ingenieros de Audio.

Al inicio de 1982, se organiza una junta de la National Association of Music Merchants (NAMM). Las principales compañias representadas en esta junta eran Sequential Circuits, Roland, Oberheim, Yamaha, Korg y Kawai. La mayor parte de estas compañias, tenian su propio estandar de comunicación entre instrumentos de su linea de productos; un compromiso tenia que ser hecho y además, el estandar especificado por USI tenia que ser simple y menos costoso que algunos de los estandares en uso, también, el USI deberia trabajar con sintetizadores de diferentes caracteristicas y poder darse el lujo de implementarse.

Con algunas de las especificaciones del USI forjadas, las compañias de la NAMM refinarian el interface. La idea original tenia dos simples objetivos: permitir que el teclado de un sintetizador tocara notas usando el generador de sonido de otro sintetizador, y sincronizar el playback de equipos que usaban secuenciador. Los ingenieros Japoneses en comunicación con los ingenieros de Sequential Circuits, dieron un paso al futuro e incluyeron en el USI códigos que permitian otros controles como ruedas de afinación (pitch wheels), interruptores de pie (foot Switches) y ruedesillas de modulación (modulation wheels) para el control de generadores de sonido remotos.

Cuando las compañias de NAMM se reunieron por segunda vez en junio de 1982, los esfuerzos de las diferentes compañias se combinaron, y el resultado de este trabajo es lo que se conoce ahora

como MIDI. El termino USI fue cambiado por MIDI: Musical Instrument Digital Interface que hoy es el estandar para la transmisión de información de eventos musicales.

Con el estandar propuesto, las compañías empezaron a incorporar el equipo necesario y los programas de microprocesadores en sus nuevos sintetizadores, permitiendo que los nuevos instrumentos se comunicaran con otros equipos a través del MIDI. Para la siguiente reunion de la NAMM en enero de 1983, una gran cantidad de nuevos sintetizadores equipados con MIDI fueron puestos a prueba. En uno de los primeros intentos, un sintetizador Sequential Circuits Prophet 600 fue conectado a un sintetizador Roland JP-6. Para el júbilo de los espectadores, tocar la tecla de un sintetizador producía una nota en el otro. El estandar MIDI había funcionado.

El MIDI no surgió sin sus problemas. Muchas secciones del estandar no fueron muy claras y se dejaron a libre interpretación. Los diferentes fabricantes, a menudo usaban estandares internos completamente diferentes como el pitch bend, resultando una confusión para el MIDI. Para aclarar la confusión, una completa especificación del MIDI fue publicada en Agosto de 1983 como MIDI 1.0 . El International MIDI User's Group (IMUG) fue formado para ayudar a distribuir las especificaciones a fabricantes y músicos, este grupo pronto se convertiría en the International MIDI Association (IMA). En Japón, las compañías de sintetizadores formaron su propio grupo de MIDI, llamado the JMSC (Japaese MIDI Standard Committee), para solidificar el estandar ahí.

Algunas discusiones surgieron por las limitaciones del MIDI y hubo intentos de la IMA para cambiar el estandar MIDI. Algunas personas no estaban contentas con la baja velocidad con la cual el MIDI transmite datos entre máquinas; otros no estaban contentos con la capacidad de los cartuchos de datos. Con el proposito de que no se destruyera el estandar o se cambiara hasta un punto en que los fabricantes no lo pudieran soportar, Jim Smerdel de Yamaha, junto con los representantes de otros fabricantes de sintetizadores, inician la MIDI Manufacturers Association (MMA) para mantener la definición del MIDI en manos de los fabricantes. En el otoño de 1984, ellos imprimieron las especificaciones completas del MIDI explicando con detalle todos los aspectos del estandar, para ayudar a eliminar cualquier confusión.

La JMSC junto con la MMA ahora controlan la definición del MIDI. Si existiera algún cambio o futura definición ambos la JMSC y la MMA deben agregarla. Aunque ellos reconocen las limitaciones del MIDI, la mayor parte de sus miembros aceptan la necesidad de que el MIDI exista en cualquier equipo de todo el rango de precios.

Desde 1984, el MIDI ha sido establecido en casi todas las marcas de la industria musical. Los usuarios encontraron lo conveniente que era el MIDI, y empezaron a demandarlo en otros instrumentos también. Los fabricantes que inicialmente no incluían MIDI en sus instrumentos, comenzaron a incluirlo al darse cuenta de que la falta de puertos MIDI se reflejaba en las ventas.

Los usuarios de computadoras rápidamente se dieron cuenta de la presencia del MIDI: Un interface digital enviando datos en forma perfecta para usarse con una computadora casera. Las compañías de software, principalmente Passport Designs, hicieron el camino con software para controlar sintetizadores desde una computadora y ayudar a los propietarios de los sintetizadores a pensar cosas imposibles para sus dos manos. El MIDI no solo se extiende a la industria de las

computadoras; los fabricantes de accesorios para sintetizadores empiezan a incluir MIDI en sus equipos como en procesadores de señal y amplificadores.

Los músicos que usan instrumentos musicales equipados con MIDI, pueden ahora construir su propio sistema musical y escoger componentes de los fabricantes que ofrecen la mejor calidad y precio.
El control computarizado sobre los instrumentos musicales extiende las posibilidades de la música y marca nuevas direcciones.

? QUE ES UN SECUENCIADOR MIDI ?

Un secuenciador es como un procesador de palabras para música. Es un dispositivo basado en un microprocesador que acepta como entradas los datos transmitidos por cualquier dispositivo capaz de transmitir por un puerto MIDI. Una vez en la máquina, los datos pueden ser alterados de diversas formas, tocados, y almacenados en alguna forma como en un floppy disk, en un cassette etc.

Un secuenciador es un equivalente electrónico de una pianola controlada con un rollo de papel perforado. En vez de un rollo de papel, el secuenciador utiliza un memoria RAM. En vez del mecanismo que la pianola utiliza para mover los martillos que golpean las cuerdas, el secuenciador envía datos a través de un puerto MIDI para que otro dispositivo (un sintetizador o un generador de tonos) genere los sonidos correspondientes.

El secuenciador no puede producir por si solo ningún sonido (a excepción de un pequeño click que se utiliza para que el músico se sincronice con el secuenciador) y tiene que utilizar modulos generadores que ofrecen un amplio espectro de sonidos. La captura de datos en un secuenciador se lleva acabo a través de un instrumento equipado con MIDI como una teclado, una guitarra o una flauta, permitiendo que un músico con experiencia capture una pieza musical en muy poco tiempo.

La música producida por un secuenciador no es como pudiera pensarse algo mecanico, repetitivo y sin sentimiento; esto se debe principalmente a que la mayor parte de los teclados, envían información como la fuerza con que se toca una nota (Key Velocity), la presión que se ejerce una vez tocada la nota (After touch), cambio de afinación , volumen , controles de sosten etc.

En el aspecto educativo, el secuenciador sin duda ha sido un gran avance. Si bien un secuenciador no es absolutamente indispensable para un estudiante de música, si brinda una gran ayuda en el desarrollo de capacidades como la composición o la orquestación. Un estudiante puede desarrollar piezas que nunca podría tocar por si solo, debido a la cantidad de instrumentos utilizados, corregirlas una y otra vez, sin la necesidad de reunirse con otros músicos para interpretarlas. Es cierto que hay personas que pueden realizar estas actividades, sin la necesidad de escuchar todo lo que han escrito, pero esto no es el caso general, y el secuenciador es un dispositivo que mejora considerablemente, algunos aspectos de los músicos que lo utilizan. Por otro lado, el músico no se ve limitado a elaborar solo aquellas composiciones o arreglos que tengan niveles de ejecución que puedan ser ejecutados por el mismo.

En el campo comercial, los secuenciadores tienen una amplia

demanda, debido a la gran cantidad de posibilidades que brindan. Una pieza se puede editar hasta lograr una perfección, difícilmente lograda por un ser humano, mejorando el efecto final y ahorrando muchas horas de estudio. Un secuenciador puede ser fácilmente sincronizado a una cinta de video por medio de señales dedicadas a esto, y lograr un perfecto acople entre la música y la imagen. Además de estas capacidades, los secuenciadores poseen una gran cantidad de características, que permiten reducir tiempo y dinero en la producción musical.

La calidad del sonido lograda con generadores actuales, permite que un secuenciador sea utilizado para producir música de una manera profesional. Además, el hecho de que un compositor pueda grabar todos los instrumentos sin ayuda de otros músicos, reduce el costo y aumenta la calidad de la producción.

Por estas y otras razones, los secuenciadores han tenido una gran demanda en todo el mundo, esto ha hecho que los fabricantes dirijan su mirada hacia ellos, y cada vez se pueden conseguir secuenciadores mejores a precios razonables. Sin duda los secuenciadores han sido uno de los más grandes avances en la música de los últimos tiempos.

CAPITULO II

La palabra "MIDI" está formada por las iniciales de "Musical Instrument Data interface". Es un estandar que los fabricantes de instrumentos musicales electrónicos, están de acuerdo en sostener, son una serie de especificaciones que se usan en la construcción de sus instrumentos musicales, así, los instrumentos de un fabricante, pueden comunicar sin dificultad, información musical, a instrumentos de otros fabricantes.

Un interface, tiene dos diferentes aspectos: el hardware y el software. El hardware, especifica la conexión física entre los instrumentos musicales. Este estipula un cierto tipo de conectores, que tipo de cable debe ser usado, especifica los detalles de las señales eléctricas que serán transmitidas a través de los cables MIDI : Las señales deben ser enviadas usando un formato estandar, voltaje estandar y una velocidad estandar.

El software de la interface, establece todo lo referente a la manera de codificar la información. Esto significa que dos instrumentos comunicandose a través del estandar MIDI, envían la información como una serie de números sobre el cable MIDI que los conecta. El estandar MIDI especifica que los números enviados como datos serán transmitidos en grupos llamados Mensajes MIDI. Cada mensaje MIDI comunica un evento musical entre máquinas. Algunos eventos musicales son acciones usuales que un ejecutante realiza mientras toca un instrumento- Acciones como presionar una tecla, mover controles deslizables, cambiar switches, y ajustar pedales.

Casi cualquier instrumento musical puede ser un dispositivo MIDI, pero todos las dispositivos MIDI, tienen un requerimiento común: deben tener un microprocesador, que pueda enviar, recibir y reaccionar a mensajes MIDI.

Los dispositivos MIDI más comúnes que se fabrican actualmente, son los sintetizadores; ya que casi todos los sintetizadores usan un microprocesador para la generación interna de sonido y para leer la actividad en el teclado del sintetizador, no es muy difícil para los fabricantes, agregar puertos MIDI y programar al sintetizador para que entienda códigos MIDI. Las máquinas de percusión (instrumentos que sintetizan sonidos de percusión y los reproducen usando ritmos almacenados) también usan microprocesadores y son fáciles de convertir en dispositivos MIDI.

Muchos instrumentos musicales electrónicos son normalmente construidos sin microprocesadores. Uno de estos es la guitarra eléctrica, la cual a menudo usa fonocaptores electromagneticos directamente bajo las cuerdas de la guitarra y sensan la vibración de la cuerda. Los fonocaptores convierten la vibración de la cuerda en voltaje que las guitarras pueden enviar directamente a un amplificador para hacer sonar una bocina. Estos fonocaptores no necesitan un microprocesador para detectar lo que esta haciendo la cuerda, por esto no se incluyen microprocesadores en una guitarra . Para transformar una guitarra en un dispositivo MIDI, el fabricante agrega un microprocesador que puede leer el voltaje proveniente de los fonocaptores y entonces puede enviar y recibir mensajes a través de un puerto MIDI.

Los instrumentos acusticos, presentan un desafio diferente, no son electricos en absoluto, y por esto no pueden llevar microprocesadores. Para transformar un instrumento acustico como un saxofon, una flauta o cualquier voz en un instrumento MIDI, el ejecutante o cantante usa un microfono (o un dispositivo similar) para recolectar las notas y transmitirlas a una dispositivo (pitch to

MIDI) con un microprocesador. El microprocesador lee las notas y envia el correspondiente mensaje sobre el puerto MIDI.

Los instrumentos musicales, no son los únicos instrumentos que pueden tener MIDI. Las computadoras pueden también ser transformadas en dispositivos MIDI. Ya que todas las computadoras tienen microprocesadores, todo lo que se necesita para transformar una computadora, es agregar un cierto tipo de hardware (un dispositivo que agregue puertos MIDI a la computadora) y algún software que diga a la computadora como transmitir y recibir mensajes MIDI.

La lista de dispositivos MIDI, incluye dispositivos que no serian normalmente asociados con instrumentos musicales. Por ejemplo, un panel de luces con puertos MIDI y un microprocesador que pueda responder a la información MIDI recibida, para cambiar el color y la intensidad de la luz. Un simple amplificador, puede ser un dispositivo MIDI si tiene un microprocesador para controlar el volumen y efectos especiales como cambios en la ecualización.

El sistema MIDI usa mensajes MIDI como una especie de código musical. Cada conjunto de números en un mensaje, ayuda a definir un evento musical. Por ejemplo, una serie de números en un mensaje MIDI, transmite el inicio de una nota con un cierto volumen y afinación, y un conjunto diferente transmite el fin de una nota. El estandar ha implementado un completo lenguaje musical de códigos numéricos. El MIDI permite que todos los elementos musicales ejecutados en una pieza , sean transmitidos digitalmente a través de un cable MIDI de un instrumento a otro.

El metodo que el MIDI emplea para enviar cada byte sobre un cable MIDI en un bit a la vez, como una serie de bits, esto es llamado transmisión de datos en serie y exactamente igual al usado por el estandar RS-232C, con un bit de inicio, 8 bits de datos y un bit de paro. Este es un metodo más lento que la transmision de datos en paralelo (en el metodo paralelo se requiere al menos ocho lineas en un cable de conexión, una por cada bit en un byte), pero la transmisión en serie tiene una ventaja, los cables son mucho mas baratos que los cables usados en transmisión paralelo, y pueden ser más largos que los cables en paralelo sin producir RFI (interferencia de radio frecuencia). Para mantener la velocidad de transmisión en niveles respetables, el MIDI envia 31,250 bits por segundo sobre los cables, una taza alta de transmisión en serie que permite al MIDI enviar cerca de 3125 bytes por segundo. (Si la operación aritmetica falla, es porque el MIDI utiliza dos bits extras por cada byte, para separar un byte de otro. Esto significa que en realidad se toman diez bits para enviar un byte de información a través de un cable MIDI.)

El primer byte en un mensaje MIDI es llamado byte de estatus. Este byte nos dice que tipo de mensaje se ha recibido. Por ejemplo, el byte de estatus puede identificar un mensaje como un mensaje de INICIO DE NOTA (Este mensaje nos dice que una nota acaba de empezar), un mensaje de cambio en la curva de afinación (este mensaje nos dice que la ruedecilla de afinación de un sintetizados ha sido movida), o cualquier otro tipo de mensaje.

Los bytes que siguen al byte de estatus son llamados bytes de datos. Cada byte de datos explica con más detalle la información dada por el byte de estatus. Por ejemplo, el primer byte de datos en un mensaje de INICIO DE NOTA, nos dice la afinación de la nota, y el segundo byte de datos nos dice la velocidad de ataque de la nota. Gran parte de los mensajes MIDI usan dos bytes de datos para portar información adicional: algunos necesitan solo un byte de datos; otros

no usan bytes de datos . La figura 2-1 muestra una tipica cadena de datos, mensajes con bytes de estatus y bytes de datos.

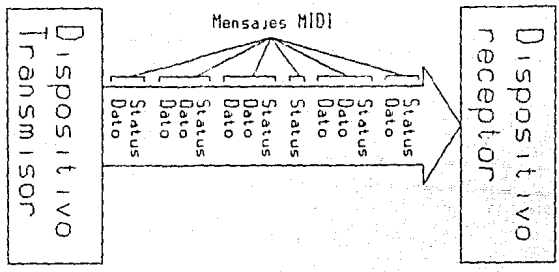


FIG. 2.1

Para distinguir los bytes de datos de los bytes de estatus, el MIDI usa solo bytes en el rango de 0 a 127 como bytes de datos (bytes con el bit más significativo en cero) y solo bytes en el rango de 128 a 255 como bytes de estatus (bytes con el bit más significativo en uno). Cuando un dispositivo MIDI recibe un byte de estatus sobre un cable MIDI, reconoce el mensaje y como interpretar cada byte de datos que le sigue. Por ejemplo, si un dispositivo recibe un byte de estatus de INICIO DE NOTA, interpreta los siguientes dos bytes como datos de afinación y velocidad de ataque. Si un dispositivo recibe un byte de estatus de cambio de afinación, interpreta los siguientes dos datos como bytes de datos de la curva de afinación.

CANALES MIDI

Consideremos la analogia con la clave Morse. Supongamos que hay muchos telegrafistas conectados a la misma linea telegráfica. Cuando un telegrafista envia un telegrama, muchos telegrafistas recibirán el mensaje. Esto posiblemente no sea práctico. Por ejemplo, si se desea enviar un telegrama al telegrafista A sin que los telegrafistas B, C y D lo reciban también, encontraríamos que es imposible.

La misma situación ocurre en el sistema MIDI. Consideremos un dispositivo maestro con varios dispositivos subordinados conectados a la misma linea. Cuando el dispositivo maestro envia un mensaje, todos los dispositivos en la cadena reciben este mensaje. El dispositivo maestro no puede activar un dispositivo sin activar a todos los demás al mismo tiempo. Esta situación es satisfactoria si se desea usar un teclado maestro para tocar los demás dispositivos al unisono, pero esto no trabajaría si se desea usar el dispositivo maestro para decir a cada dispositivo que haga cosas diferentes.

El MIDI usa CANALES MIDI para permitir la comunicación entre dispositivos individuales en un sistema MIDI. Para entender como trabaja, consideremos la linea telegráfica otra vez. El propietario de la linea telegráfica tiene dos reglas: Habrá dos clases de telegramas

- telegramas públicos que cualquiera puede recibir y leer y telegramas privados que solo telegrafistas autorizados pueden recibir y leer. La compañía de telégrafos da a cada telegrafista un número, y dice al telegrafista que lea los telegramas privados solo si ellos empiezan con ese número, cuando llega un telegrama privado y éste empieza con el número de identificación de alguien más el telegrafista rápidamente ignora el resto del telegrama. (Por supuesto, todos estos hipoteticos telegrafistas son supuestamente honestos)

El MIDI usa la misma técnica. Divide todos los mensajes en dos diferentes tipos: MENSAJES DEL SISTEMA que van a todos los dispositivos del sistema MIDI, y MENSAJES DE CANAL que van solo a un dispositivo especifico o a varios programados para recibir en el mismo canal. Cada mensaje de canal tiene un número de canal del 1 al 16 incluido en el byte de estatus. Cuando un dispositivo MIDI recibe el byte de estatus de un mensaje de canal, lee el número de canal. Entonces, checa si se le es permitido recibir mensajes de ese canal. Si es así, actua conforme al mensaje. Si no, ignora el mensaje. Como los mensajes del sistema no tienen un número de canal en el byte de estatus, todos los dispositivos que los reciben actuan de acuerdo a ellos.

Todo el proceso antes mencionados sucede a una gran velocidad. El efecto práctico es que los canales MIDI funcionan como canales de televisión más que como lineas telegráficas, como se muestra en la figura 2-2. El dispositivo maestro puede enviar mensajes en cualquiera de los 16 canales. Solo el dispositivo que se encuentra programado para recibir en el canal transmitido actuan de acuerdo al mensaje. Los dispositivos no activados para recibir en ese canal ignoran el mensaje.

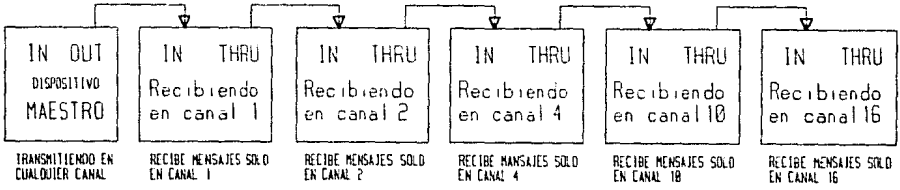


FIG. 2.2

MODOS DE RECEPCION DE CANALES MIDI

Para sintonizar un dispositivo MIDI a uno o más canales, primero se usa el panel de control para poner al dispositivo en uno de estos modos de recepción MIDI. El modo de recepción que se escoja determinará la manera en que el dispositivo MIDI obedecerá a los diferentes canales. Los cuatro modos son:

- Modo 1: Omni On/Poly
- Modo 2: Omni On/Mono
- Modo 3: Omni Off/Poly
- Modo 4: Omni Off/Mono

La primera mitad del nombre indica la manera en que el dispositivo MIDI monitoreará los canales MIDI que ingresen. Si el dispositivo se encuentra en modo Omni, entonces monitoreará todos los canales MIDI y responderá a todos los mensajes de canal, no importando el canal en que se esté transmitiendo. Si el Omni se encuentra apagado, entonces el dispositivo MIDI responderá únicamente a los mensajes de canal enviados en el canal o canales que el dispositivo esté programado para recibir.

La segunda mitad del nombre del modo nos dice como se tocarán las notas recibidas por el dispositivo MIDI. Si se activa el modo en Poly, el dispositivo puede tocar diversas notas al mismo tiempo, como un sintetizador polifónico. Cuando un mensaje llega y dice que comenzó una nota, el dispositivo MIDI mantendrá sonando esa nota aunque llegue otro mensaje que indique que debe comenzar una segunda nota. La primera nota para de sonar solo cuando el dispositivo reciba un mensaje que diga que la detenga o cuando el número de notas sonando exceda la capacidad del sintetizador. Si el modo es puesto en MONO, el dispositivo tocará notas como un sintetizador monofónico - una nota a un tiempo. Cada vez que un mensaje MIDI inicia una nueva nota, el dispositivo para de tocar cualquier nota previa aún y cuando no se reciba ningun mensaje MIDI que ordene cesar esa nota.

La combinación de los dos diferentes aspectos de un modo, nos da cuatro posibles modos descritos a continuación. Cada modo tiene una única forma de responder a los mensajes MIDI de canal.

MODO 1.-

El modo 1 es Omni On/Poly. Un dispositivo puesto en este modo recibe los mensajes de canal enviados en cualquier canal y toca las notas polifónicamente. Este es el modo en que la mayoría de los dispositivos responden cuando es la primera vez que se encienden. Este es un modo seguro porque el dispositivo responde a mensajes transmitidos en cualquier canal. No seríamos confundidos por una máquina que permanece oculta por no estar activada para recibir en el canal correcto.

MODO 2.-

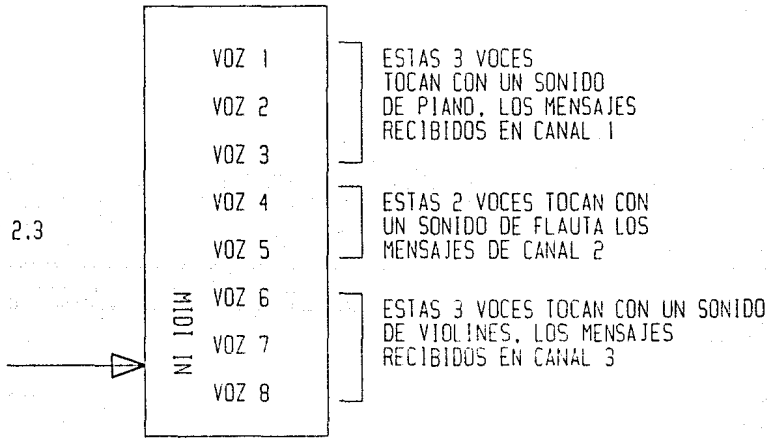
El modo 2 es Omni On/Mono. cuando se activa este modo, un dispositivo responde a mensajes de canal transmitidos en cualquier canal y toca las notas monofónicamente. Como la mayoría de la gente prefiere tener dispositivos que toquen las notas polifónicamente siempre que sea posible, el modo 2 es raramente usado.

MODO 3.-

El modo 3 es Omni Off/Poly. Un dispositivo puesto en este modo recibe mensajes de canal solo de ciertos canales y toca las notas polifónicamente. Se puede especificar el canal de recepción entre cualquiera de los 16 canales MIDI. La mayor parte de los dispositivos reciben mensajes en un solo canal cuando son puestos en el modo 3. Un termino comunmente usado en el ambiente de la música sintetizada es "MULTI TIMBRE" o "MULTI TIMBRAL", cuando un sintetizador o generador

de tonos es multi timbre puede tocar varias notas a la vez, pero usando timbres diferentes (o iguales) para cada nota. En la actualidad, la mayor parte de los sintetizadores multitimbrales, pueden recibir mensajes en diversos canales al mismo tiempo y tocar las notas de cada canal usando un timbre (comercialmente llamado patch) diferente del sintetizador. Efectivamente este modo divide al sintetizador en una serie de pequeños sintetizadores, cada uno respondiendo a un canal MIDI diferente.

FIGURA 2.3



Como un ejemplo, consideremos un sintetizador multitimbral de ocho voces, como el mostrado en la figura 2-3, puesto en modo 3. Podemos poner tres de estas voces tocando un sonido de piano electrico para responder a el canal 1 de MIDI, dos voces tocando un sonido de flauta respondiendo al canal dos, y las tres voces restantes tocando un sonido de violin respondiendo al canal 3. El sintetizador ahora actua como tres pequeños sintetizadores, cada uno con un diferente sonido y cada uno recibiendo mensajes en su propio canal MIDI.

MODO 4.-

El modo 4 es Omni Off/Mono, el cual activa a un dispositivo para recibir mensajes en un canal determinado y tocar las notas monofónicamente. Este modo es usado casi exclusivamente por los sintetizadores multitimbrales. Un sintetizador en este modo activa un diferente sonido para cada una de las voces y entonces asigna un diferente canal MIDI a cada voz. El resultado efectivo es dividir al sintetizados en muchos sintetizadores monofónicos, cada uno recibiendo mensajes en su propio canal MIDI.

Como un ejemplo, pongamos un sintetizador multitimbral de cuatro voces en modo cuatro, como se muestra en la figura 2-4. Entonces se ajusta cada una de las cuatro voces a un diferente sonido y se asigna un número de canal MIDI a cada voz.

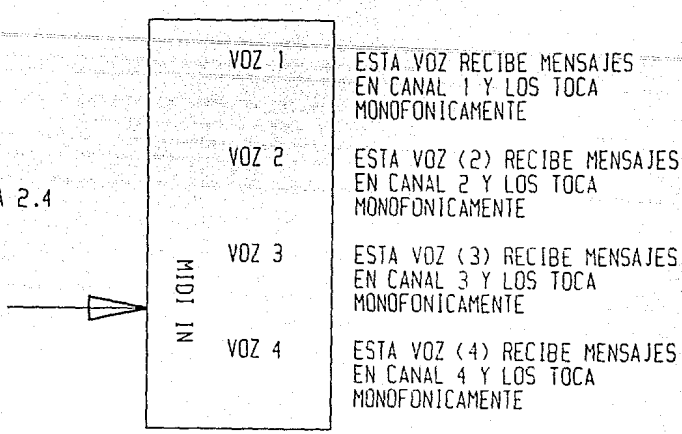


FIGURA 2.4

El modo 2 y el modo 4, son usualmente menos deseables que los modos polifónicos, pero las limitaciones técnicas de algunos sintetizadores crearon la necesidad de que existieran estos modos. Actualmente la mayor parte de los sintetizadores tienen la capacidad de tocar notas polifónicamente y de recibir mensajes de diferentes canales MIDI al mismo tiempo.

Ahora que conocemos la manera en que dispositivos MIDI usan la transmisión de datos en serie, y los canales MIDI par transmitir información, estamos listos para clasificarlos (figura 2-5). En la siguiente sección se listan los diferentes tipos de mensajes especificados por el estandar MIDI, se explica que clase de datos puede portar cada uno y se explica la manera en que responderan los dispositivos MIDI a cada mensaje.

MENSAJES MIDI

SISTEMA			CANAL	
	TIEMPO REAL	COMUNES	MODO	VOZ
- MENSAJES DE SISTEMA EXCLUSIVO.	- TIEMPO DE RELOJ - INICIO - ALTO - CONTINUAR - SENSANDO ACTIVIDAD	- POSICION DEL PUNTERO DE CANCION - SELECCION DE CANCION - SOLICITUD DE AFINACION - EXO	- CONTROL LOCAL - APAGAR TODAS LAS NOTAS - MODO OMNI INACTIVO - MODO OMNI ACTIVO - MODO MONO ACTIVO - MODO POLY ACTIVO	- INICIO DE NOTA - FIN DE NOTA - PRESION POLIFONICA DE TECLA - PRESION DE CANAL - CAMBIO DE PROGRAMA - CAMBIO DE CONTROL - CAMBIO DE PITCH BEND

FIGURA 2.5

" MENSAJES DE CANAL "

Hay dos diferentes tipos de mensajes de canal: mensaje de voz y mensaje de modo. Los mensajes de voz envían entre dispositivos MIDI datos de la ejecución actual, describiendo la acción del teclado, movimiento de controles, y presión de botones en el panel de control. Los mensajes de modo de canal determinan la manera en que el dispositivo MIDI receptor responderá a los mensajes de voz de canal. Todos estos mensajes son transmitidos con un número específico de canal y serán recibidos solo por los dispositivos que se encuentren programados para responder a ese canal.

MENSAJES DE VOZ

La mayoría de los mensajes de voz describen la música definiendo afinación, amplitud, timbre, duración, y otras cualidades del sonido. Cada mensaje tiene por lo menos uno y usualmente dos bytes de datos que acompañan al byte de estatus para describir esas cualidades del sonido.

INICIO DE NOTA.- El mensaje de Inicio de nota (nota on) señala el inicio de una nota. Es usualmente enviado por un teclado equipado con MIDI cuando un ejecutante presiona una tecla. Este mensaje dice a un dispositivo receptor que empiece a tocar una nota. Un mensaje de Inicio de nota transmite tres bytes de información: El canal y la finalidad del mensaje, la afinación de la nota (un número entre 0 y 127), y la velocidad de ataque de la nota (también un número entre 0 y 127).

El valor de afinación (0-127) representa el número de semitonos que la nota se encuentra alejada de un do natural cinco octavas abajo del do central. El valor inferior, cero, representa el DO más bajo, 60 representa el DO central, y el mayor valor, 127, representa un SOL cinco y media octavas arriba del do central. Esto se puede apreciar mejor en la figura 2-6.

El rango de valores de velocidad va desde cero hasta 127. Como los sintetizadores en general usan la velocidad de la nota para ajustar el volumen de las notas, un valor de velocidad igual a cero se considera igual que un mensaje de fin de nota. Un valor de velocidad 11 es muy suave (un pianisisimo), mientras que una velocidad de 127 es muy fuerte (como fortississimo). Un valor de 64, la mitad entre los dos extremos, es moderadamente fuerte, un lugar entre mezzo piano y mezzo forte. Los dispositivos MIDI que no pueden manejar variaciones en la velocidad de ataque envían un valor de 64 para cualquier mensaje de Inicio de nota transmitido.

FIN DE NOTA.- El mensaje Fin de nota (nota off) señala el fin de una nota. Esto ocurre cuando un ejecutante libera una tecla en un teclado equipado con MIDI. Un dispositivo recibe un mensaje de fin de nota y detiene una nota empezada a tocar con anterioridad en respuesta a un mensaje de Inicio de nota. Este mensaje contiene tres bytes de información: el canal y el tipo de mensaje, la afinación de la nota (un número entre 0 y 127), y la velocidad de liberación (también un

número entre 0 y 127). El valor de afinación representa una nota, como lo hace un mensaje de inicio de nota, y la velocidad de liberación trabaja justo como la velocidad de ataque: Un valor bajo de liberacion significa que la nota se soltó muy suavemente, y 127 significa que la nota es soltada muy rápidamente. Como la velocidad de liberación a menudo controla la manera en que las notas terminan, un valor bajo de la velocidad de liberación representa una nota desvaneciéndose lentamente, mientras que un valor alto representa una nota que termina muy rápidamente. Los dispositivos que no manejan velocidad de liberación, asignan un valor de 64 para cualquier mensaje de Fin de nota transmitido.

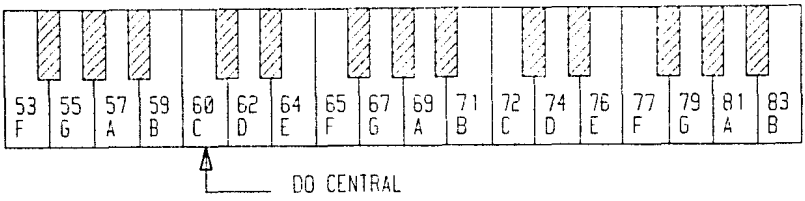


FIGURA 2.6

PRESION DE TECLA- (POLYFONIC KEY PRESSURE). Este mensaje reporta una presión en una tecla que esté siendo tocada. Es usualmente transmitido por un teclado que tiene la capacidad de detectar la presión aplicada a cada tecla individualmente. Un dispositivo recibiendo un mensaje de Poliphonic Key Pressure puede actuar de muchas maneras, la mayoría de los dispositivos agregan más vibrato a una nota cuando reciben información indicando más presión en la tecla. Polyphonic Key Pressure contiene tres bytes de información: El canal en el que el mensaje está siendo transmitido, la afinación de la nota presionada (un número entre 0 y 127), y la presión de la nota (un número entre 0 y 127).

Los valores de afinación son expresados tal como se hace en un mensaje de Inicio de nota. El rango del valor de presión va desde 0 a 127. Como la presión a menudo determina vibrato, un bajo valor de presión usualmente causa solo una pequeña cantidad de vibrato en la nota afectada, y un alto valor de presión causa una gran cantidad de vibrato.

PRESION DE CANAL (CHANNEL PRESSURE).- Este mensaje reporta una presión general de cualquier nota que se esté tocando. Este mensaje es transmitido por teclados que pueden detectar una presión general en sus teclas pero no presión en cada tecla individualmente. Un dispositivo recibiendo un mensaje de Presión de Canal puede reaccionar de diferentes formas, pero la mayoría de los dispositivos usan la información para activar el vibrato de todas las notas que el dispositivo esté tocando. Este mensaje contiene dos bytes de información: el canal en el que está siendo transmitido el mensaje y

el valor de presión (un número entre 0 y 127).

Al igual que un mensaje de Polyphonic Key Pressure, el valor de la presión en un mensaje de presón de canal tiene un mínimo en cero y un maximo en 127.

CAMBIO DE PROGRAMA (PROGRAM CHANGE).- El mensaje de cambio de programa señala un cambio de timbre (patch) en el sintetizador, el cual ocurre cuando un ejecutante presiona un boton de patch en el panel de control del sintetizador. Un dispositivo recibiendo un mensaje de cambio de programa cambia el timbre que se está usando para tocar las notas. Un cambio de programa consta de dos bytes: El canal en que se está enviando el cambio de programa, y el nuevo timbre (patch) seleccionado (un número entre 0 y 127).

Como un cambio de programa selecciona un nuevo patch en un dispositivo receptor, el número de patch enviado afecta las características del sonido de todas las notas tocadas en el dispositivo receptor. Los números de patchs no producen un efecto estandar en todos los sintetizadores. El patch específico guardado en cada número varía de sintetizador a sintetizador, y esto puede traer problemas. Por ejemplo si se selecciona un boton de patch número 4 en un sintetizador para generar un sonido de piano, el sintetizador enviará un mensaje de cambio de programa con un patch número 4 a cualquier otro sintetizador. Cuando el segundo sintetizador recibe el mensaje, cambia a su propio patch número 4, el cual podría ser un clarinete, o cualquier otra clase de timbre correspondiente al patch 4 del sintetizador.

CAMBIO DE CONTRCL.- El mensaje de cambio de control envía información acerca de un nuevo estado de un control en un dispositivo MIDI. Este mensaje puede ser enviado siempre que se mueva un control o apriete un switch en un sintetizador u otro dispositivo MIDI. Muchos sintetizadores tienen controles que no envían mensajes de cambio de control. El cambio de control de volumen en algunos sintetizadores, por ejemplo, no envía mensajes MIDI para comunicar su nuevo estado.

Un dispositivo recibiendo un mensaje de cambio de control cambia su operación para reflejar el nuevo estado del control. Un mensaje de cambio de control contiene tres bytes de información: El número de canal sobre el cual es enviado el mensaje; un número entre 0 y 127, llamado valor del control, que identifica el control que se ha movido o presionado; y un número entre 0 y 127, llamado valor del estado, que representa el nuevo estado del control o futura definición del control.

Existen tres tipos de controles que un mensaje de cambio de control puede representar: controles continuos (controles que tienen un amplio rango de estados), switches (controles que pueden estar encendidos o apagados y no pueden tener valores intermedios) y controles de datos (controles que pueden suministrar datos numéricos directamente o por pasos a través de valores en cada presión de un boton).

Las especificaciones MIDI asignan un rango de números para cada tipo de control. Los números de control de 0 a 63 son usados para controles continuos, los números de 64 a 95 son usados para switches (y además pueden ser usados para cierto tipo de controles continuos), y los números de control del 96 al 127 son usados para controles de

datos. Algunos números de control son asignados a controles especificos. Por ejemplo, el control número uno es un control de modulación (un control continuo), un control número 64 es un pedal de sosten (un switch). Muchos números de control no han sido definidos, sin embargo, los fabricantes pueden usarlos para diversos controles que no se definieron en las especificaciones MIDI.

El último conjunto de valores (0-127) comunica el nuevo estado de un control que acaba de ser movido. Para un control continuo, el valor puede reflejar 128 estados diferentes, desde 0 (girar o presionar el control al valor inferior del rango) a 127 (girar o presionar el control al maximo). Si el control continuo necesita más de 128 valores para comunicar su estado preciso, entonces un dispositivo MIDI puede enviar mensajes de cambio de control ligados. El conjunto de valores del primer mensaje combinado con el conjunto de valores del segundo mensaje permiten representar 16,384 estados diferentes. El conjunto de valores para un switch es simple. Cualquier número entre 0 y 63 significa que el switch está abierto, y cualquier valor de 64 a 127 significa que el switch está cerrado.

CAMBIO DE AFINACION (PITCH BEND CHANGE).- Este mensaje envia información acerca de un nuevo estado de un control de curva de afinación. Es usualmente enviado cuando se mueve la ruedecilla de afinación de un sintetizador. Un dispositivo recibiendo un mensaje de cambio de curva de afinación cambia la afinación de las notas que se estén tocando hacia arriba o abajo dependiendo del nuevo valor. Un mensaje de cambio en la curva de afinación consta de tres bytes, el primero indica el tipo de mensaje y el canal en el cual es enviado el mensaje, y los siguientes dos bytes son un número entre 0 y 16,383, que comunican la nueva posición del control de curva de afinación. Si el control de curva de afinación se encuentra en la posición cero: significa que la ruedecilla de afinación se encuentra en el valor inferior y no en un valor neutral; si el control se encuentra en 8192, significa que se encuentra exactamente en el centro y que la afinación no se ha movido ; si el control se encuentra en 16,383 significa que la ruedecilla de afinación se ha movido hasta el tope máximo, y la afinación esta tan alta como se puede.

La cantidad en que la afinación se mueve como resultado de un cambio en la curva de afinación varia de dispositivo a dispositivo dependiendo del rango de la curva de afinación seleccionado en el dispositivo. Por ejemplo, un dispositivo MIDI puede tener la curva de afinación hacia arriba y hacia abajo de tres semi tonos en cada dirección, mientras un segundo dispositivo MIDI tiene un rango de una octava en cada dirección. Si ambos dispositivos reciben el mismo mensaje de cambio en la curva de afinación diciendo que la posición de la ruedecilla de afinación es cero (el más bajo valor posible), el resultado será completamente diferente. El primer dispositivo bajará todas las notas que está tocando tres semi tonos, mientras que el segundo dispositivo MIDI bajará la afinación una octava completa. Para estar seguros que los dos dispositivos sonarán al unisono, es necesario ajustar los valores de cambio de afinación de los dos dispositivos para cubrir el mismo rango.

Dentro de la música, el pitch bend es equivalente a lo que se conoce como sonido con portamento, que equivale a ligar dos sonidos deslizando la afinación de un sonido hasta llegar a un segundo sonido.

Los mensaje de modo de canal seleccionan el modo de recepción de diferentes dispositivos MIDI, detiene notas no deseadas y afectan el control local de un dispositivo. Son mucho más simples que los mensajes de voz de canal y no comunican mucha información en cada mensaje.

CONTROL LOCAL (LOCAL CONTROL).- El mensaje de control local conecta o desconecta el control de un dispositivo sobre su propio generador de sonido. Es usual que este mensaje sea enviado por un secuenciador o una computadora que controla a un sintetizador. Este es un mensaje muy manual enviado si se está usando un dispositivo subordinado como un sintetizador con un teclado, y se desea que toque únicamente las notas provenientes de mensajes MIDI. Ajustando el control local del sintetizador a off, sus teclas no podrán producir ningun sonido. Este control no tiene ningun efecto en las funciones internas del sintetizador.

Este control puede apagarse también, si se desea usar el teclado del sintetizador y sus controles como un teclado remoto. Siempre que se opriman teclas o un control, el resultado será enviado sobre cables MIDI par controlar otro sintetizador sin producir sonidos con los propios generadores del sintetizador.

Un mensaje de control local comunica dos bytes de información: El canal en el cual se envía y el estado del control local, on o off.

APAGAR TODAS LAS NOTAS (ALL NOTES OFF).- Si un sintetizador que ha recibido un mensaje de inicio de nota y está tocando notas por alguna razón no recibe el mensaje de fin de nota que necesita para dejar de tocar las notas, entonces algunas notas sonarán indefinidamente. Un mensaje de todas las notas off es conveniente para manejar esta situación. Es mejor que tratar de encontrar la afinación de las notas que se encuentran sonando para enviar un mensaje de fin de nota para cada una de las 128 posibles afinaciones, un simple mensaje de todas las notas off dice a los sintetizadores receptores que paren de tocar todas las notas.

Un mensaje de todas las notas off comunica solamente el número de canal en que se está transmitiendo. No es necesario especificar alguna otra cosa por lo que no se envían bytes de datos.

MODO OMNI DESACTIVADO (OMNI MODE OFF).- Cuando un dispositivo MIDI recibe un mensaje de Omni Mode Off, cambia su modo de recepción de canal (descrito antes en este capitulo) para que el dispositivo reciba mensajes MIDI en canales MIDI discretos en vez de recibir mensajes en todos los canales. Si el dispositivo ya se encuentra recibiendo en modo de Omni Off, entonces este mensaje no tiene efecto.

El mensaje de Omni Mode Off es usualmente transmitido por un secuenciador o una computadora controlando diversos dispositivos MIDI. Para evitar que un dispositivo que recibe un mensaje de Omni Mode Off no pueda recibir los mensajes de fin de nota de las notas que se encontraba tocando al momento de recibir el mensaje de Omni Mode Off, cuando se recibe un mensajes de Omni Mode Off automáticamente se

apagan todas las notas.

El mensaje de Omni Mode Off comunica únicamente el número de canal en el cual es transmitido.

MODO OMNI ACTIVADO (OMNI MODE ON).- Un dispositivo que recibe un mensaje se Omni Mode On cambia su modo de recepción de canal (Si no se encuentra ya en el estado se cambia a Omni Mode On) y entonces recibirá mensajes MIDI de cualquier canal que sea transmitido. Como el mensaje de Omni Mode Off, el mensaje de Omni Mode On es usualmente enviado por un secuenciador o una computadora, y las notas tocadas por el dispositivo receptor se apagan inmediatamente. El mensaje de Omni Mode On comunica únicamente el canal en el cual es transmitido.

MODO MONO ACTIVADO (MONO MODE ON).- Un dispositivo MIDI que recibe el mensaje de Mono Mode On cambia su modo de recepción de canal para que las notas sean tocadas monofónicamente. Este mensaje es usualmente enviado por un secuenciador o una computadora controlando el sistema. Este mensaje apaga el modo polifónico y detiene todas las notas que está tocando el dispositivo receptor. El mensaje de Mono Mode On comunica únicamente el número de canal en el que es transmitido el mensaje.

MODO POLY ACTIVADO (POLY MODE ON).- Un dispositivo recibiendo el mensaje se poly mode on cambia el modo de recepción de canal para que el dispositivo toque polifónicamente. Este mensaje es enviado por un secuenciador o una computadora controlando el sistema. Automáticamente apaga el Modo Mono y apaga todas las notas del dispositivo receptor. Poly Mode On comunica únicamente el canal en el que es transmitido.

MENSAJES DEL SISTEMA

Los dispositivos MIDI envian mensajes del sistema sin usar número de canal para que todos los dispositivos receptores puedan actuar segun el mensajes. Hay tres tipos de mensajes del sistema: mensajes comunes del sistema, mensajes de tiempo real del sistema, y mensajes de sistema exclusivo. Los mensajes comunes del sistema ayudan a la selección de canciones y sintonia entre dispositivos MIDI, mientras que los mensajes de tiempo real del sistema sincronizan los tiempos entre secuenciadores durante la ejecución o la grabación de canciones. Los mensajes de sistema exclusivo envian datos entre dispositivos especificos que no pueden ser enviados como ninguna clase de mensajes MIDI.

Para entender como los mensajes del sistema hacen trabajar a los secuenciadores, es importante entender que es una canción en el lenguaje del MIDI. Cuando un dispositivo con un secuenciador (como una máquinas de percusiones, una computadora, o un secuenciador dedicado) graba los mensajes que llegan, los graba en el orden de arribo y graba la cantidad de tiempo que pasa entre mensajes. Estos mensajes grabados son llamados secuencia. La mayoria de los secuenciadores permiten editar mensajes en una secuencia y ligarla con diferentes secuencias

guardadas en memoria. La secuencia final lista para ejecutarse es llamada canción.

La mayoria de los secuenciadores guardan diferentes canciones en su memoria. Se puede llamar a una de las diferentes canciones preguntando por su nombre o número. La mayoria de las máquinas de percusiones, por ejemplo, tienen una serie de botones que se pueden oprimir para llamar a diferentes canciones. Cada boton que se oprime trae una canción diferente. Una vez que la canción está lista para tocarse, se toca esta como se haria en un audio cassette. Un boton inicia la canción y otro boton la detiene.

Se puede iniciar una canción en cualquier lugar, algo como adelantar o regresar un casette en una grabadora a la sección que se dese escuchar. Una caracteristica especial de los secuenciadores, es que se pude aumentar o disminuir la velocidad asi como tocar al revés, apretando algún boton a ajustando algún control.

MENSAJES DE TIEMPO REAL DEL SISTEMA

Estos mensajes son todos muy cortos y simples. Son solo de un byte de longitud y no portan bytes de datos con ellos, como número de canal, afinación, o velocidad. Los mensaje de tiempo real del sistema sincronizan dispositivos MIDI en ejecución, es muy importante que sean enviados precisamente al tiempo que son requeridos. No deben ser forzados a que otros mensajes MIDI terminen para ser transmitidos. Para evitar retardos, los mensajes de tiempo real del sistema pueden ser insertados entre el byte de estatus y el primer byte de datos de un mensaje de canal, como muestra la figura 2-7. cuando un dispositivo receptor recibe un mensaje de tiempo real, rápidamente actua de acuerdo al mensaje y regresa a la actividad previa.

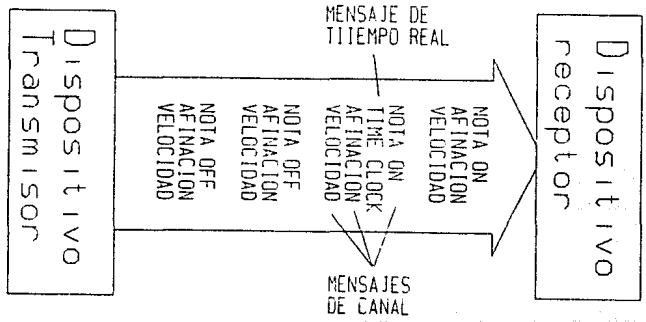


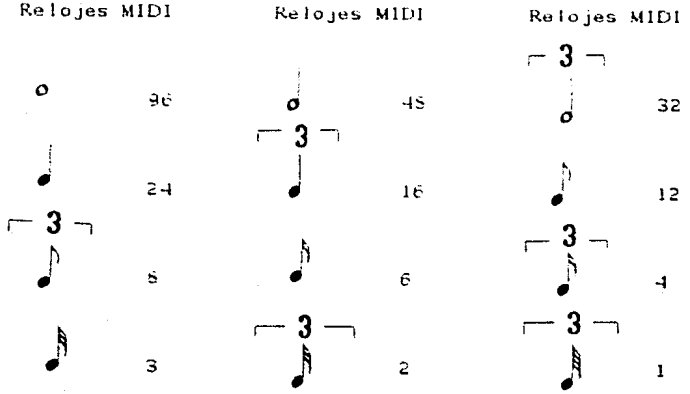
FIGURA 2.7

TIMING CLOCK.- Este mensaje (a menudo llamado reloj MIDI) es el más importante y frecuentemente enviado de los mensajes de tiempo real; ayuda a mantener secuenciadores separados en el mismo sistema MIDI

tocando al mismo tiempo. Cuando un secuenciador maestro toca una canción, envia una cadena de mensajes de reloj MIDI para comunicar el tiempo a otros secuenciadores. La rapidez con la que se reciben estos mensajes, es la rapidez con que los secuenciadores tocan una canción.

Para mantener una referencia estandar de tiempo, las especificaciones MIDI establecen que 24 relojes MIDI son igual a una nota de un tiempo (cuarto). Esto significa, por supuesto, que las notas tienen un equivalente en relojes MIDI, como se muestra en la figura 2-8.

Para ver como trabaja el reloj MIDI, consideremos un secuenciador que tiene que tocar una nota de un cuarto y dos notas de octavo, recibiendo una cadena de relojes MIDI de otro secuenciador que ajusta el tiempo de ejecución. Cuando el secuenciador receptor empieza a tocar la nota de cuarto también comienza a contar los sucesivos tiempos de reloj . Una vez que ha contado 24 relojes, termina la nota de cuarto y empieza la nota de octavo. Despues de contar 12 relojes más, termina la nota de octavo y comienza la segunda nota de octavo. por último despues de 12 relojes más, termina la segunda nota de octavo. La velocidad en que llegan los relojes MIDI determina el tiempo de las notas que toca el secuenciador.



INICIO (START).- El mensaje de inicio dice a los secuenciadores receptores que empiecen a tocar la canción que esten listos para ejecutar.

DETENERSE (STOP).- Este mensaje dice a los receptores que dejen de tocar una canción.

CONTINUAR (CONTINUE).- Este mensaje dice a los secuenciadores receptores que inicien la ejecución de la canción en el punto que se detuvieron por última vez. El mensaje de continuar permite a los secuenciadores comenzar desde el punto donde se recibió el mensaje de stop.

CENSANDO ACTIVIDAD (ACTIVE SENSING).- En un mundo perfecto, los cables MIDI nunca se desconectan cuando no deben hacerlo. En nuestro mundo, los cables se desconectan a menudo con consecuencias embarazosas: No se tiene una respuesta como debería en el teclado maestro, y lo peor, las notas tocadas en un dispositivo subordinado continúan tocando porque no recibe los mensajes de Fin dE nota de las notas que se encuentran sonando. El mensaje de detección de actividad puede ayudar a evitar esta delicada situación.

Cuando un dispositivo envía un mensaje de Active Sensing a otro dispositivo, el dispositivo receptor detecta que hay una buena conexión entre los dos dispositivos. Después de que el dispositivo receptor recibe el primer mensaje de Active Sensing, espera recibir un mensaje de detección de actividad regularmente cada 300 mili segundos (más o menos 3 mensajes por segundo). Si no recibe regularmente el mensaje de detección de actividad, asume que la conexión a sido rota y apaga las notas que está tocando. Este mensaje es usado para pruebas de conexión.

Pocos dispositivos están equipados para transmitir mensajes de detección de actividad: De cualquier forma, esto no causa problemas de compatibilidad. Si un dispositivo equipado para recibir mensajes de detección de actividad no recibe uno, entonces no espera que los mensajes lleguen a intervalos regulares. Y si un dispositivo no equipado para recibir mensajes de detección de actividad recibe uno, lo ignora.

REINICIO DEL SISTEMA (SYSTEM RESET).- El mensaje de reinicio del sistema dice a los dispositivos receptores que cambien su estado al estado que tienen por defaul, el estado en que se encuentran cuando se encienden por primera vez. Este estado inicial a menudo significa que los dispositivos están en Modo 1 de recepción de canal (Omni On/Poly), que el control local se encuentra activado con todas las voces apagadas (no se están tocando notas), que la canción en el secuenciador está lista para tocarse desde el principio, y que la reproducción de canciones está apagada. Como el estado inicial puede variar de dispositivo a dispositivo, el mensaje de reinicio del sistema puede tener diferentes resultados en diferentes dispositivos.

MENSAJES COMUNES DEL SISTEMA

Los mensajes comunes del sistema son comandos que preparan a los secuenciadores y a los sintetizadores para tocar una canción. Estos mensajes nos permiten seleccionar una canción, encontrar un lugar común de inicio en la canción, y poner a los sintetizadores en un estado adecuado.

SELECCION DE CANCION (SELECT SONG).- El mensaje de selección de canción dice a los secuenciadores que carguen una canción especifica de su memoria. Este mensaje identifica las canciones usando un número entre 0 y 127. El secuenciador responde encontrando y cargando la canción especificada por el nombre o número.

POSICION DEL PUNTERO DE CANCION (SONG POSITION POINTER).- El mensaje de puntero de canción ajusta a un secuenciador para empezar a tocar una canción en cualquier punto. Este mensaje porta con el un número entre 0 y 16,383 para ser el nuevo puntero de canción. Este número nos dice la distancia en incrementos de 6 relojes MIDI del inicio de la canción hasta el punto deseado. Ya que 6 relojes MIDI son igual a una nota de dieciseisavo, se puede pensar que el puntero de canción es la distancia medida en diesiseisavos desde el inicio de la canción.

La posición del puntero de canción no puede usarse mientras los secuenciadores están tocando. Si se usa durante la ejecución de una canción, habra dificultades en el envio del mensaje y los secuenciadores se saldrán de sincronia.

SOLICITUD DE AFINACION (TUNE REQUEST).- El mensaje de solicitud de afinación pide a los dispositivos MIDI que se afinen ellos mismos. Este mensaje es raramente usado, ya que los sintetizadores digitales no se desafinan, y los sintetizadores analógicos que lo hacen, a menudo no tienen la habilidad de afinarse ellos mismos.

EOX (END OF EXCUSIV) .- El mensaje EOX informa a los dispositivos receptores que un mensaje de sistema exclusivo acaba de terminar. Las especificaciones MIDI clasifican al mensaje EOX como un mensaje común del sistema, por esto se encuentra mencionado en esta sección, pero se explicará con más detalle en la próxima sección.

MENSAJES DE SISTEMA EXCLUSIVO

El formato de un sistema exclusivo permite a los fabricantes de dispositivos MIDI crear mensajes para transmitir entre sus dispositivos. Cada mensaje de sistema exclusivo empieza con un byte de estatus que identifica al mensaje como un mensaje de sistema exclusivo. El primer byte de datos que sigue es un número de identificación entre 0 y 127, el cual identifica la marca del dispositivo que envia el mensaje. (por ejemplo, el número 67 significa que el fabricante es YAMAHA.)

Cuando un dispositivo recibe un mensaje de sistema exclusivo, lee el número de identificación para ver si el mensaje fue transmitido por un dispositivo hecho por el mismo fabricante. Si es asi el dispositivo lee el resto del mensaje. Algunos dispositivos, como computadoras o secuenciadores pueden ser programados para recibir y entender mensajes de sistema exclusivo enviados por dispositivos hechos por otro fabricante. Estos dispositivos pueden transmitir también mensajes exclusivos con número de identificación de diferentes fabricantes.

Un mensaje de sistema exclusivo puede tener tantos bytes de datos como el dispositivo desee enviar. Estos bytes de datos (todos números entre 0 y 127) pueden representar cualquier cosa que se desee. Algunos fabricantes usan los datos del sistema exclusivo para enviar información de timbres (patches) entre sintetizadores, ya que los timbres diseñados para un sintetizador pueden ser cargados en un segundo sintetizador que utilice el mismo tipo de sintesis sin necesidad de introducir los datos por medio del panel de control. Otros fabricantes usan mensajes de sistema exclusivo para transmitir mensajes propios que usan caracteristicas especiales de sus equipos. Por ejemplo, los sintetizadores Yamaha usan mensajes de sistema

exclusivo para mensajes de notas on y off especiales que pueden cambiar la afinación común de las notas.

Cuando un dispositivo ha terminado de enviar un mensaje de sistema exclusivo, envia un mensaje de EOX para indicar el fin del mensaje. Cuando los dispositivos que han ignorado el mensaje de sistema exclusivo reciben el mensaje de EOX, empiezan a poner atención a los mensajes que llegan sobre los cables MIDI. Por supuesto, algunos mensajes, como tiempos de reloj y otros mensajes de tiempo real, pueden ser transmitidos entre los bytes de datos de un mensaje de sistema exclusivo. Como los mensajes de sistema exclusivo contienen un solo byte de estatus (números entre 128 y 255), los dispositivos receptores que están ignorando los bytes de datos del sistema exclusivo (números entre 0 y 127) pueden reconocer los mensajes de tiempo real y actuar segun estos.

TRANSMITIENDO EVENTOS SIMULTANEOS

El MIDI es un interface de datos serie: que envia mensajes simples a través de cables MIDI uno tras otro. Los eventos musicales no siempre suceden uno tras otro. Que sucede, por ejemplo, si un ejecutante toca un acorde presionando tres teclas en un sintetizador al mismo tiempo ? El MIDI debe tener algún mecanismo para transmitir estas tres notas simultaneas usando mensajes MIDI.

El MIDI soluciona este problema siendo más rapido que la mano y el oido humano. El MIDI puede enviar también mensajes largos (con excepción de mensajes de sistema exclusivo) a una velocidad de 1040 mensajes por segundo. Un dispositivo MIDI puede leer la presión de una tecla más rápido que esto. Cuando un ejecutante toca un acorde, que aparentemente suena como presiones de teclas simultaneamente, están casi siempre separadas por inaudibles microsegundos. Estas notas son tocadas tan juntas que suenan como si fuesen simultaneas. Los teclados MIDI envían estos acordes a una velocidad tan alta que suenan como notas simultaneas.

CORRIENDO EL ESTADO (RUNING STATUS).- Los dispositivos MIDI pueden usar una técnica llamada runing status para hacer la transmisión de datos más rápida. Correr el estado permite transmitir cadenas de mensajes del mismo tipo sin repetir el byte de estatus para cada mensaje. Cuando el dispositivo receptor recibe un byte de estatus, permanece en ese estado hasta que recibe un nuevo byte de status, e interpreta cualquier byte de datos subsecuente como lo indica el modo de estatus en que se encuentra.

Por ejemplo, consideremos una serie de mensajes de inicio de nota. Cada mensaje tiene un byte de estatus seguido por dos bytes de datos, uno para la afinación y otro para la velocidad. Para enviar una serie de mensajes de Inicio de nota usando Runing status, un dispositivo envia un byte de estatus de Inicio de nota seguido por un par de bytes de datos para cada nota. Cuando el dispositivo receptor recibe el primer mensaje de Inicio de nota, entra en el Modo de InicIO DE NOTA, e interpreta cada par de bytes de datos siguientes como valores de afinación y velocidad de nuevas notas. Como una nota con velocidad cero no tiene volumen, un par de bytes con velocidad cero

pueden apagar una nota sin enviar un mensaje de FIN dE NOTA que romperia la cadena de datos de inicio de nota.

Cuando el dispositivo que transmite desea enviar otro tipo de mensajes, simplemente deja de enviar datos y transmite otro mensaje tal como lo haria normalmente. Tan pronto como el dispositivo receptor recibe un nuevo byte de estatus, entra en el modo de estatus correspondiente e interpreta los nuevos datos adecuadamente. Eliminando la repetición de byte de estatus, la tecnica runing status reduce el número de bytes de los mensajes e incrementa la velocidad de transmisión.

TIPOS DE CONEXIONES

En teoria, el MIDI puede conectar un número infinito de dispositivos. En la práctica hay limitaciones. Si se conectan muchos dispositivos a un mismo sistema, y se transmiten mensajes largos simultaneamente, se pueden sobrecargar la capacidad del MIDI (esto es llamado MIDI clog). Una condición que puede limitar el número de dispositivos que se tienen en un sistema es que todos estén tratando de comunicarse al mismo tiempo.

DAISY CHAINING

Ya que la mayoria de los sistemas usan mas de dos dispositivos, es importante que exista alguna manera de conectar a los dispositivos para que los mensajes de uno puedan viajar a los demás dispositivos. Si cada dispositivo tiene un MIDI THRU, entonces se pueden conectar en daisy chain como muestra la figura 2.9. La función del puerto MIDI THRU es transmitir la señal que se recibe en el puerto MIDI IN

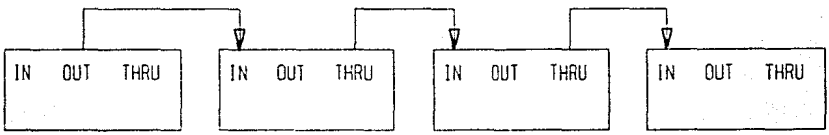


FIG 2.9

En una configuración como esta, el primer dispositivo envia mensajes a través de su MIDI OUT a el MIDI IN del segundo dispositivo. El segundo dispositivo envia una copia exacta del mensaje recibido en su MIDI IN, por el MIDI THRU a el MIDI in del tercer dispositivo. El tercer dispositivo envia una copia de este mensaje por su MIDI THRU al MIDI in del cuarto dispositivo. Esta cadena puede extenderse para incluir cualquier número de dispositivos MIDI, conectando el MIDI THRU de un dispositivo a el MIDI IN del siguiente dispositivo.

El MIDI THRU pasa el mensaje sin alterar desde el primer dispositivo en la cadena a los demás dispositivos en la cadena. La mayor parte de los dispositivos MIDI no pueden transmitir información propia usando el MIDI THRU. Esto hace a la configuración daisy chaining muy eficiente para un dispositivo maestro con una serie de dispositivos conectados a el, ya que el dispositivo maestro no necesita recibir información de los dispositivos esclavos, y los dispositivos esclavos no tienen que enviar información al dispositivo maestro ni a ningun dispositivo esclavo. Por supuesto, es posible conectar el puerto MIDI IN del dispositivo maestro, a el MIDI OUT de algún dispositivo esclavo, si se desea que el dispositivo maestro reciba información de algún dispositivo esclavo.

En teoría, es posible crear una cadena daisy chain tan larga como se quiera si todos los dispositivos MIDI empleados cuentan con MIDI THRU. En la práctica, no siempre es posible. Algunos puertos MIDI THRU, retardan la señal un poco antes que el mensaje pase de largo. La mayor parte de estos retardos son causados cuando el dispositivo MIDI pasa los mensajes recibidos al microprocesador, en vez de enviarlos directamente del MIDI in al MIDI THRU. El microprocesador lee los mensajes y entonces los envía por el MIDI THRU, y a veces agrega mensajes propios. Una vez que el mensaje MIDI pasa a través de más de tres o cuatro puertos THRU con pequeños retrasos, el mensaje tiene un retraso suficiente para producir un claro retraso audible (Este retraso es llamado MIDI lag).

El MIDI lag causa problemas de tiempo en la ejecución de la música. Si se están tocando notas en un sintetizador maestro como inicio de una gran cadena MIDI con MIDI lag, las notas que vienen desde el sintetizador hasta el final de la cadena, sonarán después que las notas que produce el sintetizador maestro. Se podrá escuchar un defasamiento en el inicio de cada nota. Afortunadamente, la mayoría de los dispositivos actuales, no retrardan la señal en sus puertos THRU (una característica que choca con las especificaciones del MIDI). Como resultado, el MIDI lag no es un problema, y se pueden crear cadenas tan largas como se deseen.

RED EN ESTRELLA

No siempre se pueden usar daisy chain para crear sistemas MIDI porque no todos los dispositivos tienen puertos THRU. Otro camino para conectar una serie de dispositivos esclavos a un dispositivo maestro, es creando una red de estrella como la mostrada en la figura 2.10. La red de estrella evita el uso de puertos MIDI THRU en una cadena de dispositivos, agregando un dispositivo llamado MIDI THRU box.

Un MIDI THRU box tienen al menos una entrada y un renglón de puertos THRU. Los mensajes MIDI que entran en el MIDI IN son copiados y pasados simultaneamente a los MIDI THRU de la caja. Conectando el MIDI out del dispositivo maestro a el MIDI in del MIDI THRU box y el MIDI in de cada dispositivo esclavo a un MIDI THRU de la caja, se transmiten mensajes del dispositivo maestro, simultaneamente a cada uno de los dispositivos esclavos.

Una cadena daisy chain o estrella, tiene distintas limitaciones: Los mensajes MIDI viajan solo en una dirección, del dispositivo maestro a los dispositivos

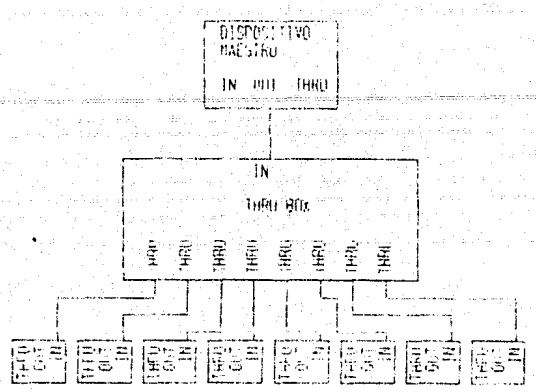


FIG 2.10

esclavos. No es posible para todos los dispositivos en una cadena o en una estrella comunicarse con cualquier dispositivo en la cadena o en la estrella. Por ejemplo, consideremos el tercer dispositivo en la cadena de la figura. El puede recibir mensajes solo del dispositivo maestro, pero no puede transmitir ningún mensaje. Para conectar cinco dispositivos y que todos puedan enviar y recibir mensajes desde y hacia cualquier otro dispositivo, se necesitarían cuatro MIDI IN y cuatro MIDI OUT en cada dispositivo y 20 cables MIDI para hacer las conexiones, cara y no práctica solución, eso si se pudieran encontrar cinco dispositivos MIDI con los puertos necesarios.

RED EN ANILLO

Una red en anillo, como la mostrada en la figura 2.11, permite a los dispositivos MIDI recibir y transmitir mensajes a cualquier dispositivo de la red sin requerir muchos cables y puertos. El MIDI OUT de cada dispositivo en el anillo es conectado a el MIDI IN del siguiente dispositivo.

Una red en anillo, no trabaja con la mayoría de los dispositivos MIDI, sin embargo, algunos dispositivos (principalmente computadoras) tienen un microprocesador que es lo suficientemente rápido y poderoso que toma los datos recibidos en el MIDI IN, y los mezcla con sus propios mensajes, entonces transmite la mezcla por el MIDI OUT. Si cada uno de los dispositivos del anillo cuenta con esta característica, cualquier mensaje enviado por un dispositivo en el anillo será recibido y transmitido por los otros dispositivos, a todos los dispositivos de la red.

Existen varios problemas con esta configuración, Por ejemplo, algún dispositivo envía un mensaje sobre la red, y recibe el mismo mensaje en su MIDI IN una vez que el mensaje a pasado por el anillo. Si el dispositivo original no es capaz de reconocer sus propios mensajes, pasará el mensaje otra vez al siguiente dispositivo, y el mensaje estará dando vueltas a través del anillo sin parar. Estos mensajes, pueden crear una realimentación que sobrecargue la red. Cada

dispositivo en esta red debe ser programado para reconocer sus propios mensajes, cuando ellos regresan a él, no debería enviarlos de nuevo. El MIDI lag puede ser también un problema, por razones discutidas anteriormente.

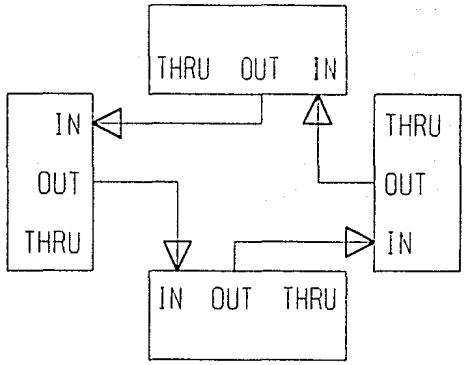


FIGURA 2.11

Una red en anillo, no es un camino muy práctico para los dispositivos MIDI, debido a que la mayoría de ellos no tienen la característica de mezclar los datos recibidos con mensajes propios. Con un software apropiado, muchas computadoras pueden ser conectadas en anillo, pero la mayoría de la gente no utiliza sistemas con solo computadoras. Como la nueva generación de sintetizadores están teniendo más capacidades de programación, tal vez en un futuro se puedan construir redes en anillo, que constituyan un sistema muy flexible.

La figura 2.12 muestra un sistema completo de grabación controlado por MIDI. Como se puede apreciar el sistema MIDI se encuentra configurado como Daisy chain, aunque con una pequeña variante, ya que las salidas de los dispositivos MIDI se encuentran conectadas a un selector (comercialmente conocido como MIDI patcher) por medio del cual podemos seleccionar manualmente un dispositivo como entrada al secuenciador. La mayor parte del tiempo, la entrada al secuenciador será el teclado sin embargo, los demás dispositivos como generadores de tono, máquinas de percusiones y demás dispositivos MIDI que no cuentan con teclado pueden ser entradas al secuenciador ya que, aunque no son capaces de transmitir mensajes MIDI como inicio de nota y fin de nota, pueden transmitir mensajes de sistema exclusivo. Esto puede ser de gran utilidad, por ejemplo, la mayor parte de los generadores de tono, máquinas de percusión y sintetizadores no cuentan con algún medio de almacenamiento masivo de información, así que si se desea tener una gran cantidad de timbres en un generador, es necesario comprar cartuchos de RAM (si el dispositivo MIDI es capaz de operarlos) que son generalmente caros y difíciles de conseguir, una solución a esto, es utilizar el medio de almacenamiento masivo de información del secuenciador, como almacenamiento de timbres de los generadores de tono. También es posible transmitir información de un sonido en particular de cada generador; una buena idea es almacenar los timbres que se utilizarán en una melodía como inicio de una

secuencia en el secuenciador, así cada que se inicie una cesión, no será necesario ajustar de nuevo los generadores de tono ya que la información requerida se encuentra almacenada al inicio de la secuencia a ejecutar. Lo anterior puede ser realizado sin realambrar el sistema gracias a el selector de entradas MIDI.

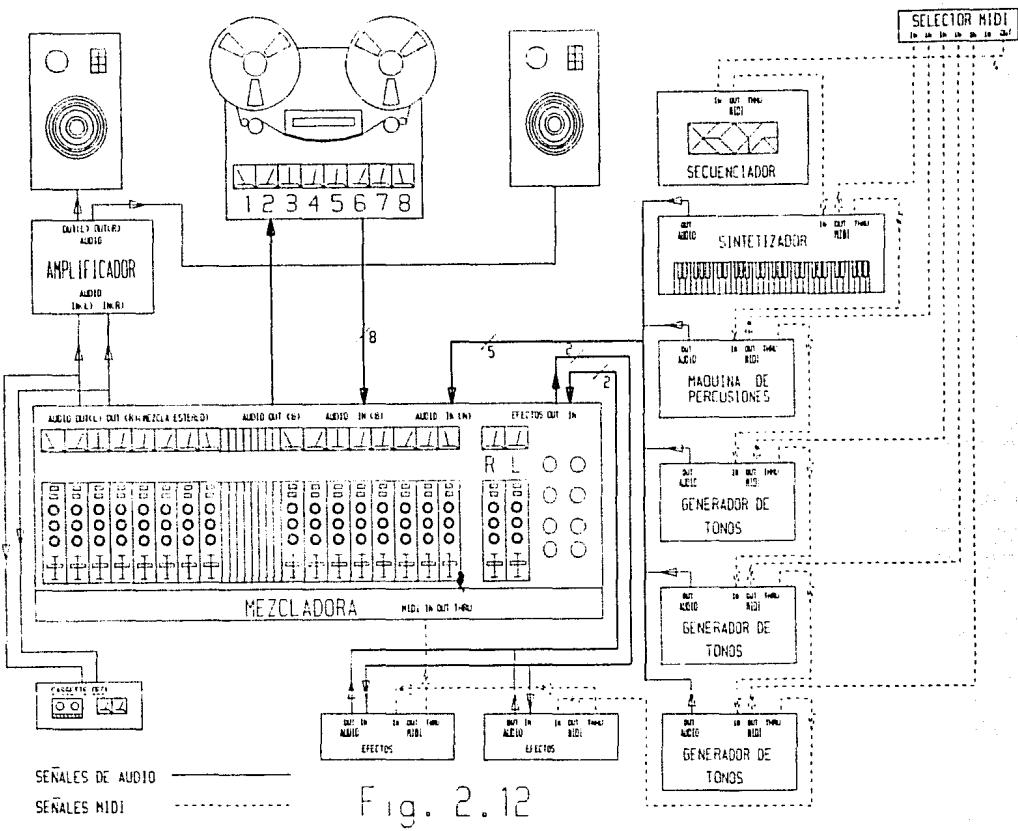


Fig. 2.12

En la figura se puede apreciar también que el secuenciador tambien controla dos unidades de efectos (como eco, compresores, reverberación, flanger, choros etc. etc. etc...), el secuenciador puede enviar cambios de programa a mitad de una secuencia que seleccionen cierto efecto a modifiquen ciertos parametro en las unidades de efectos. Existen algunas mezcladoras como la DMP7 de Yamaha que pueden ser controladas por señales MIDI. Estas mezcladoras cuentan con un determinado número de memorias que almacenan el estado de la mezcladora, de esta forma, al enviar un cambio de programa desde el secuenciador se pueden variar todos los parámetros de la mezcladora, logrando efectos dificilmente de logrados manualmente.

Con todos estos dispositivos MIDI, pudiera parecer que los 16 canales con que cuenta el estandar MIDI son insuficientes, por esto, la mayor parte de los secuenciadores MIDI e interfaces de computadora, que estan saliendo al mercado, cuentan con dos o mas salidas MIDI, estas salidas son por supuesto independientes y asi un sistema grande puede ser seccionado en varios subsistemas, evitando que se cargue toda la información en un sistema.

Como se puede apreciar, el secuenciador es una excelente opción para automatizar un estudio de grabación, sobretodo si se cuenta con dispositivos que permitan sincronizar al secuenciador con la cinta de grabación.

En la actualidad ha surgido una gran cantidad de Hardware y Software que permiten que una computadora personal pueda realizar todas las funciones de control de un sistema como el descrito.

CAPITULO III

Para determinar los elementos que constituirán el Hardware del secuenciador, es necesario determinar primero las funciones que serán realizadas por él, para conocer los requerimientos y poder seleccionar dispositivos adecuados para su construcción.

1) El sistema debe ser capaz de transmitir y recibir datos vía MIDI. Los mensajes MIDI son transmitidos usando una comunicación serial asincrona de datos estandar (un bit de inicio, 8 bits de datos y un bit de paro), tal como lo hace una computadora personal al transmitir mensajes a un Modem. Por esto, el sistema debe contar por lo menos con un puerto serie asincrono, que sea capaz de transmitir a 31,250 bits/segundo.

2) Si se desea que el secuenciador sea fácil de manejar y que cuente con diversas caracteristicas como edición de notas, monitoreo de canales, despliegue de funciones, menus etc. , entonces debe tener la capacidad de desplegar una cantidad considerable de datos. Esto se realizará a través de una pantalla de displays alfanuémericos de 2 lineas con 16 caracteres por linea (así lo hacen casi todos los secuenciadores comerciales). Por supuesto si solo se desea que el secuenciador realice las funciones elementales como grabar y reproducir, entonces se pueden usar menos displays e incluso hasta led's, pero esta opción limitaría mucho las capacidades del secuenciador.

3) El sistema debe contar también con un teclado que permita seleccionar funciones y dar entrada a datos necesarios para la grabación, reproducción, edición, almacenamiento etc.

4) Debe contar con una cantidad considerable de memoria RAM, ya que son necesarios al menos cuatro bytes para almacenar una nota y si el secuenciador no cuenta con una buena cantidad de RAM, se podrán grabar solo canciones muy cortas

5) Ya que el sistema grabará música, debe contar con un dispositivo que interrumpa al microprocesador a intervalos programados.

Una vez conociendo los requerimientos fisicos que se deben cumplir, el siguiente paso es seleccionar el procesador que se utilizará. Existen diversos criterios que se pueden aplicar para la selección del microprocesador; para éste caso en particular, no es necesario que el microprocesador cuente con alguna caracteristica especial, por lo que el sistema se implementará con un microprocesador 68000 de Motorola, debida a que se cuenta con el apoyo de un software que permite ensamblar y correr programas para este microprocesador.

CIRCUITO DE RELOJ

El reloj está compuesto por un cristal de 8Mhz, tres inversores dos capacitores y tres resistencias, conectados como se muestra en la figura 3.1, la salida es conectada al contador IC27, que divide dicha frecuencia y nos proporciona las siguientes frecuencias: 4Mhz, 2Mhz, 1Mhz, 500khz, 250khz, 125khz, 63.5 khz y 31.25 khz. Estas frecuencias, nos sirven como reloj de transmisión y recepción para los UART's y para generar las señales de Data Transfer Acknowledge.

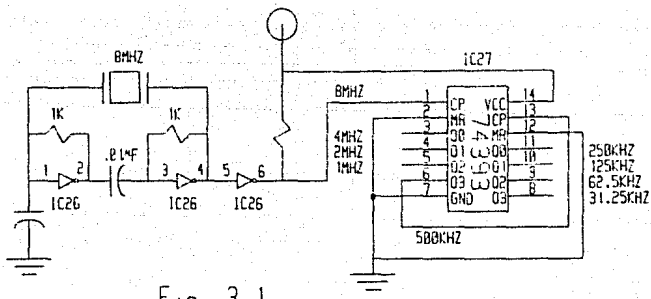


Fig. 3.1

ROM

Cuando el 68000 inicia su operación, lee el vector número cero de la tabla de vectores, que corresponde a la dirección $0 y carga cuatro bytes en el supervisor stack pointer, después, lee el vector número uno, que corresponde a la dirección $4 y carga el contador de programa con las direcciones 4, 5 ,6 , y 7. Además de estos vectores, el 68000 tiene la capacidad de direccionar 256 vectores, que son localidades de memoria de donde el procesador lee direcciones de alguna rutina de interrupción. Esta tabla de vectores de interrupción, está localizada de la dirección $000 a la $3FF, por lo que esta zona debe ser de ROM. Para fines de diseño, se pondrá memoria ROM de la dirección $0 a la $8 y de la $00100000 a la 00101FF, y de la dirección $8 a la $FFF se pondrá RAM, esto debido a que no se conocen aún las direcciones de las rutinas de excepción, y es mucho más fácil enviar la tabla de vectores a través de una computadora, que grabar EPROMS cada vez que se asigne una dirección a cierta rutina, además, una vez diseñado el software esta memoria puede ser sustituida por ROM. Por otro lado, esto nos deja 3 Kilo bytes de RAM libres, para correr pequeños programas de prueba que ahorrarán tiempo y harán más fácil el desarrollo del sistema . En las primeras direcciones de ROM se grabará el supervisor stack pointer y el contador de programa, y de $100000 a $101FF el programa que controle el secuenciador.

Para implementar lo anterior, se usarán dos RAM's 2716 de la dirección $8 a la $FFF y dos ROM's 2732 de $0 a $7 y de $100000 a $10FFFF con las direcciones $0 a $7, repetidas de $100000 a $100007. A continuación se presenta el desarrollo del decodificador de la ROM, esto se puede apreciar mejor a través de la figura 3.2

```
A23,22,21,20  19,18,17,16  15,14,13,12  11,10,9,8  7,6,5,4  3,2,1

 0  0  0  0   0  0  0  0    0  0  0  0   0  0  0  0   0  0  0  0   0  0  0
 0  0  0  0   0  0  0  0    0  0  0  0   0  0  0  0   0  0  0  0   0  1  1

A23,22,21,20  19,18,17,16  15,14,13,12  11,10,9,8  7,6,5,4  3,2,1

 0  0  0  1   0  0  0  0    0  0  0  0   0  0  0  0   0  0  0  0   0  0  0
 0  0  0  1   0  0  0  0    0  0  0  1   1  1  1  1   1  1  1  1   1  1  1
```

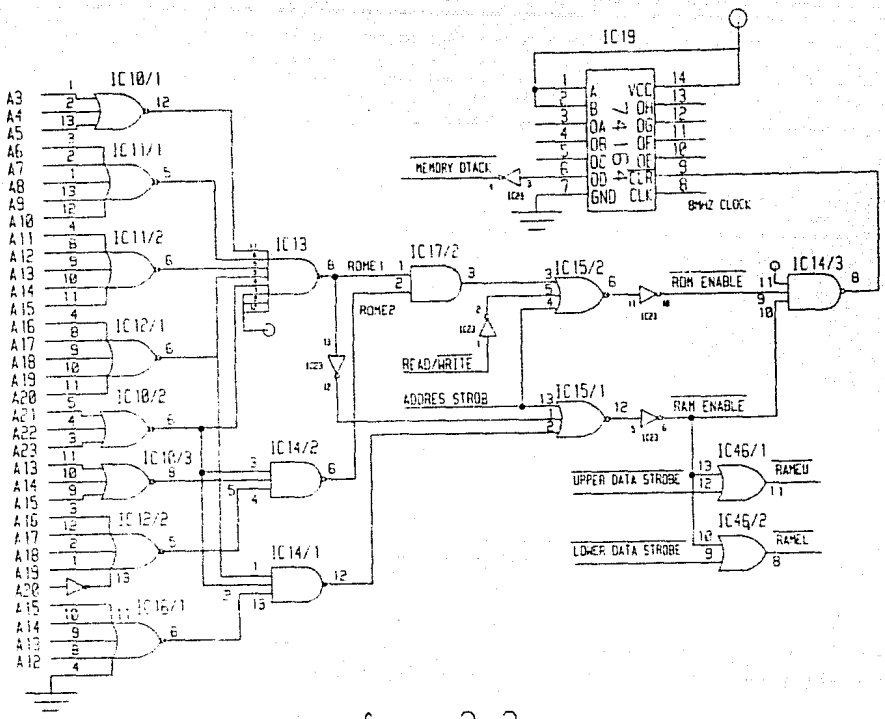


fig. 3.2

Para generar la primera función (de $0 a $7), las líneas de direcciones desde A3 hasta A23, son conectadas a las compuertas NOR 10/1, IC11/1, IC11/2, IC12/1, IC12/2 e IC10/2, si todas las entradas a una compuerta son ceros, las salidas de las compuertas serán uno; las salidas de estas compuertas NOR, son conectadas a una compuerta NAND de 8 entradas IC13 con las entradas sobrantes conectadas a VCC; si todas las entradas son "1", entonces la salida de IC13 irá a "0" y provocará que la salida de IC17/2 también vaya a cero, si además *AS="0" y *R/W="1" entonces la salida de IC15/2 irá a "1" y la pata 10 de IC23 irá a "0" con lo que se activara la ROM.

La segunda función, de $100000 a $101FFF se genera de la siguiente manera: A13,14,15,15,17,18,19,*20,21,22 y 23 son conectados a IC10/2, IC10/3 e IC12/2, si todas las entradas son "0", entonces las salidas de las compuertas irán a "1" y provocarán que la pata 6 de IC14/2 vaya a "0", con esto la salida de IC17/2 irá a "0" y si además de esto las señales *AS y *R están activas, la compuerta IC15'2

provocarán que la salida 10 de IC23 se vuelva cero y active la ROM.

RAM : Esta memoria debe activarse de la dirección $8 a la dirección $FFF; para facilitar la decodificación, se considerará que la memoria enciende de $0 a $FFF y, se condiciona la salida a la inactividad de la señal que activa la ROM de la dirección $0 a $7 (ROME1).

A23,22,21,20	19,18,17,16	15,14,13,12	11,10,9,8	7,6,5,4	3,2,1
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0
0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1	1 1 1

Las lineas de A16 hasta A23 entran a las compuertas NOR IC12/1 e IC10/2, ya usadas en la decodificación de la ROM, las lineas de dirección desde A15 hasta A12 son conectadas a IC16/1; si las lineas de A12 a A23 son cero, entonces las salidas de IC12/1, IC10/2 e IC16/1 irán a uno y provocarán que la salida de IC14/1 vaya a cero, y si además *AS=0 y la pata 8 de IC13 (ROME1) se encuentra en uno (la ROM no se encuentra activa de $0 a $7), la salida de IC15 se volverá "1" y esto traerá con sigo que la salida 6 de IC23 sea cero, esta señal es conectada a las compuertas NOR IC46/1 e IC46/2, con esto y *LDS y *USD, se activa la RAM correspondiente a la parte del bus que se desea leer.

Además de la decodificación, es necesario un circuito que active la señal DATA TRANSFER ACKNOWLEDGE (*DTACK), que indica al procesador que la transferencia de datos ha sido concluida. Cuando el procesador reconoce *DTACK durante un ciclo de lectura, los datos son latchados del bus y el ciclo termina. Cuando el procesador reconoce *DTACK en un ciclo de escritura, el ciclo termina y los datos son retirados del bus.

Para este efecto, son agregados los circuitos IC14/3 e IC19, si se realiza algún acceso ya sea a la ROM o a la RAM, la salida de IC14/3 irá a cero y con esto, se empezará a correr un uno en el registro de corrimiento IC19 y, al quinto ciclo de reloj, se dará salida a la señal MEMORY DTACK en la pata 4 de IC23.

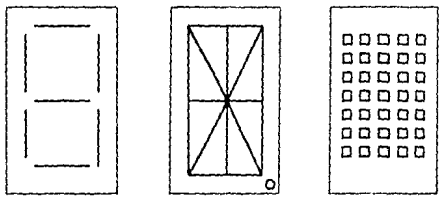
La terminal *CE de la ROM, tanto de la parte baja como de la parte alta del bus es conectada a la señal *ROM ENABLE (salida 10 de IC23). La señal ENABLE de la RAM correspondiente a la parte alta del bus, es conectada a la señal *RAMEU (pata 11 de IC46/1), y la misma señal de la RAM correspondiente a la parte baja del bus, es conectada a *RAMEL (pata 8 de IC46/2).

DISPLAY

Como se comentó anteriormente, el secuenciador debe tener la capacidad de edición de notas, por lo que debe contar con un dispositivo capaz de desplegar algunos mensajes como: tipo de dato recibido (nota on, nota off, after touch, sysex, etc.), valores de los datos recibidos, tiempo en que se recibió el dato, función actual del secuenciador, etc. Por esta razón, y para que el manejo del secuenciador no sea muy confuso, se decidió poner dos lineas de 16 caracteres cada una (como lo tienen muchos secuenciadores comerciales), para que los mensajes del secuenciador sean claros.

En dispositivos que requieren desplegar una gran cantidad de datos, usualmente se utiliza un CRT, pero en dispositivos que despliegan una cantidad considerablemente menor de datos, es común utilizar, algún tipo de display numérico o alfanumérico. Para este efecto, existen basicamente dos tipos de tecnologias, los LEDs (diodos emisores de luz) y los LCDs (displays de cristal liquido). Los displays de cristal liquido, utilizan muy poca potencia y son utilizados generalmente en dispositivos portátiles (con fuente de poder propia), los LCDs, no emiten una luz propia, sino que simplemente cambian la reflexión de la luz disponible. Los instrumentos que se utilizan en condiciones de poca luz deben incluir una fuente de luz para los LCDs o utilizar LEDs que emiten su propia luz.

Los displays de LEDs se encuentran disponibles generalmente en tres formatos. Para desplegar unicamente números hexadecimales (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F), se utiliza un display de 7 segmentos, como el mostrado en la figura 3.3a. Para desplegar el alfabeto completo, se utilizan displays de 18 segmentos como el mostrado en la figura 3.3b, o una matriz de 5 X 7 puntos como el mostrado en la figura 3.3c.



3.3a 3.3b 3.3c

En este caso, se utilizaron displays de 18 segmentos, la figura 3.4, muestra un circuito usado para manejar un display de 18 segmentos con ánodo común. En un display de ánodo común, se aplica un cero al Led que se desea encender, cuando se envia un código a los latchs 74373, estos llevan a cero los segmentos que deben encender y a uno los que no deben encender, El display continua desplegando el mismo caracter hasta que otro código sea cargado a los latchs.

Esta configuración, es conocida como "Display estático", porque la corriente fluye a través de los LEDs y permanece constante hasta que se represente otro caracter. Note que se requiere de resistencias conectadas en serie con los LEDs, para limitar la corriente que fluye de las salidas de los latchs hacia los LEDs , cada LED requiere de entre 5 y 30 mili ampers dando una luz apropiada par ser vista. La resistencia debe ser calculada como sigue: asumimos que se desea una corriente de 20mA por LED. El voltaje a través de los LEDs, debe ser de alrededor de 1.5V, un cero lógico en los 74373 debe ser de 0.4V a 40mA, asumimos que el voltaje será de 0.2V a 20mA, restando estos dos voltajes de la fuente de alimentación que es de 5V, nos da 3.3V a través de la resistencia, considerando los 20mA que deben pasar por ella, nos resulta un valor de 168 ohms. Los voltaje a través del LED y

a la salida del 74373 no son exactamente predecibles, por lo que la corriente en el LED no será exactamente de 20mA, pero esto no es muy critico y se puede escoger un valor de resistencia estandar como 150 ohms, sin ningún problema.

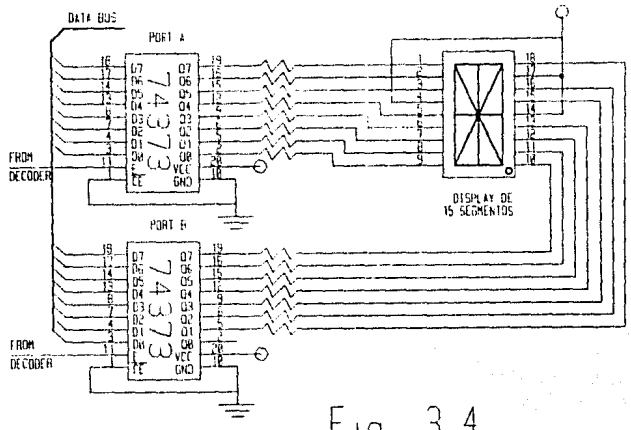


Fig. 3.4

El display de la figura 3.4, es recomendable cuando se utilizan uno o dos displays, pero si se trata una cantidad considerable de displays, el circuito resulta totalmente impráctico, el primer problema, es el consumo de potencia, un cálculo aproximado para el peor caso es el siguiente: supongamos que todos los segmentos de un caracter se encuentran encendidos, y la corriente por LED es de 20mA, este display consumirá 360mA, si además los 32 displays de encuentran encendidos, se consumirán 11.52 ampers, lo cual es mucho mayor de lo que consumen los demás circuitos y la fuente de alimentación pasa a ser un problema. Además de esto, se requerirían 64 latchs 74373, decodificador para ellos, y 576 resistencias. Una solución a esto, es utilizar un display multiplexado, como el mostrado en la figura 3.5. Aqui, todos los displays son conectados en paralelo a los puertos A y B, y se enciende un display a la vez, seleccionándolo a través de alguno de los puertos C, D, E o F. Esto se logra, poniendo a cero el bit correspondiente al display a encender en dichos puertos, al hacer esto, la base del transistor PNP va a cero y el cátodo del display va a uno, encendiendo el display con el código que se encuentre en A y B. Si se enciende un display después de otro, con el caracter correspondiente a la posición del display, a una velocidad razonable (de 40 a 200 veces por segundo), el ojo humano verá todos los displays encendidos al mismo tiempo.

El tiempo que los displays permanecen apagados respecto al tiempo que permanecen encendidos, se incrementa con el número de displays y entonces para obtener un brillo adecuado, se debe incrementar la corriente a través de los LEDs cuidando de no exceder el límite máximo que recomiendan el fabricantes del display. Si la cantidad de displays es grande, la corriente crece hasta un valor que probablemente rebasa la corriente máxima de salida de los latchs, por esta razón, puede ser necesario poner transistores a la salida de los puertos A y B para

alimentar a los displays con una corriente suficiente.

Las ventajas de un display multiplexado, son obvias, sin embargo, requiere de un cierto refresco, y si el trabajo del microprocesador involucra alguna tarea que no pueda ser interrumpida, se verá solo un caracter en el display. En tal caso, se podría utilizar algún dispositivo dedicado al display como el 8279 que permite visualizar 8 digitos y al mismo tiempo permite acoplar un teclado de matriz, sin mayores complicaciones

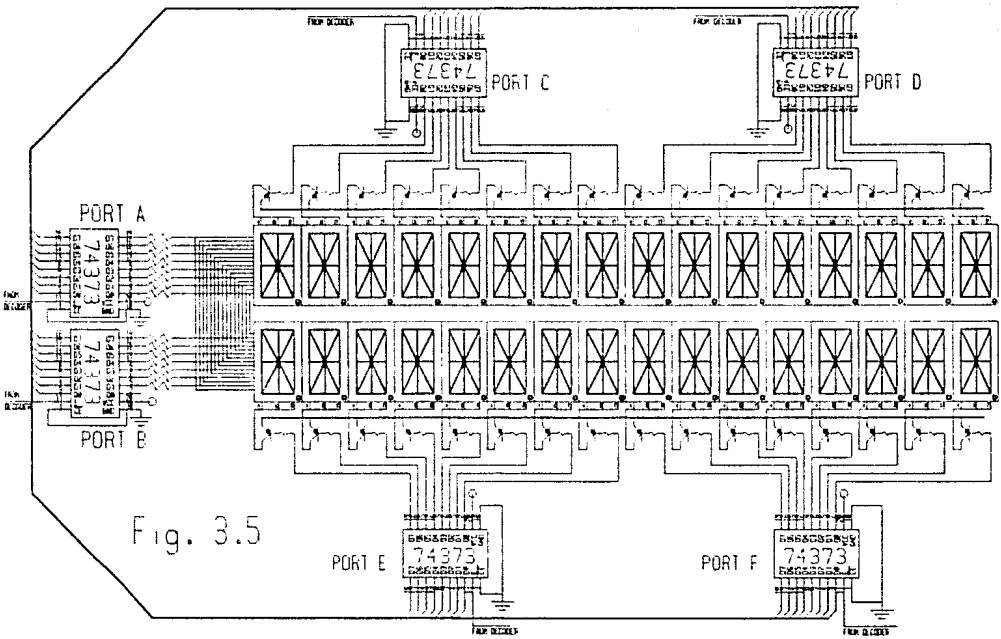
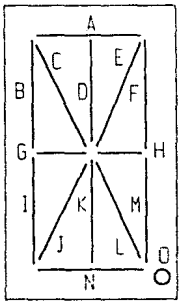


Fig. 3.5

Ahora veamos como funciona la pantalla que se implementó para el secuenciador; como ya se mencionó al inicio de este capitulo, la pantalla contará con dos líneas de 16 caracteres cada una y se utilizarán displays (de LEDs) de 15 segmentos, la figura 3.5 muestra la configuración de la pantalla. Los puertos A y B se encuentran conectados a los cátodos de cada segmento en todos los displays. Como los displays son de ánodo común, un cero lógico en una línea de los puertos A y B causará que el segmento conectado a ella se encienda, mientras que un uno mantendrá apagado el segmento. Por otro lado, si se encuentra presente un cero en alguna línea de los puertos C, D, E o F, se encenderá el display que se encuentre conectado a esa línea y desplegará el caracter correspondiente al código en A y B.

La figura 3.5b muestra la correspondencia entre los segmentos de un display y los bits de los registros A y B. Con esta información, es posible determinar el código correspondiente a cada letra en los registros A y B. Por ejemplo, para representar una letra "U", es necesario encender los segmentos B, F, I, N, y M; Los bits correspondientes a estos segmentos deberán encontrarse en cero y los bits restantes en uno, por lo que el código para el registro A será D5H (11010101) y DBH (11011011) para el registro B. En el listado compilado del software (IV-25) se encuentra la información completa de los códigos en los registros A y B y su equivalente en ASCII. Los registros C, D, E y F encienden a los displays si se encuentra un cero en la linea del display que se desea encender.



REGISTRO A	D7	D6	D5	D4	D3	D2	D1	D0	BIT
	D	C	B	G	I	J	N	K	SEGMENTO
REGISTRO B	D7	D6	D5	D4	D3	D2	D1	D0	BIT
	A	E	F	H	O	M	L	*	SEGMENTO

Fig. 3.5b

TECLADO

Antes de tomar alguna decisión de la forma de implementar el teclado, es necesario analizar los diferentes tipos, así como sus ventajas y desventajas.

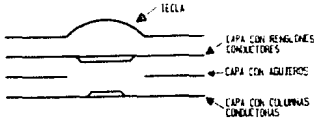
TECLADOS CON SWITCHES MECANICOS.- En estos teclados, dos piezas de metal son juntadas cuando se presiona alguna tecla, el material usado, generalmente es fósforo de bronce, con aleaciones de oro o plata solo en las partes de contacto; estas teclas cuentan generalmente con un pequeño resorte que ayuda a regresar las placas de metal a su posición original cuando se suelta la tecla y tal vez una pieza de espuma para tratar de disminuir el rebote. Los switches mecánicos, son relativamente baratos, pero tienen diversas desventajas. Primero, sufren de un rebote de contacto. Al presionar una tecla, se rompe el contacto varias veces antes de tener una conexión firme. Segundo, el contacto puede oxidarse o ensuciarse y muy pronto se tendrá una pésima conexión. Los switches mecánicos de alta calidad, tienen una vida útil de alrededor de 1 000 000 de contactos.

TECLADOS DE MEMBRANA.- Estos switches son en realidad, un caso particular de los switches mecánicos. Consisten en tres capas de plástico o goma, a manera de sandwich como se muestra en la figura 3.6a. La capa superior tiene una linea de tinta de plata, corriendo bajo cada renglón de teclas. La capa inferior tiene una linea de tinta

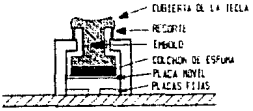
de plata corriendo bajo cada columna de teclas. Cuando se oprime una tecla, se junta la línea de tinta de la capa superior, con la línea de tinta de la capa inferior, a través de un agujero de la capa central. La ventaja de estos teclados, es que debido a su poco volumen, son muy fáciles de instalar en dispositivos portátiles; sin embargo, no tienen mucho tiempo de vida útil.

TECLADOS DE CAPACITANCIA.- Como se muestra en la figura 3.6b, un switch de capacitancia, tiene dos pequeñas capas de metal en el circuito impreso y otra en la parte inferior de la capa de espuma. Cuando se oprime una tecla, la placa móvil, es llevada cerca de las placas fijas. Esto cambia la capacitancia entre las placas fijas, un circuito detecta este cambio de capacitancia y genera un nivel lógico para indicar que una tecla ha sido oprimida. La gran ventaja de este tipo de teclados, es que no tienen contactos mecánicos que puedan oxidarse o ensuciarse. Una pequeña desventaja es que necesitan de un circuito especial que detecte el cambio de capacitancia. Este tipo de teclados tiene una vida útil de alrededor de 20 000 000 de disparos.

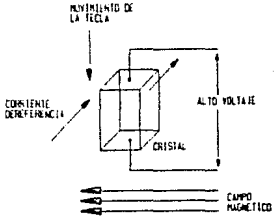
TECLADOS DE EFECTO HALL.- Este es otro tipo de Switchs que no tiene contactos mecánicos. Se toma ventaja del movimiento en un campo magnético, la figura 3.6c ilustra cómo funciona. Una corriente de referencia pasa a través de un cristal semiconductor. Cuando se presiona alguna tecla, el cristal se mueve a través de un campo magnético cuyas líneas de flujo son perpendiculares a las de la corriente que circula en el cristal. Este movimiento, causa que aparezca un pequeño voltaje en los extremos de el cristal, el voltaje se amplifica y es usado para indicar que se ha oprimido una tecla. Los teclados de efecto Hall, son más caros porque el mecanismo para mover las teclas es mas complejo, pero son muy confiables y tienen una vida útil de 100 000 000 o más contactos.



3.6a



3.6b



3.6c

La forma mas sencilla de conectar un teclado, es asignando cada tecla a una línea de entrada de un puerto acoplado al microprocesador. De esta manera, las teclas pueden ser leídas rápidamente, y el software necesario es trivial, pero esta configuración es solo práctica, cuando se tienen teclados con un número reducido de teclas.

En la mayor parte de los teclados, las teclas son conectadas en una matriz de renglones y columnas, como el mostrado en la figura 3.7, en este teclado se usan switchs mecánicos, pero es el mismo principio para los diferentes tipos de switchs. Esta configuración, aprovecha

mejor las líneas de los puertos, pero requiere de un software más complejo. El número de teclas, se maximiza con una matriz cuadrada, por ejemplo, una matriz de 8X8, puede leer 64 teclas, mientras que una de 12 X 4, puede leer solamente 48.

Para obtener un caracter del teclado, es necesario realizar básicamente tres tareas:

1) Detectar que se ha oprimido una tecla.

2) Quitar el rebote a la tecla oprimida

3) Codificarla (Producir un código estandar para la tecla oprimida)

Estas tres tareas pueden ser hechas por hardware, software o una combinación de ambos, dependiendo de la aplicación. En sistemas donde el microprocesador no se encuentra presionado por el tiempo, se puede leer el teclado por medio de software.

La figura 3.7, muestra como se puede conectar un teclado hexadecimal a dos puertos acoplados a un microprocesador, con esta configuración, se pueden realizar los trabajos anteriores como parte del programa. Los cuatro renglones de la matriz, son conectados a

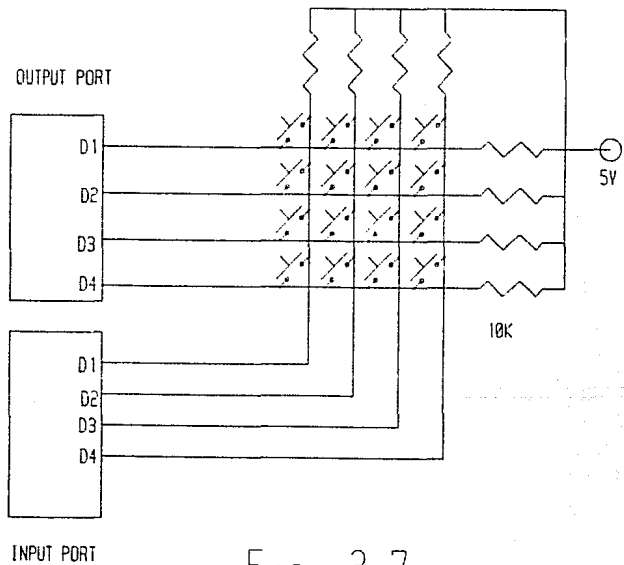


Fig. 3.7

cuatro líneas de salida de un puerto. Las cuatro columnas son conectadas a cuatro líneas de entrada. El principio aquí es que al

apretar una tecla se juntan un renglón y una columna. Cuando ninguna tecla está apretada, las líneas de las columnas son mantenidas en alto por la resistencias conectadas a 5 volts. Si un cero se encuentra presente en algún renglón, y se presiona una tecla de ese renglón, entonces el cero aparecerá en la columna que contiene la tecla que fue presionada y puede ser detectada en el puerto de entrada. Si se conoce el renglón y la columna de la tecla presionada entonces se conoce cual tecla fue oprimida, y se puede asignar un código para representar esta tecla. En la figura 3.8, se muestra un diagrama de flujo de una subrutina para detectar, quitar el rebote (debounce) y asignar un código a una tecla.

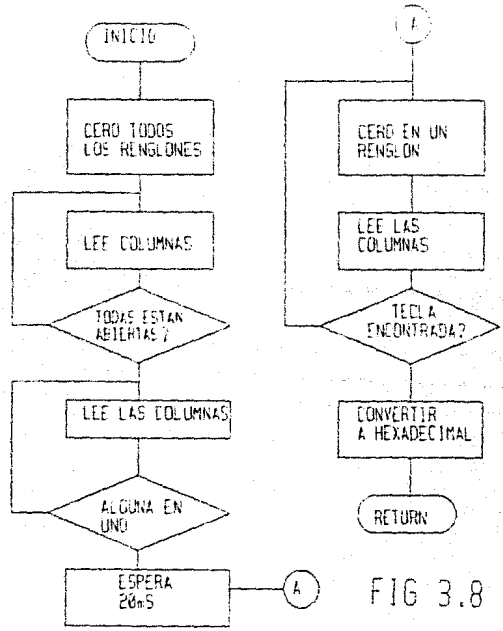
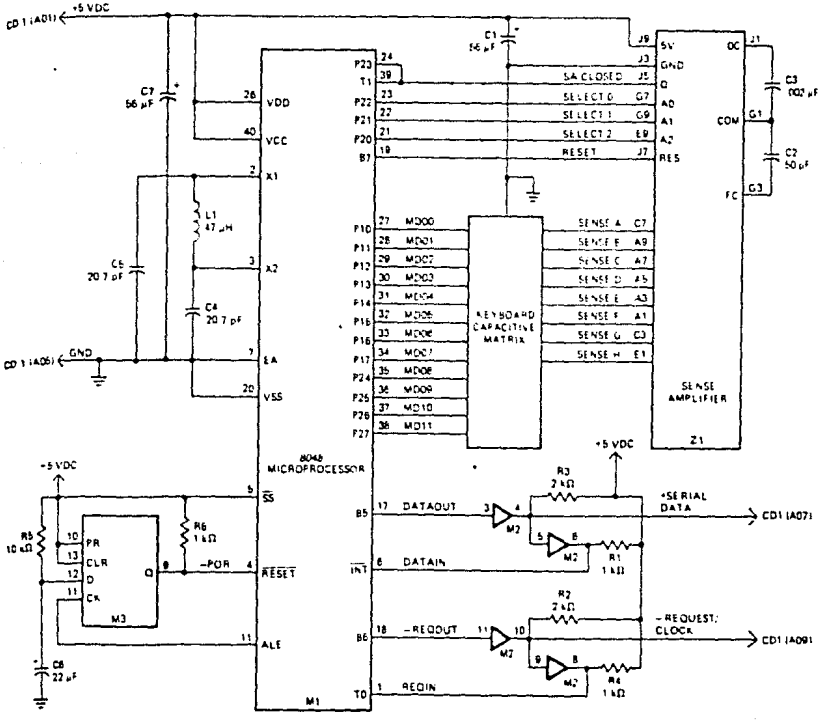


FIG 3.8

El primer paso es poner a cero todos los renglones. Después, las columnas son leídas una y otra vez hasta que todas se encuentran en uno. Esto se hace para estar seguros de que la tecla previa ha sido soltada, antes de intentar leer la siguiente. En terminología de teclados, esto es llamado tow-key lockout. Una vez que todas las columnas están en alto, el programa entra en otro loop en el que espera a que alguna columna vaya a cero, indicando que una tecla fue presionada. Una espera de 20 ms es hecha entonces, para que las condiciones de la tecla se estabilicen (se espera a que termine el rebote).

Después de este tiempo, se checa si todavía se encuentra una columna en cero. Si las columnas están todas en alto, entonces no hay ninguna tecla oprimida y la detección inicial, fue un pulso de ruido. Si cualquiera de las columnas se encuentra aún en cero, entonces efectivamente una tecla fue oprimida.

El último trabajo es determinar el renglón y la columna de la tecla que fue oprimida y convertir esta información, a un código hexadecimal correspondiente al la tecla oprimida. Para obtener información acerca del renglón y la columna, se da salida a cero, solo en una columna, y todas las columnas son leídas. Si ninguna columna está en cero, entonces la tecla no pertenece a ese renglón, entonces el cero es llevado al siguiente renglón y todas las columnas son checadas de nuevo. Este proceso es repetido hasta que un cero en un renglón produzca un cero en una columna. La tecla presionada se encuentra en el renglón que produce un cero y en la columna que es cero. Finalmente, esta información puede ser convertida facilmente a un código ASCII o cualquier otro código por medio de una tabla.



La mayor parte de las computadoras, utilizan un teclado con un microcontrolador dedicado a esto, el 8048, contiene una pequeña ROM, en la cual se encuentra grabado un programa, que realiza todo el proceso para leer una tecla, esperar el tiempo de rebote y codificar ésta en ASCII o algún otro código. En algunos teclados, se mandan dos códigos por cada tecla, uno cuando la tecla es oprimida, y otro cuando es soltada, ésto, con el objeto de dar mayor flexibilidad al software. La transferencia entre el teclado y la computadora, es realizada generalmente en serie, por utilizar solo 3 hilos para ello (alimentación, tierra y datos). Las ventajas de utilizar un teclado de este tipo, son obvias; en primer lugar, se libera al microprocesador de efectuar esta tarea y en segundo lugar, no es necesario construir y probar los amplificadores que detectan el cambio de capacitancia en las teclas. Además de esto, debido al volumen de producción de este tipo de teclados, el precio es extremadamente bajo. Un teclado de 126 teclas, puede ser comprado en México por unos 15 dolares, mientras que tratar de implementar un teclado de matriz con teclas de membrana, tiene un costo de unos 40 dolares, con la correspondiente pérdida de tiempo del microprocesador, al efectuar este trabajo y la poca vida útil de estos teclados. Por esta razón, se utilizará un teclado de PC, como el mostrado en la figura 3.9 . Para conectar este teclado, únicamente se necesita un puerto serie, como el 6850, recibiendo a 9600 bits por segundo.

PERIFERICOS

El sistema debe contar con dos UARTs, uno para manejar las señales MIDI y otro para leer caracteres del teclado. También debe incluir un TIMER que interrumpa al microprocesador a intervalos de tiempo programados. Para manejar el display (figura 3.5), son necesarias 48 líneas de salida, las cuales serán implementadas con seis latchs 74LS373.

Los UARTS, y el timer, serán de la familia de periféricos de el 6800, ya que no existen complicaciones para acoplarlos al 68000. El 6850 (figura 3.10a), ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA), es un dispositivo de 24 patas, que cuenta con cuatro registros, que pueden ser accesados, en dos direcciones, esto es: Cuando la señal *RS, se encuentra en cero, y la señal R/*W en uno, se accesa el registro de estado; con la señal *RS en cero, y R/*W en cero, se accesa el registro de control. Si la señal *RS se encuentra en uno, entonces se tiene acceso a los registros de transmisión y recepción, si se trata de una lectura, se está direccionando el registro de recepción, y si se trata de una lectura, se está direccionando el registro de transmisión.

Por otro lado, el 6840 (figura 3.10b), PROGRAMMABLE COUNTER/TIMER, es un circuito de 28 patas, que cuenta con 18 registros, que pueden ser accesados en ocho direcciones; la manera de accesar estos registros, es algo compleja para explicarse aquí, pero para fines de la decodificación, solo es necesario conocer que se accesa con ocho direcciones (3 bits).

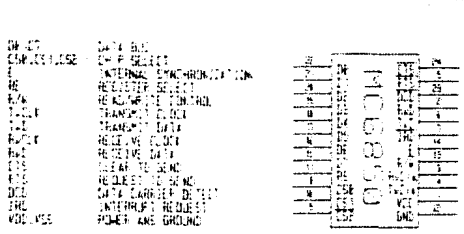


FIG. 3.10a

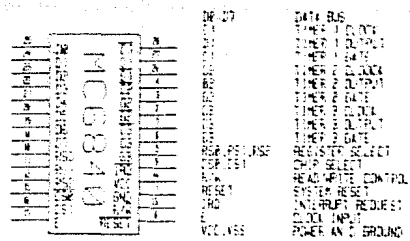
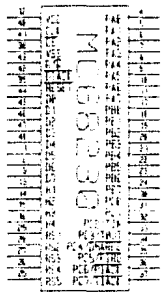


FIG. 3.10b

Por último, el display requiere de 6 direcciones (3 bits), para tener acceso a sus 6 registros (A,B,C,D,E,F).

Con el objeto de que el sistema pueda ser expandido fácilmente, se dejarán 5 bits de direcciones libres (A1,A2,A3,A4,A5), por si se desea conectar algún periférico del 68000, como el 68230 (figura 3.11a), PARALLEL INTERFACE/TIMER (PI/T), o el DUART 68681 (figura 3.11b), y solo se utilizarán 3 lineas (A6, A7 y A8) como entrada para el decodificador, lo cual nos permitirá conectar hasta ocho periféricos.

FIG 3.11a



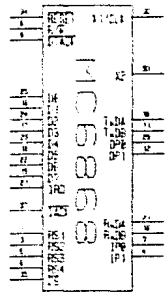


FIG 3.11b

La dirección inicial, de los puertos, será $102000 (Podria se cualquier otra), a continuación se presenta el desarrollo del decodificador.

A23,22,21,20	19,18,17,16	15,14,13,12	11,10,9,8	7,6,5,4	3,2,1
0 0 0 1	0 0 0 0	0 0 1 0	0 0	0 0	0 0 0
0 0 0 1	0 0 0 0	0 0 1 0	0 0	0 1	1 1 1 1 1 1 1

Como se puede observar en la figura 3.12, las lineas A23, A22 y A21 son conectadas a la compuerta NOR IC10/2, ya utilizada, en la decodificación de la ROM y RAM estática (figura 3.2). Las líneas *A20, A19, A18, A17, A16 son conectadas a la compuerta NOR IC12/2, también usada en la decodificación de la ROM. La líneas de direcciones A15, A14, *A13, A12 y A11, son conectadas a la compuerta NOR IC16/2 y por último, las líneas A10 y A9 se conectan a la compuerta IC15/3. Si todas las entradas a las compuertas IC10/2, IC12/2, IC16/2 e IC15/3 son cero, entonces las salidas de estas compuertas, serán uno, y provocarán que la salida de IC20/1 vaya también a uno, provocando que se active el decodificador IC21, siempre y cuando se encuentre activa la señal *AS. En las salidas de IC21, se encuentran las señales que activarán los puertos.

De la salida Y0 de IC21 (pin 15), se saca la señal que activará el display. Como éste no cuenta con un decodificador interno que active los registro en forma independiente, es necesario implementarlo. Para este fin, se encuentra el circuito IC22, que con la señal Y0 de IC21 activa, proporciona salidas independientes para activar cada registro del display. Por último, el registro de corrimiento IC24 generará una señal de *DTACK para indicar al microprocesador, que la transferencia de datos al display ha sido concluida.

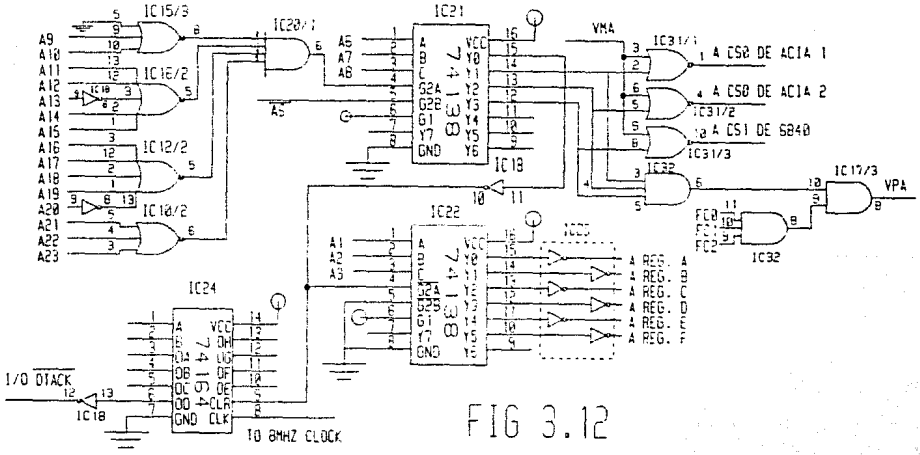


FIG 3.12

Los puertos de la familia del 6800, no requieren de un decodificador que seleccione cada registro, como en el caso del display, ni tampoco necesitan generar alguna señal de *DTACK, ya que la transferencia de datos, será sincronizada con la señal E del microprocesador; sin embargo necesitan una lógica que maneje las señales *VPA y *VMA.

Cuando el microprocesador accesa a algún periférico del 6800, se debe activar la señal *VPA, por medio del decodificador de los puertos, para indicar al microprocesador, que la transferencia de datos debe ser sincronizada con la señal E. Después de esto, el microprocesador genera la señal *VMA , para indicar al decodificador que se ha enterado, y que se debe activar el periférico correspondiente. Esto es implementado por medio de las compuertas IC32/1, IC31/1, IC31/2 e IC31/3.

La compuerta IC32, es necesaria, para generar un autovector, cuando el 68000 recibe alguna interrupción, del timer, o de algún otro dispositivo, esto se explica con detalle, más adelante. A continuación, se presenta una tabla que nos muestra las direcciones en que se encuentran localizados los periféricos:

	Lectura		Escritura
DISPLAY	$0102001		R. A
	$0102003		R. B
	$0102005		R. C
	$0102007		R. D
	$0102009		R. E
	$010200A		R. F
ACIA1	$0102041	R. STATUS	R. CONTROL
	$0102043	R. RECEPCION	R. TRANSMISION
ACIA2	$0102081	R. STATUS	R. CONTROL
	$0102083	R. RECEPCION	R. TRANSMISION
TIMER	$01020C1	R. DEV	R. DEV
	$01020C3	R. DEV + 1	R. DEV + 1
	$01020C5	R. DEV + 2	R. DEV + 2
	$01020C7	R. DEV + 3	R. DEV + 3
	$01020C9	R. DEV + 4	R. DEV + 4
	$01020CB	R. DEV + 5	R. DEV + 5
	$01020CD	R. DEV + 6	R. DEV + 6
	$01020CF	R. DEV + 7	R. DEV + 7

Los registros de los periféricos, quedan localizados en direcciones impares, porque están conectados a la parte baja de el bus de datos (de D0 a D7), y el 68000 accesa esta parte del bus cuando A0 se encuentra en uno.

Aparte de estos dispositivos, tenemos espacio para otros cuatro puertos, cuyos registros podrían ser accesados en las siguientes direcciones.

PUERTO 1	$0102101	R. DEV
	$0102103	R. DEV + 1
	.	.
	.	.
	$010213D	R. DEV + 30
	$010213F	R. DEV + 31
PUERTO 2	$0102141	R. DEV
	$0102143	R. DEV + 1
	.	.
	.	.
	$010217D	R. DEV + 30
	$010217F	R. DEV + 31
PUERTO 3	$0102181	R. DEV
	$0102183	R. DEV + 1
	.	.
	.	.
	$01021BD	R. DEV + 30
	$01021BF	R. DEV + 31
PUERTO 4	$01021C1	R. DEV
	$01021C3	R. DEV + 1
	.	.
	.	.
	$01021FD	R. DEV + 30
	$01021FF	R. DEV + 31

INTERRUPCIONES

Existen tres dispositivos que pueden interrumpir al microprocesador en este sistema, el ACIA 1 (6850) encargado de recibir y transmitir datos por el puerto MIDI, el ACIA 2, encargado de recibir caracteres del teclado, y el TIMER (6840), cuya función es precisamente interrumpir al microprocesador.

El 68000 cuenta con 3 líneas, que dan origen a 7 posibles niveles de interrupción. Como se muestra en la siguiente tabla, si las líneas IPL0, IPL1 e IPL2 son cero, no existe ninguna interrupción, mientras que si las líneas se encuentran en uno, se esta generando la interrupción con más alto valor de prioridad.

PRIORIDAD	CODIGO DE INTERRUPCION		
	IPL2	IPL1	IPL0
-----	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Existen dos formas en las que el 68000 puede obtener la dirección de la rutina de interrupción, del dispositivo que interrumpe. En la primera, cuando se interrumpe al 68000, este usa las líneas de dirección A1, A2 y A3, para indicar el nivel de interrupción a ejecutar; cuando el dispositivo que interrumpe detecta que el código de función corresponde a un ciclo de interrupt acknowledge (FC0, FC1 y FC2 son uno), debe enviar el número de vector donde se encuentra la dirección de la rutina de interrupción, utilizando la parte baja del bus de datos, como si se tratara de un ciclo de lectura de byte. Para utilizar este modo de interrupción, es necesario emplear periféricos que sean capaces de generar su propio vector.

La segunda forma de obtener la dirección de interrupción, es con un mecanismo de autovector. Este mecanismo reduce el hardware necesario para las interrupciones y también el tiempo de respuesta. En este caso, el hardware externo, solo necesita reconocer un ciclo de interrupt acknowledge y llevar a cero la señal *VPA, para indicar al 68000 que se trata de una rutina con autovector. Después el 68000 genera el vector internamente conforme a la siguiente tabla, y captura la dirección de la rutina de las direcciones indicadas:

PRIORIDAD	VECTOR	DIRECCION
1	25	64H
2	26	68H
3	27	6CH
4	28	70H
5	29	74H
6	30	78H
7	31	7CH

La figura 3.13 muestra la forma en que se implementó el mecanismo de autovector. Aquí se encuentran 7 entradas de interrupción en las líneas de datos de IC28, correspondiendo D1 al nivel 1 de interrupción y D7 al nivel 7. Los niveles lógicos de estas entradas, son latchados y sincronizados con la señal de reloj del 68000. Este latch (IC28) se encuentra para sincronizar las señales y aplicarlas al codificador de prioridad IC29.

El circuito IC29 codifica las señales de interrupción en 3 bits, para que puedan ser suministradas al 68000. Note que las entradas a este dispositivo son activas bajas, con la entrada 7 correspondiendo a la más alta prioridad y la entrada 1 correspondiendo a la más baja prioridad. El código binario correspondiente a la entrada de más alta prioridad activa, es puesta en A2 A1 A0, en la salida de IC29. Esta salida es latchada por el circuito IC30 y aplicada a las entradas IPL0, IPL1 e IPL2 del 68000.

Además de esto, es necesario un circuito que detecte los ciclos de interrupción y lleve a cero la señal *VPA. Este trabajo es realizado por la compuerta AND IC32 y el inversor IC26.

Como se puede apreciar en la figura, el TIMER está conectado al nivel 6 de interrupción, el ACIA 1 (puerto MIDI), al nivel 5 y el ACIA 2 (teclado) al nivel 4.

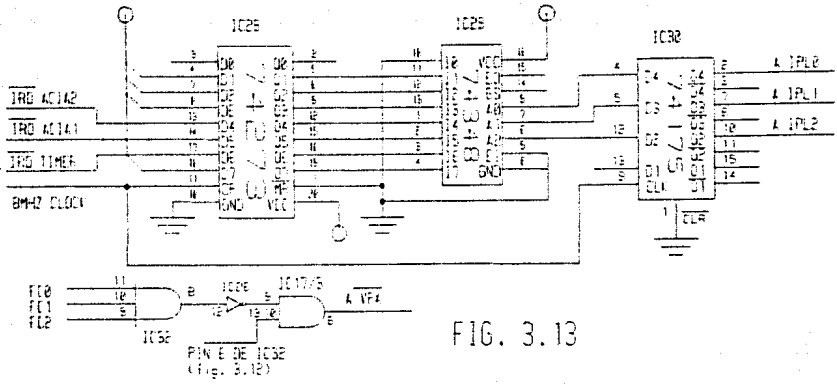


FIG. 3.13

MEMORIA RAM

Por último, el sistema contará con 32 kilo bytes de memoria ram estática que almacenará los mensajes MIDI que sean recibidos por el secuenciador y estará localizada de la dirección $20000 a la dirección $27FFF. Esto es implementado por cuatro memorias 6164 que son memorias estáticas de 8k x 8. La figura 3.14 muestra como se implemento el decodificador de la RAM.

```
A23,22,21,20   19,18,17,16   15,14,13,12   11,10,9,8   7,6,5,4   3,2,1

 0  0  0  0    0  0  1  0     0  0  0  0     0  0  0  0   0  0  0  0
 0  0  0  0    0  0  1  0     0  1  1  1     1  1  1  1   1  1  1  1
```

Las lineas de A15 a A15 son conectan a compuertas NOR (algunas ya usadas en la decodificación de la ROM), si todas las entradas a las compuertas son cero, entonces la compuerta NAND irá a cero y si ademas *Addres strob es cero, se activara la memoria a través de las compuertas OR siguientes.

La línea de direcciones A14, se conecta al decodificador 74139 que selecciona la memoria que se debe activar.

El circuito de DTACK es el mismo utilizado para la ROM y se activa por medio de la pata 11 de la compuerta IC14/3.

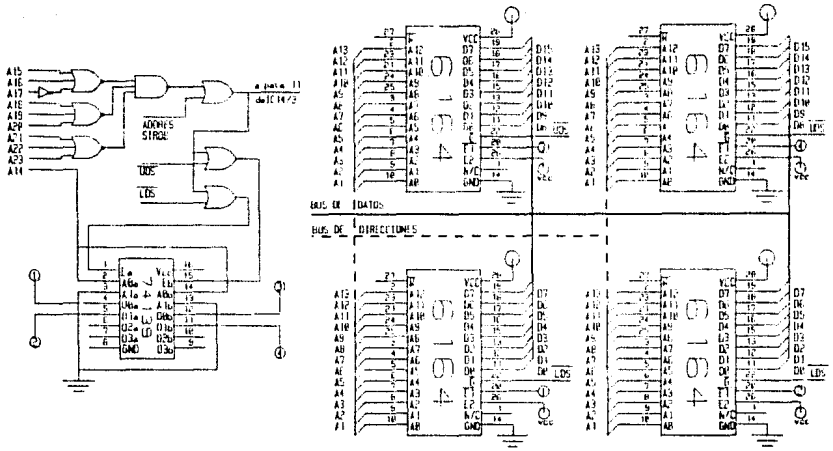


Fig. 3.14

CAPITULO IV

El hardware del secuenciador fue diseñado pensando en un secuenciador con capacidades como edición de notas, grabación de diferentes pistas, edición de sistemas exclusivos de generadores remotos etc. El programa que se documenta aquí, ejecuta solo funciones muy básicas, aunque es posible realizar todas las tareas mencionadas con anterioridad, siempre y cuando se utilice el software apropiado.

El programa explicado a continuación, ejecutará básicamente seis funciones:

1.- Grabación de señales MIDI
2.- Reproducción de señales MIDI
3.- Grabación del contenido del secuenciador en un grabador de datos MIDI (MDR; MIDI DATA RECORDER)
4.- Lectura de datos de un grabador de datos MIDI (MDR)
5.- Modificar la velocidad del reloj MIDI
6.- Transmitir un mensaje de apagar todas las notas

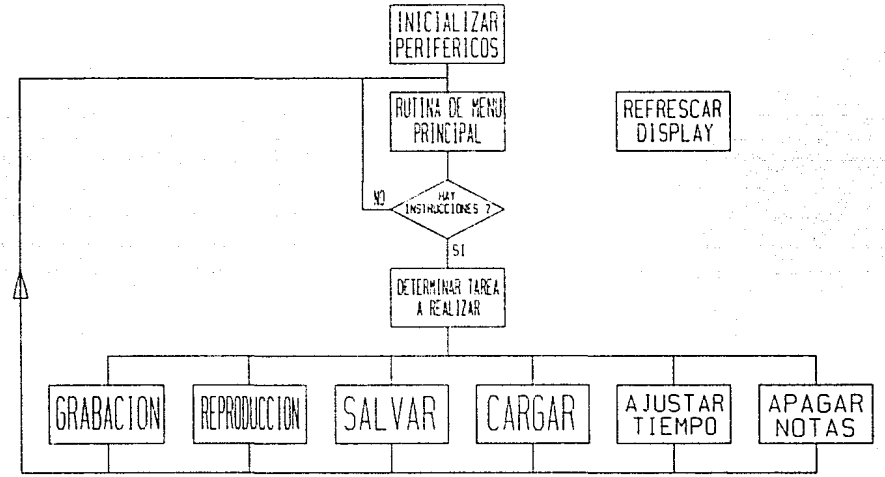


FIG 4.1

En la figura 4.1 se muestra un diagrama de bloques que ilustra el flujo del programa. Al iniciar su funcionamiento, el microprocesador debe inicializar todos los periféricos, para que respondan de una forma adecuada a las necesidades del sistema así como inicializar las variables que deban tener algún valor inicial. Una vez puesto en marcha el sistema, el secuenciador espera a que se le ordene realizar alguna tarea, esto se hace oprimiendo alguna de las teclas de funciones del teclado, por lo que el secuenciador debe determinar si

la tecla oprimida es valida y determinar de que tarea se trata para realizarla. Una vez terminada la tarea asignada, regresa a esperar a que se le asigne otra tarea. Mientras todo esto es realizado, se debe refrescar el display para que el usuario este en todo momento al tanto de lo que sucede.

A continuación de describirán cada una de las secciones del programa que acaba de mencionarse.

Es posible que al leer alguna de las rutinas no se comprenda la totalidad de las instrucciones que la forman, pero si se ha leído todo el programa será mucho más sencillo entender su funcionamiento.

INICIALIZACION

```
1)   Viewdata       equ     $102001
2)   Viewport       equ     $102005
3)   Viewport1      equ     $102005
4)   Viewport2      equ     $102007
5)   Viewport3      equ     $102009
6)   Viewport4      equ     $10200a
7)   Acialstatus    equ     $102041
8)   Acialcontrol   equ     $102041
9)   Acialtransmit  equ     $102043
10)  Acialreciv     equ     $102043
11)  Acia2status    equ     $102081
12)  Acia2control   equ     $102081
13)  Acia2transmit  equ     $102083
14)  Acia2reciv     equ     $102083
15)  timerCR1       equ     $1020c1
16)  timerCR2       equ     $1020c3
17)  timerCR3       equ     $1020c1
18)  timerMSB1      equ     $1020c5
19)  timerLSB1      equ     $1020c7
20)  timerMSB2      equ     $1020c9
21)  timerLSB2      equ     $1020cb
22)  timerMSB3      equ     $1020cd
23)  timerLSB3      equ     $1020cf
24)  timerSR        equ     $1020c3
25)  comacia1       equ     %00010101
26)  comacia2       equ     %00010101
27)       move.b  #comacia1,acialcontrol
28)       move.b  #comacia2,acia2control
29)       move.b  #1,timerCR2
30)       move.b  #1,timerCR1
31)       move.b  #0,timerCR2
32)       move.b  #%11000011,timerCR3
33)       move.b  #%11000011,timerCR2
34)       move.b  #%00000010,timerCR1
35)       move.b  #$02,timerMSB2
36)       move.b  #$00,timerLSB2
37)       move.b  #$04,timerMSB3
38)       move.b  #$ff,timerLSB3
39)       move.l  #display,100          ;Autovector1
40)       move.l  #display,104          ;Autovector2
41)       move.l  #display,108          ;Autovector3
42)       move.l  #display,112          ;Autovector4
```

```
43)       move.l  #display,116          ;Autovector5
44)       move.l  #timerinter,120       ;Autovector6
45)       move.l  #display,124          ;Autovector7
46)       move.l  #$ffe,a7
47)       move    #$200f,sr
48)       move.l  #MM,screen
```

Las instrucciones de las lineas 1 a 26, son directivas para el ensamblador, y asignan un cierto valor a cada etiqueta; el significado de cada etiqueta es el siguiente:

Viewdata	Esta dirección debe contener el codigo del caracter que se desplegará en el display
Viewport	Esta dirección es el inicio de los latches que seleccionan el display a encender
Viewport1	Primeros ocho caracteres de la primera linea del display
Viewport2	Segundos ocho caracteres de la primera linea del display
Viewport3	Primeros ocho caracteres de la segunda linea del display
Viewport4	Segundos ocho caracteres de la segunda linea del display
Acialstatus	Byte de status del 6850-1
Acialcontrol	Byte de control del 6850-1
Acialtransmit	Registro de transmision del 6850-1
Acialreciv	Registro de recepción del 6850-1
Acia2status	Byte de estatus del 6850-2
Acia2control	Byte de control del 6850-2
Acia2transmit	Registro de transmision del 6850-2
Acia2reciv	Registro de recepción del 6850-2
timerCR1	Registro de control del timer 1 (en el 6840)
timerCR2	Registro de control del timer 2 (en el 6840)
timerCR3	Registro de control del timer 3 (en el 6840)
timerMSB1	Byte más significativo del timer 1
timerLSB1	Byte menos significativo del timer 1
timerMSB2	Byte más significativo del timer 2
timerLSB2	Byte menos significativo del timer 2
timerMSB3	Byte más significativo del timer 3
timerLSB3	Byte menos significativo del timer 3
timerSR	Registro de status del timer
comacia1	Comando para el 6850-1
comacia2	Comando para el 6850-2

Las lineas 27 y 28 programan a los USART 6850-1 y 6850-2.

Las lineas 29, 30 y 31 resetean al timer 6840 y lo dejan en un estado adecuado para recibir los bytes de control.

Las lineas 32, 33 y 34 programan a cada uno de los contadores.

Las lineas 34 a 38 cargan a los contadores con un valor inicial e inician la cuenta de estos. El timer número dos funcionará como reloj de refresco del display, los displays de encenderán (25000/Número en timer 2) veces por segundo, en este caso (25000 Decimal)/($200 hexadecimal)= 48.8 veces por segundo. El timer 3 funcionará como reloj MIDI.

Las lineas 39 a 45 guardan en memoria los valores de los vectores de excepción que se accesarán cuando algún dispositivo provoque una interrupción con autovector.

Las línea 46 pone el supervisor stack pointer apuntando a la direccion $FFE, la línea 47 activa las interrupciones y por último la línea 48 pone el puntero de pantalla apuntando al mensaje MM (los mensajes se pueden ver en la sección de mensajes mas adelante).

MENU PRINCIPAL

```
1)   start:      move.l  #MM,screen
2)               bsr     GET2
3)               cmp.b   #192,d5          ;F1
4)               beq     RECORD
5)               cmp.b   #193,d5          ;F2
6)               beq     PLAY
7)               cmp.b   #194,d5          ;F3
8)               beq     WMDR
9)               cmp.b   #195,d5          ;F4
10)              beq     RMDR
11)              cmp.b   #196,d5          ;F5
12)              beq     TIME
13)              cmp.b   #213,d5          ;F6
14)              beq     PANIC
15)              bra     Start
```

En esta sección del programa, se lee el teclado y se determina si la tecla oprimida corresponde a una tarea del secuenciador, si es asi, se procede de acuerdo con la tarea, y en caso contrario, se continúa esperando una tecla valida.

La línea uno visualiza el mensaje MM en pantalla, la línea 2 obtiene un caracter del teclado (si no se ha oprimido alguna tecla, se espera a que esto ocurra) y guarda el código ASCII en D5.

La línea 3 compara el valor en D5 con 192 (192 es el valor de F1). La línea 4 determina si la tecla oprimida fue F1 si es así, el microprocesador salta a la dirección etiquetada como RECORD, donde se lleva acabo la grabación de datos MIDI.

La línea 5 compara el valor en D5 con 193 (193 es el valor de F2). La línea 6 determina si la tecla oprimida fue F2 si es así, el microprocesador salta a la dirección etiquetada como PLAY, donde se lleva acabo la reproducción de datos MIDI.

La línea 7 compara el valor en D5 con 194 (194 es el valor de F3). La línea 8 determina si la tecla oprimida fue F3 si es así, el microprocesador salta a la dirección etiquetada como WMDR, donde se lleva acabo la escritura a un MDR (MIDI DATA RECORDER).

La línea 9 compara el valor en D5 con 195 (195 es el valor de F4). La línea 10 determina si la tecla oprimida fue F4 si es así, el microprocesador salta a la dirección etiquetada como RMDR, donde se lleva acabo la lectura de datos de un MDR (MIDI DATA RECORDER).

La línea 11 compara el valor en D5 con 196 (196 es el valor de F5). La línea 12 determina si la tecla oprimida fue F5 si es así, el microprocesador salta a la dirección etiquetada como TIME, donde se lleva acabo el ajuste de tiempo del secuenciador.

La línea 13 compara el valor en D5 con 213 (213 es el valor de

F6). La linea 14 determina si la tecla oprimida fue F6 si es asi, el microprocesador salta a la dirección etiquetada como PANIC, donde se lleva acabo la transmisión del mensaje apagar todas las notas (ALL NOTES OFF). Esta función es conocida comercialmente como boton de panico.

GRABACION

```
1)   RECORD:      move.l  #$20000,a6
2)                move.w  #$0000,cuenta
3)                move.l  #M1,screen
4)   perate:      BSR     get2
5)                cmp.b   #240,d5
6)                beq     start
7)                cmp.b   #$49,d5
8)                bne     perate
9)                move.l  #M11,screen
10)               move.w  #$0,cuental
11)               move.w  #$0,cuarter
12)               move.b  #1,acticlick
13)  jk:          bsr     get1
14)               move.w  cuenta,d6
15)               ror.w   #8,d6
16)               move.b  d6,(a6)+
17)               rol.w   #8,d6
18)               move.b  d6,(a6)+
19)               move.w  #$0000,cuenta
20)               move.b  d5,(a6)+
21)               bsr     get1
22)               move.b  d5,(a6)+
23)               bsr     get1
24)               move.b  d5,(a6)+
25)  W1:          move.b  Acia2status,d5
26)               and.b   #$1,d5
27)               beq     jk
28)               move.b  Acia2reciv,d5
29)               cmp.b   #240,d5
30)               move.b  #$0,acticlick
31)               move.l   a6,FINGRAB
32)               bsr     alloff
33)               bra     start
```

A continuación se describe el algoritmo que se utilizó para implementar la rutina de grabación:

1.- Iniciar la grabación.
2.- Contar los relojes MIDI que tarda en llegar un mensaje MIDI.
3.- Guardar en memoria el número de relojes MIDI contados y el mensaje MIDI (bytes de estatus y de datos) en localidades contiguas.
4.- Si no existe orden de terminar, regresar al paso 2.
5.- Fin.

El algoritmo anterior tiene varias ventajas, la primera es que deja una estructura de datos que facilita mucho la reproducción de la secuencia grabada. Y la segunda es que es bastante sencilla de implementar. Existen otros algoritmos que se encuentran enfocados a el ahorro de memoria, ya que con el algoritmo empleado, se utilizan 5 bytes para grabar un evento MIDI y existen diferentes maneras de comprimir la información contenida en estos cinco bytes. Los cinco bytes mencionados son los siguientes: 2 bytes de cuenta de tiempos MIDI entre eventos, un byte de estatus y dos bytes de datos. Esto significa que para grabar una nota (un mensaje de inicio de nota y un mensaje de fin de nota), se requerirán de 10 bytes, por lo que el secuenciador será capaz de grabar 3276 notas ya que cuenta con 32 Kb de memoria.

A continuación se describe línea por línea la rutina de grabación de datos MIDI.

La línea 1 guarda el inicio de la memoria de datos en el registro a6.

La línea 2 borra la cuenta de relojes MIDI, la cuenta de relojes MIDI es realizada automáticamente por la rutina de interrupción del timer 6840. Cada vez que ocurre una interrupción debido al reloj MIDI, se incrementa la localidad etiquetada como cuenta.

La línea 3 visualiza el mensaje M1 en el display

La línea 4 salta a la subrutina GET2 la cual espera la llegada de un caracter del teclado y guarda el valor ASCII en el registro d5.

Las líneas 5 y 6 se encargan de regresar al menú principal en caso de que se oprima la tecla ESC.

Las líneas 7 y 8 se encargan de determinar cuando se debe iniciar la grabación, esto es hecho al oprimir la tecla "I" (Inicio).

La línea 9 se encarga de visualizar el mensaje M11 en el display.

La línea 10 pone el metronomo en el inicio de un tiempo.

La línea 11 pone el metronomo en el tiempo 1.

La línea 12 activa el metronomo.

La línea 13 obtiene un dato MIDI y la guarda en el registro d5.

Las lineas 14,15,16,17 y 18 se encargan de guardar en a6 y a6+1 el número de relojes MIDI desde que llego el último mensaje.

La línea 19 borra la cuenta entre eventos MIDI.

La línea 20 guarda el byte de estatus del Mensaje MIDI recibido.

La línea 21 obtiene el primer byte de datos y la línea 22 lo guarda en memoria.

La línea 23 obtiene el segundo byte de datos y la línea 24 se encarga de guardarlo en memoria.

Las líneas 25, 26 y 27 checan si se ha oprimido alguna tecla en el teclado del secuenciador, si no es así, se regresa a la línea 13 para seguir grabando.

Las líneas 29 y 30 checan si la tecla oprimida fue ESC.

La línea 30 desactiva el metronomo.

la línea 31 guarda en FINGRAB la posición del último byte grabado, para que la rutina de ejecución no intente transmitir datos después de esta posición.

La línea 32 envia un mensaje de apagar todas las notas a los sintetizadores.

Por ultimo la línea 33 retorna el control a la rutina de menus.

Aparentemente, esta rutina funcionará únicamente si se envian mensajes MIDI de tres bytes de longitud, pero en realidad funciona para mensajes de cualquier longitud, incluso mensajes de sistema

exclusivo, aunque el aprovechamiento de memoria no es optimo para mensajes de longitud diferente de tres bytes.

REPRODUCCION

```
1)   PLAY:          move.l  #M2,screen
2)                  move.l  #$20000,a6
3)   perate1:       BSR     get2
4)                  cmp.b   #240,d5
5)                  BEQ     Start
6)   cc1:           cmp.b   #$49,d5
7)                  bne     perate1
8)                  move.l  #M21,screen
9)                  move.w  #$0,cuenta1
10)                 move.w  #$0,cuarter
11)                 move.b  #1,acticlick
12)  st1:           cmp.l   FINGRAB,a6
13)                 bhi     regresa
14)                 clr.l   d6
15)                 move.b  (a6)+,d6
16)                 asl.l   #8,d6
17)                 move.b  (a6)+,d6
18)                 move.w  #$000,cuenta
19)  lp1:           move.w  cuenta,d5
20)                 cmp.w   d5,d6
21)                 bne     lp1
22)                 move.b  (a6)+,d6
23)                 bsr     out2
24)                 move.b  (a6)+,d6
25)                 bsr     out2
26)                 move.b  (a6)+,d6
27)                 bsr     out2
28)                 move.b  Acia2status,d5
29)                 and.b   #$1,d5
30)                 beq     st1
31)                 move.b  Acia2reciv,d5
32)                 cmp.b   #240,d5
33)  regresa:       move.b  #$0,acticlick
34)                 bsr     alloff
35)                 bra     start
```

Esta rutina es bastante sencilla ya que unicamente hay que leer los datos dejados por la rutina de grabación. los pasos a seguir son los siguientes:

1.- Borrar el contador de relojes MIDI
2.- Esperar a que el contador de relojes MIDI contenga el mismo valor que los primeros dos bytes (en la repetición de esta linea seran los dos siguientes bytes de memoria de datos) de memoria de datos.
3.- Transmitir los siguientes tres bytes de memoria.
4.- Si aun no se llega al final de los datos, regresar al paso 1.
5.- FIN

A continuación se describe la función de cada una de las líneas de esta rutina.

La línea 1 visualiza el mensaje M2 en la pantalla.
La línea 2 inicializa el puntero a6 con el valor de la primera dirección de la memoria de datos.
Las líneas 3, 4, 5, 6 y 7 se encargan de averiguar que tecla se oprimió en el teclado, si se trata de ESC se regresa al menú principal, si se trata de "I" (I-nicio), se inicia la reproducción, y si se trata de alguna otra tecla, se espera a que se oprima alguna de las teclas validas para esta rutina.
La línea 8 visualiza el mensaje M21 en el display.
La línea 9 pone al metronomo al inicio de un tiempo.
La línea 10 pone al metronomo en el tiempo uno.
La línea 11 activa el metronomo.
Las líneas 12 y 13 checan si ya se llego al fin del archivo, si es así, se salta al termino de esta rutina.
Las líneas 14, 15, 16 y 17 guardan en d6 el contenido de los dos siguientes bytes de memoria de datos.
La línea 18 borra la cuenta de relojes MIDI.
Las líneas 19, 20 y 21 esperan a que el número de relojes MIDI sea igual al contenido en d6.
La línea 22 lee el siguiente byte y la línea 23 lo transmite por el puerto MIDI.
La línea 24 lee el siguiente byte de datos y la línea 25 lo transmite por el puerto MIDI.
La línea 26 lee el siguiente byte de datos y la línea 27 lo transmite por el puerto MIDI.
La líneas 28, 29 y 30 checan si se ha oprimido alguna tecla en el teclado del secuenciador, si no es así, se regresa a la línea 19.
Las líneas 31 y 32 se encargan de determinar la tecla oprimida.
La línea 33 desactiva el metronomo.
La línea 34 transmite un mensaje de apagar todas las notas.
Por último la línea 35 regresa el mando a la rutina de menus.

ESCRITURA A MDR

```
1)   WMDR:        move.l  #M5,screen
2)                move.l  #$20000,a6
3)   peperal:     BSR     get2
4)                cmp,b   #240,d5
5)                beq     start
6)                cmp.b   #$49,d5
7)                bne     peperal
8)                move.l  #M51,screen
9)                move.l  #$f0,d6
10)               bsr     out2
11)               move.l  #65,d6
12)               bsr     out2
13)               move.l  #4680,d2
14)  loop0:       move.l  #$6,d1
15)               clr.l   d3
16)               clr.l   d7
17)  loop1:       move.b  (a6)+,d6
18)               lsr.b   #$1,d6
19)               bcc     next1
```

```
20)                 bset    d3,d7
21)  next1:         bsr     out2
22)                 add.l   #$1,d3
23)                 dbra    d1,loop1
24)                 move.b  d7,d6
25)                 bsr     out2
26)                 dbra    d2,loop0
27)                 move.l  #$f7,d6
28)                 bsr     out2
29)                 bra     start
```

Esta subrutina graba el contenido de la memoria del secuenciador en un dispositivo MIDI con unidad de discos, tal como un MDR (MIDI DATA RECORDER) o cualquier otro dispositivo que sea capaz de recibir sistemas exclusivos.

Para enviar un mensaje de sistema exclusivo, es necesario enviar primero un byte de estatus (F0) que indique que se transmitirá un sistema exclusivo, un byte que indica la marca del dispositivo que transmite, todos los bytes de datos deseados y finalmente un mensaje que indique el final del sistema exclusivo (EOX). Aunque el formato del sistema exclusivo es muy sencillo, tiene una complicación, los bytes de datos no pueden ser mayores a 127 (deben tener el bit más significativo en cero).

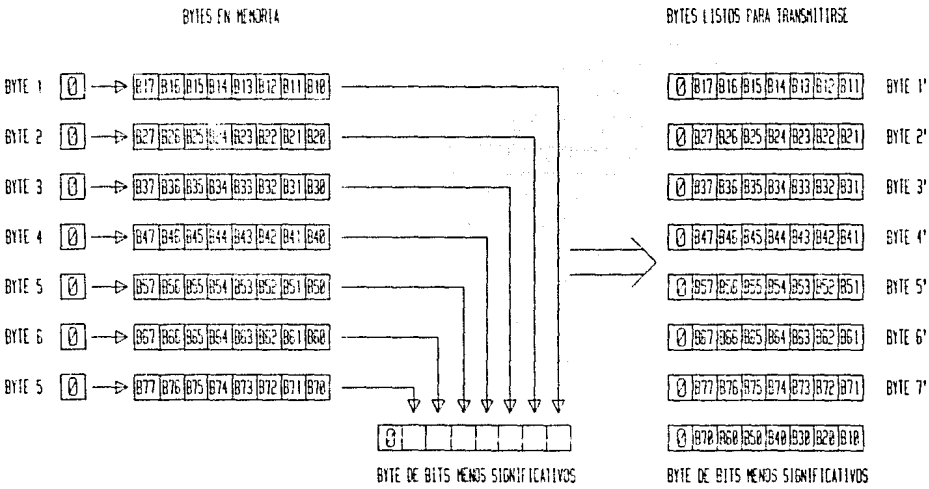


Fig. 4.2

Para poder transmitir los bytes con el bit más significativo en cero sin perder información se harán grupos de 7 bytes, se hará un corrimiento a la derecha en cada byte y el bit menos significativo de

cada byte formara parte de un nuevo bit, la figura 4.2 ilustra esto con detalle.

Ahora veamos la función de las instrucciones del programa.

La línea 1 visualiza el mensaje M5 en pantalla.

La línea 2 guarda el inicio de la memoria de datos en a6.

La línea 3 obtiene un caracter del teclado.

Las líneas 4, 5, 6 y 7 se encargan de revisar la tecla oprimida, si se trata de "I" (Inicio), se continúa con el programa, si se trata de ESC se regresa al menú principal y si se trata de alguna otro caracter, se espera otra tecla.

La línea 8 visualiza el mensaje M51 en pantalla.

Las líneas 9 y 10 transmiten el byte de estatus del sistema exclusivo.

Las líneas 11 y 12 transmiten el byte de marca de dispositivo transmisor (en este caso 65 es el número de identificación de ROLAND).

La línea 13 indica el número de veces que se repetirá el loop0-2 (4680 * 7bytes = 32Kilo Bytes).

La línea 14 indica el número de veces que se repetira el loop1-2.

La línea 17 carga el primer byte de datos que se transmitirá en d6 y la línea 18 se encarga de hacer el corrimiento a la derecha.

Las líneas 19 y 20 se encargan de guardar el bit menos significativo en el byte de bits menos significativos.

La línea 21 se encarga de transmitir el byte por el puerto MIDI.

La línea 22 suma 1 al registro d3 que sirve como puntero de bit del byte de bits menos significativos.

La línea 23 es el fin del loop1.

Las líneas 24 y 25 transmiten el byte de bits menos significativo por el puerto MIDI.

La línea 26 es el fin del loop0.

Las líneas 27 y 28 transmiten el byte de EXO (fin del sistema exclusivo), y finalmente la línea 29 regresa el mando al menú principal.

LECTURA DE MDR

```
1)   RMDR:          move.l  #bufini,a1
2)                   move.l  #bufini,a0
3)                   move.l  #$20000,a6
4)                   move.l  a6,a5
5)                   move.b  #%10010101,acialcontrol
6)                   move.l  #M6,screen
7)   pepera2:        BSR     get2
8)                   cmp.b   #240,d5
9)                   beq     start
10)                  cmp.b   #$49,d5
11)                  bne     pepera2
12)                  move.l  #M62,screen
13)  ww1:            bsr     bget1
14)                  cmp.b   #$f0,d5
15)                  bne     ww1
16)  ww2:            bsr     bget1
17)                  cmp.b   #65,d5
18)                  bne     ww2
19)                  move.l  #4680,d2
20)                  move.l  #M61,screen
```

```
21)  Rloop0:        move.l  #$6,d6
22)  Rloop1:        bsr     bget1
23)                 lsl.b   #1,d5
24)                 move.b  d5,(a6)+
25)                 dbra    d6,Rloop1
26)                 bsr     bget1
27)                 move.l  #$6,d6
28)                 clr.l   d3
29)  Rloop2:        btst    d3,d5
30)                 beq     sig1
31)                 bset    #0,(a5)+
32)                 bra     Rconti
33)  sig1:          bclr    #0,(a5)+
34)  Rconti:        add.l   #$1,d3
35)                 dbra    d6,Rloop2
36)                 dbra    d2,Rloop0
37)                 move.b  #%00010101,acialcontrol
38)                 move.l  a6,FINGRAB
39)                 bra     start
```

En la subrutina de escritura a MDR se vio como se debe transmitir un mensaje de sistema exclusivo y el contenido del secuenciador se recibira en el mismo formato que fue transmitido. Se deben recibir 7 bytes de datos y un byte de bits menos significativos, el cual debe ser integrado a los bytes de datos. Para evitar la perdida de información, está subrutina activa las interrupciones del puerto de entrada MIDI y se agrega un buffer para evitar que mientras se incorpora el byte de bits menos significativos a los bytes de datos se pierda algún byte del sistema exclusivo.

El buffer agregado tiene una longitud de 256 bytes a manera de una cola circular y está implementado por las subrutinas ACIA1TIMER que agrega datos al buffer y BGET1 que retira bytes del buffer.

Veamos ahora la rutina que realiza todo esto.

La líneas 1 y 2 inicializan los punteros del buffer.

La líneas 3 y 4 inicializan los punteros de memoria de datos a5 y a6.

La línea 5 activa las interrupciones del ACIA 1

La línea 6 visualiza el mensaje M6 en pantalla

La línea 7 lee un caracter del teclado y lo deja en d5

Las líneas 8, 9, 10 y 11 revisan la tecla leida, si se trata de I "Inicio", se continua con el programa, si se trata de ESC se regresa al menú principal, y si se trata de alguna otra tecla, se espera otra.

La línea 12 visualiza el mensaje M62 en pantalla.

Las líneas 13, 14 y 15 esperan a que el mensaje de inicio de sistema exclusivo sea recibido.

Las líneas 16, 17 y 18 checan que el mensaje de sistema exclusivo sea del secuenciador.

La línea 19 indica el número de veces que se repetira el loop Rloop0 (7 * 4680 = 32kb).

La línea 20 visualiza el mensaje M61

La línea 21 indica el número de veces que se repetirá el loop Rloop1.

Las líneas 22, 23, 24 y 25 reciben 7 bytes, hacen un corrimiento de un bit a la izquierda de cada byte y los almacenan en localidades de memoria contiguas (El porque de esto se puede entender fácilmente observando la figura 4.2)

La linea 26 recibe el byte de bits menos significativos y lo almacena en d5.

La linea 27 indica el número de veces que se repetirá el loop Rloop2 y la linea 28 borra el registro d3 que será utilizado como puntero al probar el estado de los bits del byte de bits menos significativos.

Las lineas 29, 30, 31, 32, 33, 34 y 35 incorporan el byte de bits menos significativos a los 7 bytes de datos antes leidos.

La linea 36 es el fin del loop Rloop0

La linea 37 desactiva las interrupciones del ACIA 1

La linea 38 indica a la rutina de ejecución que existen datos listos para ejecutarse y marca el fin de los datos.

La linea 39 manda el control del programa al menú principal.

AJUSTE DE TIEMPO

```
1)  TIME:       move.w  #$0,cuental
2)              move.w  #$0,cuarter
3)              move.b  #1,acticlick
4)              move.l  #m3,screen
5)  otro:       move.l  ftime,d1
6)              cmp.l   #$186a,d1
7)              bhi     sethi
8)              cmp.l   #$300,d1
9)              blo     setlow
10)             bsr     get2
11)             cmp.b   #240,d5
12)             beq     termina
13)             cmp.b   #139,d5
14)             beq     mas
15)             cmp.b   #138,d5
16)             beq     menos
17)             bra     otro
18) mas:        move.l  ftime,d1
19)             sub.l   #$30,d1
20)             move.l  d1,ftime
21)             ror.l   #8,d1
22)             move.b  d1,timerMSB3
23)             rol.l   #8,d1
24)             move.b  d1,timerLSB3
25)             bra     otro
26) menos:      move.l  ftime,d1
27)             add.l   #$30,d1
28)             move.l  d1,ftime
29)             ror.l   #8,d1
30)             move.b  d1,timerMSB3
31)             rol.l   #8,d1
32)             move.b  d1,timerLSB3
33)             bra     otro
34) termina:    move.b  #0,acticlick
35)             bra     start
36)  sethi:     move.l  #$186a,ftime
37)             bra     otro
38) setlow:     move.l  #$300,ftime
39)             bra     otro
```

Esta rutina permite ajustar el tiempo del reloj MIDI. Su funcionamiento es muy sencillo, si se presiona en el teclado del secuenciador la flecha que apunta hacia arriba, la velocidad del reloj se incrementa, si se presiona la flecha que apunta hacia abajo, la velocidad del reloj disminuye.

EL metronomo es visible en el display y lo maneja la rutina de interrupción del TIMER. La velocidad de este puede ser variada cuando se desee, por ejemplo se puede grabar una pieza mas lento de lo normal y después aumentar la velocidad del reloj MIDI para que la pieza sea reproducida a la velocidad adecuada.

Veamos ahora las instrucciones del programa.

Las líneas 1, 2 y 3 activan el metronomo y lo ponen al inicio del primer tiempo.

La línea 4 visualiza el mensaje M3

Las líneas 5, 6, 7, 8 y 9 se encargan de vigilar que el tiempo del reloj no sea menor de 40 BPM o mayor que 220 BPM (Beates Por Minuto). Si el tiempo se sale de limites, las rutinas setlow y sethi se encargan de ponerlo dentro del rango permitido.

La línea 10 se encarga de obtener un caracter del teclado.

Las líneas 11 y 12 se encargan de checar si la tecla presionada fue ESC, si es así se salta a termina para apagar el metronomo y volver al menú principal.

Las líneas 13 y 14 se encargan se revisar si la tecla oprimida fue la flecha hacia arriba, si es así se incrementa el tiempo del reloj.

Las líneas 15 y 16 se encargan se revisar si la tecla oprimida fue la flecha hacia abajo, si es así se incrementa el tiempo del reloj.

La línea 17 regresa el mando a "otro" para esperar otra a que se oprima otra tecla.

Las líneas 18 a 25 se encargan de incrementar el tiempo del reloj MIDI.

Las líneas 26 a 33 se encargan de disminuir el tiempo del reloj MIDI.

Las líneas 34 y 35 apagan el metronomo y regresan al menú principal.

Las líneas 36 y 37 ponen el tiempo del reloj MIDI a su máxima velocidad.

Las líneas 38 y 39 ponen el tiempo del reloj MIDI a su mínima velocidad.

APAGAR TODAS LAS NOTAS

```
1)  Panic:      move.l  #M8,screen
2)              bsr     Alloff
3)              move.l  #$ffffffff,d2
4)  retardo:    move.l  #$5,d1
5)  retardo1:   nop
6)              dbra    d1,retardo1
7)              dbra    d2,retardo
8)              bra     start
```

Esta rutina ordena a todos los dispositivos receptores, que cesen de tocar todas las notas que se encuentren sonando al momento de recibir el mensaje. Esto es muy útil, cuando por alguna razón los

dispositivos receptores no reciben los mensajes de nota off. Cuando esto sucede no es necesario apagar los dispositivos receptores para que dejen de tocar las notas que no recibieron el mensaje de apagado.

La linea 1 visualiza el mensaje M8 en pantalla.

La linea 2 salta a la rutina que transmite el mensaje y que es usada también por la rutina de grabación y reproducción.

Las líneas restantes se encargan de retardar el mando al menú principal, con objeto de que el mensaje M8 pueda ser Visto por el usuario.

SUBRUTINAS

La subrutina GET1 Obtiene un dato de un dispositivo MIDI a traves del ACIA 1 y lo almacena en el registro D5

```
1)  GET1:       move.b  Acialstatus,d5
2)              and.b   #1,d5
3)              beq     GET1
4)              move.b  Acialreciv,d5
5)              RTS
```

La subrutina GET2 obtiene un caracter del teclado y lo almacena en d5.

```
GET2:           move.b  Acia2status,d5
                and.b   #1,d5
                beq     GET2
                move.b  Acia2reciv,d5
                RTS
```

La subrutina OUT2 transmite el byte almacenado en d5 por el puerto MIDI OUT usando el ACIA 2.

```
OUT2:           move.b  Acia2status,d5
                and.b   #$2,d5
                beq     out2
                move.b  d6,acia2transmit
                rts
```

La subrutina BGET1 se encarga de sacar un caracter de un buffer circular que se utiliza para leer datos de un MDR (Midi Data Recorder). La variable "count" lleva la cuenta de cuantos bytes hay dentro del buffer.

```
Bget1:          cmp.b   #0,count
                beq     Bget1
                sub.b   #$1,count
                move.b  (a1)+,d5
                cmp.l    #bufINI+20,a1
                bhi      ccon1
                rts
ccon1:          move.l   #bufINI,a1
                rts
```

La subrutina Acialinter, es accesada por una interrupción del ACIA 1, solo cuando se esta ejecutando la rutina de lectura a MDR y se encarga de recibir datos del dispositivo de almacenamiento de datos MIDI y agregarlos a un buffer circular.

```
Acialinter:   move.b   Acialreciv,d7
              cmp.b    #10,count
              blo      con
              rte
con:          add.b    #1,count
              move.b    d7,(a0)+
              cmp.l    #bufINI+20,a0
              bhi      con1
              rte
con1:         move.l   #bufINI,a0
              rte
```

EXCEPCION DEL 6840 (TIMER)

Cuando el 6840 genera una interrupción, se ejecuta la rutina TIMERINTER que tiene básicamente 2 funciones: la primera es refrescar el display y es ejecutada cuando la interrupción es producida por el timer número 2. La segunda función es ejecutada cuando la interrupción es producida por el timer tres y consiste en llevar la cuenta de relojes MIDI y hacer visible el metronomo en caso de que se encuentre activado.

```
1)   timerinter:     movem.l   d0-d3/a0-a3,-(sp)
2)                   move.b  timerSR,d0
3)                   and.b   #$2,d0
4)                   beq     gg
5)                   move.b  timerMSB2,d0
6)                   move.b  timerLSB2,d0
7)                   bsr     Display
8)                   bra     ya
9)   gg:             move.b  timerMSB3,d0
10)                  move.b  timerLSB3,d0
11)                  bsr     MIDIclock
12)                  bsr     Click
13)  ya:             movem.l  (SP)+,d0-d3/a0-a3
14)                  RTE
15)  MIDIclock:      add.w   #1,cuenta
16)                  rts
17)  Click:          cmp.b     #1,acticlick
18)                  beq     sigue
19)                  rts
20)  sigue           add.w   #1,cuental
21)                  cmp.w   #24,cuental
22)                  beq     Cambio
23)                  rts
24)  Cambio:         move.l  screen,a0
25)                  add.l   #28,a0
26)                  move.b  #$20,(a0)+
27)                  move.b  #$20,(a0)+
28)                  move.b  #$20,(a0)+
```

```
29)                 move.b  #$20,(a0)+
30)                 move.w  #$0,cuental
31)                 add.b   #1,cuarter
32)                 cmp.b   #1,cuarter
33)                 beq     Cuarter1
34)                 cmp.b   #2,cuarter
35)                 beq     Cuarter2
36)                 cmp.b   #3,cuarter
37)                 beq     Cuarter3
38)                 cmp.b   #4,cuarter
39)                 beq     Cuarter4
40)                 rts
41)  cuarter1:      move.l  screen,a0
42)                 add.l   #28,a0
43)                 move.b  #$31,(a0)
44)                 rts
45)  cuarter2:      move.l  screen,a0
46)                 add.l   #29,a0
47)                 move.b  #$32,(a0)
48)                 rts
49)  cuarter3:      move.l  screen,a0
50)                 add.l   #30,a0
51)                 move.b  #$33,(a0)
52)                 rts
53)  cuarter4:      move.b  #$0,cuarter
54)                 move.l  screen,a0
55)                 add.l   #31,a0
56)                 move.b  #$34,(a0)
57)                 rts
```

La linea 1 salva todos los registros usados en esta rutina en el stack.

Las lineas 2 y 3 determinan la fuente de interrupción (timer 2 o 3).

La linea 4 salta a "gg" en caso de que la interrupción haya sido provocada por el timer 3.

Las lineas 5, 6, 7 y 8 se encargan de refrescar el display en caso de que la fuente de interrupción haya sido el timer 2.

Las lineas 9 a 12 se encargan de llevar la cuenta de relojes MIDI y hacer aparecer el metronomo en la pantalla en caso de que este se encuentre activo.

La linea 13 se encarga de volver todos los registros al estado en que se encontraban antes de que la linea 14 regrese el mando al programa principal.

La subrutina MIDIclock se encarga de incrementar la cuenta de los relojes MIDI (lineas 15 y 16).

Las lineas restantes forman parte de la rutina Click que maneja el metronomo.

SUBRUTINA DE REFRESCO DEL DISPLAY

Esta es la rutina que se encarga de manejar el display. Aqui se lleva acabo la conversión de datos de ASCII a los códigos equivalentes en los displays de 15 segmentos, se accesan los latches de la pantalla y se lleva acabo el multiplexado de los displays.

```
1)   Display:      move.l  data0,d0
2)                 move.l  data1,d1
3)                 move.l  data3,d3
4)                 move.l  data4,d4
5)                 move.l  raster,a0
6)                 lea     viewport1,a1
7)                 lea     viewdata,a2
8)                 clr.l   d2
9)                 move.b  (a0)+,d2
10)                sub.b   #$20,d2
11)                rol.w   #1,d2
12)                move.l  #tabla,a3
13)                move.w  0(a3,d2),d2
14)                movep.l d3,0(a1)
15)                movep.w d2,0(a2)
16)                move.b  d0,Viewport1
17)                ror.l   #8,d0
18)                move.b  d0,Viewport2
19)                ror.l   #8,d0
20)                move.b  d0,Viewport3
21)                ror.l   #8,d0
22)                move.b  d0,Viewport4
23)                ror.l   #7,d0
24)                dbra    d1,cont
25)                move.l  screen,a0
26)                move.l  #31,d1
27)  cont:         move.l  d0,data0
28)                move.l  d1,data1
29)                move.l  d3,data3
30)                move.l  a0,raster
31)                rts
```

MENSAJES

```
MM:          dc.b   'MENU PRINCIPAL  '
             dc.b   'SEL  F1-F6      '
M1:          dc.b   'F1-GRABACION    '
             dc.b   'SEL  I/ESC      '
M11:         dc.b   'F1-GRABANDO     '
             dc.b   'ESC = FIN       '
M2:          dc.b   'F2-REPRODUCCION '
             dc.b   'SEL  I/ESC      '
M21:         dc.b   'F2-REPRODUCIENDO'
             dc.b   'ESC = FIN       '
M5:          dc.b   'F3 ESCRITURA MDR'
             dc.b   'SEL   I/ESC     '
M51:         dc.b   'F3              '
             dc.b   'TRANSMITIENDO   '
M6:          dc.b   'F4 LECTURA   MDR'
             dc.b   'SEL   I/ESC     '
M61:         dc.b   'F4 LECTURA   MDR'
             dc.b   'RECIBIENDO      '
M62:         dc.b   'F4 LECTURA MDR  '
```

```
                dc.b    'ESPERANDO SYSEX '
M3:             dc.b    'F5-TIEMPO       '
                dc.b    'ESC = FIN       '
M8:             dc.b    '    MENSAJE     '
                dc.b    '  TRANSMITIDO   '
```

A continuación se presenta la versión compilada del programa, tal como se encuentra grabado en la memoria EPROM del sistema.

```
000400 :                    1                org     $400
000400 :                    2                include "df1:Subrutinas/Sub_Direcciones"
000400 :                    3    Viewdata    equ     $102001
000400 :                    4    Viewport    equ     $102005
000400 :                    5    Viewport1   equ     $102005
000400 :                    6    Viewport2   equ     $102007
000400 :                    7    Viewport3   equ     $102009
000400 :                    8    Viewport4   equ     $10200a
000400 :                    9    Acia1status equ     $102041
000400 :                   10    Acia1control equ    $102041
000400 :                   11    Acia1transmit equ   $102043
000400 :                   12    Acia1reciv  equ     $102043
000400 :                   13    Acia2status equ     $102081
000400 :                   14    Acia2control equ    $102081
000400 :                   15    Acia2transmit equ   $102083
000400 :                   16    Acia2reciv  equ     $102083
000400 :                   17    timerCR1    equ     $1020c1
000400 :                   18    timerCR2    equ     $1020c3
000400 :                   19    timerCR3    equ     $1020c1
000400 :                   20    timerMSB1   equ     $1020c5
000400 :                   21    timerLSB1   equ     $1020c7
000400 :                   22    timerMSB2   equ     $1020c9
000400 :                   23    timerLSB2   equ     $1020cb
000400 :                   24    timerMSB3   equ     $1020cd
000400 :                   25    timerLSB3   equ     $1020cf
000400 :                   26    timerSR     equ     $1020c3
000400 :                   27                end
000400 :                   28                ;include "df1:Subrutinas/Sub_InitPerif"
000400 :                   29    comacial    equ     %00010101
000400 :                   30    comacia2    equ     %00010101
000400 :13FC00150010204l   31                move.b  #comacia1,acialcontro:
000408 :13FC00150010208l   32                move.b  #comacia2,acia2control
000410 :13FC00010010220C3  33                move.b  #1,timerCR2             ;Activa escritura a timer  1
000418 :13FC00010010220C1  34                move.b  #1,timerCR1             ;reset al 6840
000420 :13FC00000010220C3  35                move.b  #0,timerCR2             ;Activa escritura a tiemer 3
000428 :13FC00C3001020C1   36                move.b  #%11000011,timerCR3     ;timer 3 produce interrupciones
000430 :13FC00C3001020C9   37                move.b  #%11000011,timerCR2     ;timer 2 produce interrupciones
000438 :13FC00020010220C1  38                move.b  #%00000010,timerCR1     ;timer 1 no produce interrupciones
000440 :13FC00020010220C9  39                move.b  #$02,timerMSB2          ;Los diaplays se encienden    * X e
     s el valor del
000448 :13FC00000010220B   40                move.b  #$00,timerLSB2          ;(25000/X) veces/s (X= $0200) * tim
     er 2
000450 :13FC00043010220D   41                move.b  #$04,timerMSB3          ;El timer 3 funciona como MIDI
000458 :13FC00FF0010220F   42                move.b  #$ff,timerLSB3          ;Clock
000460 :                   43                end
000460 :                   44                include "df1:Subrutinas/Sub_Vectores"
000460 :21FC000008CC0064   45     move.l  #display,100          ;Autovector1
000468 :21FC000008CC0068   46                move.l  #display,104            ;Autovector2
000470 :21FC000008CC006C   47                move.l  #display,108            ;Autovector3
000478 :21FC0000079A0070   48                move.l  #acialinter,112         ;Autovector4
000480 :21FC0000079A0074   49                move.l  #acialinter,116         ;Autovector5
000488 :21FC000007C80078   50                move.l  #timerinter,120         ;Autovector6
000490 :21FC000008CC007C   51                move.l  #display,124            ;Autovector7
000498 :                   52                end
000498 :2E7C00000FFE       53                move.l  #$ffe,a7                ;SSP
00049E :46FC200F           54                move    #$200f,sr               ;activa interrups
0004A2 :21FC000009F809F2   55                move.l  #MM,screen              ;visualiza menu prinsipal
0004AA :                   56
0004AA :                   57
0004AA :                   58 ;*************************   Menus  ********************
0004AA :                   59
0004AA :21FC000009F809F2   60 start:        move.l  #MM,screen
0004B2 :61000274           61                bsr     GET2                  ;Obtiene un caracter del teclado
0004B6 :BA3C00C0           62                cmp.b   #192,d5               ;F1
0004BA :67000028           63 >              beq     RECORD
0004BE :BA3C00C1           64                cmp.b   #193,d5               ;F2
0004C2 :67000080           65                beq     PLAY
0004C6 :BA3C00C2           66                cmp.b   #194,d5               ;F3
0004CA :                   67               ; beq     WMDR
0004CA :BA3C00C3           68                cmp.b   #195,d5               ;F4
0004CE :                   69               ; beq     RMDR
```

```
0004CE :BA3C00C4          70                cmp.b   #196,d5                 ;F5
0004D2 :67000144          71                beq     TIME
0004D6 :BA3C00D5          72                cmp.b   #213,d5                 ;F6
0004DA :670001F4          73                beq     Panic
0004DE :BA3C00D6          74                cmp.b   #214,d5                 ;F7
0004E2 :                  75               ; beq     MTHRU
0004E2 :6006              76 >              bra     Start
0004E4 :                  77
0004E4 :                  78
0004E4 :                  79  ;********************* SUBRUTINA DE GRABACION ********************
0004E4 :                  80
0004E4 :                  81                include "df1:subrutinas/Sub_rec"
0004E4 :207C00020000      82    RECORD:         move.l  #$20000,a6
0004EA :31FC0000008C4     83                    move.w  #$0000,cuenta
0004F0 :21FC00000A1609F2  84                    move.l  #M1,screen
0004F8 :6100022E          85    perate:         BSR     get2
0004FC :BA3C00F0          86                    cmp.b   #240,d5
000500 :67A8              87 >                  beq     start
000502 :BA3C0049          88                    cmp.b   #$49,d5
000506 :66F0              89 >                  bne     perate
000508 :21FC00000A3609F2  90                    move.l  #M11,screen
000510 :31FC000008C6      91                    move.w  #$0,cuental
000516 :31FC000008CB      92                    move.w  #$0,cuarter
00051C :11FC000108CA      93                    move.b  #1,acticlick
000522 :610001F0          94    jk:             bsr     get1
000526 :3C3808C4          95                    move.w  cuenta,d6
00052A :E05E              96                    ror.w   #8,d6
00052C :1CC5              97                    move.b  d6,(a6)+
00052E :E15E              98                    rol.w   #8,d6
000530 :1CC6              99                    move.b  d6,(a6)+
000532 :31FC000008C4      100                   move.w  #$0000,cuenta
000538 :1CC5              101                   move.b  d5,(a6)+
00053A :610001DB          102                   bsr     get1
00053E :1CC5              103                   move.b  d5,(a6)+
000540 :610001D4          104                   bsr     get1
000544 :1CC5              105                   move.b  d5,(a6)+
000546 :1A3900102081      106   W1:             move.b  Acia2status,d5
00054C :CA3C0001          107                   and.b   #$1,d5
000550 :6700              108 >                 beq     jk
000552 :1A3900102083      109                   move.b  Acia2reciv,d5
000558 :BA3C00F0          110                   cmp.b   #240,d5
00055C :11FC000008CA      111                   move.b  #$0,acticlick
000562 :21CE0570          112                   move.l  a6,FINGRAB
000566 :6100020C          113                   bsr     alloff
00056A :6000FF3E          114                   bra     start
00056E :                  115                   align.l
000570 :00000000          116   FINGRAB:        dcl     $0000
000574 :                  117                   end
000574 :                  118
000574 :                  119 ;********************* SUBRUTINA DE EJECUCION ********************
000574 :                  120
000574 :                  121               Include "df1:subrutinas/Sub_Play"
000574 :21FC00000A5609F2  122   PLAY:           move.l  #M2,screen
00057C :207C00020000      123                   move.l  #$20000,a6
000582 :610001A4          124   perate1:        BSR     get2
000586 :BA3C00F0          125                   cmp.b   #240,d5
00058A :6700FF1E          126                   BEQ     Start
00058E :BA3C0049          127   cc1:            cmp.b   #$49,d5
000592 :66EE              128 >                 bne     perate1
000594 :21FC00000A7609F2  129                   move.l  #M21,screen
00059C :31FC000008C6      130                   move.w  #$0,cuental
0005A2 :31FC000008CB      131                   move.w  #$0,cuarter
0005A8 :11FC000108CA      132                   move.b  #1,acticlick
0005AE :BDFB0570          133   st1:            cmp.l   FINGRAB,a6
0005B2 :6200004C          134 >                 bhi     regresa
0005B6 :4286              135                   clr.l   d6
0005B8 :1C1E              136                   move.b  (a6)+,d6
0005BA :E186              137                   asl.l   #8,d6
0005BC :1C1E              138                   move.b  (a6)+,d6
0005BE :31FC000008C4      139                   move.w  #$0000,cuenta
0005C4 :3A3808C4          140   lp1:            move.w  cuenta,d5
```

```
0005CB :BC45                141                     cmp.w   d5,d6
0005CA :66F8                142 >                   bne     lp1
0005CC :1C1E                143                     move.b  (a6)+,d6
0005CE :6100016C            144                     bsr     out2
0005D2 :1C1E                145                     move.b  (a6)+,d6
0005D4 :61000166            146                     bsr     out2
0005D8 :1C1E                147                     move.b  (a6)+,d6
0005DA :61000160            148                     bsr     out2
0005DE :1A3900102081        149                     move.b  Acia2status,d5
0005E4 :CA3C0001            150                     and.b   #1,d5
0005E8 :E714                151 >                   beq     st1
0005EA :1A3900102083        152                     move.b  Acia2reciv,d5
0005F0 :BA3C00F0            153                     cmp.b   #240,d5
0005F4 :21FC000009F609F2    154     regresa:        move.l  #MM,screen
0005FC :11FC000008CA        155                     move.b  #0,acticlick
000602 :61000170            156                     bsr     alloff
000606 :6000FEA2            157                     bra     start
00060A :6100011C            158     W2:             BSR     GET2
00060E :BA3C00F0            159                     cmp.b   #240,d5
000612 :66F6                160 >                   BNE     W2
000614 :6000FE94            161                     bra     start
000618 :                    162                     end
000618 :                    163
000618 :                    164
000618 :                    165
000618 :                    166 ;****************** Ajuste de tiempo ********************
000618 :                    167
000618 :31FC000008C6        168 TIME:           move.w  #0,cuenta1
00061E :31FC000008CB        169                 move.w  #0,cuarter
000624 :11FC00010BCA        170                 move.b  #1,acticlick
00062A :21FC0000A9609F2     171                 move.l  #m3,screen
000632 :223806CC            172 otro:           move.l  ftime,d1
000636 :B2BC00001B6A        173                 cmp.l   #$186a,d1
00063C :6200007A            174 ;               bhi     sethi
000640 :B2BC00000300        175                 cmp.l   #$300,d1
000646 :65000076            176 >               blo     setlow
00064A :610000DC            177                 bsr     get2
00064E :BA3C00F0            178                 cmp.b   #240,d5
000652 :67000054            179 >               beq     termina
000656 :BA3C00EE            180                 cmp.b   #139,d5
00065A :6700000C            181 >               beq     mas
00065E :BA3C008A            182                 cmp.b   #135,d5
000662 :67000014            183 >               beq     menos
000666 :60CA                184 >               bra     otro
000668 :                    185
000668 :223806CC            186 mas:            move.l  ftime,d1
00066C :92BC00000030        187                 sub.l   #$30,d1
000672 :21C106CC            188                 move.l  d1,ftime
000676 :E099                189                 ror.l   #8,d1
000678 :13C1001020CD        190                 move.b  d1,timerMSB3
00067E :E199                191                 rol.l   #8,d1
000680 :13C1001020CF        192                 move.b  d1,timerLSB3
000686 :60AA                193 >               bra     otro
000688 :223806CC            194 menos:          move.l  ftime,d1
00068C :02BC00000030        195                 add.l   #$30,d1
000692 :21C106CC            196                 move.l  d1,ftime
000696 :E099                197                 ror.l   #8,d1
000698 :13C1001020CD        198                 move.b  d1,timerMSB3
00069E :E199                199                 rol.l   #8,d1
0006A0 :13C1001020CF        200                 move.b  d1,timerLSB3
0006A6 :608A                201 >               bra     otro
0006A8 :11FC000008CA        202 termina:        move.b  #0,acticlick
0006AE :6000FDFA            203                 bra     start
0006B2 :                    204
0006B2 :21FC00001B6A06CC    205 sethi:          move.l  #$186a,ftime
0006BA :6000FF76            206                 bra     otro
0006BE :21FC0000030006CC    207 setlow:          move.l  #$300,ftime
0006C6 :6000FFEA            208                 bra     otro
0006CA :                    209                 align.l
0006CC :00001046            210 ftime:          def     $1046
0006D0 :                    211
```

```
0006D0 :                    212
0006D0 :                    213
0006D0 :                    214 ;***************  Grabar en MDR ***************************
0006D0 :                    215
0006D0 :                    216             ; include "df1:Subrutinas/Sub_WMDR"
0006D0 :                    217
0006D0 :                    218
0006D0 :                    219 ;***************  LEER de MDR *****************************
0006D0 :                    220
0006D0 :                    221             ;  include "df1:Subrutinas/Sub_RMDR"
0006D0 :                    222
0006D0 :                    223 ;*****************************************************
0006D0 :                    224
0006D0 :                    225
0006D0 :21FC00000B9609F2    226 Panic:          move.l  #M6,screen
0006D8 :6100009A            227                 bsr     Alloff
0006DC :243CFFFFFFFF        228                 move.l  #$ffffffff,d2
0006E2 :223C00000005        229 retardo:        move.l  #$5,d1
0006E8 :4E71                230 retardo1:       nop
0006EA :51C9FFFC            231                 dbra    d1,retardo1
0006EE :51CAFFF2            232                 dbra    d2,retardo
0006F2 :6000FDB6            233                 bra     start
0006F6 :                    234
0006F6 :                    235
0006F6 :                    236
0006F6 :                    237
0006F6 :                    238 ********** leer programa desde amiga **********************
0006F6 :                    239
0006F6 :21FC00000B7609F2    240 LDAMIGA:        move.l  #M7,screen
0006FE :61000028            241 >W7:            BSR     GET2
000702 :BA3C00F0            242                 cmp.b   #240,d5
000706 :66F6                243 >               BNE     W7
000708 :21FC00000B8609F2    244                 move.l  #MM,screen
000710 :6000FD98            245                 bra     start
000714 :                    246 ;********** leer dump ************************
000714 :                    247 ;
000714 :                    248 ;gRMDR:           move.l  #bufini,a1
000714 :                    249 ;                 move.l  #bufini,a0
000714 :                    250 ;                 move.l  #$20000,a5
000714 :                    251 ;                 move.l  a5,a5
000714 :                    252 ;                 move.b  #%10010101,acialcontrol
000714 :                    253 ;                 move.l  #M6,screen
000714 :                    254 ;gpepera2:        BSR     get2
000714 :                    255 ;                 cmp.b   #240,d5
000714 :                    256 ;                 beq     start
000714 :                    257 ;                 cmp.b   #$49,d5
000714 :                    258 ;                 bne     gpepera2
000714 :                    259 ;                 move.l  #M62,screen
000714 :                    260 ;gww1:            bsr     bget1
000714 :                    261 ;                 cmp.b   #$f0,d5
000714 :                    262 ;                 bne     gww1
000714 :                    263 ;gww2:             bsr     bget1
000714 :                    264 ;                 cmp.b   #65,d5
000714 :                    265 ;                  bne     gww2
000714 :                    266 ;                 move.l  #$7fff,d2
000714 :                    267 ;grg:             bsr     bget1
000714 :                    268 ;                 move.b  d5,(a6)+
000714 :                    269 ;                 dbra    d2,grg
000714 :                    270 ;                 bra     start
000714 :                    271
000714 :                    272 ;***************  Obtiene un caracter de Amiga o MIDI ***************
000714 :                    273
000714 :1A3900102041        274 GET1:           move.b  Acialstatus,d5
00071A :CA3C0001            275                 and.b   #1,d5
00071E :67F4                276 >               beq     GET1
000720 :1A3900102043        277                 move.b  Acialreciv,d5
000726 :4E75                278                 RTS
000728 :                    279
000728 :                    280 ;***************  Obtiene un caracter del teclado ***************
000728 :                    281
000728 :1A3900102061        282 GET2:           move.b  Acia2status,d5
```

```
00072E :CA3C0001          263                and.b   #1,d5
000732 :67F4              264 ,              beq     GET2
000734 :1A390001020B      265                move.b  Acia1reciv,d5
00073A :4E75              266                RTE
00073C :                  267  ;*************************************************************
00073C :                  268
00073C :1A390001020E1     269  OUT1:         move.b  Acia2status,d5
000742 :CA3C0002          270                and.b   #2,d5
000746 :67F4              271 ,              beq     out2
000748 :13C6001020B5      272                move.b  d6,acia2transmit
00074E :4E75              273                rts
000750 :                  274
000750 :                  275  ;*************************************************************
000750 :                  276  Bgeti:
000750 :0C39000000BBE     277                cmp.b   #0,count
000756 :67F6              278 ,              beq     Bgeti
000758 :0439000010BB6     279                sub.b   #1,count
00075E :1A19              280                move.b  (a1)+,d5
000760 :                  281
000760 :B3FC00000BCB      282                cmp.l   #bufIN1+20,a1
000766 :62000004          283 ,              bhi     cont
00076A :4E75              284                rts
00076C :227C00000BE7      285  cont:         move.l  #bufIN1,a1
000772 :4E75              286                rts
000774 :                  287  ;*************************************************************
000774 :                  288
000774 :                  289  ;          ALL NOTES OFF          D6
00077A :                  290
000774 :2[illegible]      291  ALLOFF:       move.l  #15,d1
00077A :1[illegible]      292                move.l  #$AF,d2
000780 :0439000001        293  lpp:          add.b   #1,d2
000784 :1001              294                move.b  d2,d6
000786 :61B4              295 ,              bsr     out2
000788 :1C3C007B          296                move.b  #$7b,d6
00078C :61AE              297 ,              bsr     out2
00078E :1C3C0000          298                move.b  #$00,d6
000792 :61A6              299 ,              bsr     out2
000794 :51C9FFEA          300                dbra    d1,lpp
000798 :4E75              301                rts
00079A :                  302  ;*************************************************************
00079A :                  303
00079A :                  304  Acia1inter:
00079A :1E3900010204      305                move.b  Acia1reciv,d7
0007A0 :0C38000A0BBE      306                cmp.b   #10,count
0007A6 :6E000004          307 ,              bic     con
0007AA :4E73              308                rte
0007AC :0638000010BB6     309  con:          add.b   #1,count
0007B2 :10C7              310                move.b   d7,(a0)+
0007B4 :B1FC00000BCB      311                cmp.l   #bufIN1+20,a0
0007BA :62000004          312 ,              bhi     cont1
0007BE :4E73              313                rte
0007C0 :227C00000BE7      314  cont1:        move.l  #bufIN1,a0
0007C6 :4E73              315                rte
0007C8 :                  316  ;*************************************************************
0007C8 :                  317                include "df1:subrutinas/sut_timerex"
0007C8 :48E7FUFU          318     timerinter:    movem.l  d0-d3/a0-a3,-(sp)
0007CC :1039001020C3      319                    move.b  timerSR,d0
0007D2 :C03C0002          320                    and.b   #2,d0
0007D6 :67000016          321 ,                  beq     ff
0007DA :1039001020C9      322                    move.b  timerMSB2,d0
0007E0 :1039001020CB      323                    move.b  timerLSB2,d0
0007E6 :610000E4          324                    bsr     Display
0007EA :6000001E          325 ,                  bra     ya
0007EE :1039001020CD      326     ff:            move.b  timerMSB3,d0
0007F4 :1039001020CF      327                    move.b  timerLSB3,d0
0007FA :61000000          328 ,                  bsr     MIDIclock
0007FE :61000010          329 ,                  bsr     Click
000802 :4CDF0F0F          330     ya:            movem.l (SP)+,d0-d3/a0-a3
000806 :4E73              331                    RTE
000808 :0678000108C4      332     MIDIclock:     add.w   #1,cuenta
00080E :4E75              333                    rts
```

```
Address Code            Line   Source text                                  Page   6

000810 :0C380000108CA    354    Click:      cmp.b    #1,acticlick
00081E :67000004         355 >              beq      sigue
000824 :4E75             356                rts
000826 :0C7B0001608C6    357    sigue       add.w    #1,cuenta1
00082C :0C7B0001608C6    358                cmp.w    #24,cuenta1
000832 :67000004         359 >              beq      Cambio
000838 :4E75             360                rts
00083A :                 361
00083C :207B09F2         362    Cambio:     move.l   screen,a0
000842 :D1FC0000001C     363                add.l    #28,a0
000848 :10FC0020         364                move.b   #$20,(a0)+
00084C :10FC0020         365                move.b   #$20,(a0)+
000850 :10FC0020         366                move.b   #$20,(a0)+
000854 :10FC0020         367                move.b   #$20,(a0)+
000858 :31FC0000008C6    368                move.w   #$0,cuenta1
00085E :0638000108C8     369                add.b    #1,cuarter
000864 :0C380001080B     370                cmp.b    #1,cuarter
00086A :67000022         371 >              beq      Cuarter1
000870 :0C38000208CB     372                cmp.b    #2,cuarter
000876 :67000028         373 >              beq      Cuarter2
00087C :0C380003080B     374                cmp.b    #3,cuarter
000882 :6700002E         375 >              beq      Cuarter3
000888 :0C380004080B     376                cmp.b    #4,cuarter
00088E :67000034         377 >              beq      Cuarter4
000894 :4E75             378                rts
000896 :                 379
000898 :                 380    cuarter1:
00089A :207B09F2         381                move.l   screen,a0
0008A0 :D1FC0000001C     382                add.l    #28,a0
0008A6 :10BC0031         383                move.b   #$31,(a0)
0008AA :4E75             384                rts
0008AC :                 385
0008AE :                 386    cuarter2:
0008B0 :207B09F2         387                move.l   screen,a0
0008B6 :D1FC0000001D     388                add.l    #29,a0
0008BC :10BC0032         389                move.b   #$32,(a0)
0008C0 :4E75             390                rts
0008C2 :                 391
0008C4 :                 392    cuarter3:
0008C6 :207B09F2         393                move.l   screen,a0
0008CC :D1FC0000001E     394                add.l    #30,a0
0008D2 :10BC0033         395                move.b   #$33,(a0)
0008D6 :4E75             396                rts
0008D8 :                 397
0008DA :11FC0000008C8    398    cuarter4:   move.b   #$0,cuarter
0008E0 :207B09F2         399                move.l   screen,a0
0008E6 :D1FC0000001F     400                add.l    #31,a0
0008EC :10BC0034         401                move.b   #$34,(a0)
0008F0 :4E75             402                rts
0008F2 :                 403
0008F4 :                 404
0008F6 :                 405                align.w
0008F8 :007B             406    cuenta:     defw  123
0008FA :0001             407    cuenta1:    defw  #1
0008FC :0000             408    cuarter:    defw  #0
0008FE :00               409    acticlick:  defb  $0
000900 :00               410    thru:       defb  $0
000902 :                 411                end
000904 :                 412              include "df1:subrutinas/sub_display"
000906 :203809DE         413    Display:      move.l  data0,d0          ;Solo se enciende un display
00090C :223809E2         414                  move.l  data1,d1          ;en total 32 caracteres
000912 :2E3809E6         415                  move.l  data3,d3          ;Apaga todos los displays
000918 :283809EA         416                  move.l  data4,d4
00091E :207809EE         417                  move.l  raster,a0         ;inicio de desplige en a0
000924 :43F900102005     418                  lea     viewport1,a1      ;direccion de los displays en a1
00092A :45F900102001     419                  lea     viewdata,a2       ;direccion de datos de display en a2
000930 :4282             420                  clr.l   d2
000932 :1418             421                  move.b  (a0)+,d2          ;carga un caracter ascii en d2
000934 :943C0020         422                  sub.b   #$20,d2
000938 :E35A             423                  rol.w   #1,d2
00093A :267C00000968     424                  move.l  #tabla,a3          ;inicio de tabla de conversion ascii
```

```
0008FC :34332000          425                   move.w  0(a3,d2),d2       ;carga d2 con un codigo de display
000900 :07C80000          426                   movep.l d3,0(a1)          ;apaga todos los displays
00090A :05BA0000          427                   movep.w d2,0(a2)          ;envia datos de caracter
000908 :13C000102005      428                   move.b  d0,Viewport1      ;*
00090E :E098              429                   ror.l   #8,d0             ;*
000910 :13C000102007      430                   move.b  d0,Viewport2      ;*
000916 :E098              431                   ror.l   #8,d0             ;* selecciona el display a encender
000918 :13C000102009      432                   move.b  d0,Viewport3      ;*
00091E :E098              433                   ror.l   #8,d0             ;*
000920 :13C00010200A      434                   move.b  d0,Viewport4      ;*
000926 :EE98              435                   ror.l   #7,d0             ;proximo display en d0
000928 :51C8000C          436                   dbra    d1,cont           ;Si no a sacado los 32 caracteres continua
00092C :207B09F2          437                   move.l  screen,a0
000930 :223C0000001F      438                   move.l  #31,d1
000936 :21C009DE          439   cont:           move.l  d0,data0
00093A :21C108E2          440                   move.l  d1,data1
00093E :21C309E6          441                   move.l  d3,data3
000942 :21C809EE          442                   move.l  a0,raster
000946 :4E75              443                   rts
000948 :                  444                   Align.w                   ;Forza los datos a una direccion par
00094A :414C474F204D4152  445   mensaje:        dc.b    'ALGO MARAVILLOSO'
       4156649C4C4F534F
000958 :5365343454452E2E  446                   dc.b    'SUCEDE..... VIVO'
       2E2E2E20564964F
000968 :FFFFBFF57FDFEBAF  447   tabla:          defw    $ffff,$bff5,$7fdf,$ebaf,$4c6b ;  , !, ", #, $
       4C6B
000972 :4AAB7939FFBFFFBD  448                   defw    $4aab,$7939,$ffbf,$ffbd,$bbff ; %, &, ', (, )
       BBFF
00097C :2AAD6EEFFBFFEFEF  449                   defw    $2aad,$6eef,$fbff,$efef,$fff7 ; *, +, ,, -, .
       FFF7
000986 :FBBF              450                   defw    $fbbf,                        ; /
000988 :D11B7EFFE54FED4B  451                   defw    $d11b,$7eff,$e54f,$ed4b,$cfcb ;0,1,2,3,4
       CFCB
000992 :CD6BBC56BFF5BC54B 452                   defw    $cd6b,$c56b,$ff5b,$c54b,$cf4b ;5,6,7,8,9
       CF4B
00099C :FFFFFFFFFFFFEDEF  453                   defw    $ffff,$ffff,$ffff,$edef,$ffff ;:,;,<,=,>
       FFFF
0009A6 :FFFFFFFF          454                   defw    $FFFF,$FFFF                   ;?,@
0009AA :C74B7C4BD57F7C5B  455                   defw    $c74b,$7c4b,$d57f,$7c5b,$c57f ;A,B,C,D,E
       C57F
0009B4 :C77FD56BC7CB7C7F  456                   defw    $c77f,$d56b,$c7cb,$7c7f,$f5db ;F,G,H,I,J
       F5DB
0009BE :C7BDD5FF979B97D9  457                   defw    $c7bd,$d5ff,$979b,$97d9,$d55b ;K,L,M,N,O
       D55B
0009C8 :C74FD559C74DCD6B  458                   defw    $c74f,$d559,$c74d,$cd6b,$7e7f ;P,Q,R,S,T
       7E7F
0009D2 :D5DBD3BFD3D9BBBD  459                   defw    $d5db,$d3bf,$d3d9,$bbbd,$bebf ;U,V,W,X,Y
       BEBF
0009DC :F93F              460                   defw    $f93f                         ;Z
0009DE :FFFFFFFE          461   data0:          defl    $fffffffe
0009E2 :0000001F          462   data1:          defl    31
0009E6 :FFFFFFFF          463   data3:          defl    $ffffffff
0009EA :0000FFFF          464   data4:          defl    $ffff
0009EE :00000948          465   raster:         defl    mensaje
0009F2 :00000948          466   screen:         defl    mensaje
0009F6 :                  467                   end
0009F6 :4D454E5520505249  468   MM:             dc.b    'MENU PRINCIPAL  '
       4E43495041C2020
000A0E :53454C2020463120  469                   dc.b    'SEL  F1-F20     '
       4632302020202020
000A16 :46312D4752414241  470   M1:             dc.b    'F1-GRABACION    '
       43494F4E20202020
000A26 :53454C2020492F45  471                   dc.b    'SEL  I/ESC      '
       5343202020202020
000A36 :46312D4752414241  472   M11:            dc.b    'F1-GRABANDO     '
       4E444F2020202020
000A46 :455343203D204649  473                   dc.b    'ESC = FIN       '
       4E20202020202020
000A56 :46322D524550524F  474   M2:             dc.b    'F2-REPRODUCCION '
       4A554343494F4E20
000A66 :53454C2020492F45  475                   dc.b    'SEL  I/ESC      '
```

```
Address Code                Source text
        5343202020202020
000A76 :46322D524550524F    M21:   dc.b   'F2-REPRODUCIENDO'
        4455434945 4E444F
000A86 :455343203D20464 9          dc.b   'ESC = FIN       '
        4E20202020202020
000A96 :46332D5449454D50    M3:    dc.b   'F3-TIEMPO       '
        4F20202020202020
000AA6 :455343203D204649           dc.b   'ESC = FIN       '
        4E20202020202020
000AB6 :46342D4D49444920    M4:    dc.b   'F4-MIDI THRU    '
        5448525520202020
000AC6 :53454C2020412020           dc.b   'SEL  A , I      '
        2049202020202020
000AD6 :4635204553435249    M5:    dc.b   'F5 ESCRITURA MDR'
        545552412 04D4452
000AE6 :53454C202020492F           dc.b   'SEL   I/ESC     '
        4553432020202020
000AF6 :4635202020202020    M51:   dc.b   'F5              '
        2020202020202020
000B06 :5452414E534D4954           dc.b   'TRANSMITIENDO   '
        49454E444F202020
000B16 :4636204C45435455    M6:    dc.b   'F6 LECTURA   MDR'
        524120202 04D4452
000B26 :53454C202020492F           dc.b   'SEL   I/ESC     '
        4553432020202020
000B36 :4636204C45435455    M61:   dc.b   'F6 LECTURA   MDR'
        5241202020 4D4452
000B46 :524543495649454E           dc.b   'RECIVIENDO      '
        444F202020202020
000B56 :4636204C45435455    M62:   dc.b   'F6 LECTURA MDR  '
        5241204D44522020
000B66 :4553504552414E44           dc.b   'ESPERANDO SYSEX '
        4F20535953455820
000B76 :4341524741522050    M7:    dc.b   'CARGAR PROGRAMA '
        524F4752414D4120
000B86 :444553444520414D           dc.b   'DESDE AMIGA     '
        4947412020202020
000B96 :202020204D454E53    M8:    dc.b   '    MENSAJE     '
        414A452020202020
000BA6 :202054524 14E534D          dc.b   '  TRANSMITIDO   '
        495449444F202020
000BB6 :00                  count  dc.b   0
000BB7 :00                  bufini dc.b   0
000BB8 :                           end
```

CONCLUSIONES

Gracias a los avances de la electrónica, actualmente se pude encontrar en el mercado un gran número de dispositivos especializados, que constituyen por si solos un bloque en un sistema electrónico. Como un ejemplo de esto, encontramos a los microcontroladores, que integran casi todos los elementos necesarios para construir sistemas basados en microprocesadores. Si bien casi todos las microcomputadoras, minicomputadoras, y sistemas de tamaños similares estan construidos en base a un microprocesador, existe un gran número de dispositivos electrónicos, en los cuales el mejor camino de implementación es usar un microcontrolador. De la misma forma en que la alta integración permite la construcción de microcontroladores, tambien permite la construcción de dispositivos especializados en una area determinada, como los controladores de memoria, circuitos de video, controladores de teclados, controladores de displays, displays integrados, arreglos lógicos programables, convertidores analógico-digitales, etc., etc., etc. En un mundo repleto de dispositivos como estos, parecen alejarse cada vez mas los dias en que la construcción de cada bloque se debia diseñar, probar y construir en forma discreta. La demanda de dispositivos como los mencionados, permite que los fabricantes los produzcan en grandes cantidades, brindando la posibilidad de adquirirlos a precios relativamente bajos, comparados con sistemas armados discretamente. Dentro de este contexto, parece que la forma mas sencilla, facil, rápida y económica de diseñar y construir sistemas electrónicos, es basar el diseño en la medida de lo posible en componentes que constituyan por si mismos, bloques enteros del sistema. Por supuesto, lo anterior no es siempre cierto, ya que existen campos en los que no se han producido comercialmente dispositivos dedicados a cierta area o, los requerimientos superan a los dispositivos comerciales, sin embargo, en el campo del diseño comercial, esto no es el caso general, y si existe la tendencia hacia el uso de dispositivos especializados.

En el campo de la música, el MIDI ha sido sin duda el avance mas importante de los ultimos tiempos, ademas de su uso tradicional, existen campos como el control de mezclas, sincronia con video y control de paneles de luz, donde el MIDI ya es usado y su uso continua extendiendose. Como un ejemplo de ello se encuentran los nuevos reproductores de discos compactos (CD+G+M), que incluyen puertos MIDI. Existen tambien computadoras como las ATARI, YAMAHA y otras, que en su mas sencilla configuración incluyen puertos MIDI.

Cuando el estandar MIDI fue creado, se penso en un estandar que pudiera ser expandido y se dejaron un gran número de controles para futuras definiciones. La asociación internacional de MIDI esta trabajando ya en campos como el intercambio de información de dispositivos muestreadores (comercialmente conocidos como samplers), y en la definición de nuevos mensajes con fines de sincronia, que cuenten horas, minutos, segundos y cuadros, tal como lo hace el SMPTE. Con todo esto y la tremenda competencia de los sintetizadores, el estandar tendra que continuar refinandose. Sin embargo, algunas de las caracteristicas del estandar MIDI, como la ralativamente baja taza de transmisión (31,250 bits por segundo) y su tendencia a saturarse con cierto tipo de mensajes, lo limitan, sobre todo ahora, que la tecnologia permite crear sintetizadores con caracteristicas, que crean la necesidad de transmitir mas y mas datos. De hecho ciertas definiciones del estandar son ya obsoletas, ya que los sintetizadores actuales no tienen las limitaciones técnicas que dieron origen a esas definiciones. Ademas de esto, el número de canales disponibles (16)

resulta insuficiente en ciertos casos. Aunque esto se puede resuelve dividiendo el sistema en varios subsistemas, complica el manejo y resulta ser un tipo de parche para el estandar.

Todo esto, parece indicar que surgira un nuevo estandar, pensado para funcionar y expandirse con una tecnologia mas avanzada. Sin embargo, a pesar de las limitaciones que presenta el estandar MIDI, trabaja suficientemente bien en general, y el mercado se encuentra inundado por él, por lo que parece dificil que en el futuro (almenos en un futuro cercano) sea sustituido por algun otro estandar.

Una de las razones que me motivaron para diseñar e implementar el secuenciador MIDI, es el hecho de que en nuestro pais prácticamente no existe ningun lugar, en donde se puedan adquirir los accesorios para implementar un sistema musical controlado por MIDI, como programas para computadora o tarjetas MIDI, aun cuando dichos accesorios son electrónicamente sencillos de implementar. Esto representa un problema para la mayor parte de la gente que en nuestro pais se interesa por adquirir un sistema MIDI. Es evidente tambien, que este tema es desconocido para gran parte de los ingenieros en electrónica y que existe un creciente mercado de accesorios, no sintetizadores o computadoras en donde no contamos con la infraestructura para competir, sino accesorios secundarios como sincronizadores, interfaces, MDR's, cartuchos, cables, pedales de expresión, paneles de luz, regeneradores, displays de SMPTE, etc, asi como servicio para ellos; dichos accesorios pueden ser fabricados en México. Ademas de esto, existe una carencia total de bibliografia acerca de este tema (por lo menos en el aspecto del hardware). Debido a esto, hubo varios problemas en el desarrollo del secuenciador, como tener que esperar cerca de un mes para reunir la bibliografia necesaria, para que una vez reunida se pudiera diseñar el sistema y esperar de nuevo otro mes para conseguir los componentes. Esto trajo consigo otros problemas, debido a que no fue posible conseguir todos los componentes del diseño original y se tubo que cambiar parcialmente (en este caso se tubo que optar por memoria estatica al no conseguir el controlador de memoria dinamica). Con todo esto, es un poco dificil realizar un diseño optimo si se desea implementar en un tiempo razonable. Esta quizas es la razón por la cual no existe un interes apreciable por parte de ingenieros independientes por llenar el vacio existente en este campo en nuestro pais.

APENDICE A

TABLAS DE ESPECIFICACION MIDI 1.0

Las siguientes tablas son una reproducción de las paginas 9 a 18 de las hojas de especificación de la asociación internacional de MIDI.

Los terminos referentes a mensajes MIDI se manejarán en ingles ya que resulta más sencillo buscarlos por el nombre en que se presentan en la práctica. En el capitulo 2 se encuentran todos los mensajes MIDI explicados con detalle.

TABLA 1

Resumen de bytes de estatus

HEX	STATUS Binario D7-D0	NUMERO DE BYTES DE DATOS	DESCRIPCION
MENSAJES DE VOZ			
8nH	1000nnnn	2	Note off
9nH	1001nnnn	2	Note on (una velocidad de cero es igual a Nota OFF)
AnH	1010nnnn	2	Polyphonic key pressure/ Aftertouch
BnH	1011nnnn	2	Control change
CnH	1100nnnn	1	Program change
DnH	1101nnnn	1	Channel pressure/ Aftertouch
EnH	1110nnnn	2	Pitch bend change
MENSAJES DE MODO			
BnH	1011nnnn (01111xxx)	2	Selects channel Mode
MENSAJES DEL SISTEMA			
F0H	11110000	*****	System Exclusiv
	11110sss	0 a 2	System Common
	11111ttt	0	System Real Time

NOTAS:

- **nnnn:** N-1, donde N= Número de canal, por ejemplo 0000 es Canal 1, 0001 es Canal 2 y 1111 es Canal 16
- *******:** 0iiiiiii, datos, ..., EOX
- **iiiiiii:** identification
- **sss:** 1 a 7
- **ttt:** 0 a 7
- **xxx:** Los mensajes de modo de canal son transmitidos bajo el mismo byte de status que el mensaje de cambio de control (BnH). Son diferenciados por el primer byte de datos el cual tendrá un valor entre 121 y 127 para el mensaje de modo de canal.

TABLA II

MENSAJES DE VOZ (CHANEL VOICE MESSAGES)

HEX	STATUS Binario	DATA BYTES	DESCRIPCION
8nH	1000nnnn	0kkkkkkk 0vvvvvvv	Note Off vvvvvvv:note off velocity
9nH	1001nnnn	0kkkkkkk 0vvvvvvv	Note On vvvvvvv `' 0 :velocity vvvvvvv = 0 :note off
AnH	1010nnnn	0kkkkkkk 0vvvvvvv	Poliphnic key Pressure (Aftertouch) vvvvvvv: pressure value
BnH	1011nnnn	0ccccccc	Control Change(ver Tabla III) ccccccc: control #(0-120) vvvvvvv: control value ccccccc = 121 a 127: reservado (ver tabla 4)
CnH	1100nnnn	0ppppppp	Program Change ppppppp: program nummber
DnH	1101nnnn	0vvvvvvv	Chanel pressure (Aftertouch) vvvvvvv: pressure value
EnH	1110nnnn	0vvvvvvv 0vvvvvvv	Pitch Bend Change LSB Pitch Bend Change MSB

NOTAS:

1.- nnnn: Número de canal de la voz (Voice Channel number 1-16) (veer las notas de la tabla 1)

2.- kkkkkkk: número de nota (0-127)

3.- vvvvvvv: Key velocity (se recomienda una escala logaritmica)

4.- Los controles continuos son divididos en byte Mas significativo y Menos significativo.Si solo son necesarios 7 bits de resolución para algún control en particular, solo se envia el byte mas significativo. No es necesario transmitir el byte menos significativo. Si se necesita más resolución, entonces se envian ambos bytes, primero el más significativo y despues el menos significativo. Si solo el byte menos significativo ha cambiado de valor, entonces se puede transmitir sin retransmitir el byte más significativo.

TABLA III

NUMEROS DE CONTROLES

NUMERO DEL CONTROL (VALOR DEL SEGUNDO BYTE)		FUNCION DEL CONTROL
DECIMAL	HEX	
0	00H	No definido
1	01H	Modulation wheel or lever
2	02H	Breath Controller
3	03H	No definido
4	04H	Foot Controller
5	05H	Portamento time
6	06H	Data entry MSB
7	07H	Main volume
8	08H	Balance
9	09H	No definido
10	0AH	Pan
11	0BH	Expression Controller
12-15	0CH-0FH	No definido
16-19	10H-13H	General Purpose Controllers (# s 1-4)
20-31	14H-1FH	No definido
32-63	20H-3FH	LSB for values 0-31
64	40H	Damper pedal (sustain)
65	41H	Portamento
66	42H	Sostenuto
67	43H	Soft Pedal
68	44H	No definido
69	45H	Hold 2
70-79	46H-4FH	No definido
80-83	50H-53H	General Purpose Controllers (# s 5-8)
84-90	54H-5AH	No definido
91	5BH	External Effects Depth
92	5CH	Tremolo Depth
93	5DH	Chorus Depth
94	5EH	Celeste (Detune) Depth
95	5FH	Phaser Depth
96	60H	Data increment
97	61H	Data Decrement
98	62H	Non-Registered Parameter Number LSB
99	63H	Non-Registered Parameter Number MSB
100	64H	Register Parameter Number LSB
101	65H	Register Parameter Number MSB
102-120	66H-78H	No definido
121-127	79H-7FH	Reservado para mensajes de modo

TABLA IIIa

NUMEROS DE PARAMETRO REGISTRADOS

NUMERO DE PARAMETRO LSB	MSB	FUNCION
00H	00H	Pitch Bend Sensitivity
01H	00H	Fine Tuning
02H	00H	Coarse Tuning

TABLA IV

MENSAJES DE MODO (CHANNEL MODE MESSAGES)

HEX	STATUS BINARIO	DATA BYTES	DESCRIPCION
BnH	1011nnnn 0vvvvvvv	0ccccccc	Mode Messages

ccccccc = 121: Reset All Controllers
vvvvvvv = 0

ccccccc = 122: Local Control
vvvvvvv = 0 , Local Control Off
vvvvvvv = 127, Local Control On

ccccccc = 123: All Notes Off
vvvvvvv = 0

ccccccc = 124: Omni Mode Off (All Notes Off)
vvvvvvv = 0

ccccccc = 125: Omni Mode On (All Notes Off)
vvvvvvv = 0

ccccccc = 127: Mono Mode On (Poly Mode Off)
(All Notes Off)
vvvvvvv = M, donde M es el número de canales
vvvvvvv = 0, El número de canales igual al número de voces en el receptor

ccccccc = 127: Poly Mode On (Mono Mode Off)
(All Notes Off)
vvvvvvv = 0

Notas:

1.- nnnn: Número de canal (1-16)

2.- ccccccc: Número de control (121-127)

3.- vvvvvvv: Valor del control

TABLA V

MENSAJES COMUNES (SYSTEM COMMON MESSAGES)

STATUS HEX	STATUS BINARIO	DATA BYTES	DESCRIPCION
F1H	11110001	0nnnnddd	MIDI Time Code Quarter Frame nnn: Message Type dddd: Values
F2H	11110010	01111111 0hhhhhhh	Song Position Pointer 1111111: (Least significant) hhhhhhh: (Most significant)
F3H	11110011	0sssssss	Song Select sssssss: Song #
F4H	11110100		No definido
F5H	11110101		No definido
F6H	11110110	none	Tune Request
F7H	11110111	none	EOX:"End of System Exclusive"flag

TABLA VI

MENSAJES DE TIEMPO REAL (SYSTEM REAL TIME MESSAGES)

STATUS HEX	STATUS BINARIO	DATA BYTES	DESCRIPCION
F8H	11111000		Timing Clock
F9H	11111001		No definido
FAH	11111010		Start
FBH	11111011		Continue
FCH	11111100		Stop
FDH	11111101		No definido
FEH	11111110		Active Sensing
FFH	11111111		System Reset

TABLA VII

MENSAJES DE SISTEMA EXCLUSIVO (SYSTEM EXCLUSIVE MESSAGES)

STATUS HEX	STATUS BINARIO	DATA BYTES	DESCRIPCION
FOH	11110000		Bulk dump, etc.
		0iiiiiii	iiiiiii:identificacion,nota 1
		.	
		(0ddddddd)	
		.	Aqui se pueden transmitir
		.	cualquier número de datos,
		.	para cualquier proposito,
		.	siempre y cuando tengan el
		.	bit mas significativo en
		.	cero
		(0ddddddd)	
F7H	11110111		EOX: "FIN DE SISTEMA EXCLUSIVO"

Notas:

1.- iiiiiii: identificacion ID (0-127). Si el ID es 0 los siguientes dos bytes son usados como una extension de la identificacion del fabricante.

2.- Todos los bytes entre el byte de status del sistema exclusivo y el mensaje EOX deben tener el bit mas significativo en cero.

3.- El número de identificacion para un fabricante puede ser obtenido en la MMA (Asociacion de comerciantes de musica) o en la JMSC.

4.- Cualquier byte de estatus o de datos (excepto los mensajes de tiempo real) no deben estar entrelazados con bytes de sistema exclusivo.

5.- Si se recibe algun byte de datos o de status (excepto en el caso de mensajes de tiempo real) antes de que le mensaje EOX haya sido recibido, se da por finalizado el mensaje de sistema exclusivo.

6.- Tres números de identificacion de sistema exclusivo han sido reservados para propositos especiales: 7DH esta reservado para usos no comerciales (por ejemplo en escuelas o investigación) y no será usado en productos realizados para el público. 7EH (no tiempo real) y 7FH (tiempo real) son usados como extensión de las especificaciones MIDI.

TABLA VIII

NUMEROS DE IDENTIFICACION DE SISTEMA EXCLUSIVO
(CURRENTLY DEFINED UNIVERSAL SYSTEM EXCLUSIVE ID NUMBERS)

SUB-ID#1	SUB-ID#2	DESCRIPCION
None-Real Time (7EH)		
00	--	no usado
01	(not used)	Sample Dump Header
02	(not used)	Sample Data Packet
03	(not used)	Sample Dump Request
04	nn	MIDI Time Code
	00	Special
	01	Punch In Points
	02	Punch Out Points
	03	Delete Punch In Point
	04	Delete Punch Out Point
	05	Event Start Point
	06	Event Stop Point
	07	Event Start Point with additional info.
	08	Event Stop Point with additional info.
	09	Delete Event Start Point
	0A	Delete Event Stop Point
	0B	Cue Points
	0C	Cue Points with additional info.
	0D	Delete Cue Point
	0E	Event Name in additional Info
05	nn	Sample Dump Extensions
	01	Multiple Loop Points
	02	loop points Request
06	nn	General Information
	01	Identity Request
	02	Identity Reply
7C	(non used)	Wait
7D	(not used)	Cancel
7E	(not used)	NAK
7F	(not used)	ACK
Real Time (7FH)		
00	--	Unused
01	nn	MIDI Time Code
	01	Full Message
	02	User Bits

Notas:

1.- El formato estandar para mensajes de tiempo real y no tiempo real es el siguiente:: F0H <id number> <channel number> <sud-ID#1>

<sub-ID#2>.....F7H

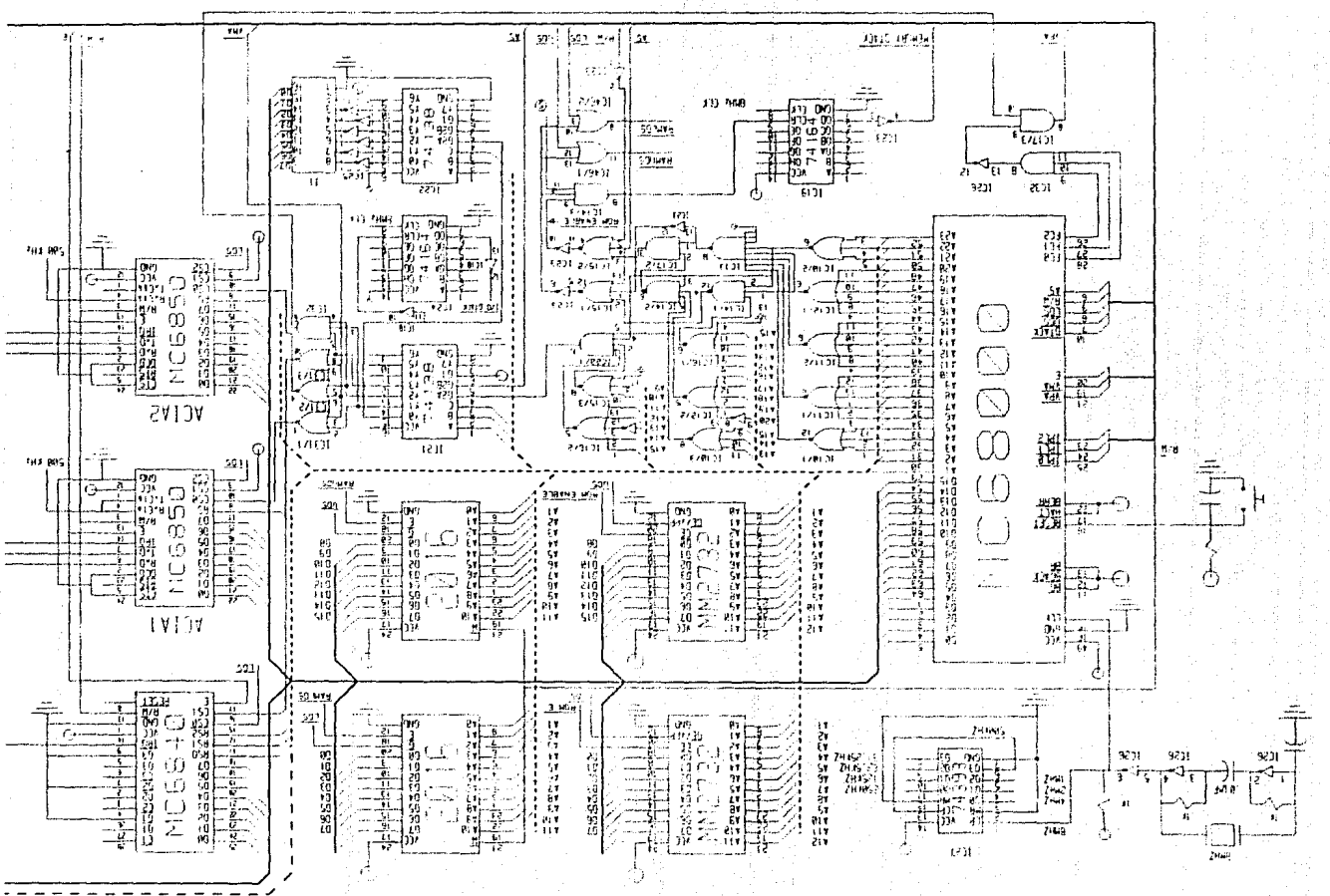
TABLA IX

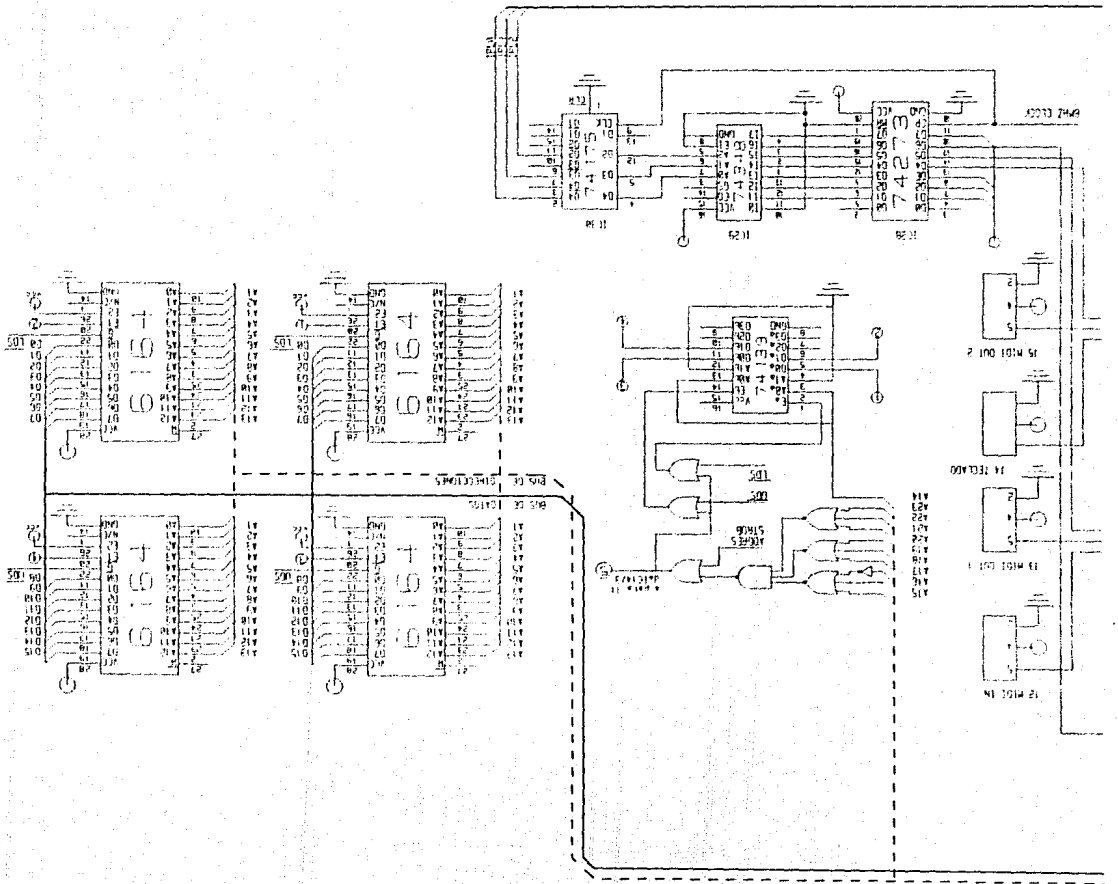
NUMERO DE IDENTIFICACION DE FABRICANTES
(SYSTEM EXCLUSIVE MANUFACTURER'S ID NUMBERS)

NUMERO	FABRICANTE	NUMERO	FABRICANTE
	GRUPO AMERICANO		GRUPO EUROPEO
01H	Sequential	20H	Passac
02H	IDP	21H	SIEL
03H	Octave-Plateau	22H	Synthaxe
04H	Moog	24H	Hohner
05H	Passport Designs	25H	Twister
06H	Lexicon	26H	Solton
07H	Kurzweil	27H	Jellinghaus MS
08H	Fender	28H	Southworth
0AH	AKG Acoustics	29H	PPG
0BH	Voyce Music	2AH	JEN
0CH	Waveframe Corp	2BH	SSL Limited
0DH	ADA	2CH	Audio Veritrieb
0EH	Garfield Elec.	2FH	Elka
0FH	Ensoniq	30H	Dynacord
10H	Oberheim		
11H	Apple Computer		GRUPO JAPONES
12H	Grey Matter Response		
14H	Palm Tree Inst.	40H	Kawai
15H	JL Cooper	41H	Roland
16H	Lowrey	42H	Korg
17H	Adams-Smith	43H	Yamaha
18H	Emu Systems	44H	Casio
19H	Harmony Systems	46H	Kamiya Studio
1AH	ART	47H	Akai
1BH	Baldwim	48H	Japan Victor
1CH	Eventide	49H	Meisosha
1DH	Inventronics	4AH	Hoshino Gakki
1FH	Clarity	4BH	Fujitsu Elect
		4CH	Sony

00H 00H 07H	Digital Music Corp.	4DH	Nisshin Onpa
00H 00H 0BH	IVL Technologies	4EH	TEAC Corp.
00H 00H 0CH	Southern Music Systems	4FH	System Product
00H 00H 0DH	Lake Butler Sound	50H	Matsushita Electric
00H 00H 10H	DOD Electronics		
00H 00H 14H	Perfect Fretworks		
00H 00H 16H	Opcode		
00H 00H 18H	Spatial Sound		
00H 00H 19H	KMX		
00H 00H 20H	Axxes		

APENDICE B





BIBLIOGRAFIA

MIDI SEQUENCING IN C
JIM CONGER
PRIMERA EDICION 1989
M&T PUBLISHING, INC.

MUSIC THROUGH MIDI
MICHAEL BOOM
EDICION 1987
MICROSOFT PRESS

KEYBOARD MAGAZINE
PUBLICADA MENSUALMENTE POR
CALTON COMMUNICATIONS COMPANY

ELECTRONIC MUSICIAN MAGAZINE
PUBLICADA MENSUALMENTE POR
ACT III PUBLISHING, INC.

MICROPROCESSORS AND INTERFACING
DOUGLAS V. HALL
TERCERA EDICION 1988
MCGRAW-HILL, INC.

THE 68000 MICROPROCESSOR
ARCHITECTURE, SOFTWARE, AND INTERFACING TECHNIQUES
WALTER A. TRIEBEL
AVTAR SINGH, PH. D.
PRENTICE-HALL

MANUALES Y ESPECIFICACIONES DE
MOTOROLA SEMICONDUCTOR PRODUCTS INC.

M68000 8-/16-/32-BIT MICROPROCESSORS
PROGRAMMER'S REFERENCE MANUAL
QUINTA EDICION
PRENTICE-HALL

PROGRAMMING THE 68000
STEVE WILLIAMS
EDICION 1988
SYBEX INC.