This paper first appeared in the Proceedings of the 1992 Rochester Forth Conference.

The Echelon Lighting Control System

Brad Rodriguez

T-Recursive Technology

A. ABSTRACT

The Teatronics Echelon is a reconfigurable control system for stage, studio, and performance lighting. Capable of controlling up to 4000 dimmers on 1000 control channels, Echelon is the third most powerful lighting control console currently manufactured.

The Echelon system uses three to ten Zilog Super8 single-chip CPUs in a parallel processing configuration. Zilog Super8s are also used for up to 30 distributed control stations on a multidrop serial link.

Forth was chosen as the major programming language for Echelon because of its expressive power and its ability to work at high levels of abstraction. The operating program includes over 16,000 lines of Forth code.

B. INTRODUCTION

In early 1990, Teatronics, Inc., the fourth largest U.S. manufacturer of theatrical dimming equipment, decided to develop a family of "memory" lighting control consoles for theatre and television use. Such consoles must capture and reproduce all of the lighting nuances of a complex stage production, using control methods familiar to the theatre industry. T-Recursive Technology was contracted to specify, design, and in large part implement this console. It was named "Echelon." [The product was later renamed the LD series due to a trademark conflict.]

C. DESIGN OBJECTIVES

1. Control Requirements of Theatre Lighting

Theatre lighting is an open-loop control problem. Its outputs must be updated faster than the human visual response: typically 20 to 30 times per second.

A memory console may have from fifty to a few thousands of control variables ("channels"). These channels can be digitally routed to thousands of electronic dimmers. Installations that "patch" a thousand dimmers to only a few hundred control channels are common.

A human operator cannot grasp a thousand control channels, so memory consoles offer "grouping" functions. One example is the "cue fader," which allows a mix of channel values ("levels") to be prerecorded, and then gradually "faded onto the stage" at the press of a button. Another example is the "submaster," which allows a recorded group of channels to be mixed onto the stage with potentiometers.

2. Echelon Specifications

The design objectives of the Echelon system are: a) configurable for 120 to 1000 control channels, and for 512 to 4096 dimmers; b) memory for up to 400 "looks" (control states); and c) available in two versions, having either 24 submasters, or 96 submasters with reduced channel capacity.

The console must output industry-standard AMX-192 (analog multiplex) or DMX-512 (250 kilobaud, RS-485 serial) control signals. It must have interfaces for RS-232 and RS-485 serial data, SMPTE ES-BUS television control signals, MIDI, and DMX-512 input, plus digital actuators and analog control signals. Echelon must use IBM PC compatible disks, printers, and video monitors (CGA or EGA).

3. The Processing Power Requirement

A typical thousand-channel console may have to perform 30,000 8-bit by 8-bit products, and 30,000 MAX operations, every 30 milliseconds. Some consoles employ a single high-performance CPU; 80286s and even 80486s are common. Others add a coprocessor -- such as the 8X305 or TMS320 -- dedicated to computation.

Either way, an ultimate limit on system size remains. Systems with the power for large applications are overpowered and overpriced for smaller applications. And new advances in microprocessor technology offer little benefit to theatre controllers, which need fast 8-bit fixed-point arithmetic, not 32 or 64 bit operations.

Indeed, smaller systems using 8-bit CPUs have set many of the standards for performance lighting control. For example, Teatronics' 48-channel Producer II+ can achieve update rates over 40 Hz.

4. Extensible Multiprocessing

By running small consoles in parallel, we can achieve high performance and large capacity. This is what Echelon does. It contains several independent processors, each performing the essential functions of a small lighting console: calculation and output. The operator interface functions, common to all these "little consoles," are performed by other processors.

A small Echelon system is assembled from only a few processors. A large system is assembled from many. This allows one console design to handle big applications, while remaining inexpensive for small jobs.

5. Why the Super8?

The Zilog Super8 is well suited to lighting control. Its 8-bit unsigned arithmetic is very fast: an 8x8 multiply -- the most frequent operation -- takes only 2.4 usec. Its on-chip UART supports the 250 kilobaud lighting industry standard (DMX-512). Also, the on-chip DMA and "fast interrupt" capabilities have been remarkably useful.

But a more compelling reason is that Teatronics is already a heavy user of Super8s. Currently, they are used in six products besides Echelon. This simplifies inventory and eliminates the need to retrain the technical staff.

D. HARDWARE ARCHITECTURE (FIGURE 1)

1. Division of Processing Functions

Four tasks consume most of the CPU time: input scanning, channel calculation, dimmer patching and output, and video display updates. So, dedicated CPUs perform these tasks in Echelon.

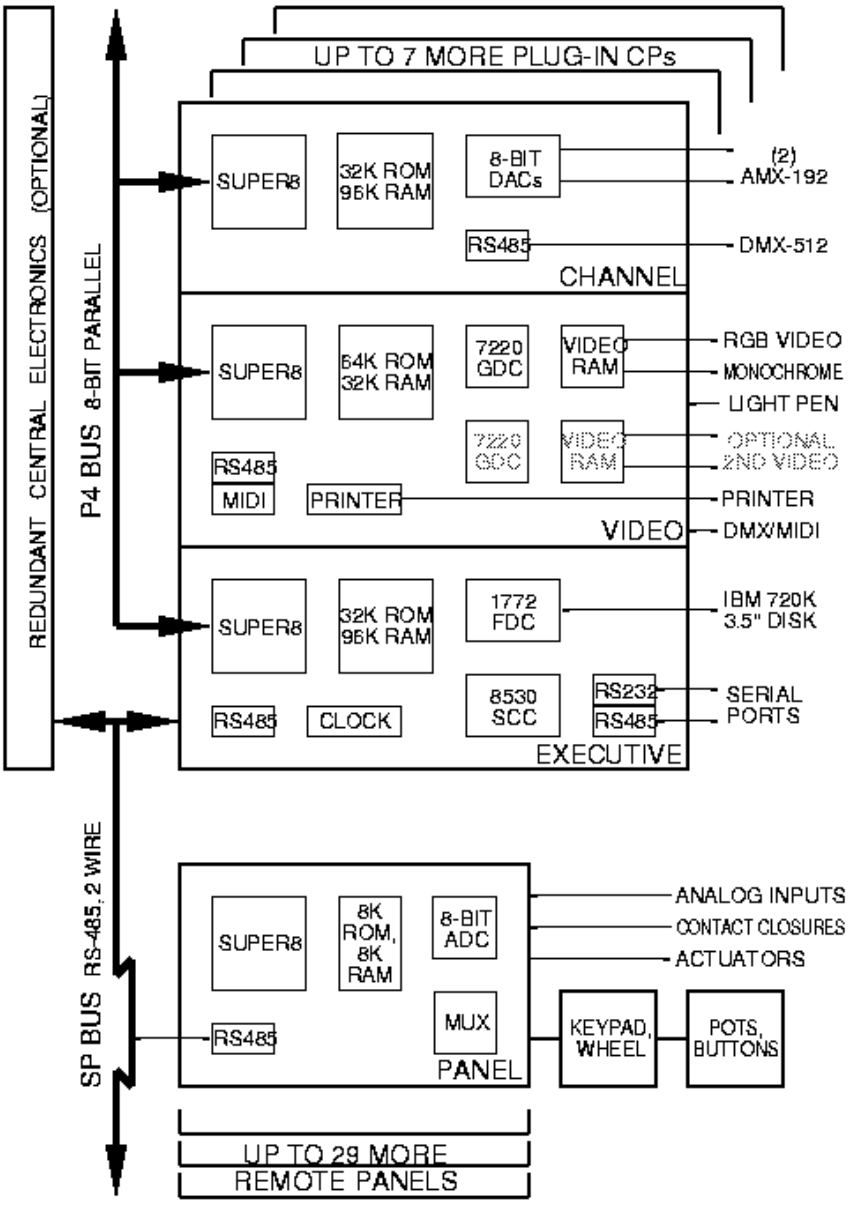
Three distinct processor types -- involving up to ten CPUs -- exist in the central electronics unit. These are the Executive, Video, and Channel processors, connected by a parallel bus.

Control stations can employ up to thirty more CPUs of a fourth type, the Panel Processor. These are connected to the central unit by a serial bus (local area network).

The central electronics unit can be duplicated, sharing the serial communications link, for redundant backup. Control surfaces can also be duplicated on the serial link for redundancy, if required.

All processors use CMOS RAM with battery backup, and have a watchdog timer.

FIGURE 1. HARDWARE ARCHITECTURE



2. Channel Processor

The Channel Processor (CP) is the "little console" which can be replicated in parallel for increased capacity. It contains the essential core of a small memory console -- processor, memory, and output interfaces -- reduced to the simplest hardware and lowest cost.

The CP performs the two most time-consuming tasks: calculation of channel levels, and output to dimmers (including proportional patching). From one to eight CPs can be installed.

3. Video Processor

The Video Processor (VP) is responsible for updating the two video display monitors -- another CPU-intensive task. Two NEC 7220A Graphic Display Controllers generate text or graphic displays on EGA, CGA, or standard monochrome monitors.

The VP's UART is usable either as a DMX-512 (250 kilobaud) input port, or as an optoisolated MIDI interface. The VP also contains the parallel printer interface and a light pen interface.

4. Executive Processor

The Executive Processor (XP) is the mastermind of the system. It receives and processes operator input, and converts it to commands that are issued to the other processors. The XP also controls timed actions (such as timed fades).

The processors of the central electronics unit (CPs, VP, and XP) communicate via an 8-bit parallel bus, known as the "P4 bus." The Executive supervises P4 bus transfers.

The Panel Processors communicate to the Executive via a multidrop serial link, called the Serial Panel bus or "SP bus." The Executive is the SP bus master.

Many common functions and interfaces are included in the XP: disk interface, auxiliary serial interfaces, and clock/calendar.

5. Panel Processor

Scanning the array of keys, switches, and slidepots is another time-consuming task, performed by the Panel Processor (PP). Dedicating a processor to this function allows the control surfaces to be remotely located; control changes are sent via serial link to the central unit.

Most Echelon consoles have one Panel Processor mounted in the central unit. Up to thirty Panel Processors may be installed, sharing a single multidrop cable.

E. SOFTWARE ARCHITECTURE (FIGURE 2)

1. Why Multitask?

At some level of complexity, multitasking makes programming easier, especially when the model of the problem contains simultaneous activities.

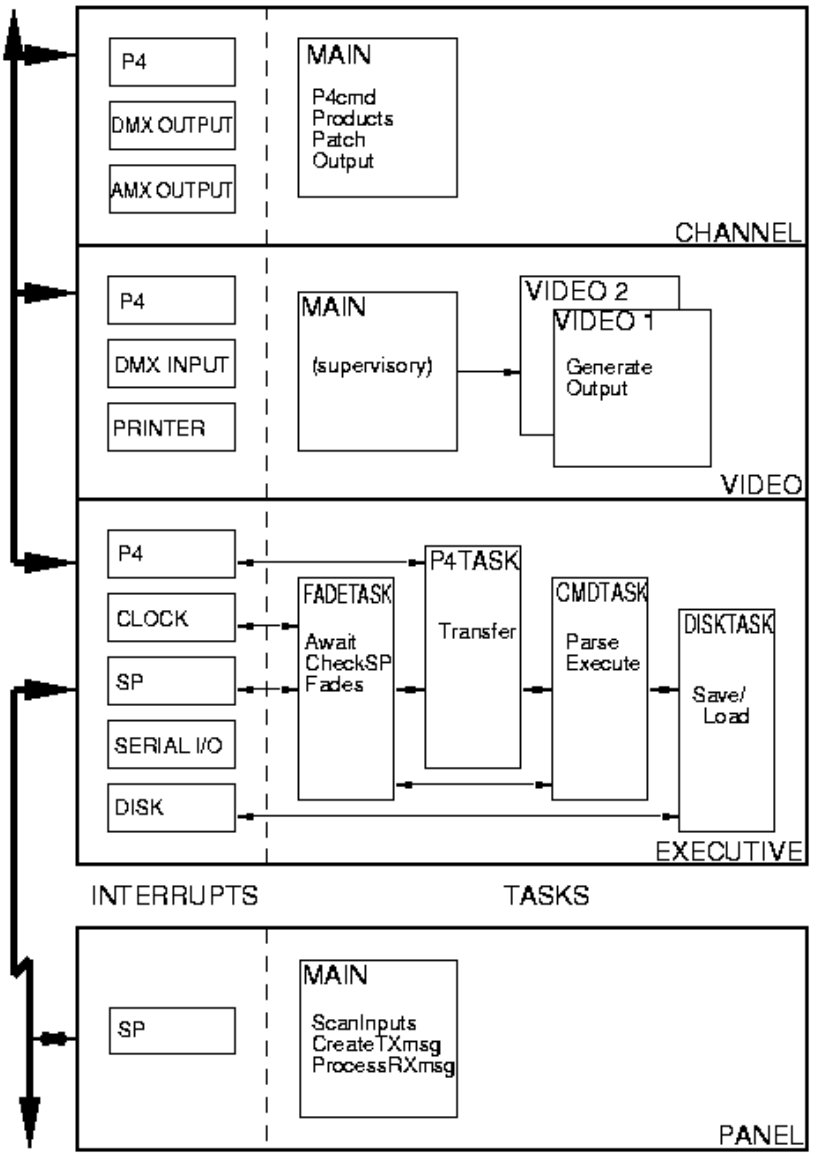
For example, the Executive Processor must simultaneously: control P4 bus transfers, update fade progress at regular intervals, interpret operator commands, and transfer data to disk. These four tasks appear continuous, but they spend much of their time waiting, i.e., idle. Multitasking makes optimum use of this idle time.

Our mental model is often one of waiting for an event. Fade processing may wait for a clock tick, interprocessor communications for a DMA transfer to end, command processing for a keypress, and disk transfer for the end of a sector. Multitasking allows programs to be written in these terms.

Some resources must be protected from simultaneous access. Semaphores make this protection automatic, and invisible to the programmer. Semaphores also allow synchronization of tasks -- in Echelon, tasks on different processors.

In general, multitasking makes the programming model closer to our mental model of the problem, and hides the implementation details.

FIGURE 2. SOFTWARE ARCHITECTURE



2. Channel Processor

The Channel Processor runs a single task program, performing a simple loop: calculate channel levels, softpatch channels to dimmers, and output dimmers. Any commands received from the Executive are processed at the beginning of the loop.

P4 bus "slave" communications are handled entirely at interrupt level. Dimmer output is also done at interrupt level, using the Super8's fast interrupt.

3. Executive Processor

The Executive Processor uses multitasking. One task is awakened every 30 msec to perform timed functions, such as fades and special effects. This task also processes control inputs from the panels.

A second task is the P4 bus master, controlling all transfers on the P4 bus. It updates data in all the processors.

A third task is the command interpreter, which sleeps until keyboard input is received. This task is meant to be replicated, for multiple users.

A fourth task is responsible for disk transfers and printing. These functions are initiated by operator command, but can proceed during other operations.

SP bus (panel) communication is handled entirely at interrupt level. P4 bus transfers and the system clock also use interrupts. Disk transfers use the fast interrupt.

4. Video Processor

The Video Processor also uses multitasking. One task performs a "supervisory" function, and processes commands received from the Executive.

Two more tasks, both running identical code, manage and update the two video displays.

P4 bus "slave" communications and printer output are handled at interrupt level. The fast interrupt receives 250 kilobaud (DMX-512) serial data.

5. Panel Processor

The Panel Processor is a very simple single task program that scans all analog and digital inputs, and formats input changes for transmission to the Executive. It also processes commands from the Executive (e.g., LED on/off).

SP bus "slave" communications are handled entirely at interrupt level.

F. NOVEL FEATURES

1. Multitasker

Zilog's Super8 and Z8 processors support multiple sets of "working registers" in the on-chip register file. Each Forth task has its own set of working registers. One machine instruction can switch most of the Forth context.

Previous work [ROD89] suggests that the figure of merit for round-robin multitaskers should not be the time to switch tasks, but the time to poll an idle task. Accordingly, the Echelon multitasker's primitive is AWAIT, which polls a semaphore flag. For maximum speed, semaphores are kept in the Super8 register file. An idle task can be polled in 4.2 usec; PAUSE (-1 AWAIT) takes 13.0 usec.

2. Multiprocessor Communications (P4 bus)

The processors of the central unit (CPs, VP, and XP) are connected via port 4 to an 8-bit wide parallel bus, accordingly known as the "P4 bus." This bus uses the Super8's DMA controller to perform direct memory-to-memory transfers between processors, either point-to-point or broadcast. The peak transfer rate is approximately 6 usec per byte, and it is transparent to the processors. Only the initiation and termination of a block transfer require processor intervention.

At present, the Executive Processor controls all transfers. It initiates a transfer by broadcasting a "cycle information block," which specifies a sending CPU, one or more receiving CPUs, an address selector, and a length (2 to 65536 bytes).

The address selector indexes into a table holding the base addresses of data structures; no CPU need know the physical address of another CPU's data. An "offset" can be selectively added to the base address, to allow arrays distributed across the CPs to be transferred to a single processor (XP or VP).

The broadcast capability is especially important for dimmer patching, since each CP must "know" the channel levels calculated by all the other CPs.

3. Local Area Network Communications (SP bus)

Up to thirty Panel Processors can be connected via a single twisted pair cable to the Executive Processor. This multidrop Serial Panel Bus, or "SP bus," carries half-duplex RS-485 asynchronous serial data. The Super8 UART's 9th-bit wake-up mode is used on this bus.

The design of the SP bus protocol takes two tips from ClusterFORTH [RAT85]. The first is the single master / multiple slave approach, for deterministic speed, faster polling, and simpler software. The second is the concept of message "phase" for zero-overhead message acknowledgment.

The lowest possible baud rate is desirable for longer cable runs on the SP bus. This requires short messages: the poll message is 2 bytes, the null reply is 2 bytes, and no message exceeds 19 bytes. Using the Super8's 9-bit (address mark) data format, and the standard rate of 38.4 kilobaud, 29 panels can be polled at the required frequency of 30 times per second.

Panel data is tightly encoded for faster transmission. The encoding scheme is "open-ended" for future expansion.

The SP bus cable also carries power, and special supervisory signals to control backup systems.

4. Optimized channel calculation

Most lighting controllers employ brute force to calculate the arrays of channel data. Echelon employs several techniques to accelerate this calculation. The exact details are proprietary; however, the common principle is to perform only those calculations that are necessary. Lighting data is usually sparse and static; speedups of 300% or more are not uncommon.

G. PROJECT MANAGEMENT

1. Why was Forth used?

The complexity of the project overwhelms assembly language programming. Forth and C are the only HLLs available for the Super8. Forth was chosen for programmer productivity and ease of learning, as well as personal preference.

Forth allows arbitrarily high levels of abstraction. This is particularly important in the multiprocessor system. As described above, multitasking Forth hides the details of synchronization and communication between processors. High level system logic is written more clearly and quickly using application-specific concepts.

Interpretive Forth also simplifies cross-debugging. The programmer can compile once, and then test several routines -- even subroutines -- with a variety of inputs, and directly examine the results in memory. This reduces the need for recompilation and "instrumentation" of the code, and eliminates the need for logic analyzers and in-circuit emulators.

Finally, Forth is easy to teach to an audience of assembly language engineers: in one case, it took only six hours.

2. Where was Forth used?

Only the Video and Executive Processors are programmed in Forth. The Channel and Panel Processors are programmed in assembly language, for several reasons.

The Channel Processor program is quite simple, but requires the utmost in speed -- so most of it would be coded in assembler anyway. Also, much of the CP code was written (for proof of concept) before Forth was available.

The Panel Processor is the simplest program, and much of it is adapted from existing assembler code.

Most significantly, both the CP and PP were delegated to a programmer with little Forth experience, who believed he would make fewer errors in assembler.

The VP and XP programs are vastly more complex, and require the full "armament" of multitasking, semaphores, and high abstraction.

3. The Forth cross-compiler

At first, no suitable Forth cross-compiler was available for the Super8. The compiler must be suited to embedded programming; it must also be well documented, and usable by newcomers to Forth. Moreover, the Forth interpreter layer would be used in the final application, as a script interpreter, a disk file translator, and to provide maintenance access, but no royalties could be paid for this interpreter.

Finally, at the 1990 Rochester Forth conference, MicroProcessor Engineering's MPE-Forth cross-compiler was chosen. This compiler has met or exceeded expectations.

4. The Development Environment

The multiprocessor nature of Echelon often requires code for three CPUs to be developed and tested simultaneously. Burning EPROMs is too slow; three in-circuit emulators are expensive; and most of the Echelon processors lack the memory for resident Forth development. Instead, EPROM emulators by LeBurg Electronics (emulating two EPROMs) and Parallax are used.

The development cycle is the edit-compile-download- test familiar to assembly language programmers. The fast compile (typically 30 seconds) and EPROM emulators make the cycle reasonably quick. MPE's Umbilical Forth was not available for the Super8 at the inception of the project, but may be used in the future.

Only the early versions of the XP software support a resident Forth compiler. The VP, having no writeable code RAM, supports a Forth interpreter but not a compiler. This greatly facilitates testing.

5. A Project Team 3000 Miles Apart

Almost all of Echelon was developed by two engineer/ programmers: one in Ontario, and one in California. (A second California engineer joined the project much later.) Face-to-face meetings, lasting about one week, were necessary every two months. Between meetings, fax, modem, and telephone sufficed.

The division of tasks made this work well.

At first it was planned for everyone to be competent to maintain all four programs -- thus, the "resident" engineer could take over from the contractor. Modules from each program were assigned to both programmers, to familiarize them with all the code. This was a dismal failure, and development languished.

Progress accelerated when complete responsibility for each program was assigned to a single programmer. The CP and PP were developed in California; the XP and VP in Ontario. This reduced the need for communication, and simplified source code management.

Only the interfaces between programs required joint effort. The most effective technique seemed to be bargaining: "I'll code this function in the XP, if you'll code that function in the CP." This optimized the use of both human and processor resources.

6. Source Code Management

With a single programmer controlling each program, limiting edit access to software modules is not a problem.

Version control is more important. In the first year, the software has gone through twelve versions (mostly in response to marketing requests). At present, there is no version control software; a simple archive of each version is saved. Only the latest version is "active." This would be better managed by Baden's FCCS or a similar source code control system, which can accommodate divergent versions.

7. Project schedule

February 1990: T-Recursive Technology submits proposal
March 1990: Specification approved; work begins
April 1990: Hardware design complete; software begins
September 1990: Prototype hardware completed
November 1990: Conversion to Forth
March 1991: First (partial) software demonstrations
August 1991: First product deliveries, version 1.00
April 1992: Full software release, version 1.11

H. LESSONS LEARNED (CONCLUSIONS)

1. Right Decisions

Forth was the right decision for fast development. Compare Echelon with a similar console developed by a competitor at the same time: Echelon had one full-time and one half-time engineer/programmer. The competitor had five full-time programmers. Both consoles required about the same time to complete. Forth gave a 3:1 productivity gain! (This gain increases if Echelon's more complex architecture, the amount of code written in assembler, and the time spent designing hardware, are factored in.)

A small project team was a good decision. Understandings are clearer, and meetings more productive, between two programmers than among five.

Dividing the work by CPU, and designing the interfaces by negotiation, optimized programmer effort.

Finally, the product specification was frozen early. The final product closely matches the original concept.

2. Wrong Decisions

Documenting the software at the end of the project was a wrong decision. Much of the thought process is lost when the code is documented after the fact. And it is difficult to bring aboard new programmers, who are not privy to the decisions of the original team. Documentation should be concurrent.

Twelve software releases in one year was a wrong decision. Fortunately, the releases do not diverge, and only the latest needs to be maintained. But much effort was spent on "throwaway" code. (The solution is called "shoot the sales force.")

3. Things To Do Better

Forth is not being used to the fullest. In particular, there is not a sufficiently high level of abstraction, in both program function and data organization.

As one example: the interprocessor communication functions could be much more abstract and flexible. At present, the bus mastership is visible to the programmer, and the control of data transfer is clumsy (requiring "magic numbers").

Forth should have been used from the outset. Much VP code was developed in assembler, then translated to Forth CODE words. There would have been higher productivity in developing the CP and PP programs, and they could have been more complex and powerful. Converting entirely to Forth would now be costly.

I. BIBLIOGRAPHY

[RAT85] Rather, E. D., "Fifteen Programmers, 400 Computers, 36,000 Sensors, and FORTH," Proceedings of the 1985 Rochester Forth Conference, published in The Journal of Forth Application and Research 3:2 (1985), pp. 47-73.

[ROD89] Rodriguez, B. J., "A Multiprocessor Forth Kernel," Forth Dimensions XI:3 (Sep-Oct 1989), pp. 14-22.