AN INVESTIGATION INTO THE USE OF IEEE 1394 FOR AUDIO AND CONTROL DATA DISTRIBUTION IN MUSIC STUDIO ENVIRONMENTS.

A thesis submitted in fulfilment of the

requirements for the degree of

MASTER OF SCIENCE

of

RHODES UNIVERSITY

By

ROBERT ALAN LAUBSCHER

February 1999

Abstract

This thesis investigates the feasibility of using a new digital interconnection technology, the IEEE-1394 High Performance Serial Bus, for audio and control data distribution in local and remote music recording studio environments. Current methods for connecting studio devices are described, and the need for a new digital interconnection technology explained. It is shown how this new interconnection technology and developing protocol standards make provision for multi-channel audio and control data distribution, routing, copyright protection, and device synchronisation. Feasibility is demonstrated by the implementation of a custom hardware and software solution. Remote music studio connectivity is considered, and the emerging standards and technologies for connecting future music studio utilising this new technology are discussed.

Acknowledgements

I gratefully acknowledge the significant help and guidance received from Richard Foss, the supervisor of my work. I would also like to thank the people from industry who have shown interest in this work and provided answers to my many questions, especially Bob Moses, Robert Sloan, Yoshi Sawada, and Dick Scheel.

Table of Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

To produce the music that has become so much a part of our everyday lives, composers, producers and recording engineers utilise a variety of electro-acoustic devices, connected via a plethora of cabling. This collection of devices, and the accompanying mass of interconnecting cables, constitute what is known as a recording studio. Part of the complexity of sound production in a studio arises from the many different standards used for device interconnectivity. Many studio devices can be controlled via a software and hardware standard called MIDI (Musical Instrument Digital Interface). Devices utilising MIDI for communication, exchange MIDI messages. MIDI messages may communicate performance data (instructing sound generators to produce sounds), configuration data (used for setting the parameters of devices), or timing information.

There are different types of recording studios, each geared towards the final product that emerges. The type of studio considered in this thesis is the kind referred to as a "project studio". A "project studio" will typically enable a composer, producer or recording engineer to produce music, perhaps combining acoustic and synthesised sounds. The advent of MIDI and MIDI sequencers enables a studio user to create a musical space of rich timbres. MIDI can further be utilised to automate studio devices and provide control of devices from a central point. MIDI is recognised to have limitations, in particular its low bandwidth and resulting transmission latencies. The widespread adoption of MIDI however, requires users to compromise. To overcome MIDI bandwidth problems, more MIDI ports can be used, but this has the effect of adding yet another cable.

A PC often plays an integral part in a project studio, with contemporary software being able to facilitate the generation, recording, and processing of sounds. Such a PC is sometimes referred to as a digital workstation, this name being indicative of the trend in recording studios towards a completely digital environment. The nature of a project studio and the role of a PC workstation in this environment are discussed in chapter 2.

A new interconnection technology, called "Firewire" by its inventors at Apple Computer, and later adopted by the Institute of Electronic and Electrical Engineers as IEEE 1394-1995 Standard for a High Performance Serial Bus, has many characteristics that would provide a possible solution for studio device interconnectivity.

The 1394 bus[1] can be considered as a hybrid interconnectivity and networking technology. Devices that utilise the 1394 technology are connected by simple point-to-point cables in a flexible topology. The combined point-to-point connections between nodes are utilised by application protocols as a high performance network bus.

At a glance, the 1394 bus offers the following features:

- Up to 400Mbps data transfer rate.
- Hot plugging and true plug and play, including auto-configuration and optimisation.
- Power distribution in the cable – if a node's power requirements are modest, they may draw their power to operate directly from the 1394 bus.
- A verified packet transmission mechanism termed asynchronous transactions.
- A guaranteed latency, reserved bandwidth data transmission mechanism, termed isochronous transactions.
- Mechanisms for broadcasting, multi-casting and single-destination data transmission.

Isochronous transactions are the key feature of the 1394 bus that has given rise to the 1394 bus being called "the multimedia connection". The guaranteed low transmission latency of isochronous transactions enables "just in time" delivery of multimedia data. The distribution of a common clock, and bandwidth reservation mechanisms further enhance the usability of isochronous transactions for multimedia applications. The complete operation of the 1394 bus is described in chapter 3.

Realising the potential for audio and MIDI data distribution using isochronous transactions, Yamaha Corporation developed the mLAN (music Local Area Network) specification. The mLAN specification extended an existing international standard (IEC 61883) for digital video distribution using the 1394 bus (an existing application), to accommodate audio and MIDI

[1] The IEEE 1394-1995 High Performance Serial Bus will be referred to as the 1394 bus in this thesis.

data. The mLAN specification has evolved into the Audio and Music Data Transmission Protocol (A/M protocol), which is currently being finalised for incorporation into the IEC 61883 standard. In chapter 4, the details of the A/M protocol are shown, and it is also discussed how developing standards allow for audio and MIDI routing between devices, copyright protection, and seamless integration with a PC workstation.

To determine the hardware and software elements required by studio devices to utilise the 1394 bus, two 1394 bus nodes were implemented that demonstrated the transmission of A/M protocol formatted digital audio using the 1394 bus. In chapter 5 it is described how two evaluation module boards, a Vitana 1394 research development kit incorporating the Philips 1394 "AV" chipset, and a Motorola 56002 evaluation module, are combined to provide this functionality. The goal was to develop a solution capable of retrofitting an existing MIDI controllable studio component, such as a synthesiser, to have 1394 bus capabilities. A device such as a synthesiser would require full-duplex operation – be able to receive MIDI data and transmit generated audio data. The link layer controller (a necessary 1394 bus hardware component) used in this implementation was however only half-duplex, a typical characteristic of link layer controllers at the time of this implementation. After this implementation however, a full-duplex version of the link layer controller has become available. Given this fact, the feasibility demonstrated by this implementation, and the industry support (particularly from Yamaha), the prospect of 1394 connectivity for audio and MIDI distribution in studio environments is exciting.

As exciting as the possibilities are, widespread adoption of 1394 connectivity in studio environments has not yet occurred. Thus, it may be premature to consider connecting future remote studios utilising 1394 connectivity. The possibilities however, given the status of enabling standards and technologies, present a compelling scenario for remote device sharing and project collaboration. This thesis concludes in chapter 6, by a brief look at the status of developing standards and technologies that would enable connectivity of remote studios utilising the 1394 bus.

Chapter 2

2 Audio and control data distribution in local studio environments.

Producers, sound engineers, and composers are some of the people who use music studios to create the music that has become a part of our everyday lives. Music recorded in studios range from popular bands and classical orchestras to the music we hear in films, videos and advertisements. In music studios, heterogeneous arrays of audio devices are connected together, often in a "spaghetti" of wiring interconnections. Using these connections, audio and control data are distributed amongst devices. Often a computer aids the process of music recording, playing an indispensable role in co-ordinating sound generation and device control from a central point. Figure 1 below illustrated the cabling requirements for a small project studio.

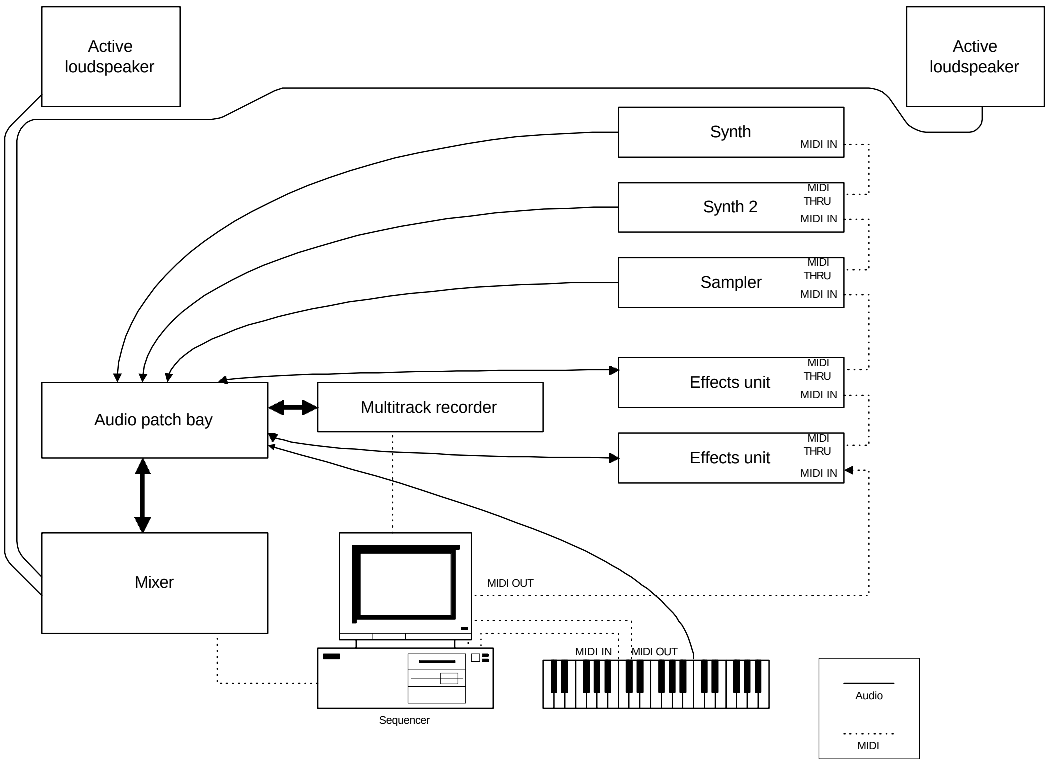


Figure 1: Cabling requirements of a small studio

This chapter gives an overview of technologies that enable audio and control data distribution in local studio environments.

2.1 Audio flow in studio environments

The final product of a music studio is typically an audio consumer product such as a compact disc, or the audio component to accompany other media such as video or film. Indeed, these final products distinguish the two major types of commercial music production studios – music recording studios and music post-production studios, respectively. Music recording studios are oriented towards recording individuals or groups of performers, whereas post-production studios typically facilitate the production of audio tracks to accompany other media.

The recording process in both types of studios can be described as capturing and manipulating sounds in an electro-acoustic environment. Sounds are captured from sound generators. The range of sound generators is vast, and includes the human voice, acoustic instruments, electronic instruments, analog and digital synthesisers, and samplers.

The sound produced from sound generators is typically fed to the inputs of a mixer device. The mixer device allows the incoming audio signals to be mixed and routed to storage and processing devices, and monitored through loudspeakers. Multitrack recorders can store separate tracks of audio simultaneously. These recorders may be analog or digital. Analog multitrack recorders write the audio data onto magnetic tape. Digital recorders may use hard drives or magnetic tape. Once recorded, audio tracks may be replayed through the mixer and processed further.

Processing devices manipulate audio signals, often adding effects in an attempt to recreate the feel of a natural acoustic space such as a large hall. Other sound processing devices perform functions such as compressing or expanding the dynamic range, shifting the pitch, injecting harmonics or frequency equalisation. All this to produce a final mix that satisfies the ears of the intended listener.

In a studio environment, as in all audio equipment, sounds are internally represented as a varying electrical voltage. The variance of the voltage at any time corresponds to the physical air-pressure variance of the corresponding acoustic sound. A microphone can be used to create this electrical signal from an acoustic sound generator, and a loudspeaker to recreate the air-pressure variance from the varying electrical signal. The recreated air-pressure variance is what we know as sound. Electronic instruments produce the electrically varying voltage representation directly. Of course, audio mixers, audio storage devices, and audio manipulation devices use an electrical representation. The electrical representation may be analog or digital. The digital representation is an alternate means to carrying the same information. Analog to digital converters (ADCs) are used to convert analog signals into the digital realm. DACs (digital to analog converters) perform the opposite function – recreating an audio signal from digital data.

A number of audio devices are inherently digital. Sound processors often employ DSPs (digital signal processors) to process audio signals. Recorders and mixers too can be completely digital in nature. Keeping an audio signal completely digital from the beginning to end of a production allows the user to maintain consistent sound quality. Digital signals are not altered in the normal process of recording or transmission over interconnects. Converting the signal to and from the analog domain introduces noise and distortion. By using digital interconnects, a receiving device receives a cloned copy of the original information.

A problem that is encountered in studio environments is how audio is routed, stored, and transmitted between the various devices. The problem arises due the many different interconnection standards, analog and digital, that devices employ. Consider how audio, analog and digital, is distributed in an audio environment.

2.1.1 Analogue audio technologies

In a studio environment, analog audio signals are distributed between devices using unbalanced or balanced cabling. Balanced cabling uses three conductors and a shield. Unbalanced cabling uses two conductors and a shield. In both instances, one conductor is a “ground” that is typically connected to the shield. Balanced cabling uses a differential transmission scheme, providing greater noise immunity than unbalanced cabling. The assumption is that any electrical interference will occur on all conductors equally. By detecting the difference between signal levels on the conductors, this noise can largely be eliminated. Balanced cabling is particularly suitable for long cable runs.

There are a number of different connectors that are used – RCA, jack, and XLR types. Balanced cabling is typically terminated in XLR, and sometimes jack connectors. Unbalanced cabling can use RCA or jack connectors.

To route analog audio between devices in a studio environment, a physical signal path needs to be established. Patch bays are used to allow flexible audio-path configurations. The audio inputs and outputs of devices are connected to the patch bay. The user can then place patch leads between the inputs and outputs to configure the required audio paths. Patch bays are typically mechanical; i.e. requiring patch leads to be inserted, but may also be solid-state. Solid-state patch bays allow audio routes to be established under control of software on a host device, such as a computer.

There are a number of conditions that occur in analog systems resulting in the degradation of the audio signal. Some of these conditions are wow, flutter, channel crosstalk, particulate noise, print-through, modulation noise, HF-squashing, azimuth error and interchannel phase errors [Rumsey 1993]. The end signal carries the sum of degradations that have been incurred at each stage through which it has passed. Since the degradations cannot be separated from the signal, nothing can be done about them. A limit is imposed on the number of stages through which a signal can pass. Consequently, each piece of equipment must be far better than necessary to produce an acceptable signal at the end. As a result, the equipment is more expensive than their digital counterparts.

2.1.2 Digital audio technologies

Rumsey suggests two answers to the question of "Why digital?"

- The reproduction quality depends only on the conversion process and not on the medium.
- Conversion to the digital domain allows opportunities denied to analog signals.

2.1.2.1 Analog to digital conversion

Before considering the conversion process, recall what a digital signal is – an alternative means of carrying the same analog information. Although there are a number of ways in which analog signals can be represented digitally, a system known as pulse code modulation (PCM) is most commonly used for audio. PCM numerically quantifies the audio waveform at discrete intervals. This process is known as sampling and quantizing. Sampling is a periodic measurement, or time discretization of the waveform. Quantizing is assigning a numerical value to the waveform at that particular instant.

2.1.2.1.1 Sampling

The input waveform is required to be sampled at a precisely regular interval. This requirement is set so that subsequent reconstruction processes may be carried out. The foremost consideration for determining a suitable sampling rate is the highest frequency that is to be represented. Nyquist’s rule states that the sampling rate must be at least twice the highest input frequency. Conversely, the highest frequency which can be represented is half the sampling rate. If inputs higher than half the sampling rate are present, aliasing will occur. Aliasing has the effect of producing the wrong waveform in the reconstruction process. Consequently, a low pass filter limiting the input frequencies is required. The design of a suitable filter is provided by [Pohlmann 1985]. Acoustic sounds have an inherent frequency limit, as do our ears. 20kHz is considered an adequate limit.

In early digital audio systems, disk drives did not have the capacity to store long recordings. Attention was turned to video recorders and a scheme developed whereby binary data was conveyed in a pseudo-video waveform using black and white levels. The sampling rate of 44.1kHz was chosen since it is a common multiple of the two most popular used video formats – 525 lines at 60Hz and 625 lines at 50Hz [Rumsey 1993]. The 44.1kHz sampling rate came to be the sampling rate used for compact discs. A sampling rate of 32kHz is used for FM stereo broadcasting and also NICAM 728 stereo TV sound systems. The 48kHz sampling rate adopted in many professional devices allows for variable-speed operation and has a simple relation to PAL video timing. To summarise, digital audio has three sampling rates to support: 32kHz for broadcasting; 44.1kHz for compact disc; and 48kHz for professional use. The use of a 96kHz sampling rate is also now emerging in professional equipment.

It is important when reconstructing an analog signal to recover the exact sampling frequency. If the sampling frequencies at the input and output stages are different, the pitch of the reconstructed signal will not match that of the sampled signal. Sample rate conversion techniques may be used to convert audio data sampled at one frequency to another. If the final frequency is higher than the original, more samples need to be generated. This may be done by polynomial interpolation or by more sophisticated synthesis techniques. If the final frequency is lower, samples need to be removed. This may be done by simply removing every “nth” sample such that the required number are left over a set period of time, or also by using more sophisticated synthesis techniques.

When an analog signal is reconstructed, a process known as timebase correction is often used. Timebase correction ensures that the instants at which digital samples are converted to analog voltages are evenly spaced and at the correct sample rate. The reconstruction process recreates a smooth continuous voltage, joining the discrete sample points. If the waveform was originally sampled with “jitter” on the sampling clock, the output waveform will not be accurately reconstructed. “Jitter” refers to the delay or advance in the period between samples.

2.1.2.1.2 Quantization

Quantization is the process by which a numerical value is assigned to a signal voltage at a particular instant. The range of numerical values that are used determines the granularity or resolution of the quantization process. The range is determined by the number of bits used in encoding the numerical value in binary format. The minimum and maximum numerical values correspond to assigned minimum and maximum signal voltages. If the signal voltage exceeds the minimum or maximum, then clipping will occur.

Using 16 bits for digital audio encoding, as is done for compact disc, permits a range of 65535 values. This sets the dynamic range to 96dB, and the noise floor limit to 1 in 65535, or –96dB. Using 24 bits, as some professional audio products do, increases the dynamic range to 144dB and the noise floor limit to –144dB.

2.1.2.2 Digital audio transmission standards

There are a number of digital interconnections standards used for transmitting digital audio data in studio environments. The most commonly used are the AES3 and IEC958 formats known as “AES/EBU” and “SPDIF” respectively [Moses 1997]. The AES3 standard, intended for professional applications, was developed by the Audio Engineering Society (AES). Sony and Philips concurrently developed the SPDIF (Sony-Philips Digital InterFace) standard for consumer applications. Consequently, there are a number of similarities between the AES/EBU and SPDIF formats, but also significant differences that do not allow seamless interoperability. The development of these standards, and the involvement of international standards bodies in the process is described by [Rumsey 1993].

Common to both the AES/EBU and SPDIF, and some other digital formats is the definition of a 64-bit frame that is divided into 2 identically structured subframes. Each subframe consists of a four bit synchronisation preamble, four auxiliary bits which may be used for additional audio resolution, a twenty bit audio sample, a validity bit(V), a user bit (U), a channel status bit (C), and a parity bit (P).

The data is transmitted serially and is self-clocking. An encoding method known as "bi-phase mark" is used. This encoding method ensures that a level transition occurs at each bit, allowing clocking information to be extracted by the receiver and also allowing DC decoupling between the transmitter and receiver. The synchronisation preambles are defined such that they violate the bi-phase mark encoding method. As a result, the receiver can clearly identify the beginning of a channel transmission.

The V, U, C, and P bits carry auxiliary information. Although they are single bits, the V,U, and C bits in particular are components of a larger word that is transmitted serially. Sampling rates and audio word lengths are two of the attributes which can be conveyed by these words. The interpretation of these bits and the corresponding words they define is what distinguishes the different digital transmission standards. The physical interface for these digital transmission formats may be balanced, unbalanced, co-axial, or optical.

These formats transfer a stereo digital audio signal between two devices using a fixed point-to-point connection. The connection may be optical or electrical. Each stereo pair of signals must be transferred over an individual cable, adding expense, and complexity for multiple channels.

A number of proprietary multi-channel digital audio systems have been invented which overcome the problems associated with AES3 and IEC958 when used for multi-channel operation. These include ADAT and TDIF developed by Alesis and Tascam respectively. ADAT and TDIF provide an eight channel digital audio connection that is typically used to connect devices such as digital mixers to digital multi-track recorders.

A non-proprietary multi-channel interface originally developed by Sony, Neve, Mitsubishi, and Solid State Logic and later adopted by the AES, called MADI (Multichannel Audio Digital Interface) in essence extends AES3 to 56 channels [Jacobs, Anderson 1994]. The standard associated with MADI is AES10. Each channel is transmitted serially and contains bits 4 to 31 of the AES3 channel frame. An important difference between AES3 and AES10 is that the sampling rate is not directly conveyed in the transmission of MADI data. A separate cable transmits the clock reference for MADI formatted digital audio [Lidbetter 1989].

It is common, and desirable in recording studios to use a single digital clock reference. Using a single clock reference maintains sample accurate synchronisation between all devices, eliminating the need for sample rate recovery and sample rate conversion. Often, professional digital devices will enable the clock rate to be provided externally. A clock generator is used and the signal is connected to the external clock inputs of all the capable devices. This is known as "house sync". The problem with "house sync" is the additional complexity and extra specialised cabling that is often required. An overview of synchronisation in digital studio environments is provided by [Shelton 1989].

2.2 Control Data Flow

2.2.1 The MIDI standard for music studio control

Foss describes the introduction of MIDI (Musical Instrument Digital Interface) as being the most significant factor contributing towards music studio control in the last decade [Foss 1996]. The MIDI standard document published by the International MIDI Association (IMA) describes a hardware standard and software protocol that allows devices to communicate performance and configuration data between each other [IMA 1989].

The hardware standard defines a 31.25kbaud unidirectional serial connection. Data transmission takes place via a standard UART (Universal Asynchronous Receiver/Transmitter) establishing a current loop that is optically coupled to the receiver. Typically MIDI devices will have MIDI “in”, “out” and “thru” ports. The “in” port receives data from another device's “out” port. The “thru” port repeats data appearing on the “in” port and allows devices to be daisy chained.

The MIDI software standard is based upon the transmission of MIDI messages. These messages are divided into five groups:

- Channel voice messages
- Channel mode messages
- System common messages
- System realtime messages
- System exclusive messages

Full details of all these messages are given by [IMA 1989]. Included in the software protocol is provision for a synchronisation method. The messages associated with synchronisation are discussed in section 2.2.2. Before that, consider how performance data and control data are distributed between devices using MIDI messages.

2.2.1.1 Performance data distribution

A musician playing some sort of MIDI-compatible controller typically produces performance data in the form of channel voice messages. The performance data can be received by a number of synthesisers (that have been daisy-chained together) if they have been “tuned” to the particular channel. The synthesisers can respond by producing pitched notes, drum sounds, or sound effects.

Channel voice messages consist of a ‘status’ byte followed by one or more bytes containing data pertinent to the message. The status byte is comprised of a 4-bit command identifier followed by a 4-bit channel number enabling the use of 16 channels. The most common channel voice message commands are “note on” and “note off”. The “note on” message is followed by two bytes of data indicating the pitch and velocity of the performed note. Other channel voice messages are controller messages allowing parameters such as panning and chorus depth to be altered in real-time, program change messages allowing the musician to select a particular sound, and also pitch bend, channel aftertouch, and poly pressure, messages conveying subtle nuances in performance. To distinguish “data” bytes from the “status” bytes, “status” bytes have the most significant bit set, and “data” bytes do not. This imposes a range of 127 possible values on the “data” bytes.

MIDI did not stop at simple controller-synthesiser interaction. Channel voice messages, as well as other MIDI messages, can be read by a MIDI interface on a computer. Software sequencers allow multiple tracks of MIDI messages to be recorded and then simultaneously played back later. Often a sequencer will allow for the editing of MIDI performance messages in a meaningful format, such as piano-roll or conventional music notation. Other editing functions are also commonly provided such as note transposition, velocity editing (loudness editing) and timing adjustments. The sequencer may use internal or external timing methods to determine the time at which MIDI messages were recorded and when they should be played back. Complex musical compositions can be created that make use of the multiple-timbres that MIDI compatible synthesisers can produce. The design of a MIDI sequencer is provided by [Garvin 1987].

The sequencer plays a powerful role in a music recording environment, not only for the creation of music compositions, but also for the centralised control of studio equipment such as sound processors, mixers and multi-track recorders.

2.2.1.2 Configuration and control data distribution

After the introduction of MIDI, many manufacturers of studio equipment incorporated MIDI control into their products. Consider how sound processors, mixers, and multi-track recorders utilise MIDI, and how in all these cases, the control can be initiated from a sequencer.

Many sound processors have the ability to perform multiple user-selectable processing functions. Some sound processors allow the parameters of their particular processing function to be altered in real-time, under MIDI control. These sound processors will typically accept MIDI controller messages (part of the channel voice messages). Other sound processors allow the user to take a snapshot of the settings of the device and build up a library of states that can then be loaded back at a later stage. The MIDI software protocol includes a group of messages called System Exclusive Messages, these SysEx messages (as they are commonly known) allow for the transmission of manufacturer-specific data. These commands may be of considerable length – such as all the parameters of a sound processing device. A component of a sequencer, often referred to as a librarian, assumes the function of co-ordinating the storage and transmission of these SysEx messages to properly configure devices before performance data is played.

MIDI controllable mixers range in complexity from simple MIDI controlled volume fading to full MIDI controlled mixing automation. In the latter case, as implemented by Yamaha's O2R series of mixers, all settings of the mixer can be controlled using MIDI. This mixer can also produce MIDI messages, corresponding to live user state changes. If these messages are captured by a sequencer, the sequencer can recreate the user settings during a live performance. "Live performance" describes either a real live music performance, or the production of a final mix onto a user format, such as compact disc.

MIDI can also be used to control "mechanical" devices such as hard disk recorders. A set of SysEx messages called MIDI Machine Control provides the mechanism. Commands defined include "Play", "Stop", "Pause", "Fast Forward", and "Rewind". By using these commands, a sequencer can provide a co-ordinated remote control of compliant devices. MIDI has found use beyond music recording studios. [Huntington 1994] gives a good account of the use of MIDI in more general entertainment control systems.

2.2.1.3 Problems with MIDI

It can be seen that MIDI is very versatile and has been adopted with enthusiasm by the music industry. MIDI is not without its critics however. The main criticisms of MIDI are:

- The slow speed.
- The limited number of channels.
- The limited parameter space.

Often control data and performance data will be transmitted over the same MIDI port, although multi-port MIDI interfaces do exist. Controller messages, especially those that alter the timbre of sounds in real-time can quickly cause congestion and resulting delays. Each 3 byte MIDI message (such as the channel messages) take 1ms to be transmitted. Moore has commented on the significance of even this short delay [Moore 1988]. When additional delays are introduced by other messages, notes that should be concurrent can be spread apart. A further weakness of MIDI is its channel capacity of only 16 channels. All the MIDI controllable resources in a studio need to share these 16 channels, unless multi-port MIDI interfaces are used. A further exploration of the weaknesses of MIDI is provided by [Lehrman, Tully 1993].

2.3 Synchronisation in music recording studios

Besides the need for digital sample rate synchronisation that was previously discussed, there are two other required synchronisation mechanisms in recording studios. The first is to have a common distributed clock that enables all devices to be time aligned. This is necessary so that multitrack recordings can be time aligned (if more tracks are added later), and also to enable non-linear positioning of recorded audio. The second synchronisation requirement, which is needed mostly in studios of the post-production kind, is to be able to synchronise to other media, such as video.

In studio environments, these two synchronisation mechanisms can be implemented using MIDI, by the use of MIDI real time messages, and MIDI time code (MTC).

2.3.1 MIDI real-time messages

The real-time MIDI message group implements what is known as the MIDI sync system. In this system, a master sequencer will send the Start message, followed by Timing Clock messages at the rate of 24 per quarter note. These messages can be interleaved with other MIDI messages. Other devices can use this clock source to produce additional MIDI performance data at a synchronised rate. Some digital multitrack recording devices can capture all MIDI sync system messages and the times at which they occurred. These messages can then be retransmitted at a later stage to the sequencer that produced them to build up a multitrack recording. Other MIDI realtime messages are the Song Position Pointer and Song Select messages. These messages enable some non-linear time alignment.

The MIDI sync system does not provide an absolute time reference, but rather a “beat” reference throughout a MIDI composition. An absolute time reference is provided by MIDI Time Code (MTC).

2.3.2 MIDI Time Code (MTC)

MIDI Time Code is a digital translation of SMPTE (Society of Motion Picture and Television Engineers) time code into reserved MIDI System Exclusive Messages. SMPTE is widely used in video editing applications to provide an absolute time reference with the resolution of a single video or film frame. Up to 24 hours worth of frames can be individually referenced.

The differences in video and film formats (in particular the frame rates) has been accounted for in SMPTE.

A full account of SMPTE and the MTC representation of it are given by [Huntington 1994]. The importance of SMPTE/MTC in a studio environment is that is provides an absolute time reference for all devices that wish to reference it. Multi-track recorders will usually use one track to record SMPTE time code. The SMPTE time code itself is a modulated analog signal representing a bi-phase encoded 80 bit time code. Devices can be bought that translate SMPTE into MTC. A connected sequencer can lock to the MTC stream and build up the multitrack recording. It is often important to be able to record each track sequentially in a studio environment since each track will typically be processed differently using different configurations of the same equipment.

2.4 Digital convergence in recording studios

The sheer processing and data throughput capabilities of contemporary PC’s has enabled an increasing trend towards a PC workstation being used for audio production. Besides MIDI sequencing, PC’s can now also assume the task of digital audio recording, audio processing and even software synthesis. The versatility of MIDI enables the digital workstation to synchronise and control connected studio devices.

Steinberg’s Cubase VST (virtual studio technology) software is a popular application that integrates audio and MIDI production on the PC [Steinberg 1998]. Multiple tracks of MIDI and audio data can be recorded, manipulated, mixed and reproduced. The usefulness of a single workstation hinges on its ability to obtain and reproduce multiple channels of audio simultaneously. By doing this, the workstation has the combined functionality of a mixer and hard disk recorder. Contemporary sound cards, such as Echo’s Layla provide for 8 independent analog audio input and output channels, as well as a digital input and output. Under the Microsoft Windows operating system, each of these input and output channels of audio are presented as individual “wave-in” and “wave-out” devices respectively. Add-on cards that allow for multiple MIDI inputs and outputs are also available. Each MIDI input and output are presented as “MIDI-in” and “MIDI-out” devices respectively.

Using Cubase, multiple effects may be applied to each individual track of recorded audio data in real-time. These effects include equalisation and other sound processing functions found in stand-alone sound processors. Many manufacturers of professional sound processing devices offer an implementation of their device in software as a plug-in for Cubase. Software synthesisers are often implemented with a virtual “MIDI-out” device. In this way, a MIDI track is assigned to the software synthesiser’s “MIDI-out” device, and digital audio is produced locally.

The problems found in studios are still present in a digital workstation environment: Connected devices are still connected by a “spaghetti” of wiring, both for audio and MIDI data. This is compounded by different standards, in the case of digital audio. Based on the problems found in studios, a new connection standard is desired that satisfies the following criteria:

- Supports completely digital multi-channel audio and MIDI data distribution.
- Allows integration with digital workstations.
- Encapsulates methods for routing audio and MIDI data between devices.
- Guarantees synchronisation between connected devices.
- Supports interoperability between multiple vendors.
- Requires minimal user effort to add or remove devices.
- Is cost effective.
- Offers the highest level of performance.
- Is robust and reliable.
- Utilises an open standard.

In the next chapter, a new digital interconnection technology that satisfies these criteria is described – the IEEE 1394 High Performance Serial Bus.

Chapter 3

3 The IEEE-1394 High Performance Serial Bus

The previous chapter showed the need for a new digital interconnection standard in studio environments. In this chapter the operation of a new digital interconnection technology, the IEEE-1394 High Performance Serial Bus, is described. The next chapter describes how this new technology can be utilised for audio and control data distribution in music studio environments. To put the 1394 bus in perspective, consider first the origins of the 1394 bus and other environments where it is being utilised.

3.1 Origins of the 1394 bus

Hoffman suggests that the 1394-bus originated as a desktop LAN [Hoffman 1995], whereas Mullgrave suggests it was designed to replace parallel SCSI in computers [Mullgrave 1997]. Both statements could be considered true. The 1394 bus has found applications where a cheap, high-performance digital interconnection has been required. The significance of the 1394 bus in consumer multimedia products was anticipated by Jonathan Zar, senior business development manager for entertainment and new media at Apple, who referred to the 1394 bus as “the RCA jack of the future” [Harcourt 1997].

Consider PC peripheral interconnectivity, where there is also a need for a single interconnection standard. The 1394 bus offers a possible solution to a process that can be described as the convergence of digital interconnection standards.

3.1.1 The Digital Interconnection Convergence

PCI, ISA, VESA, IDE, and SCSI are some of the existing interconnection standards found in PCs today. Bigger, better and faster have become synonymous with the release of new products in the "application-pull, technology-push" electronics and computing industries. In this relationship, hardware and software drive each other. As new hardware is developed allowing newer applications, newer application emerge requiring improved hardware. Multimedia applications push the resources of a PC to their limits due to the present hardware design being firmly in place before anyone considered the data throughput multimedia applications require, would be attempted [Dyke, Smolen 1997]. Hardware designers responded with faster internal busses and peripheral connection, replacing ISA and EISA busses with PCI and IDE with faster SCSI, for example. While SCSI can provide sustained throughput of 20 megabytes per second, it has limited connectability [Jansen 1996], big expensive cables, and is often difficult to configure [Harcourt 1997].

Engineers at Apple Computer recognised the need for an interface standard that could accommodate the increasing data demands that multimedia applications make on systems. Work on a completely new interconnection technology resulted in Apple engineers devising "Firewire". This later became an IEEE standard, that was numbered 1394 [Harcourt 1997].

Teener describes three factors why a serial bus, and not a parallel connection, was chosen [Teener 1993]:

- Physical constraints – systems are getting smaller and there is less space for connector technology.
- Cost – semiconductor intensive implementations will decrease in price more rapidly than connector intensive implementations due to the evolutionary path of semiconductors and the mature nature of connector technology. Serial cables and connectors are also less expensive to manufacture.
- Reliability – the physical connection in an interconnect is usually the primary point of failure.

Scaleable capabilities with technology improvements are also possible with the simple point to point connections that a serial bus provides [Moore 1996]. Carter states that the 1394 bus is an "elegant optimisation of a number of factors:

- Bit-rate performance.
- Transmission length.
- Protocol efficiency.
- Topological flexibility.
- "Hot pluggability".
- Power Consumption.
- Power Distribution.
- EMI (Electromagnetic Interference) compliance and susceptibility.
- User-friendliness.
- Implementation complexity and physical real estate.
- Cost " [Carter 1994].

3.1.2 Features of the IEEE 1394 bus

At a glance, the 1394 bus offers the following features:

- Up to 400Mbps data transfer rate
- Hot plugging and true plug and play, including auto-configuration and optimisation.
- Power distribution in the cable – if a node's power requirements are modest, they may draw their power to operate directly from the 1394 bus.
- A verified packet transmission mechanism termed asynchronous transactions.
- A guaranteed latency, reserved bandwidth transmission mechanism, termed isochronous transactions. Isochronous transactions can be transmitted on one of 64 isochronous channels.
- Mechanisms for broadcasting, multi-casting and single-destination transactions.

The 1394 bus blurs the distinction between a device interconnection technology, and a networking technology. Compliant devices will typically have multiple ports, the most common having three. Using these multiple ports, up to 63 devices can be connected in any non-cyclic tree topology such that the number of hops between any two devices does not exceed 16. The flexibility of the 1394 bus is further enhanced by the ability of devices to be "hot plugged".

The cabling required consists of thin flexible cables (less than ¼ inch in diameter) that are terminated in simple durable connectors. Two cable types are defined – one with six conductors and the other four. Two pairs of wires are used for signalling and in the case of the six conductor cables, the remaining two are used for power distribution. Cables may be up to 4.5m in length. The connectors were originally designed for the Nintendo Gameboy and have proven to be reliable.

The current standard defines data transmission rates of 98.304, 196.608, 393.216 Mbps (megabits per second) for the cable environment [Teener 1992]. The "cable environment" is specified to distinguish it from the "backplane environment", the other context in which the 1394 bus is used. The backplane environment is comparable to other backplane bus technologies, such as PCI. The backplane environment is described by [IEEE 1995]. The data transmission rates in the cable environment are known as S100, S200 and S400 respectively. This directly translates to the ability to transport 65, 130 and 260 channels of 16bit 48kHz digital audio simultaneously at the S100, S200, and S400 rates respectively (ignoring protocol overhead and bus inefficiencies). Using fewer channels leaves ample bandwidth for MIDI data.

The 1394 bus as currently defined, supports two modes of data transmission: asynchronous and isochronous. Asynchronous transactions, as they are called, are typically used to allow devices to communicate configuration information between one another. Asynchronous transactions provide a verified packet delivery service. Isochronous transactions are not verified, but provide a guaranteed low transmission latency with reserved bandwidth. Fluckiger described how the real-time transfer of multimedia is dependent on the transport technology being isochronous [Fluckiger 1997]. Transmitting devices broadcast isochronous data on one of 64 possible logical channels. Devices wishing to receive the data "listen" on the particular channel. Mechanisms are provided for channel reservation to avoid contention. This coupled with bandwidth reservation and the distribution of a clock signal, keeping all devices synchronised, makes the 1394 bus highly suitable for the transmission of real-time data such as audio and video streams. The specific mechanisms that implement these features are discussed later in this chapter.

3.1.3 Other applications of the 1394 bus

3.1.3.1 Video production studios

Identifying the capabilities of the 1394 bus in video environments, Sony was first to market a digital camcorder utilising the 1394 bus – in fact the first 1394 consumer product. Sony's Digital Video Cassette (DVC) camcorder, equipped with 1394 support, achieved better quality and cost a third of a pro-Betacam SP component system at the time [Vitaliano 1997]. The 1394 support allows a direct digital link to a 1394 equipped PC. Lossless, non-linear editing may be performed by software on the PC, eliminating the need for an expensive Betacam editing desk. Legault suggests that for professional video studios, the 1394 bus is suitable for connecting a few workstations in close proximity and transferring video between camcorders, VTR's and editing stations [Legault 1998].

Using a 1394 equipped PC enables perfect copies of digital content to be made. The prospect of copyright violation has kept publishers hesitant to use digital media. In chapter 4, proposals to protect data on the 1394 bus are discussed. The industry's experience with digital audio tape (DAT) has shown that ways around copy protection can be found. Most consumer DAT decks employ a scheme called Serial Copy Management System (SCMS) which ensures that digital copies do not get further than one generation from the master. However, devices that remove the SCMS between DAT decks can be bought. The complex issue of copyright protection will need to be resolved before content providers embrace the digital medium.

3.1.3.2 Home networks and entertainment

VESA (Video Engineering Society of America), DAVIC (Digital Audio Visual Interoperability Council), and the Home Network Group have all chosen the 1394 bus as the digital media device interconnection [Moses 1998]. The 4.5m limitation on the length of 1394 cables poses constraints for networking a home. Carter describes various methods to increase the distance of 1394 cable lengths in a home-networking environment. Methods proposed include using thicker low-loss cabling, active repeaters, and modulation onto other physical network media [Carter 1995]. Also in the development phase are optical fibre (both plastic and glass based) transceivers that can extend hops to considerable distances [Iverson 1998]. Methods of extending the usable length of the 1394 bus are considered in further detail in chapter 5.

A significant market for the 1394 bus is its use in home entertainment systems. Users will benefit from the superior digital quality and performance [Wetzel, Schell 1996]. Devices comprising an all-digital, 1394-based, home entertainment system could include: HDTVs, DVD players, surround-sound audio decoders, amplifiers, digital satellite receivers, cable set top boxes, and some sort of controlling device.

3.1.3.3 PC networking and peripheral connectivity.

Microsoft have announced that support for networking using the 1394 bus will be provided as standard in their future operating systems [Microsoft 1998]. Unibrain have already released "Firenet" – a 1394 bus network driver that runs under the Windows NT operating system [Unibrain 1998]. Networking using the 1394 bus has a number of benefits:

- High data transmission rate capabilities
- Network resource reservation
- Integration of distributed multimedia applications such as video conferencing
- Sharing of 1394 bus capable peripherals

Inside the PC there are also applications for 1394. The beginning of this chapter discussed the need for a unified digital connection. Prototype hard drives, CD-ROMS, printers, scanners, and DVD players that use the 1394 bus have all been demonstrated [1394 Trade Association 1998].

3.2 The Architecture of the 1394 bus

This section considers the underlying architecture of the 1394 bus that defines the 1394 bus. The next section (2.3) shows the protocol layers that enable the operation of the 1394 bus. The 1394 bus architecture is based on the ISO/IEC 13213 specification:

3.2.1 The ISO/IEC 13213 specification

The ISO/IEC 13213 (ANSI/IEEE 1212) Control and Status Registers (CSR) Architecture for Microcomputer Busses has the following primary goals:

- Reduce the amount of customised software required to support a given bus standard.
- Simplify and improve interoperability of bus nodes implemented on different platforms.
- Support bridging between different bus types.
- Improve software transparency between different bus types.

The IEEE-1394, IEEE-896 (Futurebus), and IEEE-1596 (Scalable Coherent Interface) working groups have adopted the CSR specification[2] which they helped to define. The CSR specification defines the following features:

- Node architectures.
- Address space.
- Common transaction types.
- Control and Status Registers (CSR’s).
- Configuration ROM formats and content.
- Message broadcast mechanisms.
- Interrupt broadcast mechanisms.

3.2.1.1 Node architecture

The 1394 bus architecture is defined in terms of nodes, units and modules. A *node* is an addressable, independently configurable entity. The address space provided by a node can be mapped to one or more *units* within the node. A *module*, which is a physical packaging concept, may contain one or more *nodes*.

[2]The ISO/IEC 13213 (ANSI/IEEE 1212) Control and Status Registers (CSR) Architecture for Microcomputer Busses specification will be referred to as the CSR specification in this thesis.

3.2.1.2 Node addressing

The 1394 bus follows the CSR specification for a 64-bit fixed addressing model, where the most significant 16 bits of the address represent a *node_ID*. The 1394 bus divides the *node_ID* further into two smaller fields – a *bus_ID*(10 bits) and a *physical_ID*(6 bits). This scheme provides for 1023 busses each with 63 independently addressable nodes.

The remaining 48 bits of the address specify a 256 terabyte address space available to each node. This address space is divided into blocks defined for specific purposes:

- Initial memory space (256TB-512MB)
- Private space (256MB)
- Register space (256MB). The register space is further divided into:
    - CSR architecture space (512B)
    - 1394 bus space(512B)
    - ROM (1K)
    - Initial units space (256MB – 2kB)

The addressing scheme is illustrated is figure 2 below.

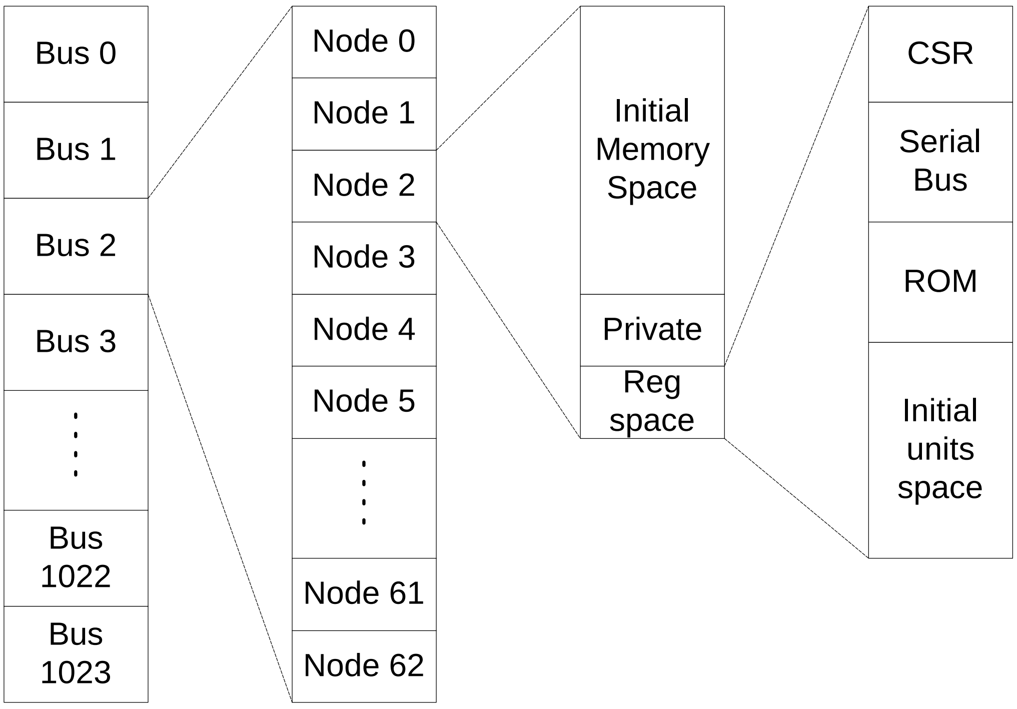


Figure 2: 1394 Node Addressing

3.2.1.3 Bus Transactions

Recall that there are two types of transaction used to transfer data on the 1394 bus – asynchronous and isochronous. Applications requiring a constant delivery rate of data without confirmation of delivery use isochronous transactions for data transfers. Applications requiring confirmation of data delivery of periodic data transfers use asynchronous transactions.

Three types of asynchronous transactions are defined:

- Reads
- Writes
- Locks

Transactions are initiated from a requester and are received by a responder. A read transaction retrieves the contents of a specified memory address from the responder node. A write transaction writes to a specified memory address on the responder node. The lock transaction is a mechanism that permits the requester to perform an atomic read-modify-write operation.

Each transaction consists of two subactions:

- Request subaction. This transfers the address and command for read transactions and also data for write and lock transactions from the requester to the responder.
- Response subaction. This returns the completion status for write transactions and also data during read and lock transactions from the responder to the requester.

The transaction layer within the 1394 protocol stack performs asynchronous transactions. This protocol layer is discussed in section 2.3.

Isochronous transactions consist of a write request subaction, which broadcasts data on a specified channel. Responder nodes are required to listen for data on a channel. 64 channels are available. Isochronous transactions are performed by the link layer within the 1394 protocol stack. The link layer is discussed in section 2.3

3.2.1.4 Control and State Registers

The CSR's provide a standard definition for easier implementation and interoperability. The CSR specification defines the following registers that may be implemented by a 1394-bus node:

Memory offset (hex)	Register Name	Description
000	STATE_CLEAR	State and control information
004	STATE_SET	Used to set the STATE_CLEAR register
008	NODE_IDS	Specifies the 16-bit node ID value
00C	RESET_START	Reset state of node
	:	
018-01C	SPLIT_TIMEOUT_HI SPLIT_TIMEOUT_LO	Implemented by transaction capable nodes to timeout split transaction retries
	:	
200-3FC	Serial Bus Dependent	Registers unique to the 1394 bus, see next table

Table 1 :The CSR register space as implemented by the 1394 bus

The CSR specification also permits bus-specific extensions. The specific CSR features as implemented by the 1394 bus are [Anderson 1998]:

Memory offset (hex)	Register name	Description
200	CYCLE_TIME	Used by isochronous capable nodes as a common time reference
204	BUS_TIME	Used by cycle-master capable nodes. Extends the CYCLE_TIME register.
208	POWER_FAIL_IMMINENT	Used to notify nodes that power is about to fail
20C	POWER_SOURCE	Used to validate power failure notifications.
210	BUSY_TIMEOUT	Timeout on transaction retries
214-218	Not used	Reserved for future use
21C	BUS_MANAGER_ID	The physical ID of the bus manager
220	BANDWIDTH_AVAILABLE	Used to manage isochronous bandwidth
224-228	CHANNELS_AVAILABLE	A 64-bit mask isochronous channel usage
22C	MAINT_CONTROL	Used for diagnostics
230	MAIN_UTILITY	Used for debugging
234-3FC	Not used	Reserved for future use

Table 2: Serial bus dependent registers in the CSR register space.

Of particular importance to audio and control data distribution, are the BANDWIDTH_AVAILABLE and CHANNELS_AVAILABLE registers. The use of these registers is discussed within the context of isochronous resource management in section 2.3.1.

The registers described are the standard registers for 1394 bus operation. Specific protocols may define additional registers, as is the case with IEC 61883-1 that is discussed in chapter 4. [Anderson 1998] gives a full account of possible values for the standard CSR registers.

3.2.1.5 Configuration ROM

Sets of ROM entries are defined to provide configuration information during initialisation. Information provided includes:

- Identifying software drivers for the device
- Identifying diagnostic software
- Specifying node capabilities
- Optionally specifying module, node, and unit characteristics and parameters.

Two ROM formats are defined:

- Minimal – contains only a vendor identifier – the VENDOR_ID field.
- General – vendor identifier, bus information block, root directory containing information entries and/or pointers to other directories.

The configuration ROM enables a process called device enumeration to take place. Device enumeration is typically initiated by a 1394 equipped PC. The process identifies attached devices and allows the appropriate drivers to be loaded. [Anderson 1998] describes the layout of the configuration ROM in more detail, as well as possible values for entries within the configuration ROM space.

3.2.1.6 Message broadcasting

Asynchronous broadcasts on the 1394 bus are accomplished by addressing node 63, which is reserved as a broadcast address. Broadcast transactions prohibit the return of any response by responding nodes to avoid bus contention. Isochronous transactions are inherently "broadcast" in nature.

3.2.1.7 Interrupt broadcasting

The 1394 protocol supports a mechanism, called a nodecast, for broadcasting interrupts to units within a given node. Units wishing to make use of the optional interrupt broadcasting mechanism implement the INTERRUPT_TARGET and INTERRUPT_MASK CSRs. [Anderson 1998] describes the mechanism in more detail.

3.3 The 1394 protocol layers

The operation of the 1394 bus is defined by the protocol layers that are implemented. The protocol layers defined by IEEE 1394-1995 utilise the 1394 architecture as discussed previously. The protocol layers are shown in figure 3 below. In the next chapter, the additional protocol layers necessary for audio and control data distribution are shown and described.

3.3.1 The Bus Management layer

Three global management roles and the services they provide as supports for a completely managed bus are:

- The cycle master providing support for:
  - Controlling the interval at which isochronous transactions are performed.
- The isochronous resource manager providing support for:
  - Allocation of isochronous bandwidth.
  - Allocation of isochronous channels.
  - Selection of the cycle master.
- The bus manager providing support for:
  - Bus power management.
  - Maintenance of the speed map.
  - Maintenance of the topological map.
  - Bus optimisation based on the speed and topological maps.

Nodes on the 1394 bus may be implemented with optional capabilities. The capabilities of the node define the CSR registers that it implements and the bus management functions that it supports. All nodes are required to support automatic bus configuration, which is a task of the physical layer.

Johansson describes six different capabilities, that nodes may implement, which build upon one another [Johansson 1996]:

- A physical repeater.

- A transaction capable node.
- An isochronous capable node.
- A cycle master capable node.
- An isochronous resource manager capable node.
- Bus manager capable node.

The simplest form of a node contains only an active physical layer. These nodes may draw their power to operate directly from the bus and are required to support automatic bus configuration. A node containing only an active physical layer functions as a rudimentary repeater. The bus configuration process is a function of the physical layer and is discussed in detail in section 2.3

Transaction capable nodes are required to be able to respond to quadlet (32-bit word) read and write transaction requests and to implement certain CSR's: NODE_IDS; RESET_START; STATE_CLEAR and STATE_SET. A configuration ROM must be present containing at a minimum the VENDOR_ID field (the minimal configuration ROM). To enable non-vendor-specific interoperability, nodes are recommended to implement the general configuration ROM as defined by the 1394 standard and discussed earlier. Optional capabilities for transaction capable nodes are the ability to respond to block read and write requests, and to generate read and write requests. The SPLIT_TIMEOUT CSR register is required for devices to generate requests.

Isochronous capable nodes are required to implement all transaction capabilities and the CYCLE_TIME register. This register is updated by the node's 24.576MHz free running clock that is required to be implemented in its link layer. Key to the synchronisation and operation of devices on the 1394 bus is the ability to share a common time reference. Jitter in each isochronous capable nodes' CYCLE_TIME register is eliminated every 125μs (nominally) by the presence of a cycle start packet on the 1394 bus which updates all nodes' CYCLE_TIME registers with a common value.

For isochronous operations to be performed on the 1394 bus, at least one node needs to generate the cycle start packets. This node is referred to as the cycle master. In addition to being isochronous capable, cycle master capable nodes must be able to generate cycle start packets at a rate of 8kHz. The cycle start packets are a broadcast write transaction into the CYCLE_TIME register of every isochronous capable node. A cycle master also needs to implement the BUS_TIME register.

Since cycle start packets are required for isochronous operations to be performed on the 1394 bus, isochronous capable nodes are recommended to also be cycle master capable to guarantee the presence of a cycle master capable node.

Nodes that are capable of being the isochronous resource manager need to be cycle master capable and also implement the BUS_MANAGER_ID, BANDWIDTH_AVAILABLE, and CHANNELS_AVAILABLE CSR registers. The BANDWIDTH_AVAILABLE register is initially set to 80% of the actual available bandwidth. This ensures that 20% of the available bandwidth may be used by asynchronous transactions. Nodes wishing to perform isochronous operations must access these registers to acquire a channel number and bus bandwidth before performing any isochronous transactions. Access to these registers is by asynchronous lock transactions. The possible values of the registers required for isochronous resource management are given by [Anderson 1998]

A bus manager capable node needs to be isochronous resource manager capable and have the ability to analyse the 1394 bus configuration and optimise it. The bus manager is required to collect and analyse self-ID packets in order to make the TOPOLOGY_MAP and SPEED_MAP registers available. The gap count parameter on the 1394 bus controls the arbitration interval of nodes. The bus manager can set the gap count to a smaller value than the default, which can result in significant performance improvements, by using the maximum hop count of the current bus topology. Self-ID packets are broadcast by nodes after a bus reset as part of the configuration process. The configuration process is a function of the physical layer and is discussed in section 2.3.

The selection of the cycle master, isochronous resource manager and bus manager takes place during the cable configuration process. The capabilities of nodes on the 1394 bus define how the bus is managed. Several possibilities exist:

- Bus is fully managed – at least one node is bus master capable.
- Bus is partially managed – at least one node is isochronous resource manager capable.

- Bus is unmanaged – there is no isochronous resource manager or bus manager capable node on the 1394 bus.

An unmanaged bus is suitable for specific bus implementations where bus usage is known. This was the case in our custom implementation, described in chapter 5. However, to allow a more generic implementation of a 1394 bus, the 1394 bus should at least be partially managed. In chapter 4, it will be shown how the routing of audio and MIDI streams requires the isochronous resource manager.

3.3.2 The Physical Layer

The 1394-bus physical layer is defined for backplane and cable environments. In the cable environment, which is considered in this thesis, a network of nodes is connected by point-to-point connections. These connections are between ports on the physical layers of the nodes. The physical layers may have multiple ports that allow branching, multi-hop interconnections. The physical layer in the cable environment translates the point-to-point topology into the virtual broadcast bus that higher protocol layers expect. The primary functions of the physical layer in the cable environment are:

- Cable configuration
- Bus arbitration
- Transmission and reception of data bits
- Provision for the electrical and mechanical interface

3.3.2.1 Cable Configuration

Cable configuration and arbitration take advantage of the point-to-point nature of the bus by having each node handshake with its immediate neighbour to determine ownership of the media. There are three phases to cable configuration: bus initialisation, tree identification, and self-identification.

Each time a node is attached to the 1394 bus, the entire bus is reset and reconfigured. The bus reset signal forces all nodes into a special state that clears all topology information and starts the tree identification phase. Bus resets can also be initiated in software. Before the tree identification phase, nodes only know whether they are a branch (more than one directly connected neighbour), a leaf (only one connected neighbour), or isolated (not connected).

The tree identification process translates the general topology into a tree. After tree identification, one node will have become the root node. All the physical connections are given an associated direction pointing towards the root node. Labelling a connector port as a parent (closer to root) or a child (further from root) sets the direction. The node that has all of its connected ports designated children, becomes the root node. [IEEE 1995] gives the mechanisms for root contention resolution. A node may bias itself towards becoming the root node by delaying its participation in tree identification. A forced root delay can be set via a software initiated physical layer configuration packet. This can be used to ensure that the cycle master becomes the root. The cycle master is required to be the root so that it has the highest natural arbitration priority.

The self-identification phase follows. The self-identification process uses a deterministic selection process to give each node an opportunity to select a physical_ID and identify itself to any management entity attached to the bus. Nodes identify themselves by broadcasting self_ID packets. Self_ID packets contain:

- Physical identifier of the node sending the packet.
- An indicator if its link and transaction layers are active.
- The current value of its gap count field.
- The node's speed capabilities.
- Whether it is a contender for isochronous resource manager or bus manager.
- Power requirements.
- Statuses of its ports – if parents and/or children are connected.

Nodes capable of being isochronous resource manager must monitor all self-ID packets to determine if there are other isochronous resource manager capable nodes. The contender having the highest physical_ID value assumes the role. Once the isochronous resource manager is known, any bus manager capable node will differentiate itself by performing a locked compare and write transaction on the BUS_MANAGER_ID field within the isochronous resource manager. The node that successfully updates this register assumes the role of bus manager.

If a bus manager is present, it will verify that the root node is cycle master capable, and if so enable it. If the bus manager is not present, the isochronous resource manager will perform this task. If the root node is not cycle master capable, other nodes are checked for cycle master capabilities. Nodes that are cycle master capable indicate this capability by setting the *cmc* (cycle master capable) bit in the bus information block section of their configuration ROM. When found, the new node is selected to become the new root by the initiation of a forced root delay and then a bus reset. The cable configuration process after a bus reset takes less than 200μs [IEEE 1998]. The enumeration process will take considerably longer however.

The cable configuration process enables the physical operation of the bus. After the physical configuration process, nodes will typically perform some kind of software configuration process. If a PC is connected to the 1394 bus, bus enumeration will typically be performed. The enumeration process typically involves retrieving the contents of configuration ROM of each attached node.

3.3.2.2 Arbitration

After the self-identification phase is complete, nodes wishing to perform a transaction must arbitrate for use of the bus. The priority is based on which node is closest to the root node. When two nodes are equidistant from the root node, the node connected via, or to, the lowest numbered port on the root node is granted access (the ports are arbitrarily numbered). [IEEE 1995] gives the specific signalling involved.

Assuming only asynchronous transactions are used, there is no need to allocate bandwidth since asynchronous transactions use a rotational priority scheme called the fairness interval. After a node wins arbitration and initiates a transaction, it clears its arbitration enable bit. Arbitration for the next transaction begins after a 10μs subaction gap. When all nodes have finished initiating transactions, the bus becomes idle. When the idle time reaches approximately 20μs, all nodes recognise an arbitration-reset gap and enable their arbitration enable bits.

Acknowledgement packets are returned by nodes for all asynchronous packets that they receive. Acknowledgements use immediate arbitration that does not wait for a normal gap timing to expire. Ownership of the media is assumed by the receiving node and enables the acknowledgement packet to be immediately transmitted. The fairness interval arbitration scheme is illustrated below in figure 4.

Assuming only isochronous transactions are being performed, arbitration for the media begins immediately (in fact after 0.04µs) following a cycle start packet. The node closest to the root node wins arbitration similarly to asynchronous arbitration. After the winning node performs its isochronous transaction, the bus returns to the idle state. This idle state is called an isochronous gap and lasts 0.04µs. Other nodes wishing to perform isochronous transactions begin arbitration after the isochronous gap. Each node is allocated bandwidth as a portion of the 125µs cycle interval. Once each node has completed its isochronous transaction, the remaining time in the cycle goes unused in the event that no asynchronous transactions are pending. If asynchronous transactions are pending, they can begin arbitration after detecting a sub-action gap. The sub action gap is significantly longer than the isochronous gap, which ensures that all isochronous transactions are transmitted first. As previously mentioned, up to 80% of the available bandwidth may be allocated for isochronous transactions. The cycle master is necessarily the root node and as such has priority arbitration after subaction gaps. This enables the cycle master to distribute the cycle start packets 8000 times a second at approximately 125µs intervals. Isochronous arbitration and combined isochronous and asynchronous arbitration are illustrated below in figure 5.

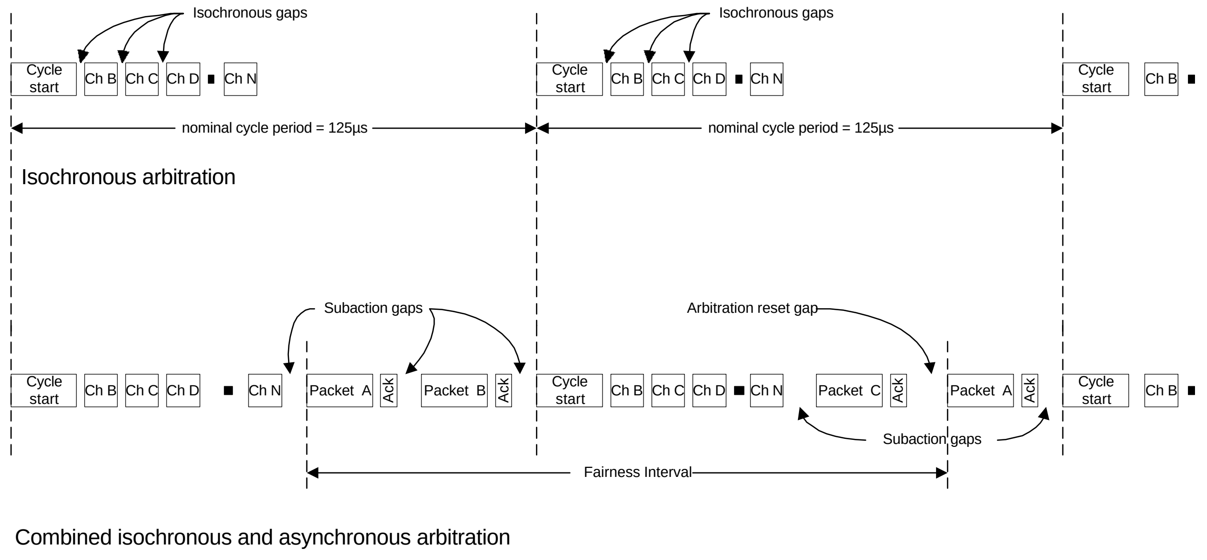
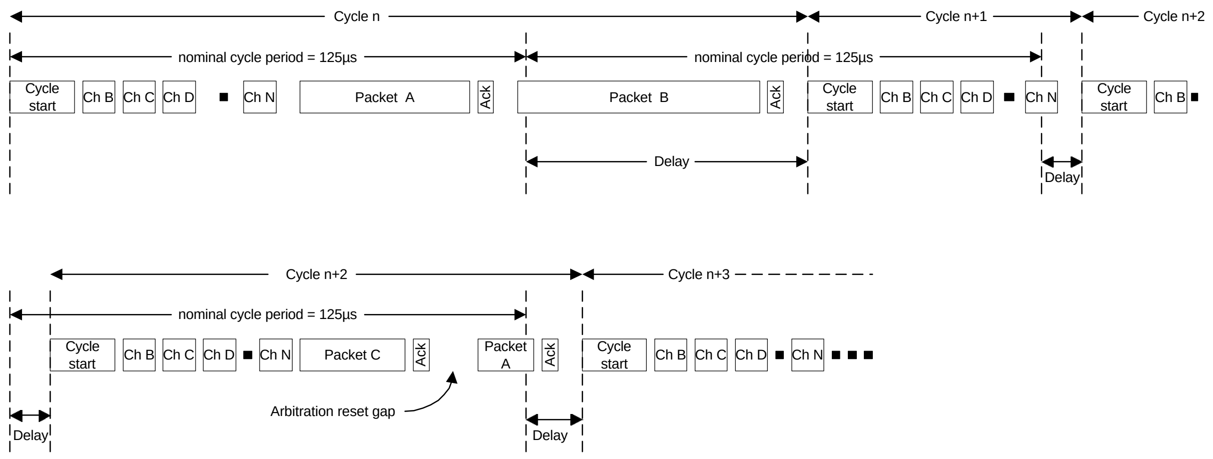


Figure 5: Isochronous arbitration, and combined asynchronous and isochronous arbitration

An asynchronous transaction initiated at the end of an isochronous cycle may delay the cycle start packet. Packet size for asynchronous transactions is limited to half the isochronous bandwidth. Consequently, the cycle start packet may be delayed by a maximum of 65µs. Since up to 100µs (80% of the total time) may be used for allocated isochronous transmissions, a couple of cycles may pass before the cycle start packet is no longer delayed. Pending isochronous transactions are giving priority over the cycle start packet when it is delayed (the cycle start packet waits for a subaction gap). Figure 6 illustrates the delay of the cycle start packet. This process may repeat indefinitely.



3.3.2.3 Transmission and reception of data bits

The cable environment uses two pairs of cables, knows as TPA and TPB, for signalling. A combination of non-return to zero (NRZ) and data-strobe encoding is used to transmit data. The NRZ data is transmitted via TPB and the data-strobe via TPA. The data strobe changes state when two consecutive NRZ data bits are the same. Using NRZ data-strobe encoding allows a clock signal to be constructed by performing a logical exclusive-or on the data and the strobe. The primary rationale for using this scheme is to improve the skew tolerance of transmitted data. In particular, transitions occurring on the strobe and data are approximately one bit period apart. The encoding scheme is shown below in figure 7.

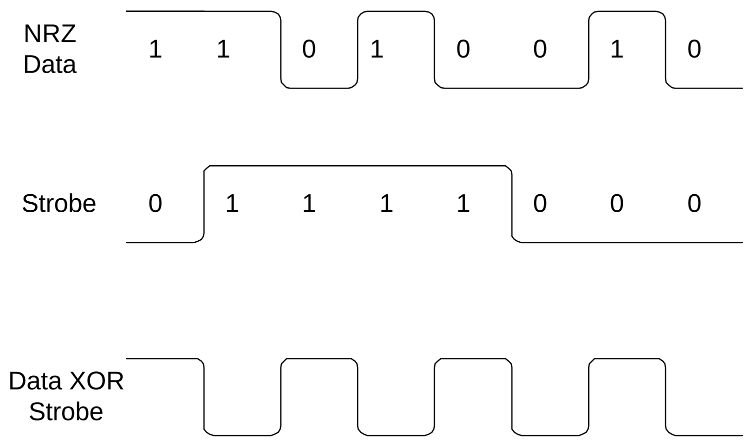


Figure 7 :NRZ data and strobe encoding

Full details of the electrical and mechanical characteristics are given by [IEEE 1995] and [Anderson 1998].

Having considered the bus management and physical layers, the application interface is now considered. As was previously shown in figure 2, an application can access the 1394 bus via the asynchronous or isochronous interfaces. The asynchronous application interface uses the transaction layer:

3.3.3 The Transaction layer

The transaction layer provides an interface between the application and the link layer for asynchronous communications. Four service primitives are provided for asynchronous read, write and lock transactions:

- Request service – initiates the request subaction.
- Indication service – complete the request subaction.
- Response service – initiates the response subaction.
- Confirmation service – complete the response subaction.

The relationship between the application and transaction layers of the requester and responder is shown below in figure 8.

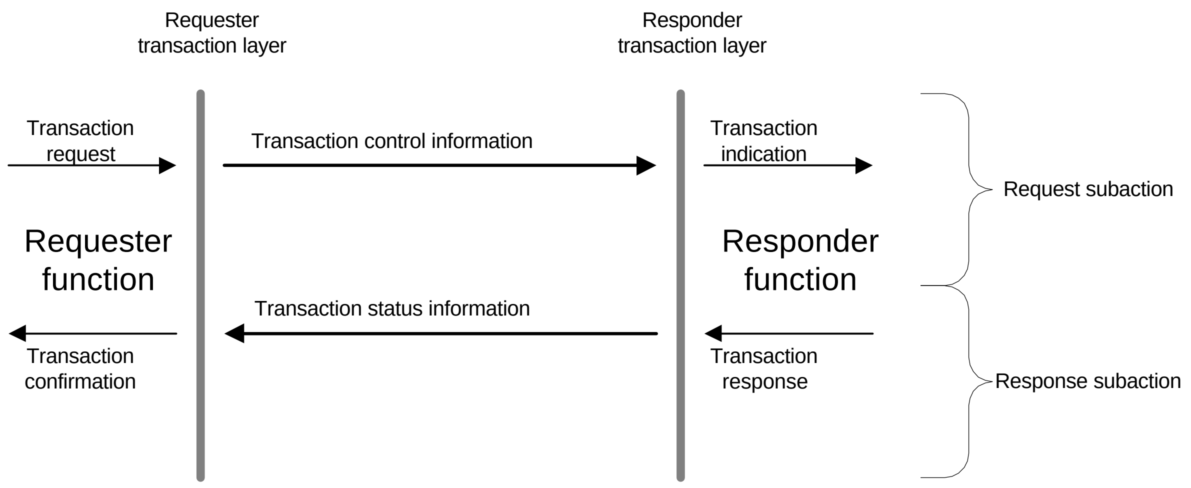


Figure 8: The transaction layer services

Since a responder may not immediately reply to a request transaction, transactions are required to be multithreaded in nature. Transactions must avoid deadlock: sending a transaction request must not block the sending of a transaction response; and avoid starvation: sending a transaction response must not block sending a transaction request. These requirements necessitate that independent queues exist for incoming and outgoing transaction requests and responses.

The 1394 specification adds verification of packet delivery to the transaction protocol of the CSR architecture model. A one-byte acknowledgement packet is returned to the sender to verify successful delivery of each packet. Retries can then be performed in the event of a failure. The link layer performs the confirmed delivery of request and response subactions:

3.3.4 The Link Layer

The link layer provides a half-duplex data delivery service, called a subaction, for asynchronous and isochronous subactions.

The subaction has three possible parts:

- An arbitration sequence where the physical layer is requested to gain control of the bus.
- The transmission of the data packet.
- An acknowledgement that is returned from the packet receiver indicating the action taken. Acknowledgements are not used for asynchronous broadcast and isochronous subactions.

The link layer also defines four service primitives for a subaction:

- Request – the primitive used by a link requester to transmit to a link responder.
- Indication – the reception of a packet by a link responder.
- Response – the transmission of an acknowledgement by a link responder.
- Confirmation – the receipt of the acknowledgement by the link requester

Figure 9 below shows the link layer subactions for performing transaction layer subactions.

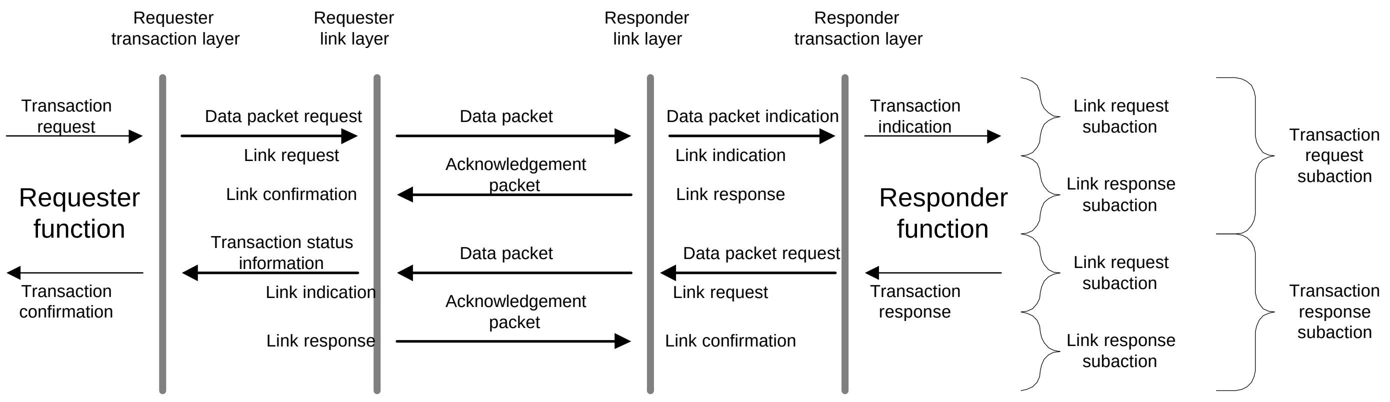


Figure 9: The link layer services

The transaction and link layer interact in a way that optimises the use of the 1394 bus:

- Write transactions can be implemented as unified or split transactions.
- Read and lock transactions are implemented as split transactions.
- Split transactions and isochronous subactions can be concatenated under special circumstances.
- A busy transaction layer can impose limited flow control on its peers.

3.3.4.1 Unified transactions

If the responder's link and transaction layers are fast enough, entire write transactions can be implemented in a single link layer subaction as shown below in figure 10.

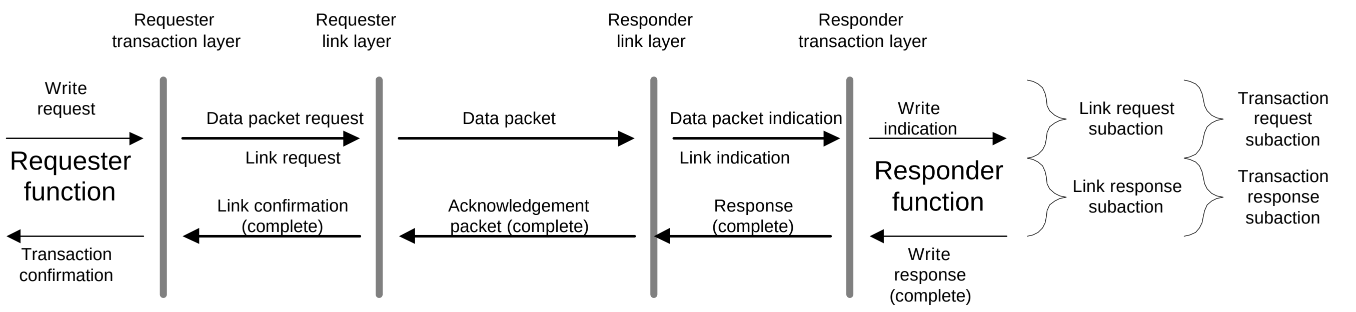


Figure 10: Unified transaction

3.3.4.2 Split transactions

When a responders link and transaction layers are slow, separate link layer subactions are required for the transaction request and response subactions. Other transactions may take place between the request and response subactions. This is illustrated below in figure 11.

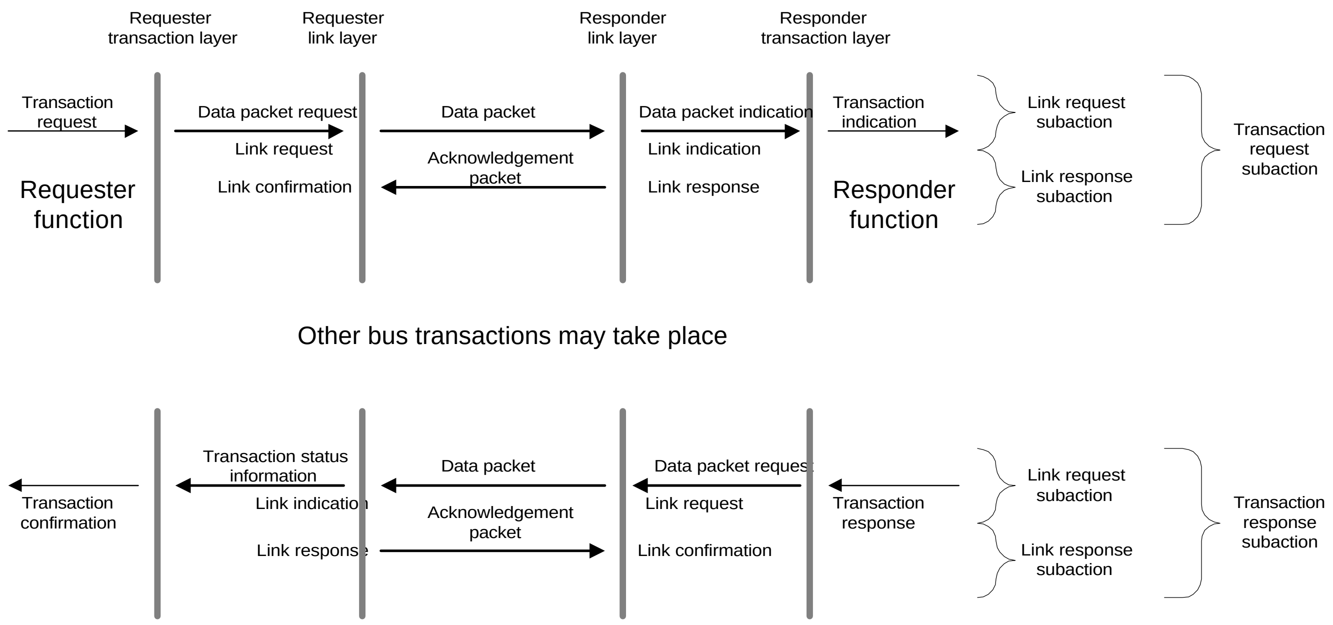


Figure 11: Split transactions

3.3.4.3 Concatenated transactions

If the read or lock response of a responder is fast enough, the request and response subactions may be concatenated by the responder immediately following the request acknowledgement with a response. The responder does not wait for a subaction gap.

3.3.4.4 Transaction retries

Asynchronous retries may be necessary under two circumstances:

- The responding node is busy
- The transfer of the packet failed

Recall that 1394 bus nodes are required to implement separate request and response queues. When a node has a queue full condition, its link layer will respond to asynchronous requests with a busy acknowledgement. The busy acknowledgement notifies the sending node that it is busy and should try back later. The acknowledgement includes the phase of the retry (how many attempts have been made), which is used by the destination node to manage the retry process. Two retry processes are defined:

- Single-phase retry
- Dual-phase retry

The single-phase retry method communicates the phase of the retry between the requester and responder. The transaction is retried until it succeeds, or the retry phase is exceeded. The retry phase limit is specified in the BUSY_TIMEOUT CSR. Single-phase retry provides no scheduling mechanism to handle older transactions that may have been retried many times before handling new ones. The dual-phase retry method provides this mechanism.

The dual-phase retry method divides retries into two groups – A and B. If “A” group retries are being accepted by the responder, other subactions are marked as “B”. Once all the “A” group retries have been accepted, the responder accepts group “B” retries and other subactions are marked as group “A”. This process continues, ensuring the batch processing of the “A” and “B” groups. A timeout is used to determine when to “swap batches”. Source nodes using the dual-phase retry method shall retry the subaction every four fairness intervals, or until the retry phase is exceeded similarly to the single-phase method. Due to tight timing constraints, retries are expected to be performed in the link layer hardware.

The transfer of packets may fail as a result of a number of conditions:

- Request packet transmission errors (detected by the CRC at the responder who notifies the requester).
- Acknowledgement packet timeout (an acknowledgement is not received in time).
- Response packet transmission error (detected by the CRC at the requester).
- Response packet timeout (response not received in time).
- Acknowledgement packet corrupt (questions integrity of preceding subaction)

The applications are notified of these error conditions, and in most circumstances, transactions will be re-initiated.

3.3.4.5 Asynchronous packet formats

The primary packet format of asynchronous request subactions is illustrated in figure 12:

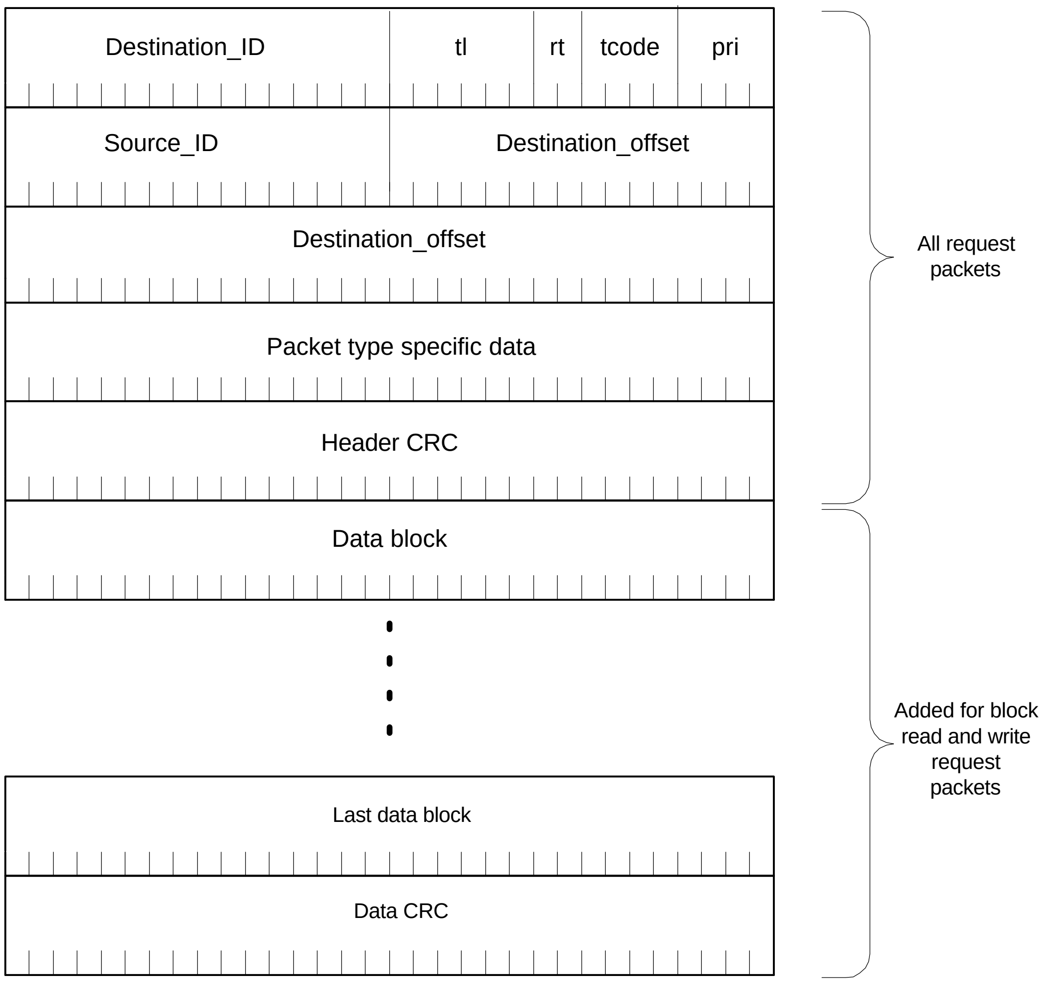


Figure 12: Asynchronous packet format

The destination_ID together with the destination_offset define a memory address in accordance with the CSR 64-bit addressing specification. This is illustrated in figure 13:

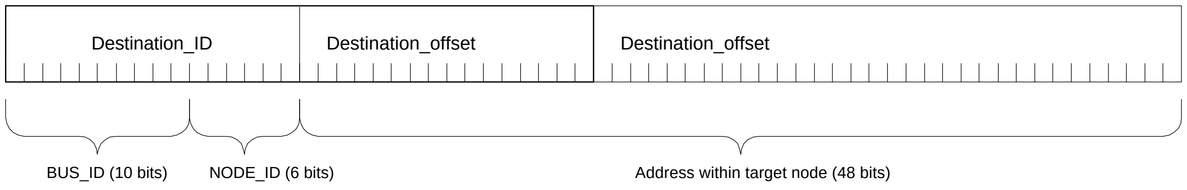


Figure 13: Destination addressing within asynchronous packet format

The transaction label, **tl**, is specified by the requester to identify the transaction.

The priority, **pri**, is not used in the cable environment.

The source identifier, **source_ID**, identifies the node sending the packet.

The retry code, **rt**, specifies whether the packet is an attempted retry and defines the retry phase.

The transaction code, **tcode**, defines the type of transaction. The possible values are shown below in table 3:

Transaction name	**Tcode**
Quadlet write request	0
Block write request	1
Write response	2
Quadlet read request	4
Block read request	5
Quadlet read response	6
Block read response	7
Cycle start	8
Lock request	9
Lock response	B

Table 3 : Transaction codes

The formats of these transactions are fully described by [Anderson 1998]. Of particular importance to audio and control data distribution, and more generally isochronous transactions is the cycle start packet. This packet is broadcast at a rate of 8kHz at nominally 125µs intervals by the cycle master. The cycle start packet format is shown below in figure 14.

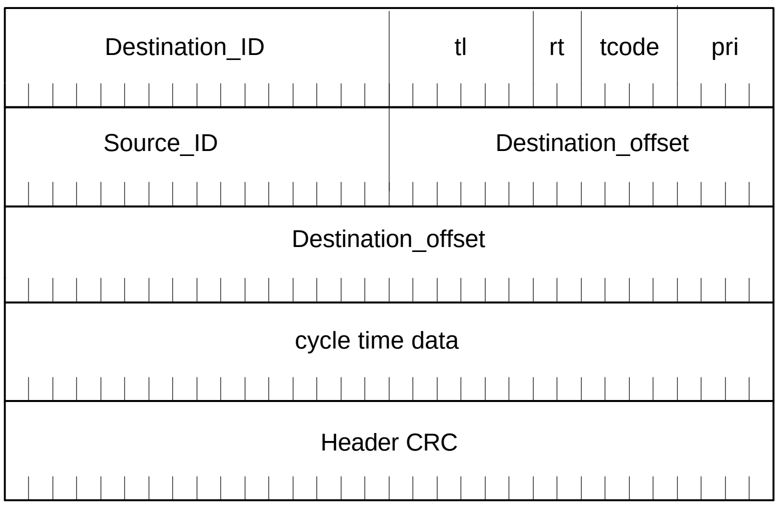


Figure 14: The format of the cycle start packet

The destination_ID is set to node 63 (3Fh). This is the reserved broadcast address. Receiving nodes recognise this broadcast address and do not send responses. The cycle time data is the value of the CYCLE_TIME register of the cycle master.

3.3.4.6 Isochronous subactions

Before isochronous transactions can be performed, an isochronous channel number, and bus bandwidth must be obtained from the isochronous resource manager. The CHANNELS_AVAILABLE and BANDWIDTH_AVAILABLE registers within the isochronous resource manager are accessed via lock transactions.

Once a channel number and bus bandwidth have been allocated, the target nodes must be configured to receive the isochronous data. Target applications accomplish this by assigning their link layers to accept data on particular channels.

3.3.4.7 Isochronous data packet format

Recall that isochronous data transfers are implemented using a single write subaction. The packet format for the write subaction is illustrated in figure 15 below.

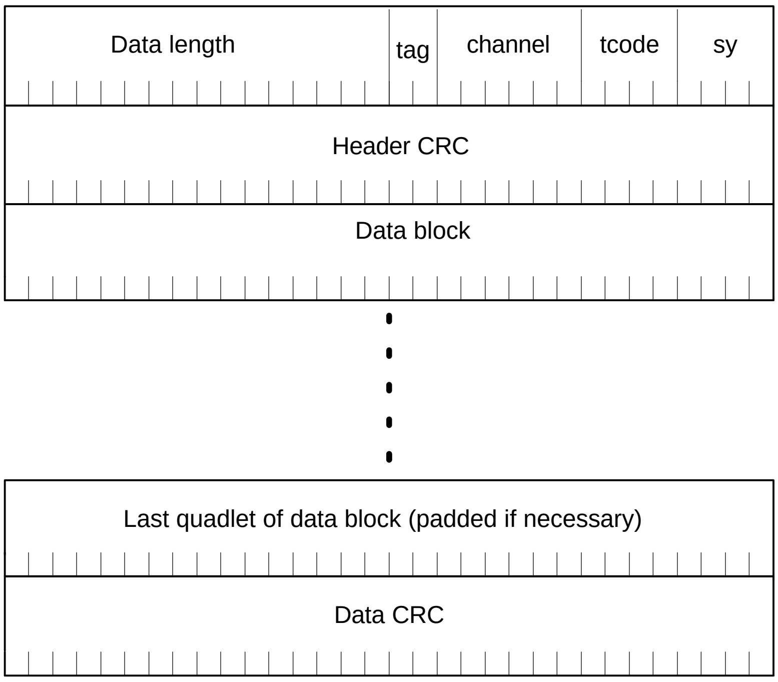


Figure 15: Isochronous packet format

The **data length** specifies the number of bytes of data.

The isochronous data format **tag** is used by higher level protocols, such as the Common Isochronous Protocol (CIP).

The isochronous **channel** number specifies the isochronous channel assigned to the packet.

The transaction code, **tcode**, for isochronous transactions is defined to be Ah.

The synchronisation code, **sy**, is application specific. This field is utilised by the 5C digital transmission copy protection standard.

In the next chapter, it is shown how higher level protocols, particularly what is known as the Audio and Music Data Transmission Protocol (A/M protocol), build upon this isochronous packet format to enable audio and control data distribution. Before this, consider the enhancements to the 1394 specification 1394a and 1394b.

3.4 Enhancements to the IEEE-1394 specification

The 1394a supplement adds new features and enhancements to IEEE 1394-1995 and attempts to clarify some ambiguities that exist. Different interpretations of the 1394 standard have led to interoperability problems. The new features and enhancements are intended to increase the performance of the bus and enhance usability and are backwardly compatible with IEEE 1394-1995. The 1394b supplement, which is being developed, defines extensions for gigabit transmission speeds. 1394b is backwardly compatible with 1394 and 1394a, although for above the S800 rate, alternative connectors such as optical-based need to be used.

3.4.1 1394a

New features provided by 1394a include [IEEE 1998]:

- Connection debouncing.
- Arbitrated bus reset.
- Ack-accelerated arbitration.
- Fly-by arbitration.
- Token style arbitration.
- Priority arbitration.
- Multi-speed packet concatenation.
- PHY pinging.
- Asynchronous streams.

3.4.1.1 Connection debouncing

A bus reset is triggered by a change in connection status at any port – when devices are added or removed. The connection process is not smooth – as the plug and connector scrape together, electrical contact is made and broken many times. Each contact initiates a bus reset (which takes at most 200μs). Since the plug insertion takes considerably longer, the result is a "storm" of bus resets. The disruption caused to bus operations is particularly serious for isochronous data. To overcome this problem, a connection timeout is specified before new connections are confirmed. A timeout of 340ms is suitable to debounce the contact scrape.

3.4.1.2 Arbitrated bus reset

The connection debouncing helps to reduce the time when the bus is unusable during initialisation. If a bus reset is initiated by software, the time taken for initialisation is still longer than necessary. This is due to the delay in propagating the BUS_RESET signal to all nodes. Recall that cable initialisation occurs in three phases: bus reset, tree-identify and self-identify. The self identifying phase requires approximately 1µs per node, tree identification takes about 10µs, but the longest phase is the bus reset which could last 167µs (167µs is the longest time needed before a subaction gap). The delay occurs due to transmitting nodes not being able to detect the BUS_RESET signal. The software initiated bus reset may disrupt two isochronous cycles.

To speed up the bus reset phase, a node gains control of the bus before initiating a software reset. All other devices will be in the "listening state" and can detect the BUS_RESET signal. The bus reset phase is reduced to approximately 1.3µs, bringing the worst case bus initialisation time to 80µs for a fully populated bus. A bus with 20 nodes, can perform initialisation in about 20µs.

3.4.1.3 Ack-accelerated arbitration

Recall that normal asynchronous arbitration requires nodes to detect a subaction gap before arbitrating for the bus except when returning acknowledgements. A node that implements ack-accelerated arbitration concatenates its pending subaction, if any, to the acknowledgement packet it is returning. This eliminates the normal subaction gap that would otherwise be required. Nodes may still only perform a single asynchronous subaction during the fairness interval. Normally, the node closest to the root wins arbitration. Using ack-accelerated arbitration changes the order that transactions are performed.

3.4.1.4 Fly-by arbitration

When a transaction is being performed a multi-port node must repeat the transaction on its other ports. If the packet being transmitted requires no acknowledgement packet, the repeating node can concatenate its packet to the end of the current packet. Fly-by arbitration is restricted to acknowledgements and isochronous packets that are moving towards the root.

Thus fly-by concatenation can only be considered when a packet is received on a child port. Once a node has used fly-by arbitration to send an asynchronous subaction, it must wait for the fairness interval before performing more subactions.

3.4.1.5 Token style arbitration

Nodes may co-ordinate bus arbitration to enhance performance. Recall that when nodes arbitrate for the bus, the node closest to the root wins arbitration. However, the co-operating nodes pass the grant to the node furthest from the root requesting the bus. As the farthest node performs a subaction, successively closer nodes can use fly-by arbitration to concatenate other subactions to the original packet. The bus idle time needed for arbitration is eliminated.

3.4.1.6 Priority arbitration

Priority arbitration allows a node to arbitrate more than once for the bus during a fairness interval if it is sending a response subaction. Since each response subaction was started by a request subaction, fair arbitration for requests inherently limits the number of responses. Also, if only a few nodes are present, individual nodes consume bus time waiting for their chance to arbitrate. A new register called the FAIRNESS_BUDGET controls the number of times that a given node may arbitrate for the bus in a fairness internal. The number of fair arbitration opportunities is divided among nodes. The bus manager must ensure that the sum of all arbitrations in a fairness interval is less than or equal to 63 minus the number of nodes.

3.4.1.7 Multi-speed packet concatenation

The IEEE 1394-1995 standard was published when only S100 physical layer chips were available. Consequently, the details of operating at higher speeds are vague and are susceptible to multiple interpretations. A contradictory requirement of IEEE 1394-1995 is that physical layers signal speed for only the first packet of a multiple packet sequence, but a speed signal is expected for every received packet. Apart from resolving ambiguities, using multi-speed packet concatenations is the building block for the specified enhancements because it permits:

- Concatenation of an arbitrary-speed read response after an acknowledge packet.

- Concatenation of isochronous packets of different speeds.
- Fly-by concatenation without regard to packet speeds.

3.4.1.8 PHY pinging

Recall that the bus manager may improve bus performance by reducing the "gap count". The gap count determines the duration of the subaction gap and the arbitration reset gap. By reducing the gap count, less bus idle time is consumed. IEEE 1394-1995 allows the bus manager to recalculate the gap count based on the greatest hop count between any two nodes. 1394a provides a mechanism for the bus manager to optimise the gap count. The bus manager may send a PHY "ping" packet to any other node. The node responds with a self-id packet. The bus manager may time the round-trip time between itself and the node that was "pinged". The bus manager can use this mechanism to calculate the round-trip time between any two nodes. The gap count is set based on the longest measured round-trip time.

3.4.1.9 Backwards compatibility

When the cycle master is compliant with IEEE 1394-1995, interoperability problems may arise with other nodes using the arbitration enhancements described in the preceding sections. The problem arises with the cycle master no longer having arbitration priority for transmitting cycle start packets. This may disrupt the flow of isochronous data.

To solve this problem, arbitration enhancements are disabled when a cycle start packet is due to be transmitted. All nodes are expected to disable arbitration enhancements from the time of the expected cycle start packet (nominally every 125μs), until the cycle start packet is observed.

3.4.1.10 Asynchronous streams

The 1394a supplement defined a new type of transaction called an asynchronous stream. This transaction is similar to an isochronous transaction except that it takes place during the asynchronous period. Recall that after a cycle start packet, isochronous transactions take place first. This period is known as the isochronous period. After the isochronous period, a subaction gap is observed and asynchronous transactions may take place. Asynchronous streams provide the mechanism for isochronous packets to be transmitted during the asynchronous interval, subject to the same arbitration and fairness as other asynchronous transactions. This isochronous packet is called an asynchronous stream packet. A further allowance is made for more than one node to transmit asynchronous stream packets with the same channel number, although this conflicts with the requirement that channels be allocated from the isochronous resource manager.

The 1394 specification does not permit the reception of an isochronous transaction out of the isochronous period. This is called “strict” isochronous. Many contemporary link layer controllers relax this requirement and can accept isochronous transactions at any time. This is known as “loose” isochronous. The 1394a supplement expressly permits the reception of isochronous packets out of the isochronous period. Asynchronous streams provide all the mechanisms for efficient multicasting of data. This is being utilised to provide IP (Internet Protocol) distribution over the 1394 bus. Further information on IP over the 1394 is given by [IETF 1998].

3.4.2 1394b

A working group within the 1394 Trade Association is developing 1394b, a supplement that extends the 1394 standard and 1394a supplement. The most significant extensions are the increased transmission speeds, and specification of alternate media.

For increased transmission speeds, a new physical layer mode has been added. This new physical layer uses a modification of the encoding used by FibreChannel and gigabit Ethernet. The new arbitration scheme called “BOSS”, dramatically reduces arbitration overhead. Arbitration runs in parallel with data transmission over a single pair. Being able to utilise the existing 1394 cables enables a full-duplex connection between physical layer chips. The result is a transmission speed of 1.6Gbps (giga bits per second), known as the S1600 rate, over the standard 4.5m 1394 cables [Teener 1999].

Specifications of alternate media include from the use of unshielded twisted pairs for the S100 rate, to fibre optic cable for the S1600 rate. The use of fibre optic cable enables significantly longer hops between devices – up to several hundreds of metres. In chapter 6, it is discussed how NEC’s implementation of 1394b transceivers can be used to provide remote studio connectivity. The current status of the 1394b specification is provided by [IEEE 1998].

Chapter 4

4 Using the 1394 bus for audio and control data distribution.

The Audio and Music Data Transmission Protocol, known commonly as the A/M protocol provides a solution for audio and control data distribution using the 1394 bus. In chapter 2, the need for a high-performance digital interconnection, allowing the transmission of multiple channels of audio and control data using a single cable, was discussed. In chapter 3, such a high performance interconnection standard was described – the 1394 bus. In this chapter, it is shown how the A/M protocol utilises the capabilities of the 1394 bus to provide mechanisms for transmitting and routing multiple channels of audio and MIDI data, and maintaining synchronisation between devices.

Informative discussions are presented of how the A/M protocol can be integrated with digital PC workstations, copyright protection mechanisms using the 1394 bus, and industry interoperability initiatives.

In the next chapter, a case study implementation of the A/M protocol is described. The implementation clarifies the roles of software and hardware elements used in implementing a 1394 node with A/M protocol capabilities.

4.1 A user's perspective of an IEEE 1394 based recording studio

Shown below in figure 16 is the same studio environment discussed in chapter 2 but, with IEEE 1394 connectivity. Notice the absence of the audio patcher.

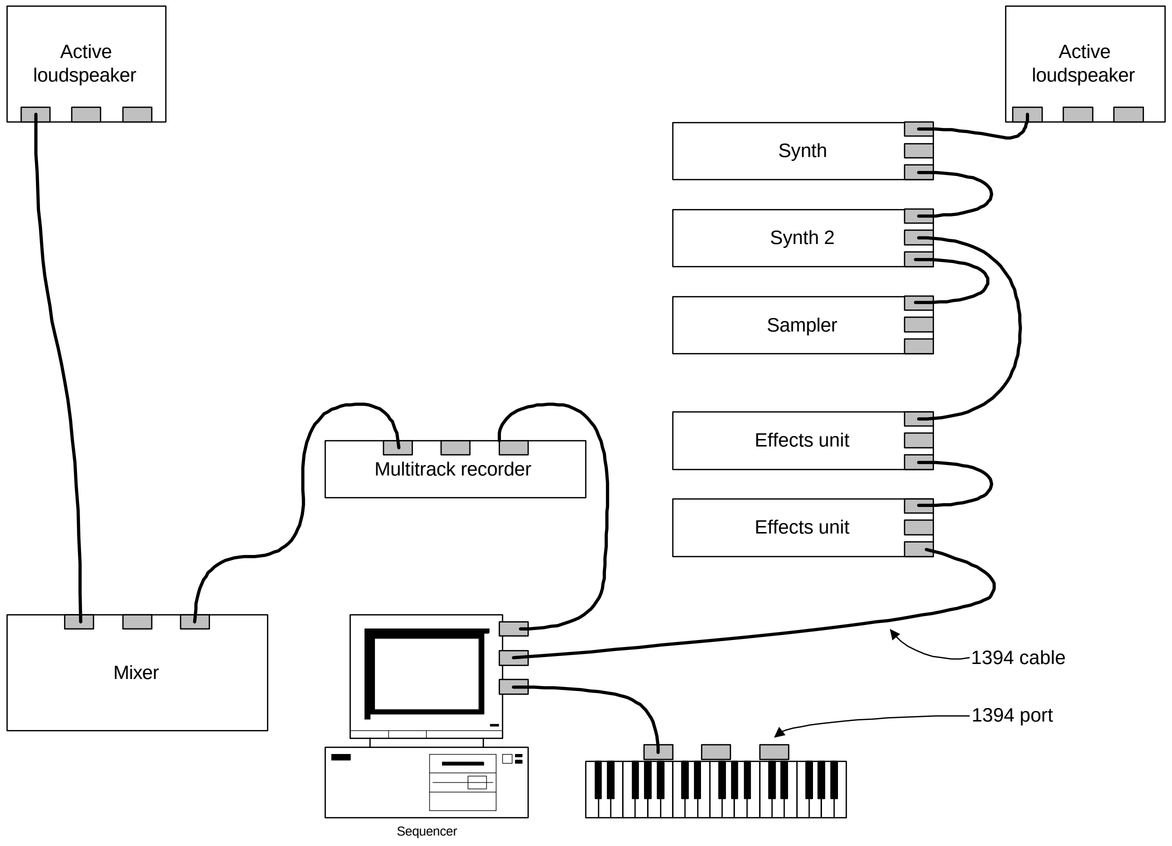


Figure 16: Studio connectivity with IEEE 1394

4.1.1 Flexibility and performance

There is tremendous potential for studio equipment to utilise the 1394 bus. Compared to existing solutions that were discussed, 1394 based connectivity offers:

- A single flexible cable that connects devices in any non-cyclic topology such that the number of hops between any two devices does not exceed sixteen.
- Up to 64 channels each possibly containing multi-channel audio and MIDI data.
- Flexible user controlled routing of audio and MIDI data.
- Mechanisms for sample rate recovery and time alignment of digital audio.
- Provisions for copy protection.

- Seamless integration with digital PC workstations – each device on the 1394 bus can appear as “wave-in”, “wave-out”, “MIDI-in”, and “MIDI-out” devices and utilise existing audio production software, such as Cubase VST.

The features just mentioned may seem too good to be true, and the question posed is “Is this available to consumers?” The answer is “not yet”. The 1394 technology is new, and protocols need to be refined and hardware improved. In this chapter it will be shown how all the mechanisms and architecture are in place for a 1394 connected studio, and how standards bodies are working to realise 1394 technology for studio use. In the next chapter, a case study implementation is presented that demonstrates the feasibility of audio and MIDI distribution over the 1394 bus.

4.1.2 mLAN and the A/M protocol

Yamaha Corporation were first to realise the potential of the 1394 bus for audio and MIDI data distribution, in their mLAN (music Local Area Network) specification [Yamaha 1996]. The mLAN specification extended an international standard for digital video interoperability using the 1394 bus – the IEC 61883 “Digital Interface for Consumer Electronic Audio/Video Equipment” to accommodate audio and MIDI data. The mLAN specification was submitted to the 1394 Trade Association for consideration, and components of mLAN were adopted as the “Audio and Music Data Transmission Protocol”. The audio working group within the 1394 Trade Association and the IEC are working to finalise the A/M protocol as part of the IEC 61883 international standard. An overview of the earlier version of the A/M protocol is given by [Fujimori, Osakabe 1996].

The IEC 61883 standard currently consists of 6 sections:

- IEC 61883-1: General. General packet formats, data flow management, and connection management for audio-visual data.
- IEC 61883-2:SD-DVCR. Standard definition digital video data transmission.
- IEC 61883-3:HD-DVCR. High definition digital video data transmission.
- IEC 61883-4:MPEG2-TS. MPEG layer-2 data transmission.
- IEC 61883-5:SDL-DVCR. High compression mode digital videocassette recorder data transmission.

- IEC 61883-6 (PAS): Audio and Music Data Transmission. Audio and MIDI data transmission.

Part 6 of the IEC 61883 standard is the publicly available specification (PAS) in the process of being finalised. Once finalised, it will be the definitive standard for audio and MIDI data transmission using the 1394 bus, for both professional studio and consumer electronic applications.

There are three main aspects of the A/M protocol:

- Audio and MIDI distribution
- Synchronisation
- Connection Management

Audio and MIDI distribution, and connection management utilise the Common Isochronous Protocol (CIP), and Connection Management Procedures (CMP) respectively, defined in IEC 61883-1. Synchronisation mechanisms also utilise features provided by IEC 61883-1.

4.2 The Common Isochronous Protocol (CIP)

Isochronous transactions have a number of features that make them suitable for distributing audio on the 1394 bus, including low transmission latency and "just in time" delivery. There are however subtle problems, or challenges, with the transmission of audio-visual isochronous data streams:

4.2.1 Challenges with isochronous data transmissions

There are two problems introduced by [Light, Bloks 1996]:

- Mapping AV (audio-visual) data to 1394 isochronous packets.
- Increasing jitter tolerance.

Only one isochronous transaction may be initiated per channel per isochronous cycle on the 1394 bus. This equates to 8000 packets per channel per second. Channels are used to represent connections between units. Thus for any particular connection, a maximum of 8000 packets per second are possible. There are many different AV data types that all group data into blocks of a constant size. The number of blocks per second may be varied or constant depending on whether variable or constant bandwidth, respectively, is required. The MPEG data type for instance, uses variable bandwidth. The problem is how these blocks should be mapped into isochronous packets. If variable bandwidth is used by a data type, the maximum bandwidth must be allocated to ensure that bandwidth will be available in every isochronous cycle. The bandwidth needs to be smoothed out to decrease the maximum and allow more isochronous channels to operate.

The arbitration protocols used on the 1394 bus can introduce considerable delay in the delivery of isochronous transactions. It was shown how the cycle start event (and subsequent isochronous packets), may be delayed by up to 65µs in section 2.3.2.2. An MPEG-2 data stream (becoming widely used for digital video) is particularly intolerant of variable delays. The MPEG-2 system standard allows a maximum delay deviation or jitter of 500ns [Wetzel, Schell 1996]. A mechanism is needed that enables a constant delay for data transport over the 1394 bus.

The IEC-61883-1 standard, which defines the common isochronous protocol (CIP), addresses these problems. The A/M protocol utilises the IEC-61883 mechanisms for isochronous data distribution.

4.2.2 Solutions offered by IEC-61883

The IEC-61883 standard for Digital Interface for Consumer Electronic Audio/Video Equipment[3] is a standard that will ensure interoperability amongst consumer electronic devices that use the 1394 bus. The IEC 61883 standard was originally intended to ensure interoperability between different digital video devices, since digital video transmission was one of the first uses of the 1394 bus. As previously mentioned, the CIP is extended in IEC 61883-6 to describe the format of audio and MIDI data types.

The CIP addresses the problems described previously in the following way:

- Data fractions – maps AV data types into 1394 packets making efficient use of bandwidth.
- Time stamping – enables a constant transport delay and reconstruction of the application packet timing.

4.2.2.1 Data block fractions

Application packets are divided into a number of smaller sized fractions called data blocks. The number of fractions can be 1,2,4 or 8. The maximum number of data blocks transmitted in a single isochronous cycle is determined from the application bit rate. This allows a single application packet to be transferred over many isochronous cycles (for low bandwidth requirements) or many application packets to be transferred in a single isochronous cycle (for high bandwidth requirements). This effectively smoothes out the bandwidth required in a single isochronous cycle. Figure 17 shows a large packet transmitted without using data fractions.

[3] "IEC 61883 standard for Digital Interface for Consumer Electronic Equipment" will be referred to as "IEC 61883" in this thesis.

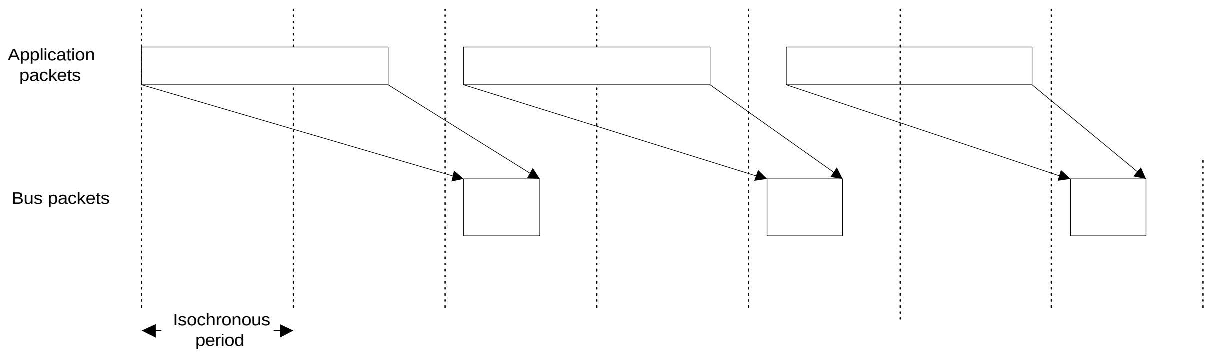


Figure 17: Isochronous transmission of large application packets without using data fractions

Recall that bandwidth is allocated in terms of a fraction of the isochronous period. In figure 17 above, the allocated bandwidth is unused in some isochronous periods. Compare this to figure 18 below, illustrating an application packet divided into four data fractions.

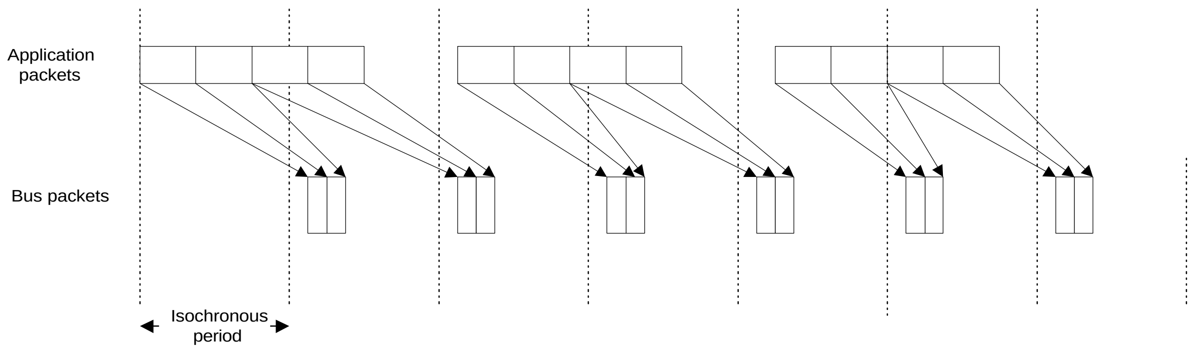


Figure 18: Using data fractions for efficient isochronous bandwidth utilisation

4.2.2.2 Time stamp insertion

In CIP data transmission, there are two distinct mechanisms provided for synchronisation via time stamps:

- Transport delay time stamp.
- Higher level application synchronisation time stamp.

The transport delay time stamp, can be added to an application packet to enable the inter-packet timing information to be reconstructed at the receiver. The time stamp is a 25 bit sample of the local link layers CYCLE_TIME register, incremented by an offset. This time stamp is the intended delivery time to the receiver. The MSB of the time stamp is padded with zeros to 32 bits.

Figure 19 shows how variable rate application packets are transmitted without timestamps – the timing information is lost.

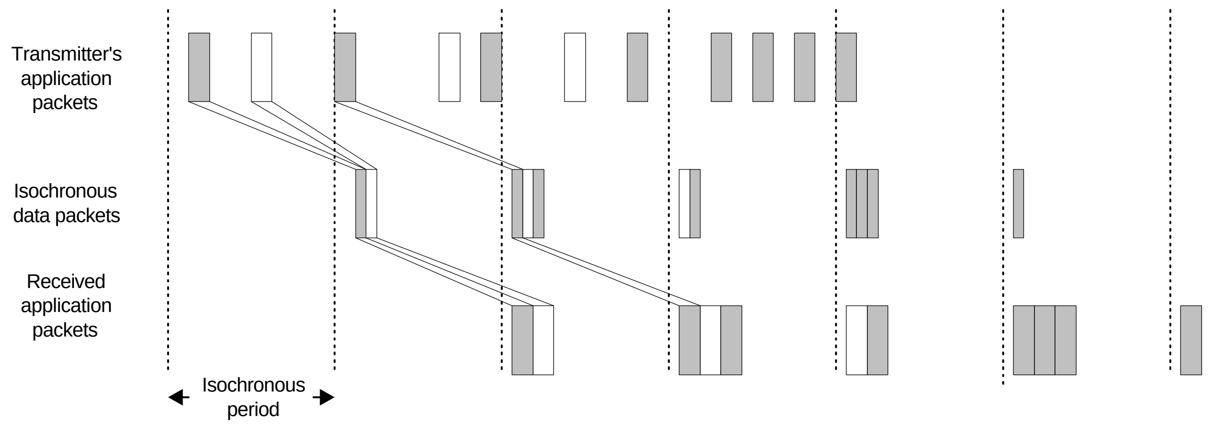

Transmitter's application packets
Isochronous data packets
Received application packets
Isochronous period

Figure 19: Isochronous data transmission without timestamps

Figure 20 below shows how by using timestamps, the receiving node can reconstruct the original packet timing information.

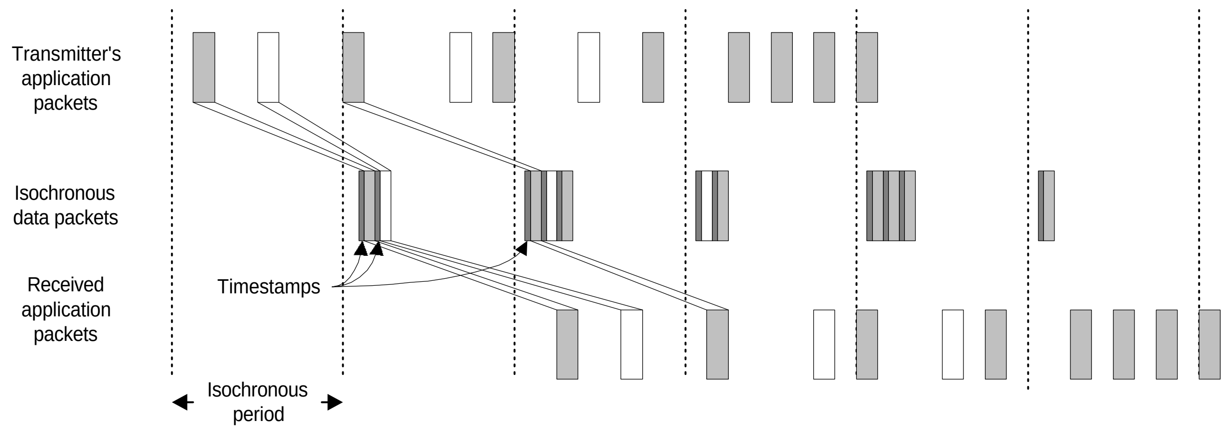

Transmitter's application packets
Isochronous data packets
Received application packets
Timestamps
Isochronous period

Figure 20: Timestamping isochronous data to allow packet timing recovery

Higher level application synchronisation is provided by the SYT field within the CIP header. The A/M protocol does not use the transport delay time stamps, but utilises the SYT field for synchronisation. Before considering how the A/M protocol uses the SYT field within the CIP header, consider the format of the CIP packet.

4.2.3 CIP packet format

The CIP packet is embedded within a standard isochronous transaction packet as illustrated in figure 21 below.

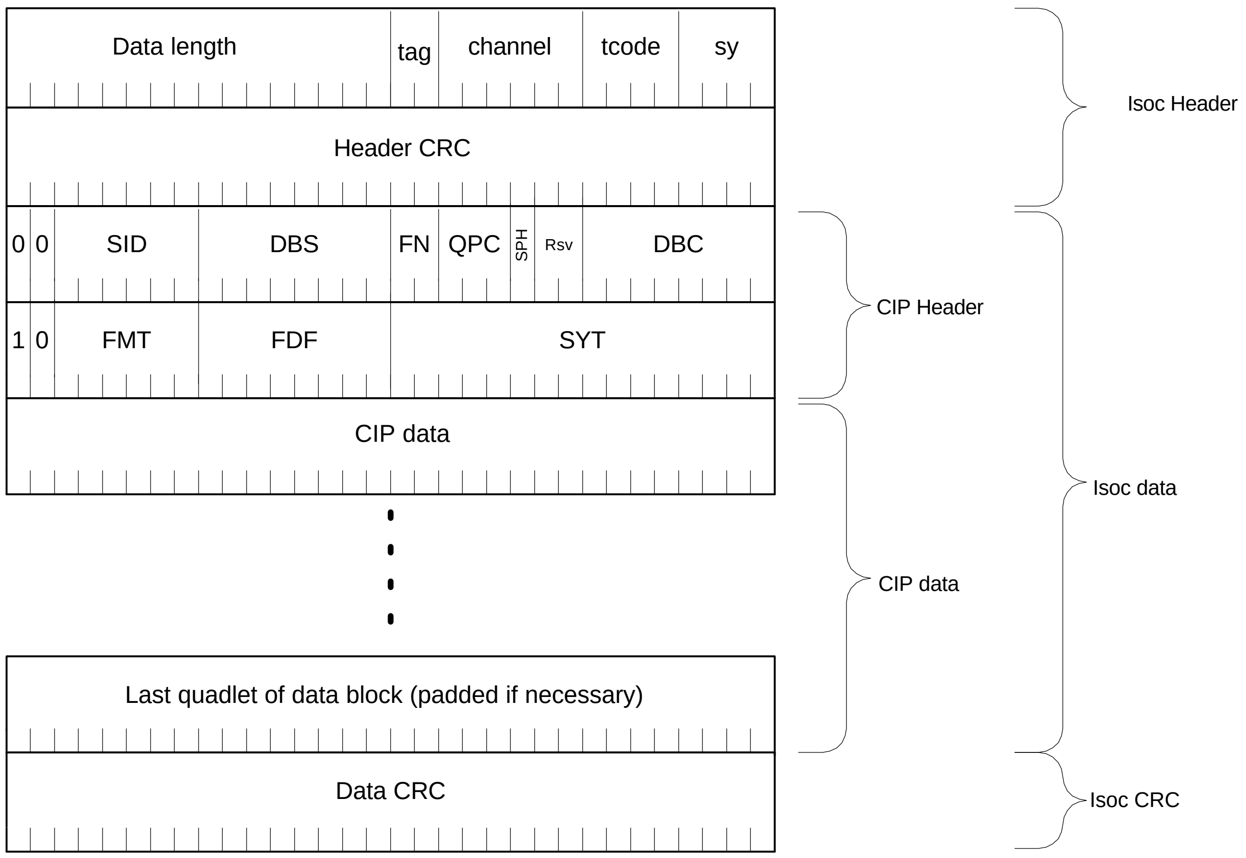


Figure 21: The Common Isochronous Protocol (CIP) packet format

The fields within the isochronous header are as previously discussed. For isochronous transactions that are CIP conformant, the **tag** field is assigned the value of 01b.

The source identifier, **SID**, contains the 6 bit physical ID of the node transmitting the packet. The **DBS** field contains the data block size.

The **FN** field contains a value that indicates the number of fractions into which the original application packet was divided (after optional stamp and padding were added): 00 = 1, 01 = 2, 10 = 4, 11 = 8. This can be expressed as a mathematical relation as well: number of data blocks = 2^{fn}. The **QPC** (quadlet padding count) field contains a value that indicates the number of padding bytes added to each application frame to make it divide evenly into 2^{fn} data blocks.

The **SPH** field indicates if a transport delay time stamp has been added by the transmitter.

The **DBC** field acts as a sequence counter and continuity checker. To allow receivers to easily resynchronize to a stream after missing an entire bus packet, or after tuning in to an already existing stream, there is an additional requirement that holds for all data types: the first data block of any source packet has a sequence number S for which the following condition must hold:

$S \textbf{ mod } 2^{fn} = 0$

Using the **dbc** and **fn** field values from the CIP header the receiver can easily determine the start of the next source packet in the incoming data stream.

The **FMT** field identifies the data type using the CIP.
The **FDF** is the format dependant field used to identify sub-formats or convey other information.

Unique values specified by the A/M protocol are shown below in table 4.

Field	**Value**	**Comment**
FMT	10h	This value indicates that the format is the A/M protocol
FN	0	Source packets are not divided into fractions
QPC	0	No quadlet padding, since all data is 32 bit aligned
SPH	0	No transport delay time stamps are present
SYT	X	The presentation time of the specified event.

Table 4 : CIP header values for the A/M protocol

The use of the CIP for other stream types is described by [Phillips 1997] and [IEC 1996].

4.2.4 CIP Data

The format of the data being transmitted by the A/M protocol is indicated by the value of the **FDF** field within the CIP header. This field also contains a nominal sampling frequency code (SFC). The data transmitted by the A/M protocol is termed to consist of 32-bit aligned events. The following events are defined:

- AM824 Data
- 24-bit * 4 Audio Pack
- 32-bit Floating-Point Data
- Provision for 32-bit or 64-bit data

Events that occur at the same time are grouped into an ordered collection called a cluster event. To clarify the terminology, a stereo audio sample would be a cluster event consisting of two AM824 data events

The A/M protocol specification gives the particulars of these events. The most relevant event type in the context of a studio environment is the AM824 data event. A 32-bit data block consisting of an 8-bit label and a 24-bit data chunk is called AM824 data.

AM824 data encodes IEC 60958 digital audio, raw audio and MIDI data into a 32-bit event. The format of the IEC 60958 AM824 event is illustrated below in figure 22.

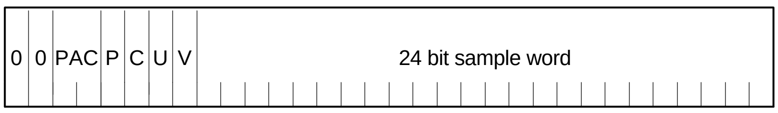


Figure 22: Format of the IEC 60958 AM824 event

The P,C,U,V carry the auxiliary information as per the IEC 60958 standard. The preamble code (PAC) identifies the subframe within the IEC 60958 frame. The values of PAC and the corresponding preamble codes are shown in table 5 below.

Value (binary)	Description
11	B

01	M
00	W

Table 5 : Preamble code definitions

The format of the raw audio AM824 event is illustrated in figure 23 below.

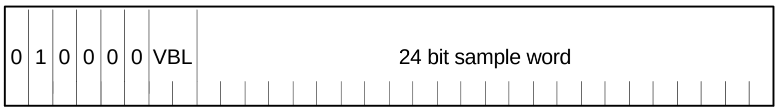


Figure 23 :Format of the raw audio AM824 event

The values of the variable bit length (VBL) field are defined in table 6 below.

Value (binary)	**word size of audio sample**
00	24
01	20
10	16
11	Reserved

Table 6 : Definition of the variable bit length (vbl) field

The format of the MIDI AM824 event is illustrated in figure 24 below.

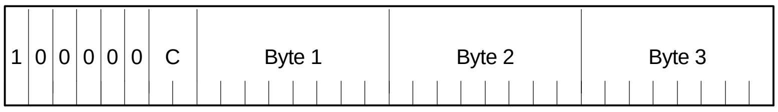


Figure 24 : The format of a MIDI AM824 event

The values of the counter field (C) are defined in table 7 below.

Value (binary)	Number of valid MIDI bytes
00	0
01	1
10	2
11	3

Table 7 : Definition of the counter field within MIDI AM824 events.

Embedding MIDI data as an AM824 data type within a single isochronous stream encapsulates the outputs that a MIDI synthesiser may produce – namely audio and MIDI data. If only MIDI data is transmitted, then using AM824 data within a CIP header is extremely inefficient. The MIDI data rate of 31.25kbps translates to the transmission of a single MIDI byte taking 320µs, and a performance MIDI message about 1ms (including start and stop bits). Isochronous data can be transmitted every 125µs. If MIDI data is transmitted back-to-back, at best every second isochronous transmission will contain valid data. MIDI data is however seldom transmitted back-to-back for any length of time, introducing more isochronous transmissions containing no data. The inefficiencies of using this method are compounded by the protocol overhead – the CIP header combined with the isochronous header is sixteen times larger than the data being transported.

As an alternative, consider the transmission of MIDI data using asynchronous write transactions: Asynchronous transactions are not guaranteed to have a specific latency. Under normal conditions, the asynchronous latency will be 125µs. Depending on the number of nodes and their utilisation of the 1394 bus, this latency may increase considerably. As an extreme measurement, consider a hypothetical fully populated (64 nodes) 1394 bus. If one node was transmitting isochronous transactions consuming 80% of the bandwidth, and the other 63 nodes were all transmitting maximum-size asynchronous transactions, the latency at the S100 rate could be over 12s! This situation is however highly unlikely to occur in any "real" systems. Under normal conditions, even if arbitration was delayed by eight isochronous cycles, the resulting 1ms latency could be considered acceptable, since this is inherent for concurrent MIDI events.

Using asynchronous transactions for MIDI distribution, besides being non-standards compliant, disconnects MIDI streams from routing mechanisms (described later in section 4.4).

4.2.5 Transmission methods

The CIP requires that nodes transmit an isochronous transaction in each isochronous period. Nodes may transmit CIP packets containing data, or empty packets. There are two transmission methods that define the behaviour of how a node transmits non-empty packets. The first method is called non-blocking transmission. In non-blocking transmission, a non-empty CIP packet is transmitted if one or more events arrive in a nominal isochronous cycle. The other method, called blocking transmission waits until a fixed number of events have arrived before transmitting the CIP packet. The differences between these two formats is illustrated below in figure 25. The A/M protocol allows either transmission method to be used.

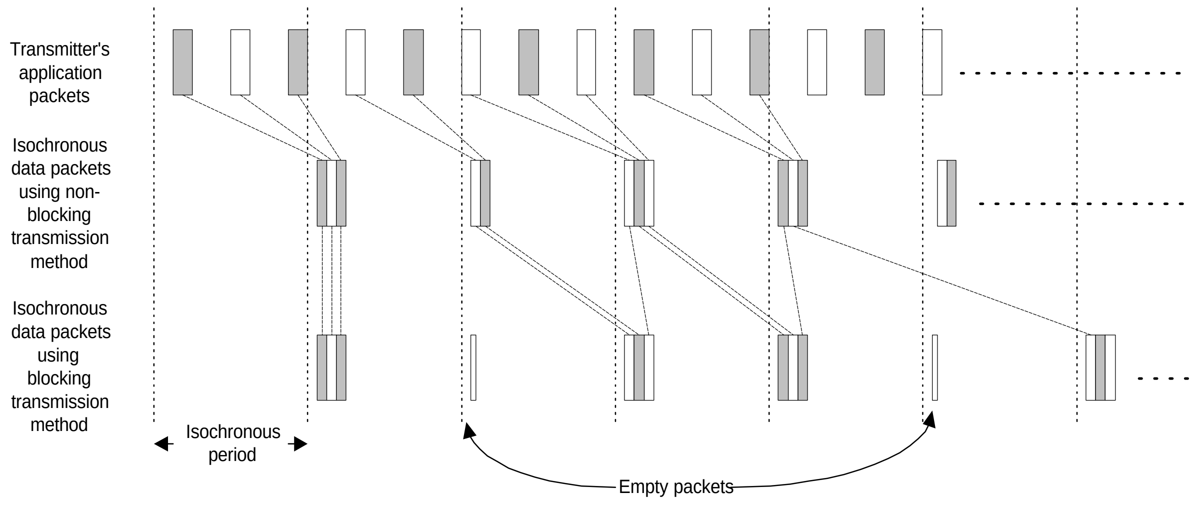

Transmitter's application packets
Isochronous data packets using non-blocking transmission method
Isochronous data packets using blocking transmission method
Isochronous period
Empty packets

Figure 25 : The blocking and non-blocking transmission methods

4.3 Synchronisation

Synchronisation in digital audio transmission systems has two main aspects:

- Sample rate recovery
- Time alignment

Both of these functions can be performed in the A/M protocol by using the higher level application synchronisation mechanisms provided by the CIP – the SYT field. Transmitting nodes place the 12 least significant bits of their CYCLE_TIME registers, incremented with an offset, into the SYT field of the CIP header. The time in the SYT field is the intended presentation time of the event to which it applies. A CIP packet may contain multiple events, and only one time stamp. In order to associate the SYT time stamp with a particular event, a unit known as the SYT_INTERVAL is used. The SYT_INTERVAL is defined to be the number of events between valid successive SYT time stamps. The SYT_INTERVAL is made known to the receiver by a standard relation to the nominal sampling rate conveyed in the SFC (sample frequency code) fields within the FDF field.

The transmitter prepares the time stamp for the cluster event that meets the condition:

$$mod(DBC, SYT\_INTERVAL) = 0$$

As an example, 48kHz digital audio requires a SYT_INTERVAL of 8. If the source is stereo, then each cluster event would contain two AM824 audio events. Every eighth cluster event would be given a timestamp.

At the receiver, the index of the cluster event to which the SYT time stamp applies can be calculated from:

$$index = mod((SYT\_INTERVAL - mod(DBC, SYT\_INTERVAL)), SYT\_INTERVAL)$$

Consider the previously mentioned example again. If the cluster events were transmitted using the non-blocking transmission method, there would nominally be 6 cluster events per CIP formatted isochronous transaction (48kHz divided by 8kHz). Consider the index of the time-stamped cluster for two consecutive CIP formatted isochronous transactions each containing 6 cluster events. Assume in the first transaction, the time stamp applies to the first cluster. This would mean that DBC is a multiple of 8, say 16. From the above formula, index = mod(8-mod(16,8),8) = 0. In the next transaction DBC = 16+6 = 22, and index = mod (8-mod(22,8),8) = 2. This would mean the timestamp applies to the third cluster, as would be expected.

The recovery of the timing for events between events to which the SYT time stamp applies is described in the A/M protocol as being implementation specific. What is required, are phase locked loop techniques to set the sample rate as a division of the elapsed time between consecutive SYT time stamps. The first event in the cluster is presented at the time indicated by the SYT time stamp. Subsequent samples are then presented at the sampling rate determined by the phase locked loop design. This achieves time alignment and sample rate recovery. While this can be readily performed by custom hardware, implementing these phase locked loops in software is computationally expensive, given the throughput of data required.

Figure 26 below illustrates how the use of SYT time stamping provides the mechanisms for time alignment and sample rate recovery.

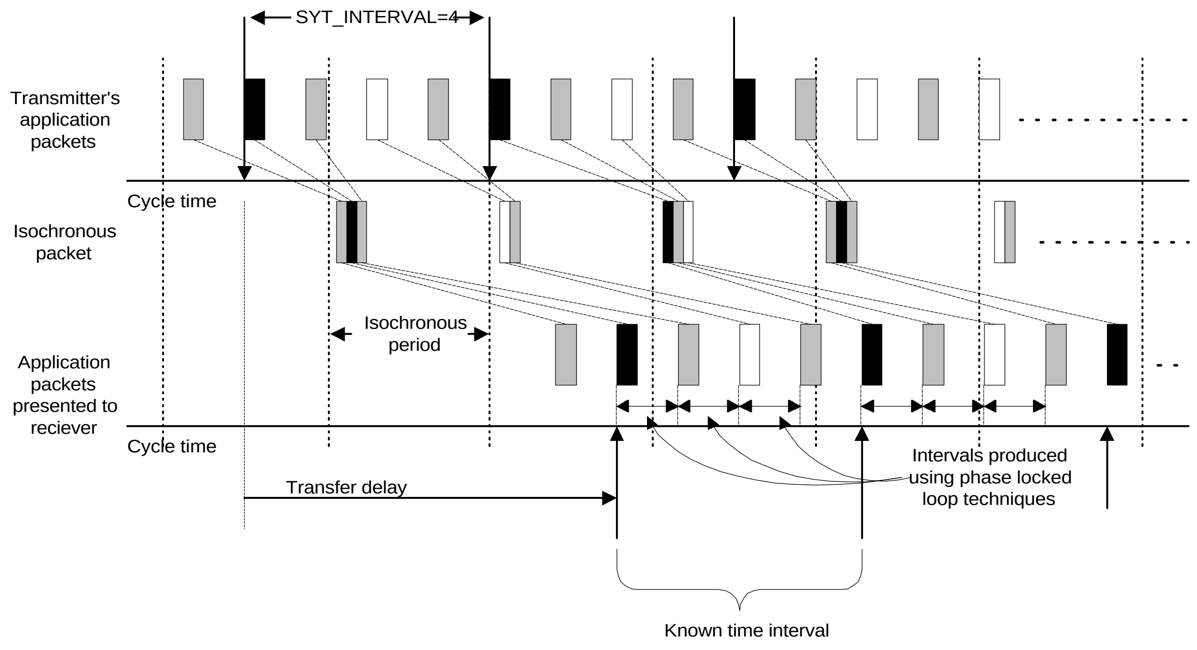

SYT_INTERVAL=4
Transmitter's application packets
Cycle time
Isochronous packet
Isochronous period
Application packets presented to reciever
Cycle time
Transfer delay
Intervals produced using phase locked loop techniques
Known time interval

Figure 26: Sample rate recovery and time alignment using SYT time stamps and non-blocking transmission method.

The key to time alignment and sample rate recovery for SYT time stamping is the sharing of a common clock between nodes – the CYCLE_TIME. Recall that each isochronous capable node is required to implement the CYCLE_TIME register that increments at a rate of 24.576MHz. The cycle start packet, broadcast nominally every 125µs updates all node's CYCLE_TIME registers with a common value. Between the cycle start packets however, the CYCLE_TIME registers are free running. Given that each node's 24.576MHz clock is required to have better than 100ppm (part per million) tolerance, it can be shown that a particular node may lead or lag the cycle master by 25ns at worst. This difference in the CYCLE_TIMER registers translates to jitter in recovered sampling rates. In practise however, this jitter can be reduced to less than 1ns using a proprietary phase locked loop design [Moses 1997].

Other methods for implementing sample rate recovery of audio (not necessarily A/M protocol) transmitted on the 1394 bus are described by [Moses 1997]. The methods described include mechanisms based on deriving sampling rates from the CYCLE_TIMER rate, and buffer filling techniques for recovering sample rates.

4.4 Connection management

Routing of audio and MIDI data on the 1394 bus can be accomplished by the connection management procedures (CMP) defined in IEC 61883-1. The connection management procedures provide an abstraction of isochronous flow management in the form of plugs. The first use of connection management procedures for audio and MIDI routing was described by [Abe, Fujimori 1996].

4.4.1 Plugs and plug control registers

Recall that for isochronous data transmission, each stream of isochronous data is assigned a channel. Conversely, each isochronous channel transmits only one isochronous stream. In the plug model, each isochronous stream is transmitted through one output plug on the transmitting device. An input plug in the receiving device receives the stream. Connections may be point-to-point, or broadcast. Figure 27 below illustrates the concept of a point to point connection.

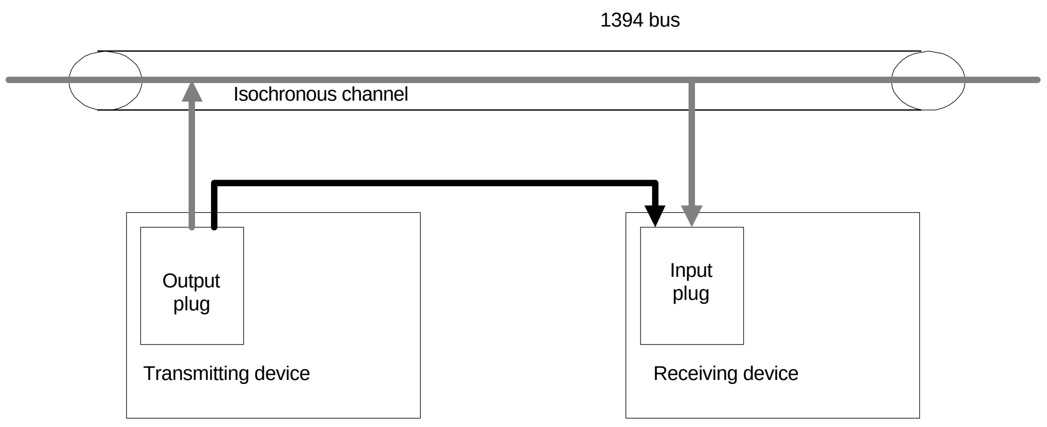


Figure 27: The plug model for point-to-point isochronous flow management

Transmitting devices are required to implement CSR registers called plug control registers. For each output plug, a transmitting device implements an associated output plug control register (oPCR). The transmitting device also implements one output master plug register (oMPR). Receiving devices implement one input plug control register (iPCR) for each input plug, and also one input master plug register (iMPR). The master plug registers control attributes common to all plugs, while the plug control registers control attributes specific to an individual plug. The format of the oPCR is shown below in figure 28.

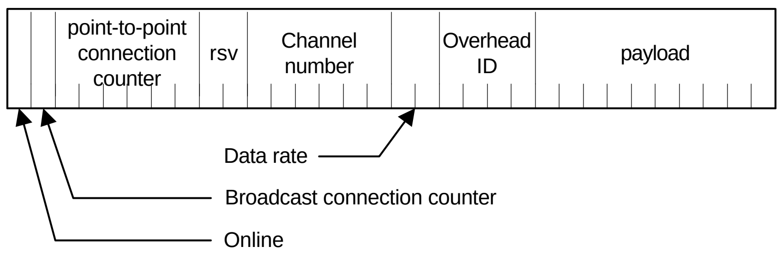

point-to-point connection counter
rsv
Channel number
Overhead ID
payload
Data rate
Broadcast connection counter
Online

Figure 28: The ouput plug control register (oPCR)

The on-line bit always indicates whether the corresponding output plug is on-line or off-line. The transition between these states is described by IEC 61883-1.. The broadcast connection counter always indicates whether a broadcast-out connection (described later) to the output plug exists. The point-to-point connection counter indicates how many input plugs are connected to the output plug. The channel number is the isochronous channel number used. The data rate is an indication of the transmission speed used (S100,S200 or S400). The overhead ID is an indication of the bandwidth required for transmission besides that required by the payload. The payload field indicates the maximum number of quadlets transmitted in a single isochronous cycle. The input plug register is illustrated in figure 29 below.

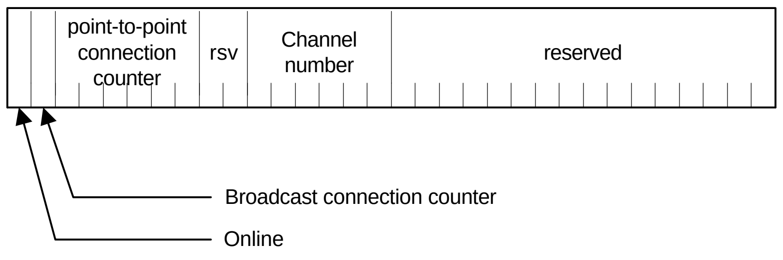

point-to-point connection counter
rsv
Channel number
reserved
Broadcast connection counter
Online

Figure 29: The input plug control register (iPCR)

The fields contained in the iPCR have the same meaning as the corresponding fields in the oPCR. The output master plug register (oMPR) is illustrated in figure 30 below.

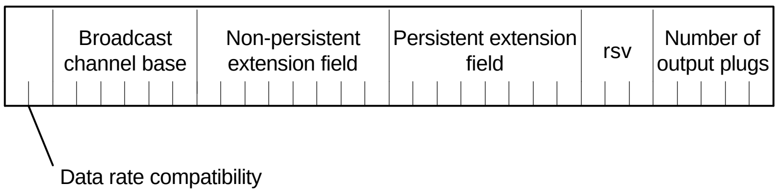


Figure 30: The output master plug register (oMPR)

The number of output plugs indicates the number of output plugs that are implemented by the transmitting device. Both extension fields are reserved for future use. The broadcast channel base specifies the channel of the first plug. Subsequent plugs can then be assigned consecutive channels. The data rate compatibility indicates the maximum speed that the transmitting device can transmit data. The input master plug register is illustrated in figure 31 below.

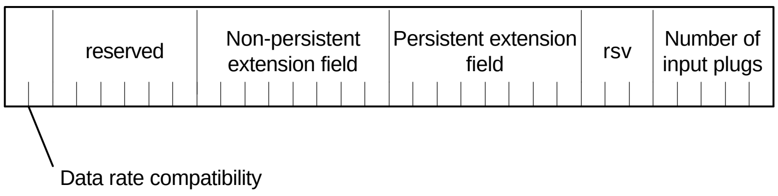


Figure 31: The input master plug register (iMPR)

The meanings of the fields within the iMPR are the same as the corresponding fields within the oMPR.

The values of these plug registers jointly describe the status of isochronous data flows on the 1394 bus. The ways in which these registers can be altered is described by the connection management procedures (CMP).

4.4.2 Connection management procedures

The CMP define procedures for managing point-to-point connections and broadcast connections. For both of these connection types, procedures are described for the action to be taken when commands are received to:

- Create a connection
- Overlay a connection
- Break a connection

The procedures describe the interaction with the isochronous resource manager and the modification of the plug registers. A significant aspect of these connection management procedures, is that point-to-point connections are protected in the sense that only the application that created the connection can break it. Broadcast connections are not protected, in that any application may break a broadcast connection. Full details are provided in the IEC 61883-1 specification.

Commands to create, overlay, and break connections are sent to a node using the Unit Commands defined by the AV/C Digital Interface Command Set [1394 Trade Association 1998]. AV/C commands and responses are transported by a protocol defined in IEC 61883-1 called the Function Control Protocol (FCP).

4.4.3 Function Control Protocol

The FCP, defined in IEC 61883-1, provides a means to encapsulate device commands and responses within 1394 asynchronous block write transactions. The FCP does not define the semantics of control messages, but provides the transport capability for command sets such as AV/C and AES24.

The format of an FCP frame is illustrated below in figure 32.

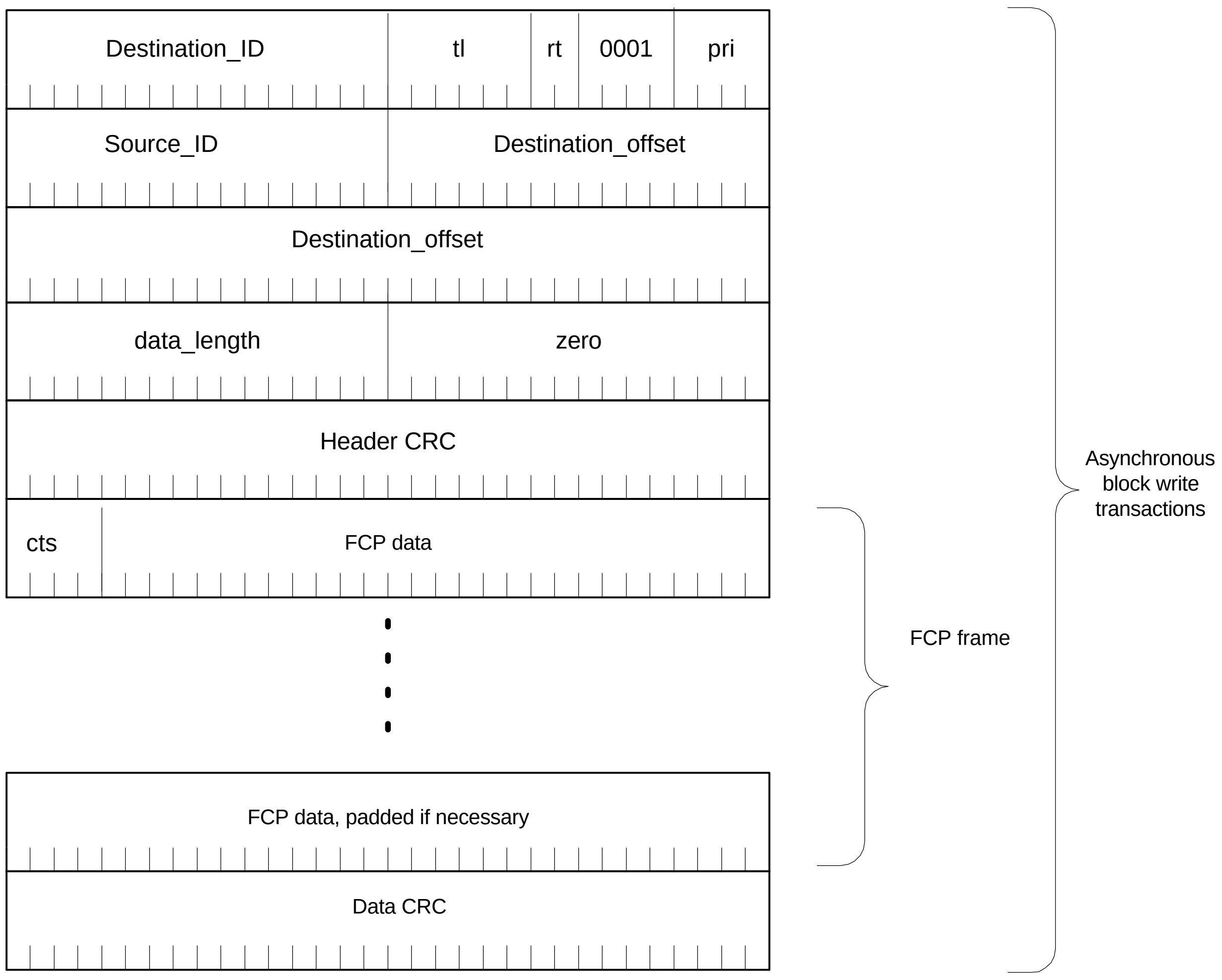


Figure 32: The Function Control Protocol (FCP) frame embedded within an asynchronous block write transaction

The format of the *Destination_ID, tl, rt, pri, source_ID, data_length* and *CRC* fields are as defined for asynchronous block write transactions (refer back to section 2.3.4.4) The *cts* field specifies the command transaction set being transported. The *cts* field has been allocated the value of 0 for the AV/C command set.

Commands are originated from a controller to the FCP_COMMAND register in the node to be controlled, the target. The target returns its response to the FCP_RESPONSE register at the controller. Nodes wishing to use the FCP must implement the FCP_COMMAND and FCP_RESPONSE registers at the published memory offsets. The data length of FCP request and response packets is limited to 512 bytes. Since the FCP does not use sequence numbers, older commands may be lost if new ones are received.

Before considering the AV/C command set and how it is used in connection management, consider briefly another command transport protocol – the Serial Bus Protocol 2 (SBP-2).

4.4.4 SBP-2

The Serial Bus Protocol 2 (SBP-2) is being developed under the auspices of the American National Standards Institute (ANSI). Originally developed to adapt SCSI capabilities to the 1394 bus, SBP-2 evolved to provide mechanisms for the delivery of commands, data, and status independent of the command set used. The "controller" of the FCP is known in SBP-2 terminology as the "initiator". Commands are issued from an initiator to a target. The goals of SBP-2 are [ANSI 1998]:

- Encapsulate commands, data and status information.
- The initiator should not need to consider implementation limits in the target.
- New tasks should not interfere with current tasks.

Moses suggests that since the SBP-2 is much more sophisticated than the FCP, it is arguably a better candidate for transporting command sets [Moses 1998].

4.4.5 AV/C Digital Interface Command Set

AV/C commands and responses are encapsulated within FCP frames. The format of the AV/C command frame is shown below in figure 33.

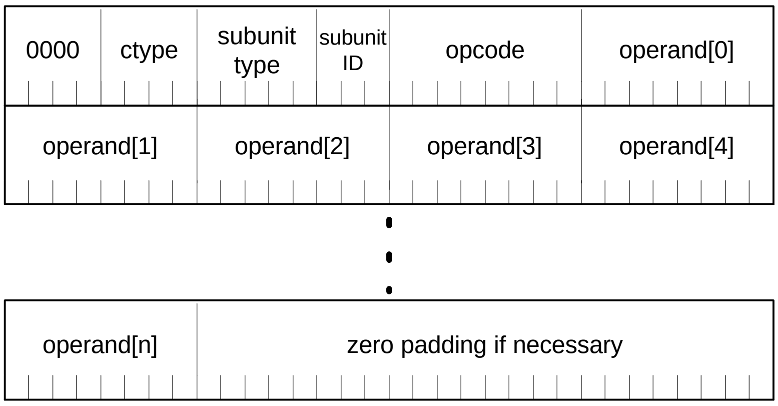


Figure 33: The AV/C Command Frame

The *ctype* field denotes the command type. The possible command types are listed in table 8 below:

Ctype value (hex)	**Command description**
0	CONTROL
1	STATUS
2	SPECIFIC ENQUIRY
3	NOTIFY
4	GENERAL ENQUIRY
5-7	Reserved for future use
8-F	Reserved for response codes

Table 8 : AV/C Command Types

The format of response frames is similar and shown below in figure 34.

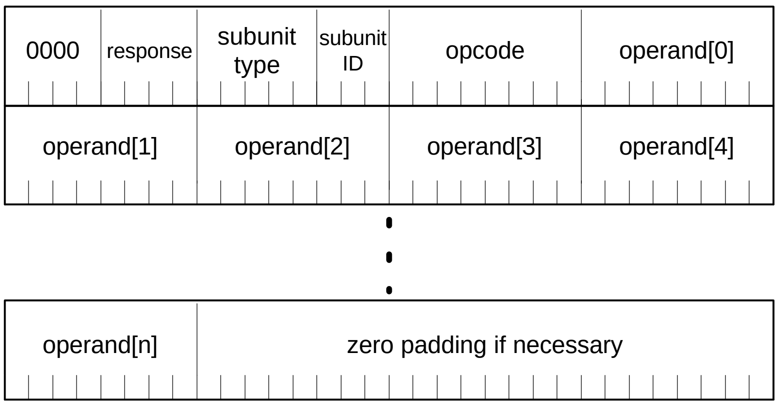


Figure 34: The AV/C Response frame

The response field indicates the response of the target. The possible values are shown below in table 9.

Reponse (hex)	Response description
0-7	Reserved for command types
8	NOT IMPLEMENTED
9	ACCEPTED
A	REJECTED
B	IN TRANSITION
C	CHANGED
D	IMPLEMENTED/STABLE
E	Reserved for future use
F	INTERIM

Table 9 : AV/C Response Types

The mechanism for control in AV/C is for a controller to send a command addressed to a unit, or sub unit within the AV target node. The unit, or subunit within the target node will respond to the controller. Standard devices have been identified by the 1394 Trade Association AV/C working groups and subunit specifications for these devices have been proposed. The specifications define the parameters of the device that can be controlled. The subunit to which a command is addressed is specified in the AV/C frame by the *subunit type* and *subunit ID* fields. Recall in the 1394 architecture that nodes are comprised of one or more units. In the AV/C specification, a unit is decomposed into controllable AV devices.

These devices are termed subunits. If a unit has more than one of a particular type of subunit, then the *subunit ID* field is used to identify the particular instance being addressed.

The AV/C specification currently defines command sets for the following subunit types:

- Video monitor
- Disc recorder/player
- Tape recorder player
- Tuner
- Video camera

Other subunit types are being defined by working groups within the 1394 Trade Association. It is with these standard command sets that interoperability of 1394 devices is achieved.

A set of commands, called unit commands, are addressed to an AV device implemented as a unit within the 1394 architecture. These unit commands are addressed to subunit type 1Fh and subunit ID 7. These unit commands are what are used for implementing the connection management procedures, and audio and MIDI data routing. The defined unit commands, specified by the opcode field, are shown below in table 10.

Command	**Opcode**	**Description**
Channel usage	12h	Report channel usage
Connect	24h	Establishes unspecified streams between plugs and subunits
Connect AV	20h	Establishes an AV stream between plugs and subunits
Connections	22h	Report connection status
Digital input	11h	Make or break broadcast connections
Digital Output	10h	
Disconnect	25h	Break stream connection
Disconnect AV	21h	Break AV stream connection
Input plug signal format	19h	Set or retrieve format of plugs
Output plug signal format	18h	

Subunit info	31h	Report subunit information
Unit info	30h	Report unit information

Table 10 : AV/C Unit commands for implemnting CMP

The formats of the operands for these commands , and the format of responses that targets should produce are specified in the AV/C specification. The AV/C specification enables far more functionality than what has been briefly described here. The intention of this brief discussion is to illustrate the mechanisms for routing audio and MIDI data.

In the preceding sections, it has been shown how audio and MIDI are transmitted as AM824 events within CIP (Common Isochronous Protocol) frames. Routing of these isochronous streams is performed by connection management procedures (CMP). The CMP are implemented by AV/C unit commands, transported in FCP (Function Control Protocol) frames, encapsulated in asynchronous write transactions.

4.5 Computer Integration

In chapter 2, it was discussed how a PC workstation can be utilised for co-ordinating MIDI and audio recording, processing and reproduction. PC audio hardware has evolved to become sufficiently sophisticated to enable professional-quality, multi-channel recording and playback. Unfortunately, the increased sophistication is often coupled to increased configuration complexity. Microsoft's digital audio initiative facilitates a PC audio model that strives for simplicity, flexibility, and optimal performance. The concept behind the digital audio initiative is to facilitate a PC audio model where all audio mixing and routing is done in the digital domain. Using a completely digital system allows the possibility of using an external digital bus to transfer audio. This offers two advantages – the simplification of developing high-fidelity solutions, removed from the electrically noisy PC enclosure, and the possibility of plug 'n play audio modules. The digital audio initiative makes provision for audio to be routed to PCI, USB (Universal Serial Bus), and IEEE 1394 devices.

The digital audio initiative is enabled by the audio architecture in Microsoft's Win32 Driver Model (WDM), currently utilised in Windows 98 and Windows NT 5.0. Figure 35 illustrates the components of the audio and 1394 bus architecture within the WDM that enable audio distribution to external 1394 devices [Microsoft 1998].

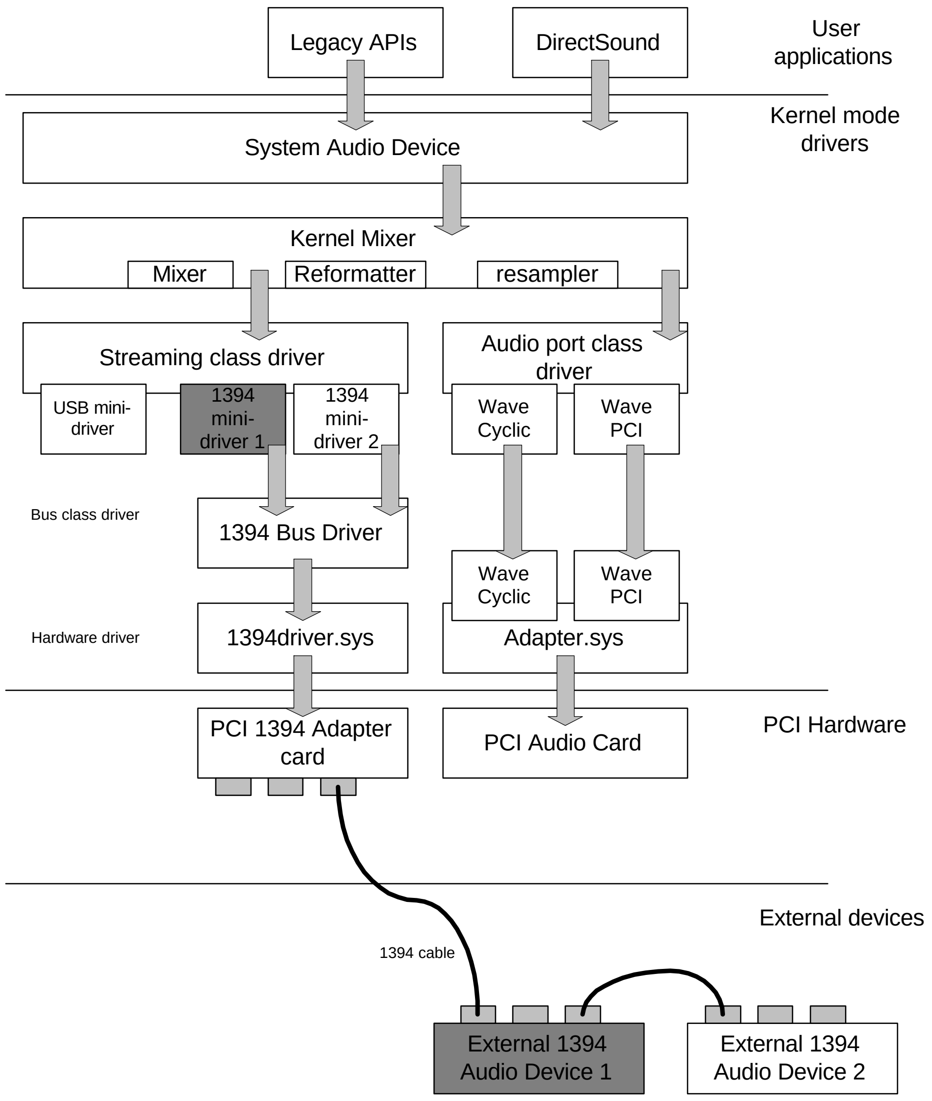


Figure 35 Components of the Win32 Driver Model (WDM) for audio distribution to 1394 devices.

Consider how audio is transmitted to an external 1394 bus device. When the external 1394 device is attached to the bus, a bus reset will occur. The 1394 bus driver detects the reset, and enumerates the devices on the bus. The enumeration process extracts the capabilities of the devices from their configuration ROM and loads any supporting drivers. If the device is determined to be an output audio device, its streaming class minidriver will be dynamically loaded. The external 1394 device will now be available to applications as a “wave out” device. An application can now use the standard DirectSound or legacy APIs to stream audio to the device.

The kernel mixer may perform sample rate conversion (to the frequency required by the audio minidriver) and mixing (if multiple applications are simultaneously using the "wave out" device) to prepare a final stream for the audio streaming miniclass driver. The audio miniclass driver will utilise the 1394 bus driver APIs to transmit the data received from the kernel mixer as clusters of events within CIP formatted, time-stamped isochronous transactions. The 1394 Bus Driver will call the necessary hardware level functions in the hardware driver.

The exciting possibility of the WDM architecture in a studio environment, is having each studio device accessible by a "wave in" or "wave out" device driver by application software. There are however some issues that need resolution, especially for "wave in" devices.

Receiving audio from an external 1394 audio device is considerably more complex than transmitting. The complexity arises from the necessity to recover the sample rate of the incoming audio stream and perform sample rate conversion to an internal timing reference. This is required to be able to reproduce the audio stream later at the correct pitch. If multiple incoming audio streams are to be received simultaneously (as can be done by contemporary PCI audio cards), the feasibility of performing this necessary sample rate recovery and conversion for each stream is questionable.

4.6 Copyright Protection

Although data streams are unlikely to require copy protection in professional studios, it is an aspect of audio distribution that needs to be taken into account. Professional equipment should at least be aware that consumer devices will employ copyright protection mechanisms if seamless interoperability (a goal of 1394 networks) is to be attained. The A/M protocol will be the definitive standard (as IEC 61883-6) for audio distribution using the 1394 bus both in consumer electronics and professional studio environment. In consumer electronic environments, home entertainment systems in particular, there is a need to prevent users from making copies of digital content. In chapter 2, it was discussed how content providers have been hesitant to fully utilise digital media, for the reason that perfect digital copies can be made.

To ease the concerns of content providers, two copy protection mechanisms have been announced that rely on strong encryption technologies to protect data on the 1394 bus. These specifications are known as "5C Digital Transmission Copyright Protection" and "XCA, or Extended Conditional Access". What follows is an informative discussion of these two standards.

4.6.1 The "5C" Digital Transmission Copyright Protection Specification

Hitachi, Intel, Matsushita, Sony and Toshiba have jointly produced the 5C Digital Transmission Copyright Protection (DTCP) specification. This specification defines four fundamental layers of copy protection:

- Copy Control Information.
- Authentication and key exchange.
- Content encryption.
- System renewability.

4.6.1.1 Copy Control Information (CCI)

CCI provides a mechanism for content owners to specify whether their content can be duplicated. There are two methods for providing CCI:

- Using the Encryption Mode Indicator (EMI), a new field defined in an unused part of the isochronous header (using the two most significant bits of the sy field).

- Embedding CCI as part of the content stream. Streams such as MPEG have fields allocated for carrying the associated CCI.

CCI labels data with one of four possible copy protection states:

- Copy freely
- Copy Never (such as a DVD movie)
- Copy one generation (such as a pay TV program)
- No more copies (marked to signify that a one generation copy has been made)

4.6.1.2 Authentication and Key Exchange (AKE)

Before a device will share protected information, receiving devices need to be authenticated. The DTCP defines two levels of AKE:

- Full authentication, which must be used for copy-never content.
- Restricted authentication, which enables the protection of copy-once and no-more copy content with reduced computational requirements.

The full authentication protocol employs the public-key-based Digital Signature Algorithm and the Diffie-Hellman key-exchange algorithm, both of which use Elliptic Curve cryptography. Details of how these algorithms are employed are described by [DTLA 1998].

Restricted authentication is based on a device being able to prove that it holds a secret shared with other devices. The protocol for restricted authentication employs asymmetric key management and common key cryptography techniques, and relies on the use of shared secrets and hash functions to respond to random challenges.

4.6.1.3 Content encryption

The AKE algorithm exchanges keys that are used to implement channel ciphers. All data is encoded before transmission to prevent “sniffing” of the bus. The channel cipher system is negotiated during authentication. As a baseline cipher, the DTCP specifies the use of Hitachi’s M6 cipher. The M6 is a common-key block cipher algorithm based on permutation-substitution. Additional ciphers that can be used are the Modified Blowfish and Data Encryption Standard ciphers.

4.6.1.4 System renewal

Devices that support full authentication can receive and process System Renewal Messages (SRM). These messages are generated by the Digital Transmission Licensing Authority (DTLA) and are distributed via content and new devices. System renewal ensures the long term integrity of the copyprotection system.

The 5C specification is a candidate for providing copy protection of data on the 1394 bus. Recently, the XCA or Extended Conditional Access copyright protection scheme was proposed by Zenith and Thompson [Zenith, Thompson 1998]. Both the 5C, XCA and other methods of copyright protection are under consideration.

4.6.2 Extended Conditional Access

Although no specification is publicly available for XCA at this time of writing, press releases from Zenith Consumer Electronics and Thompson Consumer Electronics have given an indication of what the copy protection scheme is.

The main difference between XCA and 5C is that content is only decrypted at the receiving device – not encrypted by the transmitter. This of course requires that the content is provided in encrypted form. Copies of the encrypted data may be made, but only original or first generation copies (where applicable) will be displayed. This method has been specifically designed to protect against unauthorised copying of television programs and movies and relies on the use of a “smart card” in each device [Zenith,Thompson 1999].

4.7 Adoption of standards and interoperability

The IEC 61883-6 specification will be the defining standard for audio and music data transmission for both consumer and professional applications. In this chapter, the underlying transport and control mechanisms have been described. The degree to which these mechanisms are implemented by different manufacturers poses interoperability concerns. Fortunately, two industry initiatives are working towards providing guaranteed interoperability for compliant devices. The scope of these initiatives is currently geared towards consumer electronics, home entertainment systems in particular. Since there exists considerable overlap between consumer and professional uses of the 1394 bus, these industry initiatives may conceivably extend to guarantee interoperability in professional studio environments. The two industry initiatives are:

- HAVi
- Digital Harmony

4.7.1 HAVi

The HAVi (Home Audio-Video interoperability) core architecture specification is under development by a closed consortium of eight manufacturers calling themselves the Home Network Group. The group consists of: Grundig, Sony, Sharp, Matsushita (Panasonic), Hitachi, Philips, Thompson, and Toshiba. The HAVi architecture provides a set of services that facilitate interoperability and the development of distributed applications on home networks. The HAVi architecture is intended for home networks consisting of, but not limited to, devices supporting the IEEE-1394 and IEC 61883 interface standards [Home Network Group 1998].

The architecture defines "middleware" software elements, their application programming interfaces (APIs) and protocols. The interoperability interfaces of the HAVi specification provide the infrastructure for the routing and processing of isochronous data. Devices that incorporate the HAVi core specification would provide the user with:

- Plug and play connectivity – new devices added to a network recognise the network configuration. The user would not have to adjust any settings.

- Appliance interoperability – devices will seamlessly interoperate. In future devices will be able to share functionality.
- Future-proof appliances – as new application software is developed, consumers will be able to import functions from newly bought equipment into older equipment.

Support for future products and protocols is achieved through the following software mechanisms:

- Persistent information in each device describing its capabilities
- The ability of compliant devices to execute Java bytecode at run-time. This enables devices to "download" code from each other and share functionality, and to upgrade to future protocols that newer products may implement.
- Device independent representation of user elements. This allows a device to "display itself" on another device with display capabilities. A user can then interact with all devices from a single display capable device.

Grundig, Sony, Sharp, Matsushita (Panasonic), Hitachi, Philips, Thompson Multimedia, and Toshiba plan to promote the HAVi core architecture to the entire multimedia industry. Licensing, support, and intellectual property ownership of this developing standard are unknown [Moses 1998].

4.7.2 Digital Harmony

Digital Harmony is a complete interoperability solution for IEEE 1394 home entertainment products. Digital Harmony has a four part roadmap for the development of 1394-compliant products [Digital Harmony 1999]:

- Digital Harmony Protocol Suite: A family of protocols and specifications that, when used together, enable interoperability between digital home entertainment products.
- Digital Harmony Object Code: RISC based firmware that implements the Digital Harmony Protocol Suite.
- Digital Harmony System-On-a-Chip (SOC): Licensees purchase the SOCs from regular suppliers. The SOC runs the object code.

- Digital Harmony Product Certification: All products that comply with the protocol suite must have their compliance verified by a certified test facility before bearing the Digital Harmony logo.

4.8 Summary

This chapter can be summarised by figure 36 below, illustrating the protocol hierarchy for audio and control data distribution using the 1394 bus. Audio and MIDI data can be transmitted as cluster events, containing AM824 data events. The cluster events are transmitted in CIP formatted isochronous transactions. Routing of these isochronous transactions is abstracted by the Connection Management Procedures. Commands to initiate the connection management procedures are sent as AV/C unit commands, transported by the Function Control Protocol (FCP), utilising asynchronous transactions.

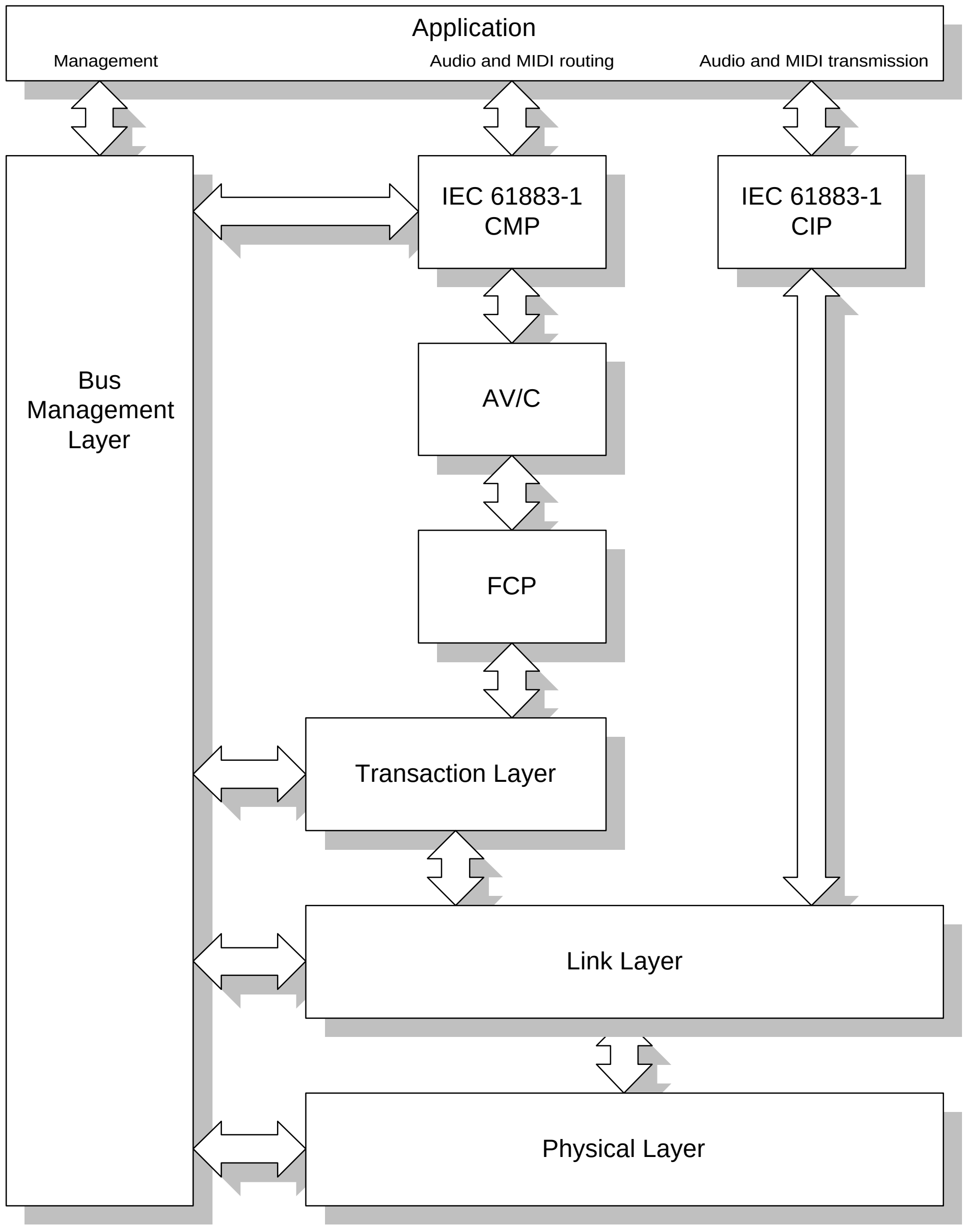


Figure 36 : Protocol layers required for audio and MIDI distribution and routing on the 1394 bus

In the next chapter, a case study implementation of a system utilising some of the protocol layers shown above in figure 36 is presented.

Chapter 5

5 Implementing audio and control data distribution using the 1394 bus.

The previous chapter concluded by illustrating the protocol hierarchy required for 1394 nodes to implement audio and control data distribution on the 1394 bus. Figure 37 below shows the protocol layers implemented in this study to assess the feasibility of audio and MIDI data distribution using the 1394 bus.

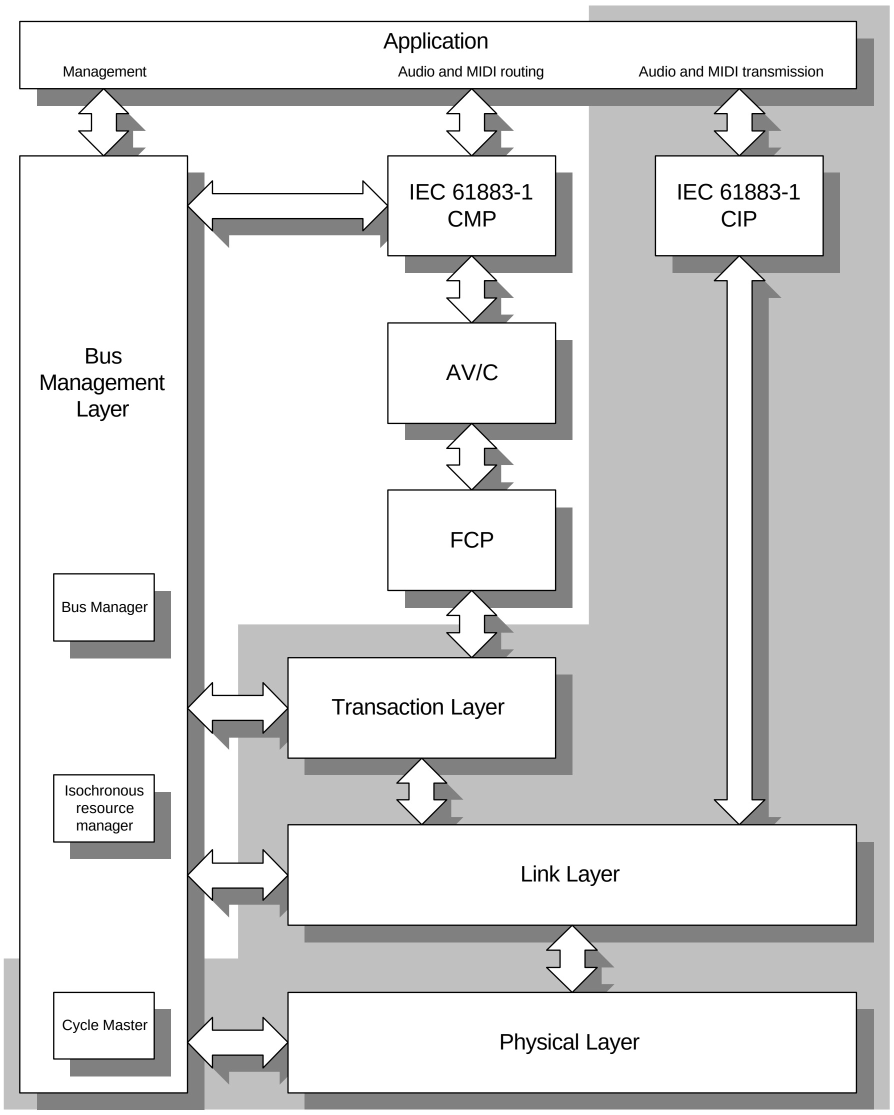


Figure 37: Protocol layers implemented to assess feasibility.

Two nodes with the capabilities shown in figure 37 were implemented, and AM824 audio events were transmitted from the one to the other using CIP formatted isochronous transactions. Audio routing capabilities were not implemented since only two nodes were produced. In this chapter the hardware and software elements, and the roles they performed in implementing this architecture are described.

5.1 Implementation considerations

The CIP operates at a conceptual layer above the defined 1394 bus architecture, making use of the transaction, link, physical, and bus management layers. The physical and link layers are typically implemented in hardware, and the transaction and bus management layers in software [Henehan 1998]. The term “software” is used loosely, referring to microprocessor firmware or programmed ASICs (Application Specific Integrated Circuits) such as FPGAs (Field Programmable Gate Arrays). Newer link layer controllers offer a high degree of integration, encapsulating audio-visual data transmission and bus management functions. Some incorporate the IEC-61883 CIP layer (such as the Philip’s PDI1394LL [Philips 1997] used in this study), and others are even bus manager capable (such as Texas Instrument’s MPEGLynx link layer controller [Texas Instruments 1998]). Consider the physical and link layer hardware options:

5.1.1 The physical layer

When selecting the physical layer component to implement an audio-visual 1394 bus node there are a number of considerations:

- Data transmission speed
- Number of cable ports
- Support for 1394a enhancements

5.1.1.1 Data transmission speed

Moses suggests that all nodes should support the S400 data transmission rate for the following reasons [Moses 1998]:

- A node slower than its peers is a bottleneck. The speed at which two nodes can exchange data is dependent on the minimum speed of each node in the communications path. This may lead to cabling constraints by forcing faster nodes to be directly connected to each other to improve bus performance. Recall that nodes simulate a bus medium from their actual peer-to-peer connectivity. An S400 node cannot be a bottleneck.

- More bandwidth intensive applications become possible. The S400 rate is required to transmit uncompressed digital video, which needs 240Mbps of bandwidth.
- Microsoft's IEEE 1394 Plug 'n Play requirements stipulate the S400 rate.

5.1.1.2 Number of cable ports

Physical layer ICs provide from one to three ports for cable connections. Consider the consequences: A node with only one port is forced to be a leaf device i.e. placed at the end of a branch on the network. Two port devices allow nodes to daisy-chain together, but restrict the number of devices possible on the network (only sixteen hops are permitted). Three port devices allow a full 63 node flexible topology to be implemented. The 1394 bus address space definition limits the number of nodes to 63. Except for battery powered devices, all nodes should use the 6-pin connectors, which are capable of distributing power.

5.1.1.3 Support for 1394a enhancements

1394a defines a number of enhancements beneficial to isochronous data transmissions, notably the arbitrated short reset. For this reason, a physical layer chip should be selected which supports the 1394a enhancements. Such a chip is Texas Instrument's TSB41LV03 [Texas Instruments 1998] that provides 3 ports at the S400 rate with support for the 1394a enhancements. At the time of implementation, 1394a physical layer chips were not available however.

5.1.2 The link layer

When selecting the link layer chip for an audio-visual application there are also a number of considerations:

- Compatibility with physical layer
- Features provided by the link layer
- The host interface
- Buffer sizes and DMA

5.1.2.1 Compatibility with the physical layer

The IEEE 1394 standard specifies the physical/link interface. This should imply that all physical and link layer chips supporting this interface be compatible. In practise however, variations in timing, voltage thresholds, and electrical isolation may create incompatibilities between chips manufactured by different vendors [Moses 1998]. It is therefor better to use physical and link layer chips from the same manufacturer operating at the same voltage, unless guarantees of compatibility can be determined. The link layer chip should also be able to match the speed of the physical layer chip.

5.1.2.2 Features provided by the link layer

The link layer should provide two interfaces to an audio-visual application: asynchronous and isochronous. Recall in figure 37 how the IEC 61883 CIP layer provides an interface between the application and the isochronous link layer interface. Some link layer chips encapsulate this functionality into their isochronous interface, reducing extra hardware and software requirements. Such is the case with Philip's PDI1394LL that was used in our experiment.

Bus management functions are also implemented to varying degrees by link layer chips. Texas Instrument's MPEGLynx, as previously mentioned, provides bus manager capabilities.

The numbers of isochronous channels that can be simultaneously supported and full-duplex operation are further considerations. Half-duplex isochronous operation limits the functionality of the link layer in the plug-model considered earlier. In our experiments, the half-duplex nature of the Philips PDI1394LL link layer controller limited the node to either talking or listening. Subsequent to implementing this study, new link layer controllers have been announced that support full duplex operation. Such is the case with the Philips PDI1394L21 AV link layer controller [Philips 1998].

5.1.2.3 The host interface

Link layer controllers are designed to interface to two types of devices: personal computers and stand-alone nodes. Link layer controllers intended for personal computer applications generally provide an interface optimised for PCI bus usage. Link layer controllers for PC use may be OHCI (Open Host Controller Interface) compliant or use a proprietary interfacing method. The OHCI specifies a set of standardised register addresses and DMA data transfer modes to implement asynchronous and isochronous data transmissions. The OHCI is published by a group called “Promoters of the 1394 Open HCI” that includes Apple Computer, Inc., Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, National Semiconductor Corporation, Sun Microsystems, Inc., and Texas Instruments, Inc. Moses suggests that the OHCI is relatively expensive to implement, but is increasingly popular in personal computer applications, due to reusable software drivers, performance, security, and error handling [Moses 1998].

For stand-alone node implementations, non-OHCI link layer controllers provide an 8,16, or 32-bit interface compatible with the external memory busses of microprocessors. Often the application will dictate the type of processor used, consequently a suitable link layer controller must be chosen to match the processor. A further consideration is whether the asynchronous and isochronous interfaces to the link layer controller are implemented separately or together. A separate isochronous interface allows an A/V codec, such as an MPEG codec, to be directly connected to the link layer controller, alleviating the host processor of the time critical isochronous operations. In this study, a DSP (Digital Signal Processor) was used to provide data to the separate isochronous port on the PDI1394LL.

5.1.2.4 Buffer sizes and DMA

Many link layer controllers use a store-and-forward scheme of data transfer. An entire packet must be buffered in memory before it may be moved out of the buffer. The buffer must be large enough to accommodate the largest packet size. Link layers that implement DMA enable data to be moved from the buffer before entire packets have been received.

5.2 Existing implementations

During the course of implementing this study, three known devices have demonstrated the transmission of audio using the 1394 bus:

- Pavo’s Papaya IEEE 1394 Audio Reference
- Diablo Research’s Sound Fire, and Philips 1394 Audio System.

The Diablo Research devices were prototypes that demonstrated the use of the Philips 1394 AV chipset for audio applications. Subsequent to this study, Pavo’s Papaya device has become commercially available for designers to evaluate audio transmission over the 1394 bus. Considering the details of these products provides a useful comparison to our implementation described in the next section.

5.2.1 Pavo - Papaya – IEEE 1394 Audio Reference

Papaya is a hardware platform for demonstrating standards-compliant audio transmission on the 1394 bus [Pavo 1998]. Papaya provides studio-quality analog and digital audio transmission compliant with both the Audio and Music Data Transmission Protocol and Yamaha’s mLAN specification. Papaya supports linear PCM encoded digital audio and compressed audio formats such as Dolby Digital/AC-3 and DTS (5 channel compressed audio formats). Two audio channels may be transmitted or received over one isochronous channel. Papaya is used as a reference device for drivers under Microsoft’s Windows 98 and NT 5.0 (using the Win32 Driver Model – refer back to section 4.5) and BE Computer’s BeOS operating systems. Figure 38 below shows a block diagram of Papaya.

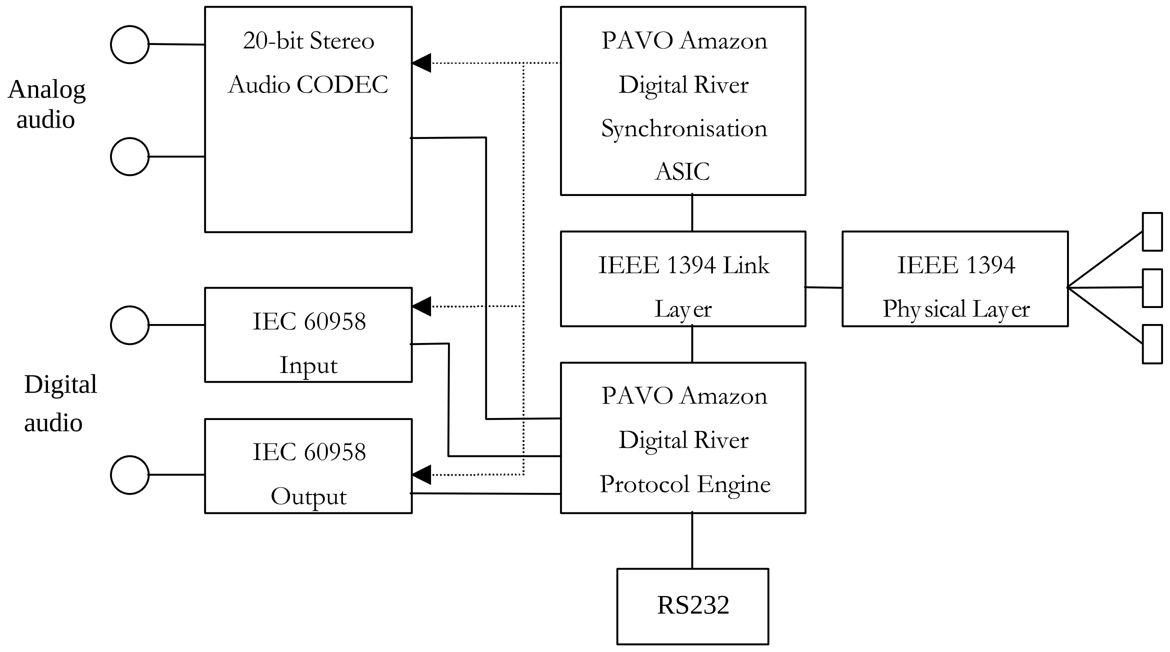

Analog audio
20-bit Stereo Audio CODEC
PAVO Amazon Digital River Synchronisation ASIC
IEEE 1394 Link Layer
IEEE 1394 Physical Layer
Digital audio
IEC 60958 Input
IEC 60958 Output
PAVO Amazon Digital River Protocol Engine
RS232

Figure 38 Block diagram of Papaya. Source: [Pavo 1998]

The protocol engine is implemented on a SHARC DSP and synchronisation and CIP parsing functions are performed by a FPGA. [Pavo 1998] gives further technical details.

5.2.2 Diablo Research – Sound Fire

Developed for Pacific Research and Engineering by Diablo Research, Sound Fire is a 1394 digital audio input/output device, first demonstrated at Comdex 97. Sound Fire provides 6 output- and 2 input channels of AES3/EBU stereo digital audio. Sound Fire connects to a Windows NT workstation via IEEE 1394. The device driver for Sound Fire is accessed using the industry-standard SoundBlaster system calls [Diablo Research Company 1997].

Sound Fire uses the Philips AV 1394 chipset, the PDI1394LL and PDI1394P11 link and physical layers respectively. A 24-bit DSP processor gives Sound Fire the ability to process audio. The clock generation circuitry has the ability to lock to an external source to synchronise the audio with other systems. The inputs are synchronised to this clock using sample rate conversion techniques. Specific implementation details are not provided.

5.2.3 Diablo Research - Philips 1394 Audio System

The Philips 1394 Audio System is a reference design developed by Diablo Research with Philips to demonstrate how the Philips 1394 AV chipset could be used to transmit and receive digital audio over IEEE 1394. This board provides for the playback and recording of digital and analog audio using Yamaha's mLAN protocol. 3 analog and 1 SPDIF outputs are provided with 1 stereo analog and 1 SPDIF stereo inputs. The board also contains a DSP, PLL clock generation, and a RS232 port for MIDI inputs [Diablo Research Company 1997]. Unfortunately, as for Sound Fire, specific implementation details are not provided.

5.3 Audio and MIDI data transmission over 1394, a feasibility study.

5.3.1 Implementation goals

The goals of the implementation were to assess the feasibility of audio and MIDI data distribution over the 1394 bus. In a broader sense, it was hoped to produce a device that could retrofit an existing MIDI synthesiser, or any MIDI controllable studio component, and provide a start for a 1394-based studio.

A device capable of retrofitting a MIDI controllable studio component, such as a sound processor, would need to stream audio and MIDI data to the device, and stream audio and MIDI data being produced by the device to other devices. A synthesiser would need to receive MIDI data and transmit audio data. This necessitates a link layer controller capable of full duplex operation. At the time of implementing this study, link layer controllers capable of full-duplex operation were not available. Consequently half-duplex nodes, transmitting only audio data, were implemented. In combination however, the two implemented nodes provide the functionality for full-duplex operation.

Demonstrating the feasibility of using the 1394 bus in a studio environment requires demonstrating the transmission and reception of the A/M protocol's AM824 events in CIP formatted isochronous transactions.

5.3.2 Hardware architecture

Recall in the A/M protocol that audio can be transmitted as AM824 events within a Common Isochronous Protocol (CIP) formatted isochronous transaction. Implementing this can be broken down into two components:

- Obtaining audio samples and producing AM824 data events.
- Transmitting the AM824 data events on the 1394 bus.

For receiving devices, the process is reversed:

- Receiving the AM824 data events from the 1394 bus.
- Reconstructing audio samples from the AM824 data events.

These components, in the case of both transmitting and receiving devices, can be described as an audio processing component, and a 1394 bus component.

The intention in this study was not to design new hardware, but to utilise existing evaluation boards. The prime motivation for doing this was to be assured of a stable hardware configuration upon which software could be developed. Since the 1394 bus technology is so new, there were only a couple of options for the 1394 bus component. At the time of this implementation, the Philips 1394 link layer controller was unique in performing CIP header processing in hardware. For this reason, the Vitana evaluation board, incorporating the Philips link and physical layer chips was selected.

The Vitana 1394 RDK (Research Development Kit) EVM (Evaluation Module) provides a flexible environment for developing IEEE 1394 applications using the Philips PDI1394LL and PDI1394P11 link and physical layer chipsets [Vitana 1998]. These link and physical layer chips are collectively known as the Philips AV chipset.

A block diagram of the Vitana evaluation board is shown below in figure 39.

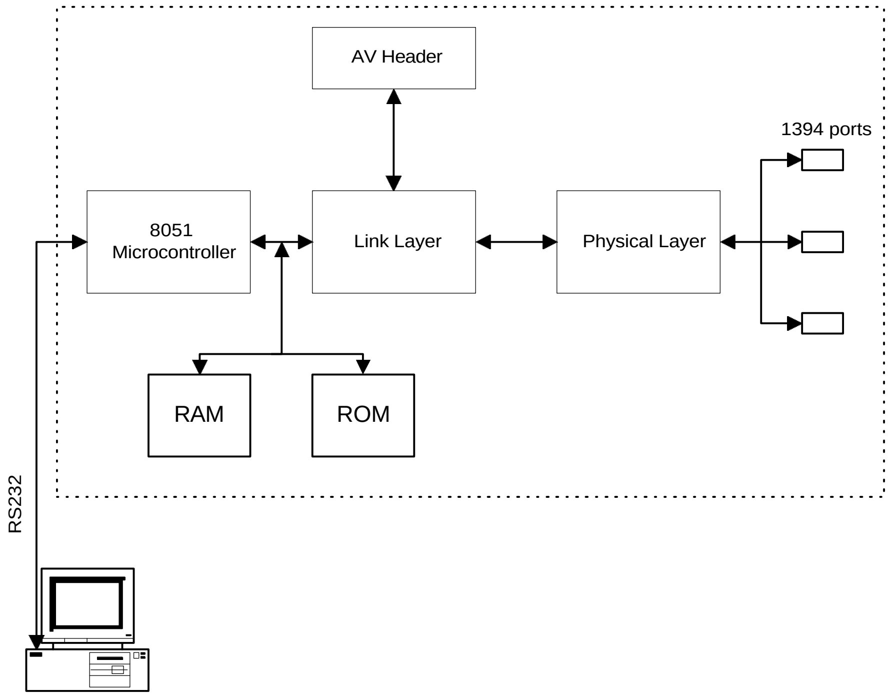


Figure 39: A block diagram of the Vitana 1394 evaluation board.

The key for flexible use of this evaluation board, is the AV header. This allows access to the AV port on the Philips link layer controller, by an external device. Data presented at the AV header is transmitted as CIP formatted isochronous transactions. There are a number of registers in the Philips link layer controller that configure the transmitting and receiving parameters of CIP packets [Philips 1997]. The 8051 microcontroller allows software on the host PC, connected via RS-232, to configure these. This configuration process is discussed later in section 5.3.3.2.

Having the AV header in place, hardware was needed that could obtain audio samples and present these as AM824 events on the AV header, and reconstruct audio samples from AM824 received from the AV header. A Motorola 56002 DSP (Digital Signal Processor) evaluation board was chosen to perform this function.

The Motorola 56002 EVM (evaluation module) board contains a Motorola 56002 DSP, a Crystal Semiconductors audio codec, and various elements of supporting circuitry [Motorola 1996]. While there are many possible hardware solutions that could drive the AV header on the Philips link layer controller, the Motorola board provided most of the required functionality at a low cost. The enabling features of the Motorola board being suitable for this purpose, are the general purpose input/output capabilities of the DSP, coupled to headers on the evaluation board allowing for easy connection to the AV header. The general purpose I/O capabilities of the DSP allow the necessary handshaking with the AV interface on the Philips link layer controller to be performed. The usability of the Motorola board is enhanced by the provided development tools that include a compiler, assembler, linker and debugger.

The essential elements of the combined hardware configuration are shown below in figure 40.

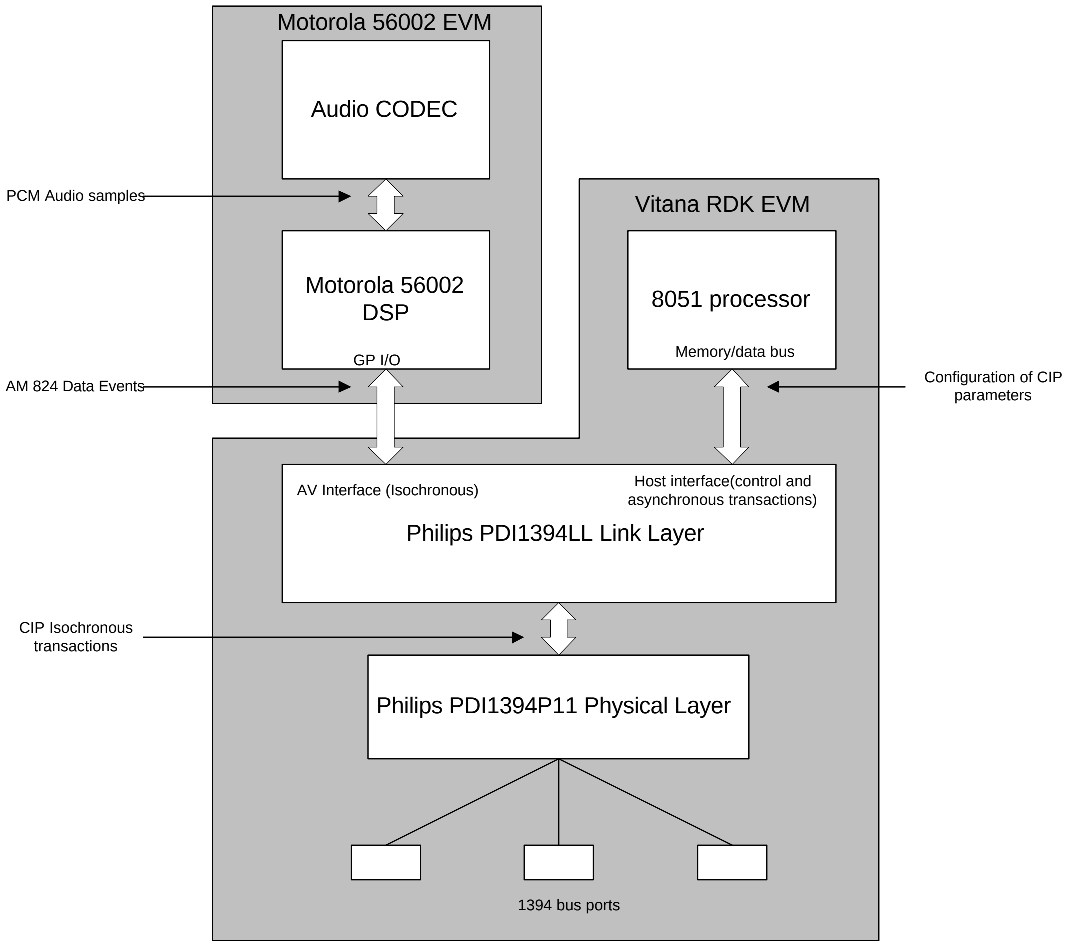

Motorola 56002 EVM
Audio CODEC
PCM Audio samples
Motorola 56002 DSP
GP I/O
AM 824 Data Events
Vitana RDK EVM
8051 processor
Memory/data bus
Configuration of CIP parameters
AV Interface (Isochronous)
Host interface(control and asynchronous transactions)
Philips PDI1394LL Link Layer
CIP Isochronous transactions
Philips PDI1394P11 Physical Layer
1394 bus ports

Figure 40: Essential hardware components in implementation study

The address and data busses of the 8051 microcontroller are connected via address decoding circuitry to the host interface of the Philips link layer controller. The 8051 processor controls the Philips link layer controller by reading from, and writing values into registers within the link layer controller. These registers enable the setting of parameters for isochronous transactions, and directly performing asynchronous transactions. The link layer controller uses interrupt mechanisms to communicate back to the 8051 microcontroller.

The AV header on the Philips link layer controller is connected via a custom ribbon cable to Port B on the Motorola DSP. Port B is the name given to a set of pins on the Motorola DSP that can perform general purpose input and output. Each pin is individually addressable allowing the necessary handshaking to be performed between the Philips link layer controller and the Motorola DSP.

To demonstrate the transmission of AM824 audio data events over the 1394 bus, two nodes were implemented with the hardware described above. The combined configuration is illustrated below in figure 41.

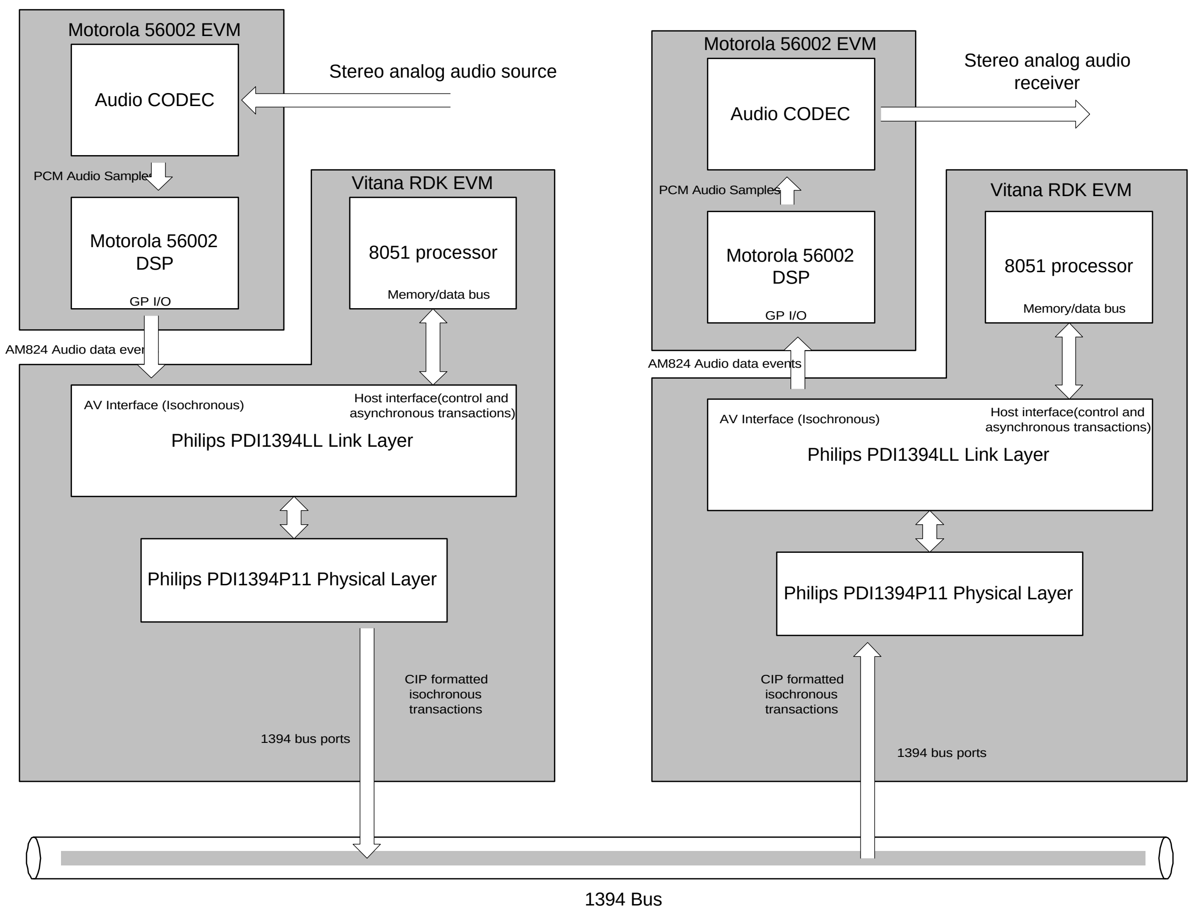

Motorola 56002 EVM
Audio CODEC
Stereo analog audio source
PCM Audio Samples
Motorola 56002 DSP
GP I/O
Vitana RDK EVM
8051 processor
Memory/data bus
AM824 Audio data events
AV Interface (Isochronous)
Host interface(control and asynchronous transactions)
Philips PDI1394LL Link Layer
Philips PDI1394P11 Physical Layer
CIP formatted isochronous transactions
1394 bus ports
Motorola 56002 EVM
Audio CODEC
Stereo analog audio receiver
PCM Audio Samples
Motorola 56002 DSP
GP I/O
Vitana RDK EVM
8051 processor
Memory/data bus
AM824 Audio data events
AV Interface (Isochronous)
Host interface(control and asynchronous transactions)
Philips PDI1394LL Link Layer
Philips PDI1394P11 Physical Layer
CIP formatted isochronous transactions
1394 bus ports
1394 Bus

Figure 41: Hardware used in AM824 transmission.

As previously mentioned, demonstrating the transmission of audio AM824 events is required to show the usability of this technology in a studio environment. In figure 41 above, the Motorola DSP on the transmitting node (on the left), receives PCM audio samples from the codec. These PCM audio samples are then extended to AM824 events and transmitted to the AV interface on the Philips link layer controller. The Philips link layer controller collects these clusters of AM824 events and transmits CIP formatted isochronous transactions. The parameters of these isochronous transactions have been previously established by the 8051 communicating serially with software on a host PC.

The link layer controller on the receiving node (on the right in figure 41), is set to listen for isochronous data on a particular isochronous channel. Any CIP formatted transactions received are parsed by the link layer controller, and the AM824 data events are buffered and made available to the DSP via the AV header. The receiving DSP extracts the AM824 events from the link layer controller and reconstructs PCM audio samples which are then presented to the codec. The details of interactions between the DSP and the AV port on the link layer controller, as well as the details of setting up the parameters of the CIP formatted isochronous transactions are described in the next section.

5.3.3 Software architecture

The software elements required to enable the implementation shown above in figure 41 are:

- The Motorola DSP – Audio input and output; AM824 data event generation and parsing; and driving the AV interface of the Philips PDI1394LL chip.
- The 8051 microprocessor – configuration of isochronous data transmissions.

5.3.3.1 The Motorola DSP

Two separate programs were implemented on each of the nodes illustrated in figure 41. One for transmitting audio data onto the 1394 bus, and the other for receiving a digital audio stream from the 1394 bus and playing it out.

The Motorola 56002 incorporates OnCE (On Chip Emulation) technology, enabling assembled code to be uploaded to, and executed from onboard RAM. The technology also allows the processor to be suspended, and the contents of internal registers and memory addresses to be examined. Provision is made for an EPROM for more permanent code. All code written for the Motorola DSP was developed in its native assembler so that the timing necessary for driving the Philips AV interface could be achieved.

5.3.3.1.1 Transmitting audio onto the 1394 bus

The first step in the process of transmitting audio onto the 1394 bus is obtaining PCM audio samples. In our implementation, the audio codec was set to sample at 48kHz with 16 bits of resolution. The sample rate is determined by the Motorola DSP, using interrupt mechanisms, and can only be set to certain discrete values. It will be discussed later how this hinders sample rate recovery at the receiver.

Once a single stereo PCM audio sample is available, each of the left and right PCM audio samples are extended to 32bit AM824 data events. This AM824 data is then transmitted 8 bits at a time to the Philips AV interface, using Port B on the Motorola DSP. This process is illustrated in figure 42 below.

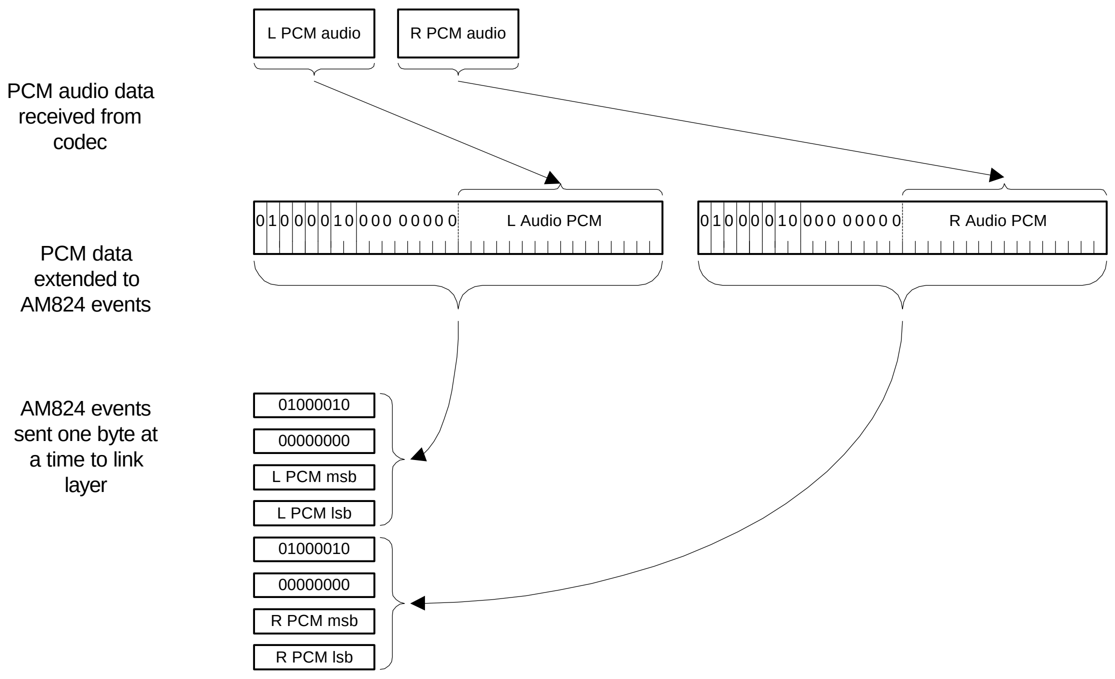


Figure 42: Placing AM824 events in the link layer buffer

The Philips AV interface is 8 bits wide for data. Passing data to the isochronous port on the link layer chip requires a minimum of three signals used for handshaking – a clock, validity indicator, and start of frame indicator. These signals are named av_clk, av_valid, and av_sync respectively [Philips 1997]. The operation is as follows:

- Data present on the eight data lines is latched on the rising edge of av_clk, providing av_valid is asserted.
- Av_sync is asserted to signify the start of a cluster. It is recognised on the rising edge of av_clk and must be accompanied by av_valid.

There are a number of other handshaking signals that can be used to provide additional functionality.

The eight data bits, av_clk, av_valid, and av_sync are all directly mapped to Port B on the Motorola DSP. Port B on the Motorola DSP is what is known as a memory mapped peripheral. The electrical state of any pin directly corresponds with a bit pattern in a known memory location. To assert one of the handshaking signals requires the corresponding bit to be set at this memory address. Transmitting each piece of 8 bit data to the Philips link layer controller is described by the pseudocode below.

```
Clear av_valid bit on Port B
Set up each data bit on Port B
Assert av_valid bit on Port B
Assert av_clk bit on Port B
Wait a short interval
Clear av_clk bit on Port B
```

It can be seen from the above pseudocode how the DSP pushes the data to the AV port on the Philips link layer controller. The rate at which data is transmitted to the link layer controller is entirely determined by the DSP. The process needs to be sufficiently fast to transmit all data within a single sample period.

5.3.3.1.2 Receiving audio from the 1394 bus

To reconstruct PCM audio samples from received AM824 data events, the reverse of the transmitting process needs to be performed. That is, the AM824 events needs to be extracted from the link layer controller, the PCM audio samples parsed from the AM824 events and given to the audio codec. This process is illustrated in figure 43 below.

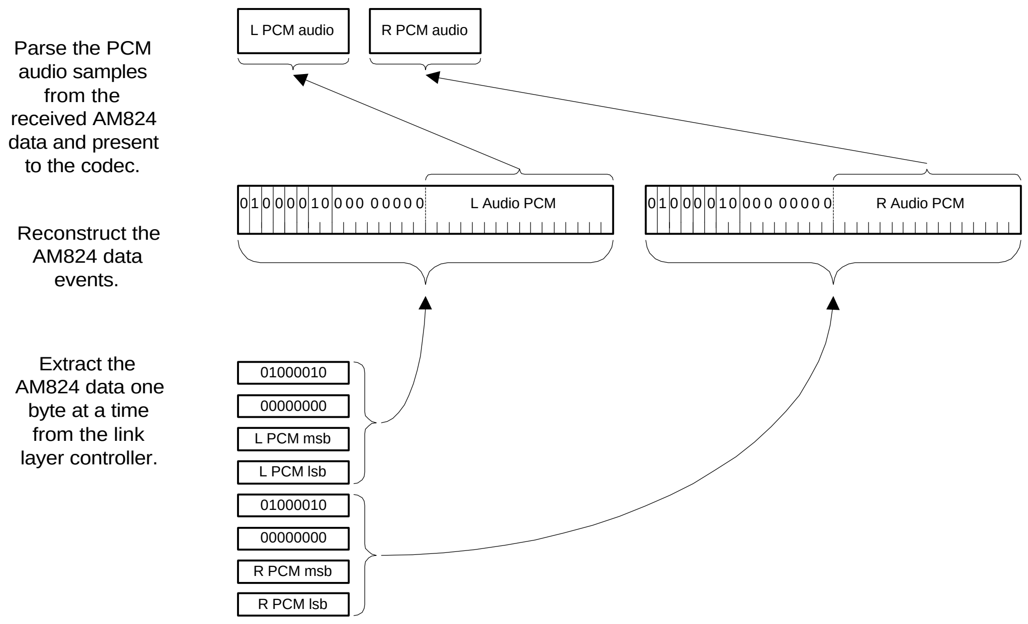

Parse the PCM audio samples from the received AM824 data and present to the codec.
Reconstruct the AM824 data events.
Extract the AM824 data one byte at a time from the link layer controller.
L PCM audio
R PCM audio
0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 L Audio PCM
0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 R Audio PCM
01000010
00000000
L PCM msb
L PCM lsb
01000010
00000000
R PCM msb
R PCM lsb

Figure 43: Receiving AM824 events and reconstructing PCM audio samples.

Similarly to transmitting audio, a minimum of three hand-shaking signals are required to receive data from the isochronous port of the link layer controller. The signals are av_clk, av_valid and av_sync. Av_clk is the only output signal from the DSP, the others are inputs. The first step is to synchronise to the start of a cluster. Clocking the link layer controller using av_clk until the link layer asserts av_sync and av_valid does this. This simultaneous assertion indicates that the first byte of an application frame is present on the data lines. The link layer is then clocked again. "Clocked" refers to driving av_clk low and then high. If av_valid is asserted, then the next byte of data is present on the data lines. This process repeats until the cluster of AM824 events is extracted. The contents of the AM824 event is inspected, and if an audio sample, the sample is stripped from the AM824 event (the 16 most significant bits are discarded). The audio sample is placed in the codec output register, when the codec is ready. The process of extracting data from the link layer controller is described by the following pseudocode:

```
clear av_clk
while (1) do
      set av_clk
      while !(av_sync AND av_valid) do
            {     clear av_clk
                  wait a bit
                  set av_clk
            }
```

```
get first eight bits of AM824 event
clear av_clk
set av_clk
while !(av_valid) do
      {     clear av_clk
            wait a bit
            set av_clk
      }
get next eight bits of AM824 event

etc
```

5.3.3.2 Isochronous transaction configuration

The Vitana 1394 RDK EVM boards are supplied with a real-time monitoring and control program (running on the 8051 controller) which interfaces to a PC via RS232. This software gives the user the perspective of the Philips link layer controller as seen by the 8051 microcontroller. Amongst other functions, the ability to modify and inspect all the link layer registers is provided. As previously mentioned, setting these registers is how the parameters of CIP formatted isochronous transactions are configured.

For transmitting CIP formatted isochronous transactions, the Philips link layer controller needs to perform a number of functions. Conceptually, the AM824 data events given to the AV port by the DSP need to be gathered into an isochronous payload. The parameters of the isochronous header and the CIP header also need to be given values. This required functionality is illustrated in figure 44 below.

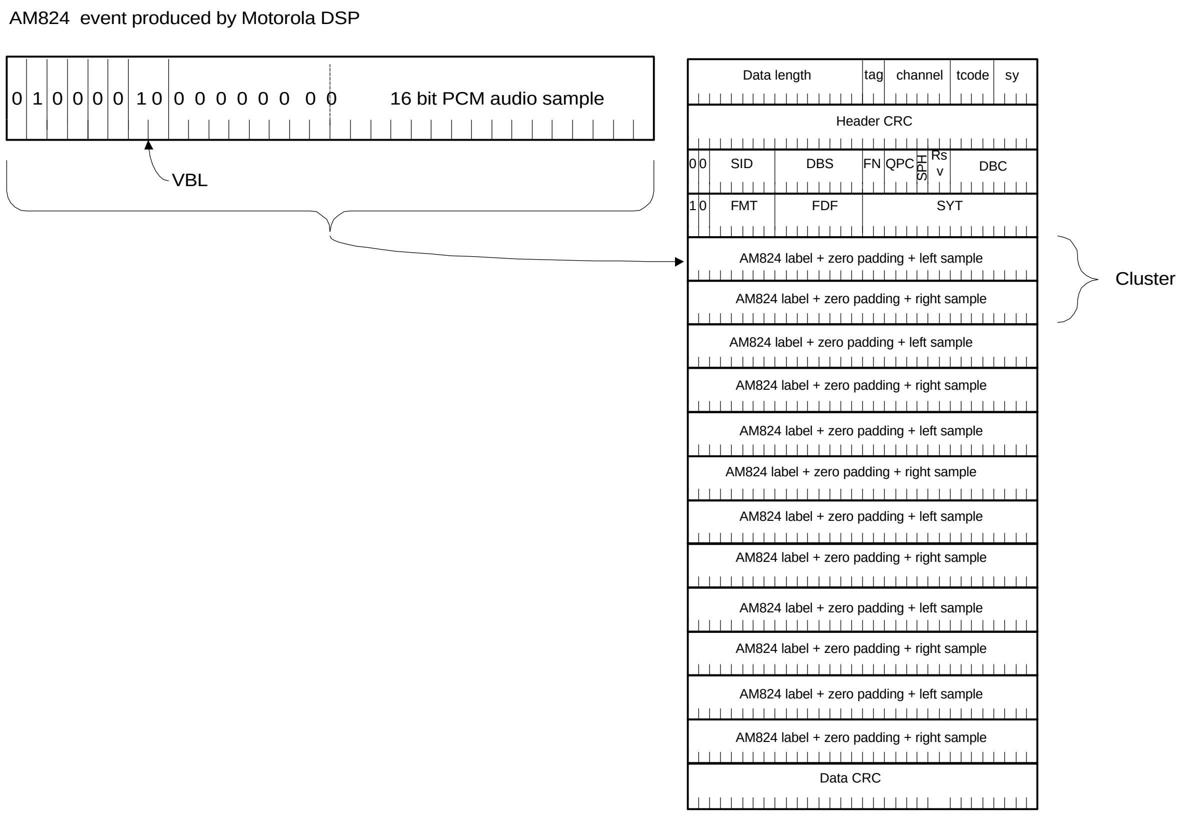


Figure 44: Format of isochronous transactions showing AM824 data and clusters

For transmitting audio, the transmission parameters need to be set according to the rules defined by the A/M protocol. As previously mentioned, audio was sampled at 48kHz with 16 bit resolution. The first parameter to define is the data block size, or dbs. Recall that the dbs is the size in quadlets of the cluster events being transmitted. In this case, the cluster size is two – a 32 bit AM824 event carrying the left PCM audio, and another for the right PCM audio data. So dbs=2.

At the receiver, this dbs information is used in conjunction with the data length to determine how many clusters are contained in the payload. The number of clusters in the payload is determined by the transmitter, and is set depending on the transmission method. In this implementation, the blocking transmission method was used. Recall that the blocking transmission method transmits non-empty isochronous transactions when a fixed number of events have been received. The number of clusters per non-empty isochronous transactions was set to 6. This value was calculated by dividing the frequency of the clusters, by the frequency of transmission (48kHz/8kHz respectively). Using the host software on the PC, the number of cluster transmitted per packet is determined by setting the max_block link layer register.

The values of the other CIP parameters assume the values defined by the A/M protocol. The values and descriptions are shown below in table 11.

Field	**Value**	**Comment**
FMT	10h	This value indicates that the format is the A/M protocol
FN	0	Source packets are not divided into fractions
QPC	0	No quadlet padding, since all data is 32 bit aligned
SPH	0	No transport delay time stamps are present.
SYT	X	The presentation time of the specified event (see 5.3.3.3)

Table 11 : Values of CIP parameters as defined by the A/M protocol.

At the receiving node, configuring the link layer controller to receive data is considerably simpler. Using the host software and communicating via RS-232 to the 8051 microcontroller, the link layer needs to be told what isochronous channel to listen on. This would of course be set to what the transmitter was previously set to. Sufficient information is transmitted within the CIP header to give the receiving node all the necessary information about the content of the stream.

5.3.3.3 Synchronisation

Throughout this thesis the need for synchronisation has been a significant issue. The two aspects of synchronisation required are sample rate recovery and time alignment. Recall that in the A/M protocol, synchronisation is implemented by the transmitter placing SYT timestamps, indicating the intended presentation time, in the SYT field. Between SYT time stamps, the receiver is required to estimate the sampling frequency. This is illustrated below in figure 45.

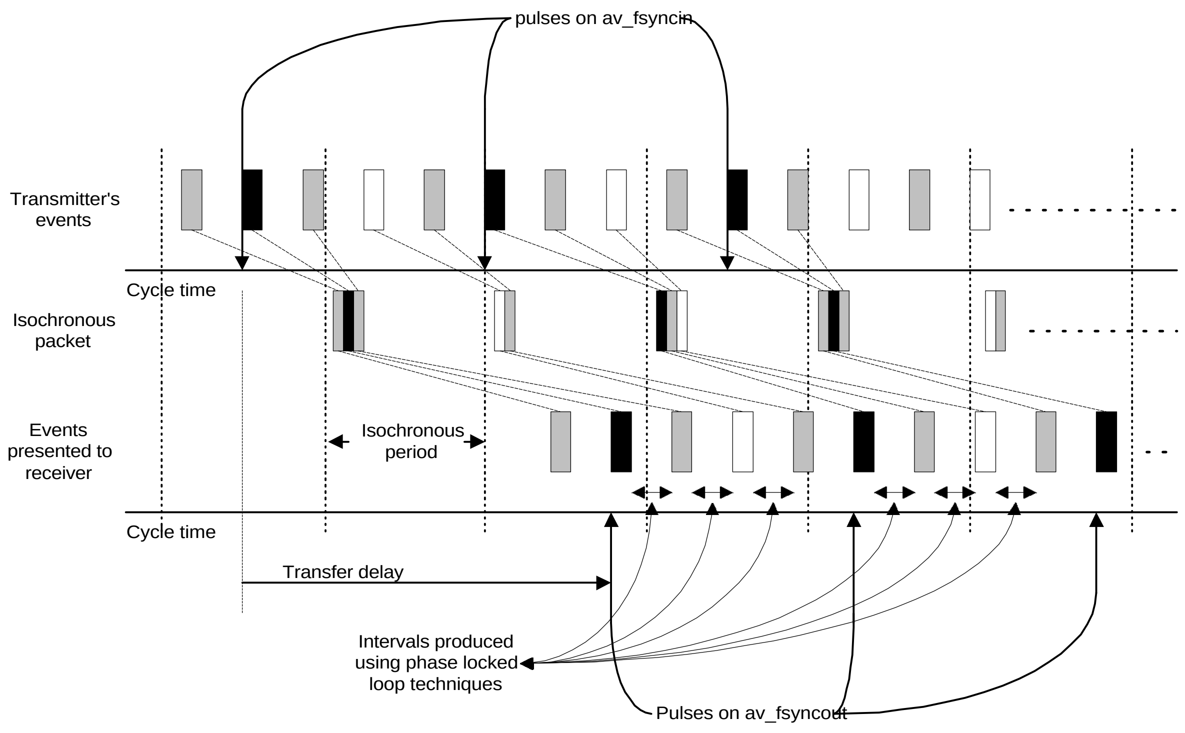


Figure 45: Philips AV chipset - SYT timestamping

The Philips PDI1394LL chip makes provision for SYT time stamping, but with certain limitations. A pin is provided on the AV interface called av_fsyncin. If a transmitting application asserts this pin, then a SYT time stamp is added to the next transmitted cluster event. The application would assert this pin every SYT_INTERVAL (determined by the sampling rate) number of clusters. The SYT timestamp is calculated in the normal way (refer back to 4.3) from the CYCLE_TIME, and incremented with a fixed value such that the intended presentation time is 375μs later (corresponding to three isochronous periods). The fixed transport delay imposed by the Philips AV link layer controller limits the flexibility of time alignment.

The receiving link layer chip will produce a pulse on a pin called av_fsyncout when its local cycle time is the time of a received SYT time stamp. In this way, the receiving application can determine the sampling rate using phase locked loop techniques. This would be done by setting the inter-sample time to the difference between received SYT time stamps divided by the SYT_INTERVAL.

A limitation with the Motorola EVM board, as previously mentioned, is the inability to set arbitrary sampling rates. Sampling rates can only be set at a few discrete values. Being “hard-wired”, there is no means to add additional PLL sample rate recovery circuitry either. In this implementation, the transmitter’s sample rate is nominally known. The sample rates of the both the transmitting and receiving codecs are interrupt driven at a fixed frequency (48kHz). Although the two Motorola EVM boards used are identical, slight variations in components that determine the clock frequency will result in a small difference between the sample rates of the two boards (although they are set to be the same).

In this implementation, crude sample rate conversion is performed: If the transmitter’s sample rate is slower than the receiver’s, the receiver will repeat the previous sample when no new sample data is available. If the receiver is slower than the transmitter, samples are simply dropped. The samples are dropped by the receiving link layer chip when its buffer becomes full. Although very crude and simplistic, the sample rates of the transmitter and receiver were in practise close enough that no audible artefacts could be perceived.

To implement sample rate recovery, additional hardware would need to be developed to implement phased locked loop techniques between pulses on the av_fsyncout pin. Sample rate recovery of audio transmitted over the 1394 bus, using phase locked loop techniques is demonstrated by Pavo’s Papaya.

5.4 Discussion of implementation

A hardware and software solution that demonstrates the transmission of AM824 events within CIP formatted isochronous transactions was presented in this chapter. The main limitations with this implementation are the inability to perform sample rate recovery, and the half-duplex nature of each node. In combination however, both nodes can be used to provide full duplex functionality.

In November 1998, after this implementation study had been finished, Philips announced a full-duplex version of the PDI1394LL chip, the PDI1394L21 [Philips 1998]. This chip provides two separate AV interfaces, each with the functionality of the AV interface described in this study. Using this new link layer controller with this implementations architecture would solve the problem of half-duplex operation. What this new implementation would still lack however, is the ability to perform sample rate recovery.

In the next chapter, consideration is given to mechanisms whereby future remote 1394 studios could be integrated. The discussion is based on the current status of developing standards and technologies. A compelling scenario is presented, the realisation of which depends on the completion of standards and maturation of the enabling technologies.

Chapter 6

6 Connecting remote 1394 based studios – a feasibility study

The previous chapters have demonstrated the feasibility of distributing audio and MIDI in local studio environments by utilising the 1394 bus. It has been shown how the A/M protocol provides the transport mechanisms for up to 64 channels of multi-channel audio and MIDI data. Connection Management Procedures, defined by IEC 61883-1, enable routing of these multi-channel streams. Provision for integration with contemporary music production software has also been discussed. The prospect is compelling – a studio environment with devices all interconnected by one single cable, in a flexible "daisy-chain" like topology.

Pending the completion of standards, and maturation of enabling 1394 bus hardware components, it is likely that future studio devices will incorporate this technology. Given that 1394 technology is not yet available to consumers, it may be premature to consider connecting 1394 based studios at remote sites. Developing standards and technologies however, provide an exciting scenario of remotely connected studios that utilise the 1394 bus technology.

6.1 Remote studio connectivity

There are two main reasons why remote studio connectivity is desirable:

- To share studio components.
- To enable remote collaboration in studio production projects.

Moving away from the "project studio" environment, consider commercial and educational music studio complexes. Typically, these will have a number of studios where studio components are utilised. A particular studio component may need to be used in a different studio for a limited period. If the required studio component is not being used, it is typically unplugged and installed into the other studio where it is needed. The problem of having studio components moving around a facility has necessitated manual booking schemes to maintain control. If all studio component are permanently configured in one location, they may be under-utilised. A studio sharing system utilising networking technology, which overcomes these problems is demonstrated by [Foss et al. 1994]. In this system, studio devices are connected via MIDI to PC workstations called MIDINet units. These MIDINet units provide the mechanism for MIDI data to be distributed to studio devices attached to other MIDINet units. A number of MIDINet units can be connected via ethernet. The MIDINet unit presents a simplified user interface of the distributed studio components.

6.1.1 A studio sharing system using MIDINet units

Figure 46 below illustrates the architecture of the studio sharing system using MIDINet units.

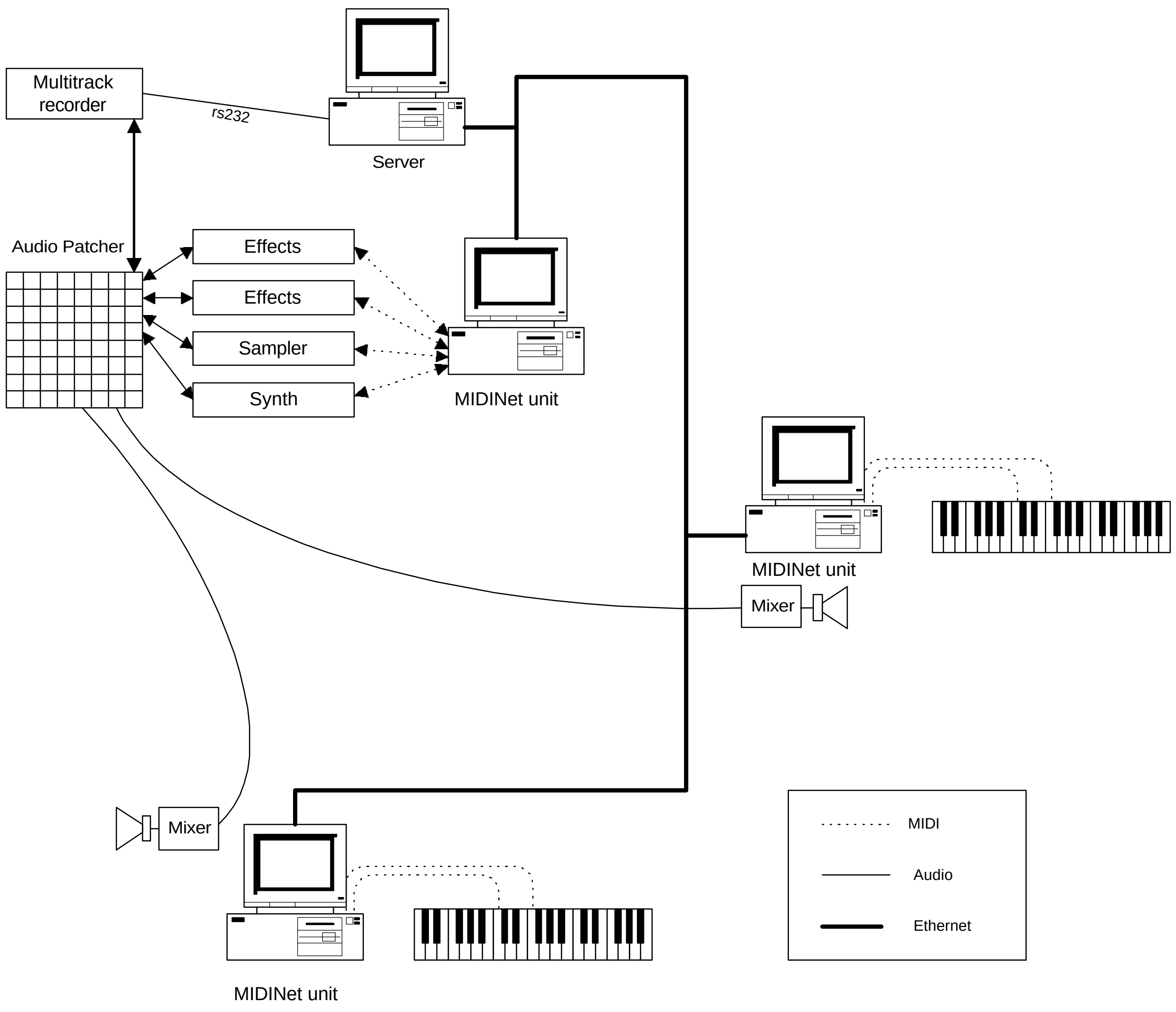


Figure 46: Sharing studio resources using MIDINet units

A user can play the MIDI keyboard at a given workstation, and the MIDINet units will route the MIDI message over ethernet to the correct device. The server functions as a remote studio engineer, and will patch the correct audio feed back to the user. Herein lies the major problem with this system: Potentially long runs of analog cabling from the audio patcher back to each workstation in a “star” type topology. These runs of audio cabling add to the already significant number of different physical control and data carriers. Furthermore, the usable length of the two-channel analog cable is limited by noise susceptibility.

For further details, please refer to [Foss, Wilks 1994].

6.1.2 Lawo’s Distributed Studio Network

On a global scale, Lawo’s Distributed Studio Network (DSN) provides the mechanisms for inter-studio connectivity. The Distributed Network Architecture enables MADI (refer to 2.2) formatted digital audio to be transmitted via ATM. The architecture is illustrated in figure 47 below.

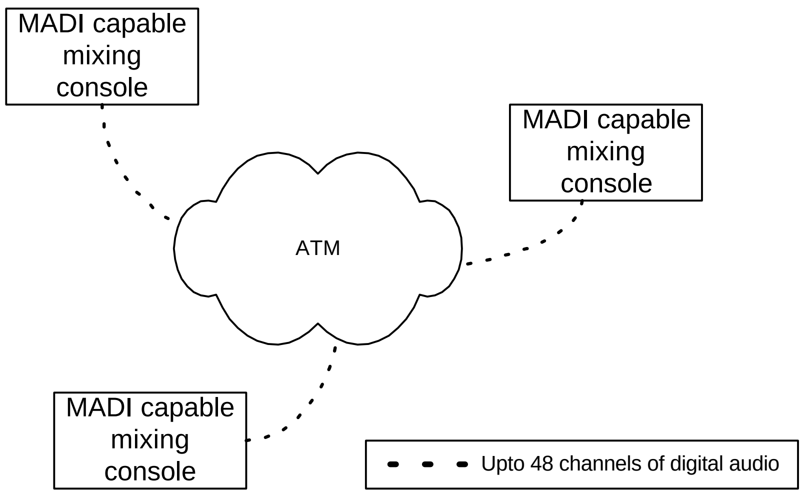

MADI capable mixing console
MADI capable mixing console
ATM
MADI capable mixing console
Upto 48 channels of digital audio

Figure 47: Distributed Studio Network (DSN) architecture.

Lawo’s m^{2}c series of mixing consoles are examples of MADI capable mixers that can utilise this technology. Given the high bandwidth and global scope of ATM technology, the DSN technology enables many applications. The DSN is enabled by custom MADI modules, that interface with the backplane of IBM ATM switches. Figure 48 below illustrates this more detailed architecture.

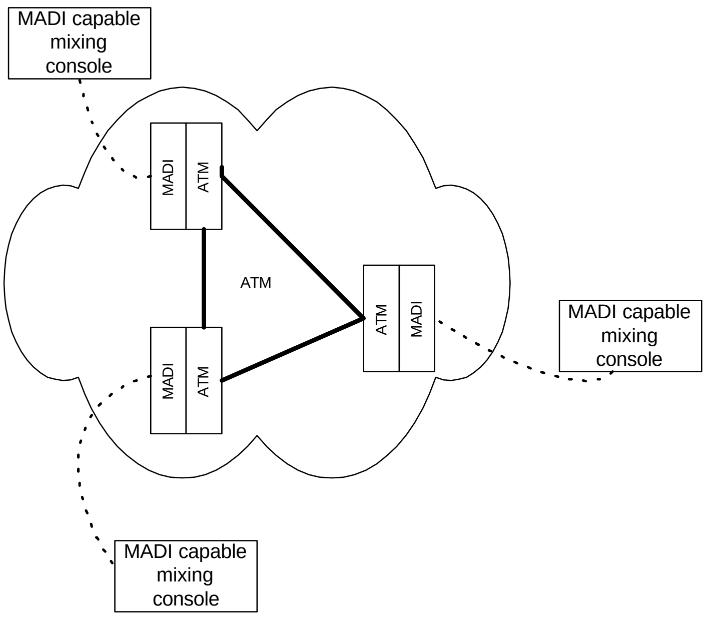

MADI capable mixing console
MADI
ATM
ATM
ATM
MADI
MADI capable mixing console
MADI
ATM
MADI capable mixing console

Figure 48: The MADI-ATM interface in the DSN architecture.

Consider the flow of audio data from one MADI capable mixer to another. A MADI capable mixer transmits an originating audio source in one of the allocated channels. This MADI signal goes into a module connected to the backplane of an IBM ATM switch. This module translates the MADI frames into ATM cells which are then transmitted to the required destination. At the receiver, the MADI module connected to the backplane of the ATM switch, will reconstruct MADI frames from the received ATM cells. Besides the MADI data, sufficient information is transmitted to allow the receiver to reconstruct timing information. The receiving mixer can then process the MADI digital audio stream in the usual way. In this way, DSN provides the mechanisms for digital audio exchange over potentially global-scale distances [Lawo 1998].

6.2 Remote 1394-based studio connectivity

Connecting 1394 based studios was depicted in Yamaha's mLAN press release [Yamaha 1996]. In this press release, a diagram shows a remote booth connected to a main studio. For convenience, this diagram is repeated below in figure 49.

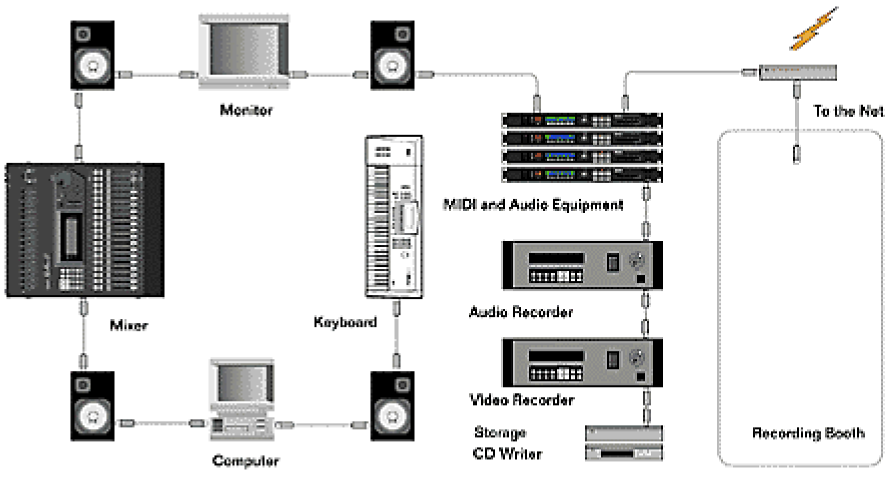


Figure 49: Remote studio connectivity. Source: [Yamaha 1996]

In the same press release, the potential of remote 1394 connectivity is discussed, and depicted in a diagram, repeated below in figure 50 for convenience.

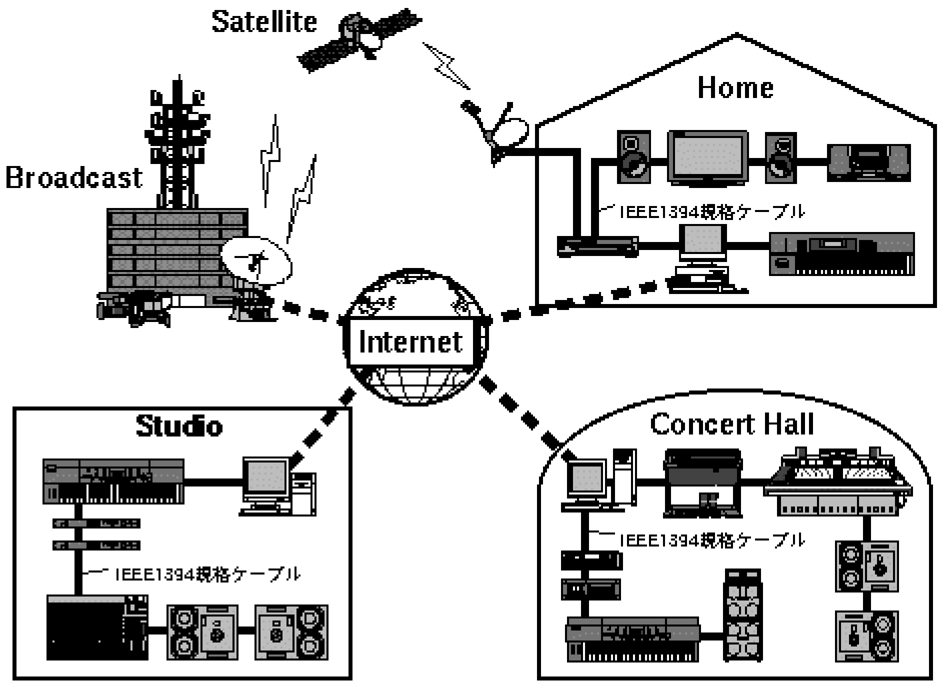


Figure 50: Distributed 1394 networks. Source: [Yamaha 1997]

Protocols for audio and control distribution within a studio, concert hall, or home are being finalised, as has been shown in the previous chapters of this thesis. Connecting remote locations using the Internet may be feasible if enough bandwidth was constantly available.

A more feasible solution however would be to use ATM, as demonstrated by Lawo’s DSN system. The main enabling features of ATM for this purpose are the guarantees of service that can be established. The suitability of ATM for multimedia distribution is discussed by [Fluckiger 1997]. The increasing availability and accessibility of ATM technology further enhances the feasibility. Hoffman describes the 1394 bus as the first and last three feet of the ATM information superhighway [Harcourt 1996].

An emerging standard, P1394.1 Draft Standard for High Performance Serial Bus Bridges[4], could provide the mechanisms for remote 1394 bus connectivity using ATM.

6.2.1 The P1394.1 draft standard.

The P1394.1 draft standard intends to standardise a model for, and the behaviour of a 1394 bus bridge. The bridge device may be used to interconnect two distinct 1394 busses. The bridge is said to consist of two portals, connected by a switching fabric. Each portal is a transaction capable node on the bus that it bridges. Each portal selectively routes asynchronous and isochronous data to the other portal. If a bridge supports isochronous routing, then a common clock distributed to both portals is required. The model is shown in figure 51.

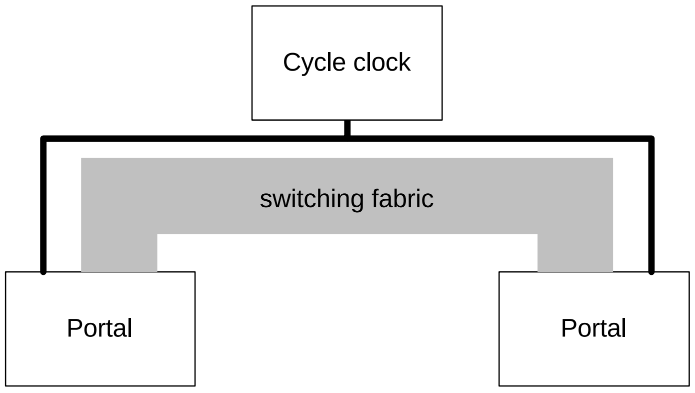


Figure 51: P1394.1 bridging model

The intended scope of the 1394.1 draft standard is for the two portals to be in close proximity, such as in the same box, or in different rooms of the same house. However, the possibility of using long-haul networking technology (such as ATM), as the switching-fabric,

[4] The P1394.1 Draft Standard for High Performance Serial Bus Bridges will be referred to as the P1394.1 draft standard in this thesis.

opens the possibility. Besides increasing the distance that 1394 bus applications can utilise, the 1394.1 draft standard also enables more than 63 nodes to be interconnected. Recall how the 64-bit 1394 addressing scheme (refer back to section 2.2) enables the addressing of a memory location on one of 63 nodes, on one of 1024 busses. The P1394.1 draft specification will provide the mechanisms for using multiple integrated busses.

If a bridge supports isochronous routing, then a common cycle clock distributed to both portals is required. This common clock is required so that synchronisation between the distinct busses is maintained. If the portals share a common time reference, and if the portals were the cycle masters of their busses, then the distributed busses would also share a common time reference. Sharing a common time reference, in the context of audio distribution, is important for audio sample rate recovery and time alignment. Portals compensate for transmission delays over the switching fabric, by adjusting the time stamp information within CIP formatted isochronous transactions. Recall that the time stamp conveys the intended presentation time. If the transmission delay over the switching fabric exceeds the intended presentation time, then portals must increase the value of the time stamp to ensure the presentation time has not already passed.

If a geographically extensive switching fabric were used, distributing a common clock to both portals poses "interesting" implementation problems. In the P1394.1 draft standard, the distribution of the clock signal is described as necessary, but implementation dependant. Distributing common clocks over long distances is however demonstrated by Lawo's DSN system.

Figure 52 shows a possible future configuration for connecting remote studios. A user would play the keyboard, generating AM824 data clusters containing MIDI data. The CIP formatted isochronous transactions would be routed to the other bus, by the portal. The synth could then receive the MIDI data, and would produce a different isochronous stream of AM824 audio clusters. The effects unit could process this data, producing another isochronous stream of the processed audio data. The remote portal could then route these CIP formatted isochronous transactions back to the local portal, and the active-speakers would reproduce the audio.

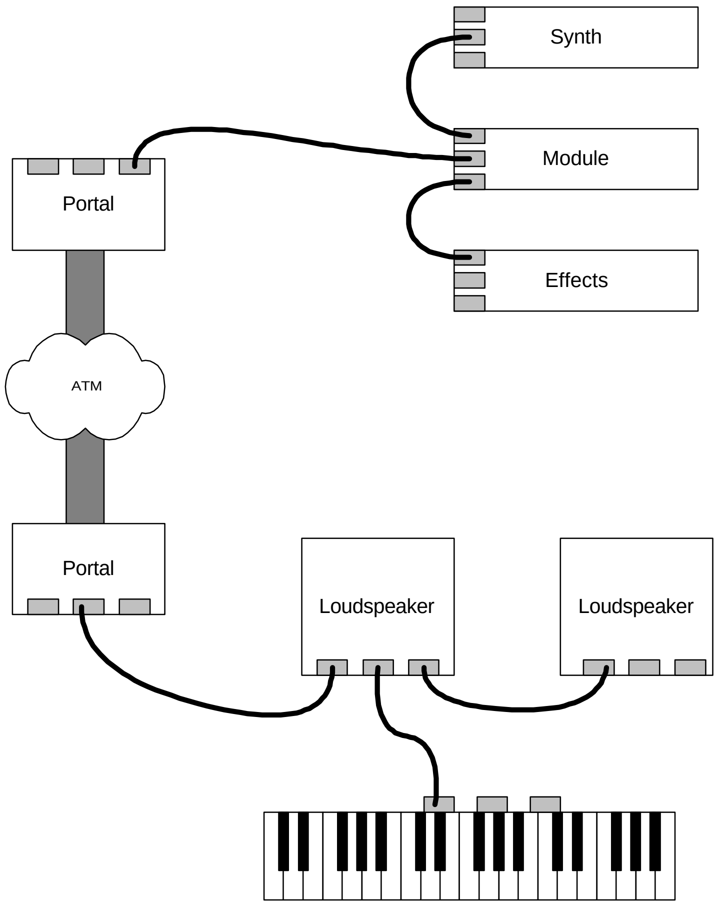


Figure 52: Possible architecture for remote studio connectivity using the P1394.1 draft standard.

The P1394.1 draft standard is still preliminary (published as version 0.03). As such, hardware utilising this standard has not been demonstrated. The P1394.1 draft specification could enable remote studio connectivity that would have the following characteristics:

- Devices at each location are interconnected with the IEEE 1394 single, flexible cable.
- A portal provides the transmission path for isochronous communications between geographically separated devices.

The usefulness of the P1394.1 specification being used for this purpose, relies on a mechanism for clock distribution over geographically extensive portal separation. If ATM is used as the switching fabric between the portals, then the distribution of a common clock is feasible, given that Lawo have already accomplished MADI clock distribution over ATM.

Another emerging technology for providing 1394 connectivity between geographically separated 1394 devices is the IEEE 1394b supplement. Unlike the P1394.1 draft standard, devices utilising the IEEE 1394b supplement have been demonstrated.

6.2.2 The IEEE 1394b supplement

Recall that the IEEE 1394b supplement makes provision for the use of optical fibre. NEC have developed optical fibre transceivers (both plastic- and glass-fibre based) for 1394 devices [NEC 1998]. Using a pair of optical fibre connected transceivers, a single "hop" of up to 500m has been demonstrated. The transceivers integrate into the 1394 bus topology, and extend the geographical extent. Remote studio connectivity using the NEC transceivers can be demonstrated by taking any inter-device connection, and making it 500m long! The constraints on the "extended" bus topology must still be satisfied – a maximum of 63 devices, with no more than 16 hops between any two devices.

Applying this technology to the shared studio system that was discussed at the beginning of this chapter, the result is shown below in figure 53.

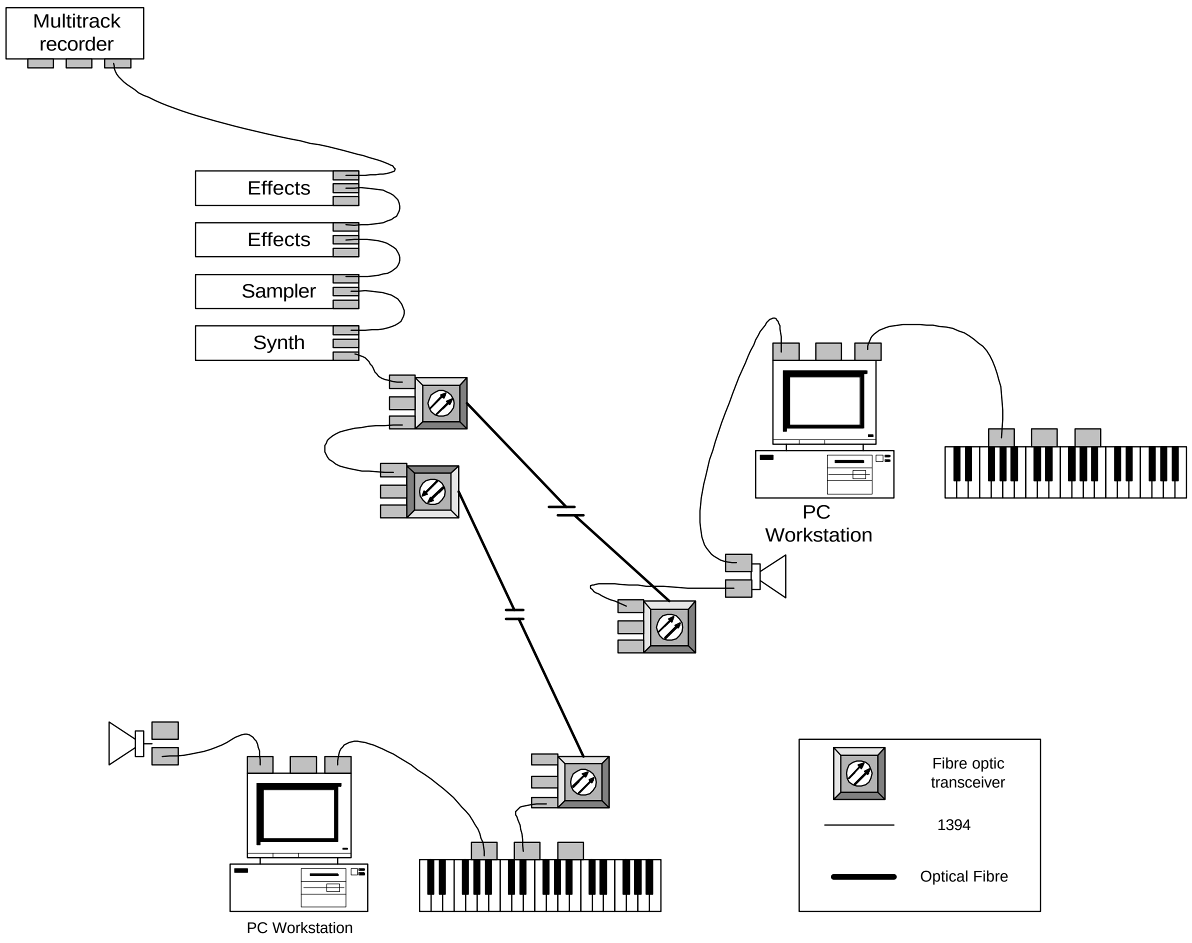


Figure 53: Remote workstation connectivity using 1394 and 1394b transceivers.

It can be seen in figure 53, how the complex cabling of the shared studio system are replaced with a single physical data carrier. Any device can distribute audio and MIDI data to any device on the distributed network. A user at a workstation could play the keyboard, generating MIDI data. The AM824 MIDI events would be transmitted in CIP isochronous transactions to a receiving device, say the synth. The synth would then respond by streaming AM824 audio events in CIP formatted isochronous transactions. This audio stream could be recorded by the user's workstation, or reproduced by the loudspeaker. As mentioned previously, 1394b transceivers have been demonstrated to transmit isochronous data over 500m hops. There are exciting possibilities for sharing studio resources using this technology.

7 Conclusion

Studio users are faced with a complex environment in which to create the music that has become so much a part of our lives. The complexity in music studio environments arises from the many different ways in which devices can be connected. Existing analog and digital connectivity technologies have been discussed, as has the use of MIDI for device control. It has been shown that although MIDI offers tremendous versatility for studio device control, the bandwidth limitation of MIDI can introduce considerable latency in control data distribution.

The IEEE 1394 High Performance Serial Bus is a new interconnectivity technology with the potential to revolutionise studio device interconnectivity. The solution provided by the 1394 bus technology enables the transmission of up to 64 independent channels, each possibly containing multi-channel digital audio and MIDI data between compliant devices. The isochronous capabilities of the 1394 bus guarantee low transmission latencies. The simplicity and usability of the technology derives from the use of simple point-to-point connections, joining all devices in a flexible topology.

Industry has realised the potential of the 1394 bus as a possible solution for digital convergence – a single high performance, digital interconnectivity technology. Many silicon manufacturers are offering hardware solutions, and the some 170 member companies of the 1394 Trade Association have announced their intentions to support the 1394 bus. This support from industry has spurred the development of international standards to guarantee interoperability between devices made by different manufacturers. These standards can broadly be divided into two categories – those defining the formats and flow management of data transmissions, and those defining device control. Data transmission is defined in the IEC 61883 standard, and device control currently by the AV/C series of documents. It is likely that AV/C, once finalised, will be incorporated in the IEC 61883 standard. Consumer products are being developed as the appropriate standards mature. Digital video transmission using the 1394 bus is one application supported by well-developed standards. As such, consumers can already buy digital video devices utilising the 1394 bus technology.

The data format and flow management for audio and MIDI data transmission using the 1394 bus, first proposed by Yamaha in their mLAN specification, is currently pending finalisation as the IEC 61883-6 standard. The current status of this standard is described by the A/M protocol. Devices developed by PAVO, Diablo Research Labs, and the implementation described by this thesis, demonstrate the transmission of A/M protocol formatted data. Doing this requires demonstrating the transmission of clusters of AM824 data events in CIP formatted isochronous transactions. In this implementation, a Motorola 56002 DSP was used to provide AM824 audio capabilities, and the Philips "AV" chipset to provide the 1394 bus capabilities. While the quality of audio transmitted by this implementation was not perceivably altered, there were two problems that arose. The problem of half-duplex transmission was imposed by the Philips "AV" link layer used. It was discussed how full-duplex capabilities are crucial for devices in studio environments. The new Philips "AV" chipset, featuring full-duplex operation, offers exciting possibilities for extensions from this work.

It was shown how the Philips "AV" chipset makes provision for recovering time stamping information from received CIP packets. The most significant flaw in this implementation was the inability to adapt the sampling rate of the receiving codec to match the sampling rate of the incoming digital audio. It was discussed how other implementations do this, and how this is a necessary component of digital audio transmission.

There are a number of issues that need resolution before studio devices utilising this technology are ready for consumer use. These issues can be divided into data flow enhancements, and device control mechanisms. Data flow enhancements include the need for sample accurate synchronisation, and time code distribution. Many devices would benefit from a mechanism, enabling sample accurate synchronisation between all devices. An audio mixer and a PC workstation are examples of such devices. Sample accurate synchronisation would enable a mixer device to efficiently combine audio signals, without the need to perform sample rate recovery using phase locked loop techniques. A mechanism for time code distribution is also required to maintain synchronisation with other media, such as video. Guaranteed interoperability is crucial for the 1394 bus to be a solution for digital connectivity convergence. A user should be able to assume that any devices with a 1394 connector can simply be plugged together. Since the range of audio devices and the functions they perform are so diverse, a common command set for all devices is necessarily complex.

Fortunately, working groups within the 1394 Trade Association are addressing these data flow enhancements and device control issues.

Pending the completion of common device-control command sets and data flow enhancements, the 1394 bus will offer a sophisticated, yet simple alternative connectivity technology for audio and control data distribution in studio environments. With support from many hardware and software vendors, and the continuing standardisation activities, the 1394 bus is a realisable solution for digital convergence in recording studios.

It has been discussed how there is a need for remote studio connectivity, the need arising from the desire to share studio resources and collaborate in project production. The studio sharing system utilising MIDINet units provides a possible solution for studio device sharing, but is burdened with the complexity of many physical data carriers. It was shown how using the already available 1394b transceiver technology, the whole studio sharing system could be connected with a single cable for providing audio and MIDI data distribution.

Lawo's Distributed Studio Network system is a real technology for implementing global-scale inter-studio connectivity. It was described how the Distributed Studio Network enables MADI formatted digital audio to be transmitted via ATM. An emerging specification, P1394.1 which describes the bridging of 1394 busses, open the possibility of having 1394 busses interconnected with ATM. It was explained how the distribution of a clock signal is critical for bridged bus operations. Given that this is achieved by Lawo's Distributed Studio Network system, there is the exciting possibility of extensions to this work to demonstrate how the 1394 bus can be the first and the last three feet of the ATM information superhighway. What better way to demonstrate this, than with a remote studio application?

References

1394 Trade Association "Quarterly Newsletter" 1394 Trade Association. Published online at http://www.13941ta.org/abouta

1394 Trade Association "Audio and Music Data Transmission Protocol ver 1.0" 1394 Trade Association, 1998. Available online at http://www.1394ta.org/abouttech/specifications/techspec.html

1394 Trade Association "AV/C Digital Interface Command Set General Specification ver 3.0" 1394 Trade Association, 1998. Available online at http://www.1394ta.org

Abe T., Fujimori J. "Distributed Connection Management for Electronic Musical Environment Using IEEE 1394" Yamaha Corporation, 1996.

Anderson D. "Firewire System Architecture" Mindshare, Inc., 1998.

Carter A. "Longer cables for the IEEE P1394 High Performance Serial Bus, A White Paper" Apple Computer, Inc. 1994.

Diablo Research Company. "Sound Fire 1394 Digital Audio Engine" Diablo Research Company, 1997. Available online at http://208.240.90.137/pages/sndfire.html

Diablo Research Company. "Philips 1394 Audio System" Diablo Research Company, 1997. Available online at http://208.240.90.137/pages/1394sys.html

Digital Harmony. "Products and Licensing" Digital Harmony Technologies, 1999. Available online at http://www.digitalharmony.com

DTLA "5C Digital Transmission Content Protection White Paper" Digital Transmission Licensing Authority, 1998. Available online at http://www.dtla.org

Dyke T., Smolen P. “Rallying Around the IEEE 1394” Published online at http://eagle.onr.com/aea/media/tvtech34.html

Fluckiger F. “Understanding Distributed Multimedia: Applications and technology” Prentice Hall, 1995.

Foss R., Wilks A. “Using Network Technology to Share Music Studio Resources” Proceedings of the AES 13th International Conference, Dallas, 1994.

Fujimori J., Osakabe Y. “Digital Audio and Performance Data Transmission Protocol over IEEE1394” Yamaha Corporation, Sony Corporation, 1996.

Garvin, M. “Designing a Music Recorder” Dr. Dobb's Journal, May 1987.

Harcourt J. “FireWire - Unravelling the Digital Connectivity Problem” Photo Electronic Imaging, Vol. 38, No. 10, 1995. Published online at http://www.peimag.com/fire.htm

Henehan B. “1394 Firewire Hardware Design Considerations” Texas Instruments, 1988. Available online at http://www.ti.com/sc/docs/msp/papers/index.htm

Hoffman G. “IEEE 1394: A Ubiquitous Bus” ” Published online at http://www.skipstone.com/compcon.html

HoffMan G. “IEEE 1394, the A/V Digital Interface of Choice” Published online at http://www.skipstone.com/newspap.html

Home Network Group “HAVi 1.0 Beta Specification” Home Network Group, 1998. Available online at http://www.havi.org

Hungtington J. “Control Systems for Live Entertainment” Focal Press, 1994.

IEC. “ISO/IEC 13213 (ANSI/IEEE 1212) Control and Status Registers (CSR) Architecture for Microcomputer Busses” International Eletrotechnical Commission, 1994.

IEC "IEC 61883 Digital Interface for Consumer Electronic Audio/Video Equipment" International Eletrotechnical Commission, 1996.

IEEE. "IEEE 1394-1995 Standard for a High Performance Serial Bus" Institute of Electronic and Electrical Engineers, 1995.

IEEE "P1394a Draft Standard for a High Performance Serial Bus (Supplement). Draft 2.0". Available online at ftp://ftp.symbios.com:/pub/standards/io/1394/P1394a

IEEE "P1394b Draft Standard for a High Performance Serial Bus (Supplement). Draft 0.14". Available online at http://www.zayante.com/p1394b

IEEE "P1394.1 Draft Standard for Serial Bus Bridges. Draft 0.03". Available online at http://grouper.ieee.org/groups/1394/1/

IETF "IPv4 over 1394". Available online at ftp://www.ietf.org

IMA. "MIDI Machine Control 1.0" the International MIDI Association, 1992.

IMA. "MIDI 1.0 Detailed Specification, Document Version 4.1" the International MIDI Association, 1989.

Iverson J. "NEC Unveils New Technology for High-Bandwidth Data Transfer" Stereophile, 1998. Published online at http://www.stereophile.com/shownews.cgi?137

Jacobs D., Anderson D. "Design Issues for Digital Audio Networks" Proceedings of the AES 13th International Conference, 1994.

Jansen M. "Firewire" Published online at http://www.engg.ksu.edu/KSE/spring96/firewire/firewire2.html

Johansson P. "Software Design for IEEE 1394 Peripherals" Congruent Software, 1996.

Lawo. "Distributed Studio Network" Lawo, 1998. Available online at http://www.lawo.de

Legault A. “The All-Digital PC-Based Studio Network” Published online at http://www.matrox.com/video/all_digital.htm, 1998

Lehrman P., Tully T. “MIDI for the Professional” Amsco Publications, 1993.

Lidbetter P. “The MADI Format: Applications and Implementation in the Digital Studio” Proceedings of the AES 7th International Conference, 1989.

Light A., Bloks R. “Using the PDI1394L11 ‘AVLink’ Controller” Philips Semiconductors, 1997.

Microsoft. “OS Support for 1394” Microsoft Corporation, 1998. Published online at http://www.microsoft.com/hwdev/presents/winhec98/winhec3-4a/tsld063.htm

Microsoft. “Introduction to Digital Audio” Microsoft Corporation, 1998. Available online at http://www.microsoft.com/hwdev/devdes/digitaudio.htm

Microsoft. “Digital Audio Initiative” Microsoft Corporation, 1998. Available online at http://www.microsoft.com/hwdev/digitalaudio/default.htm

Microsoft. “Kernel Mixer and WDM Architecture” Microsoft Corporation, 1998. Available online at http://www.microsoft.com/hwdev/audio/kmixer.htm

Moore, F. “The Dysfunctions of MIDI” Computer Music Journal, Vol 12, No. 1, Spring 1988.

Moore, D. “IEEE 1394: the Cable Connection to Complete the Digital Revolution”. Published online at http://www.skipstone.com/ss21st.html

Moses B. “Implementing Digital Audio Devices for the IEEE 1394 High Performance Serial Bus.” Audio Engineering Society preprint number 4761, 1998.

Moses B. “Applications of IEEE 1394 in Audio/Video Entertainment Systems, a white paper”. 1996. Available online at http://www.pavo.com/ieee1394/faq/wp1996.htm

Moses B., Bartlett G. “Audio Distribution and Control using the IEEE 1394 Serial Bus”. Presented at the 103rd AES Convention, 1997. Available online at http://www.pavo.com/ieee1394/faq/1997aes.htm

Motorola. “The DSP56002 Evaluation Module” Motorola Semiconductors. Available online at http://www.mot.com/SPS/DSP/products/DSP56002EVM.html

PAVO “Papaya IEEE 1394 Audio Reference” PAVO, Inc., 1998. Available online at http://www.pavo.com/ieee1394/13942loc.htm

Philips Semiconductors “PDI1394LL AV Link Layer Controller Data Sheet” Philips Semiconductors, 1997.
Available online at http://www-us2.semiconductors.philips.com/1394/products/

Philips Semiconductors “PDI1394L21 Full duplex AV Link Layer Controller Data Sheet” Philips Semiconductors, 1998. Available online at http://www-us2.semiconductors.philips.com/1394/

Pohlmann K. “Principles of Digital Audio” Howard W. Sams & Co, 1987

Rumsey F., Watkins J. “The Digital Interface Handbook” Second Edition, Focal Press, 1993

Shelton T. “Synchronisation of Digital Audio” Proceedings of the AES 7th International Conference, 1989.

Steinberg. “Cubase VST” Steinberg, 1998. Available online at http://www.steinberg.net

Teener M. “A Bus on a Diet – The Serial Bus Alternative” Apple Computer, Inc. 1993

Teener M. “Summary of 1394b.” 1999. Published online at http://www.1394ta.org/aboutta/review_1394.b.html

Texas Instruments “MPEG2Lynx link layer controller datasheet” Texas Instruments, 1998. Available online at http://www.ti.com

Texas Instruments “TSB41LV03 datasheet” Texas Instruments, 1998. Available online at http://www.ti.com

Unibrain “IEEE 1394 FireProducts” Unibrain, 1998. Published online at http://www.unibrain.com

Vitana. “Philips PDI1394L11/P11 Reference Design Kit” Vitana Corporation, 1998. Available online at http://www.vitana.com/rdk/philips/1394.html

Vitiliano F. “Why FireWire is Hot! Hot! Hot!” 21st Impact, The VXM Network. Published online at http://www.vxm.com/21R.35.html

Wetzel A., Schell M. “Consumer Applications of the IEEE 1394 Serial Bus, and a 1394/DV Video Editing System”. Texas Instruments, Inc., 1998.

Yamaha Corporation “Multi-channel audio and music data rides on the IEEE 1394 high-performance serial bus” Yamaha Corporation. Available online at http://www.yamaha.co.jp/tech/1394mLAN/Press/970620.html

Zenith, Thompson “New Digital Copy Protection Proposal Would Secure Authorised Copies” Zenith Electronics and Thompson Electronics press release, November 1988. Available online at http://www.zenith.com/main/pr_documents/pr.hdtv.111398.html

Filename: Thesis - Rob Laubscher.doc
Directory: C:\RU Digital Library\RU Theses\1999\Laubscher
Template: C:\WINDOWS\Application
Data\Microsoft\Templates\Normal.dot
Title: Chapter 1
Subject:
Author: Rob Laubscher
Keywords:
Comments:
Creation Date: 04/14/1999 10:34 AM
Change Number: 6
Last Saved On: 04/14/1999 11:22 AM
Last Saved By: Michael Preston
Total Editing Time: 44 Minutes
Last Printed On: 10/13/2000 3:58 PM
As of Last Complete Printing
Number of Pages: 145
Number of Words: 31,568 (approx.)
Number of Characters: 179,942 (approx.)