ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN
DEL TÍTULO DE INGENIERÍA

"DISEÑO E IMPLEMENTACIÓN DEL DISPOSITIVO DE INTERFAZ HUMANA: MIDI CONTROL SURFACE (MCS) PARA PLATAFORMAS DE PRODUCCIÓN DE AUDIO DIGITAL"

XAVIER ALEXIS VILLALBA TORRES

Sangolquí – Ecuador

2008

CERTIFICACIÓN

Certificamos que el presente proyecto de grado titulado “Diseño e Implementación del Dispositivo de Interfaz Humana: MIDI Control Surface (MCS) para Plataformas de Producción de Audio Digital”, fue realizado en su totalidad por el señor Xavier Alexis Villalba Torres, portador de la cédula de identidad No. 171352517-6 y bajo nuestra dirección.

Ing. Julio Larco DIRECTOR	Ing. Víctor Proaño CODIRECTOR

RESUMEN

El presente Proyecto trata sobre el diseño de una Superficie de Control MIDI (MCS) para plataformas de producción de audio digital usando el microcontrolador PIC16F877A. Una MCS es un dispositivo de interfaz humana expresamente diseñado para la transmisión de eventos MIDI dentro de un sistema modular MIDI. La MCS ha implementado las especificaciones que el Protocolo de comunicación MIDI establece en su edición 1.0, permitiendo así la transmisión de mensajes (tres bytes de datos en transmisión serial unidireccional) a través de un cable hacia otro dispositivo que sea capaz de recibir mensajes MIDI, como por ejemplo una DAW (Digital Audio Workstation) o un sintetizador.

La MCS consta de siete controladores (cinco continuos y dos tipo switch) que generan siete mensajes de manera consecutiva por medio de un polling para la adquisición de datos. Dispone además de un LCD de dos líneas y luz de fondo para la visualización del valor de volumen y canal de transmisión de la aplicación.

La MCS fue probada sobre la DAW llamada Reason en su tercera edición y sobre el sintetizador Yamaha modelo PSS-790 obteniendo una comunicación satisfactoria. El control táctil y en tiempo real de los controladores virtuales en la DAW hace que los objetivos planteados para este Proyecto sean cumplidos a cabalidad.

DEDICATORIA

Es un ejemplo de amor y perseverancia, de valor y decisión. A mi amiga, confidente y madre. Tus enseñanzas de vida van escondidas entre cada línea de este proyecto. A ti madre dedico este humilde trabajo que pretende reflejar todos los sacrificios que has hecho por mí. Este nuevo logro te lo debo a ti.

Xavier Villalba T.

AGRADECIMIENTO

Los grandes logros de la vida se los consigue rodeados de gente asimismo grandiosa, que con sus palabras de apoyo, su presencia y sus consejos te conducen a conseguir cuanto desees. A lo largo de mi carrera universitaria he estado rodeado de gente que agradezco a Dios haberla tenido cuando la necesitaba. Entre ellos a mis padres Fanny y Hermann, mis hermanos Ricardo y Christian y a mis familiares más allegados, Charito, mis abuelitos José y Ernestina. Son ellos quienes han tocado mi vida de manera positiva y son ellos a quienes debo lo que ahora soy.

He tenido la dicha de tener amigos y amigas que me han brindado su apoyo cada día de mi juventud. A José Luis, Santiago, Guillermo, Juan Pablo y Alfonso, a Ana María y Tania, amigos infallables y de gratos recuerdos. A Helena por su inmensa colaboración para la realización de este trabajo.

Gracias también al mentor de este proyecto, el Ing. Byron Navas, por haber confiado en mí la realización de uno de sus sueños. Espero sea de su agrado. Al Ing. Víctor Proaño y el Ing. Julio Larco por su valiosa ayuda y confianza. Al Dr. Isaac Álvarez por el apoyo brindado para la realización y culminación de este trabajo.

Xavier Villalba T.

PRÓLOGO

MIDI es un método eficiente que permite la interconexión de distintos instrumentos musicales electrónicos o computadoras trabajando en conjunto en un ambiente similar a una red. Las ventajas que esto implica hacen a este estándar un protocolo atractivo no sólo para compositores o desarrolladores sino también para aplicaciones informáticas que producen sonido, como son los juegos para PC o presentaciones multimedia. De allí que la implementación MIDI en aplicaciones musicales ha permitido que la interfaz entre el compositor o usuario y su instrumento sea cada vez más sensible y poderosa. Dichas aplicaciones requieren no sólo de un teclado y el ratón, sino también de dispositivos HID (Human Interface Devices) que permitan controlar los diferentes parámetros musicales; lo cual es indispensable en procesos de masterización, grabación, producción, post producción, etc.

Hoy en día existen un sin número de programas para PC de edición, grabación o masterización, que permiten la conexión de un dispositivo MIDI a la computadora. Lo que se consigue por medio de una Superficie de Control MIDI es la manipulación física y tangible de los diversos parámetros que involucran el protocolo MIDI cuando se trabaja sobre plataformas virtuales de audio digital.

El presente escrito contiene la información referente al diseño de una superficie de control MIDI con el microcontrolador PIC16F877A. La superficie dispone de siete controladores, un display de visualización y una salida (MIDI OUT).

INDICE DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

GLOSARIO

AES, Audio Engineering Society.

Aftertouch, presión ejercida sobre una tecla después de ser tocada.

ASIO, Audio Stream In/Out, protocolo de transferencia multicanal para audio digital y MIDI creado por Steinberg (Cubase).

Balance, variación de los niveles relativos entre dos fuentes independientes de sonido.

Channel Message, afectan solamente a uno de los 16 canales MIDI y sólo responden a este tipo de mensajes los instrumentos sintonizados para recibir en ese canal.

Channel Mode Message, determina la forma en que el dispositivo responde a los mensajes recibidos.

Channel Voice Messages, tienen que ver con la producción de sonido (como por ejemplo prender y apagar notas).

Controlador, sección sobre la que actúa el usuario en el momento de tocar y que envía la información al generador.

Data Byte, codifica el valor numérico verdadero al cual está ligado un status byte.

DAW, Digital Audio Workstation.

Daisy Chain, conexión típica de varios dispositivos MIDI en serie, siendo una de las más simples y más comúnmente usadas.

EOX, End Of Exclusive.

Fader, Potenciómetro de deslizamiento lineal.

Generador, sección de un dispositivo que produce el sonido propiamente.

Header, encabezamiento***.***

HID, Human Interface Device.

IMA, International MIDI Asociation.

JMSC, Japanese MIDI Standard Comitee.

Keyboard, teclado o sintetizador.

Keypad, teclado numérico.

Knob, Potenciómetro de deslizamiento circular.

LFO, Low Frecuency Oscillation.

MCS, MIDI Control Surface.

MIDI, Musical Instrument Digital Interface.

MiniMoog, sintetizador más clásico de la historia, con su aspecto amigable y su famoso sonido cálido, creado por Robert Moog.

MMA, MIDI Manufacturers Asociation.

MME, Multimedia Extensión, similar a ASIO, creado por Microsoft.

Mono, abreviatura de Monofónico. Modo en el cual el instrumento es capaz de responder monofónicamente a cada canal MIDI y de producir solamente una nota a la vez.

MSB, More Significant Bit.

MTC, MIDI Time Code*.*

Música electrónica, composición de melodías realizadas por medio de instrumentos electrónicos, esto es, dispositivos que generan señales audibles a través de fuentes eléctricas.

Nibble, Un byte está formado por dos nibbles, uno para los 4 bits más significativos y otro para los 4 menos significativos.

Octava, intervalo que separa dos sonidos cuyas frecuencias fundamentales tienen una relación de dos a uno.

Omni, se refiere a la manera en que un instrumento MIDI responderá a los 16 canales. Cuando este modo está activado el dispositivo actuará sobre todos los canales.

Panning, balance relativo de una fuente de sonido simple entre los canales derecho e izquierdo de un campo de sonido estéreo.

Patch, parche, banco entero de programas MIDI.

Pitch Bending, también conocido como Portamento. Es el efecto de deslizamiento continuo elevando o bajando la frecuencia de una nota previa.

Polling, Término usado para describir el flujo de programa a manera de bucle y que se repite indefinidamente.

Poly, abreviatura de Polifónico. Modo en el cual el instrumento es capaz de responder polifónicamente a cada canal MIDI y de producir más de una nota a la vez.

Scratching, técnica utilizada por DJ's consistente en mover un disco de vinilo hacia adelante y hacia atrás sobre el plato del tocadiscos para crear un efecto parecido al de rayar el disco.

SDS, Sample Dump Standard.

Secuenciador, dispositivo digital que almacena, edita y reproduce datos MIDI.

Semitono, intervalo musical equivalente a la distancia entre dos teclas adyacentes de un instrumento de teclado.

Sintetizador**,** instrumento musical electrónico que hace uso de múltiples generadores de sonido para crear formas de onda complejas que, cuando son combinadas, "sintetizan" una característica de sonido única.

SMPTE, Society of Motion Picture and Television Engineers.

SPP, Song Position Pointer.

Status Byte, identificador para decirle al dispositivo receptor a qué función y canal MIDI en particular son direccionados.

SysEx, System Exclusive Messages.

System Common Messages, mensajes que afectan a todos los instrumentos conectados al sistema.

System Exclusive Messages, estos mensajes comienzan con un encabezamiento que identifica la marca y modelo determinado del instrumento al cual están dirigidos, siendo ignorados por todos los demás dispositivos.

System Message, son recibidos por todos los instrumentos conectados a la red.

System Real Time Messages, mensajes que están relacionados con el funcionamiento de secuenciadores, su temporización y sincronización.

Timbre, sonido característico de una voz o instrumento.

UART, Universal Asynchronous Receiver Transmitter.

USI, Universal Synthetizer Interface. Primer protocolo de instrumentos musicales electrónicos.

VC, Voltaje de Control.

Velocidad, indica la velocidad con la que fue presionada la tecla o controlador.

Vibrato, efecto musical donde la frecuencia de la nota o sonido es elevada o bajada rápida y repetidamente sobre una pequeña porción de la duración de la nota o sonido.

Volumen, percepción subjetiva de la potencia de un determinado sonido.

INTRODUCCIÓN

MÚSICA ELECTRÓNICA

Se puede entender como Música Electrónica a la composición de melodías realizadas por medio de instrumentos electrónicos, esto es, dispositivos que generan señales audibles a través de fuentes eléctricas. En la actualidad esta definición abarca la producción de música generada por medio de sintetizadores y computadoras.

Podríamos situar el punto de partida de la Electrónica aplicada a la Música a finales del siglo XIX donde surgieron los primeros instrumentos parcialmente eléctricos. Científicos como Helmholtz, Cahill, Severy, Duddell y Gray, fueron los pioneros en la creación de estos dispositivos, en algunos casos, casi accidental y sin mayor gloria. Utilizaron una variedad de técnicas para generar sonido, tal como la rueda del tono, usada en el *Telharmonium* de Cahill (Figura 1) y el *Choralcello* de Severy. Éste era un disco de metal que rotaba en un campo magnético causando variaciones en una señal eléctrica. Duddell encontró que variando el voltaje provisto a las lámparas con arco de carbón se podían crear frecuencias audibles y controlables; lo llamó *El Arco del Cantar*. Elisha Gray creó el Telégrafo Musical, un efecto de la tecnología del teléfono.

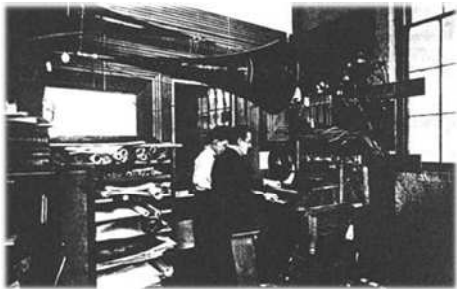


Figura 1. Telharmonium.

Para los años veinte y con el surgimiento de la tecnología de tubos de vacío, la creación de instrumentos musicales electrónicos alcanzó nuevas fronteras. Instrumentos como el *Theremin* (1917), el *Sphäraphon* (1921), el *Pianorad* (1926) y el *Ondes Martenot* (1928) (Figura 2) explotaron esta tecnología. El auge de estos instrumentos fue grandioso, mas no así la acogida de un público que seguía aferrado a los conceptos musicales tradicionales. El progreso técnico logrado estaba demasiado por delante del cultural.

Figura 2. El Ondes-Martenot

El inicio de la Segunda Guerra Mundial alejó el interés de los científicos hacia otras investigaciones tecnológicas, estancando así el surgimiento de nuevos lenguajes musicales. Durante los años cuarenta y cincuenta tan sólo unos pocos investigadores persistieron en la exploración del universo musical a través de la manipulación de cintas magnéticas, generalmente desde un terreno puramente experimental. Surgieron así los primeros prototipos de sintetizadores. Eran máquinas mucho más evolucionadas que cualquiera de los instrumentos no acústicos previos, pero resultaban muy difíciles de manejar y mantener. Pero al mismo tiempo, se estaba desarrollando un cambio importantísimo en materia de

tecnología musical: el técnico y el músico empezaban a colaborar en conjunto en la producción musical. Bandas como Pink Floyd, Tangerine Dream, Ash Ra Tempel, Kraftwerk, Cluster, Can y otras, empezaron a experimentar con esta nueva tecnología creando ideas musicales vanguardistas.

A medida que germinaba esta nueva música también lo hacía el medio tecnológico. Es así como a principios de los años 60 el nacimiento del transistor y los circuitos integrados, permitieron el progreso artístico de estas tendencias. En Estados Unidos, *Robert Moog*, ingeniero eléctrico con formación musical conoció al compositor y profesor de música Herbert Deutsch. De las necesidades técnicas del músico y la afición musical del técnico, nació del genio de Moog el primer sintetizador analógico controlado por tensión (Figura 3). “Desde 1963, cuando los Beatles utilizaron un sintetizador Moog en su álbum Abbey Road, hasta hoy, con Fat Boy Slim como fan de los Moog, el mundo de la música ha progresado gracias al invento de este físico e ingeniero eléctrico”[1]. La música elaborada exclusivamente con sintetizadores y otros instrumentos musicales electrónicos no tarda en despertar el entusiasmo de las masas. De esta manera se dio inicio a la carrera comercial en la producción de sintetizadores debido a la creciente demanda de músicos interesados en incluir dichos instrumentos en sus composiciones. Nuevas firmas comerciales como la EMS (Electronic Music Studios) con su sintetizador VSC3 apostaron por la producción en masa de estos dispositivos. Pero es sin duda el *MiniMoog* el sintetizador más clásico de la historia, con su aspecto amigable y su famoso sonido cálido.

Figura 3. El MiniMoog, el primer sintetizador analógico controlado por tensión.

[1] Tomado de la Revista Rolling Stones para Sudamérica, Número 22, pág. 16.

Desde entonces el desarrollo de sintetizadores ha ido evolucionado según los nuevos avances tecnológicos, llegando así a la generación de los sintetizadores digitales, es decir, controlados por software. Es un procesador digital el que se encarga de generar y manipular el sonido. Ahora podemos tener en un solo sintetizador desde un piano convencional hasta una orquesta completa de instrumentos musicales. Es por ello que hoy en día se puede decir que es posible crear cualquier clase de música electrónicamente y que todo compositor profesional o casero ha experimentado al menos una vez el uso de computadoras en sus trabajos, sean éstos de producción o edición musical.

Figura 4. El Yamaha DX7 (1983), el primer sintetizador totalmente digital y que incorporó puertos MIDI.

Es así como la Música Electrónica ha ido evolucionando a través de la historia siempre a la par de los avances tecnológicos y concebida por las mentes brillantes de hombres que supieron apostarle a la creación de nuevas formas de expresar belleza fusionando el arte con el incansable mundo de la tecnología.

MCS (MIDI COLTROL SURFACE)

Definición

Por sus siglas en inglés MCS (*MIDI Control Surface*) o Superficie de Control MIDI, es un Dispositivo de Interfaz Humana HID[2], que permite la manipulación externa al PC de parámetros MIDI a través de controles físicos.

[2] HID, Siglas en Inglés para Human Interface Device.

Figura 5. Producción y edición de audio a través de MCS.

Origen e Importancia

Estos dispositivos surgieron por la necesidad del usuario de disponer de un mecanismo que permitiera un control externo de las funciones que un programa de aplicación musical puede disponer (volumen, panning, efectos, etc.). Así, las Superficies de Control MIDI, permiten hoy en día que el control de varios dispositivos virtuales (faders, knobs, pushbuttons, etc.) se lo ejecute en tiempo real y de forma simultánea; ventajas significativas cuando se trata de procesos de automatización, mezcla, creación de efectos y más. Qué importante es para un compositor o Ingeniero en Sonido sentir a través de un control físico la manipulación de valores como el volumen o reverberación de su composición en tiempo real. Es por ello que las MCS son ampliamente utilizadas no sólo en estudios de grabación por profesionales sino también por usuarios caseros quienes no requieren de sofisticados equipos de edición musical.

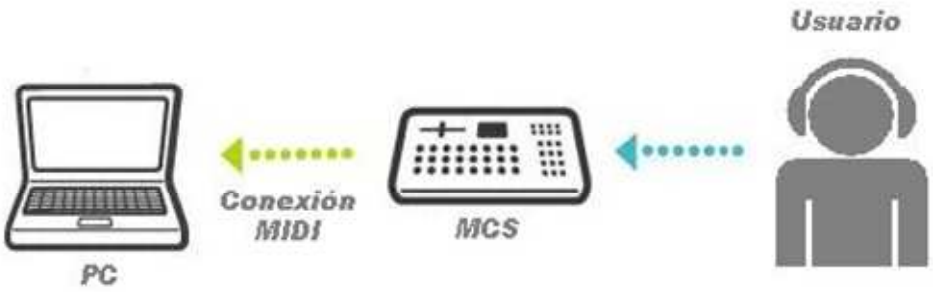


Figura 6. Esquema general de conexión de una MSC.

Cómo funciona

Una MCS utiliza un procesador integrado para generar el código necesario (Mensajes MIDI[3]) que un programa de aplicación musical requiere cuando el usuario manipula físicamente un control dedicado a un parámetro MIDI[4]. Estos controles pueden ser del tipo continuo (rueda de modulación, perillas, etc.) y del tipo interruptor (pedales de sostenimiento, push buttons, etc.). Más adelante se verá con mayor detalle estos controladores.

El software ligado a la superficie de control permite la asignación de los parámetros MIDI a los controles de la MCS. Esto lo consigue mediante el enrutamiento de los controles virtuales hacia los controles físicos de la superficie por medio del mismo software. El código generado por el procesador de la MCS es enviado según las especificaciones del Protocolo MIDI a través de un cable hacia la PC donde el software de aplicación se encarga de transformarlo y ejecutar el mismo comando que el usuario realizó. De allí que todas las plataformas de aplicación musical sin excepción hacen uso de las superficies de sontrol, debido a la gran versatilidad que éstas proveen en tareas de producción y edición musical.

[3] Para mayor detalle sobre Mensajes MIDI ver Capítulo 2.
[4] Para mayor detalle sobre Parámetros MIDI ver Capítulo 1.

Figura 7. Superficie de Control MIDI marca Behringer modelo BCF2000.

Debido a que las MCS son diseñadas bajo el mismo lenguaje de comunicación (protocolo MIDI) éstas pueden trabajar con cualquier otro dispositivo que sea capaz de recibir y transmitir mensajes MIIDI. Los dispositivos más comúnmente utilizados con MCS son los sintetizadores y estaciones de audio digital o DAW (*Digital Audio Workstations*) por sus siglas en Inglés.

CAPÍTULO 1

1. MIDI, GENERALIDADES

1.1. ORÍGENES

Debido al incesante progreso en la tecnología electrónica, esto es, el aumento en la velocidad y en la capacidad de memoria de los microprocesadores, el desarrollo de instrumentos musicales ha ido evolucionando constantemente.

En sus primeros días los sintetizadores fueron creados como dispositivos monofónicos, es decir, eran capaces de generar una sola nota a la vez, limitando así la calidad del sonido. Por tal motivo, los fabricantes buscaron la manera de comunicar más de un instrumento entre si para crear un sonido más rico en calidad y textura.

Se creó entonces la primera “red” de instrumentos musicales electrónicos donde un sintetizador actuando como dispositivo maestro, controlaba directamente el desempeño de uno o más sintetizadores trabajando como esclavos a través de una señal de control básica conocida como voltaje de control VC y una señal adicional de compuerta (Figura 1.1). Esta segunda señal era usada para sincronizar los tiempos de inicio y duración cuando una nota era tocada.

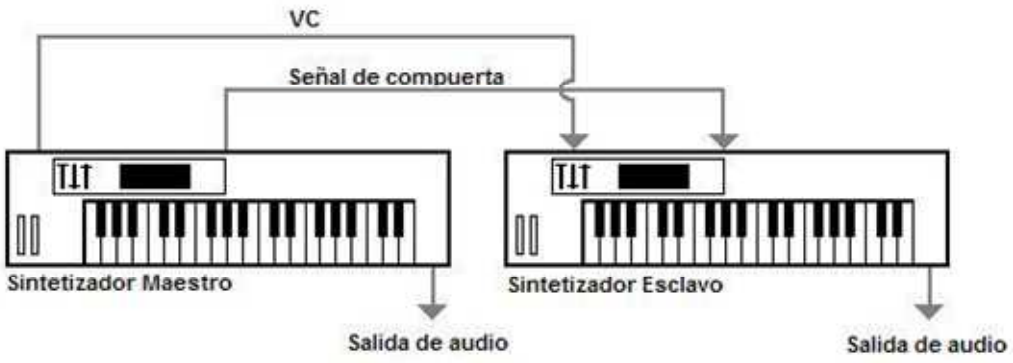


Figura 1.1. Red de instrumentos usando una señal VC y Compuerta.

Con la aparición de los primeros sintetizadores polifónicos (capaces de generar varios sonidos a la vez) surgieron nuevas inquietudes y problemas. Estaba claro que el sistema anterior ya no era capaz de controlar la comunicación entre sintetizadores y nuevos estándares empezaron a aparecer pero ninguno fue realmente universal. Cada fabricante trabajaba con un reloj distinto y la sincronización entre dispositivos pertenecientes a diferentes casas era sólo posible a través de dispositivos conversores u otro tipo de modificaciones. Este tipo de incompatibilidades condujeron a la creación del primer protocolo de instrumentos musicales electrónicos conocido como USI (*Universal Synthetizer Interface*) por parte de una de las primeras compañías fabricantes de instrumentos electrónicos, *Sequential Circuits*.

Más tarde, en 1981 a la *Audio Engineering Society* (AES) de Estados Unidos, con representantes de los mayores fabricantes de sintetizadores del mercado, se le asignó la tarea de crear un estándar universal basado en el protocolo USI.

Así, en 1983, fue presentado el Protocolo denominado MIDI (*Musical Instrument Digital Interface*). Actualmente asociaciones como la MMA (*MIDI Manufacturers Asociation*) de Estados Unidos y la IMA (*International MIDI Asociation*), son las organizaciones encargadas de la coordinación y regulación para el desarrollo y ampliación de la norma. Hoy por hoy se puede decir que todos

los instrumentos electrónicos de aplicación musical con PC[1] sin excepción incluyen una completa implementación MIDI.

1.2. GENERADORES Y CONTROLADORES

Cuando hablamos de instrumentos musicales electrónicos debemos diferenciar dos partes fundamentales que involucran los mismos. Estas son el *generador* y el *controlador*. El generador es la sección del dispositivo que produce el sonido propiamente y que puede ser programada asignándole valores determinados a sus diferentes parámetros, consiguiendo así variar las características tímbricas[2] del sonido generado. El controlador por su parte es la sección sobre la que actúa el usuario en el momento de tocar y que envía la información al generador, diciéndole por ejemplo que debe prender o apagar una determinada nota, aplicar cierto efecto, etc. El ejemplo más común de controlador son los teclados o *keyboards* del tipo piano (Figura 1.2) pero en la actualidad existen muchas otras clases. También integran el controlador otros dispositivos como pueden ser ruedas, pedales, perillas o *knobs*, que agregan expresión adicional en el momento de tocar, ya que, según cómo haya sido programado el generador, accionarlos producirá diferentes variaciones en el resultado sonoro.

Figura 1.2. Teclado Controlador MIDI UMX25 marca Behringer.

[1] PC, siglas en inglés para Personal Computer.
[2] Timbre, sonido característico de una voz o instrumento.

Existen tres tipos de controladores de tiempo real:

- **Controladores Continuos**. Permiten un barrido completo del rango de posibles valores de control. Comúnmente este rango va desde 0 hasta 127 valores, sin embargo, como se verá en el capítulo siguiente, es posible mejorar esta resolución.

- **Controladores tipo Switch.** Sólo poseen dos posibles posiciones: On y Off, sin valores intermedios.

- **Controladores de Datos.** Permiten ingresar datos a través de un teclado numérico (keypad) o por medio de botones de ingreso de datos ascendentes y descendentes.

Los controladores más estandarizados actualmente son los siguientes:

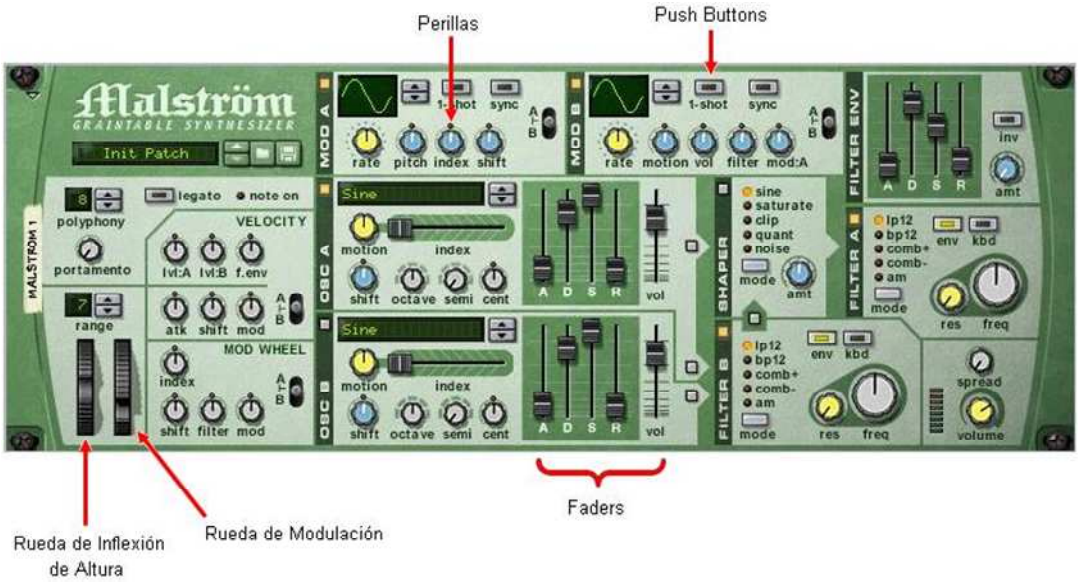


Figura 1.3. Controladores en el sintetizador Malstrom de Reason 3.

1.2.1. Rueda de Modulación (Modulation Wheel)

Se posiciona libremente entre mínimo y máximo y se usa típicamente para abrir o cerrar filtros, o controlar la amplitud y/o frecuencia de filtros de modulación (LFO[3]) para conseguir efectos de vibrato[4].

1.2.2. Rueda de inflexión de altura (Pitch-Bender)

A diferencia del controlador anterior el rango de valores no va de mínimo a máximo. El cero se encuentra en la posición central y adquiere valores positivos o negativos al desplazarlo hacia arriba o abajo, respectivamente. Tampoco se posiciona libremente ya que un resorte la mantiene en la posición central mientras no esté siendo accionada. Su función básica es efectuar inflexiones de altura (*pitch-bending*) ascendentes o descendentes.

1.2.3. Faders

Son potenciómetros de desplazamiento lineal. Son utilizados principalmente para el control de volumen de un canal o master. Se posiciona libremente entre mínimo y máximo, generalmente de forma vertical, donde el valor cero o mínimo se encuentra en el extremo inferior y el máximo en el extremo superior.

1.2.4. Perillas (*Knobs*)

Potenciómetros de desplazamiento circular que son ampliamente usados en todos los tipos de instrumentos musicales y dispositivos de control. Las

[3] LFO, siglas en inglés para Low Frecuency Oscillation.
[4] Vibrato, efecto musical donde la frecuencia de la nota o sonido es elevada o bajada rápida y repetidamente sobre una pequeña porción de la duración de la nota o sonido.

aplicaciones más comunes son el control del balance stereo (*Panning*[5]), control de bajos y altos, y control de ganancia de entradas auxiliares.

1.2.5. Botones on-off (*Push-buttons*)

Son pulsadores generalmente utilizados para las funciones de *Mute* (silenciar) y *Solo* de un canal o para activar y desactivar ecualizadores, filtros o efectos.

1.2.6. Pedal de Sostenimiento (Sustain Pedal)

Su efecto es similar al del pedal derecho del piano, esto es, mientras está oprimido no envía mensajes de apagado de las notas. Si bien es de movimiento continuo sólo adquiere dos valores: on y off.

1.2.7. Pedal de Volumen

Es de movimiento continuo y duplica el potenciómetro de volumen del instrumento. Es útil para controlar el volumen mientras se usan ambas manos para tocar.

Los generadores de los primeros modelos de sintetizador (constituidos básicamente por osciladores, filtros y amplificadores) estaban *controlados por voltaje*. Así, al accionar una tecla (controlador), ésta enviaba una corriente cuyo voltaje determinaba qué nota produciría el oscilador. Estos controladores eran muy simples pues sólo podían tocar una nota a la vez y no enviaba información adicional como la velocidad y presión con que era presionada y liberada.

[5] Panning, balance relativo de una fuente de sonido simple entre los canales derecho e izquierdo de un campo de sonido estéreo.

Para la década de los 80, la incorporación de tecnología digital en los instrumentos musicales electrónicos permitió que todas las funciones las realice un microprocesador a través de operaciones matemáticas. Ahora el controlador es capaz de enviar información completa en forma de número al generador, el cual según se haya sido programado interpretará dicho valor.

Esta forma de representar los valores del controlador posee grandes ventajas:

- Permite una mayor precisión en la programación.
- Exacta reproducibilidad, y,
- Permite el almacenamiento de bancos enteros de programas o *patches*[6].

Tanto controladores como *secuenciadores*[7] han sido beneficiados con la introducción de la tecnología digital. Los controladores ahora son más sensibles y sofisticados ya que envían información adicional como es la velocidad y el *aftertouch*[8]. Según las características del instrumento, el generador puede responder a estos matices de ejecución realizando desde pequeñas variaciones hasta drásticos cambios tímbricos.

1.3. PARÁMETROS GENERALES DE AUDIO RELACIONADOS CON MIDI

Como ya se ha visto anteriormente, MIDI es un lenguaje que permite la transmisión de información musical. Existen varios parámetros de audio que fueron añadidos al protocolo para que la producción de sonido sea de mayor calidad y textura. El conjunto de todos estos parámetros se denomina *Patch* y es el que determina el tipo de sonido resultante. Un patch puede estar programado de tal manera que pretenda reproducir el timbre de un instrumento acústico o por el contrario producir un timbre totalmente sintético.

---

[6] Patches, del inglés *patch* que significa parche.
[7] Secuenciador, dispositivo digital que almacena, edita y reproduce datos MIDI.
[8] Aftertouch, o presión ejercida sobre una tecla después de ser tocada.

A continuación en la Tabla 1.1 se presentan los parámetros de audio relacionados con MIDI:

Tabla 1.1. Parámetros de audio relacionados con MIDI

Parámetro	Descripción
Nota	Sus valores indican qué nota se tocó, en qué instante, en qué canal, con qué fuerza o volumen y durante cuánto tiempo se mantuvo presionada.
Velocidad	Indica la velocidad con la que fue presionada la tecla o controlador.
Aftertouch	Registra las variaciones de presión que se ejerce sobre una tecla luego de tocarla.
Balance	Variación de los niveles relativos entre dos fuentes independientes de sonido.
Panning	Es usado para posicionar el balance relativo de una fuente de sonido simple entre los canales derecho e izquierdo de un campo de sonido estéreo.
Pitch Bending	También conocido como **Portamento**. Es el efecto de deslizamiento continuo elevando o bajando la frecuencia de una nota previa.
Volumen	Es la percepción subjetiva de la potencia de un determinado sonido.
Mono	Abreviatura de Monofónico. Modo en el cual el instrumento es capaz de responder monofónicamente a cada canal MIDI y de producir solamente una nota a la vez.
Poly	Abreviatura de Polifónico. Modo en el cual el instrumento es capaz de responder polifónicamente a cada canal MIDI y de producir más de una nota a la vez.
Omni	Se refiere a la manera en que un instrumento MIDI responderá a los 16 canales. Cuando este modo está activado el dispositivo actuará sobre todos los canales.

1.4. VENTAJAS Y DESVENTAJAS DE MIDI

1.4.1. Ventajas:

- Prácticamente universalizado para toda clase de instrumentos musicales, lo cual permite la fácil comunicación y sincronización entre dispositivos.

- La velocidad de transmisión del protocolo hace posible que la producción de audio sea en tiempo real.
- Debido a que MIDI es un protocolo digital es muy sencilla la conexión entre instrumentos electrónicos y computadoras para post producción y edición por medio de software.
- Una composición musical creada a través de MIDI permite una reproducción infinita de veces de la misma cuando ésta es almacenada en disco.
- La transmisión de información musical a través de Mensajes MIDI permite que la programación para aplicaciones sea sencilla y que los datos sean fácilmente manipulables.
- La fácil conexión entre dispositivos MIDI ofrece la posibilidad de tener todo un estudio de producción y edición de audio en espacios reducidos.
- Los 16 canales polifónicos y expandibles permiten la inserción de varios instrumentos para conformar una orquesta completa (Ver Figura 1.4).
- No produce errores musicales de notas incorrectas o instrumentos desafinados. Se conoce a todo instante que nota se está tocando.
- El protocolo permite modificar las características tímbricas de un sonido para "asemejarlas" a las de un instrumento acústico o para crear sonidos sintéticos.
- Los archivos MIDI ocupan poca memoria en comparación con otros formatos de audio.

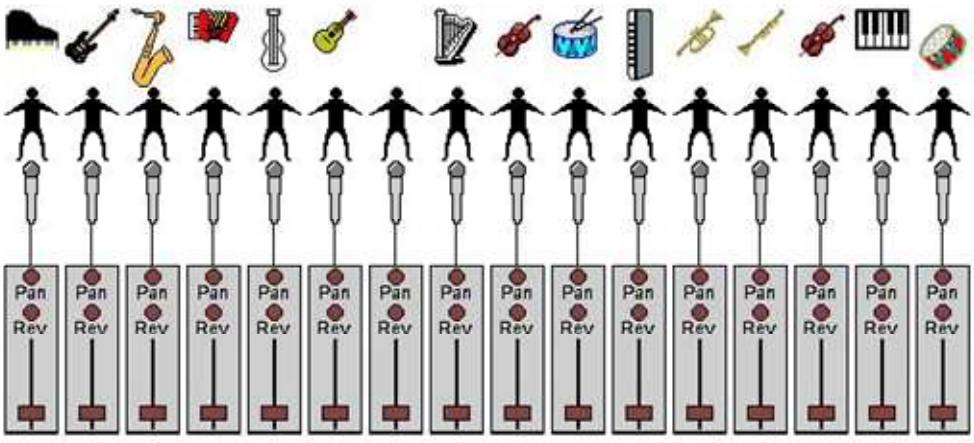


Figura 1.4. Los 16 canales MIDI representando una orquesta completa de instrumentos.

1.4.2. Desventajas:

La principal desventaja es que por más sofisticado que sea el dispositivo MIDI, el sonido que produce nunca será igual al producido por un instrumento acústico debido a su infinidad de matices y variaciones dinámicas.

1.5. APLICACIONES

La versatilidad que proporciona el protocolo MIDI permite una infinidad de aplicaciones pero existen tres tipos fundamentales de dispositivos:

1.5.1. Controladores

Como ya se ha visto anteriormente estos dispositivos transforman la acción física de presionar una tecla o control en mensajes MIDI. Estos son interpretados por una computadora la cual ejecuta la misma acción por medio de software.

El ejemplo más importante entre estos dispositivos son las *Superficies de Control* las cuales son ampliamente utilizadas a nivel mundial por usuarios caseros y profesionales a través de una gran variedad de plataformas de edición musical (Ver Figura 1.5).

Entre otras aplicaciones tenemos máquinas de ritmo, instrumentos musicales MIDI, superficies de control de iluminación, etc.

Figura 1.5. MCS de Percusión modelo Trigger Finger de M-Audio.

1.5.2. Secuenciadores

Son dispositivos que almacenan digitalmente información MIDI en memoria para su edición y reproducción en forma secuencial (Ver ejemplo Figura 1.6). Una vez que una composición ha sido grabada por un secuenciador (en forma de cadenas secuenciales de mensajes MIDI), esos eventos son entonces almacenados dentro de la memoria RAM interna del dispositivo donde pueden ser editados y, si el dispositivo lo permite, grabados en un disco para propósitos de archivo. Cuando se vuelve a tocar el archivo almacenado, el secuenciador reproduce esos mensajes en el mismo orden en el que fueron grabados originalmente.

Figura 1.6. Secuenciador MIDI marca KAWAI modelo ACR20.

1.5.3. Sintetizadores

Constituyen la primera aplicación musical del protocolo MIDI. Son instrumentos musicales electrónicos los cuales hacen uso de múltiples generadores de sonido para crear formas de onda complejas que, cuando son combinadas, “sintetizan” una característica de sonido única (Ver ejemplo Figura 1.7). Cada generador de tonos puede ser controlado con respecto a la frecuencia, amplitud, timbre y envoltura. Los sintetizadores modernos proveen un control digital sobre estos parámetros analógicos o más comúnmente, generar esas formas de onda directamente de forma digital. Ambos métodos permiten al usuario almacenar los valores de los parámetros de control para generar esos

sonidos posteriormente dentro de una memoria interna o externa a manera de un *patch*.

Figura 1.7. Sintetizador Casio CZ101, entre los primeros teclados comercialmente disponibles de MIDI

CAPÍTULO 2

2. EL PROTOCOLO MIDI

2.1. INTRODUCCIÓN

La Interfaz Digital de Instrumentos Musicales (MIDI) es un protocolo de comunicación digital que por medio de un lenguaje de control y especificaciones de hardware estandarizados permiten la comunicación y el control de datos en tiempo real de múltiples instrumentos o dispositivos musicales electrónicos.

MIDI es un formato de datos específico que debe ser estrictamente añadido por quienes diseñan y construyen instrumentos y dispositivos equipados con MIDI. De esta manera, las tareas relacionadas al desempeño y automatización de mensajes pueden ser comunicadas entre dispositivos con relativa transparencia, velocidad y facilidad. Así, cuando se realiza una tarea (como controlar múltiples instrumentos desde un teclado controlador o transmitir un *patch* de una librería hacia un sintetizador), el usuario necesita considerar solamente los parámetros de control de los dispositivos involucrados y no los parámetros del medio de transmisión mismo.

En nuestro propio lenguaje (el Español, Inglés, etc.) para poder establecer una comunicación entre personas se desarrolló desde sus inicios un determinado orden de las palabras y las letras que las conforman. Así mismo MIDI requiere un orden de sus “palabras” (*bytes*) para transmitir un mensaje. Microprocesadores y computadoras al ser dispositivos digitales de comunicación poseen la ventaja de

ser capaces de procesar números a muy altas velocidades[1]. A diferencia de nuestro sistema de numeración decimal (10 dígitos), las computadoras están limitadas al sistema de numeración binario con sólo dos dígitos: 0 y 1. Pero, así como los humanos, las computadoras poseen la capacidad de agrupar esos dígitos binarios (*bits*) para formar grandes valores numéricos a manera de palabras digitales. Palabras que pueden ser usadas para representar y comunicar información e instrucciones específicas. Este conjunto ordenado de palabras se conoce con el nombre de Mensajes MIDI.

2.2. EL MENSAJE MIDI

La información de desarrollo musical es comunicada digitalmente durante toda la producción como una cadena de mensajes MIDI la cual es transmitida a través de una línea simple MIDI. Estos datos pueden viajar solamente en una dirección desde una fuente simple a un destino. Para que la comunicación sea posible en ambos sentidos, una segunda línea de transmisión debe ser conectada desde la segunda fuente de regreso al dispositivo maestro original (Figura 2.1).

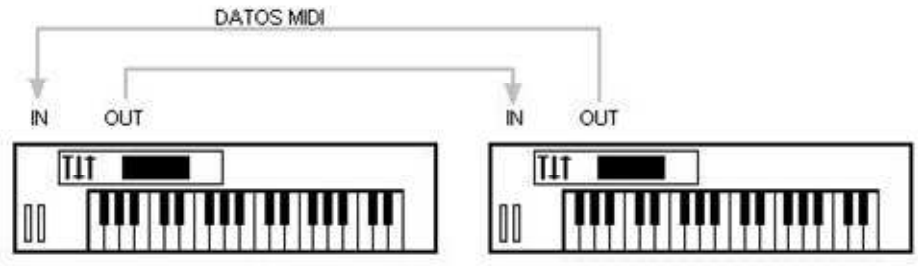


Figura 2.1. Los datos MIDI pueden viajar en una dirección a través de un cable simple.

2.2.1. Tipos de Byte

Los mensajes MIDI están formados por grupos de bytes. Existen solamente dos tipos de bytes definidos dentro de las especificaciones del protocolo. Estos

[1] La velocidad a la cual los datos pueden ser comunicados es medida en baudios (número de palabras de 8 bits que pueden ser transmitidos o recibidos por segundo).

son: de estado (status byte) y de datos (data byte), que están identificados por su MSB[2].

2.2.1.1. Status Byte.

Los bytes de estado son usados dentro del mensaje MIDI como un identificador para decirle al dispositivo receptor a qué función y canal MIDI en particular son direccionados. El status byte determina de qué tipo de mensaje se trata, por ejemplo prender o apagar una nota en un determinado canal, iniciar o detener el secuenciador, seleccionar un *patch*, etc. El MSB del status byte es siempre 1 y toma valores desde el $1000\ 0000_2$ hasta el $1111\ 1111_2$.

Tabla 2.1. Rango de valores del Status Byte

	Status Byte	
Sistema	**Desde**	**Hasta**
Binario	1000 0000	1111 1111
Decimal	128	255
Hexadecimal	80	FF

2.2.1.2. Data Byte.

Los bytes de datos son usados para codificar el valor numérico verdadero al cual está ligado un status byte. Según cuál sea el byte de estado puede estar seguido de uno o más bytes de datos con los valores correspondientes al tipo de mensaje. Por ejemplo, el status byte para prender una nota en cierto canal, debe estar seguido por dos data bytes indicando qué nota hay que prender y con qué velocidad, respectivamente.

[2] MSB, siglas en inglés para More Significant Bit.

Tabla 2.2. Ejemplo de mensaje MIDI para encender la nota E0 en el canal 5

	Status Byte		Data Byte 1	Data Byte 2
Descripción	*Status*	*Canal #*	*Nota #*	*Velocidad*
Dato binario	**1001**	**0101**	**0001 0000**	**0001 0100**
Valor	Nota on	Canal 5	16 (E0)	20

El MSB de los data bytes es siempre 0 (cero) y toma valores desde 0000 0000_2 a 0111 1111_2. Es por esto que varios parámetros como la velocidad, modulación, volumen, etc. tienen 128 valores posibles. También es la cantidad total de notas especificadas en MIDI: diez octavas[3] y 8 semitonos[4], de C1 a G9, siendo A4 la nota LA central de 440Hz. Otros parámetros, como el *pitch bending* y el posicionador dentro de una canción, precisan mayor definición; en ese caso son necesarios dos bytes consecutivos de datos, lo que da una definición de 14 bits (16.384 valores).

Tabla 2.3. Rango de valores del Data Byte

	Data Byte	
Sistema	**Desde**	**Hasta**
Binario	0000 0000	0111 1111
Decimal	0	127
Hexadecimal	00	7F

Cuando hay una sucesión de mensajes del mismo tipo no es necesario repetir todas las veces el status byte. Solamente se envía el primero de ellos y luego los bytes de datos respectivos. Si por ejemplo se trata de prender una serie de notas en un canal, se envía el status byte correspondiente y luego los dos bytes de datos para cada una de las notas. Esta sucesión de mensajes de datos se la conoce con el nombre de *Running Status*.

[3] Octava, intervalo que separa dos sonidos cuyas frecuencias fundamentales tienen una relación de dos a uno.
[4] Semitono, intervalo musical equivalente a la distancia entre dos teclas adyacentes de un instrumento de teclado.

2.2.2. Canales MIDI

La línea MIDI transmite información simultáneamente en varios canales especificados en el segundo *nibble*[5] del status byte, es decir que se puede transmitir mensajes MIDI hasta en un máximo de 16 canales (Tabla 2.4. Canales MIDI). Si se están controlando varios instrumentos a la vez, cada uno de ellos puede ser programado para responder a un canal específico, de modo que queda conformada una "orquesta" en la que se pueden tocar notas independientemente en cada instrumento. El dispositivo MIDI solo responderá a ese canal e ignorará los mensajes hacia otros canales. La mayoría de los instrumentos musicales digitales actualmente tienen capacidad multitímbrica, es decir que funcionan como varios instrumentos virtuales en uno: reciben en más de un canal simultáneamente (usualmente 8 o los 16), pudiendo asignar un *patch* diferente a cada canal. Algunos mensajes se dirigen a un canal específico; en ellos el primer *nibble* del status byte indica de qué comando se trata y el segundo *nibble* a qué canal se aplica.

Tabla 2.4. Canales MIDI

Canales MIDI		
Binario	**Decimal**	**Canal #**
0000	0	1
0001	1	2
0010	2	3
0011	3	4
0100	4	5
0101	5	6
0110	6	7
0111	7	8
1000	8	9
1001	9	10
1010	10	11
1011	11	12
1100	12	13
1101	13	14
1110	14	15
1111	15	16

[5] Un byte está formado por dos nibbles, uno para los 4 bits más significativos y otro para los menos significativos.

2.2.3. Tipos de Mensajes MIDI

Existen dos tipos de mensaje MIDI:

- **Mensajes de Canal** (*Channel Message*), y
- **Mensajes de Sistema** (*System Message*).

Los Mensajes de Canal afectan solamente a uno de los 16 canales MIDI y sólo responden a este tipo de mensajes los instrumentos sintonizados para recibir en ese canal. Los mensajes de canal se subdividen a su vez en:

- *Channel Voice*, tienen que ver con la producción de sonido (como por ejemplo prender y apagar notas).
- *Channel Mode*, determina la forma en que el dispositivo responde a los mensajes recibidos.

Los Mensajes de Sistema son recibidos por todos los instrumentos conectados a la red. Hay tres tipos de mensajes de sistema:

- *System Exclusive*, estos mensajes comienzan con un encabezamiento que identifica la marca y modelo determinado del instrumento al cual están dirigidos, siendo ignorados por todos los demás dispositivos.
- *System Common*, afectan a todos los instrumentos conectados al sistema.
- *System Real Time*, están relacionados con el funcionamiento de secuenciadores, su temporización y sincronización.

Más adelante se hablará en detalle acerca de cada uno de los tipos de mensajes MIDI.

2.2.4. Modos MIDI

Los instrumentos electrónicos frecuentemente varían en el número de sonidos que pueden producir al mismo tiempo usando su generador de sonido

interno. Pueden también variar en el número de sonidos característicos individuales que pueden ser simultáneamente producidos por un instrumento. Ciertos instrumentos son solamente capaces de producir una nota a la vez, mientras otros (conocidos como polifónicos) son capaces de generar varias notas a la vez. Estos últimos permiten al artista tocar acordes y más de una línea musical sobre un mismo instrumento. Además, para varios tipos de sintetizadores es posible producir solamente un *patch* de sonido característico a la vez. Por otra parte, un instrumento puede también ser multitímbrico por naturaleza, lo que significa que es capaz de generar más de un *patch* de sonido a la vez.

Como resultado de estas diferencias entre dispositivos, se crearon los *modos de recepción MIDI*, los cuales permiten a un instrumento transmitir o responder a los mensajes de canal MIDI de varias formas. Por ejemplo, un instrumento puede ser programado para responder a los 16 canales MIDI al mismo tiempo, mientras que otro puede ser polifónico por naturaleza y ser programado para responder solamente a un canal. Es también común para un instrumento ser polifónico y multitímbrico a la vez.

Los siguientes son los modos de recepción MIDI:

- Modo 1 – Omni on/Poly.
- Modo 2 – Omni on/Mono.
- Modo 3 – Omni off/Poly.
- Modo 4 – Omni off/Mono.

Omni on/off se refiere a cómo el instrumento MIDI responderá a los 16 canales. Cuando Omni se activa (on), el dispositivo responderá a todos los mensajes de canal que son transmitidos sobre todos los canales. Si Omni se desactiva (off) el dispositivo responderá solamente a un canal MIDI o un set de canales asignados.

Poly/Mono se refiere a la generación de notas individuales de un instrumento. En el modo *Poly*, un instrumento es capaz de responder

polifónicamente a cada canal MIDI, es decir, produce más de una nota a la vez. En el modo *Mono*, un instrumento es capaz de responder monofónicamente a cada canal, por tanto, genera una sola nota a la vez.

2.2.4.1. Modo 1 – Omni on/Poly.

En este modo el instrumento será capaz de responder polifónicamente para ejecutar instrucciones que son recibidas desde cualquier canal MIDI.

Figura 2.2. Ejemplo de asignación Voz/Canal en Modo 1 - Omni on/Poly.

2.2.4.2. Modo 2 – Omni on/Mono.

Cuando se trabaja en este modo, el instrumento asignará cualquier evento de nota recibida a una voz monofónica sin importar de qué canal MIDI se recibe. Este modo es raramente usado y se lo describe en la Figura 2.3.

Figura 2.3. Ejemplo de asignación Voz/Canal en Modo 2 - Omni on/Mono.

2.2.4.3. Modo 3 – Omni off/Poly.

En este modo un instrumento será capaz de responder polifónicamente para ejecutar instrucciones que son transmitidas sobre uno o más canales asignados. Por esto, es comúnmente usado por instrumentos poli tímbricos.

Figura 2.4. Ejemplo de asignación Voz/Canal en Modo 3 - Omni off/Poly.

2.2.4.4. Modo 4 – Omni off/Mono.

Al trabajar en este modo el instrumento será solamente capaz de generar una nota MIDI por canal. Un ejemplo práctico de este modo es frecuentemente usado en sistemas de guitarra MIDI, en donde la información es monofónicamente transmitida sobre seis canales consecutivos (un canal/voz por cuerda). Otros instrumentos electrónicos pueden hacer uso de este modo y permitir voces individuales para ser asignadas a cualquier combinación de canales MIDI.

Figura 2.5. Ejemplo de asignación Voz/Canal en Modo 4 - Omni off/Mono.

2.2.5. Mensajes de Canal (*Channel Messages*)

Como se dijo apartados anteriores, los Mensajes de Canal sólo actúan sobre los instrumentos asignados a ese canal. Estos mensajes constan de un byte de estado, seguido de uno o dos bytes de datos. En todos los mensajes de canal el *nibble* más significativo del byte de estado corresponde al comando en cuestión, mientras que el segundo *nibble* indica a cuál de los 16 canales MIDI afecta dicho comando.

2.2.5.1. Channel Voice Messages.

Estos mensajes están relacionados directamente con la producción de sonido e incluyen comandos como prender o apagar una nota, aplicar controladores, *pitch bending*, cambios de *patch*, etc. Son usados para transmitir

controladores, *pitch bending*, cambios de *patch*, etc. Son usados para transmitir datos de ejecución en tiempo real a través de un sistema MIDI conectado. Son generados cuando el controlador de un instrumento MIDI es tocado, seleccionado o variado en valor por el usuario. Ejemplo de este tipo de cambios podría ser cuando se toca un teclado (piano), botones de selección de programa o el movimiento de las ruedas de modulación o *pitch bending*.

Cada mensaje *Channel Voice* contiene un número de canal MIDI dentro de su status byte. Así, el dispositivo que esté asignado al mismo número de canal responderá a este mensaje. A continuación se presenta la Tabla 2.5 donde se sintetiza los siete *Channel Voice Messages* especificados dentro del protocolo MIDI.

Tabla 2.5. Channel Voice Messages

Channel Voice Messages					
Nombre	**Descripción**	**Status Byte**		**Data Byte 1**	**Data Byte 2**
		1º Nibble	**2º Nibble**		
		Comando	**Canal #**		
Note Off	Es usado para detener una nota MIDI específica.	1000	0000 a 1111	0000 0000 a 0111 1111	0000 0000 a 0111 1111
Note On	Es usado para indicar el inicio de una nota MIDI.	1001	0000 a 1111	0000 0000 a 0111 1111	0000 0000 a 0111 1111
Polyphonic-Key Pressure	Generados por instrumentos que son capaces de responder a los cambios de presión sobre una tecla.	1010	0000 a 1111	0000 0000 a 0111 1111	0000 0000 a 0111 1111
Control Change	Son usados para transmitir información que relaciona el control en tiempo real sobre los parámetros de desempeño de un instrumento MIDI.	1011	0000 a 1111	0000 0000 a 0111 1111	0000 0000 a 0111 1111
Program Change	Se utiliza para seleccionar un patch (programa) almacenado en la memoria del instrumento y asignarlo a un canal.	1100	0000 a 1111	0000 0000 a 0111 1111	
Channel Pressure (Aftertouch)	Generados por instrumentos que solo responden a una presión aplicada a sus controladores, sin importar el número de teclas presionadas.	1101	0000 a 1111	0000 0000 a 0111 1111	
Pitch Bend Change	Son generados por un instrumento cuando su rueda de pitch bending es movida de su posición central (sin pitch bend)	1110	0000 a 1111	0000 0000 a 0111 1111	0000 0000 a 0111 1111

		1º Nibble	**2º Nibble**	**Nota #**	**Velocidad**
		Comando	**Canal #**		
		1º Nibble	**2º Nibble**	**Nota #**	**Velocidad**
		Comando	**Canal #**		
		1º Nibble	**2º Nibble**	**Nota #**	**Presión**
		Comando	**Canal #**		
		1º Nibble	**2º Nibble**	**Controlador #**	**Valor del Controlador**
		Comando	**Canal #**		
		1º Nibble	**2º Nibble**	**Programa #**	
		Comando	**Canal #**		
		1º Nibble	**2º Nibble**	**Presión**	
		Comando	**Canal #**		
		1º Nibble	**2º Nibble**	**Pitch Bend LSB**	**Pitch Bend MSB**
		Comando	**Canal #**		

Muy pocos instrumentos responden o transmiten la velocidad de apagado de la nota. En ese caso se envía siempre un valor fijo, usualmente 0 ó 64. Cuando el segundo byte de datos es $0000\ 0000_2$ (velocidad = 0), el comando equivale a apagar nota. Esto es conveniente en *Running Status*[6] ya que se puede tanto prender como apagar notas sin necesidad de cambiar de status byte.

Un mensaje de *Control Change* o una cadena de éstos son transmitidos cuando un controlador es variado en tiempo real. De esta forma, un controlador puede ser usado para correspondientemente variar un amplio rango de posibles

[6] Referirse a la Sección 2.2.1. Tipos de Byte

parámetros de un instrumento o dispositivo en concordancia con el movimiento de dicho controlador.

Según la Tabla 2.5 el primer data byte del Control Change lleva el *Número del Controlador (Controller ID Number*). Este número es usado para especificar a cuál de los programas o parámetros del dispositivo MIDI es direccionado el mensaje. A pesar de que muchos fabricantes siguen la convención general para la asignación de estos números definida en las Especificaciones MIDI, dicha asignación es libre y cada fabricante acostumbra proveer una tabla donde se enlista los números de controlador y sus respectivas asignaciones a parámetros y controles (Ver Tabla 2.6).

Tabla 2.6. Lista de ID Numbers de controlador y sus asignaciones convencionales

14-BIT CONTROLLER MOST SIGNIFICANT BIT

Controller Number Hex	Decimal	Description
00H	0	Undefined
01H	1	Modulation Controller
02H	2	Breath Controller
03H	3	Undefined
04H	4	Foot Controller
05H	5	Portamento Time
06H	6	Data Entry MSB
07H	7	Main Volume
08H	8	Balance Controller
09H	9	Undefined
0AH	10	Pan Controller
0BH	11	Expression Controller
0CH	12	Undefined
.	.	.
.	.	.
0FH	15	Undefined
10H	16	General Purpose Controller #1
11H	17	General Purpose Controller #2
12H	18	General Purpose Controller #3
13H	19	General Purpose Controller #4
14H	20	Undefined
.	.	.
.	.	.
1FH	31	Undefined

14-BIT CONTROLLER LEAST SIGNIFICANT BIT

Controller Number Hex	Decimal	Description
20H	32	LSB Value for Controller 0
21H	33	LSB Value for Controller 1
22H	34	LSB Value for Controller 2
.	.	.
.	.	.
3EH	62	LSB Value for Controller 30
3FH	63	LSB Value for Controller 31

7-BIT CONTROLLERS

Controller Number Hex	Decimal	Description
40H	64	Damper Pedal (sustain)
41H	65	Portamento On/Off
42H	66	Sostenuto On/Off
43H	67	Soft Pedal
44H	68	Undefined
45H	69	Hold 2 On/Off
46H	70	Undefined
.	.	.
.	.	.
4FH	79	Undefined
50H	80	General Purpose Controller #5
51H	81	General Purpose Controller #6
52H	82	General Purpose Controller #7
53H	83	General Purpose Controller #8
54H	84	Undefined
.	.	.
.	.	.
5AH	90	Undefined
5BH	91	External Effects Depth
5CH	92	Tremolo Depth
5DH	93	Chorus Depth
5EH	94	Celeste (Detune) Depth
5FH	95	Phaser Depth

PARAMETER VALUE

Controller Number Hex	Decimal	Description
60H	96	Data Increment
61H	97	Data Decrement

PARAMETER SELECTION

Controller Number Hex	Decimal	Description
62H	98	Non-Registered Parameter Number LSB
63H	99	Non-Registered Parameter Number MSB
64H	100	Registered Parameter Number LSB
65H	101	Registered Parameter Number MSB

UNDEFINED CONTROLLERS

Controller Number Hex	Decimal	Description
66H	102	Undefined
.	.	.
.	.	.
78H	120	Undefined

RESERVED FOR CHANNEL MODE MESSAGES

Controller Number Hex	Decimal	Description
79H	121	Reset All Controllers
7AH	122	Local Control On/Off
7BH	123	All Notes Off
7CH	'124	Omni Mode Off
7DH	125	Omni Mode On
7EH	126	Mono Mode On (Poly Mode Off)
7FH	127	Poly Mode On (Mono Mode Off)

Los números de controlador 0 a 31 están reservados para controladores *continuos*. Enviando un byte de datos se logra una definición de 7 bits (128 valores). Estos controladores pueden sin embargo admitir una resolución de 14 bits (16.384 valores), en cuyo caso es necesario enviar un segundo mensaje portando otro byte de datos.

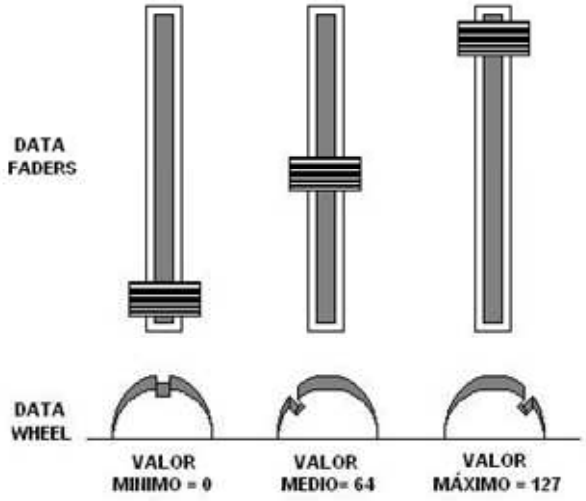


Figura 2.6. Rango de valores de datos para controladores continuos.

Los controladores 32 a 63 transmiten el segundo byte de datos de los controladores 0 a 31 respectivamente. Los controladores 64 a 95 se asignan a parámetros de baja resolución (*switches*) y sólo admiten un byte de datos, es decir, una resolución de 0 a 127. En este caso los valores 0 a 63 se reconocen como cero (Off) y los valores 64 a 127 como 127 (On).

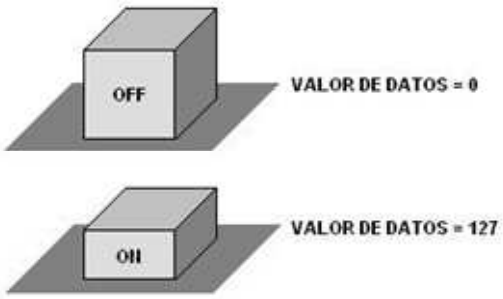


Figura 2.7. Controlador tipo switch.

El controlador #1 está asignado a la rueda de modulación. Los demás números de controlador quedan libres para ser determinados por cada fabricante, aunque hay varios tipos de controlador como son el volumen general, el pedal de sostenimiento, etc. que están asignados en forma prácticamente universal a un número determinado (ver Tabla 2.6).

El *pitch bending* no se transmite como controlador sino que tiene su propio tipo de mensaje pues requiere gran definición. Se utilizan dos data bytes para su valor de tal manera que la resolución sea de 14 bits[7] y poder hacer desplazamientos de altura bastante amplios (hasta una octava hacia arriba y hacia abajo) sin que se noten "escalones".

Los últimos siete números de controlador (121 a 127) no están asignados a controladores, sino que quedan reservados para mensajes de *Channel Mode*.

2.2.5.2. Channel Mode Messages.

Estos mensajes determinan la manera en que se recibe y responde a los datos MIDI en los diferentes canales. Los comandos de *Channel Mode* no tienen un byte de estatus propio, sino que se trasmiten bajo el formato de Control *Change Messages* con números de controlador del 121 a 127.

A continuación se presenta la Tabla 2.7 donde se sintetiza los siete *Channel Mode Messages* especificados dentro del protocolo MIDI.

[7] Recordar que el MSB del Data Byte siempre es 0, lo que deja 7 bits significativos.

Tabla 2.7. Channel Mode Messages

Channel Mode Messages					
Nombre	**Descripción**	**Status Byte**		**Data Byte 1**	**Data Byte 2**
		1º Nibble Comando	**2º Nibble Canal #**		
Reset All Controlles	Reestablece todos los controladores a sus valores por defecto.	1011	0000 a 1111	**Reset all Controllers**	**Simpre 0**
				0111 1001	0000 0000
Local Control	Desconecta el controlador de un dispositivo MIDI de su generador.			**Local Control**	**Off/On**
				0111 1010	0000 0000 0111 1111
All Notes Off	Apaga todas las notas de un canal determinado.			**All Notes Off**	**Simpre 0**
				0111 1011	0000 0000
Omni Mode Off	Desactiva el modo de recepción Omni.			**Omni Mode Off**	**Simpre 0**
				0111 1100	0000 0000
Omni Mode On	Activa el modo de recepción Omni.			**Omni Mode Off**	**Simpre 0**
				0111 1101	0000 0000
Mono Mode On	Activa el modo de recepción Monofónico.			**Omni Mode Off**	**Simpre 0**
				0111 1110	0000 0000
Poly Mode On	Activa el modo de recepción Polifónico.			**Omni Mode Off**	**Simpre 0**
				0111 1111	0000 0000

- **Reset - All Controllers.** Reestablece todos los controladores (continuos, *switch* e incrementales) a los valores que por defecto adquiere el instrumento al encenderse.

- **Local Control.** Abre o cierra la ruta que conecta el controlador (teclado u otro tipo) del dispositivo MIDI con su generador. En Local Control Off, accionar una tecla no producirá sonido en el sintetizador interno, aunque puede enviar datos vía MIDI a otros módulos. A su vez el generador puede responder a datos recibidos externamente.

- **All Notes Off.** Este mensaje apaga todas las 128 notas de un canal determinado. Se utiliza en casos de emergencia cuando una o más notas han quedado "colgadas".

- **Omni Mode Off.** Como se habló en la Sección 2.2.4 en el modo de recepción Omni, el generador responde a los mensajes MIDI transmitidos en cualquiera de los 16 canales. Omni Mode Off desactiva este modo de recepción y a la vez envía un mensaje de All Notes Off.

- **Omni Mode On.** Activa el modo Omni y envía un comando de *All Notes Off*.

- **Mono Mode On.** En modo Mono el generador responderá monofónicamente, es decir, asignando una sola voz a cada uno de los canales que esté activado, a partir de un número de canal determinado que funciona como canal base (*base channel*). El número de canales activados está determinado por el tercer byte. En el caso especial de que éste tenga valor 0, se asignan tantos canales como voces tenga el receptor, siempre a partir del canal base. Envía simultáneamente un mensaje de *All Notes Off*.

- **Poly Mode On.** Desactiva el modo Mono y habilita al dispositivo MIDI responder polifónicamente. Envía simultáneamente un mensaje de *All Notes Off*.

2.2.6. Mensajes de sistema (System Messages)

Como su nombre lo indica, los Mensajes de Sistema son globalmente transmitidos a todos los dispositivos MIDI conectados a una cadena MIDI. Esto se debe a estos mensajes no poseen una estructura de bytes que especifiquen un número de canal, lo que significa que cualquier dispositivos responderá a estos mensajes sin importar a qué canal o canales MIDI esté asignado el dispositivo. Existen tres tipos de Mensajes de Sistema:

- System Common Messages.
- System Real Time Messages.
- System Exclusive Messages.

2.2.6.1. System Common Messages.

Son usados para transmitir Código de Tiempo MIDI (MTC), el puntero de posición de canción, selección de canción, petición de tono y el fin de datos exclusivos en todo el sistema MIDI o los 16 canales de un puerto MIDI especifico.

A continuación se presenta la Tabla 2.8 donde se sintetiza los Mensajes *System Common* especificados dentro del protocolo MIDI.

Tabla 2.8. System Common Messages

System Common Messages				
Nombre	**Descripción**	**Status Byte**	**Data Byte 1**	**Data Byte 2**
			LSB	**MSB**
Song Position Pointer (SPP)	Ordena al secuenciador o máquina de percusión ubicarse en una determinada posición de la canción.	1111 0010	0000 0000 a 0111 1111	0000 0000 a 0111 1111
Song Selelct	Selecciona una canción de la memoria para ser tocada.	1111 0011	0000 0000 a 0111 1111	
Tune Request	Hace que todos los instrumentos del sistema realicen su rutina de afinación de acuerdo a su referencia interna.	1111 0110		
End Of Exclusive	Determina el fin de un mensaje System Exclusive.	1111 0111		

- **Song Position Pointer (SPP).** Permite a un secuenciador o una máquina de percusión sincronizarse a una fuente externa (como una grabadora) desde cualquier posición dentro de una canción. El Mensaje SPP es usado para referenciar un punto de posición dentro de una secuencia MIDI a la posición correspondiente del dispositivo externo. Este mensaje proporciona una referencia de tiempo que se incrementa una vez cada seis mensajes de reloj con respecto al comienzo de una composición.

- **Song Select.** Un mensaje de Selección de Canción es usado para solicitar una canción específica (identificada por su ID number) desde la memoria de secuencia interna de una máquina de percusión o secuenciador. Al ser seleccionada una canción ésta responderá a los mensajes de *stat, stop y continue*.

- **Tune Request.** Este mensaje es usado para solicitar a todos los instrumentos que inicialicen su rutina interna de afinación, si así lo permiten.

- **End Of Exclusive.** La transmisión de un mensaje *End of Exclusive* (EOX) es usado para indicar el fin de un mensaje *System Exclusive*.

- **MTC (MIDI Time Code).** Provee un medio efectivo y fácil para traducir el código de tiempo de sincronización estandarizado (SMPTE[8]) a un código equivalente conforme a las especificaciones MIDI. Permite que los comandos y códigos basados en el tiempo sean distribuidos en toda la cadena MIDI. Los mensajes MTC son transmitidos y reconocidos por dispositivos MIDI que son capaces de comprender y ejecutar comandos MTC.

2.2.6.2. System Real Time Messages.

La transmisión de un mensaje *System Real Time Message* provee un elemento de tiempo preciso para la sincronización entre dispositivos MIDI durante una aplicación. Para evitar retardos de tiempo, las Especificaciones MIDI permiten a estos mensajes ser insertados en cualquier punto dentro de la cadena de datos aún si otro mensaje MIDI se está transmitiendo.

A continuación se presenta la Tabla 2.9 donde se sintetiza los Mensajes *System Common* especificados dentro del protocolo MIDI.

[8] Código de tiempo SMPTE (Society of Motion Picture and Television Engineers), método estándar de enlace entre transportes de audio y video el cual permite la identificación de una posición exacta en una cinta magnética por asignación de direcciones digitales en cada locación.

Tabla 2.9. System Real Time Messages

System Real Time Messages		
Nombre	**Descripción**	**Status Byte**
Timing Clock	Controla la velocidad de funcionamiento de los secuenciadores esclavos.	1111 1000
Start	Ordena a los dispositivos MIDI empezar a tocar desde el inicio de sus secuencias internas.	1111 1001
Stop	Ordena a todos los dispositivos MIDI detener la ejecución de sus secuencias.	1111 1100
Continue	Ordena a todos los dispositivos conectados reanudar la ejecución de sus secuencias internas.	1111 1011
Active Sensing	Enviado para verificar que el instrumento está conectado al sistema y respondiendo.	1111 1110
System Reset	Resetea el dispositivo readoptando sus parámetros a valores iniciales por defecto.	1111 1111

- **Timig Clock.** Este mensaje es transmitido dentro de la cadena de datos MIDI a una velocidad de 24 veces por cuarto de nota. Es usado para sincronizar los relojes internos de cada dispositivo MIDI dentro del sistema.

- **Start.** Este comando ordena a todos los dispositivos MIDI conectados a empezar a tocar desde el inicio de sus secuencias internas.

- **Stop.** Este mensaje ordena a todos los dispositivos MIDI detener la ejecución de sus secuencias.

- **Continue.** Después de recibir un comando de Stop, este mensaje ordena a todos los dispositivos conectados reanudar la ejecución de sus secuencias internas desde el mismo punto en el cual se detuvieron.

- **Active Sensing.** Cuando la secuencia se encuentra en stop, este mensaje es transmitido dentro de la cadena de datos MIDI cada 300 milisegundos,

informando que el dispositivo se encuentra todavía activo (conectado) en el sistema.

- **System Reset.** Este mensaje se transmite manualmente para resetear un dispositivo o instrumento MIDI retornando éste a sus parámetros iniciales por defecto (comúnmente en Modo 1, Local Control On y All Notes Off).

2.2.6.3. System Exclusive Messages.

Los mensajes de sistema exclusivo (*SysEx*) están dirigidos a instrumentos de una marca y modelo determinados y son solamente aceptados por ellos. Cada fabricante tiene total libertad de determinar el formato y la longitud del mensaje pero debe respetar el formato del encabezamiento que incluye, precisamente, un número de identificación del fabricante. El sistema exclusivo fue pensado para poder enviar mensajes específicos de un aparato determinado, como por ejemplo el cambio de parámetros en la programación de los *patchs*, el traspaso de los programas de los bancos de memoria, etc. El *SysEx* no sólo permite este tipo de comunicación de un instrumento a otro del mismo modelo, sino también entre un instrumento y un secuenciador, ya sea con fines de almacenamiento de bancos de *patchs* o para modificar parámetros tímbricos durante una secuencia. Es utilizado además por los programas de computadora que editan y almacenan *patchs*.

El formato de transmisión consta de un encabezamiento o *header* cuyo primer byte es el de inicio de *SysEx* y el segundo byte es el ID del fabricante. Lo que sigue es libremente determinado por el fabricante pero generalmente el *header* tiene un tercer byte correspondiente al modelo. (Cuando una serie de modelos diferentes tienen arquitectura muy similar, el fabricante puede designar para todos ellos el mismo ID como forma de reservar números para el futuro. Luego viene una serie de cualquier longitud posible de bytes de datos con la única condición de que todos tengan 0 (cero) como bit más significativo. Iniciar con bit 1 significaría un byte de estatus y eso interrumpiría el *SysEx*. Los únicos bytes de

estatus que pueden intercalarse son los del tipo *System Real Time* (ver Sección 2.2.6.2). El fin del sistema exclusivo está indicado por el byte 1111 0111_2 (247_{10}).

Tabla 2.10. Formato de System Exclusive Messages

System Exclusive Messages			
	Formato		**Descripción**
Header	**Status Byte**	1111 0000	Inicio de Sistema Exclusivo.
	ID 1	0DDD DDDD	ID - número del fabricante.
	ID 2 (Opcional)	0iii iiii	Modelo.
Data	**Data Bytes**	0ddd dddd	Cualquier cantidad de bytes siempre y cuando que el bit más significativo sea 0.
		:	
		0ddd dddd	
Tail	**Status Byte**	1111 0111	EOX (End of SysEx) - fin de Sistema Exclusivo

Los *ID numbers* son distribuidos por la MMA[9] y por su contrapartida japonesa la JMSC[10]. Algunos de esos números, sin embargo, han sido asignados a mensajes de tipo universal como forma de ampliar la norma MIDI. Hay dos tipos de mensaje dentro de este formato:

- *Sample Dump Standard* (SDS).
- *Standard MIDI File* (o simplemente MIDI File)

- *Sample Dump Standard* es una forma estandarizada de traspasar *samples*, es decir muestras digitales de sonido. El formato admite gran variedad de frecuencias de sampleo y número de bits de cuantización, de forma de que puede ser implementado por cualquier *sampler*, aunque no todos los modelos se adaptan a este tipo de comunicación aparte del suyo propio. El SDS puede transmitir también los puntos de inicio y final de gran cantidad de *loops* dentro del *sample*.

[9] MMA, MIDI Manufacturers Asociation.
[10] JMSC, Japanese MIDI Standard Comitee.

- El *Standard MIDI File* es un formato también estandarizado para guardar secuencias, es decir, series ordenadas de eventos MIDI determinados en su ubicación temporal en términos de compases y tiempos. Hay diversos tipos de secuenciadores, tanto de software (programas de computadora que cumplen esa función), como de hardware (aparatos dedicados a ese fin, sean independientes o integrados a un sintetizador o *sampler*). Cada secuenciador tiene su propio formato de secuencias y el MIDI File es una forma de guardar la información básica en un formato estandarizado de manera de poder leer archivos de un secuenciador a otro. Prácticamente todos los secuenciadores, especialmente los de software, pueden tanto leer como guardar archivos en formato MIDI File.

2.3. ESPECIFICACIONES TÉCNICAS

El Protocolo MIDI permite la interconexión de sintetizadores, secuenciadores, computadoras personales, máquinas de ritmo, etc. a través de una interfaz estándar.

Cada instrumento equipado con MIDI contiene generalmente un receptor y un transmisor, aunque algunos pueden disponer de solo uno de ellos. El *receptor* recibe mensajes en formato MIDI y ejecuta los comandos MIDI respectivos. Éste consiste de un opto isolador, el módulo UART[11] y algún otro hardware requerido para ejecutar ciertas funciones. Los *transmisores*, por su parte, originan mensajes en formato MIDI y los transmiten a través de una línea UART.

La transmisión se hace en un sólo sentido, de manera que una comunicación de ida y vuelta necesita dos cables. Los dispositivos con interfaz MIDI tienen entonces un puerto de entrada y otro de salida, señalados como *MIDI IN* y *MIDI OUT*, respectivamente. En casi todos los casos se puede encontrar un tercer puerto llamado *MIDI THRU* que no sirve como puerto de envío sino

[11] UART, Universal Asynchronous Receiver Transmitter.

solamente como paso y por el que sale una copia exacta de lo que entra por el puerto MIDI IN.

Figura 2.8. Panel posterior de un dispositivo MIDI.

2.3.1. MIDI IN

El puerto MIDI IN recibe mensajes MIDI desde una fuente externa y comunica su desempeño, control y datos de tiempo al microprocesador interno del dispositivo.

Más de un puerto MIDI IN puede ser implementado dentro de un sistema para proveer funciones de mezcla o para dispositivos MIDI que soportan más de 16 canales. Otros dispositivos (como una MCS) pueden prescindir de un puerto de entrada por completo.

2.3.2. MIDI OUT

Este puerto es usado para transmitir mensajes MIDI desde una fuente simple a un microprocesador u otro instrumento o dispositivo MIDI. En el caso de estos puertos pueden requerirse más de uno de ellos en el diseño de un sistema para el simple propósito de proveer múltiples salidas MIDI (distribuyendo de la misma cadena de datos a un determinado número de instrumentos).

2.3.3. MIDI THRU

El puerto MIDI THRU provee una copia exacta de los datos entrantes en el puerto MIDI IN (Ver Figura 2.9). Es usado para transmitir estos datos hacia otro

instrumento o dispositivo que sigue en la cadena de datos sin mezclarlos con aquellos que son transmitidos en el puerto MIDI OUT.

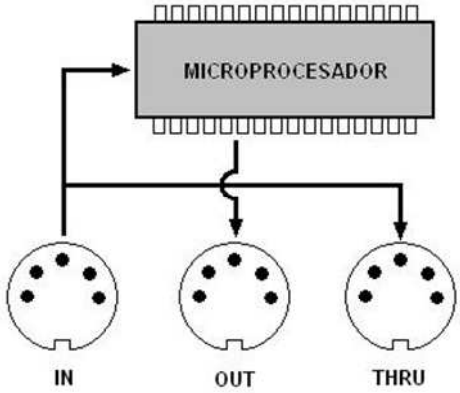


Figura 2.9. Ruta de la señal de los puertos MIDI IN, OUT y THRU.

2.3.4. MIDI ECHO

Ciertos dispositivos MIDI no incluyen un puerto MIDI THRU. Sin embargo pueden ofrecer una función de transmisión basada en software seleccionando entre los puertos MIDI OUT y MIDI ECHO. Así como el puerto THRU, la función seleccionable de eco es usada para proveer una copia exacta de la información recibida en el puerto de entrada enrutando estos datos hacia el puerto MIDI OUT/ECHO.

A diferencia del puerto de salida MIDI OUT, la función de eco puede a menudo ser seleccionada para mezclar los datos entrantes con los mismos datos generados por el dispositivo. De esta forma, más de un controlador puede ser simultáneamente colocado dentro de un sistema MIDI.

2.3.5. Velocidad de transmisión

La interfaz opera a *31.25 (±1%) Kbaudios*, asíncrona, con un bit de inicio, 8 bits de datos y un bit de parada, conformando así bytes de 10 bits con un período de 320 microsegundos por byte serial.

La corriente de transmisión es de 5mA, correspondiendo la presencia de corriente al bit lógico 0. Los puertos de entrada (MIDI IN) deben tener optoisoladores[12] que requieran menos de 5mA para dispararse y cuyo tiempo de subida y caída sea inferior a los 2 microsegundos.

2.3.6. Cables y conectores

Un cable MIDI consiste de un par entrelazado de cables conductores con conectores tipo DIN de 5 pines 180 grados en sus extremos. Las Especificaciones MIDI hacen uso de solo 3 de los 5 posibles pines. Los conectores hembra montadas en el panel del dispositivo MIDI y los machos en el cable. Las dos patas laterales (1 y 3) deben estar desconectadas para que el cable pueda transmitir. No están actualmente en uso pero están reservadas para posibles cambios en aplicaciones MIDI futuras. La pata del medio (2) está conectada a tierra y la transmisión se realiza por las dos patas restantes (4 y 5). La configuración estándar de conexión es la que se muestra en la Figura 2.10.

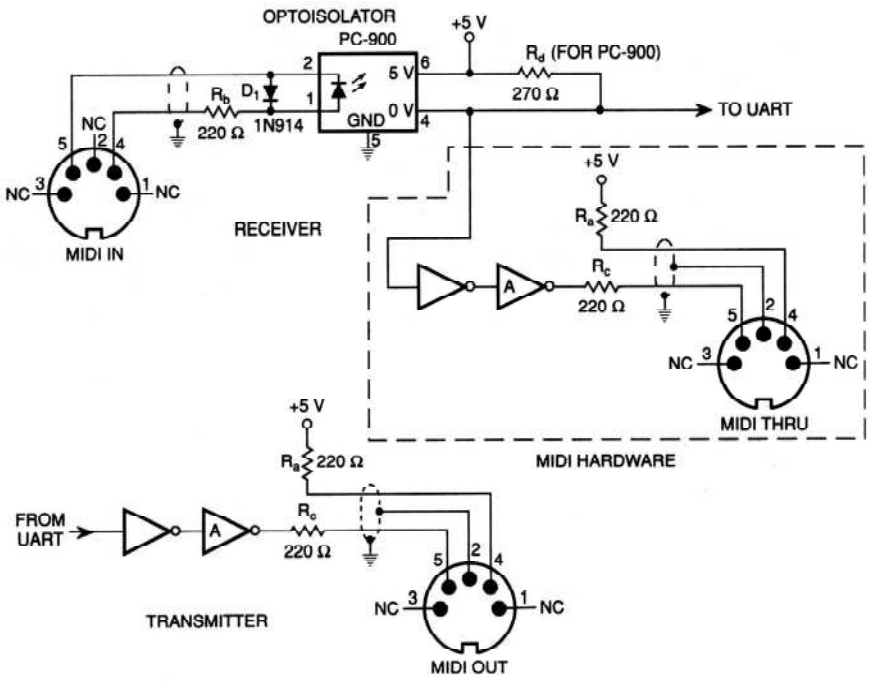


Figura 2.10. Configuración de hardware estándar para puertos MIDI IN, OUT y THRU.

[12] Se recomienda optoisoladores Sharp PC-900 y HP 6N138.

La longitud de un cable MIDI no debe superar en ningún caso los 15 metros, de modo de no generar demoras en la transmisión. Cada OUT debe conectarse a un solo IN. Si se quieren controlar varios dispositivos por una sola salida, puede hacerse una cadena usando los respectivos MIDI THRU, aunque en ese caso son necesarios optoisoladores muy veloces, ya que los errores de subida y caída de cada uno se adicionan. También hay que cuidar que la suma del largo de los cables no supere el máximo de 15 metros. En general, conviene no disponer más de tres unidades en serie y utilizar siempre cables cortos.

2.3.7. Conexiones

A pesar de que existe un amplio rango de dispositivos y diseños de sistemas MIDI, hay un determinado número de convenciones que permiten a los dispositivos MIDI conectarse fácilmente dentro de cualquier sistema. Esas configuraciones permiten a los datos MIDI ser transmitidos de la manera más eficiente posible.

Como regla principal se debe tener en cuenta que existen dos únicos métodos válidos para conectar un dispositivo MIDI con otro:

1. Conectando el puerto MIDI OUT (o el puerto MIDI Echo) de un dispositivo en el puerto MIDI IN de otro, y,
2. Conectando el puerto MIDI THRU de un dispositivo en el puerto MIDI IN de otro.

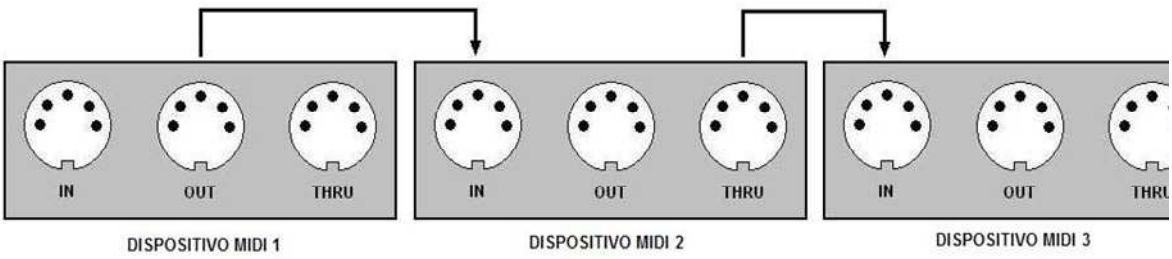


Figura 2.11. Los dos métodos válidos de conexión de un dispositivo MIDI con otro.

La Figura 2.11 muestra la conexión típica de varios dispositivos en serie conocida con el nombre de *Daisy Chain*, siendo una de las más simples y más comúnmente usadas. Este método distribuye una línea de datos MIDI a cada dispositivo dentro de un sistema transmitiendo los datos al primer instrumento y subsecuentemente pasando una copia exacta a los restantes dentro de la cadena a través de los puertos MIDI THRU, siendo el primer dispositivo la fuente de información y los restantes dispositivos esclavos.

Otra forma de conformar un sistema integrado por varios dispositivos es mediante una disposición en estrella. Para ello es necesario un *MIDI Thru Box*, un dispositivo que presenta una entrada y varias salidas. De esa forma se evita la suma de las demoras que se dan en la transmisión en serie.

Figura 2.12. MIDI Thru Box de 3 entradas y 8 salidas marca M-Audio modelo Thru 3x8.

2.4. PC DENTRO DE PRODUCCIÓN MIDI

El computador personal es a menudo el componente central dentro de la mayoría de los sistemas MIDI. Por medio de programas de software y hardware externo, el PC es comúnmente usado para procesar datos de desempeño y configuración de un dispositivo MIDI desde una posición de control integrada.

La ventaja más importante de los computadores personales es su alta velocidad de proceso digital. Esto permite al usuario o compositor añadir dispositivos externos que mejor se ajusten a sus requerimientos de producción de audio.

Otra ventaja significativa es la gran cantidad de plataformas de audio digital existentes en el mercado tanto para sistemas IBM o compatibles como para sistemas Macintosh. Es este factor el que convierte al los computadores personales en auténticos camaleones digitales debido a su capacidad de adecuarse a los requerimientos que una aplicación puede demandar. Además, el computador permite al usuario escoger los sistemas de software que mejor se acomoden a sus necesidades y sus metas personales de producción.

Como se dijo anteriormente existe una gran variedad de software dedicado a la producción de audio digital. A menudo al adquirir un instrumento o dispositivo MIDI, éste viene con su propio software de aplicación creado por el mismo fabricante. Sin embargo, debido a la aceptación universal del protocolo MIDI, todo dispositivo externo puede ser compatible y reconocido por cualquier plataforma de audio configurando sus opciones para dispositivos externos.

2.4.1. Reason de Propellerheads

La plataforma de audio digital *Reason* con su última versión 3 se presenta como una de las opciones más aceptadas a nivel mundial por productores y artistas debido a su excelente calidad en audio, la posibilidad de adquirir actualizaciones a través del Internet y por su gran capacidad de expansión.

Reason 3 es un estudio virtual de audio con todas las herramientas e instrumentos que el usuario necesita para la creación musical; entre ellos podemos encontrar sintetizadores, samplers, máquinas de ritmo, herramientas de masterización profesional, mezcladores, efectos de clase mundial, secuenciadores y más.

Figura 2.13. Paquete de software de Reason 3.

Cada unidad en el rack virtual de Reason puede ser editado desde su propio panel frontal en pantalla. Todos los *faders*, perillas, *push buttons* y funciones se encuentran en frente del usuario listos para ser movidos en tiempo real. Además, todas las acciones que el usuario ejecute en el panel frontal (ajustes de filtros, *pitch bending, panning*, etc.) pueden ser grabadas y automatizadas en el Secuenciador Reason.

Reason 3 reconoce y trabaja con una gran cantidad de sintetizadores y Superficies de Control, detectándolos automáticamente (si así lo permite el dispositivo) o configurando su instalación. Los teclados MIDI o los dispositivos de control remoto son llamados superficies de control. La entrada MIDI procedente de las superficies de control es gestionada por un sistema llamado Remote.

Los mandos, *faders* y botones de las superficies son distribuidos automáticamente a los parámetros más usados de los dispositivos Reason. Para las superficies de control que no son admitidas automáticamente se puede usar *drivers* genéricos.

2.4.1.1. Requerimientos del sistema.

- **Windows**

  - Procesador Intel Pentium III 600 MHz o mejor.
  - 256 MB en RAM.
  - 2 GB de espacio libre en disco.
  - CD-ROM drive.
  - Windows XP/2000.
  - Monitor con 800x600 pixels de resolución o mejor.
  - Tarjeta de sonido de 16-bit compatible con Windows, preferiblemente con drivers DirectX o ASIO.
  - Recomendado: teclado MIDI con interfaz interna MIDI o teclado MIDI e interfaz MIDI.

- **Mac**

  - Procesador G4 o G5.
  - 256 MB RAM.
  - 2 GB de espacio libre en disco.
  - CD-ROM drive.
  - Mac OS X 10.2, 10.3 (recomendado).
  - Monitor con 800x600 pixels de resolución o mejor.
  - Recomendado: teclado MIDI con interfaz interna MIDI o teclado MIDI e interfaz MIDI.

2.4.2. SONAR de Cakewalk

SONAR en su versión 4 *Producer Edition* tiene una reputación bien merecida, pues a provisto al mundo de la producción de audio digital una poderosa interfaz de usuario, rápida, dinámica y verdaderamente amigable.

SONAR es una herramienta profesional para crear sonidos y música en un computador personal. Está diseñado para músicos, compositores, ingenieros de audio y producción, creadores de juegos y multimedia, e ingenieros de grabación. SONAR es compatible con Wave, MP3, ondas ACIDized, WMA, AIFF, archivos MIDI y otros populares formatos y proporciona las herramientas necesarias para realizar tareas de calidad profesional de forma rápida y eficaz.

SONAR es más que un conjunto de software de creación de audio digital y MIDI integrado: es una plataforma ampliable que puede funcionar como el sistema nervioso central de un estudio de grabación.

Figura 2.14. Paquete de software de SONAR 5.

Su reputación como el secuenciador MIDI más potente del mercado lo precede. Entre muchas nuevas disponibilidades prácticas, Sonar 4 es una solución excelente para la composición, edición, la mezcla y la producción de música, audio, TV, video, etc.

2.4.2.1. Requerimientos del sistema.

- Sistema operativo: Mínimo Windows 2000 (recomendado Windows XP).
- Procesador de 1.2 Ghz o superior.
- 128 MB en RAM.
- Tarjeta de sonido: WDM o ASIO (Windows compatible).

2.5. MIDI COMO ESTÁNDAR UNIVERSAL

La aceptación mundial de MIDI se basó principalmente en la necesidad de contar con un protocolo estandarizado que permitiera la comunicación entre dispositivos de distintos fabricantes. Fue debido también, en parte, a la introducción del sintetizador DX-7 de Yamaha en el invierno de 1983, después de la cual la venta de sintetizadores empezó a crecer significativamente.

Gracias a la publicación de las Especificaciones MIDI, la aceptación del protocolo por la mayor parte de fabricantes de interfaces PC/MIDI, el soporte de Microsoft Windows y otros sistemas operativos para MIDI y la evolución de sintetizadores a bajo costo, el protocolo es ahora ampliamente usado en un creciente número de aplicaciones. Ha evolucionado hasta el punto que está siendo utilizado para toda clase de tareas; desde procesos de automatización de mezcla hasta la composición asistida por computadora, desde el control remoto de grabadoras y luces de escenario hasta la trascripción de notas musicales.

El potencial de expansión en el futuro y las capacidades de control incrementadas sobre un sistema de producción integrado ha permitido el crecimiento de una industria que es también muy personal en naturaleza. Por primera vez en la historia de la Música, es posible para un individuo realizar una producción de sonido a escala completa de una forma efectiva en costo y tiempo.

Ya sea en el estudio de grabación o en nuestros hogares, MIDI está siendo usado por profesionales y no profesionales en un sinnúmero de aplicaciones debido básicamente a su bajo costo de producción, poder de expansión y velocidad. El poder y flexibilidad de MIDI son atributos inherentes dentro de las capacidades que el protocolo trae a la producción musical moderna.

2.6. MIDI IMPLEMENTATION CHART

No siempre es necesario que un dispositivo transmita y reciba todos los posibles mensajes definidos en las especificaciones MIDI. Aparte de niveles,

calidad y prestaciones, debido a la extensión del protocolo, cada aparato sólo está ideado para realizar tareas MIDI específicas. Para averiguar qué capacidades posee un aparato determinado surgió en 1985 la llamada *Tabla de Implementación MIDI,* una hoja con formato estándar de 4 columnas por 12 filas que se adjunta a las instrucciones del dispositivo.

En la cabecera de dicha Tabla consta el modelo de aparato, versión y fecha. En la primera columna se especifican las funciones MIDI en cuestión, indicando si cada una de ellas está implementada en el aparato (O: sí ó X: no) y si lo es en transmisión (*Transmited*), recepción (*Recognized*) o en ambos modos, en las columnas segunda y tercera, respectivamente. La cuarta columna (*Remarks*) sirve para indicar algún comentario de cada apartado y ampliar su información, si procede.

A pesar de los esfuerzos para una completa estandarización de estas tablas se pueden encontrar pequeñas variaciones en cuanto a los símbolos y sus significados, detalles que son referidos, generalmente, en la esquina inferior derecha de la tabla. La Tabla 2.11 muestra un ejemplo de una Tabla de Implementación MIDI para una tarjeta de sonido.

Tabla 2.11. Ejemplo de una MIDI Implementation Chart

Función	Transmisión	Recepción	Observaciones
Canales MIDI	X	1 - 16 1 - 16	
Modo	X	3	
Número de la nota	X	0 - 127	
Velocidad Nota Activ. Nota Desact.	 X X	 9n, V = 0 - 127 8n, V = 0 - 127	
Pospulsación de teclas Pospulsación de canal	X X	X X	
Inflexión de tono *1	X	O	+/-2 octavas Inflexión de tono Sensibilidad reconocida
Cambio de control *1 0, 32 1 6, 38 7 10 11 64 91 93 100 101 120 121 123	 X X X X X X X X X X X X X X	 O O O O O O O O O O O O O O	Selección de bancos Modulación Introducción datos Volumen principal Pan Expresión Pedal de sostén Profundidad de reverb Profundidad de coro RPN LSB RPN MSB Sonidos desactivados Restablecer controlad. Todas las notas desact.
Cambio de programa	X	O 0 - 127	

Notas:

*1 : Todos los canales son modulables en volumen MIDI (incluido los de percusión)
Opciones de arranque: Inflexión = 2 semitonos; volumen principal = 100; controladores, normal.

Modo 1: OMNI ACTIV., POLI
Modo 2: OMNI ACTIV., MONO
Modo 3: OMNI DESACT., POLI
Modo 4: OMNI DESACT., MONO

O : Sí
X : No

En el pie de la tabla aparece un recordatorio de los Modos MIDI y un espacio para realizar alguna anotación genérica. Algunos aparatos incluyen unas hojas extra en los manuales de instrucción, además de la Tabla de Implementación, con

una revisión exhaustiva de la interpretación MIDI que hace el aparato. En caso de hacer uso de mensajes de Sistema Exclusivo resulta imprescindible tener a mano esa documentación extra, pues el aparato va a tener un comportamiento personalizado con los mismos. Lo mismo ocurre con dispositivos cuya finalidad se separa de la musical y tendrán su forma especial de interpretar el MIDI, incluso los mensajes más básicos y estándar.

CAPÍTULO 3

3. MIDI CONTROL SURFACES (MCS)

3.1. INTRODUCCIÓN

Un creciente número de músicos, tanto caseros como profesionales, están optando por los sistemas de grabación y edición en disco duro para tomar ventaja de las ilimitadas posibilidades que las computadoras personales pueden ofrecer. No cabe duda que la nueva tecnología ha traído un sinnúmero de cambios en la forma que los músicos piensan acerca de la ejecución de sus trabajos de producción. En un pasado no tan distante, las pistas musicales eran mezcladas enteramente por el músico sentado frente a una consola de mezcla, empujando faders arriba y abajo, girando *knobs*, tratando de alcanzar esa ecualización perfecta.

En el ambiente de las Estaciones de Trabajo de Audio Digital (DAW[1]) de hoy en día, todo lo podemos encontrar gráficamente en un simple monitor de pantalla a color (Ver ejemplo Figura 3.1). A pesar de que las virtudes de las actuales DAW parecieran incrementarse exponencialmente año tras año, el diseño básico de interfaz de un sistema de audio de escritorio ha permanecido, virtualmente, intacto por alrededor de una década. Un monitor, un teclado y un ratón, todavía son los métodos principales para manipular todos los bits y bytes en un proyecto musical asistido por computadora. Desafortunadamente, como lo han descubierto muchos músicos, un ratón y un teclado no son las mejores y más intuitivas herramientas para la producción de audio digital.

[1] DAW, siglas en Inglés para Digital Audio Workstation.

Figura 3.1. Pantalla tomada de Pro Tools 6.7, una de las más populares DAW en el mercado actual.

Suficiente tiempo ha pasado desde la introducción de las DAW para que los músicos ahora tengan una mejor perspectiva de las ventajas y desventajas tanto de las formas antiguas como de las nuevas para el control de parámetros MIDI, despertando nuevos intereses en cuanto a herramientas para la producción de audio digital.

3.2. DEFINICIÓN

Una Superficie de Control MIDI (MCS) es un Dispositivo de Interfaz Humana (HID[2]) expresamente diseñado para la transmisión de eventos MIDI dentro de una DAW o un sistema modular MIDI. Ofrece un control inmediato e intuitivo en las tareas comunes de mezcla, grabación y edición, por medio de combinaciones variables de faders, *knobs, push buttons*, ruedas de inflexión y otros controles dedicados[3].

[2] HID, siglas en Inglés para Human Interface Device.
[3] Para más información sobre Controladores referirse al Capítulo 1 sección 1.2.

Una MCS es un dispositivo que comúnmente no contiene un generador de tonos o elementos de producción de sonido, sino más bien, posee un conjunto de controladores en tiempo real y una arquitectura de memoria, que a través de un microprocesador, permiten la configuración de sistemas MIDI complejos. Esto lo consigue por medio de la transmisión de mensajes MIDI al momento que el microprocesador detecta un cambio en el valor de un controlador determinado, es decir, cuando el usuario gira una perilla, desliza un fader o presiona un botón. El microprocesador genera el código necesario (Mensajes MIDI) que un programa de aplicación musical requiere cuando el usuario manipula físicamente un control asignado a un parámetro MIDI.

El software ligado a la MCS permite la asignación de parámetros MIDI a los controles de la Superficie mediante el enrutamiento de los controles virtuales en pantalla hacia los físicos de la MCS. El código generado por el microprocesador es enviado según las especificaciones del protocolo MIDI a través de un cable hacia el PC donde el software de aplicación reconoce el mensaje y se encarga de transformarlo y ejecutar la función MIDI que el usuario realizó físicamente.

Este control dinámico y táctil de los parámetros MIDI hace que las MCS estén siendo utilizadas por un creciente número de músicos que ven en estos dispositivos una herramienta cómoda que provee un eficiente método para mantener un flujo de trabajo ágil, tarea que no es fácilmente alcanzable con tan sólo un teclado y un ratón.

3.3. COMPONENTES BÁSICOS

El grado de sofisticación de una Superficie de Control dependerá del número de funciones MIDI y controladores que pueda incluir. Cabe resaltar que estos dispositivos son diseñados y construidos para realizar tareas específicas y que, por la gran extensión que posee MIDI, no incluyen todas sus funciones. Una Superficie de Control básica puede incluir al menos los siguientes componentes:

- Potenciómetros de deslizamiento (Faders), generalmente para el ajuste de volumen de los canales MIDI. Pueden ser programados para ejecutar otras funciones como *pitch bending*, ganancia de filtros, por medio de software. Se requiere de al menos uno de estos controladores.
- Perillas (*Knobs*), para el control de *panning* (balance stereo), volumen o ajuste de altos y bajos, de un canal MIDI. Su número dependerá de cuántos canales MIDI dispone la MCS.
- *Push Buttons*, la función básica de estos controladores tipo *Switch* es la activación y desactivación de los comandos Mute y Solo, para un canal MIDI determinado. Pueden ser asignados también a funciones de encendido de filtros y ecualizadores, o para funciones de configuración de la MCS.
- Ruedas de Inflexión de altura o modulación, de las cuales se puede prescindir puesto que un potenciómetro de deslizamiento puede cumplir sus mismas funciones. La inclusión de estos controladores dan a la MCS un control extra que enriquece los procesos de mezcla, masterización o edición.

Una Superficie de Control que posea al menos estos componentes estará en capacidad de ejecutar funciones básicas de MIDI, como son el control de volumen, *mute*, *solo, pitch bending* o *panning*, para un canal MIDI determinado. Funciones que son ampliamente requeridas en procesos de mezcla y edición de audio digital.

3.4. INTERFACES

Para usar MIDI con una computadora personal se requiere generalmente de un interfaz (existen unas pocas PC que vienen equipadas con un interfaz MIDI interna como tal). No obstante, existen varias interfaces MIDI para PC. Los tipos más comunes de interfaces para IBM y compatibles son las tarjetas de sonido, las cuales se conectan a un *slot* de expansión asignado en el *main board* de la PC. Así mismo, existen también interfaces MIDI que se conectan a través del puerto serial o del puerto paralelo de la PC.

La función fundamental de un interfaz MIDI para PC es convertir los bytes de datos paralelos del la PC al formato de datos serial de MIDI y viceversa, es decir, una función UART. Sin embargo, interfaces MIDI “inteligentes” pueden proveer varias funciones más sofisticadas, como la generación de mensajes MTC (*MIDI Timing Code*)[4], *buffering*, filtración y sincronización de datos MIDI hacia dispositivos externos y más.

El estándar por defecto para tarjetas de interfaz MIDI para PC es la *Roland MPU-401* (Figura 3.2). La MPU - 401 es un interfaz inteligente el cual también soporta el modo de operación UART. Muchas tarjetas de sonido internas incluyen interfaces MIDI las cuales implementan las funciones de la MPU – 401.

Figura 3.2. Tarjeta Roland MPU - 401.

Las aplicaciones de Windows direccionan los dispositivos de hardware como interfaces MIDI, MCS o sintetizadores por medio del uso de drivers. Los drivers proveen software de aplicación con un interfaz común a través del cual el hardware puede ser accedido. Esto simplifica los problemas de compatibilidad de hardware. Muchas de las MCS que utilizan un interfaz USB poseen sus propios drivers que deben ser instalados previo a la conexión con la PC. Si la MCS no requiere de driver para su conexión, el software de aplicación se encargará de direccionar los mensajes MIDI a través de un puerto serial o paralelo según la configuración del fabricante.

[4] Referirse al Capítulo 2 sección 2.2.6.1 System Common Messages.

3.5. CONEXIONES

Como se dijo en el Capítulo 2 sección 2.3.7, existe un determinado número de convenciones que permiten a los dispositivos MIDI conectarse fácilmente dentro de cualquier sistema. La configuración en cadena o *Daisy Chain* es la más simple y más utilizada. Sin embargo, si vamos a trabajar solamente con nuestra MCS y una computadora personal, la conexión requeriría de un solo cable MIDI conectado como se muestra en la Figura 3.3.

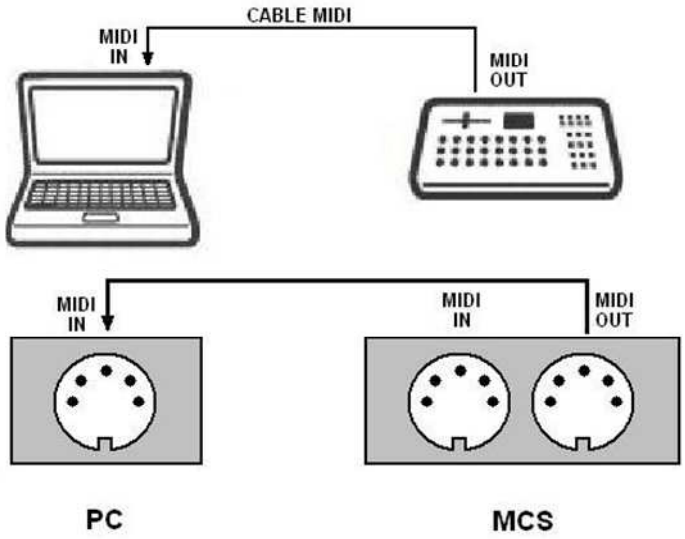


Figura 3.3. Conexión de la MCS a una computadora personal.

La conexión a través del cable MIDI conectaría el puerto MIDI OUT de la MCS con el puerto MIDI IN de la PC en una transmisión unidireccional. En la actualidad existen MCS que permiten la comunicación bidireccional a través de un interfaz USB. Este tipo de interfaz sería muy útil cuando se dispone, por ejemplo, de faders motorizados[5] en la MCS.

[5] Faders controlados tanto manual como eléctricamente (por medio de un motor DC).

3.6. APLICACIONES

No cabe duda que las MCS han encontrado su principal aplicación en el campo de la producción de Audio Digital y que de hecho, el desarrollo del presente proyecto está enfocado en este campo. Sin embargo, no se debe olvidar que, como su nombre lo indica, una Superficie de Control, es un dispositivo de transmisión y control de mensajes MIDI, y que las aplicaciones que esto conlleva pueden ser diversas.

3.6.1. MCS en Audio Digital

Como se habló en la introducción de este capítulo, las Superficies de Control MIDI nacieron por la necesidad de un mecanismo que permitiera mejorar el desempeño del músico en procesos de producción de sonido. Así, cuando empezaron a aparecer las primeras estaciones de trabajo de audio digital u otros sistemas MIDI, aparecieron también nuevas inquietudes para la creación de herramientas que abarquen ampliamente el entorno MIDI y superen las capacidades que un teclado de PC y un ratón podían ofrecer.

De allí que las aplicaciones de las MCS encontraron su contexto en su mismo origen: ser el interfaz entre el usuario y las Estaciones de Trabajo de Audio Digital. Sea para procesos de creación, grabación, edición o masterización, es decir, producción y post producción de sonido, las MCS son la herramienta más comúnmente usada por músicos profesionales y no profesionales tanto en estudios de grabación como en el hogar.

Hoy por hoy se puede afirmar que sin excepción todas las plataformas de audio digital incluyen como parte de su configuración la instalación de Superficies de Control MIDI.

3.6.2. MCS en Iluminación

Otra de las aplicaciones más importantes a las que ha sido enfocada la utilización de MCS es en el campo de la iluminación. Sonido e iluminación van de la mano cuando de montaje de espectáculos se refiere. Un seguimiento de las luces de escenario aporta en gran medida al desempeño de un espectáculo. Por tal motivo, el diseño de mecanismos que permitan un control de la iluminación a través de mensajes MIDI, ha sido desarrollado desde los inicios de éste.

En el mercado actual existen varios dispositivos que permiten el control de la iluminación (Ver ejemplo Figura 3.4). En su mayoría son módulos que se conectan a una MCS u otro dispositivo generador de mensajes a través de puertos MIDI. El dispositivo al recibir cierto evento MIDI ejecuta un comando para el control de la intensidad de una o varias salidas (canales MIDI). Un mensaje de Note On encenderá una lámpara en una salida del módulo, mientras que una rueda de modulación o fader creará efectos de desvanecimiento de luz. Todo dependerá de cómo esté programada la Superficie de Iluminación. Esto ofrece una forma muy intuitiva de controlar la intensidad de luces por medio de programación o por la misma acción de tocar un teclado MIDI, una máquina de ritmo, un secuenciador u otro dispositivo MIDI.

Figura 3.4. Superficie de Iluminación marca Logiq Electronics modelo MLD-4500, con 4 salidas controladas por MIDI.

3.7. EQUIPOS EXISTENTES: CARACTERÍSTICAS Y COSTOS

Debido a la gran acogida de las Superficies de Control MIDI por parte de músicos y productores de todos los ámbitos, las MCS se han convertido hoy en día en un equipo indispensable en todo sistema MIDI, siendo ahora un dispositivo de inclusión estándar en toda DAW.

Por tales razones, el diseño, construcción y comercialización de estos dispositivos ha crecido enormemente a nivel mundial. Existen varias casas comerciales muy reconocidas por la gran aceptación que han recibido sus equipos. Entre ellas las más importantes son:

3.7.1. X- Session de M-Audio

Figura 3.5. Superficie de Control MIDI marca M-Audio modelo X-Session.

3.7.1.1. Características Generales.

- 16 controles MIDI asignables.
- 10 botones MIDI asignables.

- Fader asignable de 60mm para mezclar y *scratching*[6].
- La asignación global de canales permite enviar datos de control a través de 16 canales MIDI con tan solo pulsar un botón.
- Funciona con aplicaciones de audio o en modo autónomo (sin necesidad de computadora)
- El programa de gestión de bibliotecas *Sysex* permite guardar y abrir configuraciones (sólo con PC).
- Indicación de asignación de controladores en la pantalla LCD
- Soporte nativo para Windows XP y Mac OS X.

3.7.1.2. Especificaciones.

- Alimentación: USB o adaptador de 9V CC 250-300mA.
- Dimensiones: 35cm x 17cm x 9cm.
- Peso: 1.1Kg.

[6] Scratching, técnica utilizada por DJ's consistente en mover un disco de vinilo hacia adelante y hacia atrás sobre el plato del tocadiscos para crear un efecto parecido al de rayar el disco y que, bien utilizado, ayuda a construir ritmos y frases melódicas.

3.7.1.3. Tabla de Implementación MIDI.

Tabla 3.1. Tabla de Implementación MIDI para el modelo X-Session

Función	Transmitido	Recibido	Comentarios
Basic :Default Channel: Changed	1 - 16 1 - 16	X	
:Default Mode :Message :Altered	--------- X ********	X	
Note Number: True Voice	0-127 *********	X	
Velocity :Note ON :Note OFF	0 X	X	
After: Keys Touch: Ch's	X 0	X	
Pitch Bend	0	X	
Control 0 - 119 Change 120 - 127	0 0	X X	
Program Change: True Number	0 - 127 *********	X	
System Exclusive	GM, GM2, MMC	Memory Dump	
Song Position Common: Song Select	X X	X	
System : Clock Exclusive: Commands	X X	X	
Aux :Local ON/OFF Messages :All Notes OFF :Active Sense :Reset	0 0 0 0	X	
Notes:		**0 = YES**	**X=NO**

3.7.1.4. Requerimientos del Sistema.

- Para PC:
    - Windows 98/Me/2000/XP.
    - Pentium II, 300MHz, 128MB en RAM.
    - Puerto USB.

- Tarjeta de sonido compatible con ASIO[7] o MME[8].

- Para Mac:

  - iMac, G3 o superior con USB.
  - OS9 o OSX.

3.7.1.5. Referencias:

- **M-Audio Nombrada "La Compañía del Año" por la Revista *Music Trades***. La revista cita su conocedor equipo de Investigación y desarrollo y su alta inversión en soporte técnico como razones claves para su continuo éxito.

3.7.1.6. Costo:

- $149.99 USD.

3.7.2. US-224 de TASCAM

Figura 3.6. Superficie de Control MIDI modelo US-224 marca TASCAM.

[7] ASIO, siglas en Inglés para Audio Stream In/Out, protocolo de transferencia multicanal para audio digital y MIDI creado por Steinberg (Cubase).
[8] MME, siglas en Inglés para Multimedia Extensión, similar a ASIO, creado por Microsoft.

3.7.2.1. Características Generales.

- Audio de alta calidad de 24 bits con dos entradas y dos salidas.
- Alimentación por interface USB compatible con PC y Mac.
- 4 faders, 4 perillas, una rueda de datos y más controles para parámetros para DAW.
- Grabación hasta en 48kHz con 24 bits de resolución.
- 16 canales MIDI con interface In/Out.
- Incluye GigaStudio 3 LE 64-voice streaming sampler software.
- Pequeño, portable y fácil de manejar.

3.7.2.2. Especificaciones Técnicas.

- Dimensiones: 30.4 x 21.1 x 6.2cm.
- Peso: 0.85Kg.
- Consumo de energía: 2W.
- Frecuencia de respuesta: 20Hz - 20KHz.
- Nivel de ruido: mejor a 92dB.

3.7.2.3. Tabla de Implementación MIDI.

Tabla 3.2. Tabla de Implementación MIDI para el modelo US-224

Función		Transmitido	Recibido	Comentarios
Basic Channel	Default Changed	X X	X X	Through
Mode	Default Messages Altered	X X ...	X X	Through
Note Number	True Voice	X ...	X	Through
Velocity	Note ON Note OFF	X X	X X	Through
After Touch	Keys Channels	X X	X X	Through
Pitch Bender		X	X	Through
Control Change		X	X	Through
Program Change	True #	X ...	X	Through
System Exclusive		X	X	Through
System Common	:Song Pos :Song Sel :Tune	X X X	X X X	Through
System Real Time	:Clock :Commands	X X	X X	Through
Aux Messages	:Local ON/OFF :All Notes OFF :Active Sense :Reset	X X X X	X X X X	Through

3.7.2.4. Requerimientos del Sistema.

- Para PC:
    - Pentium II 266MHz (o equivalente).
    - Windows 98, 2000 or XP
    - 96MB RAM.
    - Recomendado: Pentium II 300MHz, 128MB RAM o mejor.

- Para Mac:

    - MacOS 8.6 o mejor.
    - Puerto USB
    - La US-224 no ha sido probada con Mac O/S X.

3.7.2.5. Referencias:

- Los productos de TASCAM son usados a nivel mundial por los mejores productores, músicos y DJ's del momento. Artistas como Alan Parson, Dave Mustaine (Megadeath), o productores de sonido como Bart Hendrickson (Spiderman 2, Los Piratas del Caribe, The Last Samurai) han utilizado productos TASCAM y los recomiendan[9].

3.7.2.6. Costo:

- $249.00USD.

3.7.3. BCF2000 de Behringer

Figura 3.7. Superficie de Control MIDI modelo BCF2000 de Behringer.

[9] Para mayor información visitar la página web http://www.tascam.com/UserStories.html

3.7.3.1. Características Generales.

- Controlador MIDI USB con automatización total, tacto analógico, intuitiva interfaz de usuario y expansible mediante conexión en cascada.
- 8 faders motorizados ultraprecisos de 100mm para un control sin igual de mezcladores, sintetizadores y samplers virtuales.
- 16 + 4 pulsadores luminosos de asignación libre para todo tipo de funciones MIDI, desde note on/off, cambio de control, cambio de programa hasta SysEx.
- Todos los controles son libremente asignables.
- Conectores multifunción para pedal y controlador de pie que permiten un fácil control vía MIDI.
- 32 *presets* de usuario con 4 grupos de codificadores cada uno.
- Modos MIDI y USB configurables para una integración más flexible.
- 1 entrada y dos salidas MIDI; utilizable como interfaz MIDI USB.
- Pantalla multifunción de 4 dígitos con indicación de parámetros en tiempo real, así como campos para etiquetado propio.
- Entrada MIDI con función MERGE para conectar varias unidades en cascada.
- Fácil conexión a cualquier ordenador o *expansor* MIDI mediante las conexiones MIDI In/Out.

3.7.3.2. Especificaciones.

- Dimensiones: 330 x 100 x 300mm.
- Peso: 2.60Kg.
- Consumo de potencia: 10W aproximadamente.
- Pantalla LED de 4 dígitos y 7 segmentos.
- Interfaz MIDI 5 pines IN, OUT A, OUT B/THRU.

3.7.3.3. Tabla de Implementación MIDI.

Tabla 3.3. Tabla de Implementación MIDI para el modelo BCF2000

Función	Transmitido	Recibido	Comentarios
Basic Default Channel Changed	1 - 16 1 - 16	X	
Mode Default Message Altered	X X 0	X	
Note Number True Voice	0-127 0	X	
Velocity :Note ON :Note OFF	X X	X	
After: Keys Touch: Ch's	X 0	X	
Pitch Bend	X	X	
Control 0 - 119 Change 120 - 127	X X	X X	
Program Change: True Number	0 - 127	X	
System Exclusive	GM, GM2, MMC	Memory Dump	
Song Position Common: Song Select	X X	X	
System : Clock Exclusive: Commands	X X	X	
Aux :Local ON/OFF Messages :All Notes OFF :Active Sense :Reset	X X X X	X	
Notes:		**X = YES**	**0=NO**

3.7.3.4. Requerimientos del Sistema.

- Para PC:
    - Pentium II 300MHz, 128MB en RAM.
    - Windows 98, 2000 or XP
    - Puerto USB.

- Tarjeta de sonido compatible con ASIOo MME.

- Para Mac:

  - MacOS 8.6 o mejor.
  - iMac, G3 o superior con USB.

3.7.3.5. Referencias.

La marca *Behringer* es muy reconocida a nivel mundial por sus productos de alta calidad, soporte técnico y actualizaciones en línea. Algunos de sus usuarios más importantes son los siguientes:

- Mel Gaynor, baterista, cantante, compositor, ha trabajado con Behringer para artistas como Elton John, Robert Palmer y Simple Minds.
- Sammy Peralta, productor de Gloria Estefan, Janet Jackson y Mariah Carey.
- Alice Cooper, cantante y guitarrista.
- Helloween, banda de heavy metal con más de 15 años de trayectoria musical.

3.7.3.6. Costos.

- $279.99 USD.

CAPÍTULO 4

4. DISEÑO DE HARDWARE

4.1. DESCRIPCIÓN GENERAL DE LA MCS

El diseño de la Superficie de Control MIDI se basa en la implementación de los parámetros de audio más comúnmente usados en las estaciones de trabajo de audio digital, como son:

- Volumen.
- Rueda de modulación.
- Mute y Solo.
- Panning.
- Balance de altos y bajos.

Una MCS que pueda controlar estos parámetros estará en capacidad de ejecutar tareas comúnmente realizadas en edición de audio digital. A partir de esta consideración, la MCS se diseñó con 5 controladores de tipo continuo y 2 de tipo switch. Esto debido a que cada controlador podrá ser asignado por el usuario a una tarea específica, es decir, el control para volumen puede ser programado por el usuario para que ejecute el control de filtros o efectos, o el control de bajos puede ser programado para control de volumen del canal. El diseño de la MCS no es un prototipo estático en cuanto a control de parámetros, es un diseño totalmente flexible.

La transmisión de mensajes MIDI es unidireccional desde la MCS a través de un cable MIDI hasta el PC. La visualización de estos mensajes es posible por medio de un LCD.

El diseño del hardware de la consola queda como se muestra en la Figura 4.1:

Figura 4.1. MIDI Control Surface diseñada.

Así, el funcionamiento general de la MCS queda descrito en el siguiente diagrama de bloques (Ver Figura 4.2).

4.2. DIAGRAMA DE BLOQUES

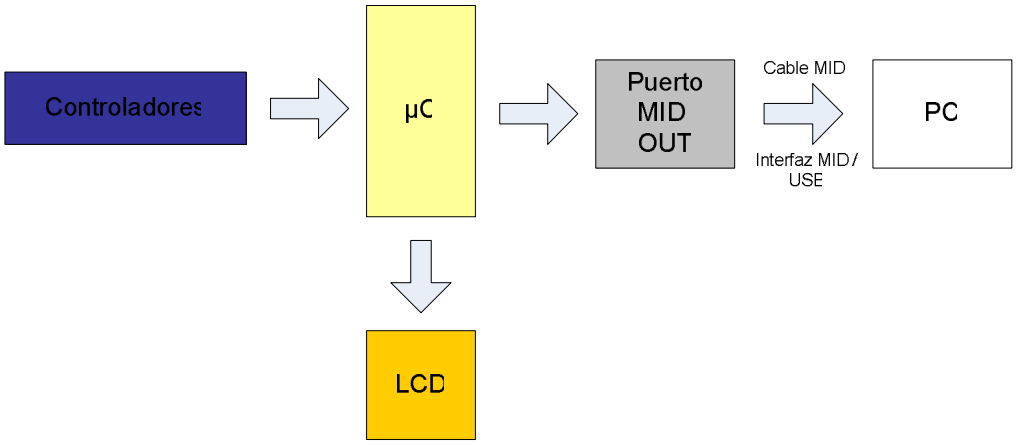


Figura 4.2. Diagrama de bloques del funcionamiento de la MCS.

El cerebro de la MCS corresponde al microcontrolador PIC en el cual se realizan, tanto la adquisición de datos, como el procesamiento de los mismos y su transmisión hacia el PC. Adicionalmente, se encarga de desplegar información necesaria para el usuario en un LCD en tiempo real.

Las entradas del microcontrolador son los valores (analógicos y on – off) generados por los controladores.

El microprocesador procesa los datos de entrada y genera los correspondientes datos digitales de salida hacia el PC (transmisión MIDI) y el LCD.

A continuación se describe cada bloque del funcionamiento de la MCS.

4.2.1. Controladores

El bloque de Controladores representa a todos los controles físicos añadidos a la MCS que desempeñan las funciones de sus correspondientes virtuales en la DAW.

Como se dijo al inicio de este Capítulo, el diseño de la MCS ha sido realizado en base a los principales parámetros de audio utilizados actualmente. En la Figura 4.3. se muestra el bloque de controladores.

Figura 4.3. Bloque de controladores de la MCS diseñada.

4.2.1.1. Volumen.

Este controlador permite cambiar el nivel de volumen que gobierna a todos los canales MIDI, esto es, a toda la estación de audio digital. Es un controlador del tipo continuo[1] y su control es realizado por medio de un potenciómetro lineal que permite desplazarse desde un nivel mínimo (silencio) hasta el nivel máximo de volumen.

[1] Referirse al Capítulo 1 sección 1.2. Generadores y Controladores.

4.2.1.2. Mute y Solo.

Estos controladores son del tipo switch y permiten activar y desactivar los modos Mute (silenciar canal) y Solo (un único canal activo) por medio de dos botones, respectivamente.

4.2.1.3. Panning.

Potenciómetro circular que permite el control del balance estéreo entre los canales derecho e izquierdo del sistema de audio. Su posición central representa un nivel balanceado entre canales, es decir, ambos canales aportan el mismo nivel de sonido. Si es girado en la dirección de las manecillas del reloj un nivel más alto se escuchará sobre el canal derecho. De igual manera, si es girado en dirección contraria a las manecillas del reloj el balance irá hacia el canal izquierdo.

4.2.1.4. Bass y Treble.

Adicionalmente se añaden estos dos controladores que a pesar que no poseen un número de controlador estandarizado, realzan el desempeño de la MCS pues permiten controlar los niveles de bajas y altas frecuencias, enriqueciendo la calidad de sonido. Son dos potenciómetros circulares que al igual que el controlador de *panning*, tienen su posición neutral (balanceado) en la posición central de desplazamiento. A medida de que sean girados en el sentido de las manecillas del reloj, se incrementará los niveles de bajos y altos, respectivamente. Así mismo, si son accionados en dirección contraria a las manecillas del reloj, los niveles de bajos y altos decrecerán.

4.2.1.5. Rueda de modulación.

Como quinto controlador de tipo continuo se implementa otro potenciómetro circular para el control de modulación. Como se dijo en apartados anteriores, cada controlador puede ser configurado según la necesidad del usuario. Por tanto este controlador bien podría servir para el control de *pitch bending* ya que

generalmente estos dos parámetros se encuentran juntos en los dispositivos MIDI como son los teclados o sintetizadores.

4.2.1.6. Circuito de controladores

Todos los controladores de tipo continuo están conectados a la línea de alimentación de 5VDC a través de un divisor de tensión que reduce el voltaje a 2,5VDC. Esto debido a que la resolución de los controladores continuos es de 0 a 127 valores, es decir, 7 bits. La conexión de los controladores continuos queda de la siguiente manera:

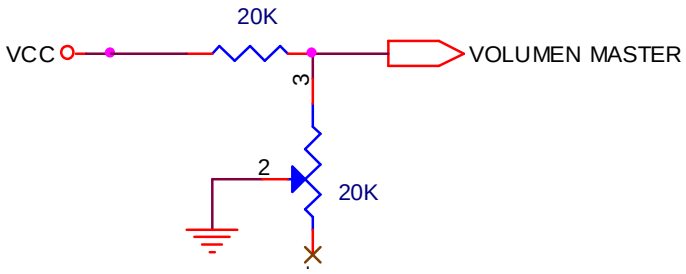


Figura 4.4. Circuito para controladores continuos.

Donde la flecha representa la señal analógica del controlador que está conectada a una entrada analógica del PIC.

Para el caso de los controladores de tipo switch, es decir, Mute y Solo, el circuito diseñado genera una señal de 0VDC para off (controlador desactivado) y de 2,5VDC para on (controlador activado). Esto debido a que para los controladores tipo switch los valores comprendidos entre 0 y 63 se leen como 0 y los valores entre 64 y 127 se leen como 127. Por esta razón estos dos controladores están conectados a dos entradas analógicas más del PIC, la cuales están configuradas como entradas analógicas mas no como digitales, según se verá en el Capítulo 5.

La conexión para los controladores tipo *switch* queda como se muestra en la Figura 4.5:

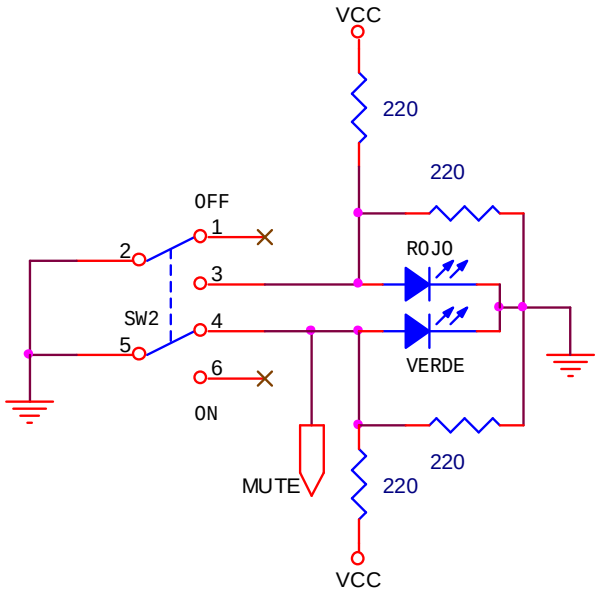


Figura 4.5. Conexión para controladores Mute y Solo.

En la Figura 4.5 se muestra el *switch* de encendido y apagado de los controladores Mute y Solo. Para ello se ha implementado un diodo led bicolor el cual para la posición *off* (controlador desactivado) se enciende de color rojo y para la posición *on* (controlador activado) de color verde. Esto de acuerdo a las características visuales que presentan las MCS más usadas a nivel mundial para encendido y apagado de controladores.

4.2.2. µC

El microprocesador escogido para este proyecto es el PIC16F877A de Microchip debido a que posee las suficientes entradas analógicas que esta aplicación requiere. Además es un microprocesador ampliamente usado y recomendado debido a las características que se resumen en la Tabla 4.1.

Tabla 4.1. Características generales del PIC16F877A

Descripción	PIC16F877A
Frecuencia de operación	20MHz
Resets (and Delays)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory (bytes)	256
Interrupciones	15
Puertos I/O	Puertos A, B, C, D, E
Timers	3
Capture/Compare/PWM modules	2
Comunicaciones seriales	MSSP, USART
Comunicaciones paralelas	PSP
Módulo A/D de 10 bits	8 canales de entrada
Comparadores analógicos	2
Set de instrucciones	35 instrucciones
Presentaciones	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

Como se dijo al inicio de este Capítulo, el microcontrolador se encarga de la adquisición de datos provenientes de los controladores así como del procesamiento de los datos de entrada, la transmisión de los mensajes MIDI hacia el PC y la visualización en el LCD.

Todos los controladores, a excepción de los botones de *Mute* y *Solo*, están conectados a las entradas analógicas (pines del Puerto A) del microcontrolador, pues generan una señal de 0 a 2,5 voltios DC. Estas señales son recogidas por el conversor analógico digital interno del PIC y son transformadas en una señal digital para su procesamiento, es decir, convertidas en mensajes MIDI mediante programación[2]. Se requiere un voltaje de hasta 2,5VDC debido a la resolución de

[2] Referirse al Capítulo 5 sección 5.2.9, Inicialización del Conversor A/D..

7 bits (128 posibles valores) que emplean los controladores continuos (volumen, *panning*, balance, etc.). Una vez transformado el voltaje DC a su valor digital, éste es enviado en el byte de datos del mensaje MIDI correspondiente al controlador.

4.2.3. LCD

Display de cristal líquido (Figura 4.6) de dos líneas 14 caracteres por línea y con luz de fondo. Es usado para la visualización del volumen master del sistema en valores de porcentaje, 0 a 100%, para volumen cero y volumen total, respectivamente, con una resolución de 1%. En la segunda línea se visualiza el canal MIDI por el cual se transmiten los mensajes.

Figura 4.6. Display de cristal líquido usado para el diseño de la MCS.

4.2.4. PUERTO MIDI OUT

Conector hembra tipo DIN de 5 pines 180º para la transmisión de los mensajes MIDI generados por el µC hacia el PC a través de un cable MIDI (cuando se disponga de una tarjeta de sonido con conector DB15) o un interfaz MIDI/USB (cuando se trabaje con una computadora portátil).

Para la transmisión de mensajes MIDI se hace uso del módulo UART del microprocesador ya que el protocolo MIDI no es más que una transmisión serial con características específicas[3].

En la Figura 4.7 se muestra tanto el puerto MIDI OUT como la entrada de voltaje DC de alimentación y el interruptor de encendido principal de la MCS:

Figura 4.7. Vista posterior de la MCS.

4.2.5. CIRCUITO REGULADOR DE VOLTAJE

La MCS implementa un circuito regulador de 5VDC para la alimentación de sus componentes. De esto se encarga el integrado LM7805 el cual recibe como alimentación un voltaje proveniente de un transformador de 120VAC a 12VDC no incluido en el diseño. El circuito se muestra en la Figura 4.8:

[3] Referirse al Capítulo 2 sección 2.3. Especificaciones Técnicas.

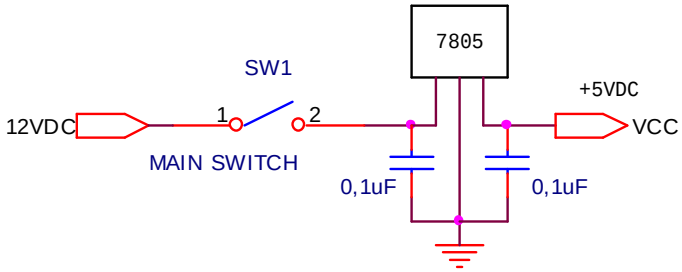


Figura 4.8. Circuito regulador de 5VDC.

El diseño completo del hardware de la MCS se muestra en el Anexo 1, Circuito MCS.

4.2.6. PC

4.2.6.1. Requerimientos de Hardware.

- Tarjeta de sonido de 16-bits compatible con Windows, preferiblemente con drivers DirectX o ASIO, con puerto MIDI de entrada o conector DB15 para Joystick.
- Procesador Intel Pentium III 600 MHz o superior.
- 256 MB en RAM.
- 2 GB de espacio libre en disco.
- Cable de interfaz MIDI/USB (opcional).

4.2.6.2. Requerimientos de Software.

- Tener instalado una Digital Audio Workstation DAW que permita configurar su dispositivo MIDI de entrada como una Superficie de Control MIDI genérica (Reason 3 recomendado).

4.3. ESPECIFICACIONES GENERALES

- **Entradas MIDI:** Ninguna.
- **Salidas MIDI:** Una salida MIDI con conector de 5 pines 180º.
- **Peso:** Aproximadamente 550g.
- **Dimensiones:** 17x15x10cm
- **Velocidad de transmisión:** 31,25Kbaund.
- **Alimentación:** Requiere transformador 120VAC/12VDC.

CAPÍTULO 5

5. DISEÑO DE SOFTWARE

5.1. DESCRIPCIÓN GENERAL

El lenguaje escogido para la programación del microcontrolador es el lenguaje C debido a que éste ha sido utilizado durante los años de estudio universitario. Además su gran aceptación a nivel mundial permite la obtención de librerías, funciones, programas de ejemplo y ayuda en línea.

El compilador a utilizar es el MPLAB IDE de *Microchip*. Es un compilador de fácil aplicación puesto que permite la creación de proyectos con ayuda (*Project Wizard*). Su entorno es amigable y permite la programación en lenguaje C.

La adquisición de datos generados por los controladores se la realiza mediante un *polling*[1] continuo desde el encendido de la MCS. Esto debido a que la velocidad de procesamiento del microcontrolador es suficientemente rápida para no requerir la implementación de interrupciones.

Cada señal de los controladores está conectada a una entrada analógica del puerto A del µC, respectivamente. El µC realiza una lectura de un canal del puerto A, ejecuta la conversión análoga - digital y transmite el dato digital hacia el PC a través del módulo UART en una comunicación serial unidireccional. Esto lo repite

[1] *Polling*, Inglés, Término usado para describir el flujo de programa a manera de bucle y que se repite indefinidamente.

para cada canal del puerto A consiguiendo así una comunicación continua con el PC para cada controlador.

El flujo del programa se puede explicar con el siguiente diagrama:

5.2. DIAGRAMA DE FLUJO

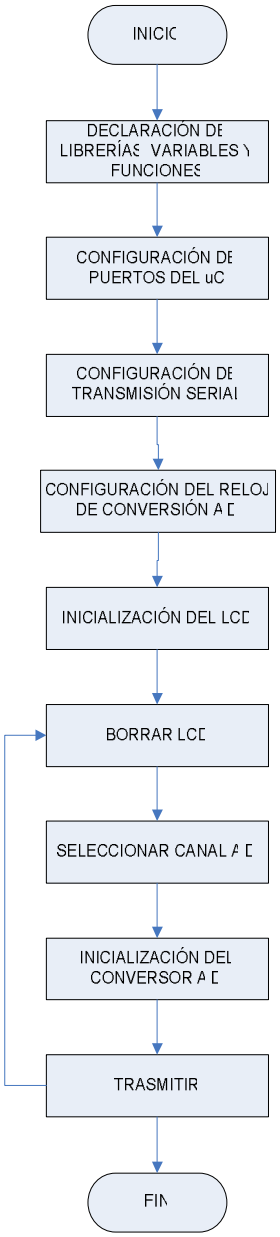


Figura 5.1. Diagrama de flujo del programa

5.2.1. Inicio

El Inicio del programa lo determina el encendido mismo de la MCS, pues como se dijo anteriormente, el programa ejecuta un *polling* continuo desde el encendido hasta el momento que el sistema es apagado.

5.2.2. Declaración de Librerías, Variables y Funciones

En este bloque se inicializan las librerías a utilizarse para el desarrollo del programa. Éstas son: pic168xa.h, stdio.h, delay.h, adc.h, lcd.h y float.h. La declaración de estas librerías queda de la siguiente forma:

```
#include <pic168xa.h>

#include <stdio.h>

#include <delay.h>

#include <adc.h>

#include <lcd.h>

#include <float.h>
```

La librería pic168xa.h contiene todas las funciones necesarias para el desempeño del microcontrolador PIC16F877A y su código se muestra en el Anexo 2. La librería stdio.h (Anexo 3) es añadida para trabajar con las funciones matemáticas y lógicas básicas. Debido a que se utiliza retardos en la transmisión serial también es necesario añadir la librería delay.h (Anexo 4). Para la conversión análoga digital es necesaria la librería adc.h (Anexo 5) y para el manejo del LCD la librería lcd.h (Anexo 6). Además se añade la librería float.h (Anexo 7) para trabajar con números flotantes.

El programa requiere solamente dos variables. Éstas son la variable “volumen” de tipo entero y la variable “dato1” de tipo char. La inicialización de variables queda de la siguiente manera:

```
int volumen;

char dato1[8];
```

Estas variables son utilizadas para la visualización del valor en porcentaje del volumen master en el LCD.

El programa requiere la llamada a tres funciones que se inicializan de la siguiente manera:

```
void selec_Channel(int a,int b,int c);

void init_ADC();

void Tx(int t);
```

5.2.2.1. Función selec_Channel(int a, int b, int c).

Esta función es creada para la selección del canal A/D del puerto A del cual se quiere obtener la lectura. La declaración de esta función es el siguiente:

```
void selec_Channel(int a,int b,int c);
```

Donde, las variables enteras a, b y c, son copiadas en los bits 5 a 3 (CHS2, CHS1 y CHS0) del registro ADCON0 para seleccionar el canal A/D. Por ejemplo, si se quiere leer el canal número 5 del puerto A se llama a esta función de la siguiente manera:

```
selec_Channel(1, 0, 1);
```

De esta manera los bits CHS2, CHS1 y CHS0 recibirán los valores de 1, 0 y 1, respectivamente, asignando así el canal 5 para la lectura del puerto.

5.2.2.2. Función init_ACD().

Para iniciar la conversión analógica digital de la lectura del canal seleccionado con la función anterior, se llama a init_ADC() cuya declaración se presenta a continuación:

```
void init_ADC();
```

5.2.2.3. Función Tx(int t).

Esta función es llamada para realizar la transmisión serial hacia el PC. La declaración de la misma es el siguiente:

```
void Tx(int t);
```

Donde t es una variable tipo entera que envía el número del controlador respectivo al canal analógico. Este valor es necesario ser añadido al mensaje MIDI para que la DAW lo reconozca y sepa qué tipo de controlador es.

5.2.3. Configuración de los puertos del uC

Para la presente aplicación se han utilizado dos de los cinco puertos disponibles en el PIC16F877A. El puerto A es ocupado como entrada para controladores mientras que el puerto B es usado como salida para el LCD. Para esto se requiere inicializar los registros TRISA y TRISB, respectivos.

```
TRISA = 0xFF;                    //Puerto A como entrada

TRISB = 0x00;                    //Puerto B como salida
```

Como se dijo en el Capítulo 4 sección 4.2.1 la MCS dispone de cinco controladores de tipo continuo y dos tipo switch. Como se vio en el Capítulo 2 sección 2.2.5.1, para controladores de baja resolución, es decir, tipo switch, los valores de 0 a 63 se leen como cero (Off) y los valores de 64 a 127 se leen como 127 (On). Por lo tanto se puede configurar los pines del puerto A para que todas las entradas funcionen como analógicas. Para ello es necesario configurar el registro ADCON1 de la siguiente manera:

```
ADCON1 = 0x00;      //AN7:AN0 = Analógicas
```

Con este valor, los pines 0 a 7 del puerto A quedan configurados como analógicos. Además, el valor de la conversión A/D se almacena en el registro ADRESH (justificado a la izquierda) y los 6 bits menos significativos del ADRESL se leen como cero.

5.2.4. Configuración de la transmisión serial

Para la transmisión de los mensajes MIDI se hace uso del módulo USART[2] del uC. Este módulo puede ser configurado en dos modos: Asíncrono y Síncrono. Puesto que la aplicación requiere un control de la tasa de baudios, es decir, debemos configurar la velocidad de transmisión, se requiere que este módulo sea configurado como asíncrono[3]. Para ello se siguen los siguientes pasos:

5.2.4.1. Inicializar el registro SPBRG.

Este registro controla el período de un *timer* interno de 8 bits así como la tasa de baudios cuando USART está configurado como asíncrono. La velocidad de transmisión se calcula mediante la siguiente Ecuación 5.1:

[2] Ver datasheet de PIC 16F87xA, USART (Universal Synchronous Asynchronous Receiver Transmitter) en www.michochip.com.
[3] Referirse al Capítulo 2, sección 2.3.5. Velocidad de transmisión.

$$\text{Tasa de baudios} = F_{osc} / [\, 64 \,(\, X + 1 \,)\,]$$

Ecuación 5.1. Tasa de baudios para transmisión serial.

Donde, F_{osc} es la frecuencia de oscilación del reloj (cristal) y X es un valor entero entre 0 y 255 para el registro SPBRG. De esta manera conociendo la tasa de baudios deseada se puede calcular el valor de SPBRG. Para transmisión MIDI la velocidad debe ser de 31,25 Kbaudios, por tanto, con una F_{osc} de 4MHz y despejando X tenemos:

$$X = \{F_{osc} / [\, 64\ (\text{Tasa de baudios})\ ]\ \} - 1 = \{4\text{MHz} / [\, 64\ (31{,}25\text{Kbaud})\ ]\ \} - 1$$

$$X = 1$$

Así, el valor para el registro SPBRG queda inicializado con 1 para transmisión MIDI.

Según la Tabla 10.3, *Baud rates for asynchronous mode*, que se encuentra en el data sheet del PIC1F877A, podemos establecer que para un valor de 0 en el bit BRGH del registro TXSTA y un valor de 1 en el registro SPBRG obtenemos un error del 0% en el cálculo de la velocidad de transmisión. Estos valores quedan inicializados con las siguientes líneas de programación:

```
SPBRG=1;        //Baud Rate 31.25Kbauds
BRGH=0;         //High Speed Off
```

5.2.4.2. Habilitar el puerto serial.

Dos bits del registro TXSTA son usados para habilitar el puerto serial: el bit SYNC y el bit SPEN. El primero configura el modo de operación del USART. Como la aplicación requiere una transmisión asíncrona el SYNC debe ser encerado. Al setear el bit SPEN se habilita el puerto serial.

```
SYNC=0;          //Modo asíncrono

SPEN=1;          //Habilitar el puerto serial asíncrono
```

Una vez habilitado el puerto serial es necesario establecer que la transmisión debe ser de 8 bits según lo visto en el Capítulo 2 sección 2.5.3. Velocidad de transmisión. El bit TX9 del registro TXSTA debe entonces ser encerado:

```
TX9=0;           //Transmisión de 8 bits
```

5.2.5. Configuración del reloj de conversión A/D

El tiempo de conversión por bit es llamado T_{AD}. Una conversión A/D requiere un mínimo de $12T_{AD}$ por conversión de 10 bits. La fuente del reloj de conversión A/D es seleccionada por software. Para una conversión A/D correcta es necesario seleccionar un reloj de conversión que asegure un tiempo T_{AD} mínimo de 1,6us. Según la Tabla 11.1 del *datasheet* del PIC, el oscilador interno RC del microcontrolador tiene un T_{AD} típico de 4us. Por tanto, el oscilador RC interno del microcontrolador asegura que la adquisición de datos sea correcta.

5.2.6. Inicialización del LCD

El código que permite el manejo del LCD se encuentra en el archivo lcd.c incluido en el proyecto del programa y que se puede encontrar en el Anexo 8. Para la inicialización del LCD se hace uso de la función lcd_init(), la cual configura el LCD para iniciar su operación, es decir, ejecuta la subrutina adecuada que establece el modo de operación (para la presente aplicación el modo usado es de 4 bits), el tamaño de carácter (5x8 puntos), el número de líneas (2 líneas) y el modo del cursor (tipo *blink*).

5.2.7. Borrar LCD

Una vez inicializadas todas las variables y funciones, y establecidas las configuraciones de puertos, se tiene todo lo necesario para iniciar el bucle de programación que permite la adquisición, conversión y transmisión de datos. Como primer paso se borra el LCD para asegurar que ningún dato no deseado se despliegue. Esto se consigue con la función lcd_clear () del archivo lcd.c.

5.2.8. Seccionar el canal A/D

A continuación se selecciona el canal A/D del cual se realiza la adquisición de datos. Los seis pines utilizados para la entrada de controladores son leídos en forma ascendente, es decir, el primer canal leído es el RA0 y el último es el RA7 (los canales RA4 y RA5 no son utilizados).

Los bits que permiten la selección del canal A/D son CHS2, CHS1 y CHS0 del registro ADCON0, de acuerdo a la Tabla 5.1.

Tabla 5.1. Bits de selección de canal A/D

CHS2	CHS1	CHS0	Canal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

La selección del canal es realizada mediante la función selec_Channel(int a, int b, int c), la cual envía tres valores enteros a, b y c que corresponden a los bits de selección de canal CHS2, CHS1 y CHS0, respectivamente. Por ejemplo, si queremos seleccionar el canal 3 lo haremos de la siguiente forma:

```
selec_Channel(0,1,1);
```

5.2.9. Inicialización del conversor A/D

Para realizar la conversión A/D es necesario primeramente encender el módulo A/D. Esto se consigue seteando el bit ADON del registro ADCON0.

Después, se requiere que transcurra el tiempo de adquisición T_{ACQ} calculado a partir de la Ecuación 11.1 *Acquisition Time* que se encuentra en el *data sheet* sección 11.1. Para esta aplicación en la que se utiliza potenciómetros de 20KΩ, el T_{ACQ} es de 25,62us. Por tanto, un retardo de 1 ms es suficiente.

A continuación se debe iniciar la conversión A/D seteando el bit GO/DONE del registro ADCON0. Cuando termina la conversión este bit es encerado automáticamente. Se espera entonces la conversión mediante un *polling* en el que se evalúa el valor de este bit. El código queda de la siguiente manera:

```
void init_ADC()

{

    ADON=1;          //Se enciende el módulo A/D.

    Delay(1);        //Se espera el tiempo de adquisición.

    ADGO=1;          //Inicia la conversión.

    while(ADGO)      //Polling de espera de conversión.

        continue;}
```

5.2.10. Transmisión

Primeramente se setea el bit 5, TXEN, del registro TXSTA para habilitar la transmisión serial. El mensaje MIDI que se transmite es el *Control Change*

Message, por lo tanto requiere de 3 bytes[4]. El registro TXREG es cargado con estos bytes para ejecutar la transmisión. El primero es el Status Byte, donde, además del *nibble* de comando se envía el número del canal MIDI al cual se va a transmitir. La presente aplicación hará uso únicamente del canal MIDI No. 1. El segundo byte enviado es el Data Byte 1 el cual lleva el número del controlador, es decir, el valor enviado a la función en la variable entera t. Finalmente, se transmite el valor de la conversión almacenado en ADRESH en el segundo byte de datos. Se añade una línea de retardo de 1 milisegundos entre cada byte enviado para asegurar que la transmisión sea recibida.

Así, el código que ejecuta la transmisión del mensaje MIDI es el siguiente:

```
void Tx(int t)

{

    TXEN=1;              //Habilitar transmisión

    TXREG=0xB0;          //Status Byte: Control Change Message Ch1

    DelayMs(1);          //Retardo

    TXREG=t;             //Data Byte #1: t, número de Controlador

    DelayMs(1);          //Retardo

    TXREG=(ADRESH);      //Data Byte #2: Valor de la conversión A/D

    DelayMs(1);}         //Retardo
```

El mensaje MIDI queda configurado de la siguiente manera:

[4] Referirse al Capítulo 2, Tabla 2.5. Channel Voice Messages.

Tabla 5.2. Mensaje MIDI a transmitir

Status Byte									Data Byte 1								Data Byte 2							
Comando				Canal #					Controlador #								Valor del controlador							
1	0	1	1	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				a					a								a							
				1	1	1	1		0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
TXREG = B0;									TXREG = t;								TXREG = ADRESH;							

5.3. PROTOCOLO MIDI

De esta manera, la transmisión de mensajes MIDI queda definida de acuerdo a las especificaciones del protocolo, esto es, transmisión a 31,25Kbauds (unidireccional para esta aplicación), asíncrona, con un bit de inicio, 8 bits de datos y un bit de parada, conformando así bytes de 10 bits; para un único canal MIDI (canal 1), es decir, en modo Onmi off/Poly[5] y de forma continua debido al modo *polling* de programación.

5.4. INTERFAZ VISUAL CON LCD

Como se dijo en el Capítulo 4 sección 4.2.3, el LCD es usado para la visualización del volumen master del sistema y del canal MIDI por el cual se transmiten los mensajes. El valor que genera la conversión A/D para volumen master es transformada a un valor en porcentaje, es decir, de 0 a 100% de volumen para un valor de conversión de 0 (volumen mínimo) y de 127 (volumen máximo), respectivamente. Para esto se encontró la curva Volumen[%] vs. Valor ADC en base a la adquisición de varios valores generados por la conversión análoga/digital de la señal del controlador de volumen y sus respectivos valores de voltaje (Tabla 5.3).

[5] Referirse al Capítulo 2, sección 2.2.4.3.

Tabla 5.3. Valores de voltaje medidos para varios valores de conversión ADC.

Valor ADC	Voltaje [V]	Volumen [%]
0	0,01	0,00
10	0,24	9,63
20	0,41	16,80
30	0,60	24,51
40	0,80	32,62
50	1,00	40,82
60	1,18	48,24
70	1,37	55,94
80	1,56	63,81
90	1,75	71,72
100	1,96	80,41
110	2,15	87,91
120	2,34	95,98
124	2,44	100,00

Los valores de Volumen se calculan con una simple regla de tres con el valor de 2,44VDC como 100% (volumen máximo). Con estos datos podemos graficar la curva y obtener la ecuación de la misma (Figura 5.2)

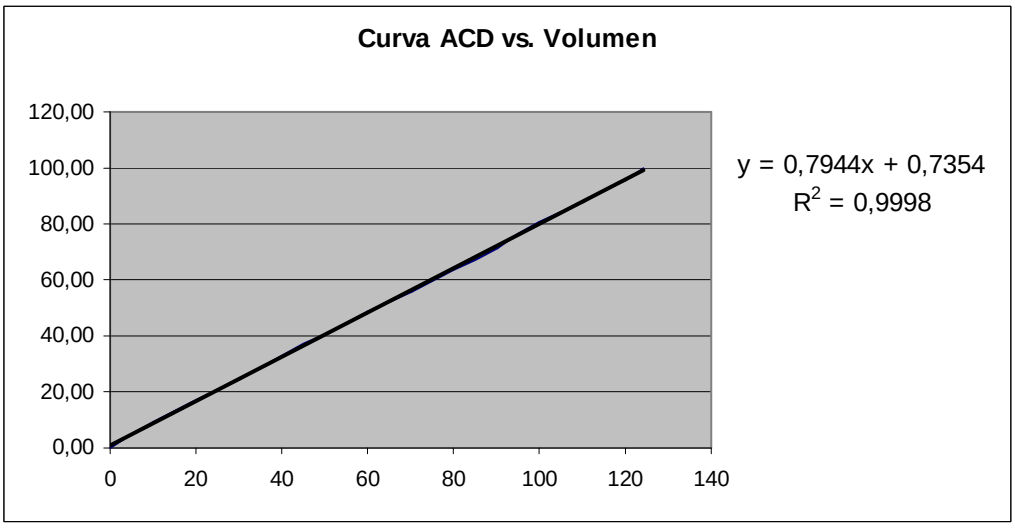


Figura 5.2. Curva ADC vs. Volumen.

Calculando la línea de tendencia obtenemos la ecuación de la curva (Ecuación 5.2) y el coeficiente de correlación R^2 que nos garantiza que la línea de tendencia es apropiada (R^2>0,95).

$$volumen = (0.7944*ADRESH+0.7354);$$

Ecuación 5.2. Porcentaje de volumen master.

Donde, ADRESH contiene el valor de la conversión análoga digital. Este valor de volumen es el que se despliega en el LCD en valor de porcentaje. Para ello se escribió el siguiente código:

```
lcd_goto(0x00);                          //Cursor en la posición 0,0.
sprintf(dato1, "Volumen: %d", volumen);  //Impresión de “Volumen:”.
lcd_puts("dato1");                       //Impresión de la variable tipo char dato1.
lcd_puts("%");                           //que contiene el valor de Volumen.
lcd_goto(0x40);                          //Salto a segunda línea.
lcd_puts("Channel 1");                   //Impresión de “Channel 1”.
```

Con esto, el dato de volumen es desplegado en el LCD cada vez que se repite el set de instrucciones de adquisición, conversión y transmisión de datos. Debido a la velocidad de procesamiento del uC el despliegue de datos en el LCD es continuo (Figura 5).

Figura 5.3. Ejemplo de despliegue del Volumen en LCD.

De esta manera la programación del PIC queda completada para iniciar la transmisión de mensajes MIDI. El código de la programación total se muestra en el Anexo 9, Código de Programación.

CAPÍTULO 6

6. GUÍA DE USUARIO Y TUTORIAL

6.1. GUIA DE USUARIO

6.1.1. Introducción

Esta guía pretende ayudarle a obtener un inicio rápido en el uso de la MCS. Para conseguir un óptimo uso de la misma sírvase leer detenidamente esta guía antes de empezar a utilizarla por primera vez.

La MCS es una solución compacta y versátil para aquellos usuarios que inician su carrera en tareas de edición de audio digital o para usuarios caseros que no requieren de mayor sofisticación. La Superficie está provista de los controles necesarios para desempeñar funciones básicas de edición y que le serán de mucha ayuda cuando dispone solamente de un ratón y un teclado en su DAW (Digital Audio Workstation).

Usted puede utilizar la MCS con otro software o hardware MIDI también. La MCS envía datos que pueden ser recibidos por cualquier otro dispositivo que disponga de un interfaz MIDI. Así la MCS puede ayudarle a controlar un estudio completo.

6.1.2. Características

6.1.2.1. Puertos:

- MIDI OUT.

6.1.2.2. Alimentación:

- 12VDC.

6.1.2.3. Controles:

- 4 controles programables* tipo knob.
- 1 control programable* tipo fader.
- 2 botones programables* on/off.

6.1.2.4. Datos MIDI de controles:

- Números de controladores MIDI.

6.1.2.5. Datos MIDI de botones:

- Note on*.
- Mute y Solo.

* Programable por software.

6.1.2.6. Características útiles:

- Todos los controles son completamente programables para cualquier número de controlador MIDI.

6.1.3. Conociendo la MCS

6.1.4. Antes de conectar la MCS

Antes de conectar y encender la MCS a su dispositivo MIDI o PC es importante tener en cuenta las recomendaciones que se presentan a continuación. Asegúrese que el botón de on/off que se encuentra en la parte posterior esté en la posición off. Si dispone de un cable de interfaz MIDI/USB como el que se muestra en la Figura 6.1 conéctelo ahora a su PC.

Figura 6.1. Cable de interfaz MIDI/USB marca Roland modelo Edirol UM-1EX.

Si el cable adaptador no es reconocido por su PC seguramente dispone de un software de instalación. Instálelo ahora. Para la mayoría de los casos, el mismo cable dispone de los *drivers* necesarios para su inmediato funcionamiento. Si aún tiene problemas con la instalación, consulte el manual de usuario provisto con el cable.

Conecte un extremo del cable MIDI al puerto MIDI OUT de la MCS y el otro extremo al puerto MIDI IN de su tarjeta de sonido o cable MIDI/USB. Ahora tiene

todas las conexiones necesarias para iniciar su desempeño con la MCS. Sus conexiones deberían ser semejantes a las que se muestran en la Figura 6.2 y 6.3.

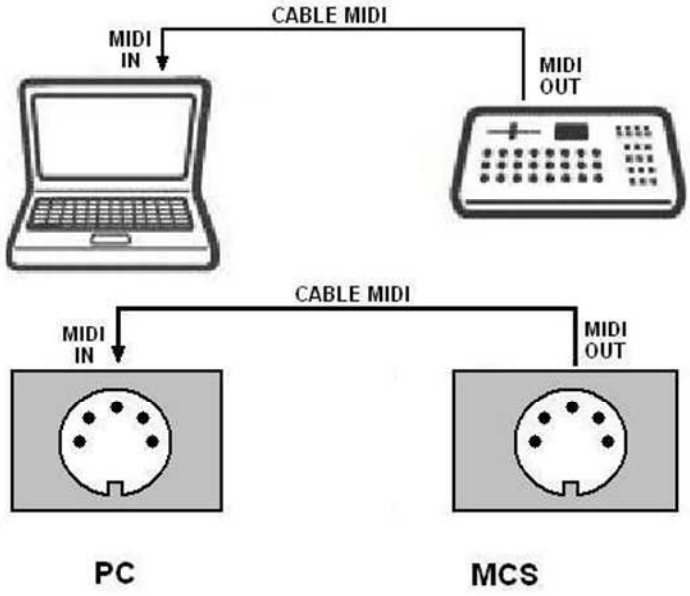


Figura 6.2. Diagrama de conexión de la MCS con cable MIDI directo.

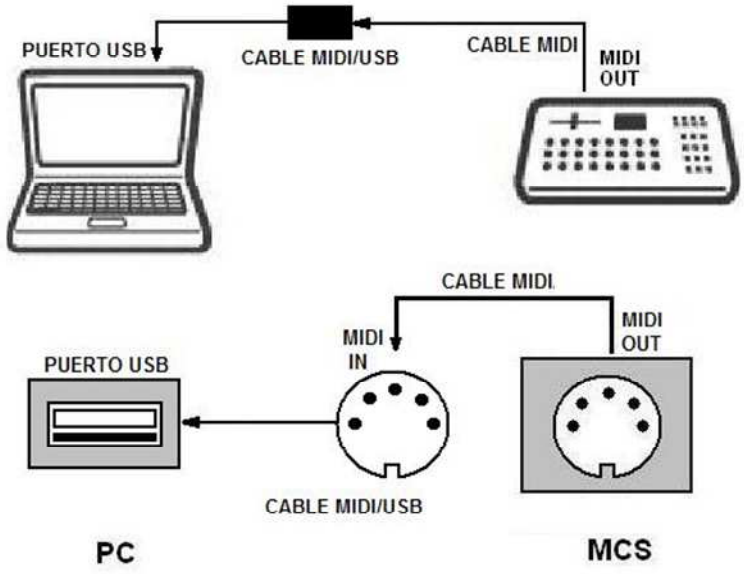


Figura 6.3. Diagrama de conexión de la MCS con cable MIDI/USB.

Puede también conectar la MCS a un sintetizador que posea un puerto de entrada MIDI. Simplemente conecte un extremo del cable MIDI al puerto de salida

de la MCS y el otro al puerto MIDI IN del sintetizador tal como se muestra en la Figura 6.4.

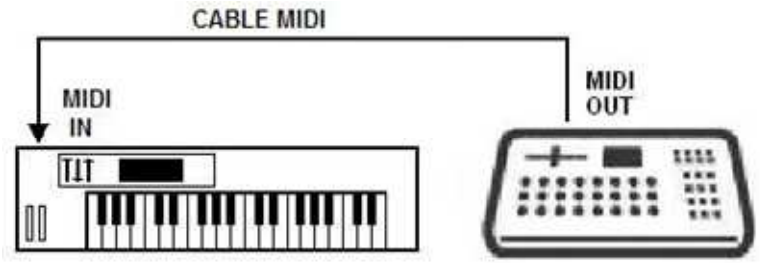


Figura 6.4. Conexión de la MCS a un sintetizador.

6.2. TUTORIAL

Como se ha dicho en apartados anteriores la Superficie de Control MIDI puede trabajar con cualquier dispositivo que soporte el protocolo y que posea un puerto MIDI de entrada. En el presente tutorial veremos el manejo de la MCS sobre dos de las aplicaciones más comúnmente usadas.

6.2.1. MCS sobre DAW

Cuando trabaje sobre un software de audio digital podrá sacar mayor utilidad a la MCS. Programas como el *Reason* de *Propellerheads* o *CakeWalk Home Studio* hacen un gran uso de las MCS dentro de sus aplicaciones. El manejo de consolas de mezcla, secuenciadotes, sintetizadores digitales, máquinas de ritmo y demás, resulta sencillo y práctico cuando una MCS es configurada para trabajar con sus controles. Asegúrese que su DAW sea capaz de soportar MCS de tipo general, es decir, dispositivos MIDI que se conectan a su PC a través de un cable MIDI o cable MIDI/USB sin marca o modelo específicos.

La asignación de controles es realizada por medio del mismo software de audio digital. He aquí un ejemplo de cómo debe configurar la MCS sobre *Reason 3*:

- Asegúrese que todas las conexiones son correctas de acuerdo a lo visto en la sección 6.1.4. Sus conexiones deberían semejarse a las que se muestran en la Figura 6.5:

Figura 6.5. Conexión con cable adaptador MIDI/USB.

- Encienda la MCS conmutando el interruptor principal a la posición de encendido.

- Inicie Reason 3. Si está utilizando un cable MIDI/USB el modelo del mismo aparecerá en la consola de entradas MIDI tal como se muestra en la Figura 6.6. Esto quiere decir que Reason ha reconocido el cable y está listo para recibir mensajes MIDI. Si utiliza un cable MIDI convencional conectado al puerto MIDI IN de su tarjeta de sonido deberá seleccionarla de una lista desplegable en el menú *Edit -> Preferentes -> Page*: *Audio -> Audio Card Driver*.

Figura 6.6. Consola de dispositivos MIDI para Reason 3.

- Una vez que Reason ha reconocido su cable MID/USB o su tarjeta de sonido, es momento de configurar la MCS. En el menú *Edit* -> *Preferentes*-> *Page*: *Control Surfaces and Keyboards*, haga clic en el botón *Add*. Se abrirá una ventana como la que aparece en la Figura 6.7.

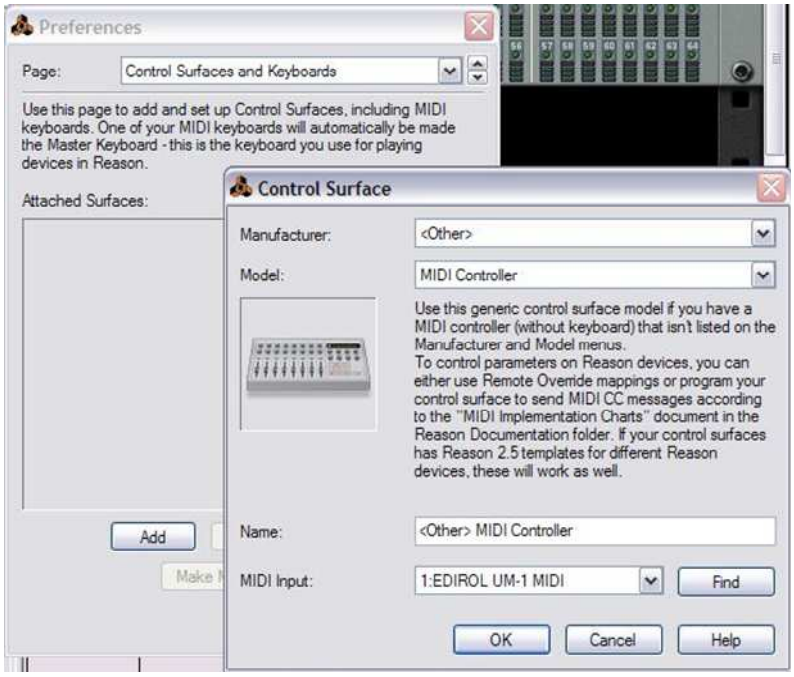


Figura 6.7. Ventana de selección de la MCS.

- En la cual deberá escoger como fabricante (*Manufacturer*) la opción *Other*, como modelo la opción *MIDI Controller* y como entrada MIDI la opción que su cable MIDI/USB le proporcione, en este caso, EDIROL UM-1 MIDI.

- Presione el botón OK. En la ventana anterior observará que aparece una ilustración que indica que una superficie de control ha sido configurada. Asegúrese que el cuadro de Usar con Reason esté marcado.

Figura 6.8. Ventana de selección activa de MCS's y keyboards.

- Cierre esta ventana. La MCS ha sido seleccionada y configurada para trabajar con Reason 3.

- A continuación haremos una prueba de funcionamiento. Primero crearemos un rack de consolas virtuales. En el menú *Create* seleccione la opción *Line Mixer* 6:2. Con esto tenemos un mezclador virtual de seis canales a la cual irán conectadas otras consolas. Nuevamente en el menú *Create* seleccione ahora un sintetizador, por ejemplo, *Malstrom Graintable Synthesizer*. Su rack de consolas debe asemejarse al de la Figura 6.9.

Figura 6.9. Rack de consolas virtual con una mezcladora y un sintetizador.

- Es momento de asignar los controles de la MCS. Haga clic derecho sobre el control de Master Volumen en el mezclador. Selecciones la opción *Edit Remote Override Mapping*. Se abrirá una ventana como la que aparece en la Figura 6.10.

Figura 6.10. Edit Override Mapping.

- Esta ventana nos permite seleccionar el número de controlador que queremos asignar al control de master volumen. Según lo visto en el Capítulo 2 Tabla 2.6 el ID Number para Main Volumen es el número 7. Seleccionamos entonces

primeramente la superficie de control <*Other*> *MIDI Controller* y en la lista desplegable de Control seleccionamos el número CC07. Presione OK. Con esto hemos asignado el número CC07 al control de volumen master del mezclador.

- Ahora ya puede controlar el volumen master desde la MCS en tiempo real.

- El mismo procedimiento puede hacerlo para los otros controles de la MCS. Consulte la Tabla de Implementación MIDI al final de este capítulo para conocer los Números de Controlador que maneja la MCS.

- Recuerde que cuando trabaje con una DAW no necesariamente está obligado a asignar cada control a su respectivo número de controlador. Los programas de audio digital como Reason 3 le permiten asignar cualquier número de controlador a cualquier control virtual. El programa solo requiere reconocer una entrada de datos MIDI para ejecutar un control. No así cuando trabaje con un dispositivo no programable como un teclado o sintetizador, el cual sólo reconoce qué tipo de mensaje y qué número de controlador está recibiendo, y ejecuta el control respectivo.

- En la Tabla 6.1 usted podrá encontrar las posibles funciones que la MCS dispone, así como los números de controladores que cumplen las mismas.

6.3. TABLA DE IMPLEMENTACIÓN MIDI

Tabla 6.1. Tabla de Implementación MIDI para la MCS

Función	Transmite	Recibe	Comentarios
Basic :Default Channel:	1	X	Channel 1
:Default Mode :Message :Altered	3 Omni off/Poly X	X	
Note Number: True Voice	X	X	
Velocity :Note ON :Note OFF	X X	X	
After: Keys Touch: Ch's	X X	X	
Pitch Bend	0	X	
1 7 10 Control 16 Change 17 18 80 81	0 0 0 0 0 0 0 0	X X X X X X X X	Es resto de controladores se marcan como X.
Program Change: True Number	X X	X	
System Exclusive	X	X	
Song Position Common: Song Select	X X	X	
System : Clock Exclusive: Commands	X X	X	
Aux :Local ON/OFF Messages :All Notes OFF :Active Sense :Reset	X X X X	X	
Notas:		**0 = SI**	**X=NO**

CAPITULO 7

7. EVALUACIÓN DE LA MCS

7.1. PRUEBAS SOBRE REASON 3

La Superficie de Control MIDI fue evaluada sobre el programa de audio digital *Reason 3* de *Propellerhead* debido a su gran aceptación a nivel mundial[1].

Una vez realizadas todas las conexiones y configuraciones necesarias vistas en el capítulo anterior se realizaron las siguientes pruebas.

7.1.1. Control de Volumen

En Reason fue creado un mezclador 14:2 (14 canales de entrada estéreo combinadas y direccionadas a las salidas maestras derecha e izquierda). El control de volumen master del mezclador fue asignado al número de controlador CC07 de la MCS tal como se muestra en la Figura 7.1.

[1] Referirse al Capítulo 2 sección 2.4.1.

Figura 7.1. Modo Edit Override Mapping que muestra las asignaciones de CC's.

Donde, en el Modo de Edición de Mapeo Remoto, se puede visualizar las asignaciones de controles activas. Para esta prueba el control virtual de volumen master del mezclador muestra un rayo amarillo, indicativo de que este control está "mapeado" al CC07 del dispositivo <*Other*> *MIDI Controller*, tal como lo muestra la etiqueta del mismo color. Las flechas de color azul indican que el control respectivo no está asignado a algún número de controlador.

Ahora se tiene el control táctil del control *Volumen Master* del mezclador correspondiente al *fader* del la MCS como se muestra en la Figura 7.2:

Figura 7.2. Prueba de control de Volumen.

7.1.2. Control de Panning, Bass, Treble y Rueda de Modulación

Ahora se asignan los cuatro controles continuos restantes. El número de controlador para *Panning* por defecto es el CC10 y para modulación es el CC01. Mientras que para *Bass* y *Treble* no existen números estandarizados; por lo tanto, se asignan números de controlador conocidos como de Propósito General, es decir, números que pueden ser asignados a cualquier función de controles según la Tabla 2.6 vista en el Capítulo 2.

Así, se asignan los números de controlador de propósito general CC15 y CC16 para *Bass* y *Treble*, respectivamente, y CC01 para la rueda de modulación, quedando el mapeo de CC's como muestra la Figura 7.2.

Figura 7.3. Mapeo para volumen master, panning, bass y treble del mezcaldor 14:2.

Con esto se tiene ya la asignación de los controles continuos de la MCS y es posible controlar los parámetros asignados desde la MCS de manera continua uno por uno o varios a la vez tal como se muestra en la Figura 7.4:

Figura 7.4. Prueba de control de Treble.

7.1.3. Control de Mute y Solo

Para estos dos parámetros no existen números de controlador establecidos. Se asignan, de igual forma como se lo hizo con los controles de *bass* y *treble*, dos números de propósito general, CC80 y CC81, números para controles de baja resolución, es decir, de 7 bits. El mapeo queda de la siguiente forma:

Figura 7.5. Mapeo para Mute y Solo del canal 1 del mezclador 14:2.

De esta manera todos los controles de la MCS han sido asignados a sus respectivos controles en la DAW.

Figura 7.6. Prueba de control de Mute.

7.1.4. Resultados

Luego de las pruebas realizadas sobre la plataforma Reason 3 se han obtenido los siguientes resultados.

- Todos los controladores en la MCS responden de forma sincronizada con sus respectivos comandos en la DAW.

- La velocidad de adquisición de datos de los controladores es adecuada para la aplicación.

- El mapeo de controladores no presenta ningún inconveniente.

- Todos los controladores pueden ser asignados satisfactoriamente.

7.2. PRUEBAS SOBRE EL SINTERIZADOR YAMAHA PSS-790

La Superficie de Control MIDI puede ser conectada a cualquier dispositivo que sea capaz de recibir mensajes MIDI. Para esta prueba se utiliza el sintetizador marca YAMAHA modelo PSS-790 el cual posee los puertos MIDI IN, OUT y THRU. Se realizan las conexiones necesarias tal como se muestra en la Figura 7.1 utilizando un cable MIDI.

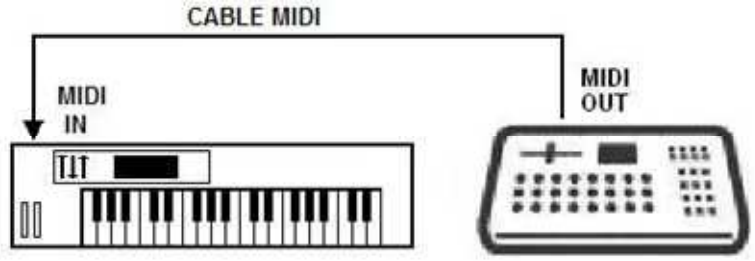


Figura 7.4. Conexión de la MCS a un sintetizador.

Antes de iniciar la transmisión se debe seleccionar en el sintetizador el canal en el cual se va a recibir los mensajes MIDI, es decir, seleccionamos el canal 1 para recepción tal como se muestra en la Figura 7.5.

Figura 7.5. Configuración MIDI en el sintetizador YAMAHA modelo PSS-790.

Con los botones + y – colocados debajo del *Multi Display* se selecciona el canal 1. Se enciende la recepción de mensajes MIDI con el botón triangular verde correspondiente a *Receive Ch/Clock* en la parte derecha de la Figura 7.5. Con esto el sintetizador está listo para recibir los mensajes MIDI que la MCS enviará a través del cable MIDI.

El sintetizador YAMAHA modelo PSS-790 es capaz de recibir los mensajes MIDI correspondientes a Volumen (CC 07) y Modulación (CC 01) provenientes de la MCS, pues estos dos números de controlador son estándares que la MCS transmite y que pueden ser recibidos por cualquier dispositivo MIDI. El resto de controladores al ser de propósito general el sintetizador no los reconoce y por tanto no ejecuta acción alguna cuando se cambia el valor de los mismos (*panning, bass, treble, pitch bending, mute* y *solo*). La Tabla de Implementación MIDI del sintetizador mostraría los números de controlador usados pero, lamentablemente, debido a que este modelo es antiguo, su Tabla de Implementación no ha sido guardada y no ha sido encontrada en Internet.

La MCS es encendida y está enviando información hacia el sintetizador.

7.2.1. Resultados

- Sólo dos de los controladores (volumen y modulación) disponibles en la MCS son reconocidos por el sintetizador.

- El nivel de volumen depende de la posición en la que se encuentre el control de volumen en el sintetizador, es decir, el control de volumen ejecutado desde la MCS abarca los valores comprendidos entre cero (posición inferior) hasta el valor máximo que el control de volumen en el sintetizador se encuentre (posición máxima).

- El control de volumen y modulación se ejecuta de forma sincronizada y sin inconvenientes.

CAPITULO 8

8. CONCLUSIONES Y RECOMENDACIONES

8.1. CONCLUSIONES

- El diseño y construcción de la Superficie de Control MIDI cumple con los objetivos planteados al inicio de la realización de este proyecto pues fue posible la transmisión de mensajes MIDI desde la MCS hacia otro dispositivo MIDI.

- La Superficie de Control supera la expectativas iniciales de transmitir un solo mensaje MIDI ya que la Superficie de Control construida permite la transmisión de siete mensajes MIDI de forma simultánea.

- La MCS dispone de siete controladores: cinco de tipo continuo (Volumen Master, Modulación, *Panning*, *Bass* y *Treble*) y dos controladores de tipo *switch* (*Mute* y *Solo*).

- La velocidad de transmisión permite un control en tiempo real de los controladores asignados en la plataforma de audio digital (DAW) o en otro dispositivo MIDI, permitiendo así un ajuste dinámico y tangible de los parámetros de audio usados.

- La configuración de la MCS sobre una DAW resulta sencilla tal como se indicó en el Capítulo 6 sección 6.2. Una vez configurada la MCS es un dispositivo Plug and Play que no requiere ser configurado nuevamente, a menos que se

trabaje sobre una nueva plataforma de audio. Para el caso de conexión con otros dispositivos MIDI como son los sintetizadores, la MCS sólo requiere ser conectada al dispositivo y encendida para iniciar la transmisión.

- La MCS puede ser conectada a cualquier dispositivo MIDI que sea capaz de recibir mensajes MIDI bajo los estándares que se establecen en el protocolo de comunicación MIDI y que se han descrito el Capítulo 2. Esto permite que la MCS desempeñe funciones diversas en el amplio mundo del audio digital, como son funciones de masterización, edición, grabación, etc.

- La MCS es capaz de formar parte de una red de instrumentos y dispositivos MIDI con el fin de controlar varios equipos simultáneamente a través de una salida THRU de otro dispositivo.

- La transmisión de mensajes MIDI se la realiza a través de un cable MIDI convencional para el caso de conexiones con dispositivos que dispongan de puertos MIDI IN, y a por medio de un cable MIDI/USB cuando la MCS sea conectada a una computadora.

- El canal único de transmisión es el número uno (*channel 1*) para todos los controladores.

- El Protocolo MIDI ha demostrado ser un lenguaje de comunicación musical muy versátil y funcional, de ahí que ha sido y continúa siendo utilizado a nivel mundial, casi sin variaciones desde su origen, en muchas y variadas aplicaciones no solo musicales sino de control.

- La MCS que fue diseñada es un prototipo compacto y práctico pues ocupa poco espacio y consume muy poca energía, lo cual permite ser llevado a cualquier parte donde se la requiera. Las funciones que otra superficie de control del mercado actual puede disponer también se las puede encontrar en

la MCS a menor escala. Los controladores utilizados se asemejan a los que se puede encontrar en superficies de control de marca registrada.

8.2. RECOMENDACIONES

- Debido a que el propósito del presente proyecto fue la construcción de un prototipo de superficie de control MIDI, el diseño de la MCS resulta un tanto limitado en cuanto a la capacidad de transmisión de mensajes MIDI. El protocolo permite la transmisión a través de 16 canales de los cuales solo el canal 1 está siendo utilizado por la MCS. Por tanto, se recomienda para futuros proyectos relacionados, que la transmisión de mensajes sea utilizando la totalidad de los canales MIDI y así obtener todas las ventajas que esto implica, como es, por ejemplo, el control de una orquesta completa.

- Se recomienda además que los futuros proyectos implementen la recepción de mensajes MIDI con la utilización de potenciómetros motorizados, lo cual permitirá una comunicación bidireccional entre la MCS y el dispositivo MIDI.

- Se puede implementar potenciómetros de mejor calidad para evitar desgastes prematuros en el desplazamiento de los mismos. Esto permite mantener los valores de impedancia constantes si los potenciómetros no están siendo desplazados, evitando así cambios repentinos entre valores adyacentes de impedancia.

- Es posible también que se incluya en el diseño circuitos de eliminación de ruido, protección y pulsadores antirebote.

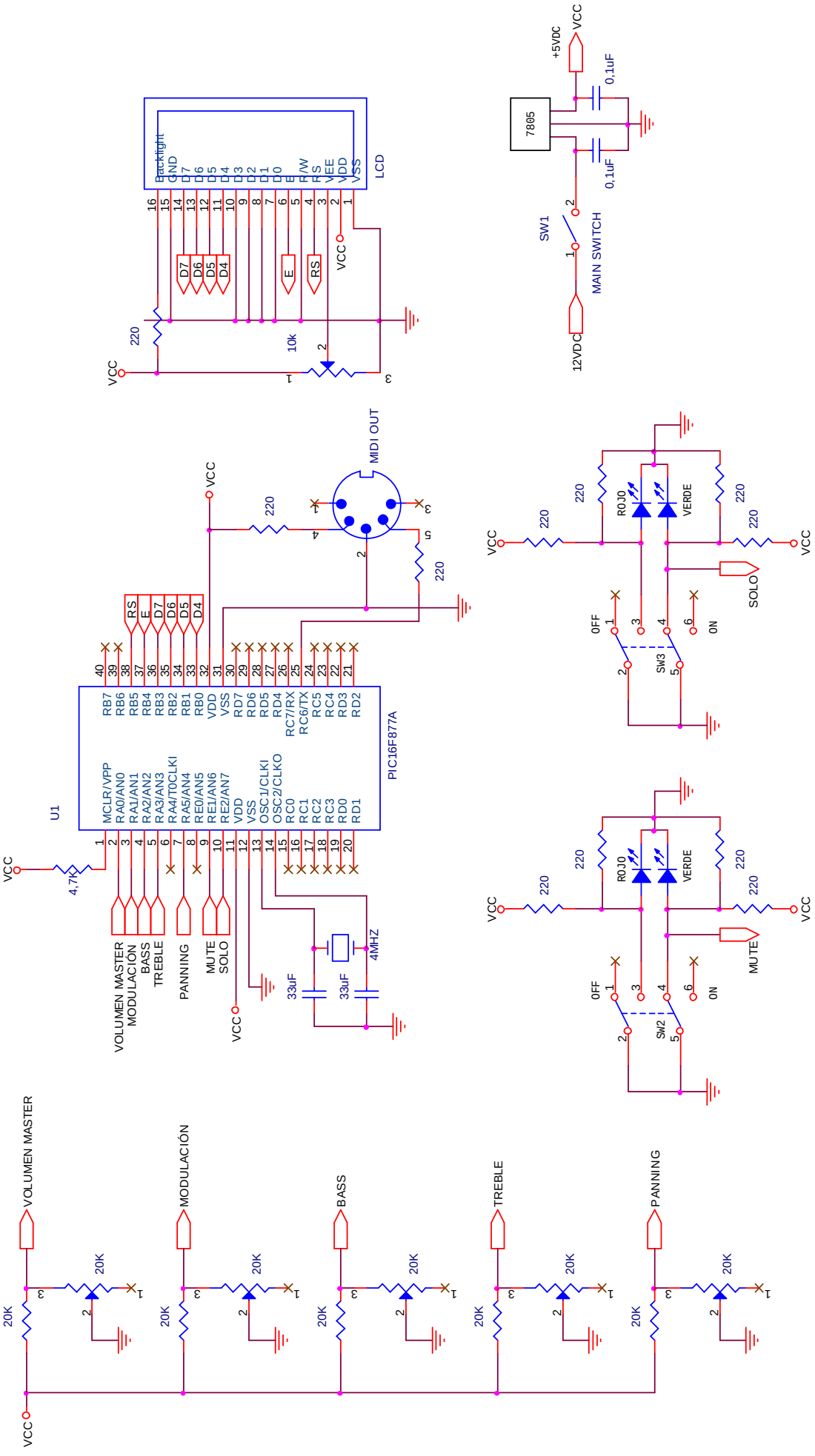
- Para la conexión con otros dispositivos MIDI se recomienda utilizar cables MIDI cortos para evitar pérdidas en la transmisión (máximo 15 metros) y enlazar hasta 3 dispositivos en serie.

- Se debe tomar en cuenta que la plataforma de audio digital permita la configuración de superficies de control MIDI de tipo general y no solamente de superficies existentes en el mercado.

- Es importante consultar las tablas de implementación MIDI antes de conectar la MCS a dispositivos externos para cerciorarse de qué mensajes está en capacidad de recibir.

ANEXOS

Anexo 1, Circuito MCS.


LCD
Backlight
GND
D7
D6
D5
D4
D3
D2
D1
D0
E
R/W
RS
VEE
VDD
VSS
VCC
220
10k
+5VDC
7805
0,1uF
SW1
MAIN SWITCH
12VDC
MIDI OUT
U1
PIC16F877A
MCLR/VPP
RA0/AN0
RA1/AN1
RA2/AN2
RA3/AN3
RA4/T0CLKI
RA5/AN4
RE0/AN5
RE1/AN6
RE2/AN7
OSC1/CLKI
OSC2/CLKO
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
RC7/RX
RC6/TX
RC5
RC4
RC3
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7
4,7K
33uF
4MHZ
VOLUMEN MASTER
MODULACIÓN
BASS
TREBLE
PANNING
MUTE
SOLO
ROJO
VERDE
SW3
SW2
OFF
ON
20K

Anexo 2, Librería pic168xa.h.

```
/*
 *      Header file for the Microchip
 *      PIC 16F873A chip
 *      PIC 16F874A chip
 *      PIC 16F876A chip
 *      PIC 16F877A chip
 *      Midrange Microcontroller
 */

#if defined(_16F874A) || defined(_16F877A)
#define __PINS_40
#endif

static volatile unsigned char INDF   @ 0x00;
static volatile unsigned char TMR0   @ 0x01;
static volatile unsigned char PCL    @ 0x02;
static volatile unsigned char STATUS @ 0x03;
static          unsigned char FSR    @ 0x04;
static volatile unsigned char PORTA  @ 0x05;
static volatile unsigned char PORTB  @ 0x06;
static volatile unsigned char PORTC  @ 0x07;
#ifdef __PINS_40
static volatile unsigned char PORTD  @ 0x08;
static volatile unsigned char PORTE  @ 0x09;
#endif
static          unsigned char PCLATH @ 0x0A;
static volatile unsigned char INTCON @ 0x0B;
static volatile unsigned char PIR1   @ 0x0C;
static volatile unsigned char PIR2   @ 0x0D;
static volatile unsigned char TMR1L  @ 0x0E;
static volatile unsigned char TMR1H  @ 0x0F;
static volatile unsigned char T1CON  @ 0x10;
static volatile unsigned char TMR2   @ 0x11;
static volatile unsigned char T2CON  @ 0x12;
static volatile unsigned char SSPBUF @ 0x13;
static volatile unsigned char SSPCON @ 0x14;
static volatile unsigned char CCPR1L @ 0x15;
static volatile unsigned char CCPR1H @ 0x16;
static volatile unsigned char CCP1CON@ 0x17;
static volatile unsigned char RCSTA  @ 0x18;
static volatile unsigned char TXREG  @ 0x19;
static volatile unsigned char RCREG  @ 0x1A;
static volatile unsigned char CCPR2L @ 0x1B;
static volatile unsigned char CCPR2H @ 0x1C;
static volatile unsigned char CCP2CON@ 0x1D;
static volatile unsigned char ADRESH @ 0x1E;
static volatile unsigned char ADCON0 @ 0x1F;

/*      bank 1 registers */
static          unsigned char bank1  OPTION @ 0x81;
static volatile       unsigned char bank1   TRISA  @ 0x85;
```

```
static volatile       unsigned char bank1   TRISB  @ 0x86;
static volatile       unsigned char bank1   TRISC  @ 0x87;
#ifdef __PINS_40
static volatile unsigned char bank1  TRISD  @ 0x88;
static volatile unsigned char bank1  TRISE  @ 0x89;
#endif
static volatile unsigned char bank1  PIE1   @ 0x8C;
static volatile unsigned char bank1  PIE2   @ 0x8D;
static volatile unsigned char bank1  PCON   @ 0x8E;
static volatile unsigned char bank1  SSPCON2@ 0x91;
static volatile unsigned char bank1  PR2    @ 0x92;
static volatile unsigned char bank1  SSPADD @ 0x93;
static volatile unsigned char bank1  SSPSTAT@ 0x94;
static volatile unsigned char bank1  TXSTA  @ 0x98;
static volatile unsigned char bank1  SPBRG  @ 0x99;
static volatile unsigned char bank1  CMCON  @ 0x9C;
static volatile unsigned char bank1  CVRCON @ 0x9D;
static volatile unsigned char bank1  ADRESL @ 0x9E;
static volatile unsigned char bank1  ADCON1 @ 0x9F;

/*     bank 2 registers */
static volatile unsigned char bank2  EEDATA @ 0x10C;
static volatile unsigned char bank2  EEADR  @ 0x10D;
static volatile unsigned char bank2  EEDATH @ 0x10E;
static volatile unsigned char bank2  EEADRH @ 0x10F;

/*     bank 3 registers */
static volatile unsigned char bank3  EECON1 @ 0x18C;
static volatile unsigned char bank3  EECON2 @ 0x18D;

/*     STATUS bits    */
static volatile bit   IRP     @ (unsigned)&STATUS*8+7;
static volatile bit   RP1     @ (unsigned)&STATUS*8+6;
static volatile bit   RP0     @ (unsigned)&STATUS*8+5;
static volatile bit   TO     @ (unsigned)&STATUS*8+4;
static volatile bit   PD     @ (unsigned)&STATUS*8+3;
static volatile bit   ZERO    @ (unsigned)&STATUS*8+2;
static volatile bit   DC      @ (unsigned)&STATUS*8+1;
static volatile bit   CARRY   @ (unsigned)&STATUS*8+0;

/*      PORTA bits      */
static volatile bit   RA5    @ (unsigned)&PORTA*8+5;
static volatile bit   RA4    @ (unsigned)&PORTA*8+4;
static volatile bit   RA3    @ (unsigned)&PORTA*8+3;
static volatile bit   RA2    @ (unsigned)&PORTA*8+2;
static volatile bit   RA1    @ (unsigned)&PORTA*8+1;
static volatile bit   RA0    @ (unsigned)&PORTA*8+0;

/*      PORTB bits      */
static volatile bit   RB7    @ (unsigned)&PORTB*8+7;
static volatile bit   RB6    @ (unsigned)&PORTB*8+6;
```

```
static volatile bit   RB5    @ (unsigned)&PORTB*8+5;
static volatile bit   RB4    @ (unsigned)&PORTB*8+4;
static volatile bit   RB3    @ (unsigned)&PORTB*8+3;
static volatile bit   RB2    @ (unsigned)&PORTB*8+2;
static volatile bit   RB1    @ (unsigned)&PORTB*8+1;
static volatile bit   RB0    @ (unsigned)&PORTB*8+0;

/*      PORTC bits      */
static volatile bit   RC7    @ (unsigned)&PORTC*8+7;
static volatile bit   RC6    @ (unsigned)&PORTC*8+6;
static volatile bit   RC5    @ (unsigned)&PORTC*8+5;
static volatile bit   RC4    @ (unsigned)&PORTC*8+4;
static volatile bit   RC3    @ (unsigned)&PORTC*8+3;
static volatile bit   RC2    @ (unsigned)&PORTC*8+2;
static volatile bit   RC1    @ (unsigned)&PORTC*8+1;
static volatile bit   RC0    @ (unsigned)&PORTC*8+0;

/*      PORTD bits      */
#ifdef __PINS_40
static volatile bit   RD7    @ (unsigned)&PORTD*8+7;
static volatile bit   RD6    @ (unsigned)&PORTD*8+6;
static volatile bit   RD5    @ (unsigned)&PORTD*8+5;
static volatile bit   RD4    @ (unsigned)&PORTD*8+4;
static volatile bit   RD3    @ (unsigned)&PORTD*8+3;
static volatile bit   RD2    @ (unsigned)&PORTD*8+2;
static volatile bit   RD1    @ (unsigned)&PORTD*8+1;
static volatile bit   RD0    @ (unsigned)&PORTD*8+0;

/*      PORTE bits      */
static volatile bit   RE2    @ (unsigned)&PORTE*8+2;
static volatile bit   RE1    @ (unsigned)&PORTE*8+1;
static volatile bit   RE0    @ (unsigned)&PORTE*8+0;
#endif

/*     INTCON bits    */
static volatile bit   GIE    @ (unsigned)&INTCON*8+7;
static volatile bit   PEIE   @ (unsigned)&INTCON*8+6;
static volatile bit   T0IE   @ (unsigned)&INTCON*8+5;
static volatile bit   INTE   @ (unsigned)&INTCON*8+4;
static volatile bit   RBIE   @ (unsigned)&INTCON*8+3;
static volatile bit   T0IF   @ (unsigned)&INTCON*8+2;
static volatile bit   INTF   @ (unsigned)&INTCON*8+1;
static volatile bit   RBIF   @ (unsigned)&INTCON*8+0;
// alternate definitions
static volatile bit   TMR0IE @ (unsigned)&INTCON*8+5;
static volatile bit   TMR0IF @ (unsigned)&INTCON*8+2;

/*     PIR1 bits      */
#ifdef __PINS_40
static volatile bit   PSPIF  @ (unsigned)&PIR1*8+7;
#endif
```

```
static volatile bit   ADIF   @ (unsigned)&PIR1*8+6;
static volatile bit   RCIF   @ (unsigned)&PIR1*8+5;
static volatile bit   TXIF   @ (unsigned)&PIR1*8+4;
static volatile bit   SSPIF  @ (unsigned)&PIR1*8+3;
static volatile bit   CCP1IF @ (unsigned)&PIR1*8+2;
static volatile bit   TMR2IF @ (unsigned)&PIR1*8+1;
static volatile bit   TMR1IF @ (unsigned)&PIR1*8+0;

/*     PIR2 bits     */
static volatile bit   CMIF   @ (unsigned)&PIR2*8+6;
static volatile bit   EEIF   @ (unsigned)&PIR2*8+4;
static volatile bit   BCLIF  @ (unsigned)&PIR2*8+3;
static volatile bit   CCP2IF @ (unsigned)&PIR2*8+0;

/*     T1CON bits    */
static volatile bit   T1CKPS1@ (unsigned)&T1CON*8+5;
static volatile bit   T1CKPS0@ (unsigned)&T1CON*8+4;
static volatile bit   T1OSCEN@ (unsigned)&T1CON*8+3;
static volatile bit   T1SYNC @ (unsigned)&T1CON*8+2;
static volatile bit   TMR1CS @ (unsigned)&T1CON*8+1;
static volatile bit   TMR1ON @ (unsigned)&T1CON*8+0;

/*     T2CON bits    */
static volatile bit   TOUTPS3@ (unsigned)&T2CON*8+6;
static volatile bit   TOUTPS2@ (unsigned)&T2CON*8+5;
static volatile bit   TOUTPS1@ (unsigned)&T2CON*8+4;
static volatile bit   TOUTPS0@ (unsigned)&T2CON*8+3;
static volatile bit   TMR2ON @ (unsigned)&T2CON*8+2;
static volatile bit   T2CKPS1@ (unsigned)&T2CON*8+1;
static volatile bit   T2CKPS0@ (unsigned)&T2CON*8+0;

/*     SSPCON bits   */
static volatile bit   WCOL   @ (unsigned)&SSPCON*8+7;
static volatile bit   SSPOV  @ (unsigned)&SSPCON*8+6;
static volatile bit   SSPEN  @ (unsigned)&SSPCON*8+5;
static volatile bit   CKP    @ (unsigned)&SSPCON*8+4;
static volatile bit   SSPM3  @ (unsigned)&SSPCON*8+3;
static volatile bit   SSPM2  @ (unsigned)&SSPCON*8+2;
static volatile bit   SSPM1  @ (unsigned)&SSPCON*8+1;
static volatile bit   SSPM0  @ (unsigned)&SSPCON*8+0;

/*     CCP1CON bits  */
static volatile bit   CCP1X  @ (unsigned)&CCP1CON*8+5;
static volatile bit   CCP1Y  @ (unsigned)&CCP1CON*8+4;
static volatile bit   CCP1M3 @ (unsigned)&CCP1CON*8+3;
static volatile bit   CCP1M2 @ (unsigned)&CCP1CON*8+2;
static volatile bit   CCP1M1 @ (unsigned)&CCP1CON*8+1;
static volatile bit   CCP1M0 @ (unsigned)&CCP1CON*8+0;

/*     RCSTA bits    */
static volatile bit   SPEN   @ (unsigned)&RCSTA*8+7;
```

```
static volatile bit   RX9    @ (unsigned)&RCSTA*8+6;
static volatile bit   SREN   @ (unsigned)&RCSTA*8+5;
static volatile bit   CREN   @ (unsigned)&RCSTA*8+4;
static volatile bit   ADDEN  @ (unsigned)&RCSTA*8+3;
static volatile bit   FERR   @ (unsigned)&RCSTA*8+2;
static volatile bit   OERR   @ (unsigned)&RCSTA*8+1;
static volatile bit   RX9D   @ (unsigned)&RCSTA*8+0;

/*     CCP2CON bits   */
static volatile bit   CCP2X  @ (unsigned)&CCP2CON*8+5;
static volatile bit   CCP2Y  @ (unsigned)&CCP2CON*8+4;
static volatile bit   CCP2M3 @ (unsigned)&CCP2CON*8+3;
static volatile bit   CCP2M2 @ (unsigned)&CCP2CON*8+2;
static volatile bit   CCP2M1 @ (unsigned)&CCP2CON*8+1;
static volatile bit   CCP2M0 @ (unsigned)&CCP2CON*8+0;

/*      ADCON0 bits   */
static volatile bit   ADCS1  @ (unsigned)&ADCON0*8+7;
static volatile bit   ADCS0  @ (unsigned)&ADCON0*8+6;
static volatile bit   CHS2   @ (unsigned)&ADCON0*8+5;
static volatile bit   CHS1   @ (unsigned)&ADCON0*8+4;
static volatile bit   CHS0   @ (unsigned)&ADCON0*8+3;
static volatile bit   ADGO   @ (unsigned)&ADCON0*8+2;
static volatile bit   ADON   @ (unsigned)&ADCON0*8+0;

/*     OPTION bits    */
static bank1 bit      RBPU   @ (unsigned)&OPTION*8+7;
static bank1 bit      INTEDG @ (unsigned)&OPTION*8+6;
static bank1 bit      T0CS   @ (unsigned)&OPTION*8+5;
static bank1 bit      T0SE   @ (unsigned)&OPTION*8+4;
static bank1 bit      PSA    @ (unsigned)&OPTION*8+3;
static bank1 bit      PS2    @ (unsigned)&OPTION*8+2;
static bank1 bit      PS1    @ (unsigned)&OPTION*8+1;
static bank1 bit      PS0    @ (unsigned)&OPTION*8+0;

/*      TRISA bits      */
static volatile bank1 bit    TRISA5 @ (unsigned)&TRISA*8+5;
static volatile bank1 bit    TRISA4 @ (unsigned)&TRISA*8+4;
static volatile bank1 bit    TRISA3 @ (unsigned)&TRISA*8+3;
static volatile bank1 bit    TRISA2 @ (unsigned)&TRISA*8+2;
static volatile bank1 bit    TRISA1 @ (unsigned)&TRISA*8+1;
static volatile bank1 bit    TRISA0 @ (unsigned)&TRISA*8+0;

/*      TRISB bits      */
static volatile bank1 bit    TRISB7 @ (unsigned)&TRISB*8+7;
static volatile bank1 bit    TRISB6 @ (unsigned)&TRISB*8+6;
static volatile bank1 bit    TRISB5 @ (unsigned)&TRISB*8+5;
static volatile bank1 bit    TRISB4 @ (unsigned)&TRISB*8+4;
static volatile bank1 bit    TRISB3 @ (unsigned)&TRISB*8+3;
static volatile bank1 bit    TRISB2 @ (unsigned)&TRISB*8+2;
static volatile bank1 bit    TRISB1 @ (unsigned)&TRISB*8+1;
```

```
static volatile bank1 bit    TRISB0 @ (unsigned)&TRISB*8+0;

/*      TRISC bits      */
static volatile bank1 bit    TRISC7 @ (unsigned)&TRISC*8+7;
static volatile bank1 bit    TRISC6 @ (unsigned)&TRISC*8+6;
static volatile bank1 bit    TRISC5 @ (unsigned)&TRISC*8+5;
static volatile bank1 bit    TRISC4 @ (unsigned)&TRISC*8+4;
static volatile bank1 bit    TRISC3 @ (unsigned)&TRISC*8+3;
static volatile bank1 bit    TRISC2 @ (unsigned)&TRISC*8+2;
static volatile bank1 bit    TRISC1 @ (unsigned)&TRISC*8+1;
static volatile bank1 bit    TRISC0 @ (unsigned)&TRISC*8+0;

#ifdef __PINS_40
/*      TRISD bits      */
static volatile bank1 bit    TRISD7 @ (unsigned)&TRISD*8+7;
static volatile bank1 bit    TRISD6 @ (unsigned)&TRISD*8+6;
static volatile bank1 bit    TRISD5 @ (unsigned)&TRISD*8+5;
static volatile bank1 bit    TRISD4 @ (unsigned)&TRISD*8+4;
static volatile bank1 bit    TRISD3 @ (unsigned)&TRISD*8+3;
static volatile bank1 bit    TRISD2 @ (unsigned)&TRISD*8+2;
static volatile bank1 bit    TRISD1 @ (unsigned)&TRISD*8+1;
static volatile bank1 bit    TRISD0 @ (unsigned)&TRISD*8+0;

/*      TRISE bits      */
static volatile bank1 bit    IBF    @ (unsigned)&TRISE*8+7;
static volatile bank1 bit    OBF    @ (unsigned)&TRISE*8+6;
static volatile bank1 bit    IBOV   @ (unsigned)&TRISE*8+5;
static volatile bank1 bit    PSPMODE@ (unsigned)&TRISE*8+4;

static volatile bank1 bit    TRISE2  @ (unsigned)&TRISE*8+2;
static volatile bank1 bit    TRISE1  @ (unsigned)&TRISE*8+1;
static volatile bank1 bit    TRISE0  @ (unsigned)&TRISE*8+0;
#endif

/*     PIE1 bits      */
#ifdef __PINS_40
static volatile bank1 bit    PSPIE  @ (unsigned)&PIE1*8+7;
#endif
static volatile bank1 bit    ADIE   @ (unsigned)&PIE1*8+6;
static volatile bank1 bit    RCIE   @ (unsigned)&PIE1*8+5;
static volatile bank1 bit    TXIE   @ (unsigned)&PIE1*8+4;
static volatile bank1 bit    SSPIE  @ (unsigned)&PIE1*8+3;
static volatile bank1 bit    CCP1IE @ (unsigned)&PIE1*8+2;
static volatile bank1 bit    TMR2IE @ (unsigned)&PIE1*8+1;
static volatile bank1 bit    TMR1IE @ (unsigned)&PIE1*8+0;

/*     PIE2 bits      */
static volatile bank1 bit    CMIE   @ (unsigned)&PIE2*8+6;
static volatile bank1 bit    EEIE   @ (unsigned)&PIE2*8+4;
static volatile bank1 bit    BCLIE  @ (unsigned)&PIE2*8+3;
static volatile bank1 bit    CCP2IE @ (unsigned)&PIE2*8+0;
```

```
/*      PCON bits      */
static volatile bank1 bit     POR     @ (unsigned)&PCON*8+1;
static volatile bank1 bit     BOR     @ (unsigned)&PCON*8+0;

/*      SSPCON2 bits */
static volatile bank1 bit     GCEN    @ (unsigned)&SSPCON2*8+7;
static volatile bank1 bit     ACKSTAT @ (unsigned)&SSPCON2*8+6;
static volatile bank1 bit     ACKDT   @ (unsigned)&SSPCON2*8+5;
static volatile bank1 bit     ACKEN   @ (unsigned)&SSPCON2*8+4;
static volatile bank1 bit     RCEN    @ (unsigned)&SSPCON2*8+3;
static volatile bank1 bit     PEN     @ (unsigned)&SSPCON2*8+2;
static volatile bank1 bit     RSEN    @ (unsigned)&SSPCON2*8+1;
static volatile bank1 bit     SEN     @ (unsigned)&SSPCON2*8+0;

/*       SSPSTAT bits    */
static volatile bank1 bit     STAT_SMP       @ (unsigned)&SSPSTAT*8+7;
static volatile bank1 bit     STAT_CKE       @ (unsigned)&SSPSTAT*8+6;
static volatile bank1 bit     STAT_DA        @ (unsigned)&SSPSTAT*8+5;
static volatile bank1 bit     STAT_P         @ (unsigned)&SSPSTAT*8+4;
static volatile bank1 bit     STAT_S         @ (unsigned)&SSPSTAT*8+3;
static volatile bank1 bit     STAT_RW        @ (unsigned)&SSPSTAT*8+2;
static volatile bank1 bit     STAT_UA        @ (unsigned)&SSPSTAT*8+1;
static volatile bank1 bit     STAT_BF        @ (unsigned)&SSPSTAT*8+0;

/*      TXSTA bits     */
static volatile bank1 bit     CSRC    @ (unsigned)&TXSTA*8+7;
static volatile bank1 bit     TX9     @ (unsigned)&TXSTA*8+6;
static volatile bank1 bit     TXEN    @ (unsigned)&TXSTA*8+5;
static volatile bank1 bit     SYNC    @ (unsigned)&TXSTA*8+4;
static volatile bank1 bit     BRGH    @ (unsigned)&TXSTA*8+2;
static volatile bank1 bit     TRMT    @ (unsigned)&TXSTA*8+1;
static volatile bank1 bit     TX9D    @ (unsigned)&TXSTA*8+0;

/*      CMCON Bits     */
static volatile bank1 bit     C2OUT   @ (unsigned)&CMCON*8+7;
static volatile bank1 bit     C1OUT   @ (unsigned)&CMCON*8+6;
static volatile bank1 bit     C2INV   @ (unsigned)&CMCON*8+5;
static volatile bank1 bit     C1INV   @ (unsigned)&CMCON*8+4;
static volatile bank1 bit     CIS     @ (unsigned)&CMCON*8+3;
static volatile bank1 bit     CM2     @ (unsigned)&CMCON*8+2;
static volatile bank1 bit     CM1     @ (unsigned)&CMCON*8+1;
static volatile bank1 bit     CM0     @ (unsigned)&CMCON*8+0;

/*      CVRCON Bits    */
static volatile bank1 bit     CVREN   @ (unsigned)&CVRCON*8+7;
static volatile bank1 bit     CVROE   @ (unsigned)&CVRCON*8+6;
static volatile bank1 bit     CVRR    @ (unsigned)&CVRCON*8+5;
static volatile bank1 bit     CVR3    @ (unsigned)&CVRCON*8+3;
static volatile bank1 bit     CVR2    @ (unsigned)&CVRCON*8+2;
static volatile bank1 bit     CVR1    @ (unsigned)&CVRCON*8+1;
```

```
static volatile bank1 bit     CVR0   @ (unsigned)&CVRCON*8+0;

/*     ADCON1 bits    */
static volatile bank1 bit     ADFM   @ (unsigned)&ADCON1*8+7;
static volatile bank1 bit     ADCS2  @ (unsigned)&ADCON1*8+6;
static volatile bank1 bit     PCFG3  @ (unsigned)&ADCON1*8+3;
static volatile bank1 bit     PCFG2  @ (unsigned)&ADCON1*8+2;
static volatile bank1 bit     PCFG1  @ (unsigned)&ADCON1*8+1;
static volatile bank1 bit     PCFG0  @ (unsigned)&ADCON1*8+0;

/*     EECON1 bits */
static volatile bank3 bit     EEPGD  @ (unsigned)&EECON1*8+7;
static volatile bank3 bit     WRERR  @ (unsigned)&EECON1*8+3;
static volatile bank3 bit     WREN   @ (unsigned)&EECON1*8+2;
static volatile bank3 bit     WR     @ (unsigned)&EECON1*8+1;
static volatile bank3 bit     RD     @ (unsigned)&EECON1*8+0;


#define CONFIG_ADDR   0x2007

/*osc configurations*/
#define RC            0x3FFF // resistor/capacitor
#define HS            0x3FFE // high speed crystal/resonator
#define XT            0x3FFD // crystal/resonator
#define LP            0x3FFC // low power crystal/resonator

/*watchdog*/
#define WDTEN         0x3FFF // enable watchdog timer
#define WDTDIS        0x3FFB // disable watchdog timer

/*power up timer*/
#define PWRTEN        0x3FF7 // enable power up timer
#define PWRTDIS               0x3FFF // disable power up timer

/*brown out reset*/
#define BOREN         0x3FFF // enable brown out reset
#define BORDIS        0x3FBF // disable brown out reset

/*Low Voltage Programmable*/
#define LVPEN         0x3FFF // low voltage programming enabled
#define LVPDIS        0x3F7F // low voltage programming disabled

/*data code protected*/
#define DP            0x3EFF // protect data code
// alternately
#define DPROT         0x3EFF // use DP
#define DUNPROT               0x3FFF // use UNPROTECT

/* Flash memory write enable/protect */
#define WRTEN         0x3FFF /* flash memory write enabled */
#define WP1           0x3DFF  /* protect 0000 - 00FF */
```

```
#define WP2           0x3BFF  /* protect 0000 - 07FF(76A/77A) / 03FF(73A/74A) */
#define WP3           0x39FF  /* protect 0000 - 1FFF(76A/77A) / 0FFF(73A/74A) */

/*debug option*/
#define DEBUGEN               0x37FF  // debugger enabled
#define DEBUGDIS      0x3FFF  // debugger disabled

/*code protection*/
#define PROTECT               0x1FFF  /* protect program code */
#define UNPROTECT     0x3FFF  /* do not protect the code */
```

Anexo 3, Librería stdio.h.

```
#ifndef _STDIO_H_
#define _STDIO_H_
#if    z80
#define BUFSIZ         512
#define _NFILE         8
#else  /* z80 */
#define BUFSIZ         1024
#define _NFILE         20
#endif /* z80 */

#ifndef _STDDEF
typedef int            ptrdiff_t;     /* result type of pointer difference */
typedef unsigned       size_t;        /* type yielded by sizeof */
typedef unsigned shortwchar_t;        /* wide char type */
#define _STDDEF
#define offsetof(ty, mem)     ((int)&(((ty *)0)->mem))
#endif /* _STDDEF */

#ifndef _STDARG
#include       <stdarg.h>
#endif

#ifndef NULL
#define NULL   (0)
#endif /* NULL */

extern int     errno;                 /* system error number */

#ifndef FILE

#if    _HOSTED
extern  struct _iobuf {
        char *         _ptr;
        int            _cnt;
        char *         _base;
        unsigned short _flag;
        short          _file;
        size_t         _size;
} _iob[_NFILE];

#define FILE           struct _iobuf

#define L_tmpnam       81             /* max length of temporary names */
#define _MAXTFILE      8              /* max number of temporary files */

#if    DOS
#define FILENAME_MAX   81             /* max length of a pathname */
#define FOPEN_MAX      5
#endif

extern struct _tfiles {
```

```
        char    tname[L_tmpnam];
        FILE *  tfp;
}       * _tfilesptr;

#else   /* _HOSTED */

struct __prbuf
{
        char *          ptr;
        void (*         func)(char);
};
#endif  /* _HOSTED */
#endif  /* FILE */

#define _IOFBF          0
#define _IOREAD         01
#define _IOWRT          02
#define _IORW           03
#define _IONBF          04
#define _IOMYBUF        010
#define _IOEOF          020
#define _IOERR          040
#define _IOSTRG         0100
#define _IOBINARY       0200
#define _IOLBF          0400
#define _IODIRN         01000   /* true when file is in write mode */
#define _IOAPPEND       02000   /* file was opened in append mode */
#define _IOSEEKED       04000   /* a seek has occured since last write */
#define _IOTMPFILE      010000  /* this file is a temporary */

#define EOF             (-1)
#define _IOSTRING       (-67)

#define SEEK_SET        0
#define SEEK_CUR        1
#define SEEK_END        2
#define TMP_MAX         255

#if     _HOSTED
#define stdin           (&_iob[0])
#define stdout          (&_iob[1])
#define stderr          (&_iob[2])
#ifdef  DOS
#define stdprn          (&_iob[3])
#endif
#define getchar()       getc(stdin)
#define putchar(x)      putc(x,stdout)
#else   /* _HOSTED */
#include        <conio.h>
#define getchar()       getche()
#define putchar(x)      putch(x)
```

```
extern int     cprintf(char *, ...);
#pragma printf_check(cprintf)
#if    defined(_MPC_) && !defined(_PIC18)
extern void    _doprnt(char *, const register char *, ...);
#else
extern int     _doprnt(struct __prbuf *, const register char *, register va_list);
#endif /* _MPC_ */
#endif /* _HOSTED */

/*     getc() and putc() must be functions for CP/M to allow the special
 *     handling of '\r', '\n' and '\032'. The same for MSDOS except that
 *     it at least knows the length of a file.
 */

#define getc(p)        fgetc(p)
#define putc(x,p)      fputc(x,p)

#define feof(p)        (((p)->_flag&_IOEOF)!=0)
#define ferror(p)      (((p)->_flag&_IOERR)!=0)
#define fileno(p)      ((unsigned short)p->_file)
#define clrerr(p)      p->_flag &= ~_IOERR
#define clreof(p)      p->_flag &= ~_IOEOF
#define clearerr(p)    p->_flag &= ~(_IOERR|_IOEOF)


#if    _HOSTED
extern int     _flsbuf(char, FILE *);
extern int     _filbuf(FILE *);
extern int     fclose(FILE *);
extern int     fflush(FILE *);
extern int     fgetc(FILE *);
extern int     ungetc(int, FILE *);
extern int     fputc(int, FILE *);
extern int     getw(FILE *);
extern int     putw(int, FILE *);
extern int     fputs(const char *, FILE *);
extern int     fread(void *, size_t, size_t, FILE *);
extern int     fwrite(const void *, size_t, size_t, FILE *);
extern int     fseek(FILE *, long, int);
extern int     rewind(FILE *);
extern void    setbuf(FILE *, char *);
extern int     setvbuf(FILE *, char *, int, size_t);
extern int     fprintf(FILE *, const char *, ...);
extern int     fscanf(FILE *, const char *, ...);
extern int     vfprintf(FILE *, const char *, va_list);
extern int     vfscanf(FILE *, const char *, va_list);
extern int     remove(const char *);
extern int     rename(const char *, const char *);
extern FILE *  fopen(const char *, const char *);
extern FILE *  freopen(const char *, const char *, FILE *);
extern FILE *  fdopen(int, const char *);
```

```
extern long    ftell(FILE *);
extern char *  fgets(char *, int, FILE *);
extern void    perror(const char *);
extern char *  _bufallo(void);
extern void    _buffree(char *);
extern char *  tmpnam(char *);
extern FILE *  tmpfile(void);

#if    unix
extern FILE *  popen(char *, char *);
extern int     pclose(FILE *);
#endif
extern void    (*_atexitptr)(void);

#pragma printf_check(fprintf)

#endif  /* __HOSTED */

#if    defined(_MPC_) && !defined(_PIC18)
extern int     _doscan(const char *, const char *, va_list);
//#define vprintf(s, l)              _doprnt(0, (s), (l))
//#define vsprintf(b, s, l)  _doprnt((b), (s), (l))
//#define vscanf(s, l)       _doscan(0, (s), (l))
//#define vsscanf(b, s, l)   _doscan((b), (s), (l))
#pragma printf_check(printf) const
#pragma printf_check(sprintf) const

#if defined(_PIC16)
extern unsigned char  sprintf(far char *, const char *, ...);
#else  /* _PIC16 */
extern unsigned char  sprintf(char *, const char *, ...);
#endif
#if    defined(_PIC18)
extern int     printf(const char *, ...);
#else
extern unsigned char  printf(const char *, ...);
#endif
#else /* _MPC_ */

extern char *  gets(char *);
extern int     puts(const char *);
extern int     scanf(const char *, ...);
extern int     sscanf(const char *, const char *, ...);
extern int     vprintf(const char *, va_list);
extern int     vsprintf(char *, const char *, va_list);
extern int     vscanf(const char *, va_list ap);
extern int     vsscanf(const char *, const char *, va_list);

#pragma printf_check(printf)
#pragma printf_check(sprintf)
extern int sprintf(char *, const char *, ...);
```

```
extern int printf(const char *, ...);

#endif /* _MPC_ */

#endif /* _STDIO_H_ */
```

Anexo 4, Librería delay.h.

```
/*
*       Delay functions for HI-TECH C on the PIC
*
*       Functions available:
*               DelayUs(x)      Delay specified number of microseconds
*               DelayMs(x)      Delay specified number of milliseconds
*
*       Note that there are range limits: x must not exceed 255 - for xtal
*       frequencies > 12MHz the range for DelayUs is even smaller.
*       To use DelayUs it is only necessary to include this file; to use
*       DelayMs you must include delay.c in your project.
*
*/

/*      Set the crystal frequency in the CPP predefined symbols list in
HPDPIC, or on the PICC commmand line, e.g.
picc -DXTAL_FREQ=4MHZ

or
picc -DXTAL_FREQ=100KHZ

Note that this is the crystal frequency, the CPU clock is
divided by 4.

*       MAKE SURE this code is compiled with full optimization!!!

*/

#ifndef XTAL_FREQ
#define XTAL_FREQ       4MHZ            /* Crystal frequency in MHz */
#endif

#define MHZ     *1000L                  /* number of kHz in a MHz */
#define KHZ     *1                      /* number of kHz in a kHz */

#if     XTAL_FREQ >= 12MHZ

#define DelayUs(x)      { unsigned char _dcnt; \
_dcnt = (x)*((XTAL_FREQ)/(12MHZ)); \
while(--_dcnt != 0) \
continue; }
#else

#define DelayUs(x)      { unsigned char _dcnt; \
_dcnt = (x)/((12MHZ)/(XTAL_FREQ))|1; \
while(--_dcnt != 0) \
continue; }
#endif

extern void DelayMs(unsigned char);
```

Anexo 5, Librería adc.h.

```
/*
 * Analog conversion stuff for 16C71
 */

/*
 * Read the adc on the specified channel - result is in ADRES
 */

extern void adc_read(unsigned char channel);
```

Anexo 6, Librería lcd.h.

```
/*
 *      LCD interface header file
*/

/* write a byte to the LCD in 4 bit mode */

extern void lcd_write(unsigned char);

/* Clear and home the LCD */

extern void lcd_clear(void);

/* write a string of characters to the LCD */

extern void lcd_puts(const char * s);

/* Go to the specified position */

extern void lcd_goto(unsigned char pos);

/* intialize the LCD - call before anything else */

extern void lcd_init(void);

extern void lcd_putch(char);

/*      Set the cursor position */

#define lcd_cursor(x)  lcd_write(((x)&0x7F)|0x80)
```

Anexo 7, Librería float.h.

```
/*      Characteristics of floating types */

#define DBL_RADIX      2              /* radix of exponent for a double */
#define DBL_ROUNDS     0              /* doubles don't round when converted to int */
#define FLT_RADIX      2              /* radix of float exponent */
#define FLT_ROUNDS     0              /* float also truncates to int */

#if    defined(i8096) || defined(m6800) || defined(m6809) || defined(i8051) || \
       defined(h8300) || defined(h8300h)|| defined(m6805) || defined(_V8) || \
       defined(__MSP430C__) || defined(__ARMC__) || defined(__DSPICC__)

#define FLT_MANT_DIG   24
#define FLT_EPSILON    1.19209290e-07
#define FLT_DIG        6
#define FLT_MIN_EXP    -125
#define FLT_MIN        1.17549435e-38
#define FLT_MIN_10_EXP -37
#define FLT_MAX_EXP    128
#define FLT_MAX          3.40282347e+38
#define FLT_MAX_10_EXP 38
#define DBL_MANT_DIG   24
#define DBL_EPSILON    1.19209290e-07
#define DBL_DIG        6
#define DBL_MIN_EXP    -125
#define DBL_MIN        1.17549435e-38
#define DBL_MIN_10_EXP -37
#define DBL_MAX_EXP    128
#define DBL_MAX          3.40282347e+38
#define DBL_MAX_10_EXP 38
#endif

#if    defined(_MPC_)

/* MICROCHIP PIC */

#define FLT_MANT_DIG   16
#define FLT_EPSILON    3.05176e-05
#define FLT_DIG        5
#define FLT_MIN_EXP    -125
#define FLT_MIN        1.17549435e-38
#define FLT_MIN_10_EXP -37
#define FLT_MAX_EXP    128
#define FLT_MAX          3.40277e+38
#define FLT_MAX_10_EXP 38
#if    sizeof(double) == 4
#define DBL_MANT_DIG   24
#define DBL_EPSILON    1.19209290e-07
#define DBL_DIG        6
#define DBL_MIN_EXP    -125
#define DBL_MIN        1.17549435e-38
#define DBL_MIN_10_EXP -37
```

```
#define DBL_MAX_EXP    128
#define  DBL_MAX         3.40282347e+38
#define DBL_MAX_10_EXP 38
#else
#define DBL_MANT_DIG   FLT_MANT_DIG
#define DBL_EPSILON    FLT_EPSILON
#define DBL_DIG        FLT_DIG
#define DBL_MIN_EXP    FLT_MIN_EXP
#define DBL_MIN        FLT_MIN
#define DBL_MIN_10_EXP FLT_MIN_10_EXP
#define DBL_MAX_EXP    FLT_MAX_EXP
#define  DBL_MAX         FLT_MAX
#define DBL_MAX_10_EXP FLT_MAX_10_EXP
#endif
#endif

#if    defined(_XA_)
#define FLT_MANT_DIG   24
#define FLT_EPSILON    1.19209290e-07
#define FLT_DIG        6
#define FLT_MIN_EXP    -125
#define FLT_MIN        1.17549435e-38
#define FLT_MIN_10_EXP -37
#define FLT_MAX_EXP    128
#define  FLT_MAX         3.40282347e+38
#define FLT_MAX_10_EXP 38
#define FLT_MANT_DIG   24             /* 24 bits in mantissa */
#if    sizeof(double) == 4
#define DBL_MANT_DIG   24
#define DBL_EPSILON    1.19209290e-07
#define DBL_DIG        6
#define DBL_MIN_EXP    -125
#define DBL_MIN        1.17549435e-38
#define DBL_MIN_10_EXP -37
#define DBL_MAX_EXP    128
#define  DBL_MAX         3.40282347e+38
#define DBL_MAX_10_EXP 38
#define DBL_MANT_DIG   24             /* 24 bits in mantissa */
#elif  __FASTDBL
#define DBL_MANT_DIG   48
#define DBL_EPSILON    2.147544363e-10
#define DBL_DIG        14
#define DBL_MIN_EXP    -16381
#define DBL_MIN        2.225073858507202e-1640
#define DBL_MIN_10_EXP -1640
#define DBL_MAX_EXP    16384
#define DBL_MAX        1.797693134862315e+1640
#define DBL_MAX_10_EXP 1641
#else
#define DBL_MANT_DIG   53
#define DBL_EPSILON    2.220446049250313e-16
```

```
#define DBL_DIG        15
#define DBL_MIN_EXP    -1021
#define DBL_MIN        2.225073858507202e-308
#define DBL_MIN_10_EXP -307
#define DBL_MAX_EXP    1024
#define DBL_MAX        1.797693134862315e+308
#define DBL_MAX_10_EXP 308
#endif
#endif


#if    defined(z80)
#define FLT_MANT_DIG   24             /* 24 bits in mantissa */
#define DBL_MANT_DIG   24             /* ditto for double */
#define DBL_MANT_DIG   24             /* ditto long double */
#define FLT_EPSILON    1.19209290e-07 /* smallest x, x+1.0 != 1.0 */
#define DBL_EPSILON    1.19209290e-07 /* smallest x, x+1.0 != 1.0 */
#define FLT_DIG        6              /* decimal significant digs */
#define DBL_DIG        6
#define FLT_MIN_EXP    -63            /* min binary exponent */
#define DBL_MIN_EXP    -63
#define FLT_MIN        1.0842021e-19  /* smallest floating number */
#define DBL_MIN        1.0842021e-19
#define FLT_MIN_10_EXP -19
#define DBL_MIN_10_EXP -19
#define FLT_MAX_EXP    63             /* max binary exponent */
#define DBL_MAX_EXP    63
#define FLT_MAX        9.223369e18    /* max floating number */
#define DBL_MAX        9.223369e18
#define FLT_MAX_10_EXP 18             /* max decimal exponent */
#define DBL_MAX_10_EXP 18
#endif  z80

#if    defined(i8086) || defined(m68k)

/*     The 8086 and 68000 use IEEE 32 and 64 bit float/doubles */

#define FLT_MANT_DIG   24
#define FLT_EPSILON    1.192092896e-07
#define FLT_DIG        6
#define FLT_MIN_EXP    -125
#define FLT_MIN        1.175494351e-38
#define FLT_MIN_10_EXP -37
#define FLT_MAX_EXP    128
#define FLT_MAX        3.402823466+38
#define FLT_MAX_10_EXP 38
#define DBL_MANT_DIG   53
#define DBL_EPSILON    2.220446049250313e-16
#define DBL_DIG        15
#define DBL_MIN_EXP    -1021
#define DBL_MIN        2.225073858507202e-308
```

```
#define DBL_MIN_10_EXP -307
#define DBL_MAX_EXP    1024
#define DBL_MAX        1.797693134862315e+308
#define DBL_MAX_10_EXP 308
#endif //      i8086 || m68k


/*      long double equates to double */


#define LDBL_MANT_DIG  DBL_MANT_DIG
#define LDBL_EPSILON   DBL_EPSILON
#define LDBL_DIG       DBL_DIG
#define LDBL_MIN_EXP   DBL_MIN_EXP
#define LDBL_MIN       DBL_MIN
#define LDBL_MIN_10_EXP        DBL_MIN_10_EXP
#define LDBL_MAX_EXP   DBL_MAX_EXP
#define LDBL_MAX       DBL_MAX
#define LDBL_MAX_10_EXP        DBL_MAX_10_EXP
```

Anexo 8, Librería lcd.c.

```
/*
 *      LCD interface example
 *      Uses routines from delay.c
 *      This code will interface to a standard LCD controller
 *      like the Hitachi HD44780. It uses it in 4 bit mode, with
 *      the hardware connected as follows (the standard 14 pin
 *      LCD connector is used):
 *
 *      PORTB bits 0-3 are connected to the LCD data bits 4-7 (high nibble)
 *      PORTB bit 5 is connected to the LCD RS input (register select)
 *      PORTB bit 4 is connected to the LCD EN bit (enable)
 *
 *      To use these routines, set up the port I/O (TRISA, TRISB) then
 *      call lcd_init(), then other routines as required.
 *
 */

#include       <pic.h>
#include       "lcd.h"
#include       "delay.h"

static bit LCD_RS     @ ((unsigned)&PORTB*8+5);     // Register select
static bit LCD_EN     @ ((unsigned)&PORTB*8+4);     // Enable

#define LCD_STROBE     ((LCD_EN = 1),(LCD_EN=0))


/* write a byte to the LCD in 4 bit mode */

void
lcd_write(unsigned char c)
{
       PORTB = (PORTB & 0xF0) |  (c >> 4);
       LCD_STROBE;
       PORTB = (PORTB & 0xF0) |  (c & 0x0F);
       LCD_STROBE;
       DelayUs(40);
}

/*
 *      Clear and home the LCD
 */

void
lcd_clear(void)
{
       LCD_RS = 0;
       lcd_write(0x1);
       DelayMs(2);
}
```

```
/* write a string of chars to the LCD */

void
lcd_puts(const char * s)
{
	LCD_RS = 1;	// write characters
	while(*s)
		lcd_write(*s++);
}

/* write one character to the LCD */

void
lcd_putch(char c)
{
	LCD_RS = 1;	// write characters
	PORTB = (PORTB & 0xF0) |  (c >> 4);
	LCD_STROBE;
	PORTB = (PORTB & 0xF0) |  (c & 0x0F);
	LCD_STROBE;
	DelayUs(40);
}


/*
 * Go to the specified position
 */

void
lcd_goto(unsigned char pos)
{
	LCD_RS = 0;
	lcd_write(0x80+pos);
}

/* initialise the LCD - put into 4 bit mode */

void
lcd_init(void)
{
	LCD_RS = 0;	// write control bytes
	DelayMs(15);	// power on delay
	PORTB = 0x3;	// attention!
	LCD_STROBE;
	DelayMs(5);
	LCD_STROBE;
	DelayUs(100);
	LCD_STROBE;
	DelayMs(5);
	PORTB = 0x2;	// set 4 bit mode
	LCD_STROBE;
```

```
        DelayUs(40);
        lcd_write(0x28);      // 4 bit mode, 1/16 duty, 5x8 font
        lcd_write(0x08);      // display off
        lcd_write(0x0E);      // display on, blink curson off
        lcd_write(0x06);      // entry mode
}
```

Anexo 9, Código de Programación.

```
#include <pic168xa.h>
#include <stdio.h>
#include <delay.h>
#include <delay.c>
#include "adc.h"
#include <lcd.c>
#include <lcd.h>
#include <float.h>

int volumen;
char dato1[15];

void init_ADC();
void selec_Channel(int a,int b,int c);
void Tx(int t);

void main(void)
{

	TRISA = 0xFF;		//Puerto A como entrada
	TRISB = 0x00;		//Puerto B como salida

	SPBRG=1;		//Baud Rate 31.25Kbauds
	BRGH=0;		//High Speed Off
	SYNC=0;		//Asynchronous Mode
	SPEN=1;		//Enable the asynchronous serial port
	TX9=0;		//8 bit transmition

	lcd_init();
	PORTA = 0x00;
	ADCS1=1;		//Oscilador interno RC
	ADCS0=1;
	ADCON1 = 0x00; //AN7=AN6=Digitales  AN5:AN0=Analógicas

	while(1)
	{
		lcd_clear();
		selec_Channel(0,0,0);
		init_ADC();
		volumen = (0.7944*ADRESH+0.7354);
		lcd_goto(0x00);
		sprintf(dato1, "Volumen: %d", volumen);
		lcd_puts(dato1);
		lcd_puts("%");
		lcd_goto(0x40);		//Segunda línea
		lcd_puts("Channel 1");
		Tx(7);			//Volumen

		selec_Channel(0,0,1);
		init_ADC();
		Tx(1);		//Modulation
```

```
                selec_Channel(0,1,0);
                init_ADC();
                Tx(16);         //Bass

                selec_Channel(0,1,1);
                init_ADC();
                Tx(17);         //Treble

                selec_Channel(1,0,0);
                init_ADC();
                Tx(18);         //Panning

                RA5=0;

                selec_Channel(1,1,0);
                init_ADC();
                Tx(80);         //Mute

                selec_Channel(1,1,1);
                init_ADC();
                Tx(81);         //Solo
        }
}

void init_ADC()
{
        ADON=1;
        ADGO=1;
        while(ADGO)
                continue;
}

void selec_Channel(int a,int b,int c)
{
        CHS2=a;
        CHS1=b;
        CHS0=c;
}

void Tx(int t)
{
        TXEN=1;                 //Habilitar transmision

        TXREG=0xB0;             //Control Change Message Ch1
        DelayMs(1);
        TXREG=t;                //t = Número de Controlador
        DelayMs(1);
        TXREG=(ADRESH);                 //Valor
        DelayMs(1);
}
```

REFERENCIAS BIBLIOGRÁFICAS

- Huber, David Miles, *The MIDI Manual,* primera edición, SAMS.
- MIDI Manufacters Association, http://www.midi.org/.
- 130 años de música electrónica, http://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_04_05/io3/public_html/historia.html.
- http://www.yio.com.ar/secciones.php?name=Sections&op=viewarticle&artid=6.
- Control Surfaces Reviews & Features, http://emusician.com/controlsurfaces.
- http://mixonline.com.
- http://www.tascam.com/Products/US-224.html.
- www.logiq-electronics.com.
- MIDI controllers, http://www.synthzone.com/ctrlr.htm.
- http://www.zzounds.com.
- www.computermusic.co.uk.
- http://www.behringer.com.
- All You Need To Know About ASIO, http://www.soundblaster.com/resources/read.asp?articleid=53937&page=1&cat=2.
- MIDI 1.0 Specification Message Summary, http://www.harmony-central.com/MIDI/Doc/table4.html.

- Computación aplicada a la música, http://musica.unq.edu.ar/personales/odiliscia/papers/midi-re.htm.
- MIDI is the language of gods, http://www.borg.com/~jglatt/.
- Guide to the MIDI Software Specification, http://www.somascape.org/midi/tech/spec.html.