DAT330 – Principles of Digital Audio
Cogswell Polytechnical College
Spring 2009

Week 9 – Class Notes

Audio Interconnection

Audio Interfaces

The most fundamental interconnection in a digital studio is that between one device to another. Digital connections do not posses the degradation from AD/DA conversions that is common with analog connections. Digital data in order to be conveyed must have a data communication channel, a common clock synchronization, as well as an audio format that is recognized by both the transmitting and receiving devices.

Data flow is usually unidirectional, directed point-to-point, and runs continuously without handshaking. Two or more audio channels, as well as auxiliary data can be conveyed (serially). The data rate is determined by the signal's sampling frequency, word length, number of channels, amount of auxiliary data, and modulation code. If proper care and correct operation are taken, the received data will be identical to the transmitted data.

Successful transmission of serial data over a coaxial cable is determined by the cable's attenuation at one-half the clock frequency of the transmitted signal. In other words, the maximum length of a cable can be gauged by the length at which the cable attenuates the frequency of half the clock frequency by 30 dB. Professional interfaces allow cable lengths of up to 300m, while consumer cables may be limited to 10m or less. Fiber cables are less affected by length loss, and as such allow for longer cable runs.

SDIF-2 Interconnection

The Sony Digital InterFace (SDIF-2) protocol is a single-channel interconnection protocol used in some professional products. For example, it can be used to digitally transfer data from recorder to recorder. Multiple audio channels require multiple unbalanced BNC coaxial connections. In addition, a separate BNC connection for word synchronization is used. Any sampling frequency can be used and the signal is structured as a 32-bit word. The MSB through bit 20 are used for digital audio data in NRZ coding. Data rates are dependent on the sampling frequency; for example, at a sampling frequency of 44.1 kHz a data rate of 1.21 Mbps is achieved, while at 48 kHz 1.53 Mbps can be attained.

Bits 21-29 are control (user) words. Bits 21-25 are held for future expansion, while bits 26-27 are for ID determined at the ADC point. Bit 28 is the dubbing prohibition bit, and bit 29 is a block flag bit signifying the beginning of an SDIF-2 block. Finally, bits 30-32 form a synchronization pattern.

Direct Stream Digital (DSD) audio data, as used in SACD, can be conveyed by either SDFI-3 or MAC-DSD. SDIF-3 is very physically similar to SDIF-2. MAC-DSD (Multi-channel Audio Connection for DSD) is a multichannel interface for DSD data for professional applications. Instead of using a coaxial cable, it uses an ethernet cable interconnection. MAC-DSD can transfer up to 24 channels of DSD audio in a bidirectional manner.

Alesis Multichannel Optical Digital Interface

Also known as ADI or Lightpipe interface. Conveys eight channels of digital audio serial information along a single optical fiber over short distances. This proprietary protocol supports 24-bit samples, multiple sampling frequencies, and is self clocking. The nominal clock frequency is 12.288 MHz at a sampling frequency of 48 kHz. Plastic fiber optical cable runs of 33 feet can be used, while optical glass fiber can extend the cable run to 100 feet. Lightpipe can be used to interconnect recorders, DAW's, mixers, keyboards, effects processors, and other devices.

AES3 (AES/EBU) Professional Interface

This standard established by the Audio Engineering Society is a serial transmission format for linearly represented digital audio data. It permits the transmission of two-channel digital audio information, including non-audio data. The specifications of the standard can be adapted for diverse applications. For example, multichannel audio and higher sampling frequencies can be supported.

The format is intended to convey data over distances of up to 100 meters without equalization, while longer distances can be attained with equalization. Both channels are multiplexed and are self-clocking and self-synchronizing. The format is independent of the sampling frequency, thus different sampling frequencies can be used. AES3 alleviates polarity shifts between channels, channel imbalances, absolute polarity inversion, gain shifts, as well as analog transmission problems such as hum and noise pickups, and high frequency loss.

Biphase mark code is a self-clocking code used as a binary modulation channel code used to convey data over AES3.

Low-capacitance cable is preferred for this format and shielding is not critical. Input (female) and output (male) connectors use an XLR type connector with pin 1carrying signal ground, and pins 2-3 carrying the unpolarized signal. For a multichannel version of AES3, a 50-pin D-sub type connector is used to transmit 16 channels.

AES10 (MADI) Multichannel Interface

MADI is an extension of the AES3 protocol to provide a standard means of connecting multichannel digital audio equipment. It supports up to 56 channels of linearly represented, serial data to be conveyed along a single length of BNC terminated cable for distances up to 50 m. Audio samples of up to 24-bit are permitted, and MADI is transparent to the AES3 protocol (it can convey the same information). By using MADI interconnection, it is possible to performa a single A/D conversion and then do subsequent processing through multiple devices without leaving the digital domain. MADI interconnection requires two cables plus a master synchronization signal for up to 56 channels.

MADI is designed to run asynchronously. To operate as such, a MADI receiver must extract timing information from the transmitted data so the receiver's clock can be synchronized. Thus, a master synchronization signal must be applied to all interconnected MADI transmitters and receivers.

The MADI interconnection is designed to be used with a standard 75 Ω video coaxial cable with BNC connector terminations, although fiber optic cables can also be used for longer distances (up to 2 km).

The audio data frequency can range from 32-48 kHz. Higher sampling frequencies could be supported by transmitting at a lower rate and using two consecutive MADI channels to achieve the desired sampling rate.

S/PDIF Consumer Interconnection

This standard is similar to the AES3 standard, and in some cases professional and consumer equipment can be directly connected. However, this is not recommended because important differences exist in the electrical specification, and in the channel status bits, so unpredictable results can occur when the protocols are mixed. Devices designed to read both AES3 and S/PDIF data must reinterpret channel status block information according to the professional or consumer's status is the block's first bit.

Byte 1 defines the category code that determines the transmitting format for different equipment (CD, MiniDisc, DAT, synthesizer, broadcast reception, etc).

The consumer interface does not require a low impedance balanced line, as does the professional standard. Instead a single-ended 75 Ω coaxial cable is used to transmit over a maximum of 10 m. Video-type cables are recommended to ensure adequate transmission bandwidth. Alternatively, some consumer equipment uses an optical Toslink connector and plastic fiber-optic cable over distances of less than 15 m. Glass fiber cables and appropriate code/decode circuits can be used for distances over 1 km. A variation of this standard describes a multichannel interface and low bit-rate data (Dolby Digital, DTS, MPEG) can be substituted for the PCM data. This data reduction allows for multichannel (such as 5.1 channels) support with only a two-channel optical or coaxial interface. Receiving equipment reads the channel status information to determine the type of bistream and directs it to the adequate decoder.

Serial Copy Management System

SCMS circuits are designed on many consumer recorders to limit the number of copies that can be derived from a recording. Users can make digital recordings of prerecorded, copyrighted work, but the copy itself cannot be copied.

SCMS is a “fair” solution because it allows users to make digital copies of purchased software. But on the other hand, it might prohibit the recopying of original recordings, which is a legitimate use. The use of SCMS is mandated in the US by the Audio Home recording Act of 1992, as passed by Congress to protect copyrighted works.

The SCMS circuit found in consumer-grade recorders with S/PDIF interfaces, is not present in professional AES3 interfaces. Equipment that does not store, decode, or interpret transmitted dat, is considered transparent and ignores SCMS flags. Digital mixers, filters, optical disc recorders, and tape recorders require different interpretations of SCMS.

By law, the SCMS circuit must be present in all consumer recorders with the S/PDIF interconnection. However, some professional recorders (upgraded consumer models) can also contain SCMS circuitry. If recordists use the S/PDIF interface, copy-inhibit flags are sometimes inadvertently set, leading to problems when subsequent copying is needed.

AES11 Digital Audio Reference Signal

This standard specifies criteria for synchronization of digital audio equipment in studio operations. It is important for interconnected devices to share a common timing signal so that individual samples are processed simultaneously; timing inaccuracies can lead to increased noise, as well as clicks and pops in the audio signal. All interconnected devices must be synchronized in both frequency and phase, and be SMPTE time synchronous as well. In addition, interconnected devices must operate with the same sampling frequency and the bits in the sending and receiving signal must begin simultaneously. Because most digital audio data streams are self-clocking, synchronization is relatively easy as long as the receiving circuits read the incoming modulation code, and reference the signal to an internal clock. Also, this can be aided with the addition of an independent synchronization signal.

For multiple interconnected devices, a common synchronization clock is recommended. External synchronizers must read SMPTE timecode and provide the time synchronization between devices.

The AES11 DARS provides a clocking signal with high frequency stability for jitter regulation. With this reference, any sample can be time aligned with any other sample, or with timecode reference, aligned to specific video frame edge. AES11 uses the same format and electrical configuration, and connectors as AES3. This format is also known as “AES3 black” because it looks like an AES3 signal with no audio data present.

A separate word clock cable is run from the reference source to each piece of digital equipment through a star-type architecture to the reference clock inputs of all digital audio devices. If ADC/DAC have no internal clocks, they derive their clocks from DARS.

AES18 User Data Channels

This standard describes a method for formatting the user data channels found within the AES3 interface. It conveys text and other message data that might be related or unrelated to the audio data. The user data channel is a transparent carrier providing a constant data rate when the AES3 interface operates at an audio sampling frequency of 48 kHz. Messages are sent as one or more data packets, each with the address of its destination; receivers only read messages addressed to them. A packet is comprised of an address byte, control byte, address extension byte, and an information field that is no more than 16 bytes. Typical user bit applications can include messages such as scripts, subtitles, editing information, copyright, performer credits, switching instructions, and other annotations.

AES24 Control of Audio Devices

The AES24 standard describes a method to control and monitor audio devices via digital data networks; it is a peer-to-peer protocol, so that any device may initiate or accept control and monitoring commands. The standard specifies the formats, rules, and meanings of commands, but do not define the physical manner in which they are transmitted; thus, it is applicable for a variety of communication networks. Using this format allows for devices from different manufacturers to be controlled and monitored with a unified command set within a standard format. Each AES24 device is uniquely addressed by the network transport network software, such as a port number. Devices may be signal processors, system controllers, or other components with or without user interfaces. Object-to-object communication is possible because each object has a unique address. Object addresses have two forms: an object path (preassigned text string) and an object address (48-bit unsigned integer).

Messages pass from one object to another, and all messages share a common format and set of exchange rules.

Message exchange example:

Complex networks are possible, as long as each device is reachable by any other device.

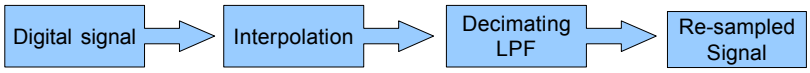
Sample Rate Converters

Although 44.1 and 48 kHz sampling frequencies are most common, there are many other sampling frequencies that are used in different applications. Devices cannot be connected if their sampling rates are different. Even if the sources where recorded with a common sampling frequency, their data streams can asynchronous and may differ by a few Hertz. Thus, they must be synchronized to an exact common frequency. If no proper care is taken, jitter may arise and degrade the signal.

A synchronous sample rate converter converts one rate to another using an integer ratio; the output is fixed in relation to the input rate. An asynchronous sample rate converter can accept dynamically changing input sampling frequency, and output a constant and uninterrupted sampling frequency, at the same or different frequency. The input and output rates can have an irrational ratio relationship. In other words, the input and output rates are completely decoupled.

Conceptually, a sample rate convertor works like so:

In practice, this procedure is done digitally though interpolation and decimation.



Musical Instrument Digital Interface (MIDI)

This protocol is widely used to interconnect electronic music instruments and other audio production equipment, as well as music notation devices. MIDI is not a digital audio interface because it does not transmit audio signals, rather, it conveys control information as well as timing and synchronization information to control musical events and system parameters of music devices.

MIDI files are small in size and can be streamed with low overhead. However, the user must have a MIDI synthesizer installed in order to read the file. MIDI is asynchronous and unidirectional serial interface. The data format uses 8-bit bytes, and most messages are conveyed as one-, two-, or three-word sequences.

The first word is a status word describing the type of message and channel number (up to 16). Status words begin with a 1 and may convey messages such as “note on” or “not off”. Subsequent words are data words. Data words begin with 0 and contain message particulars.

MIDI connections typically use a 5-pin DIN connector, but XLR connector are also used. A cable length of 50 ft maximum can be used without any issues.

MIDI Messages

MIDI information is transfered between controllers and synthesizers as a sequence of bytes. A byte is a number between 0 and 255. MIDI bytes are split into two basic groups of command bytes and data bytes:

	data bytes	command bytes
decimal:	0 -- 127	128 -- 255
binary:	0xxxxxxx	1xxxxxxx
hexadecimal:	00 -- 7f	80 -- ff

It is often useful to view MIDI bytes in binary or hexadecimal form, so the table above lists both alternate notations for the numbers in the decimal range. In binary notation, the number 0 -- 127 all start with a zero in the top bit of the MIDI byte. Commands bytes likewise start with a 1 bit in the highest position in the byte.

Data bytes are used to transmit things such as the note number, attack velocity, piano pedal positions, volume, pitch bend, and instrument numbers. Command bytes are the first byte of MIDI messages which are then followed by a fixed number of MIDI data bytes. For example, the MIDI message to turn on the note Middle C would be:

128 60 100

Notice that the first number is in the command byte range of 128-255. This must mean that it is a command -- which it is. Command byte 128 means turn on a note. This command requires two data bytes immediately following it. The first data byte indicates which note key to turn on, and the second data byte indicates how loud to play the note. In this case, the key number is 60, which is the key number for the pitch middle C (C4). The second data byte is called the attack velocity. A value of 100 for the attach velocity would be maximum if it were 127, or minimum if it were 1.

MIDI Message Types

MIDI commands can be further decomposed into a command type nibble (four bytes) and a channel number nibble. In the case of the MIDI note-on message given above, here is the binary form of the command byte 128: 10000000. Split this number into four byte segments: 1000,0000. The first nibble is 1000 which incidates the command type -- note-on command. The second nibble is 0000 which indicates the MIDI channel number the note is playing on. In this case 0000 indicates the first MIDI channel, which is also called MIDI chanel one. It is usually most convenient to view MIDI command bytes in hexadecimal format. A hexadecimal digit is equal to a single nibble which is four digits of a binary number. Here is the MIDI command byte 128 displayed in hexadecimal and binary forms:

	command nibble	channel nibble
hex	8	0
binary	1000	0000

A MIDI channel is used to play multiple instruments at the same time. Each instrument playing simultaneously would occupy a separate channel on the synthesizer. The instrument playing on a channel can be changed.

Here is a table of conversions between binary, hexadecimal and decimal forms of numbers. This table is useful to keep track of the different ways of viewing numbers in MIDI bytes:

Binary	Hex	Dec	Binary	Hex	Dec	Binary	Hex	Dec	Binary	Hex	Dec
0000	0	0	0100	4	4	1000	8	8	1100	C	12
0001	1	1	0101	5	5	1001	9	9	1101	D	13
0010	2	2	0110	6	6	1010	A	10	1110	E	14
0011	3	3	0111	7	7	1011	B	11	1111	F	15

MIDI command nibbles must start with a 1 in binary format, therefore there are three binary digits left over for specifying a command. This means that there are eight possible MIDI command types. These commands are listed in the table below:

command nibble	*command name*	*data bytes*	*data meaning*
8	note-off	2	key #; release velocity
9	note-on	2	key #; attack velocity
A	aftertouch	2	key #; key pressure
B	control-change	2	controller #; controller data
C	patch-change	1	Instrumnet #
D	channel-pressure	1	channel pressure
E	pitch-bend	2	lsb; msb
F	system-message	0 or variable	none or sysex

Open Sound Control (OSC)

Protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology. Bringing the benefits of modern networking technology to the world of electronic musical instruments, OSC's advantages include interoperability, accuracy, flexibility, and enhanced organization and documentation.

OSC has some of the following features:

1. Open-ended, dynamic, URL-style symbolic naming scheme
2. Symbolic and high-resolution numeric argument data
3. Pattern matching language to specify multiple recipients of a single message
4. High resolution time tags
5. "Bundles" of messages whose effects must occur simultaneously
6. Query system to dynamically find out the capabilities of an OSC server and get documentation

There are dozens of implementations of OSC, including real-time sound and media processing environments, web interactivity tools, software synthesizers, a large variety programming languages, and hardware devices for sensor measurement. OSC has achieved wide use in fields including computer-based new interfaces for musical expression, wide-area and local-area networked distributed music systems, inter-process communication, and even within a single application.

The advantages of OSC over MIDI are primarily speed and throughput; internet connectivity; data-type resolution; and the comparative ease of specifying a symbolic path, as opposed to specifying all connections as 8-bit numbers. OSC messages arrive as fast as the underlying network stack can transfer them, and can be delayed to take effect at a specific time, whereas MIDI ensures synchronicity of messages by transferring them at a specific clock rate.

OSC Syntax

All OSC data is composed of the following fundamental data types:

1. int32: 32-bit big-endian two's complement integer
2. OSC-timetag: 64-bit big-endian fixed-point time tag, semantics defined below
3. float32: 32-bit big-endian IEEE 754 floating point number
4. OSC-string: A sequence of non-null ASCII characters followed by a null, followed by 0-3 additional null characters to make the total number of bits a multiple of 32. (OSC-string examples) In this document, example OSC-strings will be written without the null characters, surrounded by double quotes.
5. OSC-blob: An int32 size count, followed by that many 8-bit bytes of arbitrary binary data, followed by 0-3 additional zero bytes to make the total number of bits a multiple of 32.

OSC Packets

The unit of transmission of OSC is an OSC Packet. Any application that sends OSC Packets is an OSC Client; any application that receives OSC Packets is an OSC Server. An OSC packet consists of its contents, a contiguous block of binary data, and its size, the number of 8-bit bytes that comprise the contents. The size of an OSC packet is always a multiple of 4.

The underlying network that delivers an OSC packet is responsible for delivering both the contents and the size to the OSC application. An OSC packet can be naturally represented by a datagram by a network protocol such as UDP. In a stream-based protocol such as TCP, the stream should begin with an int32 giving the size of the first packet, followed by the contents of the first packet, followed by the size of the second packet, etc.

The contents of an OSC packet must be either an OSC Message or an OSC Bundle. The first byte of the packet's contents unambiguously distinguishes between these two alternatives.

OSC Messages

An OSC message consists of an OSC Address Pattern followed by an OSC Type Tag String followed by zero or more OSC Arguments.

OSC Address Patterns

An OSC Address Pattern is an OSC-string beginning with the character '/' (forward slash).

OSC Type Tag String

An OSC Type Tag String is an OSC-string beginning with the character ',' (comma) followed by a sequence of characters corresponding exactly to the sequence of OSC Arguments in the given message. Each character after the comma is called an OSC Type Tag and represents the type of the corresponding OSC Argument. (The requirement for OSC Type Tag Strings to start with a comma makes it easier for the recipient of an OSC Message to determine whether that OSC Message is lacking an OSC Type Tag String.)

OSC Type Tags:

1. i: int32
2. f: float32
3. s: OSC-string
4. b: OSC-blob