A Building Block Approach to Network Control

Chris Hanna

THAT Corporation, Marlborough, MA 01752, USA

Adding a network interface to a microprocessor-controlled product significantly complicates the product design. This paper describes a method of minimizing this added complexity by encapsulating the network interface in its own processor and providing a simple, standard internal communications interface between the network processor and the host processor. This approach significantly eases product design and reduces development time.

1 INTRODUCTION

The audio industry is being revolutionized by digital technology. The design of audio equipment has been permanently altered by advances in microprocessors, digital signal processing, and computer networking. The number of manufacturers incorporating this technology into their equipment and the number of users demanding it is growing fast. The application of digital technology has resulted in highly sophisticated products, with increasing levels of programmability, flexibility, and performance.

The obvious benefit of networking audio equipment is the ability to control and monitor a device remotely. However, connecting equipment from various manufacturers onto a network and controlling them individually does not achieve one of the real goals of networking: interoperability. That is, different devices on a network should be able to communicate and interact with each other. This goal has yet to be achieved, but the Audio Engineering Society (AES) is actively promoting this concept through the efforts of its SC-10 Standards Subcommittee. It is up to manufacturers to follow through by implementing this concept

Another benefit of networking is the potential for enhancing the user interface to the system. Networking has, and will continue to have, a profound affect on how humans interact with audio equipment. Networking provides a link that allows manufacturers to leverage new developments in user interface software being developed for the computer industry. The user is the major beneficiary of this application of networking technology to audio equipment.

The convergence of the audio and computer industries has provided new solutions for audio, but has also resulted in new challenges and problems. The long term success of a networked audio product will depend on its ability to be adapted to changes in network technology, changing market forces, and changing customer demands.

The goal of this paper is to encourage manufacturers that are designing network capable equipment to structure their product design so it can adapt to network technology as it evolves in the audio industry.

2 TECHNOLOGIES FOR NETWORKING AUDIO EQUIPMENT

The technologies in use today for networking audio equipment vary widely. Most manufacturers of networked audio equipment have developed their own proprietary networks to suit their applications. The proliferation of these networks is a major concern of a marketplace which strongly desires interoperability. Few manufacturers have made their network technology generally available. But, as the market has developed, more manufacturers are opening their systems to all comers. Still others are developing non-proprietary networks based on open systems.

3 ISSUES IN ADDING NETWORK CONTROL

There are many issues to consider when adding network capability. First, a manufacturer must assess the value of adding a network interface to a product. For some products, there will be no choice since the network is an integral part of the operation of the system. For others, only some customers may demand network operation; they may

be willing to pay additional costs to get it. When only some customers are demanding network capability, the cost of the network interface should not unduly burden the cost to customers who do not need this capability. for many products, the cost to add network control will be sufficient to require that the network interface be an *option*, not a standard part of the product.

3.1 Network Independence

Second, a manufacturer must consider the cost-benefit ratio of network independence. This relates to how products can meet varying customer demands and be adapted to changing technology. It is not always possible to know which network the customer may require now or in the future. Yet, products must accommodate current *and* future needs. For example, if a product is designed to accommodate different network interfaces via a well defined slot in its rear panel, the final choice of network interface could be left up to the customer, who will make a choice based on the application at hand.

In the short term, the need for network independence is also driven by the proliferation of many different networks, each promoted as the *only* network which will *really* work for audio systems. Until an industry wide standard is adopted, manufacturers are well-advised to provide network interface options. Over the long term, advances in network technology will continue to be made. To accommodate this, the product design should be modular to facilitate plug and play for the short term, and intelligently designed for the long term, allowing different network interfaces to be plugged in without redesigning the product. This is what is meant by a *building block* approach.

3.2 Time to Market

Third, development time, product cost, and time to market are all interrelated concerns. By adopting a building block approach, the network interface design can be partitioned into a separately-developed, reusable piece of hardware and firmware. This significantly reduces development time and cost for future products, and can help reduce time to market. In more and mores cases, manufacturers are trading off higher product costs for shorter time to market. Central to this tradeoff is the application of available advanced technology which increases product cost but reduces design time. Note that customers are often willing to accept the increased product cost in return for more flexibility and performance, as well as protection against premature obsolescence.

4 ENCAPSULATING THE NETWORK INTERFACE

The essence of the building block approach to network control is to encapsulate the network interface into a separate physical module. This requires partitioning the product to physically separate the network functionality from the intrinsic product functionality, and provide a standard, well-defined interface between the two sections at the hardware and software levels. Intrinsic functions are controlled by a host microprocessor, while the network interface is a separate physical module with its own processor, providing the intelligence to handle low level communications with the network. This achieves the goal of network independence since the network module may be optional, and other modules for different network interfaces can be designed, provided they fit physically, electrically and in the software interface.

In some applications, this partitioning may be necessary because a single inexpensive microprocessor may not have the resources to handle both network and internal product functions. By separating these functions, the complexity of the hardware and software design is significantly reduced. In addition, the host and network code development can follow parallel paths, which reduces development time. In terms of product cost, this approach potentially increases cost, however the cost increase is shifted towards users who require the network interface, the class most likely to be willing to pay for it.

Aside from the difficulties of designing hardware and software to serve the product and the network themselves, the only added design complexity is that of defining the hardware and software interface between the host and network processors. It should be noted that this approach works extremely well for retrofitting an existing microprocessor-controlled product to include network control. In this case, minimal redesign will be required, allowing the investment in existing code and hardware to be maintained.

4.1 Hardware Interface

The choice of hardware interface depends on the data rate the host processor requires to move data to and from the network. The interface chosen should minimize complexity and cost, while fulfilling the required data rate. Ideally, it would be based on some standard. For many applications, a simple asynchronous serial interface is sufficient. A serial interface has several advantages: it is a relatively well-understood standard, it is already incorporated in almost all embedded microprocessors, and it can be implemented at very low cost. It's disadvantage is that it is limited in data rate.

For more demanding applications, a parallel interface may be required. Standard parallel interfaces could be used,

such as SCSI or IEEE488, or non-standard interfaces based on registers, FIFOs, dual port memory, etc. The parallel interface has the advantage of higher throughput, but the additional hardware required usually results in higher implementation costs. Not surprisingly, the appropriate choice will depend on the tradeoff between cost and performance.

4.2 Software Interface

Defining the software interface between the host and network processors amounts to defining a communications protocol. The protocol should be independent of the network interface itself, completely decoupling the host processor from the type of network interface. This protocol can and should be defined at the application layer, since only data needs to be exchanged.

One way to define this protocol is to use an object-based approach. The AES SC-10 subcommittee has taken this approach in an ongoing effort to define AES-24, an object-based messaging protocol for control and monitoring. On the host side, the controllable and observable elements, referred to as objects, should be thought of as a database. This 'object database' can be accessed by a simple set of messages. These messages essentially define the protocol between the host processor and network processor.

The combination of the hardware and software interface methods described above provides a foundation for developing a network-capable device. The network module itself is effectively a bridge between the network and the intrinsic device functions. This approach supports network independence, since different network interface modules can be designed: each communicates with the host processor using a common protocol. This allows a product to adapt without being redesigned as new requirements or network technology emerges.

5 A DESIGN EXAMPLE

This section describes an example design of a networked product in the context of the issues and methods discussed above. This section emphasizes the hardware and software interface between the host processor and the network module, and the design decisions that were made during the project. In this example, MediaLink[1] was chosen as the network, based on the customer's requirement.

5.1 The Network Technology

MediaLink is a proprietary network technology offered for license by Lone Wolf Corporation. An IC dubbed the ML125K implements the network protocol, and, together with support circuitry, forms the basic hardware required to implement a MediaLink network device. Firmware development tools are available to define the network accessible control and monitor elements for the device.

Early on in this example product development, the fundamental decision was made to partition the design into a host processor and a separate network module with its own processor. This partitioning was chosen largely so that the firmware development of the network module could proceed in parallel with firmware for the base product. This proved to be fortuitous, since the base product firmware eventually ended up taking most of the available host microprocessor resources, which would have left little for the substantial demands of the network processor.

5.2 The Audio Device Itself

This design example is drawn from an audio product that was developed during 1994. The product is a single channel, four-way active crossover and signal processor that uses digitally-controlled analog signal processing under network control. A rear panel slot accepts an optional network module. Both MediaLink and RS-232 versions of the network module have been developed.

The signal processing in the product includes digitally controlled analog limiters on each output. Gain, rms threshold and peak threshold of the limiters is set via the network interface. The network also allows monitoring of output levels, gain reduction levels, and clipping status.

5.3 Serial vs. Parallel Interface

To choose the required hardware interface, the data rate for controlling and monitoring the device was determined. Output and gain reduction levels each require one byte. To make remote displays appear smooth, level data is reported to the network every 50 ms (20 updates/s). All the clip indicators require another byte at the same report rate. Each of the four gain, rms threshold, and peak threshold parameters requires one byte. For each of these control parameters, 10 updates per second is allowed, for a total of 120 updates per second. (Note: if only one control parameter is being changed, it can be updated up to 120 times per second.) Finally, messages between the host processor and network processor require three bytes of overhead per message.

---

[1] MediaLink is a trademark of Lone Wolf Corporation.

Adding all this together results in an aggregate data rate for control and monitoring of this device of 840 bytes per second. This equates to 8400 bits per second for a standard asynchronous serial link. Based on this rate, a full-duplex asynchronous serial interface running at 9600 bits per second was chosen. This provided some extra bandwidth for the communications overhead.

5.4 Partitioning the Product

Figure 5-1 illustrates the partitioning of the product. The audio device itself has its own host CPU, with built-in UART. The MediaLink network module carries its own CPU (ML125K) which takes care of all communication with the network. The RS-232 version of the "network" module does not use a CPU (it merely provides hardware translation from the RS-232 to the internal serial link). The freedom of this approach allows the product designer to select the host processor based exclusively on the audio product's needs. This allows a designer to work with familiar hardware and software tools, and protects an existing investment in hardware and software when redesigning a device to add network control.

6 HARDWARE INTERFACE

The hardware interface between the host processor and the network module is based on the serial link described above. In the audio device itself, the host processor controls the gain, rms threshold, and peak threshold for each of the four channels. It also monitors the output levels, clip indicators, and computes the gain reduction levels for each of the four channels. A host processor with an on-chip UART was chosen to reduce cost and complexity.

6.1 Network Modules

The MediaLink network module consists of the ML125K chip and support logic, fiber optic transceivers, and a separate UART for communication with the host processor. UART driver firmware was developed to allow full-duplex communication at up to 31,250 bits per second. The network module was designed as a general purpose, standalone device, so that it can be reused for other products. It occupies approximately a 3" by 5" PC board which slides into a slot on the rear panel of the audio device.

The RS-232 version of the "network" module does not support a network as such. It's purpose is to allow a user who does not have access to a MediaLink network to program various parameters within the audio device. It consists of relatively simple level-translator ICs, and is designed to interface to one (and only one) external computer at a time. For compatibility, it's form factor is identical to that of the true network module.

7 SOFTWARE INTERFACE

The software interface between the host and network processors is based on a simple object-based messaging protocol. The controllable and observable elements in the device are organized into an object database. All communication with an object is done through messages.

7.1 Set and Get

In this simple protocol, there are two types of actions that can be performed on an object, "set" and "get." The set message is sent to an object to command it to assume a new state. The message contains new data for the object. When the set message is received, the object extracts the data and performs those functions associated with changing its state.

A get message is sent to an object to command it to return its current state. The data is sent back in a message from

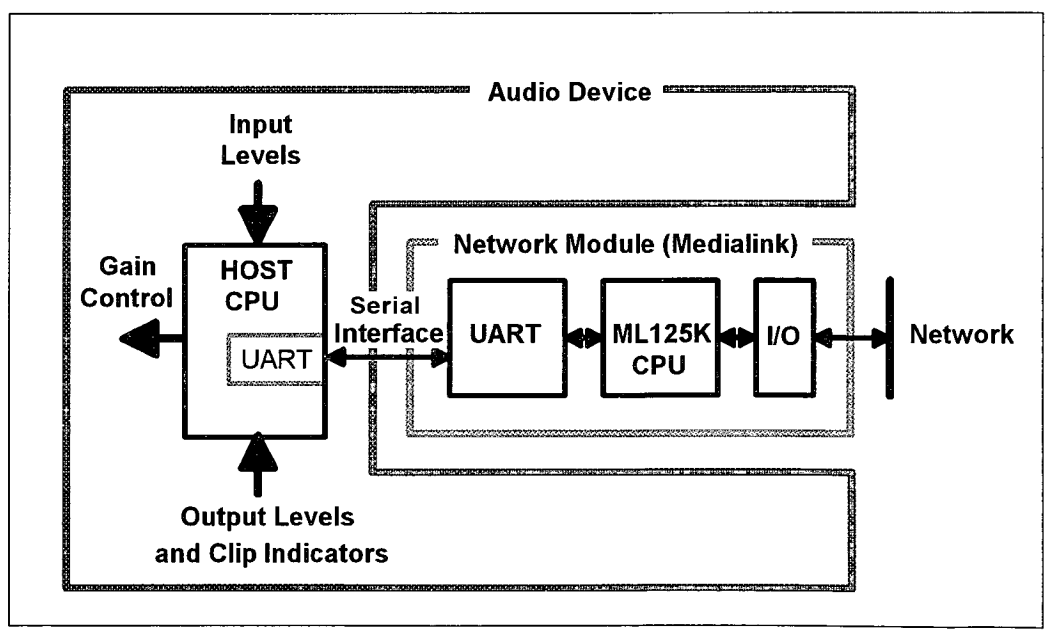

Figure 5-1. Block Diagram of the Example Product

the object. A get message normally does not contain any data.

7.2 Message Format

The message format simply consists of a header containing the object ID and type of action, followed by the data, and ending with an end-of-message delimiter. Each object is assigned an unique "object ID." An object may be only a single element, such as "CH1 gain" or a collection of objects, such as "CH1-4 output levels." The object database for this audio device is shown in Table 7-1.

Object ID	Description
1	Gain (Channel 1)
2	Gain (Channel 2)
3	Gain (Channel 3)
4	Gain (Channel 4)
5	RMS Threshold (Channel 1)
6	RMS Threshold (Channel 2)
7	RMS Threshold (Channel 3)
8	RMS Threshold (Channel 4)
9	Peak Threshold (Channel 1)
10	Peak Threshold (Channel 2)
11	Peak Threshold (Channel 3)
12	Peak Threshold (Channel 4)
13	Output Levels (CH1-4)
14	Gain Reduction Levels (CH1-4)
15	Clip Indicators (CH1-4)

Table 7-1. Object Database for the Example Product

This protocol is very loosely based on Draft AES-24, and has been oversimplified for this discussion. In this design, the development team used Draft AES-24 for both inspiration and reference. Ironically, one of the most valuable points of inspiration taken was to think about devices in terms of the object-based paradigm. As it turned out, this approach also mapped readily for use with the MediaLink device development tools.

For the RS-232 version of the "network" module, the computer at the far end of the RS-232 link is responsible for framing messages into the defined message format. Essentially, the computer to which the audio device communicates via RS-232 takes the place of the CPU which is present on the MediaLink network module.

8 CONCLUSIONS

This paper has discussed some issues involved when network control is added to a product. Based on the real-world design experience described herein, a building block approach is strongly recommended. The key concepts are 1) to *physically separate* the network functionality from the intrinsic product functionality, and 2) to provide a *standard interface* between these two sections at both the hardware and software levels. In the design example shown, the intrinsic functions of the audio device are entirely controlled by a host microprocessor. The network interface is a separate physical module with its own processor that provides the intelligence to handle low level communications with the network.

This approach results in significant benefits to the developer. First, it greatly simplifies product design, because the network development can proceed independent of the main product's development. Second, this, in turn, reduces development time *and* time to market. Third, once the network module hardware and host protocol software is developed it may be reused for other products. Fourth, by separating the network module from the main product, the product itself is insulated from changes in network technology. This makes the product more versatile and tends to lengthen it's life. Finally, once the network module is designed, existing products can be retrofitted quickly without completely redesigning hardware or rewriting code.

9 ACKNOWLEDGEMENT

The author wishes to thank Les Tyler for reviewing this paper and providing many useful suggetions.

10 REFERENCES

[1] M. Lacas, D. Warman, and R. Moses, "The MediaLink Real-Time Multimedia Network," presented at the 95th convention of the Audio Engineering Society, (1993 Oct.) preprint 3736.

[2] R. Moses, "The Object Network Environment," presented at the 94th convention of the Audio Engineering Society, (1993 Mar.) preprint 3561.

[3] C. Rosenberg and R. Moses, "Future Human Interfaces to Computer Controlled Sound Systems," presented at the 95th convention of the Audio Engineering Society, (1993 Oct.) preprint 3735.