PROGRAM NOTES

SOFTWARE NEWS

Where's the hardware?

In the early days of computers, computer operations were hardwired—a particular pattern of on and off bits in an instruction turned particular circuits on or off, so that data could be shifted from one place to another, two numbers could be added together, or control could be transferred to another portion of the program. As long as instructions were simple, this approach worked well. Digital Equipment Corp.'s PDP-8 minicomputer, for example, had a total of eight different kinds of instruction, each with variations for addressing mode and other details. But as computers' central processing units begin to execute more complex instructions, the hardwired approach becomes more and more difficult to implement. DEC's VAX-11 computers, for example, have over 300 "basic" instructions, each with any number of addressing modes; building circuitry to implement those instructions directly would be impossible.

The VAX and the vast bulk of modern computers use an alternative to hardwired instruction sets known as microcode—low-level instructions that access the actual registers, buses, and arithmetic-and-logic units of a processor. Each machine instruction corresponds to a sequence of microinstructions, so that adding the contents of two memory locations, for example, might require microinstructions to fetch the first memory location and place it in the ALU, set the ALU to perform addition, fetch the second memory location and send it to the ALU, perform the addition, and then send the sum to its destination. Some of the operations controlled by microcode, such as fetches and additions, can occur simultaneously. Because they control so many simultaneous actions, microcode instructions generally have many more bits than regular machine instructions. However, this potential concurrency also makes microcode much more difficult to write, since the programmer must keep track of myriad details and since many code combinations lead to impossible results, such as loading a value into several registers simultaneously.

In addition to making complex-instruction-set computers possible to design and build, microcoding also makes it possible for any number of computers with disparate architectures to execute the same instruction set: microcode sequences for any given instruction match what the instruction does to the hardware that is actually available. If the memory holding the microcode can be altered, then a computer's instruction set can be changed by changing the microcode, or its functionality improved as microprogrammers wring out additional performance. Microcoding has allowed IBM Corp. to build a series of computers (the 370 architecture) in which the latest model, the 308X, looks essentially the same to a program as the original 370 built in the late 1960s.

One typical organization for a microcoded machine involves an instruction decoder that takes an incoming instruction and produces the starting address of the microcode sequence that executes it. As microcoding itself becomes more complex, a number of instructions may share some of the same microcode sequence. All instructions that reference a stack, for example, would require the same microcode sequence for fetching the stack pointer and incrementing or decrementing it appropriately. To save space, the instruction decoder can be modified so that it produces the starting address of a list of microroutines, each of which is executed in sequence. This approach is called either two-level microding or nanocoding. Given a complex enough instruction set, it might be possible to imagine a situation requiring yet another level of indirection (picocoding), but the performance cost of such an approach could be considerable.

Another approach is taken by advocates of the reduced-instruction-set computer, which executes only a few relatively simple instructions, but does so very quickly. The reasoning behind this approach is that more complex instruction sets slow down the vast bulk of simple instructions in the process of supporting seldom-used complex instructions. Furthermore, by allowing the compiler direct access to the most fundamental operations of a CPU, a reduced-instruction-set computer may let high-level languages be implemented more efficiently than if they had to use a fixed set of microcoded routines.

Squeezing information

Although the cost of computer memory drops constantly, most programmers still have to minimize the amount of memory used by programs and data. One of the typical methods of saving space is to eliminate unecessary information: for example, some programs store character strings in fixed-length arrays padded with blanks, while others indicate the number of characters at the beginning of each string. Encoding can further improve efficiency; the most common technique is run-length coding, in which a sequence of the same character is represented by a repeat code followed by a count and the character involved. Common multiple-character sequences can also be encoded.

Statistical measures of the predictability of text yield a measure known as entropy per character, which determines the minimum number of bits needed to encode a single character; achieving this minimum is generally possible only through so-called Huffman codes, which use variable numbers of bits for different characters: only a few bits for common characters, and longer sequences for characters used less often. Even more compression can sometimes be attained by making a compressed list of all the words in a document and replacing the text with a list of pointers to words within the list.

Compressing a text file to a fraction of its former length is not without its tradeoffs. The time required for compression and decompression is one major factor, and the format of compressed data is another. Computer systems store information in fixed-length chunks, so that variable-length codes can be difficult to implement. Thus, the amount of compression achievable is usually limited by algorithmic difficulties rather than by the statistical properties of the text. However, the statistical properties of the text being compressed can be extremely significant under some circumstances. Any compression technique that uses more than 1 byte to encode characters or character combinations may end up inflating rather than compressing files whose statistics are different from those on which the compression algorithm is based.

Sending digital by analog

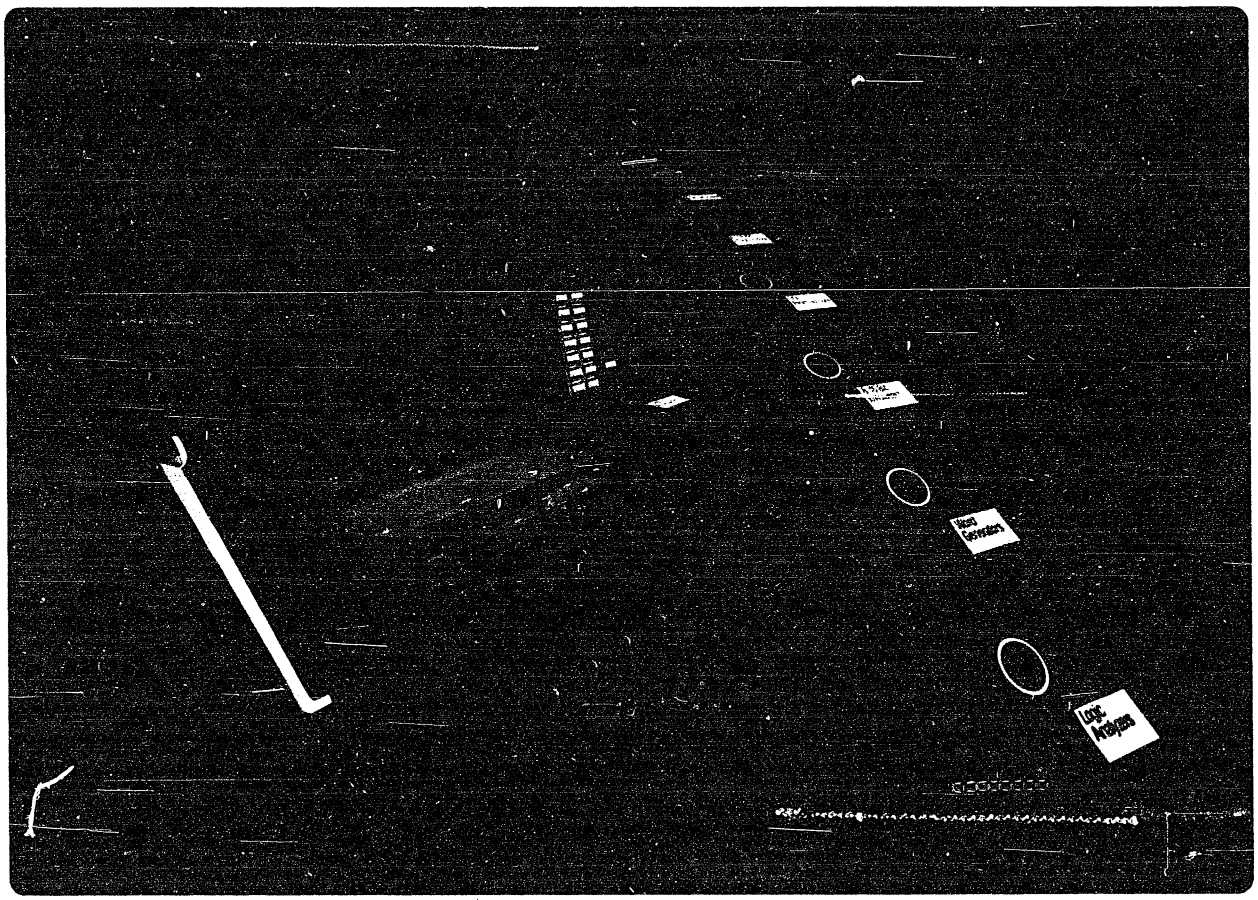
Sending data over most telecommunications lines requires that they first be converted from digital into analog form, because, though on and off pulses may be useful for ringing telephones, they are not suitable for transmitting data. (The conversion of telephone transmission lines into digital form will only help the computer with a direct link to the digital portion of the circuit; otherwise, the digital signal must be converted to analog, digitized for transmission, and converted back into analog so that a modem can recover the digital signal at the other end.) One problem a pulse-based link has is knowing when the machine on the other end has hung up, since a person at a terminal can send no data (as opposed to zeros) for an arbitrary period of time. Instead, particular frequencies are used to stand for ones and zeros in telephone-based communication. A modem using two tones can transmit information as fast as the two tones can reliably be distinguished at the other end; the most severe constraint is that telephone lines only have a bandwidth of approximately 3000 cycles per second, so that a tone sequence that changes faster than that rate will not be properly transmitted. In fact, tone sequences must generally allow for several periods of a waveform to pass so that its frequency can be identified.

The limited bandwidth of the telephone lines, coupled with the generous timing requirements of modem hardware, would seem to limit telephone transmission rates to a few hundred bits per second—300 baud, for example. High speeds are generally achieved by abandoning the simple on-off encoding of digital data and adopting multilevel schemes that transmit more than 1 bit at a time: phase-shift keying, for example, uses four possible phase shifts to encode all the possible combinations of 2 bits in a signal-tone sequence. Finer-grained codes can compress even more bits into a single tone, but since the number of distinguishable signals needed grows exponentially with the number of bits encoded, multiple-bit encoding quickly becomes unfeasible.

Dolch.

Compact-Logic-Tester

COLT 300



More Bang For Your Bug

Plug-Ins give you the ultimate in flexibility
Dolch's COLT/ATLAS concept is the most logical step into the future of logic analysis and digital test instrumentation. The integration of a general purpose computer with an instrumentation Plug-In creates the ultimate virtual instrument, providing flexibility for today's debugging needs and expandability for tomorrow's logic testing.

Plug-Ins expand your logic analyzer power
As a pioneer in the field of logic analyzers Dolch provides you with the most complete range of logic analyzer plug-in modules for state (48 channels) or timing (300MHz). μP-dedicated analyzers provide you with the ultimate logic analyzer power, carefully tailored to fit the needs for all popular 8 or 16 bitters.

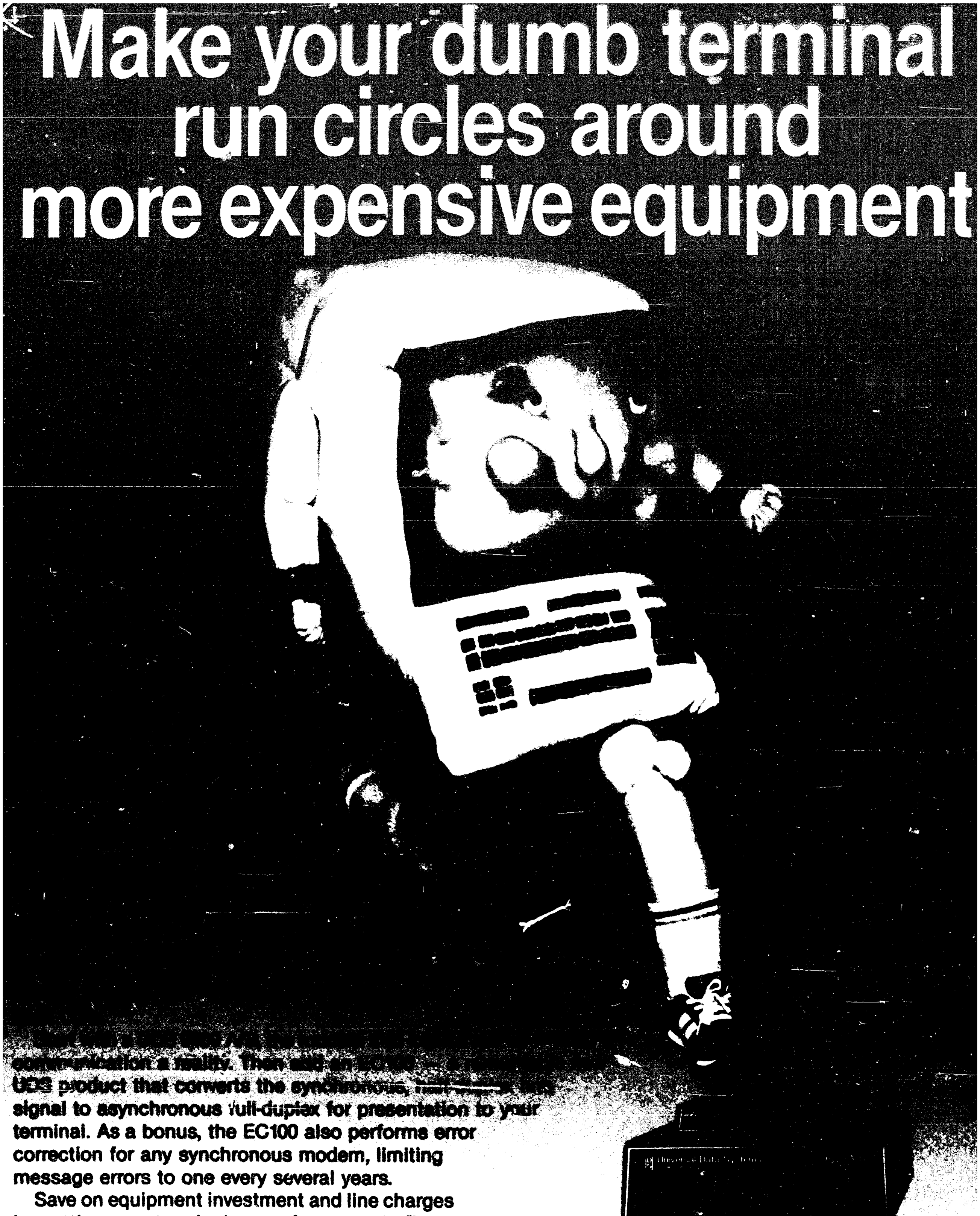
Load up your COLT today and shoot down system bugs
with Plug-Ins now available for:
- Logic Analysis
- In-Circuit Emulation
- Word Generation
- Serial Data Analysis
- In-Circuit Testing
- PAL/IFL Development
- EPROM Programming

For more Information:

Circle No. 12

Write or call for a demonstration today
2029 O'Toole Avenue, San Jose, CA 95131
(800)538-7506 (408)945-1881 (in California)

DOLCH
LOGIC INSTRUMENTS



[illegible] communication a reality. Then add an [illegible] — a [illegible] UDS product that converts the synchronous, [illegible] signal to asynchronous full-duplex for presentation to your terminal. As a bonus, the EC100 also performs error correction for any synchronous modem, limiting message errors to one every several years.

Save on equipment investment and line charges by putting your terminals on a faster track. **For details, phone 800/633-2252, ext. 356.** Universal Data Systems, 5000 Bradford Drive, Huntsville, AL 35805. Telephone 205/837-8100; TWX 810-726-2100.

Universal Data Systems

MOTOROLA INC.
Information Systems Group

UDS modems are offered nationally by leading distributors. Call the nearest UDS office for distributor listings in your area.
DISTRICT OFFICES: **Atlanta**, GA, 404/998-2715 • **Bellevue**, WA 206/455-4429 • **Blue Bell**, PA 215/643-2336 • **Boston**, MA, 617/875-8868 • **Columbus**, OH, 614/895-3025 • **Englewood**, CO, 303/694-6043 • **Glenview**, IL, 312/998-8180 • **Houston**, TX, 713/988-5506 • **Huntsville**, AL, 205/837-8100 • **Mountain View**, CA, 415/969-3323 • **Old Bridge**, NJ, 201/251-9090 • **Richardson**, TX, 214/680-0002 • **Silver Spring**, MD, 301/587-0166 • **Tampa**, FL, 813/684-0615 • **Tustin**, CA, 714/669-8001

Circle No. 10

Created by Dayner/Hall, Inc., Winter Park, FL